SIGACT News Complexity Theory Column 45

Lane A. Hemaspaandra
Dept. of Computer Science, University of Rochester
Rochester, NY 14627, USA lane@cs.rochester.edu

Introduction to Complexity Theory Column 45

Sincere thanks to Venkatesan Guruswami for writing this issue's very interesting article. Upcoming guest articles include John M. Hitchcock, Jack H. Lutz, and Elvira Mayordomo on *The Fractal Geometry of Complexity Classes*; Scott Aaronson on *NP-Complete Problems and Physical Reality*; Neil Immerman on *Recent Progress in Descriptive Complexity*; and Subhash Khot on *Recent results in PCPs and Hardness of Approximating NP-Hard Problems.*

Finally, let me mention that the wonderful complexity theorist (and very nice person) Jacobo Torán has taken over from Lance Fortnow as the editor of the sister complexity theory column that appears in the issues of the Bulletin of the EATCS. Jacobo's tenure started with the June 2004 issue of BEATCS, and in that issue you can find his article on "Space and Width in Propositional Resolution." A warmest welcome to Jacobo, and let me enthusiastically commend our sister column to all readers of this column.

Guest Column: Error-correcting Codes and Expander Graphs[1]

Venkatesan Guruswami[2]

Abstract

The central algorithmic problem in coding theory is the construction of explicit error-correcting codes with good parameters together with fast algorithms for encoding a message and decoding a noisy received word into the correct message. Over the last decade, significant new developments have taken place on this problem using graph-based/combinatorial constructions that exploit the power of *expander graphs.* The role of expander graphs in theoretical computer science is by now certainly well-appreciated. Here, we will survey some of the highlights of the use of expanders in constructions of error-correcting codes.

1 Introduction

The area of coding theory was born over 50 years ago in the seminal works of Shannon and Hamming, and the problem of explicit constructions of codes with good properties and fast algorithms has been a central pursuit of the subject ever since. Recently, this area has benefited enormously from insights and viewpoints originating in complexity theory, and numerous interconnections between codes and complexity theory have been discovered, which are surveyed for example in [25, 27].

[1]© V. Guruswami, 2004.

[2]Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195. venkat@cs.washington.edu.

The former by Sudan [25] focuses on a notion called *list decoding*, and the recent survey by Trevisan [27] focuses mainly on *sub-linear algorithms for testing and decoding codes.* In this survey, we focus on a powerful method for constructing and decoding codes, based on *expander graphs.* Expander graphs have been the subject of much study in combinatorics and computer science, and have found applications in diverse contexts. There have been significant recent breakthroughs in explicit constructions of expanders [19, 8], and the interested reader can find further pointers on expanders and their applications in those papers. This survey will focus on their uses in and interconnections to coding theory. Spielman [23] wrote a brief survey on this topic soon after his discovery with Sipser of linear-time decodable codes [21, 24]; there have been numerous improvements and developments since which we hope to highlight here.

Expanders are an umbrella term to denote sparse, yet well-connected graphs, which behave like random graphs in certain respects. At a high level, their incidence structure can be used to define either linear constraints (like parity checks) on subsets of codeword symbols, or to move symbols around during the encoding. The expansion properties make both of these approaches fruitful, and till date the resulting constructions give the only known way to get codes with linear complexity encoding/decoding algorithms.

Organization. We begin with some quick background on codes and expanders in Section 2. Section 3 describes how expanders can be obtained from codes, which is the reverse direction of the connection surveyed in the rest of the paper. In Sections 4 and 5, we describe how expanders can be used to construct near-optimal codes (in terms of rate vs. error-correctability trade-off) that also have linear time encoding/decoding algorithms. We then describe the recent development of linear-time list-decodable codes using expanders in Section 6. Finally, we mention results for probabilistic models where expander codes achieve capacity with a very good error exponent and linear time decoding.

2 Preliminaries on Codes and Expanders

2.1 Error-correcting codes

An error-correcting code C of *block length* n over an *alphabet* Σ is a collection of strings called *codewords*, each of which consists of n symbols from Σ. If $|\Sigma| = q$ and $|C| = q^k$, then such a code is referred to as an $(n,k)_q$ code. The quantity k measures the number of information symbols in each codeword, and the ratio k/n is called the *rate* of the code, and is denoted $R(C)$. When Σ is a field, and C is a subspace of dimension k of the vector space Σ^n, then C is called a *linear code* and is denoted an $[n,k]_q$ code. Such a code can be described by an $n \times k$ generator matrix $\mathbf{G}$ of rank k over Σ whose columns span C, i.e., $C = \{\mathbf{G}\mathbf{x} \mid \mathbf{x} \in \Sigma^k\}$. It can also be described as the kernel of an $(n-k) \times n$ parity check matrix $\mathbf{H}$ of rank $(n-k)$, i.e., $C = \{\mathbf{z} \in \Sigma^n \mid \mathbf{H}\mathbf{z} = \mathbf{0}\}$. We will often also think of C as a map $C : \Sigma^k \to \Sigma^n$, and denote by $C(m)$ the encoding of a message $m \in \Sigma^k$, and by $C(m)_{|i}$ the i'th symbol of $C(m)$ for $1 \le i \le n$.

The *distance* $D(C)$ of a code is the minimum Hamming distance between two distinct codewords of C. The *relative distance* of C is defined as $\delta(C) = d(C)/n$. While we would like both the rate and distance to be large, there is a natural trade-off between these. The *Singleton bound* implies that $R(C) \le 1-\delta(C)$, and over large alphabets there are codes that match the Singleton bound (and are thus optimal). For binary codes, the *Gilbert-Varshamov* bound asserts the existence of codes with rate $R(C) \ge 1 - H(\delta(C))$ where $H(x) = -x \lg x - (1-x)\lg(1-x)$ is the binary entropy function. This bound is non-constructive, and the best known explicit constructions achieve a weaker bound called the *Zyablov* bound (and some related improvements of it). For a code $C \subset \Sigma^n$ of distance d, the *unique decoding* problem consists of finding, given as input a received word $\mathbf{y} \in \Sigma^n$, the unique codeword $\mathbf{c} \in C$ if any that lies within Hamming distance $(d-1)/2$ from $\mathbf{y}$. For error bounds e that exceed $d/2$, the *list decoding* problem requires that we output all codewords of C within distance e of the received word.

