Electronic Colloquium on Computational Complexity, Report No. 24 (2002)

ECCC

List-decoding in Linear Time

Piotr Indyk

April 24, 2002

Abstract

Spielman [S96] showed that one can construct error-correcting codes capable of correcting a constant fraction $\delta << 1/2$ of errors, and that are encodable/decodable in linear time. Guruswami and Sudan [S97, GS99] showed that it is possible to correct more than 50% of errors (and thus exceed the "half of the minimum distance barrier"), with polynomial-time encoding and decoding. In this paper we show that it is possible to achieve these two properties *simultanously*. Specifically, we give a construction of constant rate, linear-time encodable codes that are capable of correcting any $\delta < 2/3$ fraction of errors in linear time. Unlike in [S97, GS99], our codes are constructed using high-quality expander graphs, along the lines of [ABN$^+$92, GI01, GI02]).

1 Introduction

Constructing efficiently encodable and decodable codes with high error-correction capability is the central problem in coding theory, and is of tremendous practical importance as well. Since the concept of error-correcting codes was conceived in 40's, there has been an amazing progress towards this goal. In particular, it is now known how to construct efficient codes which can correct a constant fraction of errors, and which can be decoded an encoded in linear time [S96]. In fact, this goal can be achieved even for codes with minimum distance arbitrarily close to the length n of the code [GI01, GI02]; this allows one to identify the original message uniquely, even when the fraction of errors is arbitrarily close to 50%. Note that if the fraction of errors exceeds 50%, unique decoding of the corrupted codeword is impossible.

In a recent breakthrough, it was shown [ALRS92, S97, GS99] that one can exceed the 50% barrier (and, in fact, correct a fraction of errors arbitrarily close to 100%), if one allows to output a *list* of potential codewords. This approach (called *list-decoding*) has been introduced in the late 50's by Elias [El57] and Wozencraft [Woz58]. Until recently, however, no efficient algorithm for list-decoding was known. In [S97, GS99] the authors presented the first *polynomial time* algorithms for decoding Reed-Solomon codes from any fraction of errors smaller than 1. Their algorithms have been subsequently improved (e.g., in [RR00], see [G01a] for a detailed survey) leading to near-quadratic time decoding algorithms.

The improvements in the efficiency of algorithms for list-decodable codes suggested that it might be possible to repeat the progress of unique decoding algorithms and construct

ISSN 1433-8092

linear time encodable and *list*-decodable codes. Unfortunately, this does not seem possible (or at least seems very hard) using current list-decoding techniques. This is due to the fact that all codes for which polynomial time list-decoding algorithms were known so far (i.e., Reed-Solomon codes and their Algebraic Geometry-based generalizations) are algebraic in nature[1]. Even the algorithm for *unique* decoding of Reed-Solomon codes has running time $\Theta(n \log^2 n)$.[2] At the same time, the codes of [S96] cannot be easily modified to correct more than $n/2$ errors, since their minimum distance is much smaller than this bound.

In this paper we depart from the approach used in the aforementioned papers. Instead, we show how to list-decode codes constructed using *expanders* [ABN$^+$92, GI01]. Our main result is a construction of constant rate, linear-time encodable codes that are capable of correcting up to $\delta < 2/3$ fraction of errors in linear time. The construction of the code is combinatorial and non-algebraic (given the expander).

To describe our results in more detail, we need to define a generalization of list-decodability called *list-recoverability*. We say that a code is (α, l, L)-recoverable, if we can decode the corrupted codeword given n lists $L_1 \ldots L_n$ of size l, such that at least α fraction of the lists contain the correct codeword symbol; the decoding procedure is allowed to output L candidate codewords. See Preliminaries for the formal definition. Here we mention that the case of $l = 1$ corresponds to list-decodability. List-recoverability in the general setting has been investigated before, e.g., in [ALRS92, BN00, TZ01, GI01].

Our first construction gives a code with constant rate and alphabet, which is encodable and $(1, l, L)$-recoverable in linear time, for $l, L = O(1)$. The code is constructed using high quality expander graphs (similarly as in [ABN$^+$92, GI01]). The decoding procedure uses the results of [AC88] on fault-tolerant properties of expanders.

Although the aforementioned code can recover a codeword from lists of arbitrary (constant) size, the fact that *all* lists contain a correct symbol is crucial for the decoding algorithm. Our second result overcomes this difficulty. In particular, we give a construction of a code which can be encoded in linear time, and is $(1 - \epsilon, 2, 2)$-recoverable for some small $\epsilon > 0$. Given the input lists, the codewords can be recovered in near-quadratic time. The recovering algorithm uses the Semidefinite Programming-based algorithm for satisfying a set of linear constraints modulo 2 [GW95, Zwick98], and in particular its near-quadratic implementation of [KH96]. The constructed code has constant rate and alphabet size.

In the next step, we show how to reduce the time needed to decode the above code. This is done by via a novel reduction of the decoding problem to the problem of finding a sparse-cut in a certain graph of linear size. We show that the latter problem can be solved using *spectral partitioning*, which results in decoding time of $O(n \log n)$.

Finally, we show that a concatenation of the latter code with a carefully chosen inner code results in a code which can be encoded *and* list-decoded (from up to 2/3 fraction of errors) in $O(n)$ time. The code has constant rate and alphabet size.

[1] This holds even for the "extractor codes" of [TZ01], since the algorithms for list decoding of those codes use the algorithms of [S97, GS99] as a black-box. The same comment applies to the codes of [GI01].

[2] Here and in the remainder of this paper n is used to denote code blocklength.

2 Preliminaries

In this section we define formally the concepts and notation used later on.

Model of computation. The model of computation used in this paper is the *unit cost* RAM. In this model, we allow the algorithm to perform any "reasonable" operation, on $O(\log n)$-bit long words, in unit time. All such operations used in this paper are computable by constant-depth circuits of size polynomial in $\log n$. In addition, we allow the algorithm to use indirect addressing (i.e., table lookup); we charge the algorithm for any preprocessing cost needed to compute the lookup table.

The unit-cost RAM is a standard and widely used model of computation. However, other (more restrictive) models of computation are known. In particular, the results of [S96] hold even in so-called *logarithmic-cost* RAM, i.e., where the cost of an operation is linear in the bit-length of its arguments. Our algorithms do not achieve $O(n)$ time in the latter model, although it is plausible that they can be modified to achieve such a bound.

Notation. For any alphabet set Σ, and any two vectors $x, y \in \Sigma^n$, we use $D(x, y)$ to denote the Hamming distance between x and y (i.e., the number of positions on which x and y differ).

For a sequence $\mathcal{L}$ of lists $L_1 \ldots L_n$ of symbols, and a vector x of n symbols, define $COMP(x, \mathcal{L})$ to be the number of indices j such that $x_j \in L_j$.

A code C of block-length n over alphabet Σ is (α, l, L)-*recoverable*, if for any sequence $\mathcal{L}$ of n lists $L_1 \ldots L_n$, where $L_j \subset \Sigma$, $|L_j| \leq l$, there are at most L codewords $c \in C$ such that $COMP(c, \mathcal{L}) \geq \alpha n$. We say that C is (α, l, L)-*recoverable* in time $T(n)$, if there is a procedure which finds such $\leq L$ codewords in time $T(n)$ given the lists $L_1, \ldots, L_n$.

A code C of block-length n over alphabet Σ is (α, L)-*decodable*, if for any vector $x \in \Sigma^n$ there are at most L codewords $c \in C$ such that $D(c, x) \leq (1 - \alpha)n$. We say that C is (α, L)-*decodable* in time $T(n)$, if there is a procedure which finds such $\leq L$ codewords in time $T(n)$ given x. In the context of a fixed codeword $c \in C$, we say that the list L_j (or the j-th position) is *corrupted*, if $c_j \notin L_j$. Otherwise, we say L_j (or the j-th position) is *correct*.

