When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

Kuan Cheng
Center on Frontiers of Computing Studies, School of Computer Science, Peking University, China

Minghui Ouyang
School of Mathematical Sciences, Peking University, China

Chong Shangguan
Research Center for Mathematics and Interdisciplinary Sciences, Shandong University, China
Frontiers Science Center for Nonlinear Expectations, Ministry of Education, Qingdao, China

Yuanting Shen
Research Center for Mathematics and Interdisciplinary Sciences, Shandong University, China

Abstract

Tanner codes are graph-based linear codes whose parity-check matrices can be characterized by a bipartite graph G together with a linear inner code C_0. Expander codes are Tanner codes whose defining bipartite graph G has good expansion property. This paper is motivated by the following natural and fundamental problem in decoding expander codes:

What are the sufficient and necessary conditions that δ and d_0 must satisfy, so that *every* bipartite expander G with vertex expansion ratio δ and *every* linear inner code C_0 with minimum distance d_0 together define an expander code that corrects $\Omega(n)$ errors in $O(n)$ time?

For C_0 being the parity-check code, the landmark work of Sipser and Spielman (IEEE-TIT'96) showed that $\delta > 3/4$ is sufficient; later Viderman (ACM-TOCT'13) improved this to $\delta > 2/3 - \Omega(1)$ and he also showed that $\delta > 1/2$ is necessary. For general linear code C_0, the previously best-known result of Dowling and Gao (IEEE-TIT'18) showed that $d_0 = \Omega(c\delta^{-2})$ is sufficient, where c is the left-degree of G.

In this paper, we give a near-optimal solution to the above question for general C_0 by showing that $\delta d_0 > 3$ is sufficient and $\delta d_0 > 1$ is necessary, thereby also significantly improving Dowling-Gao's result. We present two novel algorithms for decoding expander codes, where the first algorithm is deterministic, and the second one is randomized and has a larger decoding radius.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Expander graphs and randomness extractors; Theory of computation → Error-correcting codes; Mathematics of computing → Coding theory; Mathematics of computing → Combinatoric problems

Keywords and phrases expander codes, expander graphs, linear-time decoding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.61

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2312.16087` [10]

Funding The research of Chong Shangguan is supported by the National Key Research and Development Program of China under Grant No. 2021YFA1001000, the National Natural Science Foundation of China under Grant Nos. 12101364 and 12231014, and the Natural Science Foundation of Shandong Province under Grant No. ZR2021QA005.

Acknowledgements M. Ouyang would like to thank Extremal Combinatorics and Probability Group (ECOPRO), Institute for Basic Science (IBS, Daejeon, South Korea) for hosting his visit at the end of 2023.

1 Introduction

Graph-based codes are an important class of error-correcting codes that have received significant attention from both academia and industry. They have a long history in coding theory, dating back to Gallager's [19] celebrated *low-density parity-check codes* (LDPC codes for short). LDPC codes are a class of linear codes whose parity-check matrices can be characterized by low-degree (sparse) bipartite graphs, called *factor graphs*. Gallager analyzed the rate and distance of LDPC codes, showing that with high probability, randomly chosen factor graphs give rise to error-correcting codes attaining the Gilbert-Varshamov bound. He also presented an iterative algorithm to decode these codes from errors caused by a binary symmetric channel. Since the 1990s, LDPC codes have received increased attention due to their practical and theoretical performance (see [12, 14, 22, 35, 38, 42, 43]).

As a generalization of the LDPC codes, Tanner [49] introduced the so-called *Tanner codes*, as formally defined below. Let c, d, n be positive integers and $L := [n]$, where $[n] = \{1, \dots, n\}$. Given a (c, d)*-regular* bipartite graph G with bipartition $V(G) = L \cup R$ and a $[d, k_0, d_0]$-linear code C_0[1], the *Tanner code* $T(G, C_0) \subseteq \mathbb{F}_2^n$ is the collection of all binary vectors $x \in \mathbb{F}_2^n$ with the following property: for every vertex $u \in R$, $x_{N(u)}$ is a codeword of the inner code C_0, where $N(u) \subseteq L$ is the set of neighbors of u and $x_{N(u)} = (x_v : v \in N(u)) \in \mathbb{F}_2^d$ denotes the length-d subvector of x with coordinates restricted to $N(u)$; in other words, $T(G, C_0) := \{x \in \mathbb{F}_2^n : x_{N(u)} \in C_0 \text{ for every } u \in R\}$.

Expander codes are Tanner codes whose defining bipartite graphs have good expansion properties, namely, they are *bipartite expanders*. To be precise, for real numbers $\alpha, \delta \in (0, 1]$, a (c, d)-regular bipartite graph G with bipartition $V(G) = L \cup R$ with $L = [n]$ is called a (c, d, α, δ)*-bipartite expander* if for each subset $S \subseteq L$ with $|S| \le \alpha n$, S has at least $\delta c|S|$ neighbors in R, i.e., $|N(S)| := |\cup_{v \in S} N(v)| \ge \delta c|S|$. As each $S \subseteq L$ can have at most $c|S|$ neighbors in R, being a (c, d, α, δ)-bipartite expander means that every bounded size subset in L has as many neighbors in R as possible, up to a constant factor.

Sipser and Spielman [46] studied the Tanner code $T(G, C_0)$ with G being a bipartite expander and C_0 being a parity-check code. For simplicity, let $Par = \{(x_1, \dots, x_d) : \sum_{i=1}^{d} x_i = 0\}$ denote the parity-check code in $\mathbb{F}_2^d$. They remarkably showed that the expansion property of G can be used to analyze the minimum distance and the decoding complexity of $T(G, Par)$. Roughly speaking, they showed that for every bipartite expander G with sufficiently large *expansion ratio* $\delta > 1/2$, $T(G, Par)$ has minimum distance at least αn, which further implies that $T(G, Par)$ defines a class of *asymptotically good* codes. More surprisingly, they showed that if the expansion ratio is even larger, say $\delta > 3/4$, then for every such G, $T(G, Par)$ admits a linear-time decoding algorithm that corrects a linear number of errors in the adversarial noise model. Spielman [48] showed that expander codes can be used to construct asymptotically good codes that can be encoded and decoded both in linear time.

Besides the construction based on vertex expansion, [48] also provides a construction based on spectral expansion. This construction again inherits the general structure of Tanner code, i.e., it combines of an underlying bipartite graph and an inner code. The main difference is that the underlying graph is an edge-vertex incidence graph of a (non-bipartite) spectral expander. Spectral expander codes also have linear time encoding and decoding, and their (rate & distance) parameters are different from those of vertex expander codes. In this paper, we mainly focus on vertex expander codes. The reader is referred to references [5, 26, 34, 36, 37, 41, 47, 52, 53] for more details on spectral expanders.

[1] The reader is referred to Section 1.2 for basic definitions on graphs and codes.

Given the strong performance of expander codes, they have been of particular interest in both coding theory and theoretical computer science, and have been studied extensively throughout the years. For example, [23, 45] utilized expander codes to obtain near MDS codes with linear-time decoding. A line of research [2, 9, 16, 18, 44, 50, 51, 52] improved the distance analysis and decoding algorithm for expander codes in various settings. Very recently, a sequence of works applied expander codes on quantum LDPC and quantum Tanner code construction, finally achieving asymptotically good constructions and linear-time decoding [6, 7, 15, 17, 21, 24, 27, 29, 30, 31, 32, 33, 39, 40].

Given the discussion above, it is natural to suspect that the expansion ratio δ plays a prominent role in analyzing the properties of $T(G, Par)$. More precisely, one can formalize the following question. We always assume c, d, α, δ are constants while n tends to infinity.

▶ **Question 1.** *What is the minimum $\delta > 0$ such that every (c, d, α, δ)-bipartite expander G with $V(G) = L \cup R$ and $|L| = n$ defines an expander code $T(G, Par) \subseteq \mathbb{F}_2^n$ that corrects $\Omega_{c,d,\alpha,\delta}(n)$ errors in $O_{c,d,\alpha,\delta}(n)$ time?*

This question has already attracted considerable attention. Sipser and Spielman [46] used the bit-flipping algorithm (developed on the original algorithm of Gallager [19]) to show that $\delta > 3/4$ is sufficient to correct $(2\delta - 1)\alpha n$ errors in $O(n)$ time. Using linear programming decoding, Feldman, Malkin, Servedio, Stein and Wainwright [18] showed that $\delta > \frac{2}{3} + \frac{1}{3c}$ sufficient to correct $\frac{3\delta-2}{2\delta-1}\alpha \cdot n$ errors, while at the cost of a poly(n) decoding time. Viderman [50] introduced the “Find Erasures and Decode” algorithm to show that $\delta > \frac{2}{3} - \frac{1}{6c}$ is sufficient to correct $\Omega(n)$ errors in $O(n)$ time. Moreover, he also shows that there exists a $(c, d, \alpha, 1/2)$-bipartite expander G such that $T(G, Par)$ only has minimum distance two, and therefore cannot correct even one error. Viderman's impossibility result implies that $\delta > 1/2$ is necessary for the assertion of Question 1 holding for *every* (c, d, α, δ)-bipartite expander.

The above results only consider the case where the inner code C_0 is a parity-check code. Therefore, it is tempting to think about whether one can benefit from a stronger inner code C_0. Let us call a code *good* if it can correct $\Omega(n)$ errors in $O(n)$ time. Chilappagari, Nguyen, Vasic and Marcellin [11] showed that if G has expansion radio $\delta > 1/2$ and C_0 has minimum distance $d(C_0) \geq \max\{\frac{2}{2\delta-1} - 3, 2\}$, then every such Tanner code $T(G, C_0)$ is good. The above result implies that for $\epsilon \to 0$ and $\delta = 1/2 + \epsilon$, $d(C_0) = \Omega(\epsilon^{-1})$ is sufficient to make every Tanner code $T(G, C_0)$ good. Very recently, Dowling and Gao [16] significantly relaxed the requirement on δ by showing that for every $\delta > 0$,

$$d(C_0) \geq \Omega(c\delta^{-2}) \tag{1}$$

is sufficient[2] to make every Tanner code $T(G, C_0)$ good, and be able to correct αn errors. In particular, their result implies that, as long as the minimum distance of C_0 is large enough, any tiny positive expansion ratio is sufficient to construct a good Tanner code.

Putting everything together, it is interesting to understand how the expansion ratio δ of G and the minimum distance d_0 of C_0 affect the goodness of the Tanner code. We have the following generalized version of Question 1.

