Learning Stochastically Evolving Networks via Local Probing

Rajesh Jayaram

Advisor: Eli Upfal

April 25, 2017

Abstract

We consider the problem of learning the state of dynamic network with vertex values that are perturbed over time. Our only interface to the network comes in the form of point probes, where we are allowed to query local information about the state of a specific point in the network. In this paper, we consider network models where the values of the vertices are perturbed uniformly at random at every time step, and where we may only query adjacent vertices to determine whether or not they have the same value. This model is drawn from numerous practical examples, where determining the precise value of a vertex is impossible, but differentiating between adjacent vertices with disagreeing values is feasible.

Under this model we consider both the noiseless and noisy cases, where in the latter our probes are subject to a uniform noise with probability α. We first derive a clear inverse linear lower bound tradeoff between the number of probes and the fraction of errors in the network for either case. In the noiseless case, we design a algorithm which randomly initializes and then deterministically traverses the network to update a hypothesis state. We show that our algorithm is always within a constant factor of the lower bound for arbitrarily high polynomially many time steps with high probability. For the noisy case, the problem becomes substantially more difficult, and performance will depend on the expansion of the graph. We show that an algorithm which allows at least $k \in \Omega(\frac{\log(n)}{1-\lambda_2} r)$ probes on every time step never accumulates more than $O(\frac{n}{\sqrt{r}})$ errors at any time for arbitrarily high polynomially many time steps, assuming that $\alpha^{-1} \in \Omega(k)$, and where $(1-\lambda_2)$ is the spectral gap of the graph. An alternate analysis shows that for quadratically many time steps we can tighten this bound to $O(\frac{n}{r(1-\alpha\frac{\log(n)}{(1-\lambda_2)})})$ assuming $\alpha^{-1} \in \Omega(k^2)$. Furthermore, we demonstrate that if the number of errors accumulated at time t exceeds our bounds by a factor of M, then in expectation we return back within the bound in at most $O(n \log(M))$ steps.

Since our bounds for the noisy case are not as tight (we lose at least logarithmic factor in the probe-error tradeoff), we demonstrate experimentally that our algorithm in the noisy case appears to perform just as well as the theoretical performance of our algorithm in the noiseless case.

1 Introduction

The augmentation of a network structure with vertex values, or labels, is a natural enhancement and captures a variety of interesting of real world phenomenon. For instance, the augmentation of social network models with certain binary traits of the members, such as political classification or any number of other binary divisors of a target audience. Additionally notable is the application to computer vision, where the graph in question is the grid (or to simplify the analysis the discrete torus: $\mathbb{Z}_n \times \mathbb{Z}_n$). Here each vertex represents a pixel in an $n \times n$ video image, and vertex values can then be utilized for foreground background segmentation. Indeed, this setting was considered explicitly for the static case in [3]. Our paper can be seen as a dynamic version of [3], where vertex values are perturbed over discretized time steps. This can correspondingly be seen as moving from a static image to a dynamic video feed, where the pixel labellings, such as foreground and background, are changing as the image in the feed changes.

An important characteristic of the algorithmic framework which we consider in this paper is that each every time step we are only allowed to probe a certain, generally assumed to be constant or polylogarithmic, number of edges in the graph, and we are then told whether or not the incident vertices have agreeing labels. This is the same model of *edge observations* used in [3]. Our method of transporting a static algorithm to a dynmic setting bears resembalance to the method of [1], where quicksort is reintroduced on a stochastically evolving array.

Now our model of allowing only edge observations and not vertex observations captures an important feature of the practical applications from which our model arises. Namely, it is frequently impractical to determine, based on looking at one pixel (or individual in a social network) alone, whether or not it is in the background (or of a particular binary classifier). However, by looking at adjacent pixels (or *friends* in the network), it may be much easier to determine whether the two vertices should have the same classification. For pixels, a simple method of doing this is by computing the ℓ_1 distance between the RGB values of adjacent pixels, and determining that they are *alike* if the distance falls under a certain threshold parameter. Of course, any such comparison method will fail with some probability, which we refer to as *noise* in the edge probes. In Section 3 we introduce an algorithm to produce good results even with this noise.

Different models for stochastically evolving networks have been considered before. For instance, in [2], the authors consider a similar notion of an evolving graph where the actual topology of the graph, vertices and their neighborhoods are perturbed over time. In our paper we assume that the topology of network is fixed, and focus instead on perturbations in vertex labellings. However, it would be perhaps interesting to examine models where both the network topology and vertex labeling are perturbed over time.

1.1 Results & Techniques

The model which we consider in this paper is as follows: we have a graph with binary vertex labels which flip i.i.d. at every time step with probability β. Our algorithm is allowed to probe an edge and given the `XOR` of the two adjacent vertices. The goal is then to maintain a hypothesis vector of the values of the vertices that has close ℓ_1 distance to actual values at any time step. This is a streaming variation of the model considered in [3]. In this setting, we first derive lower bounds for

the expected number of errors for any algorithm running out algorithm in the model.

Theorem (1)**.** *In the stochastic network model where vertices are flipped by nature with probability β, any algorithm which uses at most k probes on every time step never accumulates less than $\beta\frac{n^2}{4k} - \beta^2\frac{n}{2k}$ errors at any time step in expectation.*

For practical purposes it makes sense to consider only settings where the expected number of flips on every time step is constant, thus we specify to $\beta = \frac{1}{2n}$ in this paper, although our results hold for any constant multiple of this probability. Being the case, our lower bound demonstrates an lower bound of a linear error-probe tradeoff in expectation. We then introduce the *Trailing Walks Algorithm*, which demonstrates that this lower bound can be nearly met with high probability (and not just in expectation) for polynomially many time steps.

Theorem (7)**.** *Let $n\epsilon$ be the number of initial errors of our hypothesis at time $t = 0$, and fix any $k = k'(r + s)$, and fix $\gamma > 0, \delta \geq 1$. Then with probability at least $(1 - \frac{e}{n^\gamma})(1 - e^{-\frac{\delta n}{6sk'} + \log T})(1 - \frac{3}{k'})$ the (k', r, s)-Trailing Walks Algorithm has no more than $4(\log(k'))^2 n\epsilon + (1 + \delta)\frac{n}{2sk'}$ errors at any time at all before $t = \frac{T}{n^\gamma}$ where T is as in Lemma 1.*

Remark. *Note that $T \in O((\frac{n}{2})^{(r+1)/2}\frac{1}{k'})$*

We then generalize to the case of edge noise. In other words, the case of the prior model but with the additional complication that every edge probe is given the *incorrect* `XOR` value with some fixed uniform probability α. For practical purposes, we assume that probing an edge multiple times on the same time step will not yield different results. Note that if we allowed this, then repeating the same algorithm from the noiseless section but probing each edge logarithmically many more times would yield an obvious algorithm to ensure the same bounds with high probability. It turns out, however, that after disallowing this the problem becomes quickly infeasible for constant values of α. In order to solve the problem theoretically, our algorithm requires that α is in $O(\frac{1}{k})$. Given that k is generally assumed to be at most polylogarithmic, this is as close to constant as α will practically be.

Given this, our algorithm probes random paths and, via a majority vote from our hypothesis in the prior step, decides which labeling such a path should receive. In order for such a random path sample to be uniform, which will be required as we will later see, the minimum length k' of the path must depend on the *spectral expansion* of the graph. We prove that our algorithm, known as the Expander Sampling algorithm, performs nearly as well in the case that our graph is a good expander and the edge noise is not too large. The following theorem provides our main error bound for the Expander Sampling Algorithm.

Theorem (17)**.** *Suppose vertices flip i.i.d. with probability $\frac{1}{2n}$. Then given $k = k' \geq 2560(c+q)\frac{\log(n)}{(1-\lambda_2)}$ probes per time step, and assuming $\alpha \leq \frac{1}{60(k')}$ and $k' \in o(\frac{n}{\log(n)})$, with probability at least $(1 - \frac{6}{n^c})$, the maximum number of errors that the Expander Sampling algorithm encounters before time $T = n^q$ is at most $n(\sqrt{\frac{20(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k') + 2(q + c + 1)\log(n)k'$.*

Note that M need be at most $O(q\log(n))$ for the result to hold with high probability. An alternate analysis following the proof of the above theorem gives the following stronger theorem for

a smaller time horizon if we have that $\alpha^{-1} \in \Omega(k^2)$, along with some additional properties of our algorithm. We first prove a bound on the expected number of errors at any given time step.

Theorem (18). *Assume $\alpha^{-1} > 2(k')^2 r$. Then the Exapnder Walks Sample algorithm which uses at least $k' \geq 160\frac{c\log(n)}{(1-\lambda_2)}$ probes on any one path has expected number of errors $\mathbb{E}[\epsilon_t n] \leq \frac{n}{r}(1-\alpha k')^{-1}$ for all $t \leq T$.*

And then the primary theorem:

Theorem (21). *Let G be any graph with spectral gap $(1-\lambda_2)$. Fix any integers $r, c \geq 1$, and let $k = k'r \geq 2560rc\frac{\log(n)}{(1-\lambda_2)}$ be the number of probes we allow our algorithm per time step. Further assume that the edge noise satisfies $\alpha^{-1} > 2(k')^2 r$. Then with probability at least $(1-\frac{1}{n^{c-2}})$, the maximum number of errors that Expander Sampling algorithm accumulates at any time ever before T is less than $\frac{n}{r(1-\alpha k')}(1+\sqrt{8c\log(n)})$, where $T = (\frac{\mathbb{E}[\epsilon_t n]}{k})^2 \geq (\frac{n}{k^2})^2$*

2 Noiseless Probes

2.1 The model

We first introduce our model, which can be seen as a streming variation of the model examined in [3]. Our model however only allows edge probes, and not vertex probes. Furthermore, we generalize to arbitrary graphs beyond the grid, which was the graph for which the main results of [3] hold.

Fix a graph G with vertices $v_1, \ldots, v_n$. Each vertex has a value in $\{0,1\}$ at every discrete time step t, which we denote $v_{i,t}$. Our goal is to maintain a hypothesis of the value of $v_{i,t}$, which we will denote by $h_t(v_i)$. We assume that at time $t = 0$ we start with $n\epsilon_0$ errors for some fixed $0 \leq \epsilon_0 < 1$ (our hypothesis is wrong on an ϵ_0 fraction of the vertices). The model is then as follows:

- At time $t = 0$, there are exactly $n\epsilon_0$ vertices v_i such that $v_{i,0} \neq h_0(v_i)$ has size $n\epsilon_0$
- At the start of every time step, each vertex is flipped by nature uniformly and i.i.d. with probability β, and otherwise remains unchanged with probability $1-\beta$.
- At every time step we are allowed to probe k edges, for some fixed k.
- For each edge that we probe, we are told whether the vertices on the edge agree or disagree. In other words, we are told the `xor` ($\oplus$) of the two vertices.
- The goal is to maintain a hypothesis $h_t(v_i)$ of $v_{i,t}$ such that the ℓ_1 distance $\|h_t(G) - G_t\|_1$ is small.

Thus at the start of any time step t, every vertex is flipped with some probability by nature, after which the value $v_{i,t}$ is determined for all $v_i \in G$. Then we probe k edges, and after this we define out hypothesis $h_t(v_i)$. Generally we assume k to be a constant, thus it is natural to work with probabilities for nature flipping vertices that result in a constant number of changes per time step.

If, on the other hand, the expected number of changes per time step is on an order larger than k, the problem clearly becomes intractable (this can be seen easily from Theorem 1).