We will use $|x|_p$ to denote the l_p norm of a vector x. Also, for a square matrix A, we use $|A|_p$ to denote the l_p norm of A, i.e., $\max_{|x|_p=1} |Ax|_p$.

3 A $(1, l, L)$-recoverable code with linear time encoding/decoding algorithms

In this section we show how to construct codes C' which are $(1, l, L)$-recoverable in linear time for any $l > 0$. The codes have blocklength n, constant rate $r > 0$ and finite alphabet q.

Construction: For the purpose of the construction we need:

1. A code C of block-length n, constant rate $r_1 > 0$ and constant alphabet size q_1. The codes should be encodable/decodable in linear time from $(1 - \epsilon)$ fraction of erasures for a parameter $\epsilon > 0$. Such codes (with $r_1 = \Omega(\epsilon)$) can be explicitly constructed (e.g., as in [AEL95]).

2. A bipartite graph $G = (A, B, E)$ of degree d, $|A| = |B| = n$, with the following property:

   > If we delete $(1-\delta)nd$ edges of G for any $\delta > 0$, then the resulting graph contains a connected component containing at least ϵn vertices of A

The latter graph G can be constructed from a *Ramanujan graph* [LPS88]. Specifically, let $G_R = (V, E_R)$ be a graph with n nodes, degree $d = O(1/\delta^4)$, and second eigenvalue $O(\sqrt{d})$. We convert G_R into a bipartite graph such that $A = B = V$ and $E = \{(i,j),(j,i) : \{i,j\} \in E_R\}$. The square G^2 of the graph G has degree d^2. Consider a subgraph G_A^2 of G^2 induced by A. The graph G_A^2 is also a Ramanujan graph, i.e., its second largest eigenvalue is $O(\sqrt{d^2})$. Moreover, if we remove $(1-\delta)$ fraction of edges from G (forming G'), then the graph $G_A'^2$ contains $\Omega(\delta^2)$ fraction of edges of G_A^2. The result of [AC88] states that any Ramanujan graph of degree $O(1/\eta^2)$ contains a connected component of size $\Omega(\eta n)$ even if we remove $(1-\eta)$ fraction of its edges. Thus, $G_A'^2$ contains a connected component of size $\Omega(\delta^2 n)$, and we can set $\epsilon = \Omega(\delta^2)$.

Given the two building blocks C and G, we construct the code C' in the following way. For any $x \in \{0 \ldots q_1 - 1\}^n$, define $G(x)$ to be a vector $y \in (\{0 \ldots q_1 - 1\}^d)^n$ created as follows. For $j \in B$ let $\Gamma_k(j)$ be the k-th neighbor of j in A, for $k = 1 \ldots d$. The j-th symbol y_j of y is defined as $< x_{\Gamma_1(j)}, \ldots, x_{\Gamma_d(j)} >$. In other words, we "send" a copy of each symbol x_i along all edges outgoing of the vertex i, and the symbol y_j is obtained by concatenating all symbols received by j.

The code C' is now defined as a set of all vectors $G(c)$ for $c \in C$.

Encoding: It is immediate that the code can be encoded in linear time.

Decoding: It remains to show the decoding procedure. For this purpose, assume that L_j is an ordered sequence of symbols (as opposed to a set). Let $L_1 \ldots L_n$ be a message, and let $c' \in C'$ be the codeword such that $c'_j \in L_j$ for $j = 1 \ldots n$. Let c be a codeword in C corresponding to c'. We will show how to recover c in linear time.

First we start with some notation. For every edge $(i,j) \in E$, define $L(i,j)$ to be the concatenation, over all $a =< a_1, \ldots, a_d > \in L_j$, of symbols a_k s.t. $\Gamma_k(j) = i$. Intuitively, $L(i,j)$ is an ordered list of symbols that L_j "suggests" as possible candidates for c_i. Define $[a = b]$ to be equal to 1 if $a = b$ and to be equal to 0 otherwise. Let $L(i,j) =< b_1, \ldots, b_l >$. Define $sgn(i,j) =< [b_1 = c_i], \ldots, [b_l = c_i] >$ (note that sgn is defined with respect to a fixed codeword c). Let s be a $sgn(i,j)$ vector which occurs most often among all $(i,j) \in E$. Let E_s be the set of all $(i,j) \in E$ such that $sgn(i,j) = s$. Clearly $|E_s| \geq |E|/2^l$. Let $\delta = 1/2^l$. Consider the graph $G_s = (A, B, E_s)$. By the properties of G there exists $A' \subset A$ of size $\Omega(n\delta^2)$ which is connected in G_s. Let G'_s denote the connected subgraph of G_s induced by A'.

The following decoding algorithm uses a superset of the graph G'_s (note that the graph is unknown to the algorithm) to recover a large fraction of the sumbols of the codeword c.

Decoding algorithm:

1. Initialize all n entries in a vector x to "empty"

2. Guess $i \in A'$ (this can be done with $\geq 1/2^l$ probability of success, by choosing a random $i \in A$)

3. Guess the value of c_i (out of l candidates in $L(i,j)$ for any j. This can be done with the probability $1/l$ of success). Set $x_i = c_i$.

4. Repeat the following steps whenever applicable:

   (a) If a value x_i is fixed, then for all $(i,j) \in E$ remove from L_j all symbols which are not compatible with x_i, i.e., in which the symbol corresponding to x_i is different from x_i

   (b) If, with respect to *current* lists L_j, we have $L(i,j) =< b, \ldots, b >$ (i.e., all symbols in $L(i,j)$ are equal), then then set x_i to b

5. Decode c from x, using the decoding algorithm for C

The correctness of the algorithm now follows from the following claims, which hold assuming that the initial guesses of the algorithm were correct.

Claim 1 *All symbols in x set by the algorithm are correct (i.e., $x_i = c_i$ unless x_i is empty).*

Claim 2 *All c_i for $i \in A'$ are set to some value by the algorithm.*

Proof: Let $i \in A'$ such that $x_i = c_i$ has been already set. Consider any neighbor $j \in B$ of i in G'_s, and any neighbor $i' \in A$ of j. Consider symbols $a =< a_1, \ldots, a_l >\in L_j$. Our algorithm removes all symbols a from L_j which are incompatible with c_i, i.e., such that $a_k \neq c_i$ where $\Gamma_k(j) = i$. After the removal, we have $L(i,j) =< c_i, \ldots, c_i >$. However, we known that $sgn(i,j) = sgn(i',j)$. This implies that $L(i',j) =< c_{i'}, \ldots, c_{i'} >$. Thus the algorithm will at some point set $x_{i'}$ to $c_{i'}$. Since the set A' is connected, the claim follows. $\Box$.

The above arguments show that the decoding procedure outputs a fixed codeword compatible with the input lists with probablity $1/2^{O(l)}$. It remains to show that the number of such codewords is bounded. To this end, we observe that since the code C has fractional minimum distance $1-\Theta(\delta^2)$, and the graph G is regular, the code C' has also fractional minimum distance $1-\Theta(\delta^2)$ (in fact the distance is even larger, due to the expanding properties of G). From this property we conclude that the code is $(1, l, O(l))$-recoverable, by employing the following Lemma (which is a yet another variant of the Johnson bound, but which can proved using a very simple argument).

Lemma 1 *Let C be a code with minimum distance $1-\epsilon$, where $\epsilon = 1/bl$. Then for any constant B satisfying $B(1-B/b) > 1$, the code C is $(1, l, Bl-1)$-recoverable.*

Proof: Let $L = Bl$. Assume to the contrary that there are L codewords $c^1, \ldots, c^L \in C$, and lists $\mathcal{L} = L_1, \ldots L_n$ of length l such that $COMP(c^i, \mathcal{L}) = n$ for all $i = 1 \ldots L$. Define $W(t)$, $t = 1 \ldots n$, to be the cardinality of the set $\{c_t^i : i = 1 \ldots L\}$. If we choose t uniformly at random, then

$$E_t[W(t)] \geq L(1 - L/bl)$$

Observe that if $E((W(t)) > l$, then there is a position t such that $W(t) > l$ and thus all codewords $c^1, \ldots c^L$ cannot be compatible with $\mathcal{L}$, which leads to a contradiction. To ensure this, it suffices that $L(1 - L/bl) > l$, which is equivalent to the condition in the statement of the lemma. □.