▶ **Question 2.** *What are the sufficient and necessary conditions that δ and d_0 must satisfy, so that every (c, d, α, δ)-bipartite expander G with $V(G) = L \cup R$, $|L| = n$, and every inner linear code $C_0 \subseteq \mathbb{F}_2^d$ with $d(C_0) \geq d_0$, together define an expander code $T(G, C_0) \subseteq \mathbb{F}_2^n$ that corrects $\Omega_{c,d,\alpha,\delta}(n)$ errors in $O_{c,d,\alpha,\delta}(n)$ time?*

[2] More precisely, $d(C_0) \geq 2t + c(t-1)^2 - 1$ with $t > \frac{1}{\delta}$.

The main purpose of this paper is to provide a near-optimal solution to the above question, as presented in the next subsection. On the negative side, we show that when $d_0\delta \leq 1$, there exists an extension of Viderman's construction, yielding expander codes of constant distance (see Proposition 4 below). Therefore, $d_0\delta > 1$ is a necessary condition for a general expander code $T(G, C_0)$ to be considered good (compared to the $\delta > \frac{1}{2}$ condition in the case of $T(G, Par)$). On the positive side, we show that $d_0\delta > 3$ is sufficient to make *every* expander code good (see Theorem 3 below for details). Our result only loses a multiplicity by three compared to the above necessary result.

1.1 Main results

Deterministic decoding of expander codes

Our main result, which significantly improves on (1), is presented as follows.

▶ **Theorem 3.** *Let G be a (c, d, α, δ)-bipartite expander and C_0 be a $[d, k_0, d_0]$-linear code, where $c, d, \alpha, \delta, d_0, k_0$ are positive constants. If $\delta d_0 > 3$, then there exists a linear-time decoding algorithm for the Tanner code $T(G, C_0)$ that can correct γn errors, where $\gamma = \frac{2\alpha}{d_0(1+0.5c\delta)}$.*

Theorem 3 shows that $\delta d_0 > 3$ is sufficient to make every Tanner code $T(G, C_0)$ good. On the other hand, the next proposition shows that for every $d_0 \geq 2$, $\delta d_0 > 1$ is necessary.

▶ **Proposition 4.** *For every $d, d_0 \geq 2$ and $n \geq 10d_0$, there exist constants $0 < \alpha < 1, c \geq 3$ and a $(c, d, 0.9\alpha, \frac{1}{d_0})$-bipartite expander G with $V(G) = L \cup R$ and $|L| = n$ such that for every $[d, k_0, d_0]$-linear code C_0, $T(G, C_0)$ has minimum Hamming distance at most d_0.*

Theorem 3 and Proposition 4 together show that our requirement $\delta d_0 = \Omega(1)$ is in fact almost optimal for Question 2. Moreover, we have the following conjecture on the fundamental trade-off between δ and d_0.

▶ **Conjecture 5.** *If $\delta d_0 > 1$, then for every (c, d, α, δ)-bipartite expander G and every inner code $C_0 \subseteq \mathbb{F}_2^d$ with $d(C_0) \geq d_0$, the expander code $T(G, C_0) \subseteq \mathbb{F}_2^n$ can correct $\Omega_{c,d,\alpha,\delta}(n)$ errors in $O_{c,d,\alpha,\delta}(n)$ time.*

Due to space limit, we will omit the proof of the linear-running time of our algorithm, as well as the proof of Proposition 4. They can be found in the full version of this paper [10].

Randomized decoding of expander codes

Another important direction in the study of expander codes is to understand the maximum number of errors that can be corrected in a linear-time decoding algorithm. Chen, Cheng, Li, and Ouyang [9] obtained a quite satisfactory answer to this problem for $T(G, Par)$. They showed that for every $\delta > 1/2$ and (c, d, α, δ)-bipartite expander G, $T(G, Par)$ has minimum distance at least $\frac{\alpha}{2(1-\delta)} \cdot n - O(1)$, and this is tight up to a $1 - o(1)$ factor. Moreover, for $\delta > \frac{3}{4}$, they also gave a linear-time decoding algorithm which corrects $\frac{3\alpha}{16(1-\delta)} \cdot n$ errors. A similar problem for general expander codes $T(G, C_0)$ was studied by [16].

Our decoding algorithm for Theorem 3 is deterministic and corrects γn errors in linear time. Theorem 6 shows that one can correct more errors by using a randomized algorithm.

▶ **Theorem 6.** *Let G be a (c, d, α, δ)-bipartite expander and C_0 be a $[d, k_0, d_0]$-linear code, where $c, d, \alpha, \delta, d_0, k_0$ are positive constants. If $\delta d_0 > 3$, then there exists a linear-time randomized decoding algorithm for Tanner code $T(G, C_0)$ such that if the input has at most αn errors from a codeword, then with probability $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$, the decoding algorithm can output the correct codeword.*

1.2 Notations and definitions

A graph is a pair $G = (V, E)$, where V is a set whose elements are called vertices and E is a set of 2-subsets of V, whose elements are called edges. For a vertex $u \in V$, the set of *neighbors* of u in G is denoted by $N(u) := \{v \in V : \{u, v\} \in E\}$. For a subset $S \subseteq V(G)$, let $N(S) = \cup_{u\in S} N(u)$ be the set of all the neighbors of the vertices in S. A graph G is *bipartite* if $V(G)$ admits a bipartition $V(G) = L \cup R$ such that both L and R contain no edge. Furthermore, G is (c, d)-regular if every vertex $v \in L$ has exactly c neighbors in R and every vertex $u \in R$ has exactly d neighbors in L.

Let $\mathbb{F}_2 = \{0, 1\}$ denote the finite field of size 2. A code C is simply a subset of $\mathbb{F}_2^n$. For two vectors $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n) \in \mathbb{F}_2^n$, the *Hamming distance* between x and y, denoted by $d_H(x, y)$, is the number of coordinates where x and y differ, that is, $d_H(x, y) = |\{i \in [n] : x_i \neq y_i\}|$. The *minimum distance* of a code $C \subseteq \mathbb{F}_2^n$, denoted by $d(C)$, is the minimum of $d_H(x, y)$ among all distinct $x, y \in C$. Let $\mathrm{wt}(x)$ denote the number of nonzero coordinates of x. A code $C \subseteq \mathbb{F}_2^n$ is said to be an $[n, k, d(C)]$-*linear code* if it is a linear subspace in $\mathbb{F}_2^n$ with dimension k and minimum distance $d(C)$. It is well-known that for every linear code C, $d(C) = \min\{\mathrm{wt}(x) : x \in C \setminus \{0\}\}$.

Throughout, let G be a (c, d, α, δ)-bipartite expander, and C_0 be a $[d, k_0, d_0]$ linear code. Let $T(G, C_0)$ be the Tanner code defined by G and C_0. Let Check be the error-detection algorithm of C_0, which checks whether a vector in $\mathbb{F}_2^d$ is a codeword of C_0. Assume that Check takes h_0 time. Similarly, let Decode be the correct-correction algorithm for C_0, which corrects up to $\lfloor \frac{d_0-1}{2} \rfloor$ errors. Assume that Decode takes t_0 time. Note that h_0, t_0 are constants depending only on C_0 but not on n.

Conventionally speaking, let us call the vertices in L *variables* and the vertices in R *constraints*. Given a vector $x \in \mathbb{F}_2^n$, which is corrupted from some codeword $y \in T(G, C_0)$, let us call a constraint $u \in R$ *satisfied* if $x_{N(u)} \in C_0$, otherwise call it *unsatisfied*.

1.3 Some related works

Below, we briefly review two previous works [16, 46] that are closely related to our decoding algorithms for Theorems 3 and 6. Let us start from the decoding algorithm of Sipser and Spielman [46]. We summarize as follows the so-called iterated decoding or message-passing algorithm of [46] that decodes $T(G, Par)$.

- Let $y \in T(G, Par)$ be the correct codeword that we want to decode from the received vector x. In the first round, the algorithm runs Check$(x_{N(u)})$ for every $u \in R$. If a constraint u is unsatisfied, then it sends a "flip" message to every variable in $N(u) \subseteq L$. Sipser and Spielman showed that as long as the expansion ratio of G is sufficiently large ($\delta > 3/4$) and the number of corruptions in x is sufficiently small but not identically zero (that is, $1 \leq d_H(x, y) \leq (2\delta - 1)\alpha \cdot n$), then there must exist a variable $v \in L$ that receives $> c/2$ flip messages, which implies that more than half constraints in $N(v)$ are unsatisfied. The algorithm then flips x_v and updates x and the status of the constraints in $N(v)$. Note that since Par is the parity-check code, flipping x_v makes all satisfied constraints in $N(v)$ unsatisfied and all unsatisfied constraints in $N(v)$ satisfied. Therefore, by flipping x_v one can strictly reduce the number of unsatisfied constraints.
- The algorithm then runs the above process repeatedly. As long as there are still unsatisfied constraints, it finds the desired $v \in L$ so that flipping x_v strictly reduces the number of unsatisfied constraints. As there are at most $|R| = cn/d$ unsatisfied constraints, the above process must stop in $O(n)$ rounds and therefore yields an $O(n)$ time decoding algorithm.

Dowling and Gao [16] extend Sipser and Spielman's algorithm from $T(G, Par)$ to the more general setting $T(G, C_0)$ by making use of the minimum distance of C_0. Their algorithm works for linear codes defined on any finite field, but we will describe it only for $\mathbb{F}_2$.

- The algorithm begins by setting a threshold $t \leq \lfloor \frac{d_0-1}{2} \rfloor$ and then runs Decode$(x_{N(u)})$ for every $u \in R$. If a constraint $u \in R$ satisfies $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$, then it sends a "flip" message to every variable $v \in N(u)$ with Decode$(x_{N(u)})_v \neq x_v$. Note that Decode$(x_{N(u)}) \in \mathbb{F}_2^d$ is a codeword in C_0. The algorithm then flips all x_v for those v receiving at least one flip, and then updates x. [16] showed that as long as the minimum distance d_0 of C_0 is sufficiently large (see (1)), then flipping all variables that receive at least one flip can reduce the number of corrupted variables in x by some positive fraction.
- In the next steps, the algorithm runs the above process repeatedly. As the number of corrupted variables is at most $O(n)$, the algorithm will stop in $O(\log n)$ rounds. Crucially, in order to show that the running time of the algorithm is still linear-order but not of order $n \log n$, the authors proved that the running time of every single round is within a constant factor of the number of corrupted variables at the beginning of this round. As the numbers of corrupted variables form a decreasing geometric sequence with the leading term at most n, the total running time, which is within a constant factor of the sum of this geometric sequence, is also $O(n)$.

