arXiv:2508.19898v2 [cs.DS] 26 Jun 2026

Distributed Sparsest Cut via Eigenvalue Estimation*

Yannic Maus ⓡ† Tijn de Vos

Abstract

We give new, improved bounds for approximating the sparsest cut value or in other words the conductance ϕ of a graph in the CONGEST model. As our main result, we present an algorithm running in $O(\log^2 n/\phi)$ rounds in which every vertex outputs a value $\tilde{\phi}$ satisfying $\phi \leq \tilde{\phi} \leq \sqrt{2.01\phi}$. In most regimes, our algorithm improves significantly over the previously fastest algorithm for the problem [Chen, Meierhans, Probst Gutenberg, Saranurak; SODA 25]. Additionally, our result generalizes to k-way conductance.

We obtain these results, by approximating the eigenvalues of the normalized Laplacian matrix $L := I - \mathrm{Deg}^{-1/2} A \mathrm{Deg}^{-1/2}$, where, A is the adjacency matrix and Deg is the diagonal matrix with the weighted degrees on the diagonal. We show our algorithms are near-optimal by proving a lower bound for computing the smallest non-trivial eigenvalue of L, even in the stronger LOCAL model

The previous state of the art sparsest cut algorithm is in the technical realm of expander decompositions. Our algorithms, on the other hand, are relatively simple and easy to implement. At the core, they rely on the well-known *power method*, which comes down to repeatedly multiplying the Laplacian with a vector. This operation can be performed in a single round in the CONGEST model.

All our algorithms apply to weighted, undirected graphs. Our lower bounds apply even in unweighted graphs.

This is a low-co2 research paper: `https://tcs4f.org/low-co2-v1`. This research was developed, written, submitted and presented without the use of air travel.

Acknowledgements We would like to thank Leo Wennmann, for the many hours of discussion on this topic. We would like Joachim Orthaber and Malte Baumecker, for thinking along. We would like to thank Faith Ellen, Sebastian Brandt, Alexandre Nolin, and Eva Rotenberg, for the initial discussions that were the inspiration for this research. We would like to thank Schloss Dagstuhl, since the research was initiated during the Dagstuhl seminar *Graph Algorithms: Distributed Meets Dynamic*.

*This research was funded in whole or in part by the Austrian Science Fund (FWF) `https://doi.org/10.55776/P36280` and `https://doi.org/10.55776/I6915`. For open access purposes, the author has applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission.

†The author ordering was randomized using `https://www.aeaweb.org/journals/policies/random-author-order/` generator. It is requested that citations of this work list the authors separated by `\textcircled{r}` instead of commas.

Contents

1 Introduction **1**
- 1.1 Our Contribution on Sparsest Cut Approximation . . . 2
- 1.2 Our Contributions on Spectrum Approximation . . . 3
- 1.3 Related Work . . . 5
- 1.4 Technical Overview . . . 7
- 1.5 Outline . . . 10

2 Approximating Eigenvalues **11**
- 2.1 The Graph Laplacian Matrix. . . . 11
- 2.2 Power Method . . . 11
- 2.3 Warm-up: The Largest Eigenvalue . . . 12
- 2.4 The Smallest Non-Zero Eigenvalue . . . 12
- 2.5 The k Smallest Eigenvalues . . . 15
  - 2.5.1 M_{approx} approximates M_{ideal}. . . . 16
  - 2.5.2 Approximating the k-th Smallest Eigenvalue in CONGEST . . . 18
- 2.6 Lower Bounds For Eigenvalue Approximation . . . 18
- 2.7 Approximating the Entire Spectrum . . . 19

3 Approximating the Sparsest Cut **20**
- 3.1 CONGEST Upper Bound for Approximating the Sparsest Cut . . . 20
- 3.2 CONGEST Upper Bound for Approximating the k-Way Sparsest Cut . . . 21
- 3.3 Lower Bounds . . . 21

References **23**

A The Power Method **31**
- A.1 The Power Method (Proof of Lemma 7) . . . 31
- A.2 Bit Precision Lemma (Proof of Lemma 11) . . . 34

1 Introduction

A graph cut is a partition of the vertices into two disjoint subsets, where the size of the cut is the total number of edges crossing between them, and the sparsity of the cut measures how small this cut is relative to total edge weight of the smaller side, i.e., in comparison to the sum of its vertex degrees. Graph cuts are a central object in theoretical computer science and algorithm design and have been extensively studied in various settings and models of computation, both explicitly studied, e.g., in [Kar00; ARV09; SGP09; SMPU15; KM15; DEMN21; CKGS23], and as central subroutines, e.g., in [ST04; SW19; CMGS25].

Cuts are especially important in communication networks because they represent the capacity for information to flow between different parts of the system. A sparse cut—where only a few edges connect two large subsets—can become a bottleneck, severely limiting bandwidth or throughput. A central question for guaranteeing the performance of network algorithms is to determine the minimum sparsity of cuts in a given communication network. For example the runtime of the famous PUSH-PULL protocol depends on the sparsest cut value [Gia11]. More generally, graphs without sparse cuts support fast mixing of random walks [SNPT13], and enable efficient routing due to their strong connectivity [GKS17; GL18]. This motivates the use of expander decompositions, which partition a graph into *high-conductance* clusters (i.e, no sparse internal cuts), and sparse inter-cluster connections. Such decompositions are powerful algorithmic tools: thanks to efficient routing, many algorithms run significantly faster on high-conductance subgraphs, allowing problems to be solved locally within clusters before handling the sparse inter-cluster parts. This strategy has been used in many applications, see, e.g., [CPZ19; CS19; DHNS19; EFFK+22; CGL20]. The distributed expander decompositions [CPZ19; CS19; CS20; CMGS25] are all constructed by repeatedly computing sparsest cuts.

In this work, we present simple, time and bandwidth efficient distributed algorithms to approximate the sparsest cut value. Our randomized algorithms run in the classic CONGEST model of distributed computing where a communication network is abstracted as an n-vertex graph with vertices being computing entities and edges serving as bandwidth-limited communication links. Communication happens in synchronous rounds, in each of which each vertex can send a $O(\log n)$-bit message to each of its neighbors and the complexity measure is the number of communication rounds until the vertices have computed their outputs, e.g., until they have output an approximate value of the sparsest cut.

Formally, the *sparsity* of a set (or cut) $S \subseteq V$ in a graph $G = (V, E)$ is defined as as

$$\phi(S) := \frac{w(E(S, V \setminus S))}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(V \setminus S)\}} ,$$

where $w(T) := \sum_{e \in T} w(e)$ for any edge set T and the volume $\mathrm{Vol}(S)$ of a set of vertices S is the sum of the weighted degrees of the vertices in the set: $\sum_{v \in S} \sum_{uv \in E} w(uv)$. The value $\phi(S)$ is also known as the *conductance* of the set S. The *sparsest cut* $\phi(G) \in [0, 1]$ of a graph G is the minimum sparsity of a subset $S \subseteq V$. This is also known as the *conductance* of the graph.

The relative notion $\phi(G)$ can be viewed as a normalized quantification of cut size. In contrast to computing minimum cuts, which quantify the absolute size of a cut, determining the sparsest cut or the conductance is not only NP-complete [MS90] but also APX-hard under the unique games conjecture [CKKR+06]. Consequently, although no polynomial-time algorithm can approximate the optimum arbitrarily well (unless $P = NP$), a variety of approximation algorithms with provable—albeit weaker—guarantees have been developed across different computational models. Next, we present our results and compare them with the state of the art in the field. As the sparsest cut and the conductance have been extensively studied we provide further background and related work in Section 1.3.

1.1 Our Contribution on Sparsest Cut Approximation

Our first result is an efficient CONGEST algorithm to approximate the sparsest cut.

Theorem 1. *There is a randomized* CONGEST *algorithm that, given an undirected weighted n-node graph with sparsest cut ϕ, w.h.p. gives an approximation of the sparsest cut in $O(\log^2 n/\phi)$ rounds. In particular, every vertex outputs $\tilde{\phi}$ such that $\phi \leq \tilde{\phi} \leq \sqrt{2.01\phi}$.*

In general, there are two approaches for developing approximation algorithms for the sparsest cut, the *flow approach*, that we detail in Section 1.3, and the *spectral approach*, which we use to prove Theorem 1. On a high-level, one computes the spectrum (or parts of it) of the graph, i.e., its eigenvalues, and then uses Cheeger's Inequality[1] to derive an approximation of $\phi(G)$ from the eigenvalues. Due to the limitations imposed by Cheeger's Inequality, the best possible approximation achievable through spectral methods is $\sqrt{2\phi}$. Our technical result is slightly stronger than Theorem 1. In fact, we can attain this theoretical bound – up to an arbitrarily small additive error – in $O(\log^2 n/\varepsilon^2 + \log n/\varepsilon^4)$ rounds. See the proof of Theorem 1 for the details.

For unweighted graphs, we compare our result against the previously best algorithm by Chen, Meierhans, Probst Gutenberg, and Saranurak that provides a $O(\phi \log^2 n)$-approximation of ϕ [CMGS25][2] Our approximation is more precise than [CMGS25] when $\phi = \Omega(1/\log^4 n)$, and is faster in all cases: [CMGS25] takes $O(\operatorname{poly}\log n/\phi^4)$ rounds. Upon inspection of their $\operatorname{poly}\log n$, they have a total round complexity of at least $O(\log^{20} n)$ in the regime where their approximation is more precise.

Prior to [CMGS25], Das Sarma, Molla, and Pandurangan [SMP15] gave a different algorithm to approximate the sparsest cut, which also works for weighted graphs. They computed an $O(\sqrt{\phi}\operatorname{poly}\log n)$-approximation in $O(\frac{1}{b}(n + \frac{1}{\phi})\log^2 n)$ rounds if the graph has a sparse cut of *balance* b, i.e., both sides S of the cut satisfy $\operatorname{Vol}(S) \geq 2b|E|$. Building on the methods of [SMP15], Kuhn and Molla improved the approximation ratio to $O(\sqrt{\phi \log n})$ and the runtime to $O(D + \frac{\log^2 n}{b\phi})$ [KM15]. For balanced cuts $b = \Omega(1)$, we improve their approximation ratio by a factor $\sqrt{\log n}$, while matching their running time. For unbalanced cuts, also their algorithm can take as much as $\Omega(n^2)$ time, so we do not only improve the approximation ratio, but also the running time exponentially.

We note that [CMGS25; SMP15] and [KM15] do not only provide an approximate cut value, but also give a corresponding cut.

k-Way Sparsest Cut. There exists a natural generalization of the sparsest cut to *multi-cuts* [LGT12]: the *k-way sparsest cut* or *k-way conductance* ϕ_k is defined follows:

$$\phi_k := \min_{V_1,V_2,\ldots,V_k \subset V} \max_{i=1,\ldots,k} \frac{|E(V_i, V \setminus V_i|}{\operatorname{Vol}(V_i)},$$

where the minimum is taken over disjoint, non-empty V_i. Note that $\phi_2 = \phi$. The k-way conductance captures how well-connected a graph is across multiple parts; it can describe multi-partition bottlenecks, which standard conductance may miss. This is in particular relevant for real-world networks that usually consists of multiple clusters.

We prove the following theorem for approximating the k-way sparsest cut.

Theorem 2. *There is a randomized* CONGEST *algorithm that, given a constant $k \geq 2$ and undirected weighted graph with k-way sparsest cut ϕ_k, w.h.p. gives an approximation of the k-way sparsest cut in $O(\log^2 n \operatorname{poly}(\phi_k^{-1}))$ rounds. In particular, every vertex outputs $\tilde{\phi}_k$ such that $\phi_k \leq \tilde{\phi}_k \leq O(\sqrt{\phi_k})$.*

[1] Cheeger's Inequality [AM85; Alo86; SJ89] relates the second-smallest eigenvalue λ_2 of the normalized Laplacian to the graph's conductance ϕ, showing that $\frac{1}{2}\lambda_2 \leq \phi \leq \sqrt{2\lambda_2}$. See Theorem 8 for the formal statement.

[2] They do not state this result explicitly, but they implement the cut-matching game from [SW19] (see Appendix A) in the distributed setting. As stated in [SW19], this cut-matching game gives such an approximation.

To the best of our knowledge, we are the first to study the k-way conductance in the distributed setting.

Lower bounds on approximating sparse cuts. Generally, it is known that approximating the sparsest cut takes $\Omega(D)$ rounds. In particular, Das Sarma, Molla, and Pandurangan [SMP15] show a $\tilde{\Omega}(D + \sqrt{n})$ CONGEST lower bound for any multiplicative approximation in weighted graphs. They reduce the spanning connected subgraph problem to a sparsest cut instance with edge weights that are either 0 or 1. This means that the lower bound shows that it is hard to decide whether $\phi = 0$ holds or not. For ϕ bounded away from zero, there is no such lower bound. Our upper bound of $O(\log^2 n/\phi)$ rounds, Theorem 1, indeed takes D rounds, since $D = O(\log n/\phi)$. However, for $\phi = \omega(1/\sqrt{n})$, the upper bound does not have a $\sqrt{n}$-term.

We prove a $\Omega(D)$ lower bound that holds even for unweighted graphs and in the LOCAL model. LOCAL is identical to CONGEST, except that there is no bound on the message size. Even for a class of fixed diameter, any algorithm needs $\Omega(D)$ time for a $O(n)$-approximation.

Theorem 3. *Let $D \leq n/2$ be a parameter and consider the class of graphs with diameter D. An algorithm that with probability $> \frac{1}{2}$ satisfies that every vertex v of a graph G outputs a value $\tilde{\phi}_v \geq \phi(G)$, for sparsest cut $\phi(G)$, and at least one vertex has $\phi_v \leq \Theta(n \cdot \phi)$ needs $\Omega(D)$ rounds in* LOCAL.

Note that the $\Omega(D)$ lower bound in Theorem 3 holds even in the case where only a single vertex needs to witness the cut.

The above proof is about discerning graphs with sparsity $1/n$ and $1/n^2$. If *every vertex* needs to output a correct estimation, a diameter lower bound exists for larger values of ϕ.

Theorem 4. *Let $0 < \varepsilon < 1$ be a constant and let $\phi > 1/n^{1-\varepsilon}$. An algorithm that with probability $> \frac{1}{2}$ satisfies that every vertex v outputs a value $\tilde{\phi}_v$ such that $\phi \leq \tilde{\phi} \leq O(n^{1+\varepsilon} \cdot \phi)$, where ϕ is the sparsest cut value of the graph needs at least $\Omega(D)$ rounds in the* LOCAL *model.*

Our CONGEST upper bound for approximating the sparsest cut w.h.p. is $O(\log^2 n/\phi)$ – only a factor $\log n$ away from optimal, since the graphs from the above observation have sparsest cut ϕ and diameter $\Theta(\log n/\phi)$ [CGLP18]. We note that with constant probability, we match the lower bound, see Section 1.4.

1.2 Our Contributions on Spectrum Approximation

Our core technical contributions are efficient CONGEST algorithms to compute the approximate (partial) spectrum of a graph. In essence, Theorems 1 and 2 then follow as direct consequences via Cheeger's Inequality. Nevertheless, we chose to present these theorems first, as they are more accessible to a broader audience and can be compared with prior work.

Eigenvalues of the Normalized Laplacian. The normalized Laplacian of a graph is the matrix defined as $L := I - \mathrm{Deg}^{-1/2} A \mathrm{Deg}^{-1/2}$, where A is the adjacency matrix and Deg is the (weighted) degree matrix. See Section 2.1 for more details. Its spectrum captures many central properties of the graph, e.g., various types of its connectedness like k-way conductance [LGT12] and small set expansion [ABS15], or how close it is to being bipartite [BJ09; Liu15]. We provide the following theorem to compute an approximation of that spectrum.