Theorem 1 *For any $l > 0$, and any $n > 0$, there is a code with blocklength n and finite alphabet of size $2^{2^{O(l)}}$, with rate $2^{-O(l)}$. The code is encodable and $(1, l, O(l))$-recoverable in $n2^{O(l)}$ randomized time, with constant probability of success.*

Proof: Let b and B be constants satisfying conditions of Lemma 1. We can adjust the constants defining δ so that C' has minimum fractional distance $1 - 1/bl$ (note that this distance is $1 - 2^{-\Theta(l)}$. By aforementioned lemma it follows that the code is $(1, l, Bl - 1)$-recoverable.

In order to show the algorithmic part, observe that by the preceeding discussion we have an algorithm which for any fixed codeword $c' \in C'$ compatible with $\mathcal{L}$ outputs c' with probability $2^{-O(l)}$. Thus, after $2^{O(l)}$ iterations, the algorithm outputs all $O(l)$ compatible codewords with (arbitrarily large) constant probability. Thus the running time of the decoding algorithm is as stated. □.

4 A $(1 - \epsilon, 2, 2)$-recoverable code with linear encoding and quadratic decoding algorithm

In this section present a $(1 - \epsilon, 2, L)$-recoverable code (for certain $\epsilon > 0$), constructed using a method similar to the one described in the previous section, with linear-time encoding and roughly quadratic-time decoding algorithms. The code has finite rate and alphabet.

Construction: In order to construct the code, we need two components:

- A linear-time encodable (and quadratic-time decodable) code C with blocklength n, finite rate and alphabet q, which can decode from up to $\mu = (1 - \gamma)/2$ fraction of errors for $\gamma > 0$ arbitrarily close to 0 (and defined later). Such code, with linear time encoding and decoding, is provided e.g., in [GI01].
- A bipartite graph $G = (A, B, E)$, $|A| = |B| = n$ such that
  1. For any $X \subset A$, $|X| = \mu$, the number of neighbors of X in B is at least $(1 - \eta)|B|$ for small $\eta > 0$ (η is defined later)
  2. For any $Y \subset B$, $|Y| \leq \epsilon|B|$, the fraction of nodes $i \in A$ with at least $1/3$ fraction of neighbors in Y is at most ϵ

As before, the graph G can be constructed from Ramanujan graphs with large enough degree d. In particular, it is well-known that for bipartite graphs $G = (A, B, E)$ constructed from such graphs, the following *isoperimetric* property holds: for any $X \subset A$, $Y \subset B$

$$\left| \frac{E(X:Y)}{d|X|} - \frac{|Y|}{|B|} \right| \leq \frac{\lambda}{d} \sqrt{\frac{|Y|}{|X|}}$$

where $E(X:Y)$ is the cardinality of the cut between X and Y, and $\lambda \leq 2\sqrt{d}$. Thus, setting d to large enough constant (independent from ϵ), satisfies the second condition. At the same time, setting $d = \Theta(1/\eta \cdot 1/\mu)$ so that $\lambda/d < \sqrt{\eta\mu}$ satisfies the first condition. Thus the setting $d = \Theta(1/\eta \cdot 1/\mu)$ satisfies both conditions.

The code C' is again defined as a set of all vectors $G(c)$ for $c \in C$. Let Σ be the alphabet of C'.

Encoding: Clearly, the code is encodable in linear time.

Decoding: The basic idea for the decoding procedure is as follows. We observe that if no list was corrupted (i.e., if there was a codeword c' such that $c'_j \in L_j$ for all j), then we could find c' by solving a *constraint satisfaction problem* induced by the graph G. In particular, we would need to find a vector x (corresponding to the codeword in C) and y (corresponding to the codeword in C') such that x is "compatible" with y. More specifically, we would like to find binary vectors x and y such that if we define vectors

- $v \in \Sigma^n$ such that v_j is the $(y_i + 1)$-th element of the list L_j
- $u \in \{0, \ldots, q-1\}^n$ such that u_i is equal to the k-th symbol of the $(x_i + 1)$-th element in the list L_j, where $\Gamma_k(j) = i$, for a fixed j such that $(i, j) \in E$

then $G(u) = v$. Note that the latter equality can be expressed as a set of binary constraints. If we have such a pair u, v, then either u is a codeword in C (and so we are done), or it can be used to retrieve a codeword in C.

In our case, ϵ fraction of the lists are corrupted. Therefore, instead of trying to satisfy all constraints, we will try to satisfy (say) 99% of them. To find such vectors x and y we will use the Goemans-Williamson SDP-based approximation algorithm for maximum satisfiability of a set of binary linear equations modulo 2. Although the resulting vectors u and v are not exact, we use error-correcting properties of the code C to decode the right codeword.

The details of the decoding algorithm are as follows. Let $L_1 \ldots L_n$ be lists of length 2. For each $i \in A$ let $\{a_i^0, a_i^1\}$ be the two most popular symbols among symbols in $L(i, j)$, $(i, j) \in E$; ties are broken arbitrarily. Then, for each $(i, j) \in E$ we create a linear equation (involving x_i and y_j) expressing compatibility between the symbol represented by x_i and y_j. The interpretation of x_i is that $u_i = a_i^{x_i}$. The interpretation of y_j is that v_j is the $(y_j + 1)$-th symbol from the list L_j.

The linear equations are constructed as follows. Let $< b^0, b^1 >= L(i, j)$. Consider the following cases:

1. $b^0 = b^1$ and $a^0 = a^1$: if $b^0 = a^0$ then create $0 + 0 = 0$; otherwise create $0 + 0 = 1$
2. $b^0 \neq b^1$ and $a^0 = a^1$: if $b^0 = a^0$ create $y_j + 0 = 0$; if $b^1 = a^0$ create $y_j + 0 = 1$; otherwise, create $0 + 1 = 0$

3. $b^0 = b^1$ and $a^0 \neq a^1$: symmetric with respect to the previous case

4. $b^0 \neq b^1$ and $a^0 \neq a^1$: consider the following subcases:

   (a) $\{b^0, b^1\} = \{a^0, a^1\}$: due to a symmetry, we can consider only the case $b^0 = a^0$, $b^1 = a^1$. In this case we create $x_i + y_j = 0$

   (b) $\{b^0, b^1\} \cap \{a^0, a^1\} = \{b\}$: due to a symmetry, we can consider only the case $b^0 = a^0 = b$. In this case we create $x_i + 0 = 0$ and $y_j + 0 = 0$, both with weights $1/2$

   (c) $\{b^0, b^1\} \cap \{a^0, a^1\} = \emptyset$: we create $0 + 0 = 1$

Observe that the total weight of all constraints is equal to dn.

Claim 3 *If there exists $c' \in C'$ such that $COMP(c', \mathcal{L}) \geq (1-\epsilon)n$, then there exists x and y satisfying at least $(1 - d2\epsilon)$ weighted fraction of constraints.*

Proof: Let $c \in C$ be the codeword corresponding to c'. Let $Y = \{j : c'_j \notin L_j\}$, and recall that $|Y| \leq \epsilon n$. Observe that if $c_i \notin \{a_i^0, a_i^1\}$, then at least $1/3$ fraction of edges going out of i lead to Y. By properties of G, the fraction of such i's is at most ϵ. Let X be the set of such i's.