We begin by proving a general lower bound on expected the number of errors accumulated by any algorithm in this model.

Theorem 1. *In the above model where vertices are flipped by nature with probability β, any algorithm which uses at most k probes on every time step never accumulates less than $\beta\frac{n^2}{4k} - \beta^2\frac{n}{2k}$ errors at any time step in expectation.*

Proof. We first consider, given any observation set of probes, which output hypothesis for the labels of the vertices which has the lowest expected error. Now since when probing an edge we gain only information about the two vertices in question, and since nature flips vertices independently, it follows that conditioning on an edge observation can only change the probabilities of the values of the two vertices in question. So the very best any hypothetical algorithm could do is always correcting the value of any vertex it sees in an edge probe.

Now suppose that the probes made by any k-algorithm are ordered by any time t. Let S_i be the set of vertices that are seen in the i-th to last step. Note again that each probe shows the algorithm at most 2 vertices, any algorithm can see at most $2k$ vertices at a given time step. Then the expected number of errors is at least :

$$\beta\sum_{i=1}^{t}|S_i|i - \Gamma$$

Where Γ is the expected number of vertices that flip twice in the sets $S_1, \ldots, S_{\lceil n/(2k)\rceil}$ since the algorithm viewed them last. Regardless of Γ, this sum is minimized by having $|S_i| = 2k$ for $i = 1, 2, \ldots, \lceil n/(2k)\rceil$. Thus the expected number of errors is at least

$$\geq 2k\beta\sum_{i=1}^{\lceil n/(2k)\rceil} i - \Gamma = 2k\beta\Big(\frac{\lceil n/(2k)\rceil(\lceil n/(2k)\rceil + 1)}{2}\Big) - \Gamma \geq \beta\frac{n^2}{4k} - \Gamma$$

Now note that the probability that a vertex flips twice in $\frac{n}{2k}$ steps is less than β, thus it follows $\Gamma < \frac{n}{2k}\beta^2$, so

$$\beta\frac{n^2}{4k} - \Gamma \geq \beta\frac{n^2}{4k} - \beta^2\frac{n}{2k}$$

■

Corollary 2. *If $\beta^{-1} \in \Theta(n)$ then the for any algorithm a lower bound for the expected number of errors at any time step is $\Omega(n/k)$. In particular, if $\beta = \frac{1}{2n}$, the expected number of errors is at least $\frac{n}{8k} - 1$.*

Proof. Follows from Theorem 1. ■

Remark 1. *Following from Corollary 2, for the rest of this paper we will specify to $\beta = \frac{1}{2n}$. The bounds given in the paper can be easily generalized however to any other value of β that differs by a constant factor.*

We introduce now the Trailing k walks algorithm. First we provide some justification for the construction of our algorithm algorithm, along with some intuition.

2.2 Intuition

In this model, observe the following fact: if we know that our hypothesis for a vertex v_i at time t is correct with certainty, then if v_j is any neighbor of v_i then we can probe the edge (v_i, v_j) and, along with the information $h_t(v_i)$, we can correctly determine the value of v_j. Thus if we know the value of one vertex with certainty we can determine the values of all those around it with certainty. Exploiting this fact, we could take a deterministic walk where we fix the neighbors of every vertex we see on the walk with certainty. Meaning if C is a hamiltonian cycle in G (or a cycle that visits all vertices that has length $n + O(1)$), then we can just walk down the cycle in order one step at a time fixing everything we see. If we are allowed to probe k edges at a time, we can take k deterministic walks at a time, traversing all the vertices of G in the order specified by C and fixing everything. There are only two (fatal) problems:

1. If we start on a vertex v that we are wrong about, then we are in trouble since we will change all vertices on the path to be incorrect.

2. If, in the middle of the step between v_i and v_j, either v_i or v_j is flipped then we will also be in trouble. To avoid this we then have to determine whether v_i was flipped at every time step first. To do this we may, for instance, probe the vertex before v_i to see if it agrees with what we expect. But even in this case:

3. Worstly, if both vertices are flipped (v_i and the one before it) on the same time step t, then even if we check the last edge we probed before we move on to the new one to see if any changes are made, then we may continue on making errors thinking that we are correct and that nothing changed.

Note that the probability of case 1 depends on how many errors there are when we start our algorithm, case 2 occurs in expectation once every $2n$ steps of a single path, and case 3 occurs in expectation once every $4n^2$ steps of a single path. So we need to avoid these cases.

We solve this problem by keeping track of a trailing set of "correct" vertices which we have already fixed. At every time step, we probe the edges between all these verices and reassign them the value that makes the most sense given their value in the last step. Then, and only then, do we probe the next vertex in the path and set it's value to agree with the one we last probed (which was part of our trailing correct set). On the next time step we add the new vertex to the set and subtract the oldest one from this set.

2.3 The k-Trailing Walks Algorithm

Let $v_{i,t}$ be the actual value of v_i at time t, and let $h_t(v_i)$ be our hypothesis. We assume we know the starting state with probability $1 - 1/\epsilon$, thus $h_0(v_i) \neq v_{i,0}$ with probability at most $\epsilon < 1$ for all $v_i \in G$. This means we start with ϵn errors.

Our algorithm is as follows. We fix a cycle that goes through all the vertices of G that we will traverse. We will later show how to generalize to graphs which do not contain such a cycle (see Theorem 9). As we traverse this cycle we probe the edges in it and fix the vertices along the way to agree both the edge observations and our prior hypothesis. This process was described

earlier. However, we make two crucial changes. Firstly, instead of traversing starting at one place, we traverse starting from k' places on the cycle *simultaneously*. Thus at every time step we move down the cycle at some uniform rate starting from k' distinct positions. We call each independent traversing process a *thread* of the algorithm (in the sense of multithreading). Secondly, for every thread, we keep track of a *trailing set* of the last r vertices that that thread saw. Before attempting to fix the next vertex in line for a given thread, that thread will first probe all the edges in its trailing set and update its hypothesis for the entire trailing set in tandem. Since probing all edges in a path forces the set of vertices to take on one of two possible assignments (determined uniquely by whether or not we declare any one vertex to be 1 or 0), we choose to update our hypothesis to be the assignment, out of the two, that has the least number of changes from our prior hypothesis.

So let C be a cycle such that every vertex $v \in G$ appears in C at least once. Order the vertices of the cycle $v_0, v_2, \ldots, v_{m-1}, v_1 \ldots$ where $m = |C|$. For now we assume $m = n$, and we will later show how to deal with graphs which do not contain a Hamiltonian cycle. Our algorithm will be given as input a parameter k', the number of threads, positive odd integer r, which determines the size of our trailing set, and a integer s which determines how many steps on the path we will take on every step. Thus our algorithm will use a total of $k'(r+s)$ probes every time step, where k' is the number of threads, r the size of the trailing set, and s the number of steps taken forward on every step. First several definitions useful in describing our algorithm are given, and then the algorithm is stated formally below.

Definition 1. *We call the Trailing Walks algorithm that uses $k = k'(r+s)$ probes, consisting of k' threads each with a trailing set of size r and each which take s steps on every time step a (k', r, s)-Trailing Walks Algorithm.*

Definition 2. *We say that a thread T_j is good at time t if $h_t(S_j)$ is entirely correct and contains no errors. We say that a thread is bad if it is not good. We say that a thread flips at time t if it goes from being good to bad on time t.*

Definition 3. *We say that a thread T_j is at the position v_i at time t, denoted $F_t(T_j) = v_i$, if v_i was the last edge on the path probed by Thread T_j at time t.*

Algorithm 1 k-Trailing Walks (initialization $t = 1$)

Input: (k', r, s), cycle C, and initial hypothesis h_0

Set $h_1 \leftarrow h_0$
for $i = 1, 2, \ldots, k'$ **do**
    Uniformly sample $\sigma(i) \underset{U}{\leftarrow} \{0, 1, \ldots, |C| - 1\}$.
end for
for $j \in \{1, \ldots, k'\}$ **do**
    Initialize $S_j \leftarrow \emptyset$.
    Set $F_1(T_j) \leftarrow v_{\sigma(j)}$.
    for $i = \sigma(j), (\sigma(j) + 1), \ldots, (\sigma(j) + r + s \mod |C|)$ **do**
        Probe the edge $e \leftarrow (v_i, v_{i+1})$ and pick $x \in \{0, 1\}$ so that $h_0(v_i) \oplus x = e$.
        Update $h_1(v_{i+1}) \leftarrow x$.
        Set $S_j \leftarrow S_j \cup \{v_i\}$.
        if $|S_j| > r + 1$ **then**
            $S_j \leftarrow S_j \setminus \{v_{i-(r+1) \mod |C|}\}$
        end if
        $F_1(T_j) \leftarrow i$
    end for
end for

Algorithm 2 k-Trailing Walks (time t)

for $j \in \{1, \ldots, k'\}$ **do**
    Initialize $S_j^t \leftarrow S_j^{t-1}$ and $E \leftarrow \emptyset$ and $h_{t+1} \leftarrow h_t$
    for $(v_i, v_{i+1}) \in E(S_j)$ **do**
        Probe the edge $e \leftarrow (v_i, v_{i+1})$.
        Add $E \leftarrow E \cup \{e\}$.
    end for
    Let I_1, I_2 be the two possible assignments to S_j^t which agree with E.
    Set $h_t(S_j^t) \leftarrow \arg\min_{I \in \{I_0, I_1\}} \|h_{t-1}(S_j^t) - I\|_1$.
    for $i = F_t(T_j), F_t(T_j) + 1, \ldots, F_t(T_j) + s \mod |C|$ **do**
        Probe the edge $e \leftarrow (v_i, v_{i+1})$ and pick $x \in \{0, 1\}$ so that $h_t(v_i) \otimes x = e$.
        Update $h_{t+1}(v_{i+1}) \leftarrow x$.
        $S_j^t \leftarrow S_j^t \cup \{v_i\}$.
        if $|S_j^t| > r + 1$ **then**
            $S_j^t \leftarrow S_j^t \setminus \{v_{i-(r+1) \mod |C|}\}$
        end if
        $F_t(T_j) \leftarrow i$
    end for
end for

Remark 2. *Note that if one of our threads starts on an error, then until it is flipped into a good thread we can assume that it will only ever create errors. We call such a thread a bad thread (a thread that is wrong on every one of it's trailing set hypothesis). The number of errors such a thread will cause is the distance between the thread's starting location in C and the next thread behind it that is correct, since a correct thread trailing behind a bad thread will correct the errors it made when it reaches them.*

Remark 3. *In line 8 of Algorithm 2, we pick the wrong assignment for the trailing set S_j only when more than half of the $|S_j| = r$ vertices are flipped by nature in a single time step.*

2.4 Analysis of Algorithm 2

Lemma 3. *When running the k-walks algorithm for $k = k'(r+s)$, the probability $\frac{1}{T}$ that at least one thread goes from being good to bad at any time $t \geq r$ is at most*

$$\frac{1}{T} \leq k'(\frac{2}{n})^{(r+1)/2}$$

Proof. Any one of the k' threads goes bad only if more than half the vertices in the trailing set flip in one time step. Since each trailing set consists of r distinct vertices which all flip with independent probabilities, the probability that this occurs for any one thread is given by:

$$\sum_{j=(r+1)/2}^{r} \binom{r}{j} \left(\frac{1}{2n}\right)^j (1-\frac{1}{2n})^{r-j} \leq 2^{r-1}(\frac{1}{2n})^{(r+1)/2} \leq (\frac{2}{n})^{(r+1)/2}$$

Using the union bound, the probability that at least one thread flips is at most k' times this above bound, which gives the desired inequality. ■

Now given that it is unlikely that any of our threads flip on a given time step, we now analyize the probability that we begin on an error and, if so, how many errors this may cause us. Let $B(k', \epsilon)$ be the binomial distribution (this gives the distribution over the number of starting bad threads). Let $E \subseteq \{0, 1, 2, \ldots, |C|-1\}$ be the subset of error vertices with expected size $n\epsilon$. Lemma 5 will demonstrate that it suffices to generate E by adding every vertex in the path to E i.i.d. with probability ϵ. Let $Z_{k',\epsilon}(E)$ be a random variable, as a function of E, that is sampled in the following way.