Both of the above decoding problems refer to the *adversarial noise model*, where we only assume a bound on the number of errors and not how they are distributed. For a probabilistic noise model, the intervening noisy channel is abstracted by a stochastic process and the goal is to construct codes and decoding algorithms for the specific channel with very low probability of miscommunication (called the *error probability*). Every channel has an underlying *capacity* which is the largest rate of a code that can be used and still achieve error probability tending to 0 (as the block length of the code grows). Ideally, we would like the error probability $P_e(N)$ to go down exponentially in the block length N , and the *error exponent* is then defined as $\inf_{N \to \infty} \lg(1/P_e)/N$. A basic and extremely well-studied channel is the *binary symmetric channel*, denoted BSC_p where p, $0 < p < 1/2$, is the crossover probability, which flips a transmitted bit with probability p and leaves it unaltered with probability $(1-p)$. It is well-known that the capacity of BSC_p equals $1 - H(p)$. In our discussion here, in Section 7, we will discuss linear-time encodable/decodable codes that can achieve the capacity of BSC_p and which match the best known error exponents for explicit, polynomial time decodable, codes.

2.2 Expanders

At a high level, expanders are sparse graphs with very good connectivity properties. The precise definition of the term expander varies according to the application. We will make use of two definitions given below. A bipartite graph $H = (X, Y, E)$ is said to be an $(n, m, d, \alpha, \gamma)$-*expander* if $|X| = n$, $|Y| = m$, the degree of each node in X is d, and for every $A \subseteq X$, $|A| \leq \alpha n$, the set of vertices $N(A) \subseteq Y$ that are adjacent to A satisfies $|N(A)| \geq \gamma |A|$. A higher value of γ implies a better expansion property; clearly $\gamma \leq d$ and when γ is very close to d (say $d(1-\varepsilon)$ for small ε), the expander is called *lossless*.

One way to construct such unbalanced expanders, which was used by Sipser and Spielman [21] in their first work on expander codes, is to take the edge-vertex incidence matrix of a non-bipartite expander defined according to its *spectral gap*. This spectral definition of expanders is the most commonly referred to one. A d-regular graph $G = (V, E)$ on n vertices is said to be an (n, d, λ)-*expander* if the second largest eigenvalue (in absolute value) of the adjacency matrix of G equals λd. Note that the largest eigenvalue (with all 1's eigenvector) equals d, and therefore $\lambda \leq 1$. The smaller the value of λ the better the quality of the expander, and typically for our applications we will require expanders where $\lambda \to 0$ as d grows. Such expanders have the following useful "pseudorandom" property (as opposed to just vertex-expansion): Given two subsets A and B of the vertex set V, the number of edges $E(A, B)$ between A and B is very close to the expected number of such edges in a random d-regular graph, cf. [4, Chap. 9]. This property will be quantified in Section 4.2 (see Equation (1)). Rather surprisingly, the best possible value of λ is known to be precisely $2\sqrt{d-1}/d$, and this can be achieved by explicit constructions [18].

3 Expanders from codes

Before we survey the constructions of codes using expanders, in this section we describe a simple construction of logarithmic degree expanders from binary codes due to Alon and Roichman [3].

Theorem 1 *Let* $\mathbf{M} \in \mathbb{F}_2^{n \times k}$ *be the generator matrix of a binary linear code of block length n all of whose non-zero codewords have Hamming weight between* $(1/2 - \varepsilon)n$ *and* $(1/2 + \varepsilon)n$ *(in particular,* *such a code has minimum distance at least $(1/2-\varepsilon)n$). Consider the graph $G=(V,E)$ with vertex set $V=\mathbb{F}_2^k$ such that $(x,y)\in E$ precisely when $x-y$ is one of the rows of $\mathbf{M}$ (in other words G is the Cayley graph of the additive group $\mathbb{F}_2^k$ with respect to generators that are the n rows of $\mathbf{M}$). Then G is a $(2^k,n,2\varepsilon)$-expander.*

Proof Sketch: The characters of the additive group $\mathbb{F}_2^k$ are given by χ_a for $a\in\mathbb{F}_2^k$ where $\chi_a(x)=(-1)^{a\cdot x}$. The claim follows by combining two easily verified observations: (a) The 2^k eigenvectors of $\mathbf{M}$ are all the characters χ_a, with corresponding eigenvalue $\sum_{i=1}^n \chi_a(b_i)$ where $b_1,b_2,\ldots,b_n$ are the n rows of $\mathbf{M}$, and (b) $\sum_{i=1}^n \chi_a(b_i)=n-2\cdot\text{weight}(\mathbf{M}a)$. $\square$

There are explicit constructions of binary codes with the stated property where $n=O(k/\varepsilon^3)$ (for instance, using certain algebraic-geometric codes concatenated with a Hadamard code). Since the number of nodes in the graph G is $N=2^k$, we can get degree $O(\log N)$ expanders in this manner.

4 Linear time encodable/decodable expander codes

Expander codes are the only known construction of asymptotically good error-correcting codes which can be decoded in linear time when a constant fraction of symbols are in error. In this section, we discuss how such codes are constructed using expanders; for simplicity, we will focus on binary codes.

4.1 The basic construction: Expanders as parity check matrices

A binary linear code $C\subseteq\mathbb{F}_2^n$ can be specified by its parity check matrix or in other words the linear constraints that each codeword must satisfy. These constraints can be naturally viewed in terms of a bipartite graph with codeword positions on one side and parity checks on the other side, and adjacency reflecting which symbol is involved in which parity check. When this graph is sparse (say, of constant degree), the resulting codes are called low-density parity check codes (LDPC codes) [11]. In addition, when the underlying graph is a sufficiently good expander, the resulting code can be decoded in linear time by a natural and simple algorithm, as described below.