Let x_i, $i \notin X$, be such that $c_i = a_i^{x_i}$, and let y_j, for $j \notin Y$, be such that the $(y_j + 1)$-th element of L_j agrees with the values of c_i for $i \notin X$. The other entries of x and y are set arbitrarily. It is not difficult to verify that all linear constraints corresponding to (i, j), $i \notin X$, $j \notin Y$, are satisfied. Thus x and y satisfies at least a $(1 - 2d\epsilon)$ fraction of all constraints. $\Box$.

In order to find approximate x and y, we use the Goemans-Williamson algorithm for finding the maximum cut in a graph. More specifically, we use its implementation running in *near quadratic* time [KH96], i.e., in time $O(n^2 \log^{O(1)} n)$. As observed in [Zwick98], that algorithm reports a partition which cuts at least $1 - O(\sqrt{\alpha})$, assuming there *exists* a cut which cuts $1 - \alpha$ fraction of edges; we call it an $(\alpha, O(\sqrt{\alpha}))$-approximation scheme.

Unfortunately, we cannot use the fast MAXCUT algorithm directly for our constraint satisfaction problem, since MAXCUT corresponds to satisfying constraints of the form $x + y = 1$, while our set of constraints also contain constraints $x + y = 0$. However, since the constraints are binary, we can implement the latter type of constraints as well, using the following reduction, shown to us by Venkat Guruswami.

Lemma 2 *If there is an (α, β)-approximation scheme for the weighted MAXCUT problem with running time $T(n, m)$ (where n is the number of vertices and m the number of edges), then there is an $(\alpha, 2\beta)$-approximation scheme for maximizing the (weighted) number of satisfied binary linear equations modulo 2, with running time $T(n + 2m, m)$.*

Proof: For simplicity we consider the unweighted binary linear constraints; since the weights in our set of linear constrains are only 1 and $1/2$ (or alternately 2 and 1), we simulate weights 2 by duplicating the constraint.

The reduction is as follows. For each equation of the form $x + y = 1$, an identical equation is generated. For each equation of the form $x + y = 0$, we generate two equations: $x + z = 1$

and $y + z = 1$, where z is a new variable occuring only in these two equations. The new equations are assigned weight $1/2$ each.

Let S be the set of original (general) linear equations, and let S' be the set of the created equations. The correctness of the reduction follows from the following two claims.

Claim 4 *If there is an assignment satisfying $1 - \alpha$ fraction of constraints in S, then the same assignment satisfies $1 - \alpha$ weighted fraction of constraints in S'.*

Claim 5 *If there is an assignment satisfying $1 - \beta$ weighted fraction of constraints in S', then the same asignment satisfies $1 - 2\beta$ fraction of constraints in S.*

□.

By using the above reduction, we can solve our linear constraint satisfaction problem in near quadratic time as in [KH96]. Since we know that there is an assignment satisfying $1-2d\epsilon$ fraction of the original constraints, it follows that the algorithm will report an assignment to vectors x and y satisfying $1 - \delta$ fraction of constraints, for $\delta = O(\sqrt{d\epsilon})$. The vector x induces a vector $u \in \{0, \ldots, q-1\}^n$ such that $COMP(G(u), \mathcal{L}) \geq (1 - d\delta)n$.

It is tempting to assume that u can be now decoded using the code C. Unfortunately, this does not have to be the case. In fact, in addition to a codeword $G(c)$, the lists L_j can represent $G(w)$ for an *arbitrary* vector w. However, as we will see soon, the vector u is very useful in discovering a codeword c.

Let $u' = G(u)$. Define v'_j to be only other element of $L_j - \{u'_j\}$ (if $\{u'_j\} \in L_j$) and to be equal to u'_j otherwise. In addition, let v_i be the symbol that occurs most often in the set of symbols "voting" for v_i (i.e., in the set $\{a_k :< a_1, \ldots, a_d >= v'_j \text{ and } \Gamma_k(j) = i\}$). Our algorithm decodes both u and v using the code C, and reports the result.

In the following we provide a proof of correctness of the above procedure. First, we show that (combinatorially) the code is indeed $(1 - \epsilon', 2, 2)$-recoverable for small $\epsilon' > 0$. We will assume that $\epsilon' > d\delta >> \epsilon$, thus the code is also $(1 - \epsilon, 2, 2)$-recoverable.

Lemma 3 *Let W be a set of vectors $w \in \{0, \ldots, q-1\}^n$ such that for any distinct vectors $w, w' \in W$ we have $D(w, w') \geq \mu n$, and $COMP(G(w), \mathcal{L}) > 1 - \epsilon'$ for $\epsilon' = 1/3 - \eta$. Then $|W| \leq 2$.*

Proof: Assume there are 3 distinct vectors $w, w', w'' \in W$. From the expansion properties of G we have $D(G(w), G(w')) \geq (1-\eta)n$. Moreover, $G(w'')$ differs from both $G(w)$ and $G(w')$ in at least $(1 - 2\eta)n$ positions. Thus, there are $(1 - 3\eta)$ positions in which $G(w), G(w')$ and $G(w'')$ are pairwise different. Since each of the above vectors disagrees with $\mathcal{L}$ on less than ϵ' fraction of positions, it follows that $1 - 3\eta < 3\epsilon'$, which yields contradiction. □

The above lemma implies that if the minimum distance $(1 - \gamma)n$ of C is greater than μn, then there are at most two codewords $G(c^1), G(c^2)$ which agree with $\mathcal{L}$ in at least $1 - \epsilon'$ fraction of positions. We consider two cases: (A) when there is only one such codeword $c^1 = c$, and (B) when there are two such codewords.

Case A. We can assume that $COMP(G(c), \mathcal{L}) \geq (1-\epsilon)n$, since otherwise there is nothing to recover. Consider the first recovered vector u. If $D(u, c) < \mu n$, then we can recover c from u

and we are done. Assume now that $D(u,c) \geq \mu n$. Recall that $COMP(G(u), \mathcal{L}) \geq (1-d\delta)n$. Thus, a fraction of $(1 - d\delta - \epsilon)$ of lists are compatible with both $G(u)$ and $G(c)$. Since $D(u,c) \geq \mu n$, we have $D(G(u), G(c)) \geq (1-\eta)n$. Thus, a fraction of $(1 - d\delta - \epsilon - \eta)$ of lists are compatible with both $G(u)$ and $G(c)$ *and* contain two distinct elements. On those positions, v' and $G(c)$ will be equal. Thus $D(v', G(c)) \leq (d\delta + \epsilon + \eta)n$. By a counting argument it follows that $D(v,c) \leq (d\delta + \epsilon + \eta)n$. Thus if $(d\delta + \epsilon + \eta) < \mu$, we can decode c from v.

Case B. Observe that $D(c^1, c^2) \geq \mu n$. Recall that $COMP(G(c^1), \mathcal{L}) \geq (1-\epsilon')n$, $COMP(G(c^2), \mathcal{L}) \geq (1-\epsilon')n$. In addition, $COMP(G(u), \mathcal{L}) \geq (1-\epsilon')n$. By Lemma 3 it follows that either $D(c^1, u) < \mu n$ or $D(c^2, u) < \mu n$. Without loss of generality assume the first case. Then we decode c^1 from u by using the decoding algorithm for the code C. In order to decode c^2, observe that $D(c^2, u) \geq D(c^2, c^1) - D(c^1, u) \geq \mu n$. Therefore, if we set $c = c^2$, we can apply the same argument as in Case A.