1. First pick k' values uniformly from $\{0, 1, 2, \ldots, |C|-1\}$. Call them $a_1, a_2, \ldots, a_{k'}$.
2. Set $d_i = a_{(i+1) \pmod{|C|}} - a_i \pmod{|C|}$ for $i = 1, 2, \ldots, k'$.
3. Define $D = \{d_i \mid a_{i+1} \in E\}$.
4. Return $Z_{k',\epsilon} = \sum_{d_i \in D} d_i$.

Lemma 4. *Suppose that before time T, no thread in our algorithm flips. Then if $E \subset V(G)$ is the set of bad vertices at time 0, then the random variable $Z_{k',\epsilon}(E)$ is an upper bound on the number of errors that our algorithm makes due to bad threads at any time $t \leq T$.*

Proof. If no thread has flipped yet, then we can assume that all threads that were bad initially stay bad, and vice-versa for good threads. Since the only errors that we contribute are the errors due to bad threads, we first need only identify the distribution over initial bad threads, and then determine how many errors can be attributed to each. A thread is bad if it starts in the set E as defined above, and since we choose the starting vertices of each thread uniformly at random then the distribution over initial bad threads is given by steps 1. and 2. of the algorithm for $Z_{k',\epsilon}(E)$.

Now for each bad thread T_i, the number of errors that are due to it at any time step is at most the number of vertices between it and the next good thread behind it. This is because all vertices that T_i visits will become errors, but they will only remain errors until they are fixed by the next good thread trailing behind T_i. Now if the next thread behind T_i, say T_j, is also bad, then we only attribute the errors on the vertices between T_i and T_j to the thread T_i. We do this since the total number of errors made by a sequence of consecutive bad threads is still the distance between the farthest thread and the first good thread behind it, which is equivalently the sum of the distances between each bad thread and the thread behind it, which is precisely how $Z_{k',\epsilon}(E)$ is constructed. Note that since vertices may be repeated, we may overcount a vertex that lies several times on the path between a good and bad thread. Thus $Z_{k',\epsilon}$ is a strict upper bound in this case.

■

Lemma 5. *The distribution of $Z_{k',\epsilon}(E)$ does not depend on the initial set E but only only the size of E.*

Proof. This is easy to see, since the k' initial values were chosen uniformly at random. ■

Lemma 6. *With probability no more than $\texttt{Poly}(\frac{1}{n}, \frac{1}{(k')})$ will we ever have $Z_{k',\epsilon} > 4(\log(k))^2 n\epsilon$. Furthermore, with probability at least $(1 - k'\epsilon)$ we have $Z_{k',\epsilon} = 0$.*

Proof. It suffices then to consider E generated by i.i.d. adding each vertex into E with probability ϵ. First we examine the binomial portion of the R.V. to bound the number of a_i's that begin in the set E. This is given by the binomial distribution $B(k', \epsilon)$. Using Chernoff bounds, it follows.

$$\Pr\Big[B(k', \epsilon) > (\lambda + 1)k'\epsilon\Big] \leq e^{-\frac{\lambda k'\epsilon}{3}}$$

Now let $a_1, \ldots, a_{k'}$ be the random positions in $\{0, 1, 2, \ldots, m-1\}$ that our algorithm choses, and randomly mark $B(k', \epsilon)$ of them, and call that set A. We analyze the sum $\sum_{a_i \in A} |a_i - a_{i-1 \pmod m} \pmod m|$. An upper bound can be placed on this by a balls and bins argument. First split the cycle $|C|$ into k' equally sized intervals of length $\frac{m}{k'}$. Let $X_1, \ldots, X_{k'}$ be random variables denoting which each interval the a_i's are placed into. Let $f(X)$ be the number of empty bins as a function of these R.V.'s. We now construct the Doob martingale, giving:

$$B_i = \mathbb{E}_{X_{i+1}, \ldots, X_{k'}}\Big[f(X) \mid X_1, \ldots, X_i\Big]$$

Fact 1. *The above construction $B_1, \ldots, B_{k'}$ is always a martingale.*

Clearly $|B_i - B_{i-1}| \leq 1$. Thus, using this bounded difference property, we can apply Azuma's inequality to yield:

$$\Pr[f(X_1,\ldots,X_{k'}) - \mathbb{E}[f(X_1,\ldots,X_{k'}] > \epsilon)] \leq e^{-\frac{2\epsilon^2}{k'}}$$

Now note that the expected number of empty bins is $\mathbb{E}[f] = k'(1-\frac{1}{k'})^{k'}$ when there are k' balls and bins. Thus with probability at least $1-\frac{1}{k'}$, we never have more than $k'(1-\frac{1}{k'})^{k'} + \sqrt{2k'\log(k')}$ empty bins. Let E be the number of empty bins. Now the empty bins are uniformly distributed, so the probability than any one bin is empty is $(1-\frac{1}{k'})^{k'} + \sqrt{\frac{2\log(k')}{k'}} \leq \frac{1}{e} + \sqrt{\frac{2\log(k')}{k'}} \leq \frac{1}{2}$ for reasonable values of k'. Thus the probability that ℓ consecutive bins are empty is at most $k'2^{-\ell}$. Thus with probability at least $1-\frac{1}{k'}$ there are no more than $2\log(k')$ consecutive empty bins, and we have $Z_{k',\epsilon} \leq (\lambda+1)\epsilon n\log(k')$ where λ is as before. Now we consider two cases.

Case 1: $k\epsilon \geq 1$. In this case set $\lambda = 3\log(k')$, we conclude with probability at least $(1-\frac{1}{(k')^{k\epsilon}})(1-\frac{1}{k'})^2 \geq (1-\frac{3}{k'})$ we have $Z_{k',\epsilon} \leq \log(k)(3\log(k')+1)\epsilon n \leq 4(\log(k'))^2 n\epsilon$

Case 2: $k\epsilon << 1$, then note that $Pr[B(k',\epsilon) > 0] \leq (1-\epsilon)^{k'} \geq (1-\epsilon k')$. Thus with probability at least $(1-\epsilon k')$ we have $Z_{k',\epsilon} = 0$.

■

Observe that $E[Z_{k',\epsilon}] = k'\epsilon(\frac{n}{k'}) = n\epsilon$, since the expected distance is $\frac{n}{k'}$ and we expect there to be $k\epsilon$ bad (marked) threads. We now begin the analysis of the entire algorithm.

Theorem 7. *Let $n\epsilon$ be the number of initial errors of our hypothesis at time $t=0$, and fix any $k = k'(r+s)$, and fix $\gamma > 0, \delta \geq 1$. Then with probability at least $(1-\frac{e}{n^\gamma})(1-e^{-\frac{\delta n}{6sk'}+\log T})(1-\frac{3}{k'})$ the (k',r,s)-Trailing Walks Algorithm has no more than $4(\log(k'))^2 n\epsilon + (1+\delta)\frac{n}{2sk'}$ errors at any time at all before $t = \frac{T}{n^\gamma}$ where T is as in Lemma 1.*

Remark 4. *Note that $T \in O((\frac{n}{2})^{(r+1)/2}\frac{1}{k'})$*

Proof. We prove the result by showing that with high probability no thread flips at any time before T. Now note that the probability that at least one thread flips on any time step t is independent of the total number of errors. Moreover, it depends only on the vertices that are flipped by nature during time step t. So this probability does not depend on the number of errors at time t nor on where those errors are. So let x_t be a random variable defined as:

$$x_t = \begin{cases} 1 & \text{if at least one thread flips at time } t \\ 0 & \text{otherwise} \end{cases}$$

Then x_t is independant from $x_{t'}$ for all $t \neq t'$. Using the union bound, we know that $\mathbb{E}[x_t] \leq \frac{1}{T}$ where T is as in Lemma 3. Let $X_t = \sum_{j=1}^{t} x_j$. Note that $E[X_t] \leq \frac{t}{T}$. Using Chernoff bounds we have:

$$\Pr\Big[X_t \geq (1+\delta)\frac{t}{T}\Big] \leq \Pr\Big[X_t \geq (1+\delta)\mathbb{E}[X_t]\Big] \leq \Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^{\mathbb{E}[X_t]}$$

Now set $\delta = \frac{T}{t} - 1$, we obtain:

$$\leq \Big(\frac{e^{\frac{T}{t}-1}}{(\frac{T}{t})^{\frac{T}{t}}}\Big)^{\mathbb{E}[X_t]} \leq \Big(\frac{e^{\frac{T}{t}-1}}{(\frac{T}{t})^{\frac{T}{t}}}\Big)^{\frac{t}{T}} = \frac{e^{1-\frac{t}{T}}}{\frac{T}{t}}$$

Now if we set $t = \frac{1}{n^\gamma}T$ for $\gamma > 0$, we obtain:

$$\Pr\Big[X_t \geq 1\Big] \leq \frac{e^{1-\frac{1}{n^\gamma}}}{n^\gamma} \leq \frac{e}{n^\gamma}$$

We have proven that with probability at least $1 - \frac{e}{n^\gamma}$, no thread flips from being bad to good or vice versa at any time step $t \leq \frac{T}{n^\gamma}$. This is an lower bound on the probability that we never accumulate more than $Z_{k',\epsilon}$ errors from bad threads at time $t \leq \frac{T}{n^\gamma}$. Coupled with Lemma 6, we obtain the first term in the bound.

Now we move to analyzing the $\frac{n}{2sk'}$ term. These are the errors made by nature. Note that an error from nature can only occur if nature flips a vertex we are right about into one we are wrong about. Let

$$b_{i,t} = \begin{cases} 1 & \text{if } v_{i,t} \neq v_{i,t-1} \\ 0 & \text{otherwise} \end{cases}$$

Then these random variables are *i.i.d.*, so let $B_T = \sum_{t=1}^{T}\sum_{i=1}^{n} b_{i,t}$. Chernoff bounds give:

$$\Pr\Big[B_t \geq (1+\delta)\frac{t}{2}\Big] \leq e^{-\frac{\delta t}{6}}$$

where $\delta \geq 1$. Since we see every vertex every $\frac{n}{sk'}$ steps (whether by good or bad threads), the number of errors due to nature's flips at any time is strictly less than $B_{\frac{n}{sk'}}$. We have

$$\Pr\Big[B_{\frac{n}{sk'}} \geq (1+\delta)\frac{n}{2sk'}\Big] \leq e^{-\frac{\delta n}{6sk'}}$$

Thus the probability that we never have more than $(1+\delta)\frac{n}{2sk'}$ errors due to nature in any one round is at least $1 - e^{-\frac{\delta^2 n}{6sk'}}$. In t steps we have $t\frac{sk'}{n}$ rounds. Thus, given $t \geq \frac{n}{sk'}$, the probability that in the first t time steps we never have more than $(1+\delta)\frac{n}{2sk'}$ errors due to nature is at least

$$\big(1 - e^{-\frac{\delta n}{6sk'}}\big)^{\frac{tsk'}{n}} \geq \big(1 - \frac{tsk'}{n}e^{-\frac{\delta n}{6sk'}}\big) \geq 1 - e^{-\frac{\delta n}{6sk'} + \log t}$$

Setting $t = \frac{T}{n}$, and putting together this bound with the upper bound on making more than $Z_{k',\epsilon}$ bad thread errors, we obtain the desired result in the theorem ■

Corollary 8. *If there are no initial errors, meaning $\epsilon = 0$, then setting $k' = 1$ we have $k = r + s$. Then for $\delta > 1$, with probability at least $(1 - \frac{e}{n})(1 - e^{-\frac{\delta n}{6s} + \log(T)})$ we never obtain more than $(1+\delta)\frac{n}{2s}$ errors at any time at all before $\frac{T}{n} \in O(n^{(r-1)/2})$*

Now recall that during the above analysis we assumed $|C| = n$. In other words, we assumed that the graph in question was Hamiltonian. This, however, is a rather strict requirement and not realistic in practice. However since nature flips vertices independently from their neighbors, if we partition G into cycles and run the k-Trailing Walks Algorithm on each cycle independently, distributing the relevant probes in amounts proportional to the respective sizes, then we can still apply the bounds from Theorem 7. The following Theorem formalizes that this can be done for all graphs.