Theorem 5 (Eigenvalue estimation)**.** *Let $\varepsilon > 0$ be a parameter and $k \in \mathbb{N}$ be a constant. There are randomized* CONGEST *algorithms that, given an undirected, weighted graph $G = (V, E)$, with high probability approximate the eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ of the graph's normalized Laplacian as follows:*

1. *$\lambda_1, \ldots, \lambda_k$: an additive ε-approximations in $O(\log^2 n \cdot \mathrm{poly}(\varepsilon^{-1}))$ rounds,*

2. λ_2: *an additive* ε*-approximation in* $O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon} \cdot \left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil\right) = O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon^2}\right)$ *rounds,*

3. λ_n: *a multiplicative* $(1 \pm \varepsilon)$*-approximation in* $O\left(D + \frac{\log^2 n}{\varepsilon}\right)$ *rounds.*

Eigenvalue λ_2 is the eigenvalue approximating the sparsest cut via Cheeger's Inequality. It could either be approximated via the second algorithm, or slightly slower via the first.

For the smaller eigenvalues, Theorem 5 states additive approximations avoiding the diameter dependence, e.g., when ε is an arbitrarily small constant Theorem 5 approximates λ_2 in $O(\log^2 n)$ rounds even though the diameter may be up to linear in n.

As ε can be arbitrary in Theorem 5, we can obtain multiplicative approximations by setting $\varepsilon = \varepsilon'\lambda_2$ for any $\varepsilon' > 0$. However, this does not circumvent the diameter lower bound of Theorem 6 presented below. By Theorem 8 and Lemma 21, $D = O(\log n/\lambda_2)$ holds for any graph, so the diameter term is then always asymptotically smaller than the overall runtime of the algorithm.

The main prior work for distributed eigenvalue approximation is by Kempe and McSherry [KM08]. They provide a distributed algorithm that, for constant k, approximates the k-largest/smallest eigenvalues in $O(\tau_{\text{mix}} \log^2 n)$ rounds. We note that the mixing time can be as large as $n^{\Omega(1)}$. This happens when λ_2 is really small (in which case our algorithms are also slow), but also when the graph is (close to) bipartite. In the latter case, our algorithms can be exponentially faster. Furthermore, an important caveat here is that Kempe and McSherry do not obtain their result in the CONGEST model: they do not discuss the bandwidth constraint. They do specify that they only need poly(k) messaged per edge per round. However, they do not take the *size* of these messages into account. In our work, this is a non-trivial contribution, as intermediate values may grow fast. We think similar techniques might be able to work for their approach as well, but this requires non-trivial adjustments and additional analysis.

Later, Becchetti et al. [BCNP+20] gave an algorithm for finding *balanced* cuts, where our algorithm works for *all* cuts. We provide more details on this algorithm in Section 1.4.

An alternative approach to distributed eigenvalue computation is via computing a spectral sparsifier, i.e., a sparse subgraph H with $\tilde{O}(n/\varepsilon^2)$ edges that approximately preserve the graph's spectrum. Next, the sparsifier H can be gathered at one vertex of the graph that can locally compute the eigenvalues of H and hence obtain approximations of all eigenvalues of G. Computing H is actually relatively fast and only requires poly$(\log n, \varepsilon^{-1})$ rounds [KX16]. Collecting H at a single vertex is the expensive part, requiring $\tilde{O}(n/\varepsilon^2)$ rounds. This is exponentially slower than Theorem 5, though this more general method immediately computes the full spectrum while we focus on single eigenvalues.

We note that all our upper bounds even hold in the more restricted Broadcast CONGEST model in which a vertex needs to send the same message to all of its neighbors in each round. All our algorithms become a $\log n$-factor faster if we only ask for constant probability as typical in property testing, see the discussion in Section 1.4.

Lower Bound. All of our multiplicative upper bounds take diameter time. This means that at the end of the algorithm, every vertex can easily output the same eigenvalue or sparsest cut estimate. One may ask whether one can obtain faster algorithms if the output is more local. Perhaps every vertex outputs the sparsest cut it sees in some neighborhood, and there is a guarantee that at least one vertex will see an approximately sparsest cut. In Theorem 3, we showed that this is not true for sparsest cut approximation. We also show that approximating λ_2 at even one node requires $\Omega(D)$ rounds.

Theorem 6. *Any algorithm that with probability* $> \frac{1}{2}$ *ensures that all vertices output a value* $\tilde{\lambda}_2 \geq \lambda_2$*, and at least one vertex additionally satisfies* $\tilde{\lambda}_2 \leq 2\lambda_2$ *requires* $\Omega(D)$ *rounds.*

An $\Omega(D)$ lower bound could also follow from our sparsest cut lower bounds via Cheeger's Inequality, but such an approach incurs an inherent quadratic loss due to the approximation gap in Cheeger's.

Decision problem and connections to property testing. For the lower bounds in Theorem 3 (and similarly for Theorem 6), the vertices that do not output a correct value, still need to output a value $\tilde{\phi} \geq \phi$. If we do not do this, then a trivial algorithm where every vertex outputs a random number between 0 and 1 will be correct most of the time. In other words, we aim to solve the decision problem: "is there a cut sparser than ϕ?" If all vertices say "no", the answer is no. If at least one vertex answers "yes", the answer is yes. As we are only solving the approximate version of the question this is closely related to the concept of *property testing* in CONGEST, introduced in [CFSV19] and extensively studied since then, e.g, [EFFG+17; LMR18]. "Conductance property testing" has been explicitly studied by [FV18] and [BTT24] and Theorem 1 improves significantly on their results. We detail the relation to property testing and their results in Section 1.3.

Also the eigenvalues of the adjacency matrix and the non-normalized Laplacian describe many graph properties. Among other properties, they relate to the size of independent sets (Hoffman's bound) [Hae95], the chromatic number [Wil67], the average density of cuts [Hae95], the toughness of the graph [LC10], Hamiltonicity [FN10], the matching number [GL22], and the existence of a perfect matching [BH05].

Approximating the Entire Spectrum. Via similar methods, one can compute a multiplicative approximation of the entire spectrum. However, the output will not reveal how many eigenvalues there are of a certain approximation, i.e., it gives the spectrum *without multiplicity*. The running time of such an algorithm is $n^{o(1)}(\sqrt{n} + D)/\varepsilon^2$ rounds, see Section 2.7 for details. Note that in particular this gives a multiplicative approximation of the smallest non-zero eigenvalue λ_2. In case λ_2 is small – roughly $\lambda_2 = o(1/\sqrt{n})$ – this is faster than Theorem 5.

1.3 Related Work

Sparsest Cut. As mentioned there are two approaches to solving the sparsest cut problem. The *spectral approach* used in this paper and the *flow approach*. The spectral approach encompasses techniques like Cheeger's Inequality, spectral sparsifiers (discussed above), and also random walks. For instance, the results of [DS15; KM15; FV18; BTT24] rely on random walks, which are considered spectral in nature since their convergence analysis depends on spectral properties. This approach is inherently limited by Cheeger's Inequality, capping the approximation factor at $\sqrt{2\phi}$, the guarantee we match in this work.

The flow approach, based on the *cut-matching game*, computes a number of flows that will either guarantee expansion or find a sparse cut. Theoretically, this can give a $(\phi \operatorname{poly}\log n)$-approximation. In the sequential setting, the state-of-the-art is a $O(\phi\sqrt{\log n})$-approximation by Arora, Rao, and Vazirani [ARV09]. In the distributed setting, the state-of-the-art is the aforementioned $O(\phi \log^2 n)$-approximation [CMGS25]. Before this recent result, the flow based gave worse approximations: [CPZ19; CS19] gave a $\tilde{\phi} \leq O(\phi^{1/3}\log^{5/3} n)$-approximation in $O(D \cdot \operatorname{poly}\log n, \phi^{-1})$ rounds, for $\phi = O(1/\log^5 n)$. Additionally, Chang and Saranurak [CS20] gave a deterministic version of that result with additional sub-polynomial factor in the round complexity.

Property Testing. In our paper, we are essentially solving the decision problem: "does G have sparsest cut at least ϕ?" In property testing, the goal is to solve a relaxed variant of the corresponding decision problem: if the input network satisfies a property, then, with constant

probability, all the vertices accept but if the input network is ε-far[3] from satisfying the property, then at least one vertex rejects. Testing for sparsest cuts has explicitly been targeted in this setting by [FV18] and [BTT24]. For a fixed ϕ, [BTT24], improving on [FV18], solves this question as follows. All vertices output "yes", with probability at least 2/3 if the sparsest cut is at least ϕ, and at least one vertex outputs "no", with probability at least 2/3 if G is ε-far from having sparsest cut above $\phi^2/2880$. Similar to our result, the ϕ^2 term stems from Cheeger's Inequality. Our results improve upon this in several ways. For achieving a constant probability, we only need $O(\log n/\phi)$ rounds, as compared to $O(\frac{\log n}{\varepsilon\cdot\phi^2})$ rounds in their algorithm. Most notably, we do not solve the property testing version, and do not require that the instance is ε-far from having a sparse cut above $O(\phi^2)$. Thus, in some sense we are solving their version with $\varepsilon = 1/n^2$. Additionally, our results extend to weighted graphs and the constant in our approximation factor is significantly better. Lastly, we also approximately compute the sparsest cut value instead of only solving the decision problem.

Cut Sparsifiers. We have already discussed how to use spectral sparsifiers to approximate the spectrum in $\tilde{O}(n/\varepsilon^2)$ rounds [KX16]. They can also directly be used for approximating sparsest cuts without going through Cheeger's Inequality: $(1\pm\varepsilon)$-approximate spectral sparsifiers are $(1\pm\varepsilon)$-cut sparsifiers. This means that in $\tilde{O}(n/\varepsilon^2)$ rounds, we can collect a subgraph H in one vertex, in which *all* cuts are at most a factor $(1\pm\varepsilon)$ away from the cut in G. In particular, we can find a $(1\pm\varepsilon)$-approximation of the sparsest cut, and more generally a $(1\pm\varepsilon)$-approximation of the k-way sparsest cut in $\tilde{O}(n/\varepsilon^2)$ rounds.

Laplacian Paradigm. The *Laplacian paradigm* is an umbrella term for algorithmic techniques relying on the Laplacian and its properties. This line of research was initiated by Spielman and Teng [ST04], who showed that linear equations in the Laplacian matrix of a graph can be solved in near-linear time. Algorithms within the Laplacian paradigm combine numerical techniques – from the matrix point of view – with combinatorial techniques – from the graph point of view. Nowadays, it encompasses techniques like solving Laplacian systems, spectral sparsifiers, electrical flow, effective resistance, expander decompositions, continuous optimization, interior-point methods, gradient descent, and preconditioning. This has had many applications, including but not limited to flow problems [M13; She13; KLOS14; M16; Pen16; CMSV17; LS20a; LS20b; AMV20], bipartite matching [BLNP+20], and (parallel) shortest paths [Li20; ASZ20].

Also in the distributed world, this has booked many successes, e.g., [GKKL+18; BFKL21; FGLP+21; ALHZ+22; FV22; FV23; Vos23]. With this paper, we hope to contribute to a better understanding of the Laplacian matrix and its applications in a distributed setting.

Comparison to the Min-Cut Problem. In the minimum cut (min-cut) problem, the goal is to find a cut S of minimum value $|E(S, V\setminus S)|$. Although the problem statement seems similar to the sparsest cut problem, it has a very different computational complexity. As said, in the sequential setting, it is NP-hard to approximate the sparsest cut (under the unique games conjecture) [MS90; CKKR+06]. On the other hand, there are linear time algorithms for computing the exact min-cut [Kar00].

In the distributed setting, we show that the $\Omega(\sqrt{n}+D)$ lower bound does not apply for approximating the sparsest cut problem. The min-cut problem however has a complexity of $\tilde{\Theta}(\sqrt{n}+D)$ [SHKK+12; DEMN21], where the lower bound holds even for a poly(n)-approximation.

[3]This refers to εm changes to the graph. For a precise definition, see, e.g., [BTT24].

1.4 Technical Overview

As mentioned, Theorems 1 and 2 follow as direct consequences from our eigenvalue estimation results via Cheeger's Inequality. Here, we focus on our core technical contribution: efficient CONGEST algorithms to compute the approximate (partial) spectrum of a graph. The main ingredient of our algorithms is the well-known *power method* for approximating the largest eigenvalue of a matrix.

The Power Method Algorithm. Let M be any matrix. The power method, also known as *power iteration* or *Von Mises iteration*, starts with a random vector and converges to the eigenvector corresponding to the largest eigenvalue of M. It is a standard technique in numerical linear algebra, see, e.g., [GVL13]. The procedure is as follows.

Power Method

1: **Input:** Matrix $M \in \mathbb{R}^{n \times n}$, number of iterations k.
2: **Initialize:** A random vector $x_0 \in \mathbb{R}^n$.
3: **for** $i = 1$ to k **do**
4: $\quad x_i \leftarrow M x_{i-1}$.
5: **Output:**

- Approximate dominant eigenvalue: $\lambda \approx \frac{x_k^T M x_k}{x_k^T x_k}$;
- Approximate dominant eigenvector: x_k.

With constant probability over the randomness of the start vector x_0, the power method outputs a vector x_k whose *Rayleigh coefficient* $\frac{x_k^T M x_k}{x_k^T x_k}$ is close to the *largest* eigenvalue of the matrix M. The main intuition why this happens is as follows: Any vector can be written as a combination of the matrix' eigenvectors and when you apply the matrix multiple times, the contribution from the eigenvector with the largest eigenvalue (in magnitude) dominates. So, over time, the method "filters out" all directions except the one corresponding to the largest eigenvalue. This works only if the random start vector x_0 has a nonzero component in the dominant eigenvector's direction; beyond the choice of this initial vector, the process is fully deterministic.

In this paper, we only need the power method for *positive semi-definite (PSD)* matrices, which are symmetric matrices with non-negative eigenvalues. Then all eigenvalues are nonnegative and we can assume the eigenvectors to be orthonormal. The power method also converges slightly faster on PSD matrices – by a factor 2 to be precise.

Lemma 7 (Power Method)**.** *Let M be a positive semi-definite matrix with eigenvalues $0 \leq \mu_1 \leq \cdots \leq \mu_n$. Let $x_0 \sim \{-1, 1\}^n$ be a uniformly random vector and $x_k = M^k x_0$. For $\varepsilon \geq 16/n$ and $k \geq \Theta(\log n/\varepsilon)$, it holds with constant probability that*

$$(1 - \varepsilon)\mu_n \leq \frac{x_k^T M x_k}{x_k^T x_k} \leq \mu_n.$$

The power method is suitable for distributed applications, since matrix vector multiplication is easy in the CONGEST model: this can be done in $O(1)$ rounds where each vertex of the communication network holds an entry of the vector, and each vertex knows the entries of the matrix in its row and column. Important is that the only non-zero entries in a row or column correspond to a neighboring vertex. This is exactly the case when the matrix is closely related to the adjacency matrix – such as the Laplacian.

Only the final step, of computing the Rayleigh coefficient of x_k takes $O(D)$ time. We note that the power method is a widely used routine. Also decentralized implementations have

been known for a long time, e.g, [BT15; PS14] or for the special case of PageRank [SMPU15]. However, to the best of our knowledge, the procedure and its extensions have not been studied in the context of the CONGEST model.

One difficulty with implementing the power method in the CONGEST model as is, is that values grow exponentially with k and hence after a few iterations and they cannot be send to neighbors using $O(\log n)$ bits per message. Fortunately, the method is extremely robust. It still converges quickly after truncating each value after the $O(\log n)$ most significant bits; if one additionally remembers the order of magnitude of the values one can approximately compute the Rayleigh coefficient at the end of the algorithm – see Lemma 11 for a formal statement. More involved methods, like the Lanczos method (see, e.g., [GVL13]) might be able to return similar results. However, the Lanczos method is inherently unstable. It is unclear whether this can be adapted to work (efficiently) in the CONGEST model.

Averaging Dynamics. In community detection, average dynamics is a well-known concept. The idea is that each node updates its value to the average of its neighbors. If G is connected and not bipartite, this always converges. The update step is comparable is the same in spirit as a step in the power method, but not entirely equal. In particular, Becchetti et al. [BCNP+20] use average dynamics in the distributed setting. In case of a sufficiently small balanced cut, and some additional technical assumptions, they can provide an approximate sparsest cut. Since the approximation factor depends on the technical assumptions, we will not state their exact guarantees here. With a similar approach, Beccheretti et al. [BCNP+17] can find a sparse cut efficiently if both of the parts do not contain sparse cuts. Moreover, the average dynamics has been studied as a tool for community dynamics in related models that do not directly compare to the CONGEST model, e.g., the asynchronous, opportunistic communication model [BCMN+18] and the asynchronous gossip model [MMM18].