Theorem 2 *There exists $\epsilon > 0$ for which there is a constant rate code which is linear time encodable and $(1-\epsilon, 2, 2)$-recoverable in near quadratic time.*

Proof: By the earlier arguments, it suffices to find constants $1 > \epsilon, \epsilon', \mu, \gamma, \eta, \delta > 0$ and $d > 1$, such that the following constraints are satisfied, for a certain (fixed) constant $b > 1$ (defined by the algorithm of [Zwick98] and isoperimetric inequality):

$$\begin{aligned}
\mu &= (1-\gamma)/2 && (1)\\
d &= b \cdot 1/\eta \cdot 1/\mu && (2)\\
\delta &= b \cdot \sqrt{d\epsilon} && (3)\\
\epsilon' &> d\delta && (4)\\
d\delta &> \epsilon && (5)\\
\epsilon' &= 1/3 - \eta && (6)\\
\mu &> d\delta + \epsilon + \eta && (7)
\end{aligned}$$

This constraints can be satisfied as follows. Set $\gamma = 1/2$, which implies $\mu = 1/4$. Also, set $\eta = 1/4$, which gives $\epsilon' = 1/3 - 1/4 = 1/12$. This also implies $d = 16b$. It remains to specify ϵ, δ. However, as ϵ tends to 0, δ (which is a function of ϵ specified by (3)) tends to 0 as well. In addition, the (strict) inequalities (4), (5) and (7) are satisfied when $\epsilon = \delta = 0$. Thus, by taking ϵ small enough, one can satisfy all constraints. □

5 Decoding the $(1-\epsilon, 2, 2)$-recoverable code via spectral partitioning

In this section we give another algorithm for decoding the code essentially as described in the previous section. We will assume that the graph G is constructed from a Ramanujan graph G_R as in Section 3. In addition we also assume that C can be uniquely decoded from up to $d\epsilon$ fraction of errors and $1-\alpha$ fraction of erasures, for certain α, in linear time. Also, we

assume that C can be decoded from μ fraction of errors. Note that by the results of [GI01] it is feasible as long as $1 - \alpha + 2d\epsilon < 1$ and $\mu < 1/2$.

The algorithm is based on the concept of *spectral partitioning*. The latter technique is a method for partitioning a graph into relatively dense components (in our case $k = 2$ of them), such that the ratio of the number of edges between the clusters to the number of edges inside the clusters is fairly small. This is done by computing the first k eigenvectors of the adjacency matrix of the graph, and then assigning vertices to the clusters based on the respective eigenvalues. The advantage of this approach is that, under certain natural assumptions, it can be implemented to run in $O(kn \log n)$ time, by using iterative algorithms for eigenvector computation. See [KVV01] and references therein for analysis of spectral partitioning algorithms.

In order to use the graph partitioning procedure, we need to reduce the list-recovering problem to the problem of finding "sparse" cut in a graph. Recall that in the previous section we show that list-recovering can be implemented by solving (approximately) the maximum satisfiability problem for a set of "almost" satisfiable linear equations. Another way to solve the same problem is as follows. Fix the lists $\mathcal{L} = L_1 \dots L_n$, which induces the lists $L(i, j)$. As before, let $\{a_i^0, a_i^1\}$ be the two most often occurring symbols in the multiset $M_i = \cup_{(i,j)\in E} L(i, j)$, with a_i^0 being the most frequent one (if M_i contains only one symbol repeated multiple times, we set a_i^1 to a dummy symbol). We use $M_i(b)$ to denote the multiplicity of b in M_i. For simplicity, we also define $n_i^0 = M_i(a_i^0)$ and $n_i^1 = M_i(a_i^1)$. Let c be one of the codewords of the code C to decode, i.e., such that $COMP(G(c), \mathcal{L}) \geq (1 - \epsilon)n$. We consider the following two cases.

Case A: The fraction of i's such that $n_i^0 > n_i^1$ is at least α. In this case, for all i such that $n_i^0 > n_i^1$ we set x_i to a_i^0. Clearly, at least a fraction α of x_i's is set to a symbol. Moreover, at most a fraction of $d\epsilon$ of them have edges (i, j) such that j is a corrupted position. Consider one of the other $n(\alpha - d\epsilon)$ indices i. Since there is no edge from i to a corrupted position j, we have $M_i(c_i) \geq d$. Clearly, $\sum_b M_i(b) \leq 2d$, which implies that for any $b \neq c_i$ we have $M_i(b) \leq M_i(c_i)$. Moreover, this inequality cannot be tight, since $n^0 > n^1$. Thus, $M_i(b) < M_i(c_i)$ for any $b \neq c_i$, which means a_i^0 is the correct symbol c_i.

Thus we can recover one codeword from x by unique decoding of x. The second codeword can be recovered as in the previous section.

Case B: The fraction of i's such that $n_i^0 > n_i^1$ is at most α.

We construct a graph H over the set $U = \{0, 1\} \times \{x, y\} \times [n]$. For each $(i, j) \in E$ we construct the edges in H as follows. Let $L(i, j) = \{b^0, b^1\}$. If $\{a_i^0, a_i^1\} = \{b^0, b^1\}$ (note that this implies $b^0 \neq b^1$), then

- If $a^0 = b^0$, then construct edges $(0, x, i) - (0, y, j)$ and $(1, x, i) - (1, y, j)$
- If $a^0 = b^1$, then construct edges $(0, x, i) - (1, y, j)$ and $(1, x, i) - (0, y, j)$

Otherwise, do not construct any edges.

Observe that if we did construct the edges for every (i, j), and there was at least one codeword $G(c) \in C'$ such that $COMP(G(c), \mathcal{L}) = n$, then the resulting graph would consist of two connected components. The first of them would contain vertices (t, x, i) for $a_i^t = c_i$, and (s, y, j) for $G(c)_j$ is the $(s+1)$-th element in the list L_j. The second component would contain the remaining nodes. Thus, we could recover c by computing the connected components.

In general, some of the positions j are corrupted, and for some pairs (i,j) we do not construct any edges in H. Thus, there will be some (although very few) edges between the components, and some edges within the components will be missing. However, finding a sparse cut in the graph will allow us to approximate the "original" components, which in turn will enable us to discover the codewords. The rest of this section will be devoted to completing this task.

For technical reasons it will be easier to work with the square of the graph H (denoted by H^2) instead of the graph H itself. Note that H^2 might contain loops and multiple edges. Let $U_x = \{(t,x,i) \in U\}$. Let $V(c) \subset U_x$ be a set constructed as follows: for each $i = 1 \ldots n$, if $c_i = a_i^t$ and $c_i \neq a_i^{1-t}$ for some t, then include (x,t,i) in $V(c)$; otherwise, include any one of the vertices $(x,0,i),(x,1,i)$. Let $W(c) = U_x - V(c)$. Let $D = d^2$.

In the following we show how to partition the set U into V' and W', such that V' (and W', resp.) differs from $V(c)$ (and $W(c)$, resp.) at only μ fraction of positions, where we can make μ arbitrarily small by properly adjusting α and ϵ. Once we have V', the decoding is easy. Let $u \in [q]^n$ be a vector such that $u_i = a_i^t$ iff $(t,x,i) \in V'$. It follows that $D(u,c) \leq \mu n$. Therefore, computing such u is sufficient for decoding c (by applying the decoding algorithm for C).

It remains to show how to compute V' and W'. Before that, however, we need some additional notation. Firstly, we order the elements in U_x in such a way that (a) the elements in $V(c)$ come before elements of $W(c)$ and (b) within the two subsequences, we sort all elements (t,x,i) in the increasing order of i's. Let A be the adjacency matrix of the graph $H^2_{|U_x}$ (i.e., the subgraph of H^2 induced by U_x), constructed using the above ordering.

Let B_1, B_2 be two copies of the adjacency matrix of the graph G_R^2, and let B be a block diagonal matrix with blocks B_1 and B_2.

Claim 6 *There are two non-negative integer $2n \times 2n$ matrices E' and E'', such that the total sum of their entries is at most $\delta n D$ for $\delta = 4(\alpha + d\epsilon)$, with the property that $A = B + E' - E''$.*

Proof: We will show that the graph corresponding to A can be transformed into the graph corresponding to B by changing very few edges - the changes will induce the matrices E' and E''.