Theorem 2 *[21, 8] Let $H=(X,Y,E)$ be a $(n,m,d,\alpha,d(1-\varepsilon))$-expander for some fixed ε, $0<\varepsilon<1/12$.[3] Let $C\subseteq\mathbb{F}_2^n$ be defined by $\{c\in\mathbb{F}_2^n \mid \forall i, 1\le i\le m, \sum_{j\in\Gamma(i)} c_j=0\}$ where for $i=1,2,\ldots,m$, $\Gamma(i)\subseteq[n]$ is the set of neighbors in X of the i'th node in Y (the sum is calculated in the field $\mathbb{F}_2$). Then C has rate at least $(1-m/n)$ and can be decoded from a fraction $\alpha(1-3\varepsilon)$ of errors in $O(n)$ time.*

Proof Idea: The claim about rate follows since there are at most m linearly independent constraints posed on the codewords of C and therefore C has dimension at least $n-m$. We will associate the bits of the word being decoded with nodes in X (called variable nodes) and the values of the parity checks with nodes in Y (called check nodes) in the natural way. The algorithm will proceed in several rounds of "bit-flipping," and in each round will reduce the number of corrupt bits (of X) by a constant factor, till all errors are corrected. So the number of rounds will be $O(\log n)$. The operation of a single round is as follows. Each node in X does the following in *parallel*: if flipping its value reduces the number of unsatisfied parity checks among its d neighbors by at least

[3]The $1/12$ has not been optimized, and it is conceivable that a similar statement can be shown for weaker expansion, i.e., somewhat larger ε, as well. The argument of [21] needed $\varepsilon<1/4$.

$d/3$, then it flips its value, otherwise it retains its value.[4] Each round clearly has an $O(n)$ time implementation, leading to an $O(n \log n)$ time decoding algorithm. The runtime can be improved to $O(n)$ by a careful examination of which nodes need to participate in each round (the size of this set will decrease geometrically, leading to a bound of $O(n)$ on total number of operations).

To prove correctness of the above decoding procedure, we will argue that if the set $S \subset X$ of error positions satisfies $|S| \le \alpha(1-3\varepsilon)n$, then the set S' of error positions after one bit-flipping round will satisfy $|S'| \le 2|S|/3$. Let $N(S) \subseteq Y$ denote the set of neighbors of S. By the expansion property, $|N(S)| \ge (1-\varepsilon)d|S|$. This implies by a simple counting argument that at least $(1-2\varepsilon)d|S|$ vertices in $N(S)$ have a unique neighbor in S (i.e., have degree 1 into S). Now each node in $N(S)$ that has a unique neighbor in S must correspond to an unsatisfied check. Therefore S has a lot of unsatisfied checks as neighbors which leads to the intuition that several nodes of S will flip their value. Formally, by an averaging argument, at least a fraction $(1-6\varepsilon)$ of nodes in S have at least $2d/3$ unique neighbors in $N(S)$. Therefore at least $(1-6\varepsilon)|S|$ vertices in S will correct their value.

We still have to worry about the nodes outside S flipping their value to an incorrect one. Call the set of such nodes T. Each node in T has at least $2d/3$ neighbors in $N(S)$ (since the check nodes outside $N(S)$ are all satisfied). We claim that provided $|S| \le (1-3\varepsilon)\alpha n$, $|T| < \lfloor \frac{3\varepsilon|S|}{1-3\varepsilon} \rfloor$. Indeed, otherwise consider the set $W = S \cup T'$ where T' is any subset of T of size $\lfloor \frac{3\varepsilon|S|}{1-3\varepsilon} \rfloor$. We have $|W| = \lfloor \frac{|S|}{1-3\varepsilon} \rfloor \le \alpha n$, and hence by the expansion property $|N(W)| \ge (1-\varepsilon)d|W| \ge (1-\varepsilon)d(|S|+|T'|)$. On the other hand, since each node in T' has at most $d/3$ fresh neighbors outside of $N(S)$, we have $|N(W)| \le |N(S)| + d|T'|/3 \le d|S| + d|T'|/3$. These two inequalities on $|N(W)|$ imply that $|T'| \le \frac{3\varepsilon|S|}{2-3\varepsilon}$ which contradicts the fact that $|T'| = \lfloor \frac{3\varepsilon|S|}{1-3\varepsilon} \rfloor$ (assuming, wlog, that $|S|$ is bigger than some small constant).

Hence the number of errors decreases by at least $(1-6\varepsilon)|S| - \frac{3\varepsilon|S|}{1-3\varepsilon}$, which is at least $|S|/6$ for $\varepsilon \le 1/12$. Therefore, the iterative algorithm can correct all error patterns with fewer than $\alpha(1-3\varepsilon)n$ errors, as claimed. $\square$

Recently, Capalbo et al. [8] gave an explicit construction of constant-degree lossless unbalanced bipartite $(n, m, d, \alpha, (1-\varepsilon)d)$-expanders for arbitrary $\varepsilon > 0$. Using their construction together with the above lemma gives an explicit construction of binary linear codes that can be decoded from some constant fraction of errors in linear time.[5] The construction of lossless expanders in [8] is complicated; in the next section we will present a more general way to construct codes from weaker expanders which will also lead to better parameters.

4.2 The Tanner construction

At the time of discovery of expander codes, the lossless expanders of the kind required in above construction were not explicitly known. Hence Sipser and Spielman [21] used a more general graph-based code construction scheme due to Tanner [26], defined below.

Definition 1 (Tanner Codes) *Given a bipartite graph $H = (X, Y, E)$ with n variables nodes X and m constraint nodes Y each of degree d (where we denote by $\Gamma(i,1), \Gamma(i,2), \ldots, \Gamma(i,d)$ the d neighbors of the i'th node of Y for $1 \le i \le m$), and a binary code S of block length d, the binary code*

[4] We point out that this is different from the original algorithm of Sipser and Spielman, where a value if flipped as long as it decreases the number of unsatisfied checks amongst its neighbors by at least 1. The algorithm here, which is mentioned in [8], flips only if there is an "overwhelming" reason to do so, and its analysis is simpler than that of [21], albeit it relies on a stronger expansion factor than was needed by [21].

[5] In the high rate regime, the parameters of their construction yields the following trade-off: one can correct a fraction $\delta/\log^{O(1)}(1/\delta)$ of errors with codes of rate $1-\delta$.

$C(G,S)$ consists of words $(x_1, x_2, \ldots, x_n)$ such that for each $1 \le i \le m$, $(x_{\Gamma(i,1)}, x_{\Gamma(i,2)}, \ldots, x_{\Gamma(i,d)})$ is a codeword of S.