Theorem 9. *Let G be any graph, and let $V(G) = C_1 \coprod \cdots \coprod C_d \coprod X$ be any partition of the vertices such that each C_i is a cycle. Then the errors obtained by running the k-Trailing Walks algorithm on each subgraph C_i admit the error bounds given by Theorem 7 applied to each subgraph, plus at most the additional cost $|X|$ at any time step.*

Proof. The errors in each C_i are independent of each other, so running the k-Trailing Walks Algorithm on each result in independent error bounds, so Theorem 7 can be applied to each as desired. Since we do not attempt to fix the remaining vertices X, the additional cost $|X|$ can be incurred, but clearly no more due to neglecting X. ■

We now state a simplified version of prior results which demonstrates that the desired error bound holds in expectation. While weaker than Theorem 7, it aids in giving intuition for where the asymptotic bounds in Theorem 7 come from.

Theorem 10. *Let $n\epsilon$ be the number of initial errors of our hypothesis at time $t = 0$, and fix any $k = k'r$. Then the expected number of errors of the trailing k-walks algorithm at any time step is $n\epsilon + \frac{n}{2k'}$ errors at any time $t \leq T \approx \frac{1}{k'}O(n^{\lfloor \frac{r}{2} \rfloor})$, where T is as in Lemma 1.*

Proof. Our algorithm starts k' threads on uniformly random vertices in C, each having probability ϵ of starting on an error. So in expectation, only $k'\epsilon$ threads will start in an error. For each thread that does not start in an error, we expect that it will not flip into an error thread in the first $T = O(n^{\lfloor \frac{r}{2} \rfloor})$ steps, since more than half of the trailing set S_j for a thread must be flipped in one time step in order for a good thread to become bad and vice-versa. Similarly, we do not expect bad threads to flip into good one's during this timeframe, thus we can assume that bad threads stay bad and good threads stay good for $t \leq T \approx \frac{1}{k'}O(n^{\lfloor \frac{r}{2} \rfloor})$.

Now fix any bad thread, say it starts at v_i. Then that thread will *incorrectly* set the hypothesis values for each of $v_i, v_{i+1}, \ldots, v_{i+\tau}$ after τ time steps. However, once the good thread that started at a vertex v_j closest behind v_i (of all other good threads) reaches v_i, it will fix it to be good again. Thus the number of vertices that can be incorrect at any given time because the initial bad thread made them so is the number of time steps it takes for the previous good thread to reach v_i and fix it, which $i - j$. Now there are n vertices in the cycle, and we assume that the cycle has length n (or is very nearly hamiltonian), and we start k' threads uniformly at random in this cycle. Thus we expect that $i - j = \frac{n}{k'}$, thus in expectation a bad thread creates at most $\frac{n}{k'}$ errors. Note that it is possible that m bad threads be started consecutively, in which case we expect that $i - j = m\frac{n}{k'}$ if the farther one starts at v_i. However this factor of m is already taken into account since we are multiplying by the expected number of bad threads, which is $\epsilon k'$, and this value is independent of

how far apart any two consecutive threads are. Since we expect there to be $\epsilon k'$ bad threads, we obtain an expected $\epsilon k' \frac{n}{k'} = n\epsilon$ errors due to bad threads.

Finally, node that in expectation, at any time step all vertices have been probed at least once in the last $\frac{n}{k'}$ time steps. Since the bad threads will only create errors, we need not account for nature changing the values of the bad threads (since this would only remove our errors). So we have at most n vertices which have not been probed in $\frac{n}{k'}$ time steps, each with $\frac{1}{2n}$ chance of flipping on each time step. Thus we expect no more than $\frac{n}{2k'}$ additional total changes of good vertices into bad one's as a result of nature's random perturbations. Together with the errors from the bad threads, we obtain the desired $n\epsilon + \frac{n}{2k}$ expected errors at any time $t \leq T$. ■

3 Noisy Probes

We now consider a generalization of the previous model, namely where there is *noise* in our edge probes. The model is as follows. We have a graph G with $n\epsilon_0$ initial errors, and nature flips the value of each vertex uniformly with probability $\frac{1}{2n}$. For the purposes of this analysis, we assume $\epsilon_0 = 0$, however the results of this section can be similarly obtained by setting ϵ_0 to be the expectation from Theorem 18. Further suppose that on every edge probe we have an α probability of an error for $0 < \alpha < .5$. Meaning, for each edge we probe e, with probability $1 - \alpha$ we receive the correct $\neg$XOR of the vertices in the edge as in the last model, and with probability α we recieve the incorrect value. Thus, our entire model can be summarized as follows:

- At time $t = 0$, there are exactly $n\epsilon_0$ vertices v_i such that $v_{i,0} \neq h_0(v_i)$ has size $n\epsilon_0$
- At the start of every time step, each vertex is flipped by nature uniformly and i.i.d. with probability $\frac{1}{2n}$, and otherwise remains unchanged with probability $1 - \frac{1}{2n}$.
- At every time step we are allowed to probe k edges, for some fixed k.
- For each edge that we probe, we are told whether the vertices on the edge agree or disagree with probability $1 - \alpha$, and we are told the opposite of this with probability α.
- The goal is to maintain a hypothesis $h_t(v_i)$ of $v_{i,t}$ such that the ℓ_1 distance $\|h_t(G) - G_t\|_1$ is small.

Remark 5. *As we will see shortly, the following analysis does not utilize the fact that vertex values are being flipped uniformly at random specifically, but only that we can place a bound stating that no more than logarithmically many vertex values are flipped with high probability. Thus, the following analysis, and in particular Lemma 14, will hold for any distribution over vertex value perturbations such that a high probability bound can be placed on the number of flips that occur in a given time step.*

3.1 Difficulties with obvious algorithms

In this case, if we are to use the k-Trailing Walks Algorithm, observe that each thread will flip from being good to bad if only a single one of it's edge probes is noisy. Thus, even for small values of α (such as if $\alpha^{-1} \in \Omega(n)$), our algorithm will quickly degenerate as nearly all threads become bad.

A natural question to ask then is can we simply probe random vertices? In other words, a natural algorithm is to pick a vertex v_i uniformly at random, probe all adjacent edges, and assign our hypothesis for v_i to be the value that makes the most sense given our observations – meaning the value that disagrees with less than half of the edge observations. The problem with this algorithm is the same problem that we discussed in the noiseless case. The probability of incorrectly not fixing a bad vertex is the probability that more than half the neighbors of the probed vertex are bad conditioned on the fact that we have probed a bad vertex. Now if the errors are distributed uniformly at random, this value is easily bounded. But since we are more likely to make an error if we are around other errors, as time goes on this conditional probability becomes larger and larger, since errors begin to cluster. This result is unacceptable, and makes the algorithm impossible to analyze.

The second issue is the problem of connected components of errors. If there is a large connected component of errors, then we can only fix it by probing on the boundary. In the grid example, for instance, a connected component of errors can only be fixed if we probe on its boundary, which can have size proportional to the square root of the number of errors!

3.2 Primer on Expander Graphs

The algorithm presented in this section will make heavy use of bounds on the size of the *spectral gap* of our graph. The smaller the spectral gap, the more probes we will need to obtain reasonable bounds. Graphs with large or constant-sized spectral gaps are known as *expanders*. We assume some familiarity with spectral theory, and will present now the fundamentals that will be employed in the analysis of our algorithm.

Definition 4. *Given a graph G and any subset $S \subset G$, define the boundary ∂S of S to be the set $\partial S = \{(u, v) \in E(G) \mid u \in S, v \in G \setminus S\}$. We then define the edge expansion, or Cheeger Constant, $h(G)$ of G as:*

$$h(G) = \min_{0<|S|\leq\frac{|G|}{2}} \frac{|\partial S|}{|S|}$$

Definition 5. *We say that a graph G with n vertices is an expander graph if $h(G) \in \Omega(1)$.*

Definition 6. *Given a d-regular graph G with adjacency matrix A, order the eigenvalues of A $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Then the spectral gap of G is defined to be $d - \lambda_2$.*

Lemma 11. *A d-regular graph G is an expander if and only if it's spectral gap is a constant.*

Proof. The proof follows from the well known *Cheeger Inequality*, which states:

$$\frac{1}{2}(d - \lambda_2) \leq h(G) \leq \sqrt{2d(d - \lambda_2)}$$

Thus $(d - \lambda_2) \in \Omega(h(G))$, and $h(G) \in \Omega(d - \lambda_2)$, which completes the claim. ■

The following theorem will be crucial to our analysis in section 3.4.

Theorem 12 (Expander Walk Sampling 25)**.** *Let G be a graph with normalized second largest eigenvalue λ_2. Let $f : V(G) \to \{0, 1\}$ be any function, and let $\mu = \frac{1}{n}\sum_{v_i \in V(G)} f(v_i)$ be its mean. If $Y_0, Y_1, \ldots, Y_{k'}$ is a k'-step random walk starting at a random vertex Y_0, then we have for all $\gamma > 0$:*

$$\Pr\Big[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu > \gamma\Big] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}} = p$$

and

$$\Pr\Big[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu < -\gamma\Big] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}}$$

Example 1. *The 4-regular discrete torus (i.e. the Cayley Graph of the finite group $\mathbb{Z}_n \times \mathbb{Z}_n$) with n^2 vertices has spectral gap $(1 - \lambda_2) \in O(\frac{1}{n}) = O(\frac{1}{\sqrt{|\mathbb{Z}_n \times \mathbb{Z}_n|}})$.*

Example 2. *The $(n-1)$-regular complete graph K_n on n vertices has spectral gap $((n-1) - \lambda_2) \in O(\frac{1}{n})$*

3.3 Expander Walks Algorithm

In Section 3.1 we remarked that in order to reliably fix errors in the noisy model we cannot rely on ever being *very sure* of the correctness of our hypothesis for a given vertex at any time step. This is a direct result of a non-trivial probability of noise on every probe. Thus, we must somehow obtain a uniform sample of the vertices in the graph of a sufficiently large size such that with high probability at least half of such vertices are not errors. This is a necessarily condition in order to update our hypothesis correctly, since even in a non-noisy sample the set of vertices observed in any set of probes may still take on one of two possible sets of values. Even to enforce this, such a sequence of probes must be made in path in order to relate the values of successive vertices together. But by the very nature of a path, the vertices on it are forced to be close to each other, and therefore not uniformly chosen! Thus we must take a longer path if we want the vertices on it to be, on average, uniformly distributed. Luckily, there are classes of graphs for which it suffices to take *logarithmically* long paths to obtain such a result. The property of the graph which will determine how long such a path must be in order to be uniform is known as the *expansion* of the graph. Before we introduce this notion, we will first present out algorithm.