The Graph Laplacian Matrix. We denote $\mathbb{1}$ for the vector with 1 in each entry and $\sqrt{\deg} = \mathrm{Deg}^{1/2}\,\mathbb{1}$ for the vector with $(\sqrt{\deg})_v = \sqrt{\deg(v)}$ on index (equals vertex) v.

Recall that the normalized Laplacian $L := I - \mathrm{Deg}^{-1/2}\,A\,\mathrm{Deg}^{-1/2}$ has smallest eigenvalue $\lambda_1 = 0$. This can be seen since $\sqrt{\deg}$ is an eigenvector of L eigenvalue 0:

$$L\sqrt{\deg} = (I - \mathrm{Deg}^{-1/2}\,A\,\mathrm{Deg}^{-1/2})\sqrt{\deg} = \sqrt{\deg} - \mathrm{Deg}^{-1/2}\,A\mathbb{1},$$

Note that $A\mathbb{1}$ counts the number of 1s in each row[4], so this exactly sums up to the degree: $L\sqrt{\deg} = \sqrt{\deg} - \mathrm{Deg}^{-1/2}\deg = 0$. Hence, the smallest interesting eigenvalue to study is λ_2.

Approximating λ_2. To approximate the smallest non-zero eigenvalue of the Laplacian L, we consider the matrix $M := 2I - L$, which has eigenvalues $0 \leq \mu_1 \leq \cdots \leq \mu_n$, satisfying $\mu_i = 2 - \lambda_{n-i+1}$. In particular, this means that $\mu_n = 2$ and $\mu_{n-1} = \lambda_2$. In order to compute λ_2, one would like to subtract the components in the direction $v_n = \sqrt{\deg}$, the eigenvector corresponding to the largest eigenvalue μ_n. As a result, the prior largest eigenvalue is shifted to 0, and the power method converges to the second largest eigenvalue instead.

As per the discussion above, we know the largest eigenvector $v_n = \sqrt{\deg}$ exactly. We can also subtract the components in this direction exactly. However, a significant facet here is the bit complexity. As we discussed before, small errors incurred from rounding are not an issue when we are approximating the *largest* eigenvalue. However, due to rounding when we are approximation the second-largest eigenvalue, our vector will not remain perpendicular to v_n. Hence, given enough iterations, the process will always converge to the largest eigenvalue. Naively, we can combat this by projecting the current vector x on the subspace orthogonal to v_n in every step,

[4]In weighted graphs, this is the sum of the weights in each row. Since the degrees are also weighted degrees, the proof remains the same.

or in other words, by removing components in direction v_n. This requires computing the inner product between v_n and x. The latter is no problem in the centralized setting, but computing this inner product requires $\Omega(D)$ rounds in CONGEST. Hence we want to limit how often we do this. We show that the error blows up at a rate that means that we only need to project periodically, in total at most $\lceil \frac{\lambda_2}{\varepsilon} \rceil$ times. In order to perform this efficiently, we show that all interesting cases actually have a diameter small enough to repeatedly perform these projections without increasing the runtime too much.

Approximating $\lambda_1, \ldots, \lambda_k$. We compute the eigenvalue approximations sequentially. For approximating λ_i, we use the same general approach as for λ_2. Instead of projecting on the subspace orthogonal the largest eigenvalue, we project on the subspace orthogonal to the largest $i-1$ eigenvalues, and remove the resulting parts from consideration. The key challenge is that we do not know the corresponding eigenvectors exactly but they are only approximately known from our prior steps. We show that the power method still converges despite these inaccuracies in the eigenvectors that we project on. This is one of the most technical parts of this paper and appears in Section 2.5.

Probabilistic Guarantees. The power method as stated in Lemma 7 is correct with constant probability. This means that we can for example can also obtain our sparsest cut result, Theorem 1, in $O(\log n/\phi)$ rounds but with *constant* probability. To boost the probability, we run $O(\log n)$ independent instances of the power method and return the maximum. Since we are looking for the largest eigenvalue, and we know that we cannot overshoot (see Lemma 10), this gives the required approximation with high probability.

Lower Bound Constructions. We construct graphs by combining well-understood components—such as cliques, paths, and star graphs—where the sparsest cut is easy to determine. Using these building blocks, we define two graph families with significantly different sparsest cut values, yet with locally identical neighborhoods around certain vertices. As a result, these vertices cannot distinguish between the two families and thus cannot accurately approximate the sparsest cut without global information. This implies that approximating the sparsest cut requires $\Omega(D)$ rounds in our constructions.

It is challenging to use a similar approach to obtain a lower bound for eigenvalue approximation, as determining the eigenvalues of any graph is highly non-trivial. The eigenvalues of few specific graphs have been computed, like paths, cycles, and cliques, but determining them remains hard in general. This stems partly from the fact that gluing together graphs can have big effects on the eigenvalues. Cauchy's interlacing eigenvalue theorem [Hwa04] gives some bounds, but this is unfortunately of no help. Another approach is to use the fact that we know the sparsest cut, together with Cheeger's Inequality. Rephrased Cheeger's Inequality gives $\phi^2/2 \leq \lambda_2 \leq 2\phi$. So it gives a linear upper bound on the eigenvalue given the sparsest cut, but it has a square in the lower bound. At the end of the day, this means that only using Cheeger's would result in a lower bound of the form $\Omega(\sqrt{D})$. For our $\Omega(D)$ lower bound, we need a better upper bound on the eigenvalues of the graphs used in the lower bound construction. We exploit that eigenvalues can also be bounded by specific vectors. The simplest such statement is that the largest eigenvalue of a matrix M equals $\max_{x \in \mathbb{R}^n} \frac{x^T M x}{x^T x}$. Hence any x can provide a lower bound for the largest eigenvalue. We use a similar statement for the second smallest eigenvalue and construct an appropriate vector x to prove a tighter upper bound on the eigenvalues of our constructed graphs. In total, we are able to provide upper and lower bounds that characterize the eigenvalues of the graphs used in our lower bound construction up to a constant factor.

Sparsest Cuts and Cheeger's Inequality. Finally, let us explain how the eigenvalue results enable us to derive an approximation of the sparsest cut.

Let us obtain some intuition why eigenvalue λ_2 relates to the connectedness of the graph. In the extreme case that $\lambda_2 = 0$ holds the graph is disconnected: On each connected component, the degree vector is still an eigenvector with eigenvalue 0. Hence a vector that has $\sqrt{\deg}$ on one connected component, and all 0s on the other connected component is also an eigenvector – and linearly independent of $\sqrt{\deg}$. The reverse direction, i.e., that the graph is connected if $\lambda_2 \neq 0$ holds, is also true but its proof is slightly more involved, so we omit it in this overview, see e.g. [Chu97].

Cheeger's Inequality [AM85; Alo86; SJ89] gives a more precise statement about the connectedness of the graph and the second smallest eigenvalue: it relates λ_2 to the sparsest cut ϕ.

Theorem 8 (Cheeger's Inequality)**.** *Let $G = (V, E)$ be an undirected, weighted graph, then*

$$\frac{\lambda_2}{2} \leq \phi \leq \sqrt{2\lambda_2} \ .$$

The fact that $\lambda_2 = 0$ holds if and only if the graph is disconnected is a corollary of Theorem 8.

Since Cheeger's Inequality states that the value of λ_2 approximates ϕ, we can find an approximation $\tilde{\lambda}_2$ for λ_2 with Theorem 5 and use it to output an approximation $\tilde{\phi} = \sqrt{2\tilde{\lambda}_2}$ of the sparsest cut, establishing Theorem 1.

For larger values of k, similar statements hold. First, we remark that $\lambda_k = 0$ if and only if the graph consists of at least k connected components. One direction of this statement is again easy to see: as before, on each connected component the $\sqrt{\deg}$ vector is an eigenvector with eigenvalue 0. In total, this gives k independent eigenvalues with eigenvector 0 – in other words, $\lambda_k = 0$.

Similar as for $k = 2$, this generalizes by relating λ_k to ϕ_k: Lee, Gharan, and Trevisan showed the following theorem which we then use to establish Theorem 2 via Theorem 5.

Theorem 9 (Higher-Order Cheeger's Inequality, [LGT12])**.** *Let $G = (V, E)$ be an undirected, weighted graph, then $\frac{\lambda_k}{2} \leq \phi_k \leq O(k^2)\sqrt{\lambda_k}$.*

1.5 Outline

We first give our main results on eigenvalue estimation in Section 2, i.e., we prove Theorem 5. This sections starts with the formal description of the Laplacian matrix and the power method, and then proves Theorem 5 in reverse order: (3) in Section 2.3, (2) in Section 2.4, and (1) in Section 2.5. We conclude Section 2, with our lower bound on eigenvalue estimation, Theorem 6, and a description of how to use similar methods to approximate the entire spectrum. Then, in Section 3, we provide our upper and lower bounds on (k-way) sparest cuts. To be self-contained, we give a proof for the power method in Section A.

2 Approximating Eigenvalues

In Section 2.1, we begin with the necessary background and introducing notation. In Section 2.2, we give a formal description of the power method and state its most important properties. Then, we prove Theorem 5 part (2) and part (3), Theorems 12 and 13, proceeding in the reverse order: First, we give the algorithm for the largest eigenvalue λ_n in Section 2.3. Then, we give the algorithm for λ_2 in Section 2.4.

2.1 The Graph Laplacian Matrix.

First, we will give some definitions and notation. For a more thorough introduction of spectral graph theory, we refer to, e.g., [Chu97].

The normalized Laplacian is defined as $L := I - \mathrm{Deg}^{-1/2} A \mathrm{Deg}^{-1/2}$, where Deg is a diagonal matrix with the (weighted) vertex degrees on the diagonal, and A is the adjacency matrix. Let us inspect some of its properties. First of all, we note it is *symmetric*, i.e., $L^T = L$. We know that all symmetric matrices allow for a set of *orthonormal* eigenvector $v_1, \ldots, v_n$. That means the eigenvectors are *orthogonal*: $\langle v_i, v_j \rangle = 0$ for $i \neq j$, and have norm 1: $||v_i|| = 1$ for all i. Moreover, any Laplacian matrix is *positive semi-definite*. This means that all its eigenvalues are nonnegative. Equivalently, we have for any $x \in \mathbb{R}^n$ that $x^T L x \geq 0$. More precisely, it has eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2$. The eigenvalue 0 can easily be verified since $Lv_1 = 0$ for $v_1 := \sqrt{\deg}$, the vector with in each entry the square root of the degree of the corresponding vertex.

As explained in Section 1.4, $\lambda_2 = 0$ if and only if the graph is disconnected. More generally, $\lambda_k = 0$ if and only if the graph consists of k connected components. In the CONGEST model, we may assume a connected graph, or alternatively apply the results on each component of the communication network separately. Hence we have $\lambda_2 > 0$.

Notation. Throughout, $x_1, x_2, \ldots$ denote distinct vectors. We write $(x)_v$ for the v-th entry of a vector x, i.e., the entry of the vector x that is hold by vertex v.

We reserve $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2$ for the eigenvalues of the normalized Laplacian. For a general n by n matrix M, we denote its eigenvalues by $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$. We note that in the following M and L might differ, however, they will always have the same eigenvectors. Denote these as $v_1, \ldots, v_n$, corresponding to $\mu_1, \ldots, \mu_n$ and $\lambda_1, \ldots, \lambda_n$ respectively.

2.2 Power Method

The core of our eigenvalue approximation is an efficient CONGEST implementation of the power method. We recall its main measure of progress.

Lemma 7 (Power Method)**.** *Let M be a positive semi-definite matrix with eigenvalues $0 \leq \mu_1 \leq \cdots \leq \mu_n$. Let $x_0 \sim \{-1, 1\}^n$ be a uniformly random vector and $x_k = M^k x_0$. For $\varepsilon \geq 16/n$ and $k \geq \Theta(\log n/\varepsilon)$, it holds with constant probability that*

$$(1 - \varepsilon)\mu_n \leq \frac{x_k^T M x_k}{x_k^T x_k} \leq \mu_n.$$

Most results in this section have appeared in the literature before and we do not claim any novel contribution. To be self-contained and as our statements slightly differ from statements in the literature we present proofs of all presented lemmas in Section A. To the best of our knowledge, the only exception is the bit-precision lemma that we present at the end of the section. It is central to implementing the power method in CONGEST and essentially says that rounding intermediate values does not significantly affect the performance of the power method;

also see the paragraph on 'exploding values' in Section 1.4. It may be slightly stronger than what is needed in the centralized setting and potentially also of independent interest.

Lemma 7 states that there is a constant probability that the power method fails to find the largest eigenvalue. Hence, we need to run $O(\log n)$ instances of it and return the largest eigenvalue found during any of the independent runs. The following lemma shows that we never return any value larger than the largest eigenvalue. It is also needed to control the growth of numbers which is essential to implement the method in CONGEST.

Lemma 10. *For any vector $x \in \mathbb{R}^k$ we have that $\frac{x^T M x}{x^T x} \leq \mu_n$.*

Regardless of whether in the centralized setting or in CONGEST, the intermediate vectors may consist of quickly growing values that cannot be efficiently represented. The following lemma states that truncating all values after $O(\log n)$ bits does not affect the outcome of the power method by too much. Note that the statement may sound trivial as a precision of $1/\operatorname{poly} n$ for each single number seems to simply result in an imprecision of $k/\operatorname{poly} n$ after k steps. A closer look at the power method reveals that errors may amplify and exponentially grow over time. Hence, the next lemma requires an actual proof that appears in Section A.

Lemma 11. *Let $k \geq i \geq 1$. Throughout the power method, rounding each element of $x_i := M^i x_0$ to the $\Theta(\log n)$-th bit, does not impact the outcome of $\frac{(x_k)^T M x_k}{(x_k)^T x_k}$ by more than a factor $(1 \pm n^{-\Theta(1)})$.*

2.3 Warm-up: The Largest Eigenvalue

First, we show how the power method applied directly to the Laplacian gives an approximation of the largest eigenvalue.

Theorem 12 (Approximating λ_n)**.** *Let $\varepsilon > 0$. There is a randomized* CONGEST *algorithm that, given an undirected, weighted graph $G = (V, E)$, with high probability computes a $(1 \pm \varepsilon)$-approximation of the largest eigenvalue λ_n of the normalized Laplacian. The algorithm takes $O(D + \log^2 n/\varepsilon)$ rounds.*

Proof. We use the power method, Lemma 7, with $M = L$ and $k = \Theta(\log n/\varepsilon)$ iterations. Each iteration consists of a matrix-vector multiplication, in which each vertex v needs to send its current value of $(x)_v$ to its neighbors. After this, we perform one additional matrix-vector multiplication and compute $\frac{x^T M x}{x^T x}$ in $O(D)$ rounds.

Bit precision. Lemma 11 states that we can truncate each value after the $O(\log n)$ most significant bits. We additionally remember the order of magnitude of the values. Since we have $k = \Theta(\log n/\varepsilon)$ iterations, and our numbers are of size 2^k, we need at most $\log\log(2^k) = \log(\log n/\varepsilon) = O(\log n)$ bits to store the order of magnitude. Here we use that without loss of generality $\varepsilon = \Omega(1/n^2)$. We conclude that we can communicate each value up to the required precision in $O(\log n)$ bits.

Success Probability. Note that Lemma 7 only gives constant probability where the probability is over the choice of the random start vector. Repeating the process $O(\log n)$ times and take the maximum to obtain the result with high probability, since we cannot overshoot by Lemma 10. The matrix-vector multiplications take $O(\log n \cdot \frac{\log n}{\varepsilon})$. However, for every instance, we also need to compute a final inner product that takes $O(D)$ time each. We can pipeline these inner product for the $O(\log n)$ instances to obtain congestion plus dilation time for this part: $O(\log n + D)$. Combining the running times, we obtain $O(\log^2 n/\varepsilon + D + \log n) = (\log^2 n/\varepsilon + D)$ rounds. □

2.4 The Smallest Non-Zero Eigenvalue

Next, we consider the smallest non-zero eigenvalue. The main idea is to consider $M = 2I - L$, such that the smallest eigenvalue becomes the largest, and then apply the power method.

