All You Need are Random Walks: Fast and Simple Distributed Conductance Testing

Tuğkan Batu[1][0000−0003−3914−4645], Amitabh Trehan[2][0000−0002−2998−0933], and Chhaya Trehan[1,3][0000−0002−3249−3212]

[1] London School of Economics and Political Science, U.K.
t.batu@lse.ac.uk, chhaya.dhingra@gmail.com
[2] Durham University, U.K. amitabh.trehan@durham.ac.uk
[3] University of Bristol, U.K.

Abstract. We propose a simple and time-optimal algorithm for property testing a graph for its conductance in the CONGEST model. Our algorithm takes only $O(\log n)$ rounds of communication (which is known to be optimal), and consists of simply running multiple random walks of $O(\log n)$ length from a certain number of random sources, at the end of which nodes can decide if the underlying network is a good conductor or far from it. Unlike previous algorithms, no aggregation is required even with a smaller number of walks. Our main technical contribution involves a tight analysis of this process for which we use spectral graph theory. We introduce and leverage the concept of sticky vertices which are vertices in a graph with low conductance such that short random walks originating from these vertices end in a region around them.
The present state-of-the-art distributed CONGEST algorithm for the problem by Fichtenberger and Vasudev [MFCS 2018], runs in $O(\log n)$ rounds using three distinct phases : building a rooted spanning tree (*preprocessing*), running $O(n^{100})$ random walks to generate statistics (*Phase 1*), and then convergecasting to the root to make the decision (*Phase 2*). The whole of our algorithm is, however, similar to their Phase 1 running only $O(m^2) = O(n^4)$ walks. Note that aggregation (using spanning trees) is a popular technique but spanning tree(s) are sensitive to node/edge/root failures, hence, we hope our work points to other more distributed, efficient and robust solutions for suitable problems.

Keywords: Graph Conductance · Distributed Property Testing · Random Walks.

1 Introduction

Checking whether a distributed network satisfies a certain property is an important problem. For example, this knowledge may be used to choose appropriate algorithms to be run on the network for certain tasks. For instance, the randomised leader election algorithm of [20] works in sublinear time if the underlying graph is a good expander but not otherwise. However, it may be hard

to efficiently *verify* certain global graph properties in the CONGEST model of distributed computing. In this model, each vertex of the input graph acts as a processing unit and works in conjunction with other vertices to solve a computational problem. The computation proceeds in synchronous rounds, in each of which every vertex can send an $O(\log n)$-bits message to each of its neighbours, do some local computations and receive messages from its neighbours.

Distributed decision problems are tasks in which the vertices of the underlying network have to collectively decide whether the network satisfies a global property $\mathcal{P}$ or not. If the network indeed satisfies the property, then all vertices must accept and, if not, then at least one vertex in the network must reject. For many global properties, lower bounds on the number of rounds of computation of the form $\tilde{\Omega}(\sqrt{n} + D)$ are known for distributed decision, where n is the number of vertices and D is the diameter of the network. (See [8]). It makes sense to relax the decision question and settle for an approximate answer in these scenarios as is done in the field of property testing (see [14,17]) in the sequential setting. A property testing algorithm in the sequential setting arrives at an approximate decision about a certain property of the input by *querying* only a small portion of it. Specifically, an ϵ-tester for a graph property $\mathcal{P}$ is a randomised algorithm that can distinguish between graphs that satisfy $\mathcal{P}$ and the graphs that are ϵ-far from satisfying $\mathcal{P}$ with high constant probability. An m-edge graph G is considered ϵ-far from satisfying $\mathcal{P}$ if one has to modify (add or delete) more than $\epsilon \cdot m$ edges of G for it to satisfy $\mathcal{P}$. Two-sided error testers may err on all graphs, while one-sided error testers have to present a witness when rejecting a graph. The cost of the algorithm is measured in the number of queries made. (See [14,17,16,15] for a detailed exposition of the subject.)

Distributed Property Testing: A *distributed property testing* problem is a relaxed variant of the corresponding decision problem: if the input network satisfies a property, then, with sufficiently high probability, all the vertices accept but if the input network is ϵ-far from satisfying the property, then at least one vertex rejects. The definition of "farness" in it remains the same as in the classical setting. The complexity measures are the number of communication rounds and the number of messages exchanged during the execution of the tester. Distributed property testing has been an active area of research recently. The work of [4] was the first to present a distributed algorithm (for finding near-cliques) with a property testing flavour. Later, in [5], the authors did a more detailed study of distributed property testing. There has been further study on the topic (see [10] and [12]) in the specific context of subgraph freeness.

Conductance Testing: We address the problem of testing the conductance of an unweighted, undirected graph $G = (V, E)$ in the CONGEST model. Throughout, we denote $|V|$ by n and $|E|$ by m. A distributed conductance tester can be a useful pre-processing step for some distributed algorithms (such as [20]) which perform better on graphs with high conductance. The test can help determine whether to proceed with the algorithm or not.

Given a graph $G = (V, E)$, and a set $A \subseteq V$, the *volume* of A (denoted $\mathrm{vol}(A)$) is the sum of degrees of vertices in A. We say that a graph $G = (V, E)$ is an

α–*conductor* if every $U \subseteq V$ such that $\text{vol}(U) \leq \text{vol}(V)/2$ has conductance at least α. Here, the conductance of a set A is defined as $E(A, V \setminus A)/\text{vol}(A)$, where $E(A, V \setminus A)$ is the number of edges crossing between A and $V \setminus A$. A closely related property of graphs is their expansion. We call a graph $G = (V, E)$ an α–*vertex expander* if every $U \subseteq V$ such that $|U| \leq |V|/2$ has at least $\alpha|U|$ neighbours. Here, a vertex $v \in V \setminus U$ is a neighbour of U if it has at least one edge incident to some $u \in U$. Similarly, G is called an α–*edge expander*, if every $U \subseteq V$ such that $|U| \leq |V|/2$ has at least $\alpha|U|$ edges crossing between U and $V \setminus U$. For a constant d, a graph $G = (V, E)$ is called a *bounded-degree graph* with degree bound d if every $v \in V$ has degree at most d. In this case, both the vertex and edge expansions are bounded by a constant (depending on d) times the conductance. Testing expansion (essentially testing conductance) in the bounded degree model has been studied for a long time in the classic centralised property testing model. In this setting, the problem of testing expansion was first studied by Goldreich and Ron [18] and later by [7]. Specifically, Czumaj and Sohler showed that given parameters $\alpha, \epsilon > 0$, the tester proposed by [18] accepts all graphs with vertex expansion larger than α, and rejects all graphs that are ϵ-far from having vertex expansion at least $\alpha' = \Theta(\alpha^2/\log n)$. Their work was followed by the state of the art results by [19] and [25]. Both these papers present $\tilde{O}(n^{1/2+\mu}/\alpha^2)$-query testers (for a small constant $\mu > 0$) for distinguishing between graphs that have expansion at least α and graphs that are ϵ-far from having expansion at least $\Omega(\alpha^2)$. In the general graph model (with no bound on degrees), Li, Pan and Peng [21] presented a conductance tester. Their tester essentially pre-processes the graph and turns it into a bounded degree graph while preserving (roughly) its expansion and size and then uses a tester for bounded degree graphs.

In the last few years, the same problem has also been addressed in non-sequential models of computing such as MPC [23] and distributed CONGEST [11]. There are earlier papers studying distributed random walks whose results can be adapted towards conductance testing e.g. [26,24] . However, these results yield large gaps in the conductance of the graphs that are accepted ($\Omega(\alpha)$) and that of the graphs that are rejected ($O(\alpha^2/\text{polylog}(n))$). The first distributed algorithm for the specific task of testing the conductance of an input graph that we are aware of is by Fichtenberger and Vasudev [11]. This can test the conductance of the input network in the unbounded-degree graph model (like ours).