The required unbalanced bipartite expanders H were obtained using edge-vertex incidence matrices of an expander, i.e., taking a d-regular expander $G = (V, E')$ with n *edges* and using $X = E'$ and $Y = V$ in the above construction. It is simpler and more instructive in this context to think of the codeword symbols as residing on edges of G where each node v of G imposes the constraint that the symbols residing on the edges incident to v form a codeword of S. The decoding algorithm and analysis is particularly simple when the underlying graph G itself is bipartite, as was observed in Zémor's beautiful recent paper [28]. We will therefore follow Zémor's treatment in sketching the basic idea behind the decoding.

Let A, B be the two sides of the bipartition of G. For convenience, we assume G is d-regular on both sides and $|A| = |B| = m$ with $n = md$ edges. We will require the following "pseudorandom" property from G: For every $X \subseteq A$ and every $Y \subseteq B$,

$$\left| \frac{E(X,Y)}{d|X|} - \frac{|Y|}{m} \right| \le \frac{2}{\sqrt{d}} \sqrt{\frac{|Y|}{|X|}} \ . \tag{1}$$

This property can be achieved by taking G to be the *double cover*[6] of an (m, d, λ)-expander (called Ramanujan graphs) with $\lambda \le 2\sqrt{d}$ [18]; the $2/\sqrt{d}$ factor is not important, any factor that goes to 0 as $d \to \infty$ will suffice. A proof of the above pseudorandomness property of expanders can be found in [4, Chap. 9].

The decoding algorithm for $C(H, S)$ is a natural one. The symbols on the edges of G incident to each node in $A \cup B$ must form a codeword of S. The algorithm therefore proceeds by correcting each such local projection to a codeword of S that is close to it, if such a codeword exists. To avoid conflicting decisions on an edge, this is done in tandem, with nodes in A and B alternating (this is the reason taking G to be bipartite simplifies the analysis). Let γd be the minimum distance of S. The formal decoding algorithm proceeds for $O(\log n)$ rounds, or until all errors have been corrected, whichever is earlier, where each round involves the following steps (where $x \in \{0,1\}^n$ is the current value of word of being decoded):

1. (Left wing decoding) For each $u \in A$ in parallel, check if there exists a vector $z \in \{0,1\}^d$ at distance $< \gamma d/2$ from x_{E_u} (the projection of x onto edges incident upon u), if so set x_{E_u} to z (such a z, if it exists, must be unique).

2. (Right wing decoding) Same as above, but this time for each $v \in B$ in parallel.

If the fraction of errors in the initial received word x is at most $\gamma^2/4 - \varepsilon$ (for ε small enough compared to d), it is shown in [28] that the above algorithm will correct all the errors in $O(\log n)$ rounds. Clearly each round has a linear time implementation, and with a careful examination of which subsets of nodes in A and B need to participate in each round, an $O(n)$ time implementation can also be achieved (cf. [5]).

The basic idea behind the analysis is to keep track of the sets $A_i \subseteq A$ (resp. $B_i \subseteq B$) which are the nodes in A (resp. B) which have some erroneous edge incident to it after the left wing (resp. right wing) decoding in the i'th round. For a node u to be in A_i (resp. B_i), the number of errors on the edges E_u incident upon u must be at least $\gamma d/2$ just prior to the left wing (resp. right wing)

[6]The double cover of a graph $D = (V, E)$ is a bipartite graph that is obtained by placing the vertices of D on both sides, call them left and right, and for each edge $(u, v) \in E$ connecting the left (resp. right) copy of u with the right (resp. left) copy of v.

decoding step of round i. Therefore, $d|A_1|/2$ is a lower bound on the initial number of errors, and so if we began with few errors, then $|A_1|$ must be small. Now consider the set T_i of edges of G in error just after the i'th round of decoding. By the above argument, each node in A_{i+1} must be adjacent to at least $\gamma d/2$ edges of T_i. Also, by the definition of B_i, the endpoint in B of each edge in T_i must belong to B_i. The expansion property (1) can then be used to show that in such a case, if $|B_i|$ is small, then $|A_{i+1}| \leq \rho|B_i|$ for some constant $\rho < 1$. A similar argument shows that $|B_i| \leq \rho|A_i|$ when $|A_i|$ is small. Therefore, since we begin with $|A_1|$ small, the size of the A_i's and B_i's decreases geometrically which implies that in $O(\log n)$ rounds they will be empty and all errors will be corrected.

Using codes that meet the Gilbert-Varshamov bound for the code S (which have relative distance $\gamma = \sqrt{\delta}$ and rate $1 - H(\sqrt{\delta})$), and formalizing the above ideas, yields the following result:

Theorem 3 *[21, 28] For every $\varepsilon > 0$, and for every $0 < \delta < 0.0121$, there exist an explicit family of binary linear codes of rate $1 - 2H(\sqrt{\delta})$ and relative distance $\delta - \varepsilon$ which can be decoded from a fraction $(\delta/4 - \varepsilon)$ of errors in linear time.*

A recent work by Skachek and Roth [22] improves the fraction of errors corrected in the above result to $(\delta/2 - \varepsilon)$ by incorporating Generalized Minimum Distance (GMD) style ideas from Forney's work [10] and using an error-and-erasures decoder for the code S in initial iterations. Barg and Zemor [5] improved the rate vs. distance trade-off in the above result using the idea of replicating edges of the graph G and viewing S as both a binary code as well as an additive code over $\mathbb{F}_{2^t}$ for some parameter t. This way they manage to achieve a distance $\delta = (1 - R)H^{-1}(1 - R)$ for rate R together with a linear time decoding algorithm for a fraction $\delta/4$ of errors. Very recently, Feldman and Stein [9], gave a linear programming (LP) decoder for the above codes, which also achieves the trade-off of Theorem 3 (though their decoding time is not linear).

4.3 Obtaining linear time encoding

We now briefly discuss Spielman's work [24] on getting codes that also have linear time *encoding* algorithms, in addition to linear time decoding algorithms. The construction of such codes is recursive, and involves a cascade of several expander codes. Richardson and Urbanke [20] show how certain "Tornado codes," which are LDPC codes drawn from a suitable ensemble and achieve the capacity on the erasure channel, can be encoded in linear time (with high probability over the choice of the code from the ensemble). However, for codes that handle adversarial noise, Spielman's method seems to be the only known way to get linear time encoding. It is an interesting question whether the expander codes, of say Theorem 3, have any structure that can be exploited to give a linear-time encoder for them, using say, some of the ideas in [20].