However, we are looking at the *smallest non-zero* eigenvalue. So we take additional measures to avoid finding the smallest ($= 0$) eigenvalue.

Theorem 13 (Approximating λ_2)**.** *Let $\varepsilon > 0$. There is a randomized* CONGEST *algorithm that, given an undirected, weighted graph $G = (V, E)$, with high probability computes an approximation of the smallest non-zero eigenvalue λ_2 of the normalized Laplacian. The algorithm takes*

$$O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon} \cdot \left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil\right) = O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon^2}\right)$$

rounds. At the end of the algorithm, every vertex outputs an estimation $\lambda_2 \leq \tilde{\lambda}_2 \leq \lambda_2 + \varepsilon$.

Proof. We check if $D > \Theta(\log n/\varepsilon)$ holds in $O(\log n/\varepsilon)$ rounds. If so, then we have $\log n/\varepsilon < \Theta(D) \leq \Theta(\log n/\phi)$ by Lemma 21, or equivalently $\phi < \Theta(\varepsilon)$. By Cheeger's Inequality, Theorem 8, $\lambda_2 \leq 2\phi \leq 2\varepsilon$. So we simply output $\tilde{\lambda}_2 = \Theta(\varepsilon)$ and we are done.

So for the remainder of the proof, we may assume $D = O(\log n/\varepsilon)$. The algorithm is similar as in Theorem 12, however, we now need to find the *smallest* (nonzero) eigenvalue. To flip smallest and largest, we use a standard trick and consider the matrix $M = 2I - L$ (instead of $M = L$ as in Theorem 12). M is still a PSD matrix, now with largest eigenvalue 2, for which we also know the largest eigenvector $v_n := \sqrt{\deg}$. We denote the eigenvalues of M by $0 \leq \mu_1 \leq \cdots \leq \mu_n = 2$. If we were to apply the power method on M, we would find this largest eigenvalue $\mu_n = 2$. Our goal is, however, to find μ_{n-1}. Thus, we aim to use it on the subspace orthogonal to v_n by nullifying contributions in direction v_n. Formally, we change the matrix M yet again.

To this end consider $M' := M - 2v_n v_n^T$. All eigenvectors and most eigenvalues stay the same: for $i < n$ we have

$$(M - 2v_n v_n^T)v_i = Mv_i - 2v_n\langle v_n, v_i\rangle = Mv_i = \mu_i v_i,$$

by orthogonality of the eigenvectors. Only eigenvector v_n changes its eigenvalues from 2 in matrix M to 0 in matrix M':

$$(M - 2v_n v_n^T)v_n = Mv_n - 2v_n\langle v_n, v_n\rangle = Mv_i = 2v_n - 2v_n = 0.$$

This means that the largest eigenvalue of M' is μ_{n-1} Suppose now we have found $\tilde{\mu}_{n-1}$ such that $(1-\varepsilon)\mu_{n-1} \leq \tilde{\mu}_{n-1} \leq \mu_{n-1}$ holds. Then also $\mu_{n-1} = 2 - \lambda_2$ holds, so we output $\tilde{\lambda}_2 := 2 - \tilde{\mu}_{n-1}$. We obtain

$$\begin{aligned}
&\tilde{\lambda}_2 = 2 - \tilde{\mu}_{n-1} \geq 2 - \mu_{n-1} = \lambda_2; \text{and} \\
&\tilde{\lambda}_2 = 2 - \tilde{\mu}_{n-1} \leq 2 - (1-\varepsilon)\mu_{n-1} = 2 - (1-\varepsilon)(2-\lambda_2) = (1-\varepsilon)\lambda_2 + 2\varepsilon \leq \lambda_2 + 2\varepsilon.
\end{aligned}$$

Setting $\varepsilon \leftarrow \varepsilon/2$ gives the result.

Finding the largest eigenvalue of M'. Again, we use the power method, Lemma 7, to approximate μ_{n-1}. We use the next claim to prove that for most steps, multiplying with M' is actually the same as multiplying with M.

Claim 1. *For all x, we have*

1. $M'x = Mx - 2v_n v_n^T x = Mx - 2v_n\langle x, v_n\rangle$;
2. $\langle M'x, v_n\rangle = 0$.

Proof. The first part of the claim is immediate. For the second part we compute

$$\begin{aligned}
\langle M'x, v_n\rangle = \langle (M - 2v_n v_n^T)x, v_n\rangle &= \left\langle (M - 2v_n v_n^T)\sum_{i=1}^n \langle x, v_i\rangle v_i, v_n \right\rangle \\
&= \left\langle \sum_{i=1}^n \mu_i\langle x, v_i\rangle v_i - 2v_n\langle x, v_n\rangle, v_n \right\rangle = \sum_{i=1}^n \mu_i\langle x, v_i\rangle\langle v_i, v_n\rangle - 2\langle x, v_n\rangle\langle v_n, v_n\rangle \\
&= 2\langle x, v_n\rangle - 2\langle x, v_n\rangle = 0.
\end{aligned}$$

◁

By Claim 1 part 1, we can compute $x_1 = M'x_0$ by a multiplication with M and inner product between x_0 and v_n. The latter can be computed in $O(D)$ time. Due to Claim 1 part 2, we have that the inner product between x_i and v_n is zero for all $i \geq 1$. Hence, after the first iteration we obtain $M'x_i = Mx_i$ and we can simply multiply with M instead of M'.

Bit precision. As in Theorem 12, we have by Lemma 11 that $O(\log n)$ bits suffice to store each value.

However, rounding might introduce a small error in the direction of v_n. As opposed to Lemma 11, this error could blow up over time: it scales with μ_n^l after l rounds, while the vector's component we want scales with μ_{n-1}^l. The number of iterations l after which this becomes a problem that depends on the ratio μ_n/μ_{n-1}. We will let this error blow up for l iterations, and then project again on the subspace orthogonal to v_n, using diameter time as above. Before we describe that, we prove how large l can be.

Formally, we prove the following claim, where δv_n denotes the error. For errors perpendicular to v_n, we can refer to Lemma 11, which also shows rounding errors are of size at most $\delta \leq n^{-\Theta(1)}||x_i||$.

Claim 2. *Let x_i be a vector such that $\langle x_i, v_n\rangle = 0$, and $0 < \delta \leq n^{-\Theta(1)}||x_i||$ be a parameter. Then for $l \leq O(\frac{\log n}{\lambda_2})$ we have*

$$\left|\frac{(x_i+\delta v_n))^T M^{2l+1}(x_i+\delta v_n)}{(x_i+\delta v_n)^T M^{2l}(x_i+\delta v_n)} - \frac{x_i^T M^{2l+1} x_i}{x_i^T M^{2l} x_i}\right| \leq n^{-\Theta(1)}\frac{x_i^T M^{2l+1} x_i}{x_i^T M^{2l} x_i}.$$

Proof. We first investigate the first term, i.e., the Rayleigh coefficient including the error after l iterations

$$\frac{(x_i+\delta v_n))^T M^{2l+1}(x_i+\delta v_n)}{(x_i+\delta v_n)^T M^{2l}(x_i+\delta v_n)} = \frac{x_i^T M^{2l+1} x_i + \delta^2\mu_n^{2l+1}}{x_i^T M^{2l} x_i + \delta^2\mu_n^{2l}} \leq \frac{x_i^T M^{2l+1} x_i + n^{-\Theta(1)}||x_i||^2\mu_n^{2l+1}}{x_i^T M^{2l} x_i - n^{-\Theta(1)}||x_i||^2\mu_n^{2l}},$$

where the equality follows from orthonormality of the eigenvectors, and the inequality follows from our bound on the error. If we now show for the numerator that $n^{-\Theta(1)}||x_i||^2\mu_n^{2l+1} \leq n^{-\Theta(1)}x_i^T M^{2l+1}x_i$, then this also holds for the denominator (since this is a factor of M less), and hence we obtain that the Rayleigh coefficient after l iterations changed by a factor at most $(1+n^{-\Theta(1)})$. Hereto, we inspect $x_i^T M^{2l+1}x_i$. We note that $|\langle x_0, \rangle| \geq \frac{3}{16}$. We obtain

$$x_0^T M^{2(i+l)+1}x_0 = x_0^T M^{2(i+l)+1}x_0 \geq \mu_{n-1}^{2(i+l)+1}|\langle x_0, \rangle|^2 \geq \mu_{n-1}^{2(i+l)+1}\left(\frac{3}{16}\right)^2.$$

On the other hand, we have that $||x_i||^2 = x_0^T M^{2i} x_0 \leq \mu_{n-1}^{2i} \cdot ||x_0||^2 = \mu_{n-1}^{2i} \cdot n$. Putting this together, we obtain that

$$\begin{aligned} n^{-\Theta(1)}||x_i||^2\mu_n^{2l+1} &\leq n^{-\Theta(1)} \cdot \mu_{n-1}^{2i} n \cdot \mu_n^{2l+1} = n^{-\Theta(1)} \cdot \mu_{n-1}^{2(i+l)+1} \cdot \left(\frac{\mu_n}{\mu_{n-1}}\right)^{2l+1} \\ &\leq n^{-\Theta(1)} \cdot \left(\frac{\mu_n}{\mu_{n-1}}\right)^{2l+1} \cdot x_0^T M^{2(i+l)+1} x_0. \end{aligned}$$

If we now show that $\left(\frac{\mu_n}{\mu_{n-1}}\right)^{2l+1} \leq n^{\Theta(1)}$, then we are done. Here we use that l is bounded: after many iterations, the largest eigenvector will always dominate. By assumption, we have $l = \Theta(\frac{\log n}{\lambda_2})$. Using this, together with $\mu_n = 2$ and $\mu_{n-1} = 2 - \lambda_2$, we see

$$\left(\frac{\mu_n}{\mu_{n-1}}\right)^l = \left(\frac{2}{2-\lambda_2}\right)^l \leq (1+\lambda_2)^l \leq \exp(\lambda_2 \cdot l) = \exp(\Theta(\log n) = n^{\Theta(1)}. \qquad \triangleleft$$

This claim shows that for $l = \Theta(\log n/\lambda_2)$ iterations, the rounding error does not propagate too much. After every $\Theta(\log n/\lambda_2)$ iterations, we orthogonalize again. We can simply do this by computing $x - \langle x, v_n\rangle v_n$, since v_n is known: $v_n = \sqrt{\deg}$. However, this leads to $O(\left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil)$ additional inner products.

Success Probability. Again, we note that Lemma 7 only gives constant probability. We can simply repeat the process $O(\log n)$ times and take the maximum to obtain the result with high probability, since we cannot overshoot by Lemma 10. As before, we can stream the $O(\lceil \lambda_2/\varepsilon \rceil)$ inner products, so the total time becomes:

$$O\left(\frac{\log^2 n}{\varepsilon} + \log n \left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil + D \left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil\right) = O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon} \cdot \left\lceil \frac{\lambda_2}{\varepsilon} \right\rceil\right).$$

We can also simplify this to $O\left(\frac{\log^2 n}{\varepsilon} + \frac{\log n}{\varepsilon^2}\right)$, using that $\lambda_2 \leq 2$. ◀

2.5 The k Smallest Eigenvalues

In this section, we generalize the method from Section 2.4. The general idea remains to change the matrix by projecting on the space orthogonal to the found eigenvectors. However, the found eigenvectors are now no longer exact, but approximate. We first show that the power method can work with such approximations as well. The following lemma captures the main result of this section.

Lemma 14. *Let $k \geq 0$ be a constant. Suppose $\tilde{\mu}_i, \tilde{v}_i$ for $i \geq n-k$ are orthonormal vectors that approximate the eigenvalues thus-far: $\tilde{\mu}_i = \tilde{v}_i^T M \tilde{v}_i$ and $(1 - \text{poly}(\varepsilon))\mu_i \leq \tilde{\mu}_i \leq \mu_i$. Performing $O(\log n)$ instances of the power method – each with $O(\log n/\varepsilon)$ iterations – with the matrix $M_{\text{approx}} := M - \sum_{i>n-k} \tilde{\mu}_i \tilde{v}_i \tilde{v}_i^T$ gives w.h.p. either a $(1-\varepsilon)$-approximation $\tilde{\mu}_{n-k}$ of the k-th largest eigenvalue μ_{n-k}, together with a corresponding vector $\tilde{v}_{n-k}$, or we can output that $\mu_{n-k} < \varepsilon$.*

Suppose we know the exact eigenvalues and eigenvectors μ_i, v_i for $i > n-k$, consider $M_{\text{ideal}} := M - \sum_{i>n-k} \mu_i v_i v_i^T$. This matrix has the same eigenvectors v_i, but now largest eigenvalue μ_{n-k}. This follows from orthonormality of the eigenvalues. Indeed we see for $i > n-k$

$$(M - \sum_{i>n-k} \mu_i v_i v_i^T) v_j = M v_j - \mu_j v_j = 0$$

and for $i \leq n-k$

$$(M - \sum_{i>n-k} \mu_i v_i v_i^T) v_j = M v_j = \mu_j v_j.$$

So if we obtain a $(1-\varepsilon)$-approximation of the largest eigenvalue of matrix M_{ideal}, we are done. We will show that the largest eigenvalue of M_{approx} is a $(1-\varepsilon)$ approximation of the largest eigenvalue of M_{ideal}. The power method gives with constant probability a $(1-\varepsilon)$-approximation of the largest eigenvalue. Repeating this $O(\log n)$ times and taking the maximum over these iterations gives the result with high probability. Note that we cannot overshoot by Lemma 10, hence taking the maximum indeed works. We can stream these iterations such that the $\log n$-factor does not multiply with D.

We note that a big difference to Theorem 13 is that we now do not have exact orthogonality *at any point* in time. This also means that rounding does not change this fact. Such errors are corrected by taking inner products out *every step* – rather than periodically.

Furthermore, the reader might notice that we state Lemma 7 for positive semi-definite matrices, and we no longer claim that M_{approx} is positive semi-definite. There are two solutions for this: 1) the power method also works on general matrices [GVL13]. The running time is

a constant factor slower, and it finds the *dominant* eigenvalue rather than the largest. 2) We note that M_{ideal} *is* positive semi-definite. Since we show that the eigenvalues of M_{ideal} and M_{approx} are close, this means that the eigenvalues in M_{approx} are at least $-\varepsilon$. By running the power method on $M_{\text{approx}} + \varepsilon I$, we have a PSD matrix again, and can use Lemma 7 as before. The additive matrix $+\varepsilon I$ has a linear effect on the eigenvalue approximation, so we can simply remove it at the end.

It remains to show that M_{approx} approximates M_{ideal}, which we do in Section 2.5.1. Then we show how this applies to the Laplacian in the CONGEST model in Section 2.5.2.

2.5.1 M_{approx} approximates M_{ideal}.

First, we note that we can assume the spectrum to be relatively simple.

Lemma 15. *Without loss of generality, we can assume that the eigenvalues take values in* $\{\mu_n, (1-\varepsilon)\mu_n, (1-\varepsilon)^2\mu_n, \dots\}$.

Proof. Instead of analyzing the matrix M, with orthonormal eigensystem $\{(\mu_i, v_i)\}$, we analyze the following system. We round every eigenvalue μ_i down to $\hat{\mu}_i$, the next value in $\{\mu_n, (1-\varepsilon)\mu_n, (1-\varepsilon)^2\mu_n, \dots\}$. Then we consider the matrix

$$M' := M - \sum_{i \in [n]} (\mu_i - \hat{\mu}_i) v_i v_i^T.$$

By orthonormality of the eigenvectors, M' has the same eigenvectors. Its eigenvalues, however, have changed to $\hat{\mu}_i$. This gives an error of at most $\epsilon\mu_i$ for each eigenvalue μ_i. Setting $\varepsilon \leftarrow \varepsilon/2$, makes that this additional error does not impact the final answer. □

In particular, this means that the second largest eigenvalue is at most $(1-\varepsilon)\mu_n$. This gives us the following lemma, stating that the previous iterations are well aligned with the eigenvectors.