A typical algorithm for the problem in the sequential, as well as non-sequential, models can be thought of as running in two *phases* (after possibly a *pre-processing phase*). In the first phase, the algorithm performs a certain number of short ($O(\log n)$-length) random walks from a randomly chosen starting vertex. The walks should mix well on a graph with high conductance and should take longer to mix on a graph which is far from having high conductance (at least from some fraction of starting vertices). In the second phase, the algorithm then checks whether those walks mixed well or not. For that, the algorithm gleans some information from every vertex in the graph and computes some aggregate func-

tion. Specifically in the classical and MPC settings, the algorithms count the total number of pairwise collisions between the endpoints of the walks.

The distributed algorithm for the problem by Fichtenberger and Vasudev [11] precedes the first phase by building a rooted BFS spanning tree of the input graph.[4] This spanning tree is used for collecting information from the endpoints of the random walks in the second phase. Specifically, their algorithm estimates the discrepancy of the endpoint probability distribution from the stationary distribution by collecting the estimate of discrepancy on each endpoint at a central point. If the overall discrepancy is above a certain threshold, the algorithm rejects the graph. This process of building a spanning tree and collecting information at the root to decide if the property holds or not takes a global and centralised view of the testing process.

The following natural question arises in the context of the second phase:

Question 1. Is it possible to execute the second phase without computing a global aggregate function?

In the classic setting, one strives for testers that make a sublinear (in n or m) number of queries which translates to running a sublinear number of walks. With only a sublinear ($O(\sqrt{n})$) number of walks, one hardly expects to see any useful information by itself on any individual vertex or in a small constant neighbourhood around it to know if the walks mixed well or not. Therefore, one has to rely on an aggregate function such as the total number of pairwise collisions between the endpoints of the walks. In the non-sequential settings such as distributed CONGEST, one can utilise the parallelism to run a superlinear number of short walks while keeping the run time proportional to the length of the walks. This inspires us to stick to Question 1 in distributed setting and investigate what information one should store at each vertex during phase 1 and how it should be processed *locally* to allow each node to decide locally whether it is part of a good conductor or not. This leads us to our main result provided all the nodes know n and m beforehand. Note that one can overcome the requirement of knowing m by performing a rooted spanning tree construction as in [11] and using this tree to count the number of edges. Note that we will not use this tree for collecting information about the random walks.

Our Results: Our main result is the algorithm presented in Section 3 (Pseudocodes 1 and 2). The main theorem is restated below:

Theorem (Theorem 3). *For an input graph $G = (V, E)$, and parameters $0 < \alpha < 1$ and $\epsilon > 0$, the distributed algorithm described in Section 3*

- *outputs Accept, with probability at least $2/3$, on every vertex of G if G is an α-conductor.*

[4] If the construction of BFS tree takes longer than $O(\log n)$ rounds the algorithm rejects without proceeding to the first phase since all good conductors have small diameter. However, a bad conductor such as a dumbbell graph may also have small diameter, so their algorithm still needs to proceed with the test after the successful construction of the spanning tree.

– *outputs Reject, with probability at least* $2/3$*, on at least one vertex of* G *if* G *is* ϵ*-far from any* $(\alpha^2/2880)$*-conductor.*

The algorithm uses $O_{\epsilon,\alpha}(\log n)$ *communication rounds.*

To be precise, the algorithm runs $2m^2$ walks of length $\frac{32}{\alpha^2}\log n$ from each of $\theta(1/\epsilon)$ starting vertices with the number of communication rounds equal to length of each random walk i.e. $\frac{32}{\alpha^2}\log n$ and messages at most $O(m \log n)$. A lower bound theorem in [11] (Theorem 2) states that any distributed tester with this gap requires $\Omega(\log(n+m))$ rounds of communication even in LOCAL model, hence, our running time is optimal.

Testing in a single phase: The advantage of not having to collect global information is that it lets us do away with the wasteful construction of a spanning tree and information accumulation at the root. Since we do not need to construct a spanning tree, we do not need a pre-processing phase unlike [11].

Note that setting up a spanning tree creates multiple points of failures for the aggregation phase. One could attempt to handle failure of the root of a single tree by setting up multiple spanning trees simultaneously. However, note that a single node failure (of a node internal to all these trees) could disconnect all these trees and if this happens early in the phase 2, we may not get enough information for the root(s) to make their decisions. Since our phase 2 is 'instant' i.e. involves no communication, we do not have any failure issues. This opens up the possibility of a fully-fault-tolerant tester for dynamic networks if a fault-tolerant phase 1 (i.e. fault-tolerant random walks) could be designed.

1.1 Technical Overview

In this section, we give a general overview of the concepts used in our algorithm. Like all the previous algorithms for conductance testing, we perform a certain number of random walks from a randomly selected starting vertex. To boost the success probability of the process, we repeat this process in parallel from a constant number of randomly selected starting vertices. The main technical challenge in running random walks in parallel from different starting vertices is the congestion on the edges. As done by [11], we overcome this problem by not sending the entire trace of the walk from its current endpoint to the next. For each starting point q and for all the walks going from u to v, we simply send the ID of q and the number of walks destined for v to v. At the end of this process, for each starting point q, we simply store at each vertex v, the number of walks that ended at v. Finally, each vertex $v \in V$ looks at the information stored at v to check if the number of walks received from any starting vertex is more than a certain threshold. If so, it outputs *Reject* and, otherwise, it outputs *Accept*.

Stickyness Helps: To show that the number of walks received by a vertex v is sufficient to decide whether v is part of a good conductor or not, we proceed as follows. A technical lemma from [22] implies that if a graph G is ϵ-far from being an α-conductor, then there exists a set $S \in V$ of sufficiently low conductance (of cut $(S, V \setminus S)$, see Definition 1)) and sufficiently high volume. It follows

intuitively that it is likely that a short random walk starting from a randomly selected starting vertex in S should not go very far and end in S. In particular, we show that there exist a subset $P \subseteq S$ such that short walks starting from any $v \in P$ end in a large enough region T (subset of S) around v. We make this notion precise by using spectral graph theory to show that a large portion of the volume of low-conductance set S (as described above) belongs to *sticky* vertices. We call a vertex $v \in S$ sticky if there exists a set $T \subseteq S$ such that $v \in T$ and short random walks starting from v end in T with a *sufficiently* high probability. We define $\text{trap}(v, T, \ell)$ as the probability that an ℓ length walk starting from $v \in T \subseteq S$ ends in T.

Trap Probability: Observe that our definition of trap probability is slightly different from the one generally used in the analysis of similar problems. The notion of trap probability is generally used to bound the probability of an ℓ-step random walk staying in a specific set for its entire duration (in each of the ℓ steps). See for example the definitions of *remain* and *escape* probabilities in [13]. Similarly, Czumaj and Sohler also implicitly use the concept of trap probability in their expansion tester [7] and they also bound the probability of a walk staying inside a set of low conductance for its entire duration. We relax the definition a bit and only care about the walk *ending* in a subset of a low conductance set. This allows us to also use the walks that may have briefly escaped the low conductance region when counting the number of trapped walks. Thus, if we run sufficiently many walks from one of the sticky vertices, then a lot of them will end in a subset T of S and some vertex in T will see a lot more walks than any vertex in a good conductor should. To ensure that we pick one of the sticky vertices as a starting vertex, we sample each vertex to be a starting vertex with appropriate probability.