Lastly, it is worth mentioning that there are several very recent works on the explicit constructions of bipartite graphs with good vertex expansion properties, including the lossless expanders [8, 13, 20] and the unique-neighbor expanders [1, 3, 4, 25, 28].

1.4 Key new ideas in our work

In this subsection, we briefly introduce the key new ideas in our work. Let us focus on the deterministic decoding algorithm that proves Theorem 3. Let us begin by analyzing the following two possible places where the previous algorithm in [16] could be improved.

In every decoding round of the above algorithm, the constraints in R which satisfy $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$ (and hence send at least one and at most $t-1$ flips to L) in fact have two statuses, as detailed below. Let A be the set of constraints $u \in R$ that sends at least one flip and Decode$(x_{N(u)})$ computes the correct codeword in C_0 (i.e., Decode$(x_{N(u)}) = y_{N(u)}$); let B be the set of constraints $u \in R$ that sends at least one flip and Decode$(x_{N(u)})$ computes an incorrect codeword in C_0 (i.e., Decode$(x_{N(u)}) \neq y_{N(u)}$).

Two possible places where the previous algorithm could be improved

(i) It could be the case that every constraint $u \in A$ satisfies $d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) = 1$ and hence sends only one correct flip to L; in the meanwhile, every constraint $u \in B$ may satisfy $d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) = t-1$ and sends as many as $t-1$ flips to L, which could be all wrong. In this case, the constraints in R altogether send $|A|$ correct flips and $(t-1)|B|$ wrong flips to the variables in L.

(ii) Unfortunately, the situation could be worse. Since our bipartite graph G is (c,d)-regular, it could be the case that the neighbors of the constraints in A are highly concentrated (e.g., all $|A|$ correct flips are received by as few as $|A|/c$ variables in L), and the neighbors of the constraints in B are highly dispersed (e.g., all $(t-1)|B|$ possibly wrong flips are received by as many as $(t-1)|B|$ variables in L). Consequently, a small number of corrupted variables but a large number of correct variables in L receive flip messages.

Given the two issues above, if we flip all variables that receive at least one flip, then in the worst case, we could correct $|A|/c$ old corrupt variables but produce $(t-1)|B|$ new corrupt variables. Recall that to make the algorithm in [16] work, in each round, we need to reduce the number of corrupted variables by at least a positive fraction, which implies that in the worst case it is necessary to have $|A|/c \geq (t-1)|B|$. Together with some lower bound on $|A|$ and upper bound on $|B|$ (see [16] for details), one can prove that in such worst scenario (1) is necessary for Dowling and Gao's algorithm to work.

Our new algorithm begins by noting that we could indeed fix the two problems mentioned above. To do so, we introduce several new ideas as briefly presented below.

Key new ideas in our work

Let $F := \{i \in [n] : x_i \neq y_i\}$ be the set of corrupt variables in x. Similarly to [16], our new algorithm begins by setting a threshold $t = \lfloor \frac{1}{\delta} \rfloor$ and then runs $\text{Decode}(x_{N(u)})$ for every $u \in R$.

(a) To fix the first problem, if a constraint $u \in R$ satisfies $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$, then instead of sending a flip message to every $v \in N(u)$ with $\text{Decode}(x_{N(u)})_v \neq x_v$, the new algorithm just arbitrarily picks *exactly one* such variable v, and sends a flip message to *only* this specific v. So, every constraint in $A \cup B$ sends exactly one flip to L.

(b) To fix the second problem, we associate each $v \in L$ with a counter $\tau_v \in \{0, 1, \dots, c\}$ that counts the number of flips received by v. For each $m \in [c]$, let S_m denote the set of variables that receive exactly m flips. Then, instead of flipping every variable that receives at least one flip, i.e., instead of flipping $\cup_{m=1}^{c} S_m$, we only flip S_m for some $m \in [c]$. Crucially, we show that if the number $|F|$ of corrupt variables is not too large, then there must *exist* some $m \in [c]$ such that $|S_m|$ has the same order as $|F|$, and more importantly, a $(1/2+\kappa)$-fraction of variables in $|S_m|$ are corrupted (and therefore can be corrected by the flipping operation), where κ is an absolute positive constant. Therefore, it follows that by flipping all variables in S_m, one can reduce $|F|$ by some positive fraction.

Note that the details of (a) and (b) can be found in Section 3.2, where we call the algorithm corresponding to (a) and (b) "EasyFlip" and write $\text{EasyFlip}(x, m)$ as the output of the EasyFlip if x is the input vector and S_m is flipped (see Algorithm 2).

However, there is still a gap that needs to be fixed, that is, how to find the required S_m? A plausible solution is to run $\text{EasyFlip}(x, m)$ for every $m \in [c]$. This would roughly increase the total running time by a c factor, which will still be $O(n)$, provided that the original running time is $O(n)$. Unfortunately, by doing so we still cannot *precisely* identify the required S_m, as in general we do not know how to count the number of corrupted variables in some corrupted vector. We will fix this issue by introducing our third key new idea:

(c) Note that what we can explicitly count in each round of the algorithm is the number of unsatisfied constraints. Roughly speaking, our strategy is to run EasyFlip iteratively for a large but still constant number of times and then pick the final output that significantly reduces the number of unsatisfied constraints.

More precisely, assume that we will run EasyFlip iteratively for s rounds. Let $x^0 := x$ and write $x^1 := \text{EasyFlip}(x^0, m_1)$ as the output of the 1st EasyFlip invocation where the variables in S_{m_1} are flipped for some $m_1 \in [c]$; more generally, for $k \in [s]$, write $x^k := \text{EasyFlip}(x^{k-1}, m_k)$ as the output of the kth EasyFlip invocation where the variables in S_{m_k} is flipped for some $m_k \in [c]$. Note that in Algorithm 3 we call the above iterated invocations of EasyFlip as "DeepFlip", and write $x^k := \text{DeepFlip}(x, (m_1, \dots, m_k))$ as the output of the kth EasyFlip invocation. For $0 \leq k \leq s$, let $F^k \subseteq L$ and $U^k \subseteq R$ denote the sets of corrupted variables and unsatisfied constraints caused by x^k, respectively. We prove that there are constants $0 < \epsilon \ll \epsilon' \ll \epsilon'' < 1$ such that the following two wordy but useful observations hold:

- **(c1)** If the number of corrupted variables is reduced *dramatically* then the number of unsatisfied constraints is reduced *significantly*, i.e., if for some $k \in [s]$, $|F^k| \le \epsilon |F^0|$, then $|U^k| \le \epsilon' |U|$;
- **(c2)** If the number of unsatisfied constraints is reduced *significantly*, then the number of corrupted variables must be reduced by a least a constant fraction i.e., if for some $k \in [s]$, $|U^k| \le \epsilon' |U^0|$, then $|F^k| \le \epsilon'' |F|$.

In the following, we will briefly argue how we will make use of the two observations (c1) and (c2). Recall that in (b) we have essentially guaranteed that for every $k \in [s]$, there exists some $m_k^* \in [c]$ such that by flipping $S_{m_k^*}$ in EasyFlip, one could reduce the number of corrupted variables by an η-fraction for some $\eta \in (0,1)$. It follows that if we run DeepFlip iteratively for $(m_1, \ldots, m_s) = (m_1^*, \ldots, m_s^*)$, then we have $|F^s| \le (1-\eta)^s |F| < \epsilon |F|$, provided that $s > \log_{(1-\eta)^{-1}} \epsilon^{-1}$ is sufficiently large (but still a constant independent of n). Therefore, if we run DeepFlip thoroughly for all $(m_1, \ldots, m_s) \in [c]^s$, then by (c1) there must exist at least one[3] $x^k := \text{DeepFlip}(x, (m_1, \ldots, m_k))$ with $k \le s$ such that $|U^k| \le \epsilon' |U|$. Moreover, using the last inequality, such x^k and $(m_1, \ldots, m_k)$ can be explicitly identified. Now, by (c2) we can conclude that the number of corrupted variables is indeed reduced by at least a constant fraction.

Note that the above brute-force search only increases the total running time by at most a c^s factor. The details of (c) and the analysis of DeepFlip can be found in Section 3.3 and Algorithm 3. Moreover, we call the algorithm that runs $\text{DeepFlip}(x, (m_1, \ldots, m_s))$ thoroughly for all $(m_1, \ldots, m_s) \in [c]^s$ as "HardSearch", and is discussed in Section 3.4 and Algorithm 4. The discussion above basically shows that every HardSearch invocation could reduce the number of corrupted variables by a constant fraction.

Running HardSearch iteratively for $O(\log n)$ rounds, the total number of corrupted variables will be smaller than $\lfloor \frac{d_0 - 1}{2} \rfloor$, which can be easily corrected by running Decode for every $u \in R$. The main algorithm that puts everything together is called "MainDecode", and is presented in Section 3.1 and Algorithm 1.

To show that the total running time is still linear in n, we adopt an argument similar to that in the previous works (e.g., [16]). We show that the running time of every HardSearch invocation is within a constant factor of the number of corrupted variables at the beginning of this invocation.

Lastly, we would like to mention that our randomized decoding algorithm (see Algorithm 5), which proves Theorem 6 and has a larger decoding radius than the deterministic algorithm, basically follows from the same framework mentioned above. Loosely speaking, the high-level idea of the randomized algorithm is to reduce the number of corruptions to a moderate size that can be handled by the deterministic algorithm. For that purpose, we design a random flip strategy which can be summarized as follows.

Recall that for every $m \in [c]$, S_m denotes the set of all variables that receive exactly m flips. First, for every constraint $u \in R$ satisfying $1 \le d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \le t$, we arbitrarily pick exactly one variable $v \in N(u)$ with $\text{Decode}(x_{N(u)})_v \ne x_v$ and send a flip message to this specific v. Then, we collect all suspect variables that receive at least one flip. Subsequently, we design a random sampling procedure to select a subset of $\cup_{m \in [c]} S_m$ to flip. We show that this procedure can ensure, with high probability, that this subset has more corrupted variables than correct variables, as long as the total number of corruptions is at most αn. By applying this strategy iteratively, we can show that in each iteration, the number of corruptions will be reduced by a positive fraction. Then, after running a

[3] Clearly, $\text{DeepFlip}(x, (m_1^*, \ldots, m_s^*))$ gives a candidate for such x^k.

constant number of iterations, the number of corrupted bits can be reduced to a range that the deterministic algorithm can handle. At this point, the deterministic algorithm is invoked to correct all remaining corrupted variables. Moreover, as n increases, the fail probability of each iteration tends to 0. So the overall random algorithm will succeed with high probability.