Definition 7. *At time step t, let ϵ_t be the fraction of bad vertices (errors) in the graph. In other words, at time t there are $n\epsilon$ vertices v_i such that $v_{i,t} \neq h_t(v_i)$.*

Our algorithm uses $k = k'r$ probes on every time step, where k' will be the length of each random path sampled and r will be the number of such paths sampled on each time step.

Expander Walk Sampling Algorithm:

1. On each time step, for $j = 1, 2, \ldots, r$:
2. Pick a random vertex v_i, and take a k' step random walk starting from v_i, proving each edge along the way.

3. Let $\vec{Y} = Y_0, Y_1, \ldots, Y_{k'}$ be the (not necessarily unique) vertices on the walk. Assuming that all edge probes are correct, assign $h_t(\vec{Y})$ to be the value (out of the two possible) which minimizes the ℓ_1 distance from our prior hypothesis $h_{t-1}(\vec{Y})$.

Algorithm 3 Expander Walk Sampling

Input: r, k', G

Initialize paths $P_1, \ldots, P_r = \emptyset$
for $j = 1, 2, \ldots, r$ **do**
    Uniformly sample $v_{1,j} \xleftarrow[U]{} V(G)$.
    for $i = 2, \ldots, k'$ **do**
        $v_{i,j} \xleftarrow[U]{} N(V_{i-1,j})$
    end for
    $P_j \leftarrow \{v_{1,j}, v_{2,j}, \ldots, v_{k,j}\}$
end for

Output: $P_1, P_2, \ldots, P_r$

Algorithm 4 Expander Hypothesis Update (time t)

Input: Paths $P_1, P_2, \ldots, P_r$ and hypothesis $h_{t-1}(G)$

$h_t \leftarrow h_{t-1}$
for $i \in \{1, 2, \ldots, r\}$ **do**
    Probe all edges $E_i \in P_i$
    Let I_1, I_2 be the valid assignments to P_i given E_i.
    $h_t(P_i) \leftarrow \arg\min_{I \in \{I_0, I_1\}} \|h_{t-1}(P_i) - I\|_1^1$
end for

Output: $h_t(G)$

The above algorithm uses $k = rk'$ probes per time step. Suppose we have a d-regular graph G with normalized second largest eigenvalue λ_2. We will show that if G is a good expander then $k' \in O(\frac{\log(n)}{(1-\lambda_2)})$ is sufficient to obtain an expected number of errors $E[n\epsilon_t] \leq \frac{n}{2r}(1-\alpha)^{-k}$, and that $n\epsilon_t$ is highly concentrated around it's mean for $t \in O(n^2)$.

3.4 Main Analysis

In the following two sections we will conduct the principle analysis of our algorithm that will justify our main theorems, Theorem 17 and Theorem 21. First we prove two simple technical Lemmas. Afterwards, we introduce Lemma 15, which will allow us in the analysis that follows to assume that at every time step we have fewer than $\epsilon_t \leq 1/4$ errors in the graph with high probability. Using similar techniques as in this Lemma, we prove a bound on the number of errors made by our algorithm for polynomially many time steps in Theorem 17.

Afterwards, we consider looser bounds in the case that we allow fewer probes. In Theorem 18, we will prove that the expected number of errors of our algorithm at any time step before $T = n^2$ is at most $\frac{n}{r}(1 - \alpha k')^{-1}$. Using this fact, we will then observe that if we have more than this expected number of errors, then *in expectation* the number of errors will decrease on the next time step. Technically, this means that the number of errors we have exceeding this expected value is a *supermartingale*. Using this fact, we apply Azuma's inequality for supermartingales to obtain our main Theorems: Theorem 20 and Theorem 21.

Lemma 13. *Suppose we have $0-1$ random variables $x_1, \ldots, x_n$ that are not i.i.d. but $Pr[x_i = 1|X] \leq p$ where X is any set of observations of the values x_j's not including x_i. Then we can apply the standard Chernoff bounds, using $\Pr[x_i = 1] = p$, to bound the probability that $\sum x_i$ is large.*

Proof. We prove $Pr(\sum_{t\in T'} x_t > c) \leq Pr(\sum_{t\in T'} Y_t > c)$ where Y_i are Bernoulli(p). But this is easy to see, since $Pr(\sum_{t\in T'} x_t > c) = \sum_{|I|\geq c} Pr[x_i = 1,\ \forall i \in I] \leq \sum_{|I|\geq c} p^{|I|}(1-p^{|I|}) = Pr(\sum_{t\in T'} Y_t > c)$, which completes the claim.

■

Lemma 14. *With probability at least $1 - \frac{1}{n^c}$, nature never flips more than $4(c+q)\log(n)$ at any time before $T = n^q$.*

Proof. Let E_t be the number of vertices flipped by nature at time t. Note that $E_1, \ldots, E_T$ are independent with expectation $\frac{1}{2}$. Then by Chernoff bounds, using $\delta = 6(c+q)\log(n)$:

$$\Pr\Big[E_t \geq \frac{1}{2}(1 + 6(c+q)\log(n))\Big] \leq e^{-(c+q)\log(n)} = \frac{1}{n^{c+q}}$$

Since each E_T are independent, and since $\frac{1}{2}(1 + 6(c+q)\log(n)) < 4(c+q)\log(n)$, it follows via the union bound that with probability at least $1 - \frac{1}{n^c}$ the maximum, taken over all $t \leq T = n^q$, number of vertices that nature flips in a given time step is less $4(c+q)\log(n)$.

■

Lemma 15. *Given $k' \geq 2560(c+q)\frac{\log(n)}{(1-\lambda_2)}$ probes in every random walk, and assuming $\alpha < \frac{1}{240k'}$ and $k' \in o(\frac{n}{\log(n)})$, then with probability at least $(1 - \frac{2}{n^c})$ at no time before $T = n^q$ will we ever have more than an $\epsilon_t = \frac{1}{4}$ fraction of errors.*

Proof. First note that via Lemma 14 bounds the probability that nature flips more than $4(c+q)\log(n))$ vertices at any time step before $T = n^q$ is at most $\frac{1}{n^c}$ which is a bound we will now use.

Supposing $\epsilon_t = 1/8$, we will bound the probability that we obtain $\frac{n}{4}$ errors before dropping below $\frac{n}{8}$. Now we have picked our k' large enough so that by Theorem 25, the probability that we see more than $\epsilon_t + 1/16$th errors on our path is at most $\frac{1}{n^c}$. Similarly, the probability that we see less than $\epsilon_t - \frac{1}{16}$-th errors is at most $\frac{1}{n^c}$. Thus the probability that neither of these occur on any one path is at least $1 - \frac{2}{n^c}$. Thus with probability at least $1 - \frac{2}{n^c}$, since $k' \geq 2560(c+q)\frac{\log(n)}{(1-\lambda_2)}$, if there is no noise and if $t \in T'$, then we fix at least

$$\frac{1}{16}k' - 4(c+q)\log(n) = 160(c+q)\frac{\log(n)}{(1-\lambda_2)} - 4(c+q)\log(n) > 156(c+q)\frac{\log(n)}{(1-\lambda_2)} \geq \frac{k'}{20}$$

errors after probing any one path. For any path where we fix at least this many errors, we call such a path a *successful path*. Note that we have subtracted off the maximum number of errors that nature can create on any time step (conditioned on the fact that it never makes more). Thus if we sample multiple paths on any time step we are subtracting nature's error quantity multiple times within a given time step, making our bound slightly worse, but nevertheless sufficient. Observe that after an unsuccessful path there can be at most $k' + 7(c+q)\log(n) \leq 2k'$ new errors, thus at most 40 successful paths fully removes the errors from one unsuccessful path. This fact will be shortly useful.

Now using the union bound, the probability that there is no noise on a path is at least $(1-k'\alpha)$. So we fix at least $\frac{k'}{20}$ many errors with a path sample with probability at least $(1-\frac{2}{n^c})(1-k'\alpha) \geq (1-2k'\alpha)$ (assuming $(1-\frac{2}{n^c}) > (1-k'\alpha)$, which if did not hold would only give a tighter analysis), and create at most $2k'$ errors with probability at most $2k'\alpha$. Thus, starting at any time t, we fix $2k'$ errors before any unsuccessful path with probability at least $(1-80k'\alpha)$. Now divide the range $(0,n)$ into intervals of size $2k'$, and set

$$X_t = \begin{cases} X_{t-1}+1 & \text{if the number of errors goes up one interval since the last change in interval} \\ X_{t-1}-1 & \text{if the number of errors goes down one interval since the last change in interval} \end{cases}$$

Note that we can never skip an interval in one time step since they have size $2k'$. Then X_t is a bias random walk with descent probability at least $(1-80k'\alpha)$, since by the union bound this is a lower bound on the probability that we have 40 successful paths in a row.

Now suppose at time t we have $\frac{n}{8}$ errors. Then it would take at the very least $\frac{n}{8}\frac{1}{2k'} = \frac{n}{16k'}$ increases in intervals before we could attain $\frac{n}{4}$ many errors, in other words when X_t hits $\frac{n}{16k'}$. Applying Lemma 24, if we start with $X_0 = 0$, then the probability that X_t hits $\frac{n}{16k'}$ before -1 is at most $(\frac{80\alpha k'}{1-80\alpha k'})^{\frac{n}{16k'}}$. Since $\alpha < \frac{1}{240k'}$ by assumption, the probability that we hit $\frac{n}{4}$ errors before decreasing below $\frac{n}{8}$ is at least $(\frac{1}{2})^{\frac{n}{16k'}}$, so the probability that this never happens before time $T = n^q$, where there are rn^q path samples, is at most $n^q r(\frac{1}{2})^{\frac{n}{16k'}}$. Conditioned on the fact that nature never makes too many errors on one time step before n^q (which we showed occurs with probability $\geq 1-\frac{1}{n^c}$), and given $rn^q(\frac{1}{2})^{\frac{n}{16k'}} < \frac{1}{2n^c}$ which holds as long as $k' \in o(\frac{n}{\log(n)})$, it follows that the probability that we never exceed $\frac{n}{4}$ errors is at least $(1-rn^q(\frac{1}{2})^{\frac{n}{16k'}})(1-\frac{1}{n^c}) \geq (1-\frac{2}{n^c})$, which is the desired bound.

■

Here, utilizing similar techniques as in Lemma 15, we prove a reliable error bound for our algorithm. This error bound holds for much more general distributions, where all we require is a high probability bound on the number for flips made at any time step. In the subsequent section we will present a stronger bound for the i.i.d. case which holds for a much smaller time horizon which works for smaller α, such as $\alpha^{-1} \in \Omega(k^2)$. The bound we present in this section holds for a much longer time horizon.