The basic entity in Spielman's construction is the notion of an "error-reduction code." These are "systematic" codes (where the codeword consists of the message bits followed by some check bits) that have a very simple encoding structure. Specifically, each output check bit is the XOR of a fixed constant number of input bits (thus, these codes are *locally encodable*). Of course this immediately implies the encoding can be performed in linear time. Also, the structure governing which check bit depends on which message bit is given by a bipartite expander as in Theorem 2. Therefore, in this construction, the expanders are used to specify the generator matrix of the code, instead of the parity check matrix as in Theorem 2.

Such locally encodable objects are too weak to serve as codes in the conventional sense, however they can still have the following useful *error-reduction* property. Given a word that differs from a codeword $\mathbf{w}$ in $t \leq \alpha n$ of the check bits, and $v \leq \alpha n$ of the message bits, a simple linear-time decoding algorithm (similar to that of Theorem 2) will output a word that differs from $\mathbf{w}$ in at most $t/2$ of the message bits. Note that when $t = 0$ and there are no errors in the check bits, then all message bits will also be corrected (in fact, this is exactly the situation of Theorem 2 where the parity checks all had to be met and so by default all check bits must be zero and hence we know them without any errors.) It is not difficult to use these error-reduction codes in a recursive scheme to construct linear-time encodable/decodable codes; we point the reader to [24, 23] for further details. We end this section with the remark that one can also get such codes with rate arbitrarily close to 1; specifically we can state the following result which we will later allude to in Section 5.2.

Theorem 4 *For every $\gamma > 0$, there is an explicit family of binary linear codes of rate $1/(1+\gamma)$ which can be encoded in linear time, and decoded from a fraction $\Omega(\gamma^2/\log^2(1/\gamma))$ of errors in linear time.*

5 Expanders to boost error-resilience

The results discussed in the previous section permit linear-time error recovery from a small fraction of errors (the fraction is about 1% for linear-time decodable codes, and only about 10^{-6} if we require linear time encoding as well). In this section, we will discuss how expanders can be used (again!) to improve the relative distance all the way up to $1/2 - \varepsilon$ together with linear time decoding up to fraction $1/4 - \varepsilon$ of errors (which is the best possible for binary codes as follows from the Plotkin bound $R(C) \le 1 - 2\delta(C)$).

5.1 The basic construct

We will now describe a very useful way in which expanders can be used to distribute the symbols of a code around in order to enhance its distance and error-correcting capabilities.

Definition 2 *[2] Given a bipartite graph $G = (X, Y, E)$ with $X = \{1, 2, \ldots, n\}$, $Y = \{1, 2, \ldots, m\}$, where each node in X (resp. Y) has degree c (resp. d), and two error-correcting codes C and C_0 with the following properties: (i) C has block length n and is defined over the alphabet $\{0,1\}^a$, and (ii) C_0 is a binary code of dimension a and block length c; the code $G(C, C_0)$, of block length m over the alphabet $\{0,1\}^d$ is defined as follows. For $1 \le i \le m$, let $\Gamma(i,1), \ldots, \Gamma(i,d)$ be the neighbors in X of node $i \in Y$ and let $f(i,j) \in \{1, 2, \ldots, c\}$ for $1 \le j \le d$ be such that $i \in Y$ is the $f(i,j)$'th neighbor of $\Gamma(i,j)$ (as per some fixed ordering of the neighbors of nodes in X, Y). Then the i'th symbol of the encoding of a message m by $G(C, C_0)$ is a d-tuple whose j'th component, $1 \le j \le d$ equals $C_0\big(C(m)_{|\Gamma(i,j)}\big)_{|f(i,j)}$.*

The formal definition might look complicated, but the idea is very simple. The message is first encoded by a code C, then each symbol of the codeword is encoded by C_0 (so far this is just encoding by the *concatenated code* of C and C_0). Then the symbols of the various codewords of C_0 are "pushed" along the edges of G to Y, where the ℓ'th symbol of the r'th encoding (for $1 \le \ell \le c$ and $1 \le r \le n$) is sent to the ℓ'th neighbor in Y of node $r \in X$. The nodes in Y then "collect" d symbols, one from each of its neighbors, and juxtaposes them together to get a symbol in $\{0,1\}^d$. Note that the rate of $G(C, C_0)$ equals the product of the rates of C and C_0, since G is only used to redistribute symbols and does not introduce any redundancy.

The above scheme was introduced in [2] and has been put to good use in [2, 14]. Alon et al. [1] had earlier introduced a special case of the above scheme with C_0 being the repetition code (that maps 0 to 0^c and 1 to 1^c). In this case, it is easily argued that if G is an (α, β)-*disperser*, namely has the property the neighborhood of every subset of X of size at least αn contains at least βm nodes of Y, and C is a binary code of relative distance α, then $G(C, C_0)$ has relative distance at least β and rate $R(C)/c$. Since there are explicit $(0.1, 1-\varepsilon)$-dispersers of degree $c = O(1/\varepsilon)$ (constructed for example using Ramanujan expanders), using this approach [1] gave explicit constructions of codes of relative distance $(1-\varepsilon)$ (for $\varepsilon > 0$ as small as we like) and rate $\Omega(\varepsilon)$ over an alphabet of size $2^{O(1/\varepsilon)}$ (using for C any explicit asymptotically good binary code of relative distance 0.1). (Algebraic-geometric codes achieve better parameters than this construction, but their construction is complicated.) Note that by the Singleton bound, this is off from the optimal rate by only a constant factor. Thus, with a very simple construction, expanders enable to amplify the distance at the expense of worsening the rate and alphabet size.

Definition 3 *The special case when C_0 is a repetition code is itself quite useful in various contexts, so we will use the notation $G(C)$ in such cases, with the understanding that if G has left degree c and C is defined over alphabet Σ, then $G(C)$ stands for $G(C, C_0)$ where C_0 is a rate $1/c$ repetition code over Σ.*

5.2 Near-optimal linear time codes

We now discuss how the above scheme can be used to achieve a near-optimal trade-off between rate and relative distance, together with linear time encoding/decoding algorithms.