2 An auxiliary lemma

Given two subsets $S, T \subseteq V(G)$, let $E(S,T)$ denote the set of edges with one endpoint in S and another endpoint in T. For every positive integer t, let

- $N_{\leq t}(S) = \{u \in V(G) : 1 \leq |N(u) \cap S| \leq t\}$,
- $N_t(S) = \{u \in V(G) : |N(u) \cap S| = t\}$,
- and $N_{\geq t}(S) = \{u \in V(G) : |N(u) \cap S| \geq t\}$.

We will make use of the following crucial property of bipartite expander graphs.

▶ **Proposition 7** (Folklore). *Let G be a (c,d,α,δ)-bipartite expander. Then, for every set $S \subseteq L$ with $|S| \leq \alpha n$ and every integer $t \in [d]$, we have that$|N_{\leq t}(S)| \geq \frac{\delta(t+1)-1}{t} \cdot c|S|$.*

3 Deterministic decoding: Decoding $\Omega(n)$ corruptions in $O(n)$ time

We need to set some parameters. Suppose that $d_0 > \frac{3}{\delta} - 1$. Let $t = \lfloor \frac{1}{\delta} \rfloor$. Take $\epsilon_0 > 0$ such that $d_0 > \frac{3}{\delta} - 1 + 2\epsilon_0$ and $\lfloor \frac{1}{\delta} + \epsilon_0 \rfloor = \lfloor \frac{1}{\delta} \rfloor$. For every $0 < \epsilon_1 < \frac{\epsilon_0 \delta^2}{100}$, let $\epsilon_2 = \frac{\epsilon_1}{c+1} \cdot \frac{\delta(t+1)-1}{t} > 0$ and $\epsilon_3 = \epsilon_2 \left(2(1-\epsilon_1) \left(\frac{1}{2} + \frac{\epsilon_0 \delta^2}{2} \right) - 1 \right) > 0$. It is not hard to check that ϵ_1, ϵ_2 and ϵ_3 are all well-defined. Lastly, let $\epsilon_4 = \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1 - \epsilon_3)$, $\ell = \left\lceil \log_{1-\epsilon_3} \left(\lfloor \frac{d_0-1}{2} \rfloor \frac{1}{\gamma n} \right) \right\rceil$ and $s_0 = \left\lceil \log_{1-\epsilon_3} \left(\epsilon_4 \frac{\delta d_0 - 1}{d_0 - 1} \right) \right\rceil$.

3.1 The main decoding algorithm – MainDecode

Given a corrupt vector $x \in \mathbb{F}_2^n$ with at most γn corruptions, our main decoding algorithm (see Algorithm 1 below) works as follows. The algorithm is divided into two parts. In the first part (see steps 2-10 below), it invokes HardSearch (see Algorithm 4 below) recursively for ℓ rounds, where in every round the number of corrupt variables is reduced by a $(1-\epsilon_3)$-fraction. After ℓ executions of HardSearch, the number of corrupt variables is reduced to at most $\lfloor \frac{d_0-1}{2} \rfloor$. Then, in the second part of the algorithm (see steps 11-13 below), the decoder of the inner code C_0 is applied to finish decoding.

The next two lemmas justify the correctness and the linear running time of MainDecode.

▶ **Lemma 8.**

(i) *Let x be the input vector of* HardSearch *and let F be the set of corrupt variables of x. Let $x' :=$* HardSearch(x) *and F' be the set of corrupt variables of x'. If $|F| \leq \gamma n$, then $|F'| \leq (1-\epsilon_3) \cdot |F|$.*

(ii) *In step 11 of* MainDecode*, the number of corrupt variables in x^ℓ is at most $\lfloor \frac{d_0-1}{2} \rfloor$.*

▶ **Lemma 9.**

(i) *Let x be the input vector of* HardSearch *and let F be the set of corrupt variables of x. If $|F| \leq \gamma n$, then the running time of* HardSearch *is at most $O(n + |F|)$.*

(ii) *Furthermore, if the number of corrupt variables in the input vector of* MainDecode *is at most γn, then the running time of* MainDecode *is $O(n)$.*

Algorithm 1 Main decoding algorithm for expander codes – MainDecode.

```
   Input: $G$, $C_0$, $x \in \mathbb{F}_2^n$
   Output: $x' \in \mathbb{F}_2^n$
1  Set $i = 1$ and $x^0 = x$;
2  for $1 \leq i \leq \ell$ do
3      $U^{i-1} \leftarrow \{u \in R : x^{i-1}_{N(u)} \notin C_0\}$;
4      if $|U^{i-1}| = 0$ then
5          return $x' \leftarrow x^{i-1}$;
6      else
7          $x^i \leftarrow$ HardSearch($x^{i-1}$);
8          $i \leftarrow i + 1$ ;
9      end
10 end
11 for every $u \in R$ such that $x^{\ell}_{N(u)} \notin C_0$ do
12     $x^{\ell}_{N(u)} \leftarrow$ Decode($x^{\ell}_{N(u)}$)
13 end
14 return $x' \leftarrow x^{\ell}$;
```

Proof of Theorem 3. Let $y \in T(G, C_0)$ be a codeword and $x \in \mathbb{F}_2^n$ be a corrupted vector. Let $F = \{i \in [n] : x_i \neq y_i\}$ be the set of corrupt variables of x with respect to y. To prove the theorem, it suffices to show that as long as $|F| \leq \gamma n$, MainDecode finds y correctly in linear time. We will analyze the following two cases:

- If the algorithm returns x^i for some $0 \leq i \leq \ell - 1$, then as $|U^i| = 0$, we must have $x^i \in T(G, C_0)$. Let F^i be the set of the corrupt variables of x^i. Then it follows by Lemma 8 (i) that $d(x^i, y) = |F^i| \leq (1 - \epsilon_3)^i |F| \leq (1 - \epsilon_3)^i \gamma n < d(T(G, C_0))$, which implies that $x^i = y$.
- If the algorithm does not return x^i for any $0 \leq i \leq \ell - 1$, then it follows by Lemma 8 (ii) that $d(x^{\ell}, y) \leq \lfloor \frac{d_0 - 1}{2} \rfloor$. Therefore, one can find y by running Decode for every $u \in R$.

Moreover, by Lemma 9 the running time of MainDecode is $O(n)$. ◀

The remaining part of this section is organized as follows. In Section 3.2 below, we will introduce the basic building block of deterministic decoding – EasyFlip, which also corresponds to items (a) and (b) in Section 1.4. In Section 3.3 we will introduce the algorithm DeepFlip, which runs EasyFlip iteratively for a constant number of times. DeepFlip corresponds to item (c) in Section 1.4. In Section 3.4 we will introduce HardSearch, which is designed by running DeepFlip thoroughly for all choices of $(m_1, \ldots, m_s)$ until the number of unsatisfied constraints is significantly reduced. The proof of Lemma 8 is also presented in Section 3.4.

3.2 The basic building block of deterministic decoding – EasyFlip

In this subsection, we will present the algorithm EasyFlip (see Algorithm 2 below), which is the basic building block of our deterministic decoding. It contains the following two parts:

- EasyFlip (i): in the first part (see steps 1-6 below), it invokes Decode for each constraint $u \in R$ and sends flips to some variables $v \in L$;
- EasyFlip (ii): in the second part (see steps 7-11 below), it counts the number of flips received by each variable in L and flips all variables that receive exactly m flips.

Algorithm 2 Flip all variables receiving exactly m flips – EasyFlip.

```
   Input: $G$, $C_0$, $x \in \mathbb{F}_2^n$, and $m \in [c]$
   Output: $x' \in \mathbb{F}_2^n$
1  for every $u \in R$ do
2      $\omega \leftarrow \text{Decode}(x_{N(u)})$;
3      if $1 \le d_H(\omega, x_{N(u)}) \le t$ then
4          send a "flip" to an arbitrary vertex $v \in N(u)$ with $\omega_v \neq x_v$
5      end
6  end
7  for every $v \in L$ do
8      if $v$ receives exactly $m$ flips then
9          flip $x_v$
10     end
11 end
12 return $x' \leftarrow x$
```

Our goal is to show that there must exist an integer $m \in [c]$ such that by flipping all variables $v \in L$ that receive exactly m flips, one can reduce the number of corrupt variables in x' by a $(1-\epsilon_3)$-fraction, as compared with x. Note that for this moment, it suffices to prove the existence of such an m and we do not need to find it explicitly. In fact, later we will find the required m by exhaustive search.

We make the discussion above precise by the following lemma.

▶ **Lemma 10.** *Let x be the input vector of* EasyFlip *and let F be the set of corrupt variables of x. If $|F| \le \alpha n$, then there exists an integer $m \in [c]$ such that the following holds. Let $x' = \text{EasyFlip}(x, m)$ be the output vector of* EasyFlip *and F' be the set of corrupt variables of x'. Then $|F'| \le (1-\epsilon_3)|F|$.*

3.2.1 Proof of Lemma 10

Let us first introduce some notation and easy inequalities. Let $y \in T(G, C_0)$ be the correct codeword that we want to decode from x. Let A be the set of constraints $u \in R$ that sends a flip and $\text{Decode}(x_{N(u)})$ computes the correct codeword in C_0 (that is, $\text{Decode}(x_{N(u)}) = y_{N(u)}$). Similarly, let B be the set of constraints $u \in R$ that sends a flip and $\text{Decode}(x_{N(u)})$ computes an incorrect codeword in C_0 (i.e., $\text{Decode}(x_{N(u)}) \neq y_{N(u)}$).