Lemma 16. *Suppose* $||y|| = 1$ *and* $\frac{y^T M y}{y^T y} \geq (1-\delta)\mu_n$. *Then*

$$\langle y, v_n \rangle \geq \sqrt{1 - \delta/\epsilon}.$$

Proof. Since $|y|| = 1$, we have $y^T M y \geq (1-\delta)\mu_n$. We use this to see:

$$\begin{aligned}\sum_{i=1}^{n} \langle y, v_i \rangle^2 \mu_i &= \left(\sum_{i=1}^{n} \langle y, v_i \rangle v_i\right)^T M \left(\sum_{i=1}^{n} \langle y, v_i \rangle v_i\right) \\ &= y^T M y \geq (1-\delta)\mu_n.\end{aligned}$$

Since the eigenvalues are $(1-\varepsilon)$ apart, let j the highest index such that $\mu_j < (1-\varepsilon)\mu_n$, which means that $\mu_i = \mu_n$ for $i > j$. So

$$\sum_{i>j} \langle y, v_n \rangle^2 \mu_n + \sum_{i \leq j} \langle y, v_n \rangle^2 (1-\varepsilon)\mu_n \geq (1-\delta)\mu_n.$$

Using that y has norm 1, and that $\mu_n > 0$

$$\sum_{i>j} \langle y, v_n \rangle^2 + (1 - \sum_{i>j} \langle y, v_n \rangle^2)(1-\varepsilon) \geq (1-\delta).$$

Rearranging gives

$$\sum_{i>j} \langle y, v_i \rangle^2 \geq 1 - \delta/\epsilon.$$

However, v_i for $i > j$ all have the same eigenvalue μ_n, so we can just rearrange the orthonormal basis such that $\langle y, v_n \rangle \geq \sqrt{1 - \delta/\epsilon}$. □

Intuitively, this means that if we set $\delta = \varepsilon^2$, we get $\langle y, v_n \rangle \geq \sqrt{1-\varepsilon} \geq 1 - \varepsilon$. In reality, the analysis is a bit more involved – so we carry the δ for now.

Next, we show that if the inner product of two vectors is small, the difference in their projections is also small. Here, we denote $||A||$ for the *matrix norm* or *operator norm* of a matrix A, defined by $||A|| := \max_{||x||=1} ||Ax|| = \max_{||x||=1} x^T A^T A x$. Note that in our case of symmetric, positive semi-definite matrices, the norm of a matrix is bounded by its largest eigenvalue squared.

Lemma 17. *If $||v|| = ||w|| = 1$, $\langle v, w \rangle \geq 1-\eta$ and $a, b \geq 0$, $0 < \eta < 1$ such that $(1-\eta)a \leq b \leq a$, then we have $||avv^T - bww^T|| \leq 5\eta b$.*

Proof. First we note that

$$\begin{aligned} ||avv^T - bww^T|| &= ||(a-b)vv^T + b(vv^T - ww^T)|| \leq ||(a-b)vv^T|| + ||b(vv^T - ww^T)|| \\ &= |a-b| \cdot ||vv^T|| + b||vv^T - ww^T|| \leq b(\eta + ||vv^T - ww^T||). \end{aligned}$$

Next, we bound $||vv^T - ww^T||$:

$$\begin{aligned} ||vv^T - ww^T|| &= ||(v-w)v^T + w(v-w)^T|| \leq ||(v-w)v^T|| + ||w(v-w)^T|| \\ &= ||v-w|| \cdot ||v|| + ||w|| \cdot ||v-w|| = 2||v-w||, \end{aligned}$$

where we know that

$$||v-w|| = \langle v-w, v-w \rangle = \langle v, v \rangle + \langle w, w \rangle - 2\langle v, w \rangle = 2(1 - \langle v, w \rangle).$$

Putting this together we obtain

$$||avv^T - bww^T|| \leq b(\eta + ||vv^T - ww^T||) \leq b(\eta + 2||v-w||) \leq b(\eta + 4(1 - \langle v, w \rangle)) \leq 5\eta b.$$

□

Finally, we use the following theorem from perturbation theory (see, e.g., [HJ12]).

Theorem 18 (Weyl's Inequality)**.** *Let A, E be real, symmetric n by n matrices. Let $\mu_i(A), \mu_i(A+E)$ denote the i-th eigenvalue of A, $A+E$ respectively, for $i = 1, \ldots n$. Then we have*

$$|\mu_i(A+E) - \mu_i(A)| \leq ||E||.$$

We apply Lemma 16, Lemma 17 and Theorem 18 to our case:

$$\begin{aligned} |\mu_i(M_{\text{ideal}}) - \mu_i(M_{\text{approx}})| &\leq ||M_{\text{ideal}} - M_{\text{approx}}|| \\ &= \left|\left| \sum_{i>n-k} \tilde{\mu}_i \tilde{v}_i \tilde{v}_i^T - \sum_{i>n-k} \mu_i v_i v_i^T \right|\right| \\ &\leq \sum_{i>n-k} \left|\left| \tilde{\mu}_i \tilde{v}_i \tilde{v}_i^T - \mu_i v_i v_i^T \right|\right| \\ &\leq \sum_{i>n-k} 5 \cdot \delta/\varepsilon \cdot \mu_i \\ &\leq k \cdot 5 \cdot \delta/\varepsilon \cdot \mu_n. \end{aligned}$$

We need this to be at most $\varepsilon\mu_{n-k}/2$, where the factor two comes from Lemma 15. Hereto, we set $\delta_i := \frac{\varepsilon^2}{10} \frac{\mu_n}{\mu_{n-k}}$. Finally, we use that $\mu_n \leq 2$ and that we can assume $\mu_{n-1} \geq \varepsilon$. The latter holds, because if $\mu_{n-1} < \varepsilon$, we still have that $\tilde{\mu}_{n-1} \leq \mu_{n-1}$. It might be below $(1-\varepsilon)\mu_{n-1}$, but we can recognize this case, as needed for Lemma 14. So we can bound μ_n/μ_{n-k} by $2/\varepsilon$. To obtain our result, Lemma 14, it hence suffices to set $\delta_i = \frac{\varepsilon^3}{20k}$.

2.5.2 Approximating the k-th Smallest Eigenvalue in CONGEST

Theorem 19 (Approximating $\lambda_1, \ldots, \lambda_k$)**.** *Let $\varepsilon > 0$ be a parameter and $k \geq 1$ be a constant. There exists a* CONGEST *algorithm that, given an undirected, weighted graph $G = (V, E)$, with high probability finds k approximations $\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3, \ldots, \tilde{\lambda}_k$ of the first k eigenvalues of the normalized Laplacian such that*

$$\lambda_i \leq \tilde{\lambda}_i \leq \lambda_i + \varepsilon,$$

in $O(\log^2 n \operatorname{poly}(\varepsilon^{-1}))$.

Proof. First, we show that we can assume the diameter to be small. We check in $O(k \log n/\varepsilon)$ rounds if the diameter is bigger than $\Theta(k \log n/\varepsilon)$. As seen in Lemma 21, if $D > \Theta(k \log n/\varepsilon)$, we have that $\phi_k \leq \varepsilon$. By Theorem 9, we have that $\lambda_k \leq 2\phi_k \leq 2\varepsilon$. So we set $\varepsilon \leftarrow \varepsilon/2$. And can then output $\tilde{\lambda}_i = \varepsilon$ for all $i \leq k$. For the remainder, we can assume that $D = O(k \log n/\varepsilon)$.

By Lemma 14, we need to run the power method with the matrix $M_{\text{approx}} = M - \sum_{i>n-k} \tilde{\mu}_i \tilde{v}_i \tilde{v}_i^T$. We need to run $O(\log n)$ instances to obtain our result with high probability, and each instance needs $O(\log n/\delta)$ iterations, with $\delta = \operatorname{poly}(1/\varepsilon)$. Note that each iteration requires the previous iteration to be more precise. In the end reaching a bigger $\operatorname{poly}(\varepsilon^{-1})$.

Note that we are not running the power method with a locally defined matrix anymore. We use $M_{\text{approx}} := M - \sum_{i>n-k} \tilde{\mu}_i \tilde{v}_i \tilde{v}_i^T$, where $M = 2I - L$ *is* locally defined. Compute $M_{\text{approx}} x$ consists of multiple steps. First we compute $Mx = (2I - L)x$, which can be done in $O(1)$ rounds. Next, we need to compute each of the at most k inner products $\tilde{v}_i^T x$. We can pipeline these inner products for the $\log n$-instances in $O(D + \log n)$ times. Recall that $\lambda_1 = 0$, with eigenvector $\sqrt{\deg}$. For M_{approx}, this means the largest eigenvalue $\mu_n = 2$ with eigenvector $v_n = \sqrt{\deg}$. We know this vector exactly, and do not need to use an approximation.

So we get total time $O(\log n \cdot \log n/\delta + \log n/\delta \cdot k \cdot (D + \log n) = O(\log^2 n/\delta + \log nD)$. Filling in $D = O(k \log n/\varepsilon)$ and $\delta = \operatorname{poly}(1/\varepsilon)$ gives the result. □

2.6 Lower Bounds For Eigenvalue Approximation

For our lower bounds, we always assume that the graph is connected, as in standard in the LOCAL and CONGEST model. We allow for the general case that vertices know the value n.

Theorem 6. *Any algorithm that with probability $> \frac{1}{2}$ ensures that all vertices output a value $\tilde{\lambda}_2 \geq \lambda_2$, and at least one vertex additionally satisfies $\tilde{\lambda}_2 \leq 2\lambda_2$ requires $\Omega(D)$ rounds.*

Proof. We consider two classes of graphs. If with probability $> \frac{1}{2}$ it is correct on each, then there has to exist output that is correct on both. We consider this case.

First, consider a cycle on k vertices C_k, connect this to a clique on $l = \Omega(n)$ vertices K_l. We denote this graph by $G_{k,l}$.

Second, consider a cycle on k vertices C_k, connect this to a clique on l vertices K_l on one side, and a clique on l' vertices. We denote this graph by $G_{k,l,l'}$.

Claim 1. We have $\lambda_2(G_{k,l}) = \Theta(1/k^2)$. We consider the following eigenvector of the cycle C_k: $v_i = \sin(\frac{2\pi i}{k})$, which has eigenvalue $1 - \cos(\frac{2\pi}{k})$ [Chu97]. We note in particular that $v_0 = 0$. Since we connect the cycle to the clique at v_0, we can extent this vector to a vector that is all 0s at other places. Now we have

$$\frac{v^T L(G_{k,l}) v}{v^T v} = \frac{v^T L(C_k) v}{v^T v} = 1 - \cos(\tfrac{2\pi}{k}) = O(1/k^2). \tag{1}$$

Further, by orthogonality we have that $v \perp w$, where $w = \sqrt{\deg}$ is the eigenvector corresponding to eigenvalue 0. One can show that $\lambda_2 = \min_{v \perp w} \frac{v^T L(G_{k,l}) v}{v^T v}$ [Chu97]. This means that for *any* $v \perp w$ we have

$$\lambda_2 \leq \frac{v^T L(G_{k,l}) v}{v^T v}.$$

We have $\phi(G_{k,l}) = \Theta(1/k)$, there are two types of cuts to inspect to proof this: cutting through the cycle or cutting through the clique. Cutting through the cycle is most balanced, hence sparsest, if the cut S is closest to the cycle. In this case we have

$$\phi(S) = \frac{|E(S, V \setminus S)|}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(V \setminus S)\}} = \frac{2}{\min\{\Theta(l^2), 2(k-1)\}} = \Theta(1/k).$$

Cutting through the clique always gives a bigger value: say we have $l' < l$ vertices of the clique in S, then

$$\phi(S) = \frac{|E(S, V \setminus S)|}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(V \setminus S)\}} = \frac{l'(l-l')}{\min\{l'(l-1), (l-l')(l-1) + 2k\}} \geq \frac{l-l'}{l-1} \geq \frac{1}{l-1} = \Theta(1/n).$$

Since Cheeger's Inequality, Theorem 8, gives us that $\lambda_2 \geq \phi^2/2$, we get $\lambda_2(G_{k,l}) = \Theta(1/k^2)$.

Claim 2. We have $\lambda_2(G_{k,l,l'}) = O(1/k^2)$. The upper bound follows from the same argument as in Claim 1, with the additional observation that $v_{k/2} = 0$ as well.

Suppose a vertex v sees its r-hop neighborhood, for some $r < k/2$. This means the vertex sees at most one clique, say of l vertices. The graph can still be $G_{k,l}$ or $G_{k,l,l'}$. For any $k \geq 2r$ (and $k \leq n - l$). So we need to output $\tilde{\lambda}_2 \geq \Theta(1/r^2)$. However, the graph can also be $G_{2D,l,l'}$ with $\lambda_2(G_{2D,l,l'}) \leq O(1/D^2)$. So we need $\Theta(1/r^2) \leq (1+\varepsilon)O(1/D^2)$, so $r = \Omega(D)$. □

2.7 Approximating the Entire Spectrum

Via similar methods, one can also approximate the entire spectrum. However, the output will not reveal how many eigenvalues there are of a certain approximation, i.e., it gives the spectrum *without multiplicity.*

Remark 20. *We can compute a multiplicative approximation of the entire spectrum (without multiplicity) in $n^{o(1)}(\sqrt{n} + D)/\varepsilon^2$ rounds.*

Sketch. Note that this statement does not have a linear dependency on n, even though there are n eigenvalues and from a high level point of view we would need to address them sequentially with the power method. The linear dependency does not appear in the runtime as we only compute the spectrum approximately and without multiplicity. Thus, essentially we only need to test $O(\log(Wn)/\varepsilon)$ regimes in $[0, 2]$ for the existence of an eigenvalue.

The algorithm is now as follows. One can also find an eigenvalue with σ by the power method on $(L - \sigma I)^{-1}$, with $\tilde{\sigma}$ some estimate of σ. By running this with $\sigma = 2/(1+\varepsilon)^i$ for $i = 1, \cdots, O(\log(Wn)/\varepsilon)$, we cover the entire spectrum. Now we have to perform matrix-vector multiplication with $(L - \sigma I)^{-1}$, which takes $n^{o(1)}(\sqrt{n} + D)$ rounds in each step [FGLP+21]. Multiplying this by $O(\log n/\varepsilon)$ iterations, we obtain in total $n^{o(1)}(\sqrt{n} + D)/\varepsilon^2$ rounds. □

3 Approximating the Sparsest Cut

In this section, we use the approximate eigenvalues from Section 2.7 to approximate the sparsest cut in the CONGEST model. We also give the complexity of the problem in the LOCAL model and we provide a lower bound in *unweighted* graphs.

k-Way Sparsest Cut and Diameter We first observe that the diameter of a graph is related to its k-way sparsest cut. The result follows from a ball growing argument, that is (at least) known for the case $k = 2$. We prove the generalization for completeness.

Lemma 21. *If $G = (V, E)$ has k-way sparsest cut ϕ_k, then we have $D = O(k \log n/\phi_k)$.*

Proof. Suppose not. This means that from any vertex v_1, there are vertices up to distance $\Theta(k \log n/\phi_k)$. Select vertices $v_2, v_3, \ldots, v_k$ such that $d(v_i, v_j) > \Theta(\log n/\phi_k)$ for $i \neq j$. Now, for any v_i, suppose for contradiction that there are no cuts S within a distance $\Theta(\log n/\phi_k)$ with $\phi(S) \leq \phi_k$. Then for any $j \leq \log n/\phi_k$, the j-hop neighborhood $N_j(v_i)$ satisfies $|E(N_j(v_i), V \setminus N_j(v_i))| \geq \phi_k \operatorname{Vol}(N_j(v_i))$, so within $\log_{1+\phi_k} n = O(\log n/\phi_k)$ steps, we cover the entire graph, a contradiction. □

3.1 CONGEST Upper Bound for Approximating the Sparsest Cut

We combine the Cheeger's Inequality with our eigenvalue approximation to obtain the following result.

Theorem 1. *There is a randomized CONGEST algorithm that, given an undirected weighted n-node graph with sparsest cut ϕ, w.h.p. gives an approximation of the sparsest cut in $O(\log^2 n/\phi)$ rounds. In particular, every vertex outputs $\tilde{\phi}$ such that $\phi \leq \tilde{\phi} \leq \sqrt{2.01\phi}$.*

Proof. By Theorem 13, we can find $\tilde{\lambda}_2$ such that

$$\lambda_2 \leq \tilde{\lambda}_2 \leq \lambda_2 + \delta.$$

This takes $O\left(\frac{\log^2 n}{\delta} + \frac{\log n}{\delta} \cdot \left\lceil \frac{\lambda_2}{\delta} \right\rceil\right)$ rounds.