Spectral Approach: In the analysis of the convergence behaviour of random walks (to the stationary distribution) using the eigenvectors and eigenvalues of the random walk matrix M (first introduced by Kale And Seshadhri [19] and later refined by [11] for unbounded degree graphs), they divide the set of eigenvectors of M into *heavy* and *light* sets. All the eigenvectors with eigenvalues above a certain threshold (appropriately chosen) are considered heavy and the rest and considered light. This lets one drop all the light eigenvectors from the analysis since their contribution to the convergence behaviour is minimal. In our analysis, we use a similar technique where we focus on the heavy eigenvectors of the random walk matrix M to lower bound the trap probability of a random walk from an appropriately chosen starting vertex. To further tighten our analysis, we further divide the set H of heavy eigenvectors into the *heaviest* eigenvector $\vec{e}_1$ (with the maximum eigenvalue 1) and the set $H \setminus \{\vec{e}_1\}$. We use both (but separately and not as one bundle H) in our analysis. This also makes intuitive sense since the heaviest eigenvector makes the maximum contribution to the trap probability and treating it separately tightens our bound. We note that [19] and [11] analyse a different measure - the discrepancy between their final endpoint probability distribution and the stationary distribution; and the contribution of $\vec{e}_1$ to this measure is zero, so their analysis does not benefit from

segregating the heaviest eigenvector from the set H of the heavy eigenvectors.
Organisation: The rest of the paper is organised as follows. In Section 2, we provide necessary definitions and state some basic lemmas that are used in rest of the paper. In Section 3, we provide a detailed description of our testing algorithm. Section 3.1 is dedicated to the proof of our main theorem. For any missing proofs not provided here, we refer the reader to the full version of the paper [3].

2 Preliminaries

Let $G = (V, E)$ be an unweighted, undirected graph on n vertices and m edges. We assume that the vertices of G have unique identifiers. For a given vertex $v \in V$, $\deg(v)$ denotes the *degree* of v. For sets $A, B \subseteq V$, we denote by $E(A, B)$ the number of edges that have one endpoint in A and the other in B. A cut is a partition of the vertices into two disjoint subsets. Given a graph $G = (V, E)$ any subset $S \subseteq V$ defines a cut denoted by $(S, \overline{S})$, where $\overline{S} = V \setminus S$.

Definition 1. *Given a cut $(S, \overline{S})$ in G, the conductance of $(S, \overline{S})$ is defined as $\frac{E(S,\overline{S})}{\min\{\mathrm{vol}(S),\mathrm{vol}(\overline{S})\}}$, where $\mathrm{vol}(A) = \sum_{v\in A} \deg(v)$. Alternatively, we also refer to the conductance of a cut $(S, \overline{S})$ as the* conductance of set S*. The conductance of a graph is the minimum conductance of any cut in the graph.*

Throughout the paper, all vectors $\vec{x} \in \mathbb{R}^n$ are column vectors. For a vector $\vec{x} \in \mathbb{R}^n$, we denote by $\vec{x}^\mathsf{T}$ the transpose of $\vec{x}$. For two vectors $\vec{x}$ and $\vec{y}$ in $\mathbb{R}^n$, $\langle \vec{x}, \vec{y} \rangle$ denotes their inner product. We denote the $n \times n$ adjacency matrix of the input graph G by A, where $A_{ij} = 1$, if $(i, j) \in E$ and 0 otherwise. Let D denote the $n \times n$ diagonal degree matrix of G, where $D_{ij} = \deg(i)$ if $i = j$ and 0 otherwise. The main technical tool in our analysis will be random walks on the input graph G. We denote a random walk by its transition matrix M. For a pair of vertices $u, v \in V$, let M_{uv} be the probability of visiting u from v in one step of M. In the standard definition of a random walk, M_{uv} is defined as $1/\deg(v)$ if there is an edge from u to v, and 0 otherwise.

We use a slightly modified version of the standard random walk called a *lazy random walk*. A *lazy random walk* currently stationed at $v \in V$, stays at v with probability 1/2 and, with the remaining probability 1/2, it visits a neighbour of v uniformly at random. Let M be a lazy random walk on G, the transition probabilities for M are defined as follows: for a pair $v, w \in V$ such that $v \neq w$, $M_{wv} = \frac{1}{2\deg(v)}$, if $(v, w) \in E$ and 0, otherwise. Further, for $v \in V$, we define $M_{vv} = 1/2$. Algebraically, M can be expressed as $M = \frac{1}{2}(I + AD^{-1})$, where I is the $n \times n$ identity matrix. Let π be the stationary distribution of M. In the stationary distribution of a lazy random walk, each vertex is visited with probability proportional to its degree. More formally, $\pi = \frac{\vec{d}}{2m}$, where $\vec{d}$ is an n-dimensional vector of vertex degrees and m is the number of edges in G.

In the following, we provide a brief exposition of relevant concepts from spectral graph theory. We refer the reader to the textbook [6] by Fan Chung for a detailed treatment of the subject. Note that for irregular graphs, M is

an asymmetric matrix and may not have an orthogonal set of eigenvectors. For analyzing random walks on G in terms of the eigenvalues and eigenvectors of its associated matrices, we rely on a related symmetric matrix called the *normalized Laplacian* of G denoted by N. The normalized Laplacian N is defined as

$$N = I - D^{-1/2}AD^{-1/2}.$$

We show below a way to express M in terms of N.

$$\begin{aligned} M &= 1/2(I + AD^{-1}) = I - 1/2(I - AD^{-1}) \\ &= I - 1/2D^{1/2}(I - D^{-1/2}AD^{-1/2})D^{-1/2} = D^{1/2}(I - N/2)D^{-1/2}. \end{aligned}$$

Since N is a symmetric matrix, it has a set of n real eigenvalues and a corresponding set of mutually orthogonal eigenvectors. Throughout we let $\omega_1 \leq \omega_2 \leq \ldots \leq \omega_n$ denote the set of eigenvalues and $\vec{\zeta_1}, \vec{\zeta_2}, \ldots, \vec{\zeta_n}$ denote the corresponding set of eigenvectors. It is well known that eigenvalues of N are $0 = \omega_1 \leq \omega_2 \leq \ldots \leq \omega_n \leq 2$. It is easy to verify that $\sqrt{\vec{d}} = (\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$ is the first eigenvector $\vec{\zeta_1}$ of N with corresponding eigenvalue $\omega_1 = 0$. Each of the orthogonal eigenvectors $\vec{\zeta_i}$ can be normalized to be a unit vector as $\vec{e_i} = \vec{\zeta_i}/\|\vec{\zeta_i}\|_2$ Together, these orthogonal unit eigenvectors define an orthonormal basis for $\mathbb{R}^n$. Observe that the first unit eigenvector $\vec{e_1}$ of this orthonormal basis is $\sqrt{\vec{d}}/\sqrt{2m}$. Also observe that the stationary distribution π of M is equal to $D^{1/2}\vec{e_1}/\sqrt{2m}$.

It is easy to verify that for every unit eigenvector $\vec{e_i}$ of N, we have a corresponding right eigenvector $D^{1/2}\vec{e_i}$ of M with eigenvalue $1 - \omega_i/2$.

On a connected, graph, a lazy random walk M can be viewed as a reversible, aperiodic Markov chain with state space V and transition matrix M.