By the definitions of A and $N_{\le t}(F)$, it is easy to see that

$$A = \{u \in R : 1 \le |N(u) \cap F| \le t\} = N_{\le t}(F). \tag{2}$$

Therefore, it follows by (2) and Proposition 7 that

$$|A| \ge \frac{\delta(t+1)-1}{t} \cdot c|F|. \tag{3}$$

Moreover, since a constraint $u \in R$ computes an incorrect codeword in C_0 only if it sees at least $d_0 - t$ corrupt variables in its neighbors (recall that $d(C_0) \ge d_0$), we have that

$$B = \{u \in R : |N(u) \cap F| \ge d_0 - t \text{ and } \exists\, \omega \in C_0 \text{ s.t. } 1 \le d_H(\omega, x_{N(u)}) \le t\} \subseteq N_{\ge d_0 - t}(F). \tag{4}$$

By counting the number of edges between F and $N(F)$, we see that

$$(d_0 - t)|B| \le |E(F, B)| \le |E(F, N(F))| = c|F|,$$

which implies that

$$|B| \leq \frac{c|F|}{d_0 - t}. \tag{5}$$

Consider the following two equalities,

$$\sum_{k=1}^{d} k \cdot |N_k(F)| = c|F| \quad \text{and} \quad \sum_{k=1}^{d} |N_k(F)| \;=\; |N(F)| \;\geq\; \delta c|F|.$$

By multiplying the second by $\frac{1}{\delta} + \epsilon_0$ and subtracting the first one, we have

$$\sum_{k=1}^{t} (\frac{1}{\delta} + \epsilon_0 - k)|N_k(F)| - \sum_{k=t+1}^{d} (k - \frac{1}{\delta} - \epsilon_0)|N_k(F)| \geq \left(\left(\frac{1}{\delta} + \epsilon_0\right)\delta - 1\right) c|F| \geq \epsilon_0 \delta c|F|,$$

Moreover, it follows by (2) and (4) that

$$\begin{aligned}
&\sum_{k=1}^{t} (\frac{1}{\delta} + \epsilon_0 - k)|N_k(F)| - \sum_{k=t+1}^{d} (k - \frac{1}{\delta} - \epsilon_0)|N_k(F)| \\
&\leq \sum_{k=1}^{t} (\frac{1}{\delta} + \epsilon_0 - k)|N_k(F)| - \sum_{k=d_0-t}^{d} (k - \frac{1}{\delta} - \epsilon_0)|N_k(F)| \\
&\leq (\frac{1}{\delta} + \epsilon_0 - 1)|N_{\leq t}(F)| - (d_0 - t - \frac{1}{\delta} - \epsilon_0)|N_{\geq d_0 - t}(F)| \\
&\leq (\frac{1}{\delta} + \epsilon_0 - 1)|A| - (d_0 - t - \frac{1}{\delta} - \epsilon_0)|B|.
\end{aligned}$$

As $d_0 > \frac{3}{\delta} - 1 + 2\epsilon_0$ and $t = \lfloor \frac{1}{\delta} \rfloor$, we have $d_0 - t - \frac{1}{\delta} - \epsilon_0 > \frac{1}{\delta} + \epsilon_0 - 1$. Combining the above two inequalities, one can infer that

$$\epsilon_0 \delta c|F| \leq (\frac{1}{\delta} + \epsilon_0 - 1)|A| - (d_0 - t - \frac{1}{\delta} - \epsilon_0)|B| \leq (\frac{1}{\delta} + \epsilon_0 - 1)(|A| - |B|) \leq \frac{1}{\delta}(|A| - |B|),$$

which implies that

$$|A| - |B| \geq \epsilon_0 \delta^2 c|F|. \tag{6}$$

On the other hand, since A and B are disjoint subsets of $N(F)$, we have that

$$|A| + |B| \leq |N(F)| \leq c|F|. \tag{7}$$

For every integer $m \in [c]$, let S_m be the set of variables in L that receive exactly m flips. Then the variables in S_m receive a total number of $m|S_m|$ flips. In EasyFlip, every constraint in $A \cup B$ sends exactly one flip to L. The total number of flips sent by constraints in R and received by variables in L is exactly

$$|A| + |B| = \sum_{m=1}^{c} m|S_m|. \tag{8}$$

Let Z be the set of correct variables that receive at least one flip, i.e., $Z = (\cup_{m=1}^{c} S_m) \setminus F$. Observe that the set F' of corrupt variables in the output vector x' consists of corrupt variables not flipped by EasyFlip, which is $F \setminus S_m$, and correct variables that are erroneously flipped by EasyFlip, which is $S_m \cap Z$. Therefore,

$$F' = (F \setminus S_m) \cup (S_m \cap Z). \tag{9}$$

Let α_m be the fraction of corrupt variables in S_m. Then we have that

$$\alpha_m = \frac{|S_m \cap F|}{|S_m|} \quad \text{and} \quad 1-\alpha_m = \frac{|S_m \cap Z|}{|S_m|}. \tag{10}$$

Let β_m denote the fraction of flips sent from A to S_m among all flips received by S_m, i.e.,

$$\beta_m = \frac{\text{the number of flips sent from } A \text{ to } S_m}{m|S_m|}. \tag{11}$$

The following inequality is crucial in the analysis of EasyFlip.

▷ Claim 11. For every $m \in [c]$, $\alpha_m \geq \beta_m$.

Proof. As every variable in S_m receives the same number of m flips, by (10) the number of flips received by $S_m \setminus F$ is $(1-\alpha_m)m|S_m|$. Moreover, by (11) the number of flips sent from B to S_m is $(1-\beta_m)m|S_m|$. Since the constraints in A always compute the correct codewords in C_0, they always send correct flips to their neighbors in L. Therefore, the flips received by $S_m \setminus F$ (which are the wrong flips) must be sent by B, which implies that $(1-\alpha_m)m|S_m| \leq (1-\beta_m)m|S_m|$, where the inequality follows from the fact that B could also send flips to $S_m \cap F$ (which are the correct flips). Thus, $\alpha_m \geq \beta_m$, as needed. ◁

The following result shows that there exists an integer $m \in [c]$ such that there exists a *large* set S_m that contains *many* corrupt variables.

▷ Claim 12. If $|F| \leq \alpha n$, then there exists an integer $m \in [c]$ such that $\alpha_m \geq (1-\epsilon_1)\frac{|A|}{|A|+|B|}$ and $|S_m| \geq \epsilon_2|F|$.

Proof. Suppose for the sake of contradiction that for every $m \in [c]$, we have either $\alpha_m < (1-\epsilon_1)\frac{|A|}{|A|+|B|}$ or $|S_m| < \epsilon_2|F|$. Then, by counting the number of flips sent from A to L (which is exactly $|A|$), we have that

$$\begin{aligned}|A| &= \sum_{m=1}^{c} \beta_m m|S_m| \leq \sum_{m=1}^{c} \alpha_m m|S_m| < (1-\epsilon_1)\frac{|A|}{|A|+|B|}\sum_{m=1}^{c} m|S_m| + \sum_{m=1}^{c} m\epsilon_2|F| \\ &= (1-\epsilon_1)|A| + \frac{c(c+1)}{2}\cdot\epsilon_2|F|,\end{aligned}$$

where the first inequality follows from Claim 11, the second inequality follows from our assumption on α_m and $|S_m|$, and the last equality follows from (8).

Rearranging gives that $|A| < \frac{\epsilon_2(c+1)}{2\epsilon_1}\cdot c|F| = \frac{\delta(t+1)-1}{2t}\cdot c|F|$, contradicting (3). ◁

Next, we will show that by flipping all the variables in S_m, where m satisfies the conclusion of Claim 12, one can reduce the size of the set of corrupt variables by a $(1-\epsilon_3)$-fraction, thereby proving Lemma 10.

Proof of Lemma 10. Let $m \in [c]$ satisfy the conclusion of Claim 12. Combining the two inequalities (6) and (7), one can infer that

$$\frac{|A|}{|A|+|B|} = \frac{1}{2} + \frac{|A|-|B|}{2(|A|+|B|)} \geq \frac{1}{2} + + \frac{\epsilon_0\delta^2 c|F|}{2c|F|} = \frac{1}{2} + \frac{\epsilon_0\delta^2}{2}. \tag{12}$$

Therefore, it follows by (12) that

$$\alpha_m \geq (1-\epsilon_1)\frac{|A|}{|A|+|B|} \geq (1-\epsilon_1)\left(\frac{1}{2}+\frac{\epsilon_0\delta^2}{2}\right). \tag{13}$$

It follows by (9) that

$$\begin{aligned}|F'| &= (|F| - |S_m \cap F|) + |S_m \cap Z| = |F| - (2\alpha_m - 1)|S_m| \\ &\leq |F| - \left(2(1-\epsilon_1)\left(\frac{1}{2} + \frac{\epsilon_0 \delta^2}{2}\right) - 1\right)|S_m| = |F| - (\epsilon_3/\epsilon_2)|S_m| \leq |F| - \epsilon_3|F|,\end{aligned}$$

as needed, where the second equality follows by (10), the first inequality follows by (13), the last equality follows by the definition of ϵ_3 and the last inequality follows by Claim 12. ◀

We will conclude by the following inequality, which shows that for an arbitrary $m \in [c]$, flipping S_m would not significantly increase the number of corrupt variables.

▷ Claim 13. For arbitrary $x \in \mathbb{F}_2^n$ and $m \in [c]$, let $x' := \text{EasyFlip}(x, m)$. Let F and F' be the sets of corrupt variables of x and x', respectively. Then $|F'| \leq (1 + \frac{c}{d_0 - t})|F|$.

Proof. Since the constraints in A always compute the correct codewords in C_0, they always send correct flips to their neighbors in L. Therefore, the wrong flips must be sent by B. Therefore, in the worst case (i.e., assuming that $A = \emptyset$), we have that

$$|F'| \leq |F| + |B| \leq \left(1 + \frac{c}{d_0 - t}\right)|F|,$$

where the second inequality follows from (5). ◁

3.3 Running EasyFlip iteratively for a constant number of times – DeepFlip

In this subsection, we will present and analyze DeepFlip (see Algorithm 3 below), which is designed by running EasyFlip iteratively for s times for a particular choice of $(m_1, \ldots, m_s) \in [c]^s$. By iteratively we mean a sequence of operations $x^0 := x, x^1 := \text{EasyFlip}(x^0, m_1), \ldots, x^s = \text{EasyFlip}(x^{s-1}, m_s)$.

Algorithm 3 Running EasyFlip iteratively for a particular choice $(m_1, \ldots, m_s) \in [c]^s$ – DeepFlip.

```
   Input: $G$, $C_0$, $x \in \mathbb{F}_2^n$, and $(m_1, \ldots, m_s) \in [c]^s$
   Output: $x^s \in \mathbb{F}_2^n$ or $\perp$
1  Set $k = 1$ and $x^0 = x$;
2  for $1 \leq k \leq s$ do
3      $x^k \leftarrow \text{EasyFlip}(x^{k-1}, m_k)$;
4      $U^k \leftarrow \{u \in R : x^k_{N(u)} \notin C_0\}$;
5      if $|U^k| > (1-\epsilon_3)^k \cdot c\gamma n$ then
6          return $\perp$
7      else
8          $k \leftarrow k+1$
9      end
10 end
11 return $x^s$
```

Our goal is to show that as long as the number of corrupt variables in x is not too large, by running EasyFlip iteratively for a large enough (but still constant) number of times, there exists a vector $(m_1, \ldots, m_s) \in [c]^s$ such that the number of corrupt variables in the final output x^s is at most a $(1-\epsilon_3)$-fraction of the number of corrupt variables in the initial input x. Most importantly, later we will show that such a vector $(m_1,\dots,m_s)$ can be found *explicitly* and *efficiently*.