Theorem 5 *[14] For every r, $0 < r < 1$, and all $\varepsilon > 0$, there is an explicit family of codes of rate r and relative distance at least $(1 - r - \varepsilon)$ such that codes in the family can be encoded as well as unique decoded from a fraction $(1 - r - \varepsilon)/2$ of errors in time linear in the block length.*

Proof Idea: We will use the scheme from Definition 2 with suitable components to obtain the result. Let $\gamma = \varepsilon/4$. The code C will be drawn from the family guaranteed by Theorem 4 and will have rate $1/(1+\gamma)$, block length n (say), and can be decoded from a fraction $\beta > \gamma^3$ of errors in linear time. We will pick $G = (X, Y, E)$ to be a d-regular bipartite graph with n nodes on each side with pseudorandom Property (1) with $d = \Theta(1/\beta\varepsilon^2)$. For such choice of d, it can be checked that (1) implies that for every $R \subseteq Y$ with $|R| \geq \alpha n$, the set of nodes $L \subseteq X$ which have at least $(\alpha - \varepsilon/4)d$ of their neighbors into R satisfies $|L| \geq (1-\beta)n$. The code C_0 will be a constant-sized Reed-Solomon code of block length d and rate $r(1+\gamma)$. To construct $G(C, C_0)$, we will block together symbols of the codeword of C into blocks of suitable size and then encode each such block by C_0. The overall rate of $G(C, C_0)$ will be the product of the rates of C and C_0 and thus equal r. It can clearly be encoded in linear time since C can be.

We now sketch the fairly natural approach to decoding $G(C, C_0)$. Given a received word r with at most $(1 - r - \varepsilon)n/2$ errors, in the first step we "invert" the encoding process via G (since G just permutes symbols) to obtain candidate received words for each block of encoding by C_0. Each of these will be decoded to their closest Reed-Solomon codeword (even by brute-force search this takes only constant time per block) to give a received word for C. Finally this word is decoded up to a radius of βn using the linear time decoding algorithm for C. To prove correctness, note that by the above stated property of G, at least $(1-\beta)n$ of the received words for C_0 will have at most $\left(\frac{1-r}{2} - \frac{\varepsilon}{4}\right)d$ errors, and these will be corrected by the Reed-Solomon decoding step. Therefore, in the final step at most βn errors will remain, which can be handled by the decoding algorithm for C. The overall complexity is clearly linear since the decoding of C can be performed in linear time and the prior steps can be performed in constant time per decoded symbol. $\square$

Comments: The near-optimal trade-off (rate r for distance close to $(1-r)$) that almost matches the optimal Singleton bound comes from using the Reed-Solomon codes. The overall scheme can be viewed as using several independent constant-sized RS codes; the role of the expander is then to "spread out" the errors among the different copies of the RS code, so that most of these copies can be decoded, and the remaining small number of errors can be corrected by the left code C. Since only a small fraction of errors needs to be corrected by C it can have rate close to 1 and there is not much overhead in rate on top of the Reed-Solomon code.

The above codes are defined over a large alphabet (whose size depends exponentially on $1/\varepsilon$). But they can be concatenated with constant-sized binary codes that lie on the Gilbert-Varshamov bound to give constructions of binary codes which meet the so-called Zyablov bound. In particular, we can have linear-time binary codes of rate $\Omega(\varepsilon^3)$ to correct a fraction $(1/4 - \varepsilon)$ of errors for arbitrary $\varepsilon > 0$. We point the reader to [14, 13] for further details.

6 Linear time list decoding

Under the adversarial error model, with the restriction that the decoder output a unique codeword, the maximum fraction of correctable errors is half the relative distance. In particular, this limits the maximum tolerable noise fraction to 50%. A notion called *List Decoding* has been proposed and studied in the literature to deal with error fractions bigger than 1/2. In fact, one can perform meaningful recovery using list decoding even when the fraction of errors approaches 1! We refer the reader to the survey by Sudan [25] for detailed information on list decoding, and here focus only on a specific construction of list-decodable codes using expanders.

A code C of block length n is said to be (ρ, L)-list decodable if every Hamming ball of radius ρn contains at most L codewords of C. Below we will sketch the idea behind the work of Guruswami and Indyk [15] who prove the following result.

Theorem 6 *For every $\varepsilon > 0$, there exists an explicit family of $(1-\varepsilon, O(1/\varepsilon))$-list decodable codes of positive rate $R(\varepsilon) > 0$ which can be encoded as well as list decoded from a fraction $(1-\varepsilon)$ of errors in linear time.*[7]

Expanders play a prominent role throughout the above construction and proof. The notion of *list recoverable* codes also plays a pivotal role. A code C of block length n over alphabet Σ is said to be (α, ℓ, L)-*list recoverable* if for every n-tuple of sets $(S_1, \ldots, S_n)$ with each $S_i \subset \Sigma$, $|S_i| \le \ell$, the number of codewords $\mathbf{c} = \langle c_1, \ldots, c_n \rangle \in C$ with the property that $c_i \in S_i$ for at least αn values of i is at most L. Such a code is said to be (α, ℓ, L)-list recoverable in time $T(n)$ if in addition there is a time $T(n)$ algorithm that produces the list of up to L such codewords given the input sets S_i. The problem of constructing such list-decodable codes is first reduced to the problem of constructing linear-time $(1-\gamma, \ell, \ell)$-list recoverable codes for every integer $\ell \ge 1$, for some $\gamma = \gamma_\ell > 0$. (Roughly, such codes imply $(1 - 2/\ell, \ell)$-list decodable codes.) This reduction is achieved by using the $G(C)$ from Definition 3 for a suitable expander G, and with C being the list recoverable code. The details are quite simple and may be found in [15, Lemma 1].

The construction $(1 - \gamma_\ell, \ell, \ell)$-list recoverable codes proceeds recursively. For the recursion, it is convenient to construct a slightly stronger object; for a fixed small α, we will construct $(1-\gamma_\ell, \alpha, \ell, \ell)$-list recoverable codes which will have at most ℓ codewords "consistent" with $(1-\gamma_\ell)$ of the input sets S_i, even if a fraction α of the sets S_i are not constrained by $|S_i| \le \ell$ and are allowed to be the entire alphabet (these positions correspond to "erasures" where we get no useful information about the codeword symbol). Suppose we already have a code $C_{\ell-1}$ of block length n which is $(1-\gamma_{\ell-1}, \alpha, \ell-1, \ell-1)$-list recoverable. A code C_ℓ which is $(1-\gamma_\ell, \alpha, \ell, \ell)$-list recoverable is constructed as $C_\ell = G_2(G_1(C_{\ell-1}))$ where both G_1, G_2 are $n \times n$ bipartite constant-degree expanders

[7]The dependence of the rate on ε is rather poor—it is $R(\varepsilon) = 2^{-2^{O(1/\varepsilon)}}$.