Definition 2. *Let $\mathcal{M}$ be a reversible, aperiodic Markov chain on a finite state space V with stationary distribution π. Furthermore, let $\pi(S) = \sum_{v \in S} \pi(v)$. The Cheeger constant or conductance $\phi(\mathcal{M})$ of the chain is defined as*

$$\phi(\mathcal{M}) = \min_{S \subset V : \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \in V \setminus S} \pi(x)\mathcal{M}(y,x)}{\pi(S)}.$$

Here $\mathcal{M}(y,x)$ is the probability of moving to state y from state x in one step.

The definition of the lazy random walk matrix M and the the fact that the stationary distribution π of our lazy random walk is $\vec{d}/2m$ together imply that the Cheeger constant $\phi(M)$ (henceforth, ϕ_*) of the walk M is

$$\phi_* = \min_{U \in V, \mathrm{vol}(U) \leq \mathrm{vol}(V)/2} \frac{E(U, V \setminus U)}{2 \cdot \mathrm{vol}(U)}.$$

For an α-conductor, we get that

$$\phi_* = \frac{\alpha}{2}. \tag{1}$$

3 A Distributed Algorithm for Testing Conductance

Given an input graph $G = (V, E)$, a conductance parameter α, and a distance parameter ϵ, our distributed conductance tester distinguishes, with probability at least 2/3, between the case where G is an α-conductor and the case where G is ϵ-far from being an $\Omega(\alpha^2)$-conductor. A key technical lemma from [22] implies that, if G is ϵ-far from being an $\Omega(\alpha^2)$-conductor, then there exists a low-conductance cut $(S, \overline{S})$ such that $\text{vol}(S) \geq \epsilon m/10$. We build on this lemma to show, using spectral methods (see Lemma 3 and Corollary 1), that there exist a set of *sticky* vertices with high enough volume in S. Recall that a vertex x in a low-conductance set S is sticky if there exists a large enough subset $T \subset S$ such that $x \in T$ and a short random walk starting from x *ends* in T with a sufficiently high probability. Intuitively, random walks starting from sticky vertices tend to stick to a small region around them. This leads to some vertex in the graph receive more than their *fair share* of number of walks. On the other, hand if a graph is a good conductor, then the random walks from anywhere in the graph mix very quickly. This ensures that the fraction of the total number of walks received by any vertex in the graph is proportional to its degree.

We randomly sample a set Q of $\Omega(1/\epsilon)$ *source* vertices (each vertex $v \in V$ is sampled with probability proportional to its degree). Then we run a certain number of short random walks from each source in Q. Since a large part of the volume of our high-volume, low-conductance set S consists of only sticky vertices in a bad conductor, some vertex in Q will be sticky with sufficiently large probability. We use the number of walks received by each vertex from a specific source as a test criteria. We implement sampling of the set Q by having each vertex v sample itself by flipping a biased coin with probability $5000 \cdot \deg(v)/(2\epsilon m)$. Therefore, we get that $|Q| = 5000/\epsilon$ in expectation. It follows from Chernoff bound that the probability of Q having more than $5500/\epsilon$ vertices is at most $e^{-23/\epsilon} \leq e^{-23}$. Then, we perform K random walks of length ℓ starting from each of the chosen vertices in Q. The exact values of these parameters are specified later in the sequel. The pseudocode of the algorithm is presented in Algorithm 1. At any point before the last step of random walks, each vertex $v \in V$ contains a set W of tuples (q, count, i), where *count* is the number of walks of length i originating from source q currently stationed at v. All these walks are advanced by one step (for ℓ times) by invoking Algorithm 2. At the end of the last step of the walks, Algorithm 2 outputs a set of tuples C_v. Each tuple in C_v is of the form (q, count), where *count* is the total number of ℓ-step walks starting at q that ended at v. Then, in Algorithm 1, processor at vertex v goes over every tuple (q, count) in C_v (see Lines 11 to 15 of Algorithm 1), and if the *count* value of any of them is above a pre-defined threshold τ, it outputs *Reject*. If none of the tuples have their *count* value above threshold, it outputs *Accept*. The exact value of τ is specified later in the sequel.

When advancing the walks originating at a source $q \in Q$ by one step, we do not send the full trace of every random walk. Instead, for every source $q \in Q$, every vertex $v \in V$ only sends a tuple (q, k, i) to its neighbour w indicating that k random walks originating at q have chosen w as their destination in their ith

step. Since the size of Q is constant with high enough probability, we will not have to send more than a constant number of such tuples on each edge. Moreover, each tuple can be encoded using $O(\log n)$ bits (given the values of parameters ℓ and K specified in the sequel). Hence, we only communicate $O(\log n)$ bits per edge in any round with high probability. To ensure no congestion, we check the length of every message (Algorithm 2: lines 13 - 15). If a message appears too large to send, we simply output *Reject* on the host vertex and abort the algorithm. Note that the number of tuples we ever have to send along any edge is upper bounded by $|Q|$ and $|Q| \leq 5500\epsilon$, with probability at least $1 - e^{-23}$. Therefore, we may rarely abort the algorithm before completing the execution of the random walks. If that happens, then the probability of accepting an α-conductor is slightly reduced. Hence the following observation follows:

Observation 1 *Algorithm 1 rejects an α-conductor due to congestion with probability at most e^{-23}.*

We set the required parameters of Algorithm 1 as follows:

- the number of walks $K = 2m^2$,
- the length of each walk $\ell = \frac{32}{\alpha^2} \log n$
- the rejection threshold for vertex $v \in V$, $\tau_v = m \cdot \deg(v) \cdot (1 + 2n^{-1/4})$.

Algorithm 1 Distributed algorithm running at vertex v for testing conductance.

Algorithm DISTRIBUTED-GRAPH-CONDUCTANCE-TEST($G, \epsilon, \alpha, \ell, K$)
▷ The algorithm performs K random walks of length ℓ from a set Q of $\Theta(1/\epsilon)$ starting vertices, where every starting vertex is sampled randomly from V.
ℓ : The length of each random walk
K : The number of walks
W_v : Set of tuples $(q, count, i)$ ▷ where *count* is the number of walks originating at source q currently stationed at v
C_v : Set of tuples $(q, count)$ ▷ where *count* is the total number of ℓ step walks starting at q that ended at v
τ_v : maximum number of ℓ-length walks v should see from a given source on an α-conductor.
Flip a biased coin with probability $p = 5000 \deg(v)/(\epsilon 2m)$ to decide whether to start K lazy random walks.
If chosen, initialise W_v as $W_v \leftarrow \{(v, K, 0)\}$.
Call Algorithm 2 for ℓ synchronous rounds.
while there is some tuple $(q, count)$ in C_v **do**
    if $count > \tau_v$ **then** ▷ Received too many walks from q.
        Output *Reject* and stop all operations.
    else
        Remove $(q, count)$ from C_v
Output *Accept*

Algorithm 2 Algorithm for moving random walks stationed at v by one step.