The above assertion will be made precise by the following lemma.

▶ **Lemma 14.** *Let x be the input vector of DeepFlip and let F be the set of corrupt variables of x. If $|F| \le \gamma n$, then for every $s \ge s_0$ there exists a nonempty subset $M \subseteq [c]^s$ such that the following holds for every $(m_1,\dots,m_s) \in M$. Let $x^s :=$DeepFlip$(x,(m_1,\dots,m_s))$ be the output vector of* DeepFlip *and F^s be the set of corrupt variables of x^s. Then $|F^s| \le (1-\epsilon_3)|F|$.*

3.3.1 Proof of Lemma 14

Given $(m_1,\dots,m_s) \in [c]^s$ and $x^0 := x$, for each $k \in [s]$, let $x^k :=$EasyFlip(x^{k-1},m_k). With this notation, $x^s = \text{EasyFlip}(x^{s-1},m_s) = \text{DeepFlip}(x,(m_1,\dots,m_s))$, is exactly the output vector of DeepFlip. Let F be the set of corrupt variables in x and U be the set of unsatisfied constraints with respect to x. Sometimes, we will also use $F^0 := F$ and $U^0 := U$. For $k \in [s]$, define F^k and U^k similarly with x replaced by x^k. Then

$$N_{\le d_0-1}(F) \subseteq U \subseteq N(F), \tag{14}$$

where the first inclusion holds since $d(C_0) = d_0$.

The following lemma can be viewed as an "idealized" version of Lemma 14.

▶ **Lemma 15.** *With the above notation, the following holds. If $|F| \le \alpha n$, then there exists a vector $(m_1,\dots,m_s) \in [c]^s$ such that*

(i) $|F^s| \le (1-\epsilon_3)^s|F|$;

(ii) *for each $k \in [s]$,* $|U^k| \le (1-\epsilon_3)^k \cdot c|F|$;

(iii) $|U^s| \le (1-\epsilon_3)^s \cdot \frac{d_0-1}{\delta d_0-1} \cdot |U|$.

Proof. As $|F| \le \alpha n$, by Lemma 10, there exists $m_1 \in [c]$ such that $x^1 = \text{EasyFlip}(x,m_1)$ satisfies $|F^1| \le (1-\epsilon_3)|F| \le \alpha n$. Continuing this process, it follows by Lemma 10 that for each $k \in [s]$, there exists $m_k \in [c]$ such that $x^k = \text{EasyFlip}(x^{k-1},m_k)$ satisfies

$$|F^k| \le (1-\epsilon_3)|F^{k-1}| \le (1-\epsilon_3)^k|F| \le \alpha n. \tag{15}$$

Such a vector $(m_1,\dots,m_s) \in [c]^s$ clearly satisfies property (i).

To prove (ii), note that it follows by (14) and (15) that for each $k \in [s]$,

$$|U^k| \le |N(F^k)| \le c|F^k| \le (1-\epsilon_3)^k \cdot c|F|.$$

To prove (iii), as $|F| \le \alpha n$, applying Proposition 7 in concert with (14) gives that $\frac{\delta d_0-1}{d_0-1} \cdot c|F| \le |N_{\le d_0-1}(F)| \le |U|$. Combining the equation above and (i) gives that $|U^s| \le c|F^s| \le (1-\epsilon_3)^s \cdot c|F| \le (1-\epsilon_3)^s \cdot \frac{d_0-1}{\delta d_0-1} \cdot |U|$, completing the proof of (iii). ◀

Lemma 15 (i) indicates that there exists an "ideal" choice, say $(m_1^*,\dots,m_s^*) \in [c]^s$, such that if $|F| \le \alpha n$, then after the execution of EasyFlip iteratively for s times (directed by $(m_1^*,\dots,m_s^*)$), the number of corrupt variables in the final output x^s is at most a $(1-\epsilon_3)^s$-fraction of the number of corrupt variables in the initial input $x^0 = x$.

Unfortunately, in general, there is no way to compute the number of corrupt variables in the input and output of each execution of EasyFlip. From this perspective, there is no easy way to *explicitly* find the ideal $(m_1^*,\dots,m_s^*) \in [c]^s$. However, Lemma 15 (iii), which is a consequence of Lemma 15 (i), essentially shows that if the number of corrupt variables reduces *dramatically*, then the number of unsatisfied constraints also reduces *significantly* - fortunately, it is clear that this quantity can be computed in linear time! The analysis of our deterministic decoding algorithm relies heavily on this observation.

The above discussion motivates the following definition.

▶ **Definition 16.** *Given the input vector x of* DeepFlip*, let M be the set consisting of all vectors $(m_1, \dots, m_s) \in [c]^s$ which satisfy the following two properties:*
(a) *for each $k \in [s]$, $|U^k| \leq (1-\epsilon_3)^k \cdot c\gamma n$;*
(b) *$|U^s| \leq \epsilon_4 |U|$, where $\epsilon_4 = \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1-\epsilon_3)$.*

The following result is an easy consequence of Lemma 15.

▷ Claim 17. If $|F| \leq \gamma n$ and $s \geq s_0$, then $M \neq \emptyset$.

Proof. Since $|F| \leq \gamma n < \alpha n$, there exists a vector $(m_1, \dots, m_s) \in [c]^s$ that satisfies Lemma 15. By substituting $|F| \leq \gamma n$ into Lemma 15 (ii), it is easy to see that such a vector also satisfies Definition 16 (a). Moreover, by substituting $s \geq s_0 = \left\lceil \log_{1-\epsilon_3}\left(\epsilon_4 \frac{\delta d_0 - 1}{d_0 - 1}\right)\right\rceil$ into Lemma 15 (iii), it is not hard to see that Definition 16 (b) also holds. Therefore, $M \neq \emptyset$, as needed. ◁

As briefly mentioned above, in general one cannot explicitly find the ideal $(m_1^*, \dots, m_s^*) \in [c]^s$ which dramatically reduces the number of corruptions. Instead, under a stronger condition $|F| \leq \gamma n$ (recall that Lemma 15 assumes $|F| \leq \alpha n$), Lemma 14 shows that for every $(m_1, \dots, m_s) \in M$, $x^s = \text{DeepFlip}(x, (m_1, \dots, m_s))$ reduces the number of corrupt variables of x by a $(1-\epsilon_3)$-fraction, which makes every member of M an *acceptable* (which may be not ideal) choice for DeepFlip.

Now we are ready to present the proof of Lemma 14.

Proof of Lemma 14. First of all, we would like to show that Lemma 14 is well defined, namely, for every $|F| \leq \gamma n$ and $(m_1, \dots, m_s) \in M$, $\text{DeepFlip}(x, (m_1, \dots, m_s))$ does not return $\perp$. Indeed, as $(m_1, \dots, m_s) \in M$, by Definition 16 (a) we have that for every $1 \leq k \leq s$, $|U^k| \leq (1-\epsilon_3)^k \cdot c\gamma n$, which implies that U^k always passes the test in step 5 of Algorithm 3. Therefore, under the assumption of Lemma 14, the output of DeepFlip is a vector $x^s \in \mathbb{F}_2^n$.

To prove the lemma, assume for the moment that $|F^s| \leq \alpha n$. Given the correctness of this assertion, applying Proposition 7 in concert with (14) gives that

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^s| \leq |N_{\leq d_0 - 1}(F^s)| \leq |U^s|.$$

Moreover, by combining the above equation and Definition 16 (b), we have

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^s| \leq |U^s| \leq \epsilon_4 |U| \leq \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1-\epsilon_3) \cdot c|F|,$$

which implies that $|F^s| \leq (1-\epsilon_3)|F|$, as needed.

Therefore, it remains to show that $|F^s| \leq \alpha n$. We will prove by induction that for each $0 \leq k \leq s$, $|F^k| \leq \frac{\alpha n}{1 + c/(d_0 - t)} \leq \alpha n$. For the base case $k = 0$, it follows by assumption that $|F^0| \leq \gamma n < \frac{\alpha n}{1 + c/(d_0 - t)}$ as $d_0 \geq 3$, $\delta d_0 > 3$ and $t = \lfloor \frac{1}{\delta} \rfloor$. Suppose that for some $k \in [s]$ we have $|F^{k-1}| \leq \frac{\alpha n}{1 + c/(d_0 - t)}$. Since $x^k = \text{EasyFlip}(x^{k-1}, m_k)$, it follows by Claim 13 that $|F^k| \leq (1 + \frac{c}{d_0 - t})|F^{k-1}| \leq \alpha n$. Therefore, we have

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^k| \leq |N_{\leq d_0 - 1}(F^k)| \leq |U^k| \leq (1-\epsilon_3)^k \cdot c\gamma n \leq c\gamma n,$$

where the first inequality follows from Proposition 7, the second inequality follows from (14), and the third inequality follows from Definition 16 (a). The last equation implies that $|F^k| \leq \frac{d_0-1}{\delta d_0 - 1}\gamma n < \frac{d_0}{2}\gamma n \leq \frac{\alpha n}{1+c/(d_0-t)}$, as needed, where the second inequality follows from the assumption $\delta d_0 > 3$ and the last inequality follows from the definition of γ in Theorem 3, $\delta d_0 > 3$ and $t = \lfloor \frac{1}{\delta} \rfloor$. The proof of the lemma is thus completed. ◀

3.4 Running DeepFlip thoroughly until significantly reducing the number of unsatisfied constraints – HardSearch

In this subsection, we describe and analyze HardSearch (see Algorithm 4 below). Given an input vector $x \in \mathbb{F}_2^n$ with at most γn corruptions, HardSearch runs DeepFlip$(x, (m_1, \dots, m_s))$ over all choices of $(m_1, \dots, m_s) \in [c]^s$ until it finds one, say $(m'_1, \dots, m'_s)$, such that the number of unsatisfied constraints with respect to DeepFlip$(x, (m'_1, \dots, m'_s))$ is at most an ϵ_4-fraction of the number of unsatisfied constraints with respect to x. Then Lemma 8 shows that the number of corruptions in x' is at most a $(1-\epsilon_3)$-fraction of the number of corruptions in x. Therefore, running HardSearch iteratively for ℓ rounds gives us a $(1-\epsilon_3)^\ell$-reduction on the number of corruptions.