Firstly, observe that for any i, if $n_i^0 = n_i^1$, and there is no edge (i,j) leading to a corrupted position j, then $n_i^0 = n_i^1 = d$ and $L(i,j) = \{a_i^0, a_i^1\}$ for all j's. Denote the set of such i's by $I(c)$. The size of the set $I(c)$ is at least $(1 - \alpha - d\epsilon)n$.

Consider any triple of vertices i, i', j in G_R, such that $\{i,j\}, \{i',j\}$ are edges in G_R. Note that each such triple corresponds to an edge $\{i,i'\}$ in G_R^2. Let $(t,x,i),(t',x,i') \in V(c)$; this implies that $(1-t,x,i),(1-t',x,i') \in W(c)$. Consider the case when $i,i' \in I(c)$. Since j is not corrupted and $L(i,j) = \{a_i^0, a_i^1\}$, it follows that H includes edges $(t,x,i) - (s,y,j)$ and $(1-t,x,i) - (1-s,y,j)$ for some $s \in \{0,1\}$. By the same argument, H includes edges $(t',x,i') - (s',y,j)$ and $(1-t',x,i') - (1-s',y,j)$ for some $s' \in \{0,1\}$. However, it must be the case that $s = s'$. This holds due to the fact that both a_i^t and $a_i^{t'}$ are compatible with exactly one symbol in L_j, and both of them are compatible with $G(c)_j$, which means that they are compatible with the same element from L_j, which is equivalent to $s' = s$.

Since $s = s'$, the triple i, i', j induces edges $(t, x, i) - (t', x, i')$ and $(1-t, x, i) - (1-t', x, i')$ in the graph H^2. This directly corresponds to the two edges (i, i') in the adjacency matrices B_1 and B_2. Note that different j's can induce multiple edges.

Thus, all differences between A and B occur for triples i, i', j such that not both i, i' are in $I(c)$. The total number of such triples is at most $2 \cdot 2 \cdot n(\alpha + d\epsilon)D$.

□

Observe that A is a real, symmetric matrix. Therefore, all of its eigenvalues are non-negative (note that some of them could be equal to 0 if A does not have full rank). Let $\lambda_1 \ldots \lambda_n$ be the eigenvalues of A sorted in a non-increasing order. Moreover, let $v_1 \ldots v_n$ denote the eigenvectors of A corresponding to the eigenvalues. Recall that the eigenvectors are orthogonal; we will assume that $|v_i|_2 = 1$ for $i = 1 \ldots n$.

The partitioning algorithm consists of the following steps:

1. Compute the two eigenvectors v_1, v_2 of the matrix A.
2. Let 1 be a unit vector with all coordinates equal to $1/\sqrt{n}$. Compute a projection $\hat{1}$ of 1 onto $\mathrm{span}(v_1, v_2)$ (i.e., the space spanned by the vectos v_1, v_2) and normalize it.
3. Compute a unit vector $\hat{s} \in \mathrm{span}(v_1, v_2)$ which is orthogonal to $\hat{1}$.
4. Compute a vector $s' = 1/\sqrt{n} \cdot \mathrm{sgn}(\hat{s})$
5. Compute the set V' consisting of all elements whose corresponding coordinates in s' are negative, and $W' = U - V'$. Output the pair $\{V', W'\}$.

The intuition behind the above (perhaps somewhat unintuitive) algorithm is as follows. Let s be a vector with entries $-1/\sqrt{n}$ for coordinates in $V(c)$, and entries $1/\sqrt{n}$ for coordinates in $W(c)$. Also, let $E = E' - E''$. Clearly $|As|_2 \geq |Bs|_2 - |Es|_2$. In the following we will show that the 2-norm of E is quite small (less than βD for small β). On the other hand, $|Bs|_2 = D$. Thus, $|As|_2 \geq (1-\beta)D$. At the same time, s is orthogonal to 1. For the latter vector, we also know that $|A1|_2 \geq |B1|_2 - |E1|_2 \geq (1-\beta)D$. Thus, 1 and s are "close" to (resp.) the first and second eigenvectors of A. If we were lucky, v_1 would be close to 1, v_2 would be close to s and then we could partition the set U according to v_2.

Unfortunately, the difference between λ_1 and λ_2 could be very small (in fact, as far as we know, these two values could be equal). In this case, we are only ensured that the *span* of v_1 and v_2 is close to the span of 1 and s. However, since we know v_1, v_2 and 1, we can approximate s via the "orthogonalization" procedure described above.

In the following we provide a formal proof of correctness of the above procedure. For now we assume that $|E|_2 \leq \beta D$ for a (small) constant $\beta > 0$, we will prove this fact later.

Let $\lambda_i(A)$ denote the ith largest eigenvalue of the matrix A. It follows from the above discussion that $\lambda_1(A) \geq D(1-\beta)$. In addition, from standard linear algebra we know that $\lambda_2(A) \geq \lambda_2(B) - \lambda_1(E) \geq D(1-\beta)$. Since the spectrum of B is a union of spectra of B_1 and B_2, we can also conclude that $\lambda_3(B) = \lambda_2(B_1)$. Since B_1 is an adjacency matrix of G^2, we know that $\lambda_3(B) \leq 4\sqrt{D}$. Therefore, $\lambda_3(A) \leq 4\sqrt{D} + \lambda_1(E) \leq 4\sqrt{D} + \beta D = \beta' D$.

Let $1 = \sum_i p_i v_i$ and $s = \sum_i q_i v_i$. We know that

$$|A1|_2 = |\sum \lambda_i(A) p_i v_i|_2 \geq (1-\beta)D$$

Since $\lambda_i(A) \leq \beta' D$ for $i \geq 3$, we know that

$$(1-\beta)D \leq |\sum \lambda_i(A) p_i v_i|_2 \leq \sqrt{\lambda_1^2(A)p_1^2 + \lambda_2^2(A)p_2^2} + \beta' D \leq D(1+\beta)\sqrt{p_1^2+p_2^2} + \beta' D$$

Thus we can conclude that $\sqrt{p_1^2+p_2^2} \geq (1-\beta-\beta')/(1+\beta) = 1-\beta''$.

In a similar way we obtain that $\sqrt{q_1^2+q_2^2} \geq 1-\beta''$. Thus, we are in the following setting. We have a pair of orthonormal vectors v_1, v_2, and a pair of orthonormal vectors 1 and s, the latter from $\{-1/\sqrt{n}, 1/\sqrt{n}\}^n$. Moreover, we know that the norm of the projections of both 1 and s on $\mathrm{span}(v_1, v_2)$ is at least $1-\beta''$. Our task is: given v_1, v_2 and 1, find s' such that $s \cdot s' \geq 1 - f\beta''$, for some constant f (independent from β''). In the following, we show that the above algorithms completes this task.

We start from expressing the relation between 1, $\hat{1}$, s, s' and $\hat{s}$ in terms of *angles* rather than dot products. Then we will use the fact that the angle between vectors satisfies triangle inequality. We will denote the angle between vectors v and v' by $\angle(v, v')$.

By the previous discussion, we know that (for certain absolute constant f') we have $\angle(1, s) = \pi/2$, $\angle(1, \hat{1}) \leq \arccos(1-\beta'') \leq f'\sqrt{\beta''}$ This implies that $\angle(\hat{1}, s) \geq \pi/2 - 2f'\sqrt{\beta''}$. Since $\angle(\hat{s}, \hat{1}) = \pi/2$, it follows that either $\angle(s, \hat{s}) \leq 2f'\sqrt{\beta''}$ or $\angle(s, -\hat{s}) \leq 2f'\sqrt{\beta''}$; we will assume the first case.