Algorithm MOVE-WALKS-AT-v
  W_v : Set of tuples $(q, count, i)$ ▷ where *count* is the number of walks originating at source q currently stationed at v just after their ith step..
  D_v : Set of tuples $(q, count, dest)$ ▷ where *count* is the number of walks starting at q that are to be forwarded to *dest*.
  C_v : Set of tuples $(q, count)$ ▷ where *count* is the total number of walks starting at q that have v as their final destination or endpoint.
  $D_v \leftarrow \emptyset$.
  while there is some tuple (q, k, i) in W_v **do**
    if $i \neq L$ **then** ▷ If not the last step, process the next set of destinations.
      Draw the next set of destinations for the k walks and update the set D_v.
      Remove (q, k, i) from W_v
  ▷ If last step of the walks, update how many ended at v.
      Update C_v to reflect the k walks that ended in v.
  ▷ Prepare the messages to be sent
  while there is some tuple $(q, count, dest)$ in D_v **do**
    Add tuple $(q, count, i+1\rangle$ to the message to be sent to *dest*
  ▷ Check each message for length
  For each message M to be sent
  if the number of tuples in $M > 5500/\epsilon$ **then**
    Output *Reject* and stop all operations.
  Send all the messages to their respective destinations.
  ▷ Process the messages received
  For each source s from which a total of s_{count} walks are received,
add tuple $(s, s_{count}, i+1)$ to W_v

3.1 Analysis of the Algorithm

The main idea behind our algorithm is that, in a bad conductor, a random walk would converge to the stationary distribution more slowly and would initially get trapped within sets of vertices with small conductance. We provide a lower bound on the probability of an ℓ-step random walk starting from a vertex chosen at random (with probability proportional to its degree) from a subset T of a low-conductance set S finishing at some vertex in T.

Definition 3. *For a set $T \subseteq V$, and a vertex $u \in T$, let* $\text{trap}(u, T, \ell)$ *(henceforth* trap probability*) denote the probability of an ℓ-step random walk starting from $u \in T$ finishing at some vertex in T. When the starting vertex is chosen at random from T with probability proportional to its degree, we denote by* $\text{trap}(T, \ell)$ *the average trap probability (weighted by vertex degrees) over set T:* $\text{trap}(T, \ell) = \frac{1}{\text{vol}(T)} \sum_{u \in T} \deg(u) \cdot \text{trap}(u, T, \ell)$.

Given a set S of conductance at most δ and $T \subseteq S$, we establish a relationship between the average trap probability $\text{trap}(T, \ell)$ and conductance δ of S in the next two lemmas. We first consider the case $T = S$ in Lemma 1, and then obtain a bound when T is a subset (of sufficiently large volume) of S in Lemma 3.

Lemma 1. *Consider a set $S \subseteq V$ such that the cut $(S, \overline{S})$ has conductance at most δ. Then, for any integer $\ell > 0$, the following holds*

$$\text{trap}(S, \ell) \geq \frac{\text{vol}(S)}{2m} + \left(\frac{5}{6} - \frac{\text{vol}(S)}{2m}\right)(1 - 3\delta)^{\ell}.$$

Proof. Let $\mathbb{1}_S$ denote the n-dimensional indicator vector of set S. We pick a source vertex $v \in S$ with probability $\deg(v)/\text{vol}(S)$, where $\deg(v)$ is degree of v and $\text{vol}(S)$ is the sum of degrees of vertices in set S. This defines an initial probability distribution denoted by $\vec{p}_S^{\,0}$ on the vertex set V, where, $\vec{p}_S^{\,0}(v) = d(v)/\text{vol}(S)$ for $v \in S$ and $\vec{p}_S^{\,0} = 0$ for $v \notin S$. Note that $\vec{p}_S^{\,0} = \frac{1}{\text{vol}(S)} D\mathbb{1}_S$, where, D is the diagonal degree matrix of G. Denote by M the transition matrix of a lazy random walk on G. The endpoint probability distribution $\vec{p}_S^{\,\ell}$ of an ℓ-step lazy random walk on G starting from a vertex chosen from S according to $\vec{p}_S^{\,0}$ is

$$\vec{p}_S^{\,\ell} = M^{\ell}\vec{p}_S^{\,0} = (1/\text{vol}(S))M^{\ell}D\mathbb{1}_S.$$

Recall that M can be expressed in terms of the normalized Laplacian matrix $N = I - D^{-1/2}AD^{-1/2}$ of G as $M = D^{1/2}(I - N/2)D^{-1/2}$. (See Section 2.)

It follows therefore that $\vec{p}_S^{\,\ell} = (1/\text{vol}(S))\left(D^{1/2}\left(I - N/2\right)D^{-1/2}\right)^{\ell} D\mathbb{1}_S$.

The trap probability $\text{trap}(S, \ell)$ of an ℓ-step lazy random walk starting from a random vertex in S picked according to $\vec{p}_S^{\,0}$ can be expressed as the inner product of vectors $\vec{p}_S^{\,\ell}$ and $\mathbb{1}_S$:

$$\begin{aligned}\text{trap}(S, \ell) &= \frac{1}{\text{vol}(S)}\mathbb{1}_S^{\mathsf{T}}M^{\ell}D\mathbb{1}_S = \frac{1}{\text{vol}(S)}\mathbb{1}_S^{\mathsf{T}}\left(D^{1/2}\left(I - N/2\right)D^{-1/2}\right)^{\ell} D\mathbb{1}_S \\ &= \frac{1}{\text{vol}(S)}(D^{1/2}\mathbb{1}_S)^{\mathsf{T}}\left(I - N/2\right)^{\ell}(D^{1/2}\mathbb{1}_S).\end{aligned}$$

Recall that $0 = \omega_1 \leq \omega_2 \leq \cdots \leq \omega_n < 2$ are the eigenvalues of N and $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ denote the corresponding unit eigenvectors. We can express $D^{1/2}\mathbb{1}_S$ in the orthonormal basis defined by the eigenvectors of N as $D^{1/2}\mathbb{1}_S = \sum_i \alpha_i \vec{e}_i$. It follows that

$$\sum_i \alpha_i^2 = \langle D^{1/2}\mathbb{1}_S, D^{1/2}\mathbb{1}_S\rangle = \text{vol}(S). \tag{2}$$

Taking the quadratic form of N for vector $D^{1/2}\mathbb{1}_S$, we get

$$\begin{aligned}(D^{1/2}\mathbb{1}_S)^{\mathsf{T}}N(D^{1/2}\mathbb{1}_S) &= (D^{1/2}\mathbb{1}_S)^{\mathsf{T}}I(D^{1/2}\mathbb{1}_S) - (D^{1/2}\mathbb{1}_S)^{\mathsf{T}}(D^{-1/2}AD^{-1/2})(D^{1/2}\mathbb{1}_S) \\ &= \text{vol}(S) - (D^{1/2}\mathbb{1}_S)^{\mathsf{T}}(D^{-1/2}AD^{-1/2})(D^{1/2}\mathbb{1}_S) = \text{vol}(S) - \mathbb{1}_S^{\mathsf{T}}A\mathbb{1}_S.\end{aligned}$$

Note that the term $\mathbb{1}_S^{\mathsf{T}}A\mathbb{1}_S$ corresponds to the number of edges in $S \times S$. Therefore. it follows that

$$(D^{1/2}\mathbb{1}_S)^{\mathsf{T}}N(D^{1/2}\mathbb{1}_S) = E(S, \overline{S}).$$

Since the conductance of the cut $(S, \overline{S})$ is at most δ, we have that

$$(D^{1/2}\mathbb{1}_S)^{\mathsf{T}}N(D^{1/2}\mathbb{1}_S) = E(S, \overline{S}) \leq \delta \cdot \text{vol}(S). \tag{3}$$

Expressing $D^{1/2}\mathbb{1}_S$ as $\sum_i \alpha_i \vec{e}_i$, the quadratic form of N for $D^{1/2}\mathbb{1}_S$ can also be written as

$$(D^{1/2}\mathbb{1}_S)^{\mathsf{T}} N (D^{1/2}\mathbb{1}_S) = (\sum_i \alpha_i \vec{e}_i)^{\mathsf{T}} N (\sum_i \alpha_i \vec{e}_i) = \sum_i \alpha_i^2 \omega_i. \tag{4}$$