Algorithm 4 Running DeepFlip over all $(m_1, \dots, m_s) \in [c]^s$ until finding an "acceptable" one – HardSearch.

```
   Input: $G$, $C_0$, $x \in \mathbb{F}_2^n$, and $s = s_0$
   Output: $x' \in \mathbb{F}_2^n$
1  $U \leftarrow \{u \in R : x_{N(u)} \notin C_0\}$;
2  for every $(m_1, \dots, m_s) \in [c]^s$ do
3      $x' \leftarrow$ DeepFlip$(x, (m_1, \dots, m_s))$;
4      if $x' \neq \perp$ then
5          $U' \leftarrow \{u \in R : x'_{N(u)} \notin C_0\}$;
6          if $|U'| \leq \epsilon_4 |U|$ then
7              return $x'$
8          end
9      end
10 end
```

3.4.1 Proof of Lemma 8

To prove (i), let M be the set of vectors in $[c]^s$ which satisfy the two conditions in Definition 16 with respect to F and s, where $|F| \leq \gamma n$ and $s = s_0$. By our choices of F and s, it follows by Claim 17 that $M \neq \emptyset$. By Lemma 14, as long as HardSearch finds a vector $(m_1, \dots, m_s) \in M$, it would output a vector $x' =$ DeepFlip$(x, (m_1, \dots, m_s))$ such that $|U'| \leq \epsilon_4 |U|$[4] and $|F'| \leq (1-\epsilon_3)|F|$, as needed.

It remains to prove (ii), which is an easy consequence of (i). Let F^i be the set of corruptions in x^i for all $0 \leq i \leq \ell$. Then by (i) for every $0 \leq i \leq \ell - 1$, we have either $x^i \in T(G, C_0)$ (if $|U^i| = 0$) or $|F^{i+1}| \leq (1-\epsilon_3)|F^i|$ (if $|U^i| \neq 0$). Therefore, after at most $\ell = \left\lceil \log_{1-\epsilon_3}\left(\lfloor \frac{d_0-1}{2} \rfloor \frac{1}{\gamma n}\right) \right\rceil$ iterative executions of HardSearch, the number of corrupt variables is at most $(1-\epsilon_3)^\ell \gamma n \leq \lfloor \frac{d_0-1}{2} \rfloor \frac{1}{\gamma n} \cdot \gamma n = \lfloor \frac{d_0-1}{2} \rfloor$, as needed.

[4] This holds since $(m_1, \dots, m_s) \in M$ satisfies Definition 16 (b).

4 Randomized decoding: Reduce large corruptions to a moderate size

In this section, we present our randomized decoding for Tanner codes which can correct more errors. The general strategy is as follows. First, we use a voting process to derive a set $S := \cup_{m \in [c]} S_m$ of candidate variables to flip. More precisely, each constraint $u \in R$ that satisfies $1 \le d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \le t$ sends exactly one flip to an arbitrary variable $v \in N(u)$ with $\text{Decode}(x_{N(u)})_v \ne x_v$. We then design a special sampling process to pick a large fraction of variables from S and flip them. This process can, with high probability, reduce the number of corrupted variables by a positive fraction. We repeat the above random process until the number of corrupted variables drops below γn, in which case our deterministic decoding MainDecode in Algorithm 1 can work correctly, or we run out of time and stop. Finally, we use MainDecode to get the codeword.

Let γ be the relative decoding radius of Theorem Theorem 3. The exact randomized decoding is given as Algorithm 5, which yields the following result.

Algorithm 5 Randomized Decoding.

```
Input: $x \in \mathbb{F}_2^n$ with at most $\alpha$ fraction errors
Output: a codeword in $T(G, C_0)$ or $\perp$
Set $t = \lfloor \frac{1}{\delta} \rfloor$;
for $\ell = 1, \ldots, \left\lceil \frac{\log \frac{\gamma}{\alpha}}{\log\left(1 - \frac{3\epsilon(\delta(t+1)-1)}{4t}\right)} \right\rceil$ do
    for every $u \in R$ do
        $\omega \leftarrow \text{Decode}(x_{N(u)})$;
        if $1 \le d_H(\omega, x_{N(u)}) \le t$ then
            send a "flip" message to the vertex $v \in N(u)$ with the smallest index such
              that $\omega_v \ne x_v$
        end
    end
    $\forall m \in [c], S_m \leftarrow \{v \in L : v$ receives $m$ "flip" messages$\}$;
    $S \leftarrow \bigcup_m S_m$;
    Randomly pick $P \subseteq S$: for every $m \in [c]$, for each variable in $S_m$, pick it with
      probability $\frac{m}{2c}$, using independent randomness;
    Flip all bits in $P$;
    $U \leftarrow \{u \in R : x_{N(u)} \notin C_0\}$;
    if $|U| \le \left(\delta - \frac{1}{d_0}\right) c\gamma n$ then
        return MainDecode($x$)
    end
end
return $\perp$
```

The following two lemmas demonstrate the correctness and linear running time of Algorithm 5, respectively[5].

▶ **Lemma 18.** *If the input has distance at most αn from a codeword, then with probability $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$, Algorithm 5 outputs the correct codeword.*

[5] We only prove Lemma 18. The proof of Lemma 19 can be found in the full version of this paper [10]

▶ **Lemma 19.** *If the input has distance at most αn from a codeword, then Algorithm 5 runs in linear time.*

Assuming the correctness of the above lemmas, we can prove Theorem 6 as follows.

Proof of Theorem 6. If the input word has at most αn errors, then by Lemma 18, with probability $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$, the decoding outputs the correct codeword. Furthermore, the running time is linear by Lemma 19. ◀

4.1 Proof of Lemma 18

Recall that we defined A as the set of constraints $u \in R$ that sends a flip and Decode$(x_{N(u)})$ computes the correct codeword in C_0 (see (2)), and B to be the set of constraints $u \in R$ that sends a flip and Decode$(x_{N(u)})$ computes an incorrect codeword in C_0 (see (4)). Also, recall we let α_m denote the fraction of corrupt variables in S_m (see (10)) and we let β_m denote the fraction of flips sent from A to S_m among all flips received by S_m (see (11)). Now we let $M := A \cup B$.

First, we bound the size of P in an arbitrary iteration.

▷ Claim 20. For every constant $\epsilon > 0$, with probability $\geq 1 - \exp\{-\Theta_{c,\epsilon}(|M|)\}$, the size of P is in $\left[(1-\epsilon)\frac{|M|}{2c}, (1+\epsilon)\frac{|M|}{2c}\right]$.

Proof. In Algorithm 5, for every $m \in [c]$, each variable in S_i is picked independently with probability $\frac{m}{2c}$. For each $v \in S$, let X_v be the indicator random variable of the event that the variable v is picked. So for every $v \in S_m$, $\Pr[X_v = 1] = \frac{m}{2c}$. Let $X = \sum_{v \in S} X_v$. It is easy to see that $X = |P|$. By the linearity of expectation, we have that

$$\mathrm{E}X = \sum_{v \in S} \mathrm{E}X_v = \sum_{m \in [c]} \frac{m}{2c}|S_m| = \frac{|M|}{2c}.$$

By Hoeffding's inequality,

$$\Pr\left[X \in \left[(1-\epsilon)\frac{|M|}{2c}, (1+\epsilon)\frac{|M|}{2c}\right]\right] \geq 1 - 2\exp\left\{-\frac{2\left(\epsilon\frac{|M|}{2c}\right)^2}{|S|}\right\} \geq 1 - 2\exp\left\{-\frac{\epsilon^2|M|}{2c^2}\right\},$$

where the second inequality follows from the fact that $|S| \leq |M|$. ◁

Next, we show that P contains significantly more corrupted variables than correct variables.

▷ Claim 21. There exists a constant ϵ such that with probability $\geq 1 - \exp\{-\Theta_{c,\epsilon}(|M|)\}$, the number of corrupted variables in P is at least $(1/2 + \epsilon)\frac{|M|}{2c}$.

Proof of Claim 21. For every $v \in S$, let Y_v be the indicator random variable of the event that $X_v = 1$ and $v \in F$. Let $Y = \sum_{v \in S} Y_v$. By definition, $Y = P \cap F$. Note that for every $v \notin S \cap F$, $\Pr[Y_v = 1] = 0$. By the linearity of expectation, we have that

$$\mathrm{E}Y = \sum_{v \in S} \mathrm{E}Y_v = \sum_{m \in [c]} \sum_{v \in S_m} \mathrm{E}Y_v = \sum_{m \in [c]} \sum_{v \in S_m \cap F} \frac{m}{2c} = \sum_{m \in [c]} \frac{m}{2c}\alpha_m|S_m| \geq \sum_{m \in [c]} \frac{m}{2c}\beta_m|S_m|, \tag{16}$$

where the inequality follows from Claim 11.

By the definition of β_m, $m\beta_m|S_m| = |\{$ The number of "flips" sent from A to $S_m\}|$. Hence, one can infer that

$$\begin{aligned}\mathrm{E}Y &\geq \sum_{m\in[c]} \frac{|\{\text{ The number of "flips" sent from } A \text{ to } S_m\}|}{2c}\\ &= \frac{|\{\text{ The number of "flips" sent from } A \text{ to } S\}|}{2c} = \frac{|A|}{2c},\end{aligned}$$

where the last equality is due to that each constraint in R can only send at most 1 message.

Set $\epsilon = \frac{\epsilon_0\delta^2}{4} > 0$. It follows by (12) that $|A| \geq \left(\frac{1}{2} + \frac{\epsilon_0\delta^2}{2}\right)|M| = \left(\frac{1}{2} + 2\epsilon\right)|M|$. Thus, we can infer that $\mathrm{E}Y \geq \frac{|A|}{2c} \geq \left(\frac{1}{2} + 2\epsilon\right)\frac{|M|}{2c}$.

By Hoeffding's inequality,

$$\Pr\left[Y \leq \left(\frac{1}{2} + \epsilon\right)\frac{|M|}{2c}\right] \geq 1 - \exp\left\{-\frac{2\left(\epsilon\frac{|M|}{2c}\right)^2}{|S|}\right\} \geq 1 - \exp\left\{-\frac{\epsilon^2|M|}{2c^2}\right\},$$

where the second inequality follows from that $|S| \leq |M|$. ◁

The following claim shows that as long as the number of unsatisfied constraints is small enough, we can ensure that the number of corrupt variables is at most γn. Hence, we can handle the matter with Algorithm 1.