Recall that $s \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^n$. If we set $s' = 1/\sqrt{n} \cdot \mathrm{sgn}(\hat{s})$, then $s' \cdot \hat{s} = \max_{s' \in \{-1/\sqrt{n}, 1/\sqrt{n}\}^n} s' \cdot \hat{s}$. Thus, $s' \cdot \hat{s} \geq s \cdot \hat{s}$. In other words, $\angle(s', \hat{s}) \leq \angle(s, \hat{s}) \leq 2f'\sqrt{\beta''}$. Therefore, $\angle(s', s) \leq \angle(s', \hat{s}) + \angle(s, \hat{s}) \leq 4f'\sqrt{\beta''}$, which can be expressed as $s' \cdot s \geq 1 - f\beta''$ for certain constant f. If we set $\mu = f\beta''$, it implies that the retrieved set V' differs from $V(c)$ in at most μ fraction of positions.

It remains to show that the 2-norm of E is small. We will show that the 2-norm of E' is small (E'' can be handled in the same way). To this end, we use the fact that E' is an adjacency matrix of a sparse subgraph of an expander graph (note that sparsity alone is not sufficient). We first prove the neccessary lemma for a non-bipartite case, and show how to proceed with the bipartite expanders.

Lemma 4 *Let J be a Δ-regular graph over n vertices with second eigenvalue λ. Let F be a subgraph of J containing $\delta\Delta n$ edges of J. Then the 2-norm of F is at most $\sqrt{\delta}\Delta b + \lambda(1 + 2/b) + 2\Delta/b$, for any $b > 2$.*

Proof: We will use the same notation for a graph and its adjacency matrix. Let x be the first eigenvector of F; without loss of generality we can assume that $x \geq 0$ coordinate-wise.

Let x' be a vector obtained from x by rounding all coordinates in x which are smaller than $b/\sqrt{n}$ to $b/\sqrt{n}$. Let y be a binary vector, having 1s at the positions where x' is equal to $b/\sqrt{n}$. Moreover, let $z = x' - \sqrt{b}/n \cdot y$. Clearly $|z|_2 \leq 1, z \geq 0$.

We need to bound $|Fx|_2$ from the above. Since $|Fx|_2 \leq |Fx'|_2 \leq |Fz|_2 + |F(b/\sqrt{n} \cdot y)|_2$, it is sufficient to upper bound $|Fz|_2$ and $|F(b/\sqrt{n} \cdot y)|_2$.

Bound for $b/\sqrt{n} \cdot |Fy|_2$. Recall that all coordinates in y are at most 1. Moreover, recall that the graph F has at most $\delta n\Delta$ edges. Let d_i be the degree of node i in F. Clearly $\sum_i d_i \leq 2\delta\Delta n$.

We know that

$$|F(b/\sqrt{n})y|_2^2 \leq \sum_i (d_i \cdot b/\sqrt{n})^2 = b^2/n \sum_i d_i^2$$

Since $\sum_i d_i \leq 2\delta\Delta n$, it follows that the above expression achives the maximal value of $b^2/n \cdot 2\delta n\Delta^2 = 2\delta b^2\Delta^2$. Thus, $|F(b/\sqrt{n})y|_2 \leq \sqrt{2\delta}b\Delta$.

Bound for $|Fz|_2$. Since $z \geq 0$, we know that $|Fz|_2 \leq |Jz|_2$. Thus, it is sufficient to bound $|Jz|_2$. Observe that at most $k \leq n/b^2$ positions in z are nonzero. These positions correspond precisely to the positions which are equal to 0 in y. Thus, $|z|_1 \leq \sqrt{n/b^2}$. Let $t = \frac{|z|_1}{n-k} \leq \frac{|z|_1}{n/2}$. Finally, define $z' = z - ty$. From the definition it follows that $z' \cdot 1 = 0$. Thus $|Jz'|_2 \leq \lambda|z'|_2 \leq \lambda(|z|_2 + t|y|_2) \leq \lambda(1 + \frac{|z_1|}{n/2}\sqrt{n}) \leq \lambda(1 + 2/b)$. Therefore

$$|Jz|_2 \leq |Jz'|_2 + |J(ty)|_2 \leq \lambda(1+2/b) + t|J|_2 \cdot |y|_2 \leq \lambda(1+2/b) + \frac{\sqrt{n/b^2}}{n/2}|J|_2\sqrt{n} \leq \lambda(1+2/b) + 2\Delta/b$$

By combining the bounds the theorem follows.

□

In our case, the matrix E' can be decomposed into a sum of two matrices of subgraphs of G_R^2 (which can be directly handled using the above theorem) and a matrix E_3 of a subgraph of a bipartite graph (call it G'^2) with all edges between $V(c)$ and $W(c)$. To handle E_3, one can observe that G'^2 is a *double cover* of G_R^2. We say that a bipartite graph $K = (V, V, S)$ is a double cover of $K' = (V, S')$ if S contains all edges $(i, j), (j, i)$ for $\{i, j\} \in S'$. By increasing the number of edges in E_3 by at most a factor of 2, we can ensure that the graph E_3 is a double cover of a subgraph (say $\hat{E}$) of G_R^2. By applying Lemma 4 to $\hat{E}$ and G_R^2, it follows that $|\hat{E}|_2 \leq 2Db\sqrt{2\delta} + \lambda(1 + 2/b) + 2D/b$. By standard arguments it follows that $|E_3|_2$ is at most twice that bound.

Thus, $|E|_2 \leq f''(Db\sqrt{\delta} + \lambda(1 + 2/b) + 2D/b)$, for a constant f''.

Running time. To estimate the running time, observe that we can use the Orthogonal Iteration method to compute eigenvectors v_1, v_2 and the corresponding eigenvalues. The running time of that method is upper bounded by the time needed to multiply the matrix A by a vector (which is $O(n)$) times the number of iterations. The latter quantity is upper bounded by $\log n$ times the eigenvalue gap between the kth and $(k+1)$th eigenvalues of A. Since $\lambda_2(A) \geq D(1 - \beta)$, while $\lambda_3(A) = O(\sqrt{D}) + \beta D$, it follows that the number of iterations is $O(\log n)$. Thus, the total running time is $O(n \log n)$.

Theorem 3 *There exists $\epsilon > 0$ such that for any block-length n there is a constant rate code which is linear time encodable and $(1 - \epsilon, 2, 2)$-recoverable in time $O(n \log n)$.*

Proof: By the previous discussion, it suffices to fix the values of constants $1 > \epsilon, \delta, \alpha, \beta, \beta', \beta'', \mu > 0$, $b > 2$, $d > 3$, such that for a certain (fixed) values of constants f, f', f'' the following constrains are satisfied:

$$\begin{aligned} 1 - \alpha + 2d\epsilon &< 1 \\ \mu &< 1/2 \\ \delta &= 4\alpha + 4\epsilon d \end{aligned}$$

$$\begin{aligned} D &= d^2 \\ 4\sqrt{D} + \beta D &= \beta' D \\ 1 - \beta'' &= (1 - \beta - \beta')/(1 + \beta) \\ \mu &= f\beta'' \\ \beta D &\leq f''[Db\sqrt{\delta} + \sqrt{D}(1 + 2/b) + 2D/b] \end{aligned}$$

It is not difficult to verify that by setting d to be sufficiently large and then α, ϵ to be sufficiently small, one can satisfy all constraints. $\Box$.

6 A $(1/3 + \epsilon, O(1))$-decodable code with linear encoding and decoding time

In this section we show that by concatenating a $(1 - \epsilon, 2, 2)$-recoverable code described in the previous section with a carefully chosen inner code, we can obtain a $(1/3+\epsilon, O(1))$-decodable code, with constant rate and alphabet size, which can be encoded and decoded in $O(n)$ time. To this end, we first observe that the $O(n \log n)$ running time of the spectral partitioning algorithm is preserved even if the alphabet size q of the "left" code C is non-constant, but e.g. $\Theta(n)$ (note that the finiteness of q was not assumed anywhere). Thus, if we take C to be a code of block-length n with alphabet size n^a for some $a > 0$, then the resulting code C' has alphabet size $q_{out} = (n^a)^b$ for certain constant b. Such a code C, with $O(n)$ encoding and (unique) decoding time, can be constructed as in [GI01].