Combining from Eq.s (2), (3) and (4), we get that

$$(D^{1/2}\mathbb{1}_S)^{\mathsf{T}} (I - N/2)(D^{1/2}\mathbb{1}_S) = \sum_i \alpha_i^2 - \frac{1}{2}\sum_i \alpha_i^2 \omega_i \geq \mathrm{vol}(S) - \frac{\delta}{2}\mathrm{vol}(S). \tag{5}$$

Recall that $0 = \omega_1 \leq \omega_2 \leq \ldots \leq \omega_n < 2$ are the eigenvalues of N and let $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ be the corresponding unit eigenvectors. Correspondingly, we can define a set of eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$ and the same set of eigenvectors $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n$ for $I - N/2$. Notice that for each i, $\lambda_i = 1 - \omega_i/2$. With this translation of eigenspace, we get that

$$(D^{1/2}\mathbb{1}_S)^{\mathsf{T}} (I - N/2)(D^{1/2}\mathbb{1}_S) = \sum_i \alpha_i^2 \lambda_i.$$

We call the quantity $\sum_i \alpha_i^2$ the *coefficient sum of the eigenvalue set*. We also call an eigenvalue λ_i of $I - N/2$ (and the corresponding eigenvector $\vec{e}_i$) *heavy* if $\lambda_i \geq 1 - 3\delta$. We denote by H the index set of the heavy eigenvalues and let $\overline{H}$ be the index set of the rest. Since $\sum_i \alpha_i^2 \lambda_i \geq (1 - \delta/2)\,\mathrm{vol}(S)$ is large for a set with small conductance, we expect many of the coefficients α_i^2 corresponding to heavy eigenvalues to be large. This would slow down the convergence of the random walk and make the trap probability for our low-conductance set S large. The following lemma establishes a lower bound on the contribution of the index set H to the coefficient sum.

Lemma 2. *For $\{\alpha_i\}_i, H$, and s as defined above, $\sum_{i\in H} \alpha_i^2 \geq \frac{5}{6}\,\mathrm{vol}(S)$.*

See the full version [3] for the proof. Lemma 2 gives us a lower bound on the average trap probability of set S in terms of the conductance of the cut $(S, \overline{S})$.

$$\begin{aligned}
\mathrm{trap}(S,\ell) &= \frac{1}{\mathrm{vol}(S)} (D^{1/2}\mathbb{1}_S)^{\mathsf{T}} (I - N/2)^{\ell} (D^{1/2}\mathbb{1}_S) \\
&= \frac{1}{\mathrm{vol}(S)} (\sum_i \alpha_i \vec{e}_i)^{\mathsf{T}} (I - N/2)^{\ell} (\sum_i \alpha_i \vec{e}_i) \\
&= \frac{1}{\mathrm{vol}(S)} (\sum_i \alpha_i \vec{e}_i)^{\mathsf{T}} (\sum_i \alpha_i \lambda_i^{\ell} \vec{e}_i) = \frac{1}{\mathrm{vol}(S)} \sum_i \alpha_i^2 \lambda_i^{\ell}.
\end{aligned}$$

Further, focusing on the contribution of the index set H to the trap probability,

$$\begin{aligned}
\mathrm{trap}(S,\ell) &= \frac{1}{\mathrm{vol}(S)} \sum_i \alpha_i^2 \lambda_i^{\ell} \geq \frac{1}{\mathrm{vol}(S)} \sum_{i\in H} \alpha_i^2 \lambda_i^{\ell} \\
&= \frac{1}{\mathrm{vol}(S)} (\alpha_1^2 \lambda_1 + \sum_{i \in H\setminus\{1\}} \alpha_i^2 \lambda_i^{\ell}) \\
&\geq \frac{1}{\mathrm{vol}(S)} \left(\alpha_1^2 + (5\,\mathrm{vol}(S)/6 - \alpha_1^2)\,(1 - 3\delta)^{\ell}\right).
\end{aligned} \tag{6}$$

The last inequality follows by the definition of a heavy eigenvalue and by Lemma 2 (we have that $\sum_{i\in H}\alpha_i^2 \geq 5\,\mathrm{vol}(S)/6$). By definition, $\lambda_1 = 1$ and $\vec{e}_1 = \sqrt{\vec{d}}/\sqrt{2m}$, where, $\vec{d}$ is the vector of vertex degrees. It follows that $\alpha_1 = \langle D^{1/2}\mathbb{1}_S, \vec{e}_1\rangle = \mathrm{vol}(S)/\sqrt{2m}$. Plugging in the values of α_1 in (6), we get

$$\begin{aligned}\mathrm{trap}(S,\ell) &\geq \frac{1}{\mathrm{vol}(S)}\left(\frac{(\mathrm{vol}(S))^2}{2m} + \left(\frac{5\,\mathrm{vol}(S)}{6} - \frac{(\mathrm{vol}(S))^2}{2m}\right)(1-3\delta)^\ell\right)\\ &= \frac{\mathrm{vol}(S)}{2m} + \left(\frac{5}{6} - \frac{\mathrm{vol}(S)}{2m}\right)(1-3\delta)^\ell.\end{aligned}$$

Next lemma states that every subset $T \subset S$ of large enough volume has high trap probability.

Lemma 3. *Consider sets $T \subseteq S \subseteq V$, such that the cut $(S,\overline{S})$ has conductance at most δ and that $\mathrm{vol}(T) = (1-\eta)\,\mathrm{vol}(S)$ for some $0 < \eta < 5/6$, then for any integer $\ell > 0$, there exists a vertex $v \in T$ such that*

$$\mathrm{trap}(v,T,\ell) \geq \frac{\mathrm{vol}(T)}{2m} + \left(\frac{5}{6}(1-\sqrt{(6\eta)/5})^2 - \frac{\mathrm{vol}(T)}{2m}\right)(1-3\delta)^\ell. \tag{7}$$

Refer to the full version [3] for the proof.

Lemma 3 implies the following corollary.

Corollary 1. *Consider a set $S \subset V$, such that the cut $(S,\overline{S})$ has conductance at most δ. Given any $0 < \eta < 5/6$ and integer $\ell > 0$, there exist a set of volume at least $\eta \cdot \mathrm{vol}(S)$, such that every vertex in this set is sticky. In other words, for every vertex v is this set, there exists $T \subseteq S$ of volume $\mathrm{vol}(T) = (1-\eta)\cdot\mathrm{vol}(S)$ such that $\mathrm{trap}(v,T,\ell)$ is given by (7).*

Refer to the full version [3] for the proof.

We build on the following combinatorial lemma from [22]:

Lemma 4 (Lemma 9 of [22]). *Let $G = (V,E)$ be an m-edge graph. If there exists a set $P \subseteq V$ such that $\mathrm{vol}(P) \leq \epsilon m/10$ and the subgraph $G[V \setminus P]$ that is induced by the vertex set $V \setminus P$ has conductance at least ϕ', then there exists an algorithm that modifies at most ϵm edges of G to get a graph $G' = (V,E')$ with conductance at least $\phi/'3$.*

In the following lemma, we show the existence of a high volume set A with low enough conductance in a graph that is far from being a good conductor.

Lemma 5. *Let $G = (V,E)$ be an n-vertex, m-edge graph such that G is ϵ-far from having conductance at least $\alpha^2/2880$, then there exists a set $A \subseteq V$ such that $\mathrm{vol}(A) \geq \epsilon m/10$ and conductance of cut $(A,\overline{A})$ is at most $\alpha^2/960$.*

Refer to the full version [3] for the proof.

Finally, we need the following classical relation between the conductance or Cheeger constant of a Markov chain and its second largest eigenvalue.