Now we output $\tilde{\phi}_2 = \sqrt{2\tilde{\lambda}_k}$. Using Cheeger's Inequality, Theorem 8, we see that

$$\tilde{\phi} = \sqrt{2\tilde{\lambda}_k} \geq \sqrt{2\lambda_k} \geq \phi$$

and that

$$\tilde{\phi} = \sqrt{2\tilde{\lambda}_2} \leq \sqrt{2\lambda_2 + 2\delta} \leq \sqrt{2\phi + 2\delta}.$$

When we set $\delta = \varepsilon^2/2$ we see that

$$\tilde{\phi} \leq \sqrt{2\phi + 2\delta} = \sqrt{2\phi + \varepsilon^2} \leq \sqrt{2\phi} + \varepsilon.$$

Hence we can attain the $\sqrt{n\phi}$-bound of Cheeger's Inequality up to an arbitrarily small additive error ε in $O(\log^2 n/\varepsilon^2 + \log n/\varepsilon^4)$ rounds. When we set $\delta = 0.005\phi$, we get the stated approximation: $\tilde{\phi} \leq \sqrt{2.01\phi}$. This takes

$$O\left(\frac{\log^2 n}{\delta} + \frac{\log n}{\delta} \cdot \left\lceil \frac{\lambda_2}{\delta} \right\rceil\right) = O\left(\frac{\log^2 n}{\phi} + \frac{\log n}{\phi} \cdot \left\lceil \frac{\lambda_2}{\phi} \right\rceil\right)$$

rounds. Cheeger's Inequality states that $\frac{\lambda_2}{2} \leq \phi$, so $\frac{\lambda_2}{\phi} \leq 2$. This simplifies the running time to $O\left(\frac{\log^2 n}{\phi} + \frac{\log n}{\phi}\right) = O\left(\frac{\log^2 n}{\phi}\right)$ rounds.

Note that, of course, ϕ is not known at the beginning of the algorithm, but we can approximate it without additional asymptotic costs via an exponential search with $\phi = 1/2, 1/4, \ldots$. □

3.2 CONGEST Upper Bound for Approximating the k-Way Sparsest Cut

We combine the generalized Cheeger's Inequality with our eigenvalue approximation to obtain the following result.

Theorem 2. *There is a randomized* CONGEST *algorithm that, given a constant $k \geq 2$ and undirected weighted graph with k-way sparsest cut ϕ_k, w.h.p. gives an approximation of the k-way sparsest cut in $O(\log^2 n \operatorname{poly}(\phi_k^{-1}))$ rounds. In particular, every vertex outputs $\tilde{\phi}_k$ such that $\phi_k \leq \tilde{\phi}_k \leq O(\sqrt{\phi_k})$.*

Proof. By Theorem 19, we can find $\tilde{\lambda}_k$ such that

$$\lambda_k \leq \tilde{\lambda}_k \leq \lambda_k + \varepsilon.$$

This takes $O(\log^2 n \operatorname{poly}(\varepsilon^{-1}))$ rounds.

Now we output $\tilde{\phi}_k = \Theta(k^2)\sqrt{\tilde{\lambda}_k}$. Using the higher order Cheeger's Inequality, Theorem 9, we see that

$$\tilde{\phi}_k = \Theta(k^2)\sqrt{\tilde{\lambda}_k} \geq \Theta(k^2)\sqrt{\lambda_k} \geq \phi_k$$

and that

$$\tilde{\phi}_k = \Theta(k^2)\sqrt{\tilde{\lambda}_k} \leq \Theta(k^2)\sqrt{\lambda_k + \varepsilon} \leq \Theta(k^2)\sqrt{2\phi_k + \varepsilon}.$$

Setting $\varepsilon = \phi_k$ gives the result. □

3.3 Lower Bounds

As before, we assume that the graph is connected, as in standard in the LOCAL and CONGEST model. We allow for the general case that vertices know the value n. First, we give a diameter lower bound for all possible diameters.

Theorem 3. *Let $D \leq n/2$ be a parameter and consider the class of graphs with diameter D. An algorithm that with probability $> \frac{1}{2}$ satisfies that every vertex v of a graph G outputs a value $\tilde{\phi}_v \geq \phi(G)$, for sparsest cut $\phi(G)$, and at least one vertex has $\phi_v \leq \Theta(n \cdot \phi)$ needs $\Omega(D)$ rounds in* LOCAL.

Proof. We consider two types of graphs with n vertices and diameter D. If with probability $> \frac{1}{2}$ it is correct on each, then there has to exist output that is correct on both. We consider this case. Let N denote $(n - D + 2)/2$.

Let G_1 be a path of length $D - 2$, connected to a clique on N vertices on both sides. This graph has diameter D and sparsest cut $\Theta(1/n^2)$. For the latter, we consider a cut S, cutting the graph anywhere on the path. The volume on both sides is $\Theta(n^2)$, since each clique has n^2 edges, so this gives a cut of sparsity

$$\phi(S) = \frac{|E(S, V \setminus S)|}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(V \setminus S)\}} = \frac{1}{\min\{\Theta(n^2), \Theta(n^2)\}} = \Theta(1/n^2).$$

Cutting through the clique always gives a bigger value: say we have $M < N$ vertices of the clique in S, then

$$\phi(S) = \frac{|E(S, V \setminus S)|}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(V \setminus S)\}} = \frac{M(N - M)}{\min\{M(N-1), \Theta(n^2)\}} = \frac{N - M}{N - 1} \geq \frac{1}{N - 1} = \Theta(1/n).$$

Second, consider G_2, a path of length D, connected to a clique on N vertices on one side, and connected to the center of a star graph on N vertices on the other side. This graph also

has diameter D, but the sparsest cut is $\Theta(1/n)$. As before, cutting through the clique gives $\phi(S) \geq \Theta(1/n)$. When we cut anywhere not on the path, we are cutting a tree. To get the biggest volume (and hence smallest sparsity), we cut the first edge after the clique

$$\phi(S) = \frac{|E(S, V \setminus S)|}{\min\{\text{Vol}(S), \text{Vol}(V \setminus S)\}} = \frac{1}{\min\{D + N, \Theta(n^2)\}} = \frac{1}{\min\{\Theta(n), \Theta(n^2)\}} = \Theta(1/n).$$

No vertex in G_1 can discern between those two graphs in less than $\Theta(D)$ time: a vertex can only see at most one clique. The other side could be the star. So every vertex v needs to output $\tilde{\phi}_v \geq \Theta(1/n)$, but this is only a $\Theta(n)$ approximation for $\phi(G_1) = \Theta(1/n^2)$. □

Second, we give a diameter lower bound for most values of ϕ.

Theorem 4. *Let $0 < \varepsilon < 1$ be a constant and let $\phi > 1/n^{1-\varepsilon}$. An algorithm that with probability $> \frac{1}{2}$ satisfies that every vertex v outputs a value $\tilde{\phi}_v$ such that $\phi \leq \tilde{\phi} \leq O(n^{1+\varepsilon} \cdot \phi)$, where ϕ is the sparsest cut value of the graph needs at least $\Omega(D)$ rounds in the* LOCAL *model.*

Proof. Fix $\phi > 1/n^{1-\varepsilon}$. Chierichetti et al. [CGLP18] showed that there exist graphs with sparsest cut ϕ and with diameter $D = \Theta(\log n/\phi)$. Consider such a graph G. Say that after $r = D/3$ rounds every vertex outputs $\tilde{\phi}_v$ such that $\phi \leq \tilde{\phi}_v \leq \Theta(n^{1+\varepsilon} \cdot \phi)$. If every vertex has more than $n/2$ vertices in its r-hop neighborhood, then the diameter is at most $2r < D$, a contradiction. Consider a vertex $v \in V$ with most $n/2$ vertices in its r-hop neighborhood. By assumption v needs to output $\tilde{\phi}_v \geq \phi$.

Now consider a graph G', consisting of v and its r-hop neighborhood as before, but the remaining vertices form 2 cliques on $n/4$ vertices, C_1 and C_2. C_1 connected to C_2 by one edge, and C_2 is connected to the r-hop neighborhood of v. This graph has sparest cut at most $\Theta(1/n^2)$. Indeed, consider the cut S between C_1 and C_2: this has one edge, and volume $\Theta(n^2)$ on each side, so $\phi(S) \leq \Theta(1/n^2)$. By assumption, v needs to output $\tilde{\phi}_v \leq \Theta(n^{1+\varepsilon} \cdot 1/n^2) = \Theta(1/n^{1-\varepsilon}) < \phi$, for appropriately chosen constants.

If with probability $> \frac{1}{2}$ the algorithm is correct on each G and G', then there has to exist output that is correct on both. We consider this case.

Since v cannot distinguish between these two graphs, it needs to output $\tilde{\phi}_v \geq \phi$ *and* $\tilde{\phi}_v < \phi$, a contradiction. □

References

[ABS15] Sanjeev Arora, Boaz Barak, and David Steurer. "Subexponential Algorithms for Unique Games and Related Problems". In: *J. ACM* 62.5 (2015). Announced at FOCS '10, 42:1–42:25. DOI: 10.1145/2775105. URL: https://doi.org/10.1145/2775105 (cit. on p. 3).

[ALHZ+22] Ioannis Anagnostides, Christoph Lenzen, Bernhard Haeupler, Goran Zuzic, and Themis Gouleakis. "Almost Universally Optimal Distributed Laplacian Solvers via Low-Congestion Shortcuts". In: *Proc. of the 36th International Symposium on Distributed Computing, DISC 2022*. Vol. 246. LIPIcs. Announced at PODC 2022. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 6:1–6:20. arXiv: 2109.05151 (cit. on p. 6).

[Alo86] Noga Alon. "Eigenvalues and expanders". In: *Comb.* 6.2 (1986), pp. 83–96. DOI: 10.1007/BF02579166. URL: https://doi.org/10.1007/BF02579166 (cit. on pp. 2, 10).

[AM85] Noga Alon and V. D. Milman. "λ_1, Isoperimetric inequalities for graphs, and superconcentrators". In: *J. Comb. Theory B* 38.1 (1985), pp. 73–88. DOI: 10.1016/0095-8956(85)90092-9. URL: https://doi.org/10.1016/0095-8956(85)90092-9 (cit. on pp. 2, 10).

[AMV20] Kyriakos Axiotis, Aleksander Mądry, and Adrian Vladu. "Circulation Control for Faster Minimum Cost Flow in Unit-Capacity Graphs". In: *Proc. of the 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 2020, pp. 93–104. arXiv: 2003.04863 (cit. on p. 6).

[ARV09] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. "Expander flows, geometric embeddings and graph partitioning". In: *J. ACM* 56.2 (2009). Announced at STOC'04, 5:1–5:37. DOI: 10.1145/1502793.1502794. URL: https://doi.org/10.1145/1502793.1502794 (cit. on pp. 1, 5).

[ASZ20] Alexandr Andoni, Clifford Stein, and Peilin Zhong. "Parallel approximate undirected shortest paths via low hop emulators". In: *Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 322–335. arXiv: 1911.01956 (cit. on p. 6).

[BCMN+18] Luca Becchetti, Andrea Clementi, Pasin Manurangsi, Emanuele Natale, Francesco Pasquale, Prasad Raghavendra, and Luca Trevisan. "Average Whenever You Meet: Opportunistic Protocols for Community Detection". In: *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*. Ed. by Yossi Azar, Hannah Bast, and Grzegorz Herman. Vol. 112. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 7:1–7:13. DOI: 10.4230/LIPICS.ESA.2018.7. arXiv: 1703.05045. URL: https://doi.org/10.4230/LIPIcs.ESA.2018.7 (cit. on p. 8).

[BCNP+17] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, Prasad Raghavendra, and Luca Trevisan. "Friend or Foe? Population Protocols can perform Community Detection". In: *CoRR* abs/1703.05045 (2017). arXiv: 1703.05045. URL: http://arxiv.org/abs/1703.05045 (cit. on p. 8).

[BCNP+20] Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. "Find Your Place: Simple Distributed Algorithms for Community Detection". In: *SIAM J. Comput.* 49.4 (2020). Announced at SODA'17, pp. 821–864. DOI: 10.1137/19M1243026. arXiv: 1511.03927. URL: https://doi.org/10.1137/19M1243026 (cit. on pp. 4, 8).

[BFKL21] Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. "Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models". In: *SIAM J. Comput.* 50.3 (2021). Announced at DISC 2017, pp. 815–856. arXiv: 1607.05127 (cit. on p. 6).

[BH05] Andries E Brouwer and Willem H Haemers. "Eigenvalues and perfect matchings". In: *Linear Algebra and its Applications* 395 (2005), pp. 155–162 (cit. on p. 5).

[BJ09] Frank Bauer and Jürger Jost. "Bipartite and neighborhood graphs and the spectrum of the normalized graph Laplacian". In: *Communications in Analysis and Geometry* 21 (Oct. 2009). DOI: 10.4310/CAG.2013.v21.n4.a2 (cit. on p. 3).

[BLNP+20] Jan van den Brand, Yin Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. "Bipartite Matching in Nearly-linear Time on Moderately Dense Graphs". In: *Proc. of the 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 2020, pp. 919–930. arXiv: 2009.01802 (cit. on p. 6).

[BT15] Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015 (cit. on p. 8).

[BTT24] Tuğkan Batu, Amitabh Trehan, and Chhaya Trehan. "All you need are random walks: Fast and simple distributed conductance testing". In: *International Colloquium on Structural Information and Communication Complexity*. Springer. 2024, pp. 64–82 (cit. on pp. 5, 6).

[CFSV19] Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. "Fast distributed algorithms for testing graph properties". In: *Distributed Comput.* 32.1 (2019). Announced at DISC '16, pp. 41–57. DOI: 10.1007/S00446-018-0324-8. arXiv: 1602.03718. URL: https://doi.org/10.1007/s00446-018-0324-8 (cit. on p. 5).

[CGL20] Keren Censor-Hillel, François Le Gall, and Dean Leitersdorf. "On Distributed Listing of Cliques". In: *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. Ed. by Yuval Emek and Christian Cachin. ACM, 2020, pp. 474–482. DOI: 10.1145/3382734.3405742. arXiv: 2007.05316. URL: https://doi.org/10.1145/3382734.3405742 (cit. on p. 1).

[CGLP18] Flavio Chierichetti, George Giakkoupis, Silvio Lattanzi, and Alessandro Panconesi. "Rumor Spreading and Conductance". In: *J. ACM* 65.4 (2018), 17:1–17:21. DOI: 10.1145/3173043. URL: https://doi.org/10.1145/3173043 (cit. on pp. 3, 22).

[Chu97] Fan RK Chung. *Spectral graph theory*. Vol. 92. American Mathematical Soc., 1997 (cit. on pp. 10, 11, 18).

[CKGS23] Li Chen, Rasmus Kyng, Maximilian Probst Gutenberg, and Sushant Sachdeva. "A Simple Framework for Finding Balanced Sparse Cuts via APSP". In: *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*. Ed. by Telikepalli Kavitha and Kurt Mehlhorn. SIAM, 2023, pp. 42–55. DOI: 10.1137/1.9781611977585.CH5. URL: https://doi.org/10.1137/1.9781611977585.ch5 (cit. on p. 1).

[CKKR+06] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. "On the Hardness of Approximating Multicut and Sparsest-Cut". In: *Comput. Complex.* 15.2 (2006). Announced at CCC'05, pp. 94–114. DOI: 10.1007/S00037-006-0210-9. URL: https://doi.org/10.1007/s00037-006-0210-9 (cit. on pp. 1, 6).

[CMGS25] Daoyuan Chen, Simon Meierhans, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. "Parallel and Distributed Expander Decomposition: Simple, Fast, and Near-Optimal". In: *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*. Ed. by Yossi Azar and Debmalya Panigrahi. SIAM, 2025, pp. 1705–1719. DOI: 10.1137/1.9781611978322.53. arXiv: 2410.13451. URL: https://doi.org/10.1137/1.9781611978322.53 (cit. on pp. 1, 2, 5).