Theorem 16. *Suppose vertices flip according to any distribution D such that the nature never flips more than $\zeta_D \leq k$ vertices at any time step before n^q with probability at least $1-\frac{1}{n^c}$. Then given*

$k = k' \geq 2560(c+q)\frac{\log(n)}{(1-\lambda_2)}$ probes in every random walk, setting the parameter $r = 1$, and assuming $\alpha \leq \frac{1}{60(k')}$ and $k' \in o(\frac{n}{\log(n)})$, then with probability at least $(1-\frac{5}{n^c})(1-n^q 2^{-M})$ the maximum number of errors that the Expander Sampling algorithm encounters before time $T = n^q$, running on a graph where vertices flip according to D, is at most $n(\sqrt{\frac{10(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k' + \frac{\zeta_D}{k'}) + 2Mk'$.

Proof. First apply Lemma 15 to condition on the fact that we never have more than $\epsilon_t \leq 1/4$ errors, which occurs with probability at least $1 - \frac{2}{n^c}$. The proof will proceed much as in the proof of Lemma 15. Again, for any path where there is no noise, call such a path a *successful path*. Using the notation of Theorem 25, suppose that we have a $\epsilon_t \geq \mu = \sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}} + \frac{\zeta_D + 12\alpha(k')^2}{k'}$ fraction of errors at time t, and let $Y_0, \ldots, Y_{k'}$ be the vertices on the random path sampled by our algorithm. Let $f(Y_i)$ a binary function evaluating to 1 if and only if $f(Y_i)$ is an error, and 0 otherwise. Note that since by assumption we have $\alpha \leq \frac{1}{60(k')}$, it follows that $\epsilon_t < 1/4$, so we are not violating the assumption obtained by conditioning on Lemma 15 to begin with. By Theorem 25 we have

$$\Pr\Big[\sum_{i=0}^{k'} f(Y_i) < \zeta_D + 12\alpha(k')^2\Big] \leq \Pr\Big[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \epsilon_t < -\sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}}\Big] \leq \frac{1}{n^c}$$

Now utilizing the lower bound on k' we establish a similar bound:

$$\Pr[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \epsilon_t > \frac{1}{4}] \leq \frac{1}{n^c}$$

The above is a upper bound on the probability that we incorrectly assign our hypothesis for the vertices on a successful path, since this occurs only if $\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) > 1/2$, and we know $\epsilon_t < 1/4$. Thus we have established the following fact: with probability at least $1 - \frac{2}{n^c}$ we fix at least $\zeta_D + 12\alpha(k')^2$ errors if we have $1/4 > \epsilon_t \geq \sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}} + \frac{\zeta_D + 12\alpha(k')^2}{k'}$ and there is no noise on the path. Since with probability at least $1 - \frac{1}{n^c}$ nature never creates more than ζ_D errors on any time step before n^q, it follows that with probability at least $1 - \frac{3}{n^c}$ the net decrease in errors is at least $12\alpha(k')^2$ after every successful path probe when $\epsilon_t \in (\sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}} + \frac{\zeta_D + 12\alpha(k')^2}{k'}, 1/4)$.

Now we proceed as in the proof of Lemma 15. First break the interval $(0, n)$ up into sub-intervals of size $2k'$, and let

$$X_t = \begin{cases} X_{t-1} + 1 & \text{if the number of errors goes up one interval since last change in interval} \\ X_{t-1} - 1 & \text{if the number of errors goes down one interval since last change in interval} \end{cases}$$

Now note that $(k' + \zeta_D) \leq 2k'$ is an upper bound on the difference between errors on subsequent time steps, thus it is impossible to skip over intervals within one time step. Furthermore note that it less than $\frac{2k'}{12\alpha(k')^2} = \frac{1}{6\alpha k'}$ successful paths to go from one interval to the one below it. Then if we have $\epsilon_t \in (\sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}} + \frac{\zeta_D + 12\alpha(k')^2}{k'}, 1/4)$ we know that the probability that $X_t = X_{t-1} - 1$ is at

least $(1-\frac{k'}{n^c})(1-\alpha(k')\frac{1}{6\alpha k'}) \geq (1-2\alpha(k')\frac{1}{6\alpha k'}) \geq 2/3$. Thus X_t is a random walk with probability of descent at least $2/3$. Thus applying Lemma 24 with $p \leq 1/3$, supposing at time t we have $n\mu$ errors, then it follows that with probability at least $(1-(\frac{1}{2})^M)$ we decrease the number of errors below $n\mu$ before reaching $n\mu + 2Mk'$ many errors. Thus, using the union bound, it follows that with probability at least $(1-n^q 2^{-M})$ we never reach $n\mu + Mk' = n(\sqrt{\frac{10c\log(n)}{(1-\lambda_2)k'}} + \frac{\zeta_D}{k'} + 12\alpha k') + 2Mk'$ many errors before time n^q. Incorporating the $(1-\frac{3}{n^c})$ probability lower bound of fixing at least 1 net error on any successful path and the $(1-\frac{2}{n^c})$ lower bound from Lemma 15 yields the desired result.

■

Theorem 17. *Suppose vertices flip i.i.d. with probability $\frac{1}{2n}$. Then given $k = k' \geq 2560(c+q)\frac{\log(n)}{(1-\lambda_2)}$ probes per time step, and assuming $\alpha \leq \frac{1}{60(k')}$ and $k' \in o(\frac{n}{\log(n)})$, with probability at least $(1-\frac{6}{n^c})$, the maximum number of errors that the Expander Sampling algorithm encounters before time $T = n^q$ is at most $n(\sqrt{\frac{20(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k') + 2(q+c+1)\log(n)k'$.*

Proof. The proof follows directly from Theorem 16, setting $\zeta_D = 4(c+q)\log(n) < k$ as per Lemma 14, and setting $M = (q+c+1)\log(n)$, and noting then that:

$$n(\sqrt{\frac{10(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k' + \frac{\zeta_D}{k'}) + 2Mk'.$$

$$= n(\sqrt{\frac{10(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k' + \frac{4(c+q)\log(n)}{k'}) + 2(q+c+1)\log(n)k$$

$$\leq n(\sqrt{\frac{20(c+q)\log(n)}{(1-\lambda_2)k'}} + 12\alpha k') + 2(q+c+1)\log(n)k'$$

■

3.5 Further Analysis

We now explore a different set of bounds which will are stronger (given sufficiency small values of α), but will hold only to at most quadratically many time steps. Furthermore, we place a bound on the expected number of errors our algorithm has at a given time step. The main technique we will utilize in this section is Azuma's inequality for supermartingales. In order to reach this point, we first establish an upper bound on the expected number of errors of our algorithm at any given time step.

Theorem 18. *Assume $\alpha^{-1} > 2(k')^2 r$. Then conditioned on the fact that $\epsilon_t < 1/4$ for all $t \leq T = n^2$, the Exapnder Walks Sample algorithm which uses at least $k' \geq 160\frac{c\log(n)}{(1-\lambda_2)}$ probes on any one path has expected number of errors $\mathbb{E}[\epsilon_t n] \leq \frac{n}{r}(1-\alpha k')^{-1}$ for all $t \leq T$.*

Proof. We utilize Theorem 25 to prove the claim. We first analyze the performance of our algorithm if we probe a full path with no noise. Now if there is no noise in any of our probes, then our algorithm correctly assigns all vertices in $\vec{Y} = \langle Y_0, Y_1, \ldots, Y_{k'} \rangle$ if no more than half of the vertices in $\vec{Y}$ are bad at time t. Otherwise, we incorrectly assigns all vertices in $\vec{Y}$. So define the function $f(v_i)$ in the Theorem 25 to take on the value 0 if v_i is good at time t and 1 if v_i is bad at time t (recall v_i is bad at time t if $h_t(v_i) \neq v_{i,t}$). We have $\mu = \epsilon_t$ by definition. Now assuming there is no noise, we only make an error if more than half of the vertices on our path are errors. Furthermore, by Lemma 15, we will never have more than $\epsilon_t = 1/4$ with high probability. So setting $\gamma = \frac{1}{4}$ in Theorem 25 gives an upper bound on probability that we make an error in our assignment of any path (given that there was no noise on that path). Then taking at least $k' = 160\frac{c\log(n)}{(1-\lambda_2)}$ gives $p \leq \frac{1}{n^c}$, where p is as in Theorem 25. Thus the probability that we make an error on a noiseless path is at most $\frac{1}{n^c}$.

Now we consider the number of vertices that we fix given that we do not make an error. Since Y_0 is random, if we probe r paths we will expect to fix at least $r\epsilon_t$ errors at time t if we do not make an error and there is no noise on the path. If there is noise on the path, however, then we can be wrong on all of the vertices. Using the union bound, the probability that there is noise on any one path is at most $\alpha k'$. So let us compute the expected number of errors we contribute or remove at time t. Thus an upper bound for the expected number of errors we *add* or *remove* on a given time step is:

$$-(1 - \frac{1}{n^c})r\epsilon_t(1 - \alpha k') + (\alpha k')k + \frac{1}{n^c}k$$

Now nature in expectation adds 1/2 errors per time step. Thus, in order to determine the ratio of errors ϵ_t such that in expectation we neither gain nor lose errors on the next time step, we must solve the following equation:

$$\mathbb{E}\Big[\epsilon_{t+1}n - \epsilon_t n \mid \epsilon_t n\Big] = -(1 - \frac{1}{n^c})r\epsilon_t(1 - \alpha k') + (\alpha k')k - \frac{1}{n^c}k - 1/2 = 0$$

for ϵ_t, which will give the fixed point. This fixed point will be the expected number of errors $\mathbb{E}[n\epsilon_t]$ at any time step of the algorithm, as the value will be a sink towards which all other states of the system evolve towards. Then taking $\epsilon_t = \frac{1}{2r(1-\frac{1}{n^c})}(1 - \alpha k')^{-1}(1 + (\alpha k')k + \frac{k}{n^c})$ suffices to solve the equation. Thus we conclude $\mathbb{E}[n\epsilon_t] \leq n\frac{1}{2r(1-\frac{1}{n^c})}(1 - \alpha k')^{-1}(1 + (\alpha k')k + \frac{k}{n^c})$. Now we have $(1 - \frac{1}{n^c})^{-1}(1 + (\alpha k')k + \frac{k}{n^c}) \leq 2$, which occurs because $\alpha^{-1} > 2(k')^2 r$, thus we have $\mathbb{E}[n\epsilon_t] \leq \frac{n}{r}(1 - \alpha k')^{-1}$ as desired. ■

Lemma 19. *Conditioned on the fact that $\epsilon_t < 1/4$ for all $t \leq T$, then the random variable $\max\{\epsilon_t n, \mathbb{E}[\epsilon_t]\}$ is a supermartingale for $t = 1, 2, \ldots, T$.*

Proof. Consider the following expectation from the last theorem, which holds conditioned on the fact that $\epsilon_t n < 1/4$ for all $t \leq T$,:

$$\mathbb{E}\Big[\epsilon_{t+1}n - \epsilon_t n \mid \epsilon_t n\Big] = -(1 - \frac{1}{n^c})r\epsilon_t(1 - \alpha k') + (\alpha k')k - \frac{1}{n^c}k - 1/2 = 0$$

It follows directly that if ϵ_t is increased then this expectation is decreased. Thus $\mathbb{E}[\epsilon_{t+1}n|\epsilon_t n] \leq \epsilon_t n$, which is the desired property.