Theorem 2 ([2,1,9]). *Let P be a reversible lazy chain (i.e., for all x, $P(x,x) \geq 1/2$) with Cheeger constant ϕ_*. Let λ_2 be the second largest eigenvalue of P. Then, $\frac{\phi_*^2}{2} \leq 1 - \lambda_2 \leq 2\phi_*$.*

We can now state our main theorem.

Theorem 3. *For an input graph $G = (V, E)$, parameters $0 < \alpha < 1$ and $\epsilon > 0$, the distributed algorithm described in Section 3*

- *outputs Accept, with probability at least $2/3$, on every vertex of G if G is an α-conductor.*
- *outputs Reject, with probability at least $2/3$, on at least one vertex of G if G is ϵ-far from any $(\alpha^2/2880)$-conductor.*

The algorithm uses $O(\log n/\alpha^2)$ communication rounds.

Proof. Let us start by showing that, with high enough probability, the algorithm outputs *Accept* on every vertex if G is an α-conductor. By Observation 1, we may reject G and abort the algorithm due to congestion with probability at most e^{-23}. For now, let us assume that this event did not occur. Denote by λ_2 the second largest eigenvalue of the lazy random walk matrix M on G. It is well known (see, e.g., [27]) that, for a pair $u, v \in V$, $\left|M^\ell(v,u) - \deg(v)/(2m)\right| \leq \lambda_2^\ell \leq e^{-\ell(1-\lambda_2)}$. It follows from Theorem 2 that $\left|M^\ell(v,u) - \frac{\deg(v)}{2m}\right| \leq e^{-\ell\phi_*^2/2} \leq e^{-\frac{\ell\alpha^2}{8}}$, where the second inequality above follows from the fact that, for a random walk on an α-conductor, $\phi_* = \alpha/2$ (see (1)). Thus, in an α-conductor, for $\ell = (32/\alpha^2)\log n$, any starting vertex $u \in V$ and a fixed vertex $v \in V$, we have that

$$\deg(v)/(2m) - 1/n^4 \leq M^\ell(v,u) \leq \deg(v)/(2m) + 1/n^4.$$

Recall that the number K of random walks and rejection threshold τ_v for vertex v are set as $K = 2m^2$ and $\tau_v = m \cdot \deg(v) \cdot (1 + 2n^{-1/4})$. Let $X_{u,v}$ denote the number of random walks starting from u that ended in v. It follows that

$$\mathbb{E}X_{u,v} = K \cdot M^\ell(v,u) \leq 2m^2 \cdot \frac{\deg(v)}{2m} + \frac{2m^2}{n^4} \leq m \cdot \deg(v) + 1.$$

The random variable $X_{u,v}$ is the sum of K independent Bernoulli trials with success probability $M^\ell(v,u)$. Applying multiplicative Chernoff bounds, we get the following for large enough n.

$$\Pr[X_{u,v} > (1 + n^{-1/4}) \cdot \mathbb{E}[X_{u,v}]] < \exp(-n^{-1/2} \cdot (m \cdot \deg(v) + 1)/3) \leq \exp(\frac{-n^{1/2}}{3}).$$

The second inequality above follows from the fact that $m \cdot \deg(v) > n - 1$ for a connected graph. Thus, each vertex y receives at most

$$(1 + n^{-1/4}) \cdot \mathbb{E}[X_{u,v}] \leq (1 + n^{-1/4}) \cdot (m \cdot \deg(v) + 1) < m \cdot \deg(v) + 2m/n^{1/4} \cdot \deg(v)$$

walks from u, with probability at least $1 - n^{-2}$. Taking union bound over all $y \in V$ and all starting vertices u, we get that, with high probability, our algorithm outputs *Accept* on every vertex of G for every starting point if G is

an α-conductor. Finally, taking the union bound over the events that we rejected due to congestion or due to receiving too many walks at some vertex, the claim follows.

Next, we analyse the probability of rejecting if G is far from having the desired conductance. By Lemma 5, there exists a set $S \subset V$, with $\mathrm{vol}(S) \geq \epsilon m/10$, such that the conductance of S is at most $\alpha^2/960$. Further applying Corollary 1 with $\eta = 5/486$ on S as above, we get that there exists a set P of sticky vertices such that $\mathrm{vol}(P) \geq (5\,\mathrm{vol}(S))/486$ and for every $v \in P$, there exists a set $T \subseteq S$, $\mathrm{vol}(T) = (481\,\mathrm{vol}(S))/486$, such that

$$\begin{aligned}\mathrm{trap}(v,T,\ell) &\geq \frac{\mathrm{vol}(T)}{2m} + \left(\frac{160}{243} - \frac{\mathrm{vol}(T)}{2m}\right)\left(1 - \frac{\alpha^2}{320}\right)^{\ell} \\ &\geq \frac{\mathrm{vol}(T)}{2m} + \left(\frac{160}{243} - \frac{\mathrm{vol}(T)}{2m}\right) \cdot e^{-\alpha^2\ell/160}\end{aligned}$$

where the last inequality follows from that $1 - x > e^{-2x}$, for $0 \leq x < 1/2$, provided that $\alpha^2/320 \leq 1/2$. For $\ell = (32/\alpha^2)\log n$ and $\mathrm{vol}(T) \leq \mathrm{vol}(S) \leq \mathrm{vol}(V)/2 = m$, we get that, for every $v \in P$, $\mathrm{trap}(v,T,\ell) \geq \frac{\mathrm{vol}(T)}{2m} + \frac{77}{486} \cdot \frac{1}{n^{1/5}}$.

Let us assume that a vertex $u \in P \subset A$ is picked as the starting vertex of $K = 2m^2$ random walks in G. By Corollary 1, a set T with $\mathrm{vol}(T) = (481\,\mathrm{vol}(S))/486$ with $\mathrm{trap}(v,T,\ell) \geq \mathrm{vol}(T)/2m + 77/486 \cdot n^{-1/5}$ will exist, for every $v \in P$. Also note that $\mathrm{vol}(T) \leq \mathrm{vol}(S) \leq \mathrm{vol}(V) = 2m$. For some appropriate constant c_1, $\mathrm{vol}(T) = c_1 \cdot m = \Theta(m)$. Further, let $Y_{u,T}$ be the number of walks that ended in the set T (corresponding to u as in Corollary 1) after ℓ steps. It follows that

$$\mathbb{E}Y_{u,T} \geq K \cdot \left(\frac{\mathrm{vol}(T)}{2m} + \frac{77}{486}n^{-1/5}\right) \geq m \cdot \mathrm{vol}(T) + \frac{154}{486} \cdot \frac{m^2}{n^{1/5}}.$$

Let $c_2 = \frac{77}{486}$. By an application of Chernoff bound,we get

$$\begin{aligned}&\Pr\left[Y_{u,T} < \left(1 - 3(m \cdot \mathrm{vol}(T))^{-1/2}\right)\mathbb{E}[Y_{u,T}]\right] \\ &< \exp\left(-4(m \cdot \mathrm{vol}(T))^{-1} \cdot \mathbb{E}[Y_{u,T}]\right) \\ &\leq \exp\left(-4(m \cdot \mathrm{vol}(T))^{-1} \cdot \left(\mathrm{vol}(T) \cdot m + 2c_2m^2n^{-1/5}\right)\right) \\ &\leq \exp\left(-4 \cdot (1 + \Theta(n^{-1/5})\right) < 1/10.\end{aligned}$$