[CMSV17] Michael B. Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. "Negative-Weight Shortest Paths and Unit Capacity Minimum Cost Flow in $\tilde{O}(m^{10/7} \log W)$ Time (Extended Abstract)". In: *Proc. of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*. SIAM, 2017, pp. 752–771. arXiv: 1605.01717 (cit. on p. 6).

[CPZ19] Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. "Distributed Triangle Detection via Expander Decomposition". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. Ed. by Timothy M. Chan. SIAM, 2019, pp. 821–840. DOI: 10.1137/1.9781611975482.51. arXiv: 1807.06624. URL: https://doi.org/10.1137/1.9781611975482.51 (cit. on pp. 1, 5).

[CS19] Yi-Jun Chang and Thatchaphol Saranurak. "Improved Distributed Expander Decomposition and Nearly Optimal Triangle Enumeration". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 66–73. DOI: 10.1145/3293611.3331618. arXiv: 1904.08037. URL: https://doi.org/10.1145/3293611.3331618 (cit. on pp. 1, 5).

[CS20] Yi-Jun Chang and Thatchaphol Saranurak. "Deterministic Distributed Expander Decomposition and Routing with Applications in Distributed Derandomization". In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. Ed. by Sandy Irani. IEEE, 2020, pp. 377–388. DOI: 10.1109/FOCS46700.2020.00043. arXiv: 2007.14898. URL: https://doi.org/10.1109/FOCS46700.2020.00043 (cit. on pp. 1, 5).

[DEMN21] Michal Dory, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. "Distributed weighted min-cut in nearly-optimal time". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 1144–1153. DOI: 10.1145/3406325.3451020. arXiv: 2004.09129. URL: https://doi.org/10.1145/3406325.3451020 (cit. on pp. 1, 6).

[DHNS19] Mohit Daga, Monika Henzinger, Danupon Nanongkai, and Thatchaphol Saranurak. "Distributed edge connectivity in sublinear time". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Ed. by Moses Charikar and Edith Cohen. ACM, 2019, pp. 343–354. DOI: 10.1145/3313276.3316346. arXiv: 1904.04341. URL: https://doi.org/10.1145/3313276.3316346 (cit. on p. 1).

[DS15] Kinkar Das and Shaowei Sun. "Extremal graph on normalized Laplacian spectral radius and energy". In: *The Electronic Journal of Linear Algebra* 29 (2015), pp. 237–253 (cit. on p. 5).

[EFFG+17] Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. "Three Notes on Distributed Property Testing". In: *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. Ed. by Andréa W. Richa. Vol. 91. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 15:1–15:30. DOI: 10.4230/LIPICS.DISC.2017.15. URL: https://doi.org/10.4230/LIPIcs.DISC.2017.15 (cit. on p. 5).

[EFFK+22] Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. "Sublinear-time distributed algorithms for detecting small cliques and even cycles". In: *Distributed Comput.* 35.3 (2022). Announced at DISC '19, pp. 207–234. DOI: 10.1007/S00446-021-00409-3. URL: https://doi.org/10.1007/s00446-021-00409-3 (cit. on p. 1).

[FGLP+21] Sebastian Forster, Gramoz Goranci, Yang P. Liu, Richard Peng, Xiaorui Sun, and Mingquan Ye. "Minor Sparsifiers and the Distributed Laplacian Paradigm". In: *Proc. of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*. IEEE, 2021, pp. 989–999. arXiv: 2012.15675 (cit. on pp. 6, 19).

[FN10] Miroslav Fiedler and Vladimir Nikiforov. "Spectral radius and Hamiltonicity of graphs". In: *Linear Algebra and its Applications* 432.9 (2010), pp. 2170–2173 (cit. on p. 5).

[FV18] Hendrik Fichtenberger and Yadu Vasudev. "A two-sided error distributed property tester for conductance". In: *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2018, pp. 19–1 (cit. on pp. 5, 6).

[FV22] Sebastian Forster and Tijn de Vos. "The Laplacian Paradigm in the Broadcast Congested Clique". In: *Proc. of the ACM Symposium on Principles of Distributed Computing, PODC 2022*. ACM, 2022, pp. 335–344. arXiv: 2205.12059 (cit. on p. 6).

[FV23] Sebastian Forster and Tijn de Vos. "Brief Announcement: The Laplacian Paradigm in Deterministic Congested Clique". In: *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023*. Ed. by Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu. ACM, 2023, pp. 75–78. DOI: 10.1145/3583668.3594577. arXiv: 2304.02315. URL: https://doi.org/10.1145/3583668.3594577 (cit. on p. 6).

[Gia11] George Giakkoupis. "Tight bounds for rumor spreading in graphs of a given conductance". In: *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*. Ed. by Thomas Schwentick and Christoph Dürr. Vol. 9. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011, pp. 57–68. DOI: 10.4230/LIPICS.STACS.2011.57. URL: https://doi.org/10.4230/LIPIcs.STACS.2011.57 (cit. on p. 1).

[GKKL+18] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. "Near-Optimal Distributed Maximum Flow". In: *SIAM J. Comput.* 47.6 (2018). Announced at PODC 2015, pp. 2078–2117. arXiv: 1508.04747 (cit. on p. 6).

[GKS17] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. "Distributed MST and Routing in Almost Mixing Time". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*. Ed. by Elad Michael Schiller and Alexander A. Schwarzmann. ACM, 2017,

pp. 131–140. DOI: 10.1145/3087801.3087827. URL: https://doi.org/10.1145/3087801.3087827 (cit. on p. 1).

[GL18] Mohsen Ghaffari and Jason Li. "New Distributed Algorithms in Almost Mixing Time via Transformations from Parallel Algorithms". In: *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*. Ed. by Ulrich Schmid and Josef Widder. Vol. 121. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 31:1–31:16. DOI: 10.4230/LIPICS.DISC.2018.31. arXiv: 1805.04764. URL: https://doi.org/10.4230/LIPIcs.DISC.2018.31 (cit. on p. 1).

[GL22] Xiaofeng Gu and Muhuo Liu. "A tight lower bound on the matching number of graphs via Laplacian eigenvalues". In: *Eur. J. Comb.* 101 (2022), p. 103468. DOI: 10.1016/J.EJC.2021.103468. URL: https://doi.org/10.1016/j.ejc.2021.103468 (cit. on p. 5).

[GVL13] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013 (cit. on pp. 7, 8, 15, 31).

[Hae95] Willem H Haemers. "Interlacing eigenvalues and graphs". In: *Linear Algebra and its applications* 226 (1995), pp. 593–616 (cit. on p. 5).

[HJ12] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012 (cit. on p. 17).

[Hwa04] Suk-Geun Hwang. "Cauchy's interlace theorem for eigenvalues of Hermitian matrices". In: *The American mathematical monthly* 111.2 (2004), pp. 157–159 (cit. on p. 9).

[Kar00] David R. Karger. "Minimum cuts in near-linear time". In: *J. ACM* 47.1 (2000). Announced at STOC '96, pp. 46–76. DOI: 10.1145/331605.331608. URL: https://doi.org/10.1145/331605.331608 (cit. on pp. 1, 6).

[KLOS14] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. "An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and its Multicommodity Generalizations". In: *Proc. of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*. SIAM, 2014, pp. 217–226. arXiv: 1304.2338 (cit. on p. 6).

[KM08] David Kempe and Frank McSherry. "A decentralized algorithm for spectral analysis". In: *J. Comput. Syst. Sci.* 74.1 (2008). Announced at STOC'04, pp. 70–83. DOI: 10.1016/J.JCSS.2007.04.014. URL: https://doi.org/10.1016/j.jcss.2007.04.014 (cit. on p. 4).

[KM15] Fabian Kuhn and Anisur Rahaman Molla. "Distributed Sparse Cut Approximation". In: *19th International Conference on Principles of Distributed Systems, OPODIS 2015, December 14-17, 2015, Rennes, France*. Ed. by Emmanuelle Anceaume, Christian Cachin, and Maria Gradinariu Potop-Butucaru. Vol. 46. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, 10:1–10:14. DOI: 10.4230/LIPICS.OPODIS.2015.10. URL: https://doi.org/10.4230/LIPIcs.OPODIS.2015.10 (cit. on pp. 1, 2, 5).

[KX16] Ioannis Koutis and Shen Chen Xu. "Simple Parallel and Distributed Algorithms for Spectral Graph Sparsification". In: *ACM Trans. Parallel Comput.* 3.2 (2016), 14:1–14:14. DOI: 10.1145/2948062 (cit. on pp. 4, 6).

[LC10] Bolian Liu and Siyuan Chen. "Algebraic conditions for t-tough graphs". In: *Czechoslovak mathematical journal* 60.4 (2010), pp. 1079–1089 (cit. on p. 5).

[LGT12] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. "Multi-way spectral partitioning and higher-order cheeger inequalities". In: *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*. Ed. by Howard J. Karloff and Toniann Pitassi. ACM, 2012, pp. 1117–1130. DOI: 10.1145/2213977.2214078. arXiv: 1111.1055. URL: https://doi.org/10.1145/2213977.2214078 (cit. on pp. 2, 3, 10).

[Li20] Jason Li. "Faster parallel algorithm for approximate shortest path". In: *Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 308–321. arXiv: 1911.01626 (cit. on p. 6).

[Liu15] Shiping Liu. "Multi-way dual Cheeger constants and spectral bounds of graphs". In: *Advances in Mathematics* 268 (2015), pp. 306–338. arXiv: 1401.3147 (cit. on p. 3).

[LMR18] Reut Levi, Moti Medina, and Dana Ron. "Property testing of planarity in the CONGEST model". In: *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. 2018, pp. 347–356 (cit. on p. 5).

[LS20a] Yang P. Liu and Aaron Sidford. "Faster Divergence Maximization for Faster Maximum Flow". In: *CoRR* abs/2003.08929 (2020). arXiv: 2003.08929. URL: https://arxiv.org/abs/2003.08929 (cit. on p. 6).

[LS20b] Yang P. Liu and Aaron Sidford. "Faster energy maximization for faster maximum flow". In: *Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 803–814. arXiv: 1910.14276 (cit. on p. 6).

[Mą13] Aleksander Mądry. "Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back". In: *Proc. of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*. IEEE Computer Society, 2013, pp. 253–262. arXiv: 1307.2205 (cit. on p. 6).

[Mą16] Aleksander Mądry. "Computing Maximum Flow with Augmenting Electrical Flows". In: *Proc. of the IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE Computer Society, 2016, pp. 593–602. arXiv: 1608.06016 (cit. on p. 6).

[MMM18] Frederik Mallmann-Trenn, Cameron Musco, and Christopher Musco. "Eigenvector Computation and Community Detection in Asynchronous Gossip Models". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 159:1–159:14. DOI: 10.4230/LIPICS.ICALP.2018.159. arXiv: 1804.08548. URL: https://doi.org/10.4230/LIPIcs.ICALP.2018.159 (cit. on p. 8).

[MS90] David W. Matula and Farhad Shahrokhi. "Sparsest cuts and bottlenecks in graphs". In: *Discret. Appl. Math.* 27.1-2 (1990), pp. 113–123. DOI: 10.1016/0166-218X(90)90133-W. URL: https://doi.org/10.1016/0166-218X(90)90133-W (cit. on pp. 1, 6).

[Pen16] Richard Peng. "Approximate Undirected Maximum Flows in $O(m\text{polylog}(n))$ Time". In: *Proc. of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*. SIAM, 2016, pp. 1862–1867 (cit. on p. 6).

[PS14] Federico Penna and Sławomir Stańczak. "Decentralized eigenvalue algorithms for distributed signal detection in wireless networks". In: *IEEE Transactions on Signal Processing* 63.2 (2014), pp. 427–440 (cit. on p. 8).

[SGP09] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. "Sparse Cut Projections in Graph Streams". In: *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*. Ed. by Amos Fiat and Peter Sanders. Vol. 5757. Lecture Notes in Computer Science. Springer, 2009, pp. 480–491. DOI: 10.1007/978-3-642-04128-0_43. URL: https://doi.org/10.1007/978-3-642-04128-0_43 (cit. on p. 1).

[She13] Jonah Sherman. "Nearly Maximum Flows in Nearly Linear Time". In: *Porc. of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*. IEEE Computer Society, 2013, pp. 263–269. arXiv: 1304.2077 (cit. on p. 6).

[SHKK+12] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. "Distributed Verification and Hardness of Distributed Approximation". In: *SIAM J. Comput.* 41.5 (2012). Announced at STOC '11, pp. 1235–1265. DOI: 10.1137/11085178X. arXiv: 1011.3049. URL: https://doi.org/10.1137/11085178X (cit. on p. 6).

[SJ89] Alistair Sinclair and Mark Jerrum. "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains". In: *Inf. Comput.* 82.1 (1989), pp. 93–133. DOI: 10.1016/0890-5401(89)90067-9. URL: https://doi.org/10.1016/0890-5401(89)90067-9 (cit. on pp. 2, 10).

[SMP15] Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. "Distributed Computation of Sparse Cuts via Random Walks". In: *Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN 2015, Goa, India, January 4-7, 2015*. Ed. by Sajal K. Das, Dilip Krishnaswamy, Santonu Karkar, Amos Korman, Mohan J. Kumar, Marius Portmann, and Srikanth Sastry. ACM, 2015, 6:1–6:10. DOI: 10.1145/2684464.2684474. arXiv: 1310.5407. URL: https://doi.org/10.1145/2684464.2684474 (cit. on pp. 2, 3).

[SMPU15] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. "Fast distributed PageRank computation". In: *Theor. Comput. Sci.* 561 (2015). Announced at ICDCN'13, pp. 113–121. DOI: 10.1016/J.TCS.2014.04.003. arXiv: 1208.3071. URL: https://doi.org/10.1016/j.tcs.2014.04.003 (cit. on pp. 1, 8).

[SNPT13] Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. "Distributed Random Walks". In: *J. ACM* 60.1 (2013), 2:1–2:31. DOI: 10.1145/2432622.2432624. arXiv: 1302.4544. URL: https://doi.org/10.1145/2432622.2432624 (cit. on p. 1).

[ST04] Daniel A. Spielman and Shang-Hua Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Proc. of the 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*. ACM, 2004, pp. 81–90. arXiv: cs/0310051 (cit. on pp. 1, 6).

[SW19] Thatchaphol Saranurak and Di Wang. "Expander Decomposition and Pruning: Faster, Stronger, and Simpler". In: *SODA*. SIAM, 2019, pp. 2616–2635 (cit. on pp. 1, 2).

[Tre15] Luca Trevisan. *Graph Partitioning and Expanders (Lecture 3: The Power Method)*. 2015. URL: https://theory.stanford.edu/~trevisan/expander-online/lecture03.pdf (cit. on p. 31).

[Vos23] Tijn de Vos. “Minimum Cost Flow in the CONGEST Model”. In: *Structural Information and Communication Complexity - 30th International Colloquium, SIROCCO 2023, Alcalá de Henares, Spain, June 6-9, 2023, Proceedings*. Ed. by Sergio Rajsbaum, Alkida Balliu, Joshua J. Daymude, and Dennis Olivetti. Vol. 13892. Lecture Notes in Computer Science. Announced at PODC'23. Springer, 2023, pp. 406–426. DOI: 10.1007/978-3-031-32733-9_18. arXiv: 2304.01600. URL: https://doi.org/10.1007/978-3-031-32733-9_18 (cit. on p. 6).

[Wil67] Herbert S Wilf. “The eigenvalues of a graph and its chromatic number”. In: *Journal of the London mathematical Society* 1.1 (1967), pp. 330–332 (cit. on p. 5).

A The Power Method

The power method is a standard technique in numerical linear algebra, see, e.g., [GVL13]. Although it works for any type of matrix, both its statement and analyses are simpler for positive semi-definite instances. Since this suffices fo our purposes, we restrict to this case. In Section A.1, we prove the known Lemma 7, following [Tre15]. In Section A.2 we prove the bit precision lemma (Lemma 11) required for an efficient CONGEST implementation.

A.1 The Power Method (Proof of Lemma 7)

Algorithm. The power method starts with initial vector $x_0 \sim \{-1,1\}^n$, chosen uniformly at random vector. We then compute $x_k := M^k x_0$. The goal is to show that the Rayleigh coefficient of x_k approximates the largest eigenvalue μ_n well:

$$(1-\varepsilon)\mu_n \leq \frac{x_k^T M x_k}{x_k^T x_k} \leq \mu_n.$$

Start vector. Notably, the start vector x has to have a significant inner product with the largest eigenvector v_n. Choosing x uniformly at random ensures that this is true with constant probability. For this proof, we use the Paley-Zygmund Inequality.