■

We have now concluded conclude that ϵ_t, when restricted to the interval $(\mathbb{E}[\epsilon_t], 1/4)$ is a *supermartingale*. Formally we have shown that the random variable $\min\{\max\{\epsilon_t, \mathbb{E}[\epsilon_t]\}, 1/4\}$ is a supermartingale. This motivates the following analysis. We will bound the probability that ϵ_t becomes too much larger than the expected number of total errors. We will first bound the probability that nature makes too many errors in any step. Then by the chain rule for probability, the probability that we never have too many errors is then the probability that we never have too many errors conditioned on the fact that ϵ_t never exceeds $1/4$ and nature never makes too many errors in any step, times the probability that nature never makes too many errors on any step times the probability that ϵ_t never exceeds $1/4$ (the last two events are independent).

Definition 8. *We say that a vertex v_i is an error at time t if $h_t(v_i) \neq v_{i,t}$. We say that an error v_i at time t is an error due to nature if there exists a $\tau \leq t$ such that $h_\tau(v_i) = v_{i,\tau}$, nature flipped v_i on time $\tau + 1$, and $h_{t'}(v_i) = h_\tau(v_i)$ for all $\tau \leq t' \leq t$. We say that v_i is an error due to us if it is an error that is not due to nature.*

Remark 6. *Recall from our lower bound, Theorem 1, for any algorithm that probes k edges on any time step we have $\mathbb{E}[\epsilon_t n] \geq \frac{n}{k}$. Thus the following bounds hold up to time $T \geq \frac{n^2}{k^4}$.*

Theorem 20. *Assume $\alpha^{-1} > 2(k')^2 r$. Then at any fixed time $t \leq T = (\frac{\mathbb{E}[\epsilon_t n]}{k})^2$, using $k = k'r \geq 2560rc\frac{\log(n)}{(1-\lambda_2)}$ probes, with probability at least $(1 - \frac{3}{n^c})$, the above algorithm has less than $\frac{n}{r(1-\alpha k')}(1 + \sqrt{8c\log(n)})$ errors.*

Proof. By the law of total probability, the desired probability is at least the probability that $\epsilon_t \leq \frac{n}{r(1-\alpha k')}(1 + \sqrt{8c\log(n)})$ conditioned on the fact that $\epsilon_t < 1/4$ for all $t < T$, times the probability that $\epsilon_t < 1/4$ for all $t < T$ (See Lemma 15, where we show that this latter probability is at leat $1 - \frac{1}{n^c}$).

So let $X_t = \max\{\epsilon_t n, \mathbb{E}[\epsilon_t n]\}$. Then X_t is a supermartingale by Theorem 19. Note that we can never create or remove more than k errors on any time step, since we change the values of at most k vertices in our hypothesis. To show that nature cannot create too many errors, we must now bound the probability that nature never creates too many errors and then condition on this fact. By Lemma 14, we have that nature never creates more than $\frac{7(c+1)(1-\lambda_2)}{2560rc}k \leq k$ errors at any time before T with probability at least $1 - \frac{1}{n^c}$. Conditioning on this fact, we have the bounded difference $|X_t - X_{t-1}| \leq 2k$ for all $1 \leq t \leq T$ (at most k from us and k from nature). Then for any time step T, by applying Azuma's inequality for supermartingales we obtain:

$$\Pr\Big[X_T - X_0 > \lambda\Big] \leq \exp\Big(\frac{-\lambda^2}{2(2k)^2 T}\Big)$$

Setting $\lambda = \sqrt{8T(k)^2 c\log(n)}$ for any $c \geq 1$ gives

$$\Pr\Big[X_T > \mathbb{E}[\epsilon_t n] + \sqrt{8Tc(k)^2\log(n)}\Big] \leq \frac{1}{n^c}$$

Setting $T = (\frac{\mathbb{E}[\epsilon_t n]}{k})^2$ gives

$$\Pr\Big[X_T > \mathbb{E}[\epsilon_t n] + \mathbb{E}[\epsilon_t n]\sqrt{8c\log(n)}\Big] \leq \frac{1}{n^c}$$

Using our upper bound on the expectation $\mathbb{E}[\epsilon_t n] \leq \frac{n}{r(1-\alpha k')}$ derived in Theorem 18, we obtain:

$$\Pr\Big[X_T \geq \frac{n}{r(1-\alpha k')}\Big(1+\sqrt{2c\log(n)}\Big)\Big] \leq \Pr\Big[X_T > \mathbb{E}[\epsilon_t n]\Big(1+\sqrt{2c\log(n)}\Big)\Big] \leq \frac{1}{n^c}$$

Which is the desired result. Note that we conditioned on the fact that $\epsilon_t \leq 1/4$ for all $t \leq T$, and the fact that nature never creates too many errors for any $t \leq T$, both of which hold with probability at least $(1-\frac{1}{n^c})$. Multiplying all three of these lower bounds gives the desired lower bound $(1-\frac{1}{n^c})^3 \geq (1-\frac{3}{n^c})$

■

Theorem 21. *Let G be any graph with spectral gap $(1-\lambda_2)$. Fix any integers $r, c \geq 1$, and let $k = k'r \geq 2560rc\frac{\log(n)}{(1-\lambda_2)}$ be the number of probes we allow our algorithm per time step. Further assume that the edge noise satisfies $\alpha^{-1} > 2(k')^2 r$. Then with probability at least $(1-\frac{1}{n^{c-2}})$, the maximum number of errors that the (k', r)-Expander Walk Algorithm accumulates at any time before T is less than $\frac{n}{r(1-\alpha k')}(1+\sqrt{8c\log(n)})$, where $T = (\frac{\mathbb{E}[\epsilon_t n]}{k})^2$*

Proof. By Theorem 20 the probability that error bound is exceeded at any one time step t is at most $\frac{1}{n^c}$. Using the union bound, along with the fact that $T \leq n^2$, it follows that the probability of the error bound ever being exceeded is at most $\frac{1}{n^{c-2}}$ as desired.

■

3.6 Recovering from high errors

In the prior section we derived error bounds which hold with high probability. Unfortunate, our theoretical bounds do not extend past quadratic $T \geq n^2$ time steps. Thus, we proceed in our stochastic analysis by deriving the *expected return time* to the expected number of errors given that we have exceeded it. In other words, conditioned on the fact that at time t we have a $\epsilon_t = M\mathbb{E}[\epsilon_t]$ fraction of errors, we place an upper bound on the difference $t' - t$ such that $\mathbb{E}[\epsilon'_t] \leq 2\mathbb{E}[\epsilon_t]$.

Lemma 22. *Assume $\alpha^{-1} > 2(k')^2 r$. Suppose at time t there are $\epsilon_t = 2M\mathbb{E}[\epsilon_t]$ errors for $M \geq 2$. Then the expected number of time steps until $\epsilon_t = M\mathbb{E}[\epsilon_t]$ is at most $4\frac{n}{r}(1-\alpha k')^{-1}$*

Proof. Let $\mathcal{E} \geq \mathbb{E}[\epsilon_t]$ be the solution to the below equation.

$$-\Big[(1-\frac{1}{n^c})2r\mathcal{E}(1-\alpha k') - \alpha k' k - \frac{1}{n^c}k\Big] + 1/2 = 0$$

Recall that this is the expected number of changes in errors for any $\epsilon_t < 1/4$. First note that the bracketed portion of the above expectation is linear in $\mathcal{E}$. Thus doubling $\mathcal{E}$ doubles the expected the number we fix on any time step (and does not effect the expected number of errors nature adds). In general, if we have $M\mathcal{E}$ errors, then we expect the total number of errors to decrease by $M/2 - 1/2$ on the next time step. Let X_t be the expected change in total errors from time step $(t-1)$ to t. Thus if $\epsilon_t \geq M\mathbb{E}[\epsilon_t]$, then we have $\mathbb{E}[X_t] \leq -\frac{1}{2}(M-1)$. Now while the X_i's are not independent, regardless of the realization of $\{X_j\}_{j<i}$, we still have $\mathbb{E}[X_t] \leq -\frac{1}{2}(M-1)$ given that the barrier $\epsilon_t = M\mathbb{E}[\epsilon_t]$ has not yet been hit (and under the consistent $\epsilon_t \leq 1/4$ assumption of this entire section

given by Lemma 15). So in order to fix the $nM\mathbb{E}[\epsilon_t]$ required errors, we need T large enough so that $-\mathbb{E}[\sum_{i=1}^{T} X_i] = nM\mathbb{E}[\epsilon_t]$. Since $-\mathbb{E}[\sum_{i=1}^{T} X_i] \geq \frac{T}{2}(M-1)$, after setting $\frac{T}{2}(M-1) = nM\mathbb{E}[\epsilon_t]$ the it follows that at most

$$T \leq nM\mathbb{E}[\epsilon_t]\frac{2}{M-1} \leq 4n\mathbb{E}[\epsilon_t] \leq 4\frac{n}{r}(1-\alpha k')^{-1}$$

steps are required in expectation, where the last inequality holds following Theorem 18, which is the desired result. ■

Theorem 23. *Assume $\alpha^{-1} > 2(k')^2 r$. Suppose at time t there are $\epsilon_t = M\mathbb{E}[\epsilon_t]$ errors for $M \geq 1$. Then the expected number of time steps until $\epsilon_t = 2\mathbb{E}[\epsilon_t]$ is at most* $\log(M)4\frac{n}{r}(1-\alpha k')^{-1}$

Proof. Let τ be the total time taken, and let $\tau(a,b)$ be the number of time steps between the first time $\epsilon_t \leq a\mathbb{E}[\epsilon_t]$ until the first time $\epsilon_t \leq b\mathbb{E}[\epsilon_t]$. Then $\tau = \sum_{i=0}^{\log(M)-1} \tau(\frac{M}{2^i}, \frac{M}{2^{i+1}})$, and the result follows immediately from linearity of expectation and repeated application of Lemma 22. ■

3.7 Discussion

What we have shown is that with the addition of noise α, the problem becomes feasible on graphs with good expansion. The particular results of this paper have shown that given fixed noise α and fixed expansion $(1-\lambda_2)$, the range of probe values k for which our bounds hold with high probability is $7680r\frac{\log(n)}{(1-\lambda_2)} \leq k \leq \frac{\alpha^{-1}}{k'}$. This follows by taking $c = 3$ in Theorem 21. This gives a range of the possible number of probes that we could use on a given graph with fixed noise while still obtaining the theoretical bound prove in the prior section.

The intuition for why better theoretical bounds for larger values of α are not possible is due to the fact that, when we have on the order of $O(n/r)$ errors, we can only say that we expect to fix at most 1 error on any given successful path, whereas an unsuccessful path will yield k' new errors. Since a path is unsuccessful with probability $\alpha k'$, it must be the case that α^{-1} is at least $(k')^2$ in order to yield any sort of equilibrium. To obtain the desired $O(n/r)$ bound, we need to add an additional factor of r, namely requiring that $\alpha^{-1} \geq (k')^2 r = kk'$. As we will see in the experimental section, however, this condition is almost certainly not necessary in practice, and α^{-1} as small as $2k$ suffices to acceptable results on the order of $\frac{n}{k}$ mean errors.

4 Experiments

Now our bounds for the performance of the expander walk algorithm are understandably not as tight as those we derived for the noiseless case. Furthermore, it is difficult to get a precisely handle on the statistical properties of the stochastic process that is the number of errors. Thus, we simulated the expander walk algorithm on randomly generated d-regular expander graphs. In each case, the spectral gap of the graph was roughly $1/5$, coinciding with the 5-regularity and definition of spectral expansion, and therefore was constant with respect to increasing n. The techniques used to generate the random expanders were fairly straightforward, using results of random spectral graph theory which roughly guarantee than randomly generated Erdos-Renyi graphs will be good expanders.