With probability at least 9/10,the total number of walks received by set T is

$$\begin{aligned}&\geq (1 - 3(m \cdot \mathrm{vol}(T))^{-1/2})\left(m \cdot \mathrm{vol}(T) + 2c_2m^2/n^{1/5}\right) \\ &\geq m \cdot \mathrm{vol}(T) + 2c_2m^2/n^{1/5} - 3\sqrt{m}\sqrt{\mathrm{vol}(T)} - 6c_2m^{3/2}(\mathrm{vol}(T))^{-1/2}n^{-1/5}.\end{aligned}$$

The number of walks received by any vertex $v \in T$ is minimum when the walks within T have mixed well reaching their stationary distribution with respect to T. It follows that the number of walks received by a vertex $v \in T$ is at least

$$\deg(v) \cdot \left(m + 2c_2\frac{m^2}{n^{1/5} \cdot \mathrm{vol}(T)} - 3\frac{\sqrt{m} \cdot \sqrt{\mathrm{vol}(T)}}{\mathrm{vol}(T)} - 6c_2\frac{m^{3/2}}{(\mathrm{vol}(T))^{3/2}n^{1/5}}\right).$$

Recalling that $\mathrm{vol}(T) = c_1 m$, the expected number of walks received by a vertex $v \in T$ is at least

$$\deg(v)\cdot\left(m + 2\frac{c_2}{c_1}\frac{m}{n^{1/5}} - \frac{3}{\sqrt{c_1}} - 6\frac{c_2}{(c_1)^{3/2}n^{-1/5}}\right) = \deg(v)\cdot\left(m + 2\frac{c_2}{c_1}\frac{m}{n^{1/5}} - O(1)\right).$$

Therefore, on average, vertex $v \in T$ of degree $\deg(v)$ receives more than the threshold $\tau_v = m \cdot \deg(v) + 2m \cdot \deg(v)/n^{1/4}$ number of walks for large enough n. Thus, some vertex in T will receive more than τ_v walks and output *Reject*.

Let $\mathcal{E}$ be the event that none of the vertices in P is sampled to be one of the starting points in Q. Since each vertex $u \in V$ is sampled with probability $5000 \cdot \deg(v)/(2\epsilon \cdot m)$ and $\mathrm{vol}(P) \geq (5\epsilon \cdot m)/4860$, it follows that

$$Pr[\mathcal{E}] \leq (1 - 5000 \cdot \deg(v)/(2\epsilon \cdot m))^{\frac{5\epsilon \cdot m}{4860}} \leq e^{-2.5} = 0.08.$$

Taking a union bound over the probability of the event $\mathcal{E}$ and the probability of set T around a starting vertex $u \in P$ not receiving enough walks, we get that with probability at most $0.1 + 0.08 = 0.18$, no vertex will output *Reject*. Thus, our distributed algorithm will output *Reject* with probability at least $2/3$, on at least one vertex of G. Finally, the upper bound on the number of communication rounds follows from the length $\ell = \frac{32}{\alpha^2}\log n$ of each random walk.

References

1. Alon, N.: Eigenvalues and expanders (1986), https://doi.org/10.1007/BF02579166
2. Alon, N., Milman, V.D.: Eigenvalues, expanders and superconcentrators (extended abstract). In: 25th Annual Symposium on Foundations of Computer Science. IEEE Computer Society (1984), https://doi.org/10.1109/SFCS.1984.715931
3. Batu, T., Trehan, A., Trehan, C.: A distributed conductance tester without global information collection (2023), https://doi.org/10.48550/arXiv.2305.14178
4. Brakerski, Z., Patt-Shamir, B.: Distributed discovery of large near-cliques. In: Distributed Computing (2009)
5. Censor-Hillel, K., Fischer, E., Schwartzman, G., Vasudev, Y.: Fast Distributed Algorithms for Testing Graph Properties. Distributed Computing (2019)
6. Chung, F.R.K.: Spectral Graph Theory. AMS, Providence, RI (1997)
7. Czumaj, A., Sohler, C.: Testing Expansion in Bounded-Degree Graphs. Combinatorics, Probability & Computing (2010)
8. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. STOC '11 (2011), https://doi.org/10.1145/1993636.1993686
9. Dodziuk, J.: Difference equations, isoperimetric inequality and transience of certain random walks. Transactions of the American Mathematical Society **284**(2) (1984), http://www.jstor.org/stable/1999107
10. Even, G., Fischer, O., Fraigniaud, P., Gonen, T., Levi, R., Medina, M., Montealegre, P., Olivetti, D., Oshman, R., Rapaport, I., Todinca, I.: Three Notes on Distributed Property Testing. In: 31st International Symposium on Distributed Computing (DISC) (2017), http://drops.dagstuhl.de/opus/volltexte/2017/7984

11. Fichtenberger, H., Vasudev, Y.: A two-sided error distributed property tester for conductance. In: 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018 (2018), https://doi.org/10.4230/LIPIcs.MFCS.2018.19
12. Fischer, O., Gonen, T., Oshman, R.: Distributed Property Testing for Subgraph-Freeness Revisited. CoRR (2017), http://arxiv.org/abs/1705.04033
13. Gharan, S.O., Trevisan, L.: Approximating the expansion profile and almost optimal local graph clustering. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012 (2012), https://doi.org/10.1109/FOCS.2012.85
14. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. In: Proceedings of 37th Conference on Foundations of Computer Science (1996). https://doi.org/10.1109/SFCS.1996.548493
15. Goldreich, O.: Introduction to Testing Graph Properties. Springer Berlin Heidelberg (2010), https://doi.org/10.1007/978-3-642-16367-8_7
16. Goldreich, O.: Introduction to Property Testing. Cambridge University Press (2017). https://doi.org/10.1017/9781108135252
17. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing. STOC '97 (1997), https://doi.org/10.1145/258533.258627
18. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. Electron. Colloquium Comput. Complex. (2000), https://eccc.weizmann.ac.il/eccc-reports/2000/TR00-020/index.html
19. Kale, S., Seshadhri, C.: An expansion tester for bounded degree graphs. SIAM J. Comput. **40** (2011), https://doi.org/10.1137/100802980
20. Kutten, S., Pandurangan, G., Peleg, D., Robinson, P., Trehan, A.: Sublinear bounds for randomized leader election. In: Distributed Computing and Networking (2013)
21. Li, A., Pan, Y., Peng, P.: Testing conductance in general gr phs. Electron. Colloquium Comput. Complex. **TR11** (2011), https://api.semanticscholar.org/CorpusID:15721089
22. Li, A., Peng, P.: Testing small set expansion in general graphs. In: 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015 (2015), https://doi.org/10.4230/LIPIcs.STACS.2015.622
23. Łącki, J., Mitrović, S., Onak, K., Sankowski, P.: Walking randomly, massively, and efficiently. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. STOC 2020 (2020), https://doi.org/10.1145/3357713.3384303
24. Molla, A.R., Pandurangan, G.: Distributed computation of mixing time. In: Proceedings of the 18th International Conference on Distributed Computing and Networking. ICDCN '17, Association for Computing Machinery (2017), https://doi.org/10.1145/3007748.3007784
25. Nachmias, A., Shapira, A.: Testing the expansion of a graph. Information and Computation (2010), https://doi.org/10.1016/j.ic.2009.09.002
26. Sarma, A.D., Molla, A.R., Pandurangan, G.: Distributed computation of sparse cuts via random walks. In: Proceedings of the 16th International Conference on Distributed Computing and Networking. ICDCN '15, Association for Computing Machinery (2015), https://doi.org/10.1145/2684464.2684474
27. Sinclair, A.: Algorithms for Random Generation and Counting: A Markov Chain Approach. Birkhauser Verlag (1993)