▷ Claim 22. If $|U| \leq \left(\delta - \frac{1}{d_0}\right)c\gamma n$ and $|F| \leq \alpha n$, then $|F| \leq \gamma n$.

Proof. Suppose that $\gamma n < |F| \leq \alpha n$. By Proposition 7, we have that

$$|U| \geq \frac{\delta d_0 - 1}{d_0 - 1}c|F| > \left(\delta - \frac{1}{d_0}\right)c\gamma n,$$

which is a contradiction. ◁

Now, we can give the proofs of Lemma 18 and Lemma 19, respectively, as follows.

Proof of Lemma 18. In each iteration, consider the case that the number of errors $|F|$ is at most αn. If $|U| \leq \left(\delta - \frac{1}{d_0}\right)c\gamma n$, then by Claim 22, $|F| \leq \gamma n$. Therefore, it follows by Theorem 3 that when $\delta d_0 > 3$, all errors can be corrected. Otherwise, we claim that the number of corrupt variables can be decreased by a constant fraction in this iteration.

Recall that $M = A \cup B \subseteq U$. It follows by (3) that

$$|M| = |A| + |B| \geq \frac{\delta(t+1) - 1}{t}c|F|. \tag{17}$$

Note that by Claim 21, with probability $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$, the number of corruptions in P is at least $(1/2 + \epsilon)\frac{|M|}{2c}$ where $\epsilon > 0$ is a constant. Also note that by Claim 20, with probability $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$, the size of P is in $\left[(1 - \epsilon/2)\frac{|M|}{2c}, (1 + \epsilon/2)\frac{|M|}{2c}\right]$. When both of these events occur, by flipping all variables in P, the number of corruptions is reduced by at least $\frac{3\epsilon|M|}{4c}$. It follows by (17) that $\frac{3\epsilon|M|}{4c} \geq \frac{3\epsilon(\delta(t+1)-1)}{4t}|F|$. This shows the number of corrupt variables indeed is decreased by a constant fraction in this iteration.

As a result, after at most $\frac{\log\frac{\gamma}{\alpha}}{\log\left(1 - \frac{3\epsilon(\delta(t+1)-1)}{4t}\right)}$ iterations, the number of corruptions is at most γn. Then the decoding can call Algorithm 1 to correct all errors. ◀

References

1 Noga Alon and Michael Capalbo. Explicit unique-neighbor expanders. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 73–79. IEEE, 2002.

2 Sanjeev Arora, Constantinos Daskalakis, and David Steurer. Message-passing algorithms and improved lp decoding. *IEEE Trans. Inf. Theory*, 58(12):7260–7271, 2012. `doi:10.1109/TIT.2012.2208584`.

3 Ron Asherov and Irit Dinur. Bipartite unique neighbour expanders via ramanujan graphs. *Entropy*, 26(4):348, 2024.

4 Oren Becker. Symmetric unique neighbor expanders and good ldpc codes. *Discrete Applied Mathematics*, 211:211–216, 2016.

5 Avraham Ben-Aroya and Amnon Ta-Shma. A combinatorial construction of almost-ramanujan graphs using the zig-zag product. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 325–334, 2008.

6 Nikolas P. Breuckmann and Jens Niklas Eberhardt. Balanced product quantum codes. *IEEE Trans. Inf. Theory*, 67(10):6653–6674, 2021. `doi:10.1109/TIT.2021.3097347`.

7 Nikolas P Breuckmann and Jens Niklas Eberhardt. Quantum low-density parity-check codes. *PRX Quantum*, 2(4):040101, 2021.

8 Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 659–668, 2002.

9 Xue Chen, Kuan Cheng, Xin Li, and Minghui Ouyang. Improved decoding of expander codes. *IEEE Trans. Inf. Theory*, 69(6):3574–3589, 2023. `doi:10.1109/TIT.2023.3239163`.

10 Kuan Cheng, Minghui Ouyang, Chong Shangguan, and Yuanting Shen. Improved decoding of expander codes: fundamental trade-off between expansion ratio and minimum distance of inner code. *arXiv preprint*, 2023. `arXiv:2312.16087`.

11 Shashi Kiran Chilappagari, Dung Viet Nguyen, Bane Vasic, and Michael W. Marcellin. On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes. *IEEE Trans. Inf. Theory*, 56(4):1600–1611, 2010. `doi:10.1109/TIT.2010.2040962`.

12 Sae-Young Chung, G David Forney, Thomas J Richardson, and Rüdiger Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE Communications letters*, 5(2):58–60, 2001.

13 Itay Cohen, Roy Roth, and Amnon Ta-Shma. Hdx condensers. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1649–1664. IEEE, 2023.

14 Alexandros G. Dimakis, Roxana Smarandache, and Pascal O. Vontobel. Ldpc codes for compressed sensing. *IEEE Trans. Inf. Theory*, 58(5):3093–3114, 2012. `doi:10.1109/TIT.2011.2181819`.

15 Irit Dinur, Min-Hsiu Hsieh, Ting-Chun Lin, and Thomas Vidick. Good quantum LDPC codes with linear time decoders. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 905–918. ACM, 2023. `doi:10.1145/3564246.3585101`.

16 Michael Dowling and Shuhong Gao. Fast decoding of expander codes. *IEEE Trans. Inf. Theory*, 64(2):972–978, 2018. `doi:10.1109/TIT.2017.2726064`.

17 Shai Evra, Tali Kaufman, and Gilles Zémor. Decodable quantum LDPC codes beyond the $\sqrt{n}$ distance barrier using high dimensional expanders. *CoRR*, abs/2004.07935, 2020. `arXiv:2004.07935`.

18 Jon Feldman, Tal Malkin, Rocco A. Servedio, Cliff Stein, and Martin J. Wainwright. Lp decoding corrects a constant fraction of errors. *IEEE Trans. Inf. Theory*, 53(1):82–89, 2007. `doi:10.1109/TIT.2006.887523`.

19 Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

20 Louis Golowich. New explicit constant-degree lossless expanders. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4963–4971. SIAM, 2024.

21 Shouzhen Gu, Christopher A. Pattison, and Eugene Tang. An efficient decoder for a linear distance quantum LDPC code. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 919–932. ACM, 2023. doi:10.1145/3564246.3585169.

22 Venkatesan Guruswami. Iterative decoding of low-density parity check codes (a survey). *arXiv preprint cs/0610022*, 2006.

23 Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 812–821. ACM, 2002. doi:10.1145/509907.510023.

24 Matthew B. Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the $n^{1/2}$ polylog(n) barrier for quantum LDPC codes. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1276–1288. ACM, 2021. doi:10.1145/3406325.3451005.

25 Jun-Ting Hsieh, Theo McKenzie, Sidhanth Mohanty, and Pedro Paredes. Explicit two-sided unique-neighbor expanders. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 788–799, 2024.

26 Tali Kaufman and Izhar Oppenheim. Construction of new local spectral high dimensional expanders. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 773–786, 2018.

27 Tali Kaufman and Ran J. Tessler. New cosystolic expanders from tensors imply explicit quantum LDPC codes with $\Omega(\sqrt{n}\log^k n)$ distance. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1317–1329. ACM, 2021. doi:10.1145/3406325.3451029.

28 Swastik Kopparty, Noga Ron-Zewi, and Shubhangi Saraf. Simple constructions of unique neighbor expanders from error-correcting codes. *arXiv preprint*, 2023. arXiv:2310.19149.

29 Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. Quantum expander codes. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 810–824. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.55.

30 Anthony Leverrier and Gilles Zémor. Quantum tanner codes. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 872–883. IEEE, 2022. doi:10.1109/FOCS54457.2022.00117.

31 Anthony Leverrier and Gilles Zémor. Decoding quantum tanner codes. *IEEE Trans. Inf. Theory*, 69(8):5100–5115, 2023. doi:10.1109/TIT.2023.3267945.

32 Anthony Leverrier and Gilles Zémor. Efficient decoding up to a constant fraction of the code length for asymptotically good quantum codes. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1216–1244. SIAM, 2023. doi:10.1137/1.9781611977554.CH45.

33 Ting-Chun Lin and Min-Hsiu Hsieh. Good quantum ldpc codes with linear time decoder from lossless expanders. *arXiv preprint*, 2022. arXiv:2203.03581.

34 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

35 Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, 2001.

36 GA Margulis. Explicit constructions of expanders (russian), problemy peredaci informacii 9 (1973), no. 4, 71–80. *MR0484767*, 1973.

37 Moshe Morgenstern. Existence and explicit constructions of q+ 1 regular ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994.

38 Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. Ldpc codes achieve list decoding capacity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 458–469, 2020. doi:10.1109/FOCS46700.2020.00050.

39 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 375–388. ACM, 2022. doi:10.1145/3519935.3520017.

40 Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Trans. Inf. Theory*, 68(1):213–229, 2022. doi:10.1109/TIT.2021.3119384.

41 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 3–13. IEEE, 2000.

42 Thomas J Richardson, Mohammad Amin Shokrollahi, and Rüdiger L Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inf. Theory*, 47(2):619–637, 2001.

43 Thomas J Richardson and Rüdiger L Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory*, 47(2):599–618, 2001.

44 Noga Ron-Zewi, Mary Wootters, and Gilles Zémor. Linear-time erasure list-decoding of expander codes. *IEEE Trans. Inf. Theory*, 67(9):5827–5839, 2021. doi:10.1109/TIT.2021.3086805.

45 Ron M. Roth and Vitaly Skachek. Improved nearly-mds expander codes. *IEEE Trans. Inf. Theory*, 52(8):3650–3661, 2006. doi:10.1109/TIT.2006.878232.

46 Michael Sipser and Daniel A Spielman. Expander codes. *IEEE Trans. Inf. Theory*, 42(6):1710–1722, 1996.

47 Vitaly Skachek. Minimum distance bounds for expander codes. In *2008 Information Theory and Applications Workshop*, pages 366–370. IEEE, 2008.

48 DA Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996.

49 R Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27(5):533–547, 1981.

50 Michael Viderman. Linear-time decoding of regular expander codes. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–25, 2013.

51 Michael Viderman. LP decoding of codes with expansion parameter above 2/3. *Inf. Process. Lett.*, 113(7):225–228, 2013. doi:10.1016/J.IPL.2013.01.012.

52 Gilles Zémor. On expander codes. *IEEE Trans. Inf. Theory*, 47(2):835–837, 2001. doi:10.1109/18.910593.

53 Gillés Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.