(with degrees d_1, d_2 say) with appropriate properties. Let $G_1 = (X, Y, E_1)$ and $G_2 = (Y, Z, E_2)$, so that we identify the left side of G_2 with the right side of the bipartition of G_1 in a natural way. Clearly, linear time encodability as well as positive rate is maintained in this process. The alphabet of C_ℓ is $\Sigma_\ell = \Sigma_{\ell-1}^{d_1 d_2}$.

The list recovering of C_ℓ proceeds in two steps/cases as described below. Note that the input is a collection of sets $S_1, S_2, \ldots, S_n$ with each $S_i \subseteq \Sigma_\ell$ where all but αn of them have at most ℓ elements. In the first step, each $i \in Y$ collects a set $L_2(i,j) \subseteq \Sigma_{\ell-1}^{d_1}$ of at most ℓ possible symbols from the set S_j for each of its d_2 neighbors, and then computes $K_i = \cap_{(i,j)\in E_2} L_2(i,j)$. Since the fraction of "errors" ε_ℓ is very small, for each candidate codeword that has to be output, most of the K_i's will have the correct symbol. Clearly $|K_i| \le \ell$ for all i. Suppose that in fact at least αn of the K_i's satisfy $|K_i| \le \ell - 1$. In this case intuitively we have made progress since the amount of "ambiguity" in the those symbols has gone down to $\ell - 1$ from ℓ. However, this happens only for a small fraction α of symbols, but this can be transferred to a large (say, $1-\alpha$) fraction of symbols of $C_{\ell-1}$ using the expander G_1 (the specific property needed for G_1 will thus be that any α fraction of nodes in Y are adjacent to at least a $(1-\alpha)$ fraction of nodes in X). We are now in a position to apply the list recovering algorithm for $C_{\ell-1}$ (since there are at most a fraction α of "erased" positions) to finish the decoding.

The more difficult and interesting case is when $|K_i| < \ell$ for less than a fraction α of the nodes $i \in Y$. This means that for most nodes $i \in Y$, all the sets $L_2(i,j)$ are in fact equal to K_i (and contain ℓ distinct elements). One can then define a certain kind "consistency graph" $\tilde{H} = (X \times \{1, 2, \ldots, \ell\}, Y \times \{1, 2, \ldots, \ell\}, \tilde{E})$ whose vertex set corresponds to the ℓ possible choices for the correct symbol at each node of X, Y and the edge set is defined based on which symbol of $L_2(i,j)$ matches the r'th symbol of K_i for $r = 1, 2, \ldots, \ell$ (as per some fixed ordering of the elements in each K_i and $L_2(i,j)$). The problem of finding a consistent codeword can then be cast as finding a very dense subgraph (which internally is closely resembles a copy of the expander G_2) with few outgoing edges in the graph G. Using the fact that this subgraph resembles G_2 and that G_2 is a high quality Ramanujan expander, the authors of [15] demonstrate how spectral techniques can be used to find such subgraphs and thus all the solution codewords in $O(n \log n)$ time (and this suffices to get a linear time decoding algorithm in the RAM model using codes over a growing alphabet). The detailed formalism of the above vague description as well as the technical details of the spectral partitioning are quite complicated, so we will refrain from trying to elaborate further and instead point the interested reader to the paper [15].

We note that since ℓ reduces by 1 at each step, and there is at least a constant factor loss in rate at each recursive step, the best rate one can hope for using the above approach is $2^{-O(\ell)}$. However, due to the intricacies of the spectral partitioning step and the resulting dependencies amongst the various parameters, the rate achieved is in fact worse and is only $2^{-2^{O(\ell)}}$. For the case $\varepsilon_\ell = 0$ or "error-free" list recovering where the goal is to only output codewords $\mathbf{c}$ that satisfy $c_i \in S_i$ for *every* i, Guruswami and Indyk [16] present a more efficient recursive construction of linear-time codes that achieve a rate of $1/\ell^{O(1)}$ (this is accomplished by reducing ℓ by a constant multiplicative factor as opposed to the above additive amount at each step). They also manage to achieve a rate of $\Omega(1/\ell)$ which matches the best known construction with polynomial time algorithms for an even simpler model, namely that of recovering a garbled mixture of codewords. These results hopefully mark the first (small) steps in the program of using expander-based or related combinatorial ideas to construct linear time list-decodable codes that come close or even match the performance of the best known constructions with polynomial time decoding algorithms.

We conclude by mentioning a recent result due to the author [12] which uses certain extractors as the list-recoverable code in the above general scheme to construct explicit list-decodable codes of close-to-optimal rate to correct a fraction $(1-\varepsilon)$ of errors in sub-exponential time (specifically, the achieved rate is $\varepsilon / \log^{O(1)}(1/\varepsilon)$, with the best possible rate that one could hope for being $O(\varepsilon)$). This result is the first to beat the quadratic (rate ε^2) barrier for list decoding from a fraction $(1-\varepsilon)$ of errors (Reed-Solomon codes achieve a rate ε^2 [17] with polynomial time list decoding). This work shows that expander-based constructions are useful not only to achieve faster algorithms, but hold the promise of sometimes obtaining better rate or trade-offs than other constructions.

7 Probabilistic Noise Models

We have so far discussed results for adversarial noise assuming only a bound on the total number of errors effected by the channel. Here the hallmark of expander codes was that we could achieve what was earlier possible with polynomial time algorithms with linear time encoding/decoding algorithms. Expander codes also provide good constructions (with good performance *and* linear time algorithms) of codes for probabilistic channels like the binary symmetric channel BSC_p. We briefly mention some of the results in this direction.

A decoding algorithm that corrects a fraction $p+\varepsilon$ of adversarial errors also achieves, by Chernoff bounds, exponentially small error probability on the BSC_p. But the adversarial setting is too general and this reduction fails to achieve the capacity of BSC_p, i.e., does not permit communication at rates close to $1-H(p)$. However, using slight variants of the decoding algorithms of Section 4.2, it is possible to not only achieve capacity, but also a very good error exponent [5, 6, 7].