Restricting to d-regularity was accomplished by selective dropout of edges, followed by a careful reintoduction. The results are detailed below.

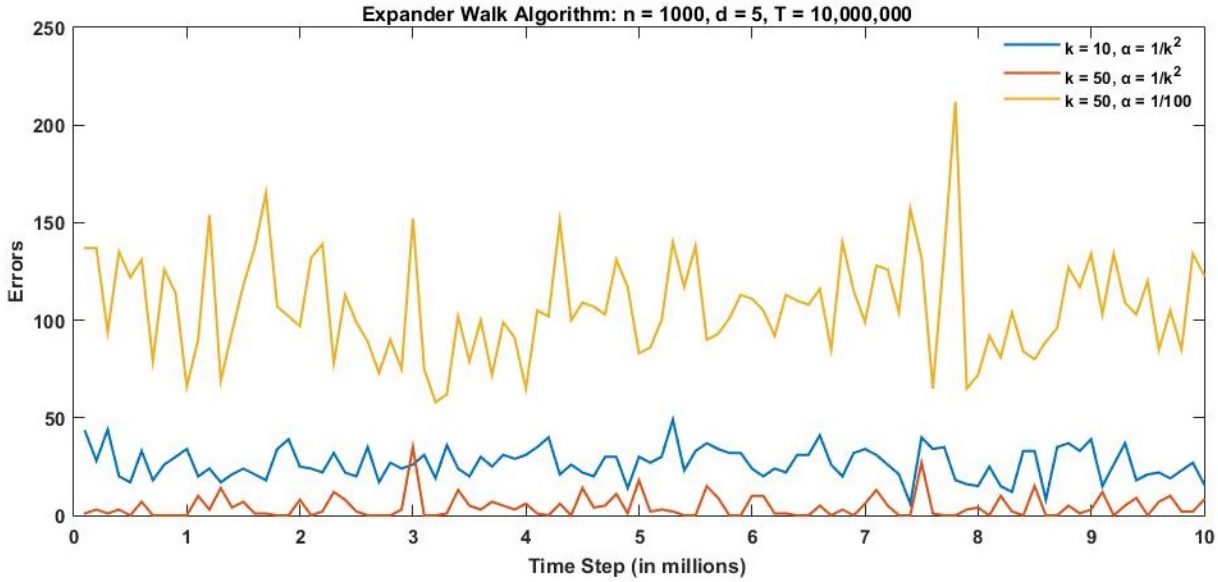

Figure 1: Expander Walk algorithm simulated for different noise and probe parameters

Parameters	Max Error	Min Error	Mean	Median	Variance (σ^2)
$k = 10,\ \alpha = 1/k^2$	72	4	26.6204	26	60.5892
$k = 50,\ \alpha = 1/k^2$	54	0	4.9799	3	36.2155
$k = 50,\ \alpha = 1/100$	212	22	104.6569	103	594.3254

The above figure shows the performance of our algorithm on varying probing-noise ratios. In each case we only ran one path per time step, as we found increasing the number of paths sampled on a time step did not substantially effect the error rate as opposed to simply increasing k. First note that the probe-error tradeoffs observed are closer to the $\frac{n}{k}$ expected number of errors rather than the upper bounds derived in this paper. Also note when keeping the noise constant $\alpha = \frac{1}{100}$, increasing the number of probes from $k = 10$ to $k = 50$ results in roughly a 5 fold decrease on the mean error rate. This further suggests the linear probe-error tradeoff derived in the prior analysis.

Below we provide the results of two further experiments for larger expanders graphs with $n = 10,000$ vertices. The first of which we vary the number of probes taken and consider noise that varies in $\frac{1}{k^2}$. In the last set of experiments we fix the number of probes at $k = 50$ and vary the noise. Note that even in the case that $\alpha = \frac{1}{2k}$, the performance of our algorithm still does not substantially degenerate.

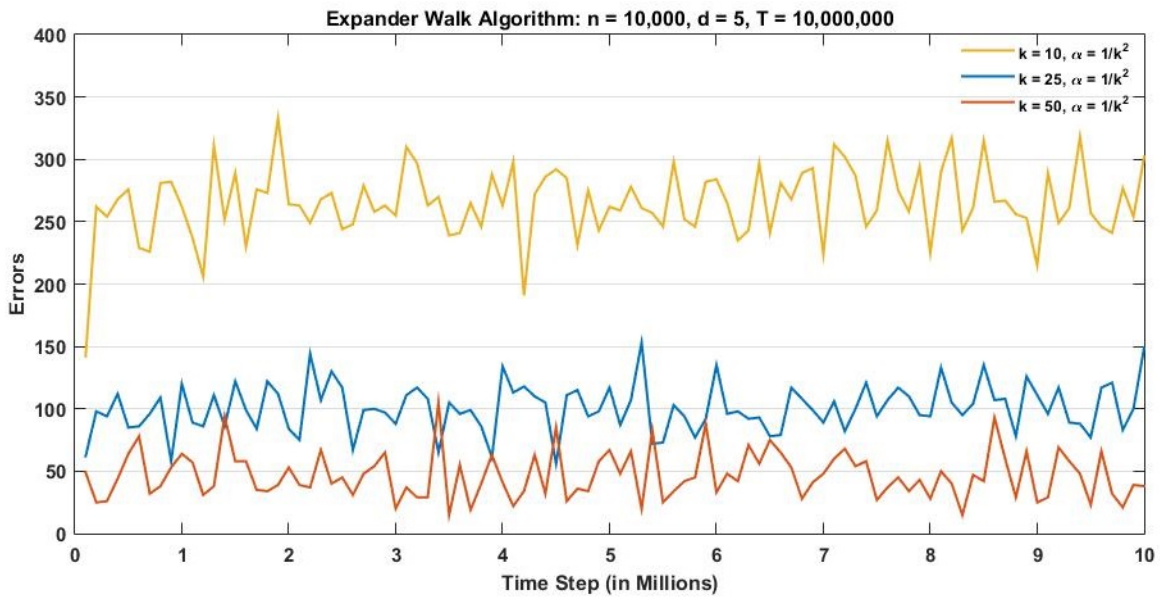

Figure 2: Simulating the Expander Walk algorithm on varying values of k, where the noise scales inverse quadratically with k.

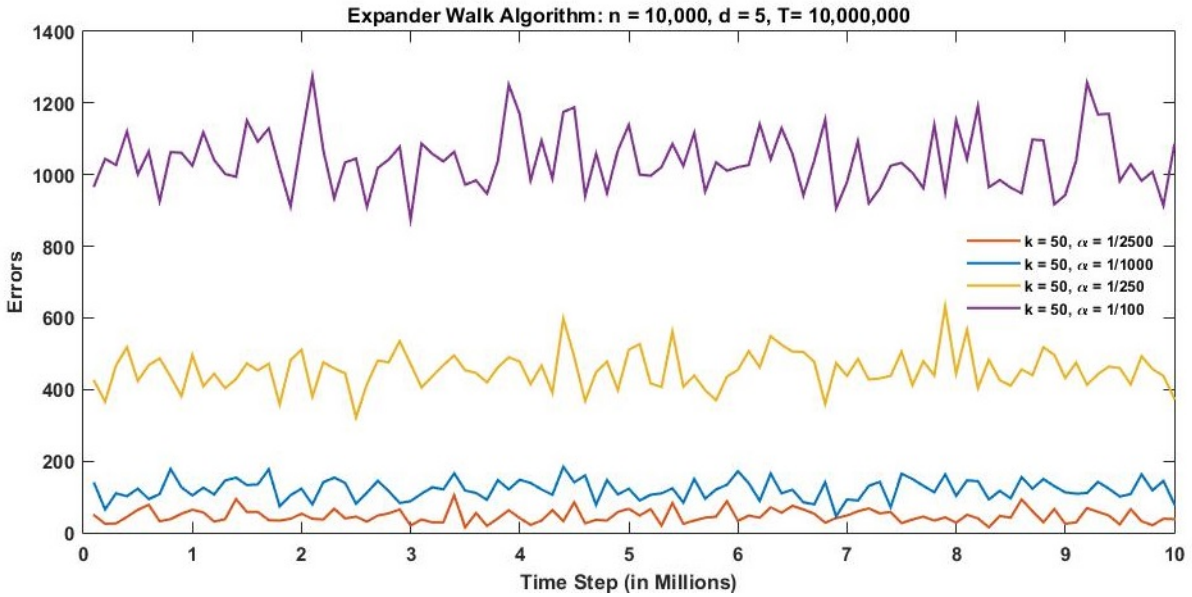

Figure 3: Simulating the Expander Walk algorithm with constant $k = 50$ probes per time step and varying amounts of noise.

References

[1] Aris Anagnostopoulos, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. Sorting and selection on dynamic data. *Theoretical Computer Science*, 412(24):2564–2576, 2011.

[2] Aris Anagnostopoulos, Jakub Łącki, Silvio Lattanzi, Stefano Leonardi, and Mohammad Mahdian. Community detection on evolving graphs. In *Advances in Neural Information Processing Systems*, pages 3522–3530, 2016.

[3] Amir Globerson, Tim Roughgarden, David Sontag, and Cafer Yildirim. How hard is inference for structured prediction? In *ICML*, pages 2181–2190, 2015.

[4] Alexander D Healy. Randomness-efficient sampling within nc 1. *Computational Complexity*, 17(1):3–37, 2008.

[5] Tom Leighton and Ronitt Rubinfeld. Lecture notes: Mathematics for computer science. `http://web.mit.edu/neboat/Public/6.042/randomwalks.pdf`, December 2006.

5 Appendix

Lemma 24 (Bias Gambler's Ruin)**.** *Let $X_0, X_1, \ldots$ be a bias random walk with probability p of increasing by 1 and probability $1-p$ of decreasing by 1. Let P_n be the probability that the random walk starting at $X_0 = n$ hits the value $T = n + m$ before it hits 0. Then*

$$P_n \leq \Big(\frac{p}{1-p}\Big)^m$$

Proof. The Lemma is well known, and a more detailed proof can be found in Leighton and Rubinfeld's lecture notes [5]. The sketch is as follows. By definition of P_n we obtain the recursion $P_n = pP_{n+1} + (1-p)P_{n-1}$. This is a linear homogenous recurrence with boundary conditions $P_T = 1$ and $P_0 = 0$. Solving the recurrence one obtains

$$P_n = \frac{\Big(\frac{1-p}{p}\Big)^n - 1}{\Big(\frac{1-p}{p}\Big)^T - 1} \leq \frac{\Big(\frac{1-p}{p}\Big)^n}{\Big(\frac{1-p}{p}\Big)^T} = \Big(\frac{p}{1-p}\Big)^m$$

as desired. ■

Theorem 25 (Expander Walk Sampling)**.** *Let G be a graph with normalized second largest eigenvalue λ_2. Let $f : V(G) \to \{0,1\}$ be any function, and let $\mu = \frac{1}{n}\sum_{v_i \in V(G)} f(v_i)$ be its mean. If $Y_0, Y_1, \ldots, Y_{k'}$ is a k'-step random walk starting at a random vertex Y_0, then we have for all $\gamma > 0$:*

$$\Pr\Big[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu > \gamma\Big] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}} = p$$

and

$$\Pr\Big[\frac{1}{k'}\sum_{i=0}^{k'} f(Y_i) - \mu < -\gamma\Big] \leq e^{-\frac{\gamma^2(1-\lambda_2)k'}{10}}$$

Proof. The theorem, up to improved constants, is well known. A detailed proof can be found in [4]. ■