Lemma 22 (Paley-Zygmund Inequality)**.** *Let $X \geq 0$ be a random variable with finite variance. For all $0 \leq \delta \leq 1$ holds that*

$$\mathbb{P}[X \geq \delta \cdot \mathbb{E}(X)] \geq (1-\delta)^2 \cdot \frac{\mathbb{E}(X)^2}{\mathbb{E}(X^2)}.$$

The power method fails to work if the random start vector is orthogonal to the eigenvector of the largest eigenvalue. The following lemma states that it is unlikely to happen.

Lemma 23. *Let $x \sim \{-1,1\}^n$ be a uniformly random vector and $y \in \mathbb{R}^n$ a vector with $\|y\| = 1$. Then we have*

$$\mathbb{P}\left[|\langle x, y\rangle| \geq \tfrac{1}{2}\right] \geq \tfrac{3}{16}.$$

Proof. Let $X = \sum_{i\in[n]} x_i y_i$ be the random variable defined by the inner product $\langle x, y\rangle$. As X is not necessarily non-negative, we base our calculation on X^2 which is always non-negative. First, we consider the second moment of X.

$$\begin{aligned}\mathbb{E}(X^2) &= \mathbb{E}\left(\left(\sum_{i\in[n]} x_i y_i\right)^2\right) = \sum_{i,j\in[n]} y_i y_j \mathbb{E}(x_i x_j) = \sum_{i\in[n]} y_i^2 \mathbb{E}(x_i^2) + \sum_{i,j\in[n],i\neq j} y_i y_j \mathbb{E}(x_i x_j) \\ &= \sum_{i\in[n]} y_i^2 + \sum_{i,j\in[n],i\neq j} y_i y_j = \sum_{i\in[n]} y_i^2 = 1\end{aligned}$$

In the fourth step, we apply $\mathbb{E}(x_i x_j) = 0$ and $\mathbb{E}(x_i^2) = 1$. The last equality follows from the fact that $\|y\| = 1$. Next, consider the fourth moment of X.

$$\begin{aligned}\mathbb{E}(X^4) &= 3 \cdot \sum_{i,j\in[n],i\neq j} \mathbb{E}(x_i^2) y_i^2 \cdot \mathbb{E}(x_j^2) y_j^2 + \sum_{i\in[n]} \mathbb{E}(x_i^4) y_i^4 = 3 \cdot \sum_{i,j\in[n]} y_i^2 y_j^2 - 3 \cdot \sum_{i\in[n]} y_i^4 + \sum_{i\in[n]} y_i^4 \\ &= 3 \cdot \left(\sum_{i\in[n]} y_i^2\right)^2 - 2 \cdot \sum_{i\in[n]} y_i^4 = 3 - 2 \cdot \sum_{i\in[n]} y_i^4 \leq 3.\end{aligned}$$

For the sake of simplicity, the first step leaves out any summations that contain $\mathbb{E}(x_i)$ as all of them are equal to zero. In the second step, we use the fact that $\mathbb{E}(x_i^2) = 1 = \mathbb{E}(x_i^4)$.

Now, we are in the position to apply the Paley-Zygmund inequality. Let $\delta = 1/4$. As X^2 is a non-negative random variable with finite variance, it follows from Lemma 22 that

$$\mathbb{P}[X^2 \geq \delta \cdot \mathbb{E}(X^2)] = \mathbb{P}[X^2 \geq \tfrac{1}{4}] \geq (1 - \tfrac{1}{4})^2 \cdot \frac{\mathbb{E}(X^2)^2}{\mathbb{E}(X^4)} = (\tfrac{3}{4})^2 \cdot \tfrac{1}{3} = \tfrac{3}{16}. \qquad \square$$

Lower Bound. Lemma 7 states that the Rayleigh coefficient output by the power method is lower bounded by $(1-\varepsilon)\mu_n$. Next, we prove a slightly weaker lower bound; the original lower bound is obtained in the proof of Lemma 7 by adjusting ε.

Lemma 24. *Let $x_0 \in \mathbb{R}^n$ and set $x_k = M^k x$. For any $\varepsilon > 0$ and $k \geq \log n/\varepsilon$, we have*

$$\frac{x_k^T M x_k}{x_k^T x_k} \geq (1-\varepsilon)\mu_n \cdot \frac{1}{1 + \frac{\|x\|^2}{\langle x_0, v_n\rangle^2}(1-\varepsilon)^{2k}}. \tag{2}$$

Proof. We start by dividing the eigenvalues $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$ of M in two groups. Let $j \in [n]$ be the first index for which $(1-\varepsilon)\mu_n \leq \mu_j$ holds, then we have $\mu_1, \ldots, \mu_{j-1} < (1-\varepsilon)\mu_n$ and $(1-\varepsilon)\mu_n \leq \mu_j, \ldots, \mu_n$. As a direct consequence, it holds that

$$\sum_{i \in [j-1]} \mu_i^{2k} \leq n(1-\varepsilon)^{2k}\mu_n^{2k} \leq \frac{\mu_n^{2k}}{n}.$$

Intuitively, the contribution of the small eigenvalue group to the spectral decomposition of M^{2k+1} is insignificant for large k. With this insight, we use the spectral decomposition of M^{2k+1} to derive a lower bound for the nominator

$$\begin{aligned}
x_k^T M x_k = x_0^T M^{2k+1} x_0 &= x_0^T \left(\sum_{i \in [n]} \mu_i^{2k+1} v_i v_i^T \right) x_0 \\
&= \sum_{i \in [n]} \mu_i^{2k+1} \langle x_0, v_i \rangle^2 \\
&\geq \sum_{i \in \{j, \ldots, n\}} \mu_i^{2k+1} \langle x_0, v_i \rangle^2 \\
&\geq (1-\varepsilon)\mu_n \sum_{i \in \{j, \ldots, n\}} \mu_i^{2k} \langle x_0, v_i \rangle^2.
\end{aligned}$$

Second, we derive an upper bound for the denominator

$$\begin{aligned}
x_k^T x_k = x_0^T M^{2k} x_0 &= \sum_{i \in [n]} \mu_i^{2k} \langle x_0, v_i \rangle^2 \\
&= \sum_{i \in \{j, \ldots, n\}} \mu_i^{2k} \langle x_0, v_i \rangle^2 + \sum_{i \in \{1, \ldots, j-1\}} \mu_i^{2k} \langle x_0, v_i \rangle^2 \\
&\leq \sum_{i \in \{j, \ldots, n\}} \mu_i^{2k} \langle x_0, v_i \rangle^2 + (1-\varepsilon)^{2k}\mu_n^{2k} \sum_{i \in \{1, \ldots, j-1\}} \langle x_0, v_i \rangle^2 \\
&\leq \sum_{i \in \{j, \ldots, n\}} \mu_i^{2k} \langle x_0, v_i \rangle^2 + (1-\varepsilon)^{2k}\mu_n^{2k} \|x_0\|^2,
\end{aligned}$$

where the last inequality follows from $\sum_{i\in[j-1]}\langle x_0, v_i\rangle^2 \le \|x_0\|^2$. Using both upper and lower bound, we can bound the ratio as follows

$$\begin{aligned}\frac{x_k^T M x_k}{x_k^T x_k} = \frac{x_0^T M^{2k+1} x_0}{x_0^T M^{2k} x_0} &\ge \frac{(1-\varepsilon)\mu_n \sum_{i\in\{j,\dots,n\}} \mu_i^{2k}\langle x_0, v_i\rangle^2}{(1-\varepsilon)^{2k}\mu_n^{2k}\|x_0\|^2 + \sum_{i\in\{j,\dots,n\}} \mu_i^{2k}\langle x_0, v_i\rangle^2} \\ &= \frac{(1-\varepsilon)\mu_n}{1 + \frac{(1-\varepsilon)^{2k}\mu_n^{2k}\|x_0\|^2}{\sum_{i\in\{j,\dots,n\}} \mu_i^{2k}\langle x_0, v_i\rangle^2}} \\ &\ge \frac{(1-\varepsilon)\mu_n}{1 + \frac{(1-\varepsilon)^{2k}\mu_n^{2k}\|x_0\|^2}{\mu_n^{2k}\langle x_0, v_n\rangle^2}} \\ &= \frac{(1-\varepsilon)\mu_n}{1 + (1-\varepsilon)^{2k}\frac{\|x_0\|^2}{\langle x_0, v_n\rangle^2}}.\end{aligned}$$

□

Upper Bound. Next, we show that we cannot overshoot, i.e, that the output of the power method is upper bounded by μ_n.

Lemma 10. *For any vector $x \in \mathbb{R}^k$ we have that $\frac{x^T M x}{x^T x} \le \mu_n$.*

Proof. We decompose x into the eigenvectors, i.e., let $\alpha_j \in \mathbb{R}$ be $\alpha_j := \langle x, v_j\rangle$, and hence $x = \sum_{j\in[n]} \alpha_j v_j$. Now we observe

$$\begin{aligned}x^T M x &= \left(\sum_{j\in[n]} \alpha_j v_j\right)^T \left(M\left(\sum_{j\in[n]} \alpha_j v_j\right)\right) = \sum_{j\in[n]} \alpha_j^2 \mu_j \\ &\le \sum_{j\in[n]} \left(\max_{k\in[n]} \mu_k\right) \alpha_j^2 = \mu_n \sum_{j\in[n]} \alpha_j^2 = \mu_n \left(\sum_{j\in[n]} \alpha_j v_j\right)^T \left(\sum_{j\in[n]} \alpha_j v_j\right) = \mu_n x^T x.\end{aligned}$$

□

Conclusion. We conclude by putting the pieces together.

Lemma 7 (Power Method)**.** *Let M be a positive semi-definite matrix with eigenvalues $0 \le \mu_1 \le \cdots \le \mu_n$. Let $x_0 \sim \{-1, 1\}^n$ be a uniformly random vector and $x_k = M^k x_0$. For $\varepsilon \ge 16/n$ and $k \ge \Theta(\log n/\varepsilon)$, it holds with constant probability that*

$$(1-\varepsilon)\mu_n \le \frac{x_k^T M x_k}{x_k^T x_k} \le \mu_n.$$

Proof. Due to Lemma 23 and $\|x\|^2 = n$, we have $\|x\|^2/\langle x, v_1\rangle^2 \le 4n$. Thus, the right denominator in Equation (2) can be bounded by

$$1 + \frac{\|x\|^2}{\langle x, v_1\rangle^2}(1-\varepsilon)^{2k} \le 1 + 4n \cdot e^{-2\log n} \le 1 + \tfrac{4}{n}.$$

For any $\varepsilon \ge 8/n$ and $\varepsilon' = 2\varepsilon - \varepsilon^2$, Equation (2) simplifies to

$$\frac{x_k^T M x_k}{x_k^T x_k} \ge (1-\varepsilon)\mu_n \cdot \frac{1}{1+\frac{4}{n}} \ge (1-\varepsilon)\mu_n \cdot (1 - \tfrac{8}{n}) \ge (1-\varepsilon')\mu_n.$$

By Lemma 10, we get the upper bound. □

A.2 Bit Precision Lemma (Proof of Lemma 11)

The following lemma shows that truncating or rounding after the $\Theta(\log n)$-th bit only impact the outcome by an inverse polynomial factor. This means we can round up to $\text{poly}(n)$ times without any issues. In particular, we can round in every step of the algorithm with a negligibly small effect on the outcome.

Lemma 11. *Let $k \geq i \geq 1$. Throughout the power method, rounding each element of $x_i := M^i x_0$ to the $\Theta(\log n)$-th bit, does not impact the outcome of $\frac{(x_k)^T M x_k}{(x_k)^T x_k}$ by more than a factor $(1 \pm n^{-\Theta(1)})$.*

Proof. The power method starts with $x_0 \sim \{+1, -1\}$ uniformly at random, so $||x_0|| = \sqrt{n}$, and with constant probability, Lemma 23, we have $|\langle x_0, v_n \rangle| \geq \frac{3}{16}$. In which case the power method gives a good approximation. We investigate the numerical stability of this case.

We need to compare before and after rounding for

$$\frac{(x_k)^T M x_k}{(x_k)^T x_k} = \frac{x_0^T M^{2k+1} x_0}{x_0 M^{2k} x_0} = \frac{x_i^T M^{2l+1} x_i}{x_i M^{2l} x_i},$$

where $l := k - i$. Denote the difference occurred by rounding as y, i.e., after rounding we have $x_i + y$ and since we round after $\Theta(\log n)$ bits, we have $|(y)_v| \leq \delta |(x_i)_v|$ for some $\delta = n^{-\Theta(1)}$. In particular $||y|| \leq \delta ||x_i||$.

Now, let use make some observations. First of all,

$$|\langle x_i, v_n \rangle| \geq \mu_n^i |\langle x_0, v_n \rangle| \geq \mu_n^i \left(\frac{3}{16}\right)^2.$$
$$||x_i|| \leq \mu_n^i ||x_0|| = \mu_n^i \sqrt{n}.$$

Next, we use this to bound our main inner product:

$$x_i^T M^{2l+1} x_i \geq \mu_n^{2l+1} \langle x_i, v_n \rangle^2 \geq \mu_n^{2k+1} \cdot \frac{3}{16}.$$

Or equivalently, $\mu_n^{2k+1} \leq \Theta(1) x_i^T M^{2l+1} x_i$. Next, we use Cauchy-Schwarz to express the incurred error in terms of this inner product.

$$\begin{aligned}
|y^T M^{2l+1} x_i| &\leq ||y|| \cdot ||M^{2l+1} x_i|| \leq \delta ||x_i|| \mu_n^{2l+1} \leq \delta \mu^{2k+1} \sqrt{n} \leq \Theta(\sqrt{n}) x_i^T M^{2l+1} x_i \\
|x_i^T M^{2l+1} y| &\leq ||M^{2l+1} x_i|| \cdot ||y|| \leq \delta ||x_i|| \mu_n^{2l+1} \leq \delta \mu^{2k+1} \sqrt{n} \leq \Theta(\sqrt{n}) x_i^T M^{2l+1} x_i \\
|y^T M^{2l+1} y| &\leq ||y||^2 \mu_n^{2l+1} \leq \delta^2 ||x_i|| \mu_n^{2l+1} \leq \delta^2 \mu_n^{2k+1} \sqrt{n} \leq \delta \mu_n^{2k+1} \leq \Theta(\sqrt{n}) x_i^T M^{2l+1} x_i,
\end{aligned}$$

Now, we are ready to consider $\frac{(x_i+y)^T M^{2l+1} (x_i+y)}{(x_i+y) M^{2l} (x_i+y)}$. We do this in two steps: numerator and denominator. First we note for the numerator that

$$\begin{aligned}
(x_i + y)^T M^{2l+1} (x_i + y) &\geq x_i^T M^{2l+1} x_i - |y^T M^{2l+1} x_i| - |x_i^T M^{2l+1} y| - |y^T M^{2l+1} y| \\
&\geq (1 - 3\delta\Theta(\sqrt{n})) x_i^T M^{2l+1} x_i.
\end{aligned}$$

For the denominator, we obtain via similar arguments that

$$(x_i + y) M^{2l} (x_i + y) \leq (1 + 3\delta\Theta(\sqrt{n})) x_i M^{2l} x_i.$$

So in total we have

$$\frac{(x_i + y)^T M^{2l+1} (x_i + y)}{(x_i + y) M^{2l} (x_i + y)} \geq \frac{1 - 3\delta\Theta(\sqrt{n})}{1 + 3\delta\Theta(\sqrt{n})} \cdot \frac{x_i^T M^{2l+1} x_i}{x_i M^{2l} x_i}.$$

For $\delta \leq n^{-\Theta(1)}$, we obtain

$$\frac{(x_i+y)^T M^{2l+1}(x_i+y)}{(x_i+y)M^{2l}(x_i+y)} \geq (1-n^{-\Theta(1)}) \cdot \frac{x_i^T M^{2l+1} x_i}{x_i M^{2l} x_i}.$$

Similar arguments give the upper bound with $(1+n^{-\Theta(1)})$. □