By an inspection of the argument in Section 4.2, we recall that the decoding succeeds whenever $|A_1|$ is smaller than some bound, say s, i.e., whenever the very first round of left wing decoding corrects the errors on all the edges incident on most vertices of A. Since S is a code of constant size, we can replace the first left wing decoding step by a maximum likelihood decoding of S. Picking a good S (which in particular performs at least as well as a random code) leads to an exponentially small probability that $|A_1| > s$, and this same probability serves as an upper bound on the probability of incorrect or unsuccessful decoding. This was analyzed in [5] and shown to already suffice to achieve capacity. This was the first such result for low-density parity check codes and simple iterative decoding procedures. By using the idea of replicating edges of the expander and using the properties of S viewed as a 2^t-ary code, they improved the error exponent to $f_2(R,p) = \max_{R \le R_0 < 1-H(p)} E(R_0,p)(R_0-R)/2-\varepsilon$ (i.e., they gave a linear time algorithm with an error probability of $2^{-f_2(R,p)N}$ for codes of rate R where N is the block length). We note that $f_2(R,p) > 0$ for all $R < C$ where $C = 1-H(p)$ is the capacity of BSC_p. In follow-up work [7], the behavior of the error exponent was further analyzed for code rates approaching the capacity. The improved estimates obtained therein relied on non-trivial modifications of the basic expander decoding scheme.

The best known error exponent for explicit constructions has long been achieved by Forney's concatenated codes coupled with GMD decoding [10] (the decoding time of this scheme can be made near-linear). In particular, the above referred to exponent $f_2(R,p)$ is worse than the Forney exponent. Since the work of Guruswami and Indyk [14] gives codes almost matching the Singleton bound with a linear-time *errors-and-erasures* decoding algorithm, their result also implies linear-time codes that attain the Forney error exponent; the journal version [13] has more details on this. Recently, Barg and Zémor [6] showed how to attain the Forney exponent using expander codes and simple iterative decoding by modifications to the basic construction in Section 4.2 together with the use of "soft decision decoding" in initial iterations of the iterative decoding algorithm.

Feldman and Stein [9] have given a linear programming (LP) based approach to decoding expander codes and they prove that it achieves a positive error exponent for all rates less than the capacity. The LP decoder is not as simple as the iterative decoding algorithms (in particular, the decoding complexity is super-linear in the block length) and the proven error exponent is worse than those shown in above works for iterative decoding. However, their work could lead to further insights in decoding expander codes (or related LDPC codes) and it also possesses an elegant "ML certificate" property—if the decoder outputs a codeword, then that *must be* the maximum likelihood answer and thus output codewords come with a proof of optimality.

8 Concluding thoughts

Expander-based code constructions have by now matched the performance of previously known algebraic counterparts in settings that include unique decoding under adversarial errors and decoding under probabilistic channels like the binary symmetric channel. In addition, these codes come equipped with encoding and decoding algorithms that run in linear time (and they are so far the only such codes known). Expander-based constructions, being combinatorial, are more "hands-on," and this enables the design and analysis of simple and fast iterative decoding algorithms. In our opinion, the potential of expanders and related pseudorandom objects in coding theory is yet to be fully tapped. In particular, constructing simple combinatorial constructions of list-decodable codes that better the trade-offs achieved in [15] is a good, concrete challenge.

Acknowledgments

I thank Madhu Sudan for all the coding-theoretic expertise and perspective that he has kindly shared with me over the past several years. I'm most grateful to Piotr Indyk for the fruitful collaborations on expander-based code constructions (many of which were mentioned in this survey). I thank Salil Vadhan for help with the proof of Theorem 2 on simple linear-time decodable codes from lossless expanders.

References

[1] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

[2] N. Alon, J. Edmonds, and M. Luby. Linear time erasure codes with nearly optimal recovery. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519, 1995.

[3] N. Alon and Y. Roichman. Random Cayley graphs and expanders. *Random Structures and Algorithms*, 5:271–284, 1994.

[4] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley and Sons, Inc., 1992.

[5] A. Barg and G. Zémor. Error exponents of expander codes. *IEEE Transactions on Information Theory*, 48(6):1725–1729, June 2002.

[6] A. Barg and G. Zémor. Concatenated codes: serial and parallel. *Manuscript*, 2003.

[7] A. Barg and G. Zémor. Error exponents of expander codes under linear-complexity decoding. *SIAM J. Disc. Math.*, 17(3):426–445, January 2004.

[8] M. R. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 659–668, 2002.

[9] J. Feldman and C. Stein. LP decoding achieves capacity. *Manuscript*, 2004.

[10] G. D. Forney. *Concatenated Codes.* MIT Press, Cambridge, MA, 1966.

[11] R. G. Gallager. *Low-density parity-check codes.* MIT Press, Cambridge, MA, 1963.

[12] V. Guruswami. Better Extractors for Better Codes? In *Proceedings of 36th Annual ACM Symposium on Theory of Computing (STOC)*, June 2004.

[13] V. Guruswami and P. Indyk. Linear time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory.* To appear.

[14] V. Guruswami and P. Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 812–821, 2002.

[15] V. Guruswami and P. Indyk. Linear-time encodable and list decodable codes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 126–135, June 2003.

[16] V. Guruswami and P. Indyk. Linear-time list decoding in error-free settings. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, July 2004.

[17] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[18] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[19] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155:157–187, 2002.

[20] T. Richardson and R. Urbanke. Efficient encoding of low density parity check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, Feb 2001.

[21] M. Sipser and D. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[22] V. Skachek and R. Roth. Generalized Minimum Distance iterative decoding of expander codes. In *Proceedings of IEEE Information Theory Workshop (ITW)*, pages 245–248, 2003.

[23] D. Spielman. Constructing error-correcting codes from expander graphs. In *Emerging Applications of Number Theory, IMA volumes in mathematics and its applications*, volume 109, 1996.

[24] D. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996.

[25] M. Sudan. List decoding: algorithms and applications. *SIGACT News*, 31(1):16–27, March 2000.

[26] M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[27] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 2004. To appear.

[28] G. Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.