The inner code C_{in} is chosen to be a $(1/3 + \epsilon/2, 2)$-decodable code, with constant rate r_{in} and constant alphabet size q_{in}. The number of bits needed to reprent each codeword is equal to $ab \log q_{in} \log n / r_{in}$. We set a such that the latter quantity is less than $\log n$. As a result, we can decode C_{in} in unit time by performing a table lookup, and the table size is at most n. Also, given the list of all codewords of C_{in}, one can prepare the lookup table in $O(n)$ time.

The code C_{in} itself is chosen to be a random "pseudolinear" code. By Lemma 9.2 ([G01a], p. 194), one can adjust q_{in} and ϵ so that $r_{in} > 0$. Such a code can be constructed probabilistically in $O(\log^{O(1)} n)$ time.

The final component of the construction is bipartite graph G. Specifically, let G be a bipartite graph $G = (A, B, E)$, $|A| = n$, with left degree d and right degree d'. The left degree d is equal to the block-length n_{in} of C_{in}. The graph G should have the following properties:

- d' is a constant independent from n
- for any $S \subset B$, $|S| \geq (1/3+\epsilon)|B|$, the fraction of $i \in A$ which have less than $1/3+\epsilon/2$ fraction of neighbors in S, is at most ϵ

The graph can be again constructured from Ramanujan graphs. In particular, let $G' = (A', B, E')$ be a Ramanujan graph with degree d', such that $|A'| = |B|$. We construct G by splitting A' into sets of size d/d', and collapsing each set into one vertex. The resulting

graph G satisfies the aforementioned properties for proper value of the degree d'. Indeed, assume that there exists $S \subset B$, $|S| \geq (1/3+\epsilon)|B|$ such that the set

$$S' = \{i \in A : i \text{ has less than } 1/3+\epsilon/2 \text{ fraction of neighbors in } S\}$$

is of size at least $\epsilon|A|$. In this case let the set $S'' \subset A'$ contain all vertices collapsed into any vertex of S'. It follows that $\frac{E(S'':S)}{d|S''|} < 1/3+\epsilon/2$, while $\frac{|S|}{|B|} \geq 1/3+\epsilon$. By isoperimetric inequality, it follows that setting $d' = \Theta(1/\epsilon^3)$ is sufficient for this scenario not to occur.

We define $G_{C_{in}} : [q_{out}]^n \to [[q_{in}]^{d'}]^{|B|}$ in the following way. For any $x \in [q_{out}]^n$, we first create a vector $x' \in [q_{in}]^{dn}$ by replacing each symbol x_i by $C_{in}(x_i)$. Then we set $G_{C_{in}}(x) = G(x')$.

The set of all vectors $G_{C_{in}}(c)$, $c \in C$, forms our final code C'. The code has block-length $\Theta(n \log n)$, finite rate and alphabet size. We mention that this "graph-concatenation" operation on C, C_{in} and G has been introduced in [AEL95] and further explored in [GI02].

It is easy to verify that the code C can be encoded in $O(n)$ time. Assume we are given a "corrupted" codeword x and we want to decode a codeword c such that $D(x,c) \geq (1-\epsilon)|B|$. In the first step, each node i constructs a vector y_i by concatenating all symbols x_j, $(i,j) \in E$. Then it applies the list-decoding algorithm for code C_{in} on y_i and outputs a list L_i (of length 2). From the properties of G it follows that the fraction of i's that have less than $(1/3+\epsilon/2)d$ neighbors in the set of symbols in x that agree with c, is less than ϵ. For all other i's, the list L_i contains c_i. Thus, we can decode c from the lists $L_1 \ldots L_n$ by using the decoding algorithm for C.

The total running time of the algorithm is $O(n \log n)$. Since the block-length of the code C' is equal to $|B| = \Theta(n \log n)$, the running time is linear in the block-length of C'. Thus, we obtain the following theorem.

Theorem 4 *For any $\epsilon > 0$ and block-length n, there is a code which is encodable and $(1/3+\epsilon, 2)$-decodable in time $O(n)$. The code can be constructed in randomized $O(\log^{O(1)} n)$ time. The code has constant rate and alphabet size. The big-Oh notation in the running time hides factors not depending on n.*

We also mention that we can improve the above theorem even further, and make the codes explicit, while keeping the running times unchanged. To this end we construct a code C'' in the same way as C', but we use a modification of the code from Theorem 4 as the inner code. The modified code has still block length $O(\log n)$, but is constructed from a "left" code with constant alphabet size (and thus no further concatenation with a random code is needed). Thus, the inner code is explicit, and the lookup table for its decoding can be constructed in $O(n)$ time. The latter bound is subsumed by the encoding time.

Theorem 5 *For any $\epsilon > 0$ and block-length n, there is an explicit code which is encodable and $(1/3+\epsilon, 2)$-decodable in time $O(n)$. The code has constant rate and alphabet size.*

7 Conclusions

In this paper we presented several constructions of list-decodable or recoverable codes. All our codes have linear-time encoding algorithms, and linear or near-quadratic time decoding procedures.

The main open problem left out by this work is: is it possible to correct up to $1 - \epsilon$ fraction of errors in linear time, for any $\epsilon > 0$, while preserving the constant rate of the code ? Very recently, Venkat Guruswami and the author discovered a code construction which indeed enables us to achieve such a result. The construction follows by a generalization of the approach used in this paper. We will include this result in the full version of this paper.

Acknowledgements. The author would like to thank Venkat Guruswami, Jon Feldman and Mihai Badoiu, for very helpful comments on a preliminary version of this paper.

References

[ABN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ronny Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

[AC88] Noga Alon and Fan R. K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Math.* 72 (1988), pp. 15-19.

[AEL95] Noga Alon, Jeff Edmonds and Michael Luby. Linear time erasure codes with nearly optimal recovery. *Proceedings of FOCS'95.*

[ALRS92] Sigal Ar, Richard J. Lipton, Ronitt Rubinfeld and Madhu Sudan. Reconstructing Algebraic Functions from Mixed Data. *Proceedings of FOCS'92.*

[BN00] Daniel Bleichenbacher and P. Nguyen, Noisy Polynomial Interpolation and Noisy Chinese Remaindering. *Proceedings of EUROCRYPT '2000*, LNCS vol. 1807, Springer-Verlag, pp. 53-69, 2000.

[El57] Peter Elias. List decoding for noisy channels. *Wescon Convention Record*, Part 2, Institute of Radio Engineers (now IEEE), pp. 94-104, 1957.

[G01a] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes.* Ph.D thesis, Massachusetts Institute of Technology, August 2001.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, NV, October 2001.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal Linear-Time Codes for Unique Decoding and New List-Decodable Codes Over Smaller Alphabets. *Proceedings of STOC'02.*

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995.

[KH96] Philip N. Klein and Hsueh-I Lu. Efficient Approximation Algorithms for Semidefinite Programs Arising from MAX CUT and COLORING. Proceedings of STOC'96, pp. 338-347.

[KVV01] Ravi Kannan, Santosh Vempala and Adrian Vetta. On clusterings: good, bad and spectral. Proceedings of FOCS'00.

[LPS88] Alex Lubotzky, R. Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[RR00] Ronny Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46(1):246–257, January 2000.

[S95] Daniel Spielman. *Computationally Efficient Error-Correcting Codes and Holographic Proofs*. Ph.D thesis, Massachusetts Institute of Technology, June 1995.

[S96] Daniel Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996.

[S97] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[TZ01] Amnon Ta-Shma and David Zuckerman. Extractor Codes. *Proceedings of STOC'01.*

[Woz58] J. M. Wozencraft. List Decoding. *Quarterly Progress Report*, Research Laboratory of Electronics, MIT, Vol. 48 (1958), pp. 90-95.

[Zwick98] Uri Zwick. Finding almost satisfying assignments. *Proceedings of STOC'98*, pp. 551-560.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'