Appendix E

Explicit Constructions

> *It is easier for a camel to go through the eye of a needle, than for a rich man to enter into the kingdom of God.*
>
> Matthew, 19:24.

Complexity theory provides a clear definition of the intuitive notion of an explicit construction. Furthermore, it also suggests a hierarchy of different levels of explicitness, referring to the ease of constructing the said object. The basic levels of explicitness are provided by considering the complexity of fully constructing the object (e.g., the time it takes to print the truth-table of a finite function). In this context, explicitness often means outputting a full description of the object in time that is polynomial in the length of that description. Stronger levels of explicitness emerge when considering the complexity of answering natural queries regarding the object (e.g., the time it takes to evaluate a fixed function at a given input). In this context, (strong) explicitness often means answering such queries in polynomial-time. The aforementioned themes are demonstrated in our brief overview of explicit constructions of *error correcting codes* and *expander graphs*. These constructions are, in turn, used in various parts of the main text.

Summary: We review several popular constructions of error correcting codes, culminating with the construction of a concatenated code that combines a Reed-Solomon code with a "mildly explicit" construction of a small code. We also review briefly the notions of locally testable and locally decodable codes, and a useful "list decoding bound" (i.e., bounding the number of codewords that are close to any single sequence).

We review the two standard definitions of expanders, two levels of explicitness, and two properties of expanders that are related to (single-step and multi-step) random walks on them. We then review two explicit constructions of expander graphs.

E.1 Error Correcting Codes

In this section we highlight some issues and aspects of coding theory that are most relevant to the current book. The interested reader is referred to [209] for a more comprehensive treatment of the computational aspects of coding theory. Structural aspects of coding theory, which are at the traditional focus of that field, are covered in standard textbook such as [156].

Loosely speaking, an error correcting code is a mapping of strings to longer strings such that any two different strings are mapped to a corresponding pair of strings that are far apart (and not merely different). Specifically, $C : \{0,1\}^k \to \{0,1\}^n$ is a (binary) code of distance d if for every $x \neq y \in \{0,1\}^k$ it holds that $C(x)$ and $C(y)$ differ on at least d bit positions.

It will be useful to extend this definition to sequences over an arbitrary alphabet Σ, and to use some notations. Specifically, for $x \in \Sigma^m$, we denote the i^{th} symbol of x by x_i (i.e., $x = x_1 \cdots x_m$), and consider codes over Σ (i.e., mappings of Σ-sequences to Σ-sequences). The mapping (code) $C : \Sigma^k \to \Sigma^n$ has distance d if for every $x \neq y \in \Sigma^k$ it holds that $|\{i : C(x)_i \neq C(y)_i\}| \geq d$. The members of $\{C(x) : x \in \Sigma^k\}$ are called codewords (and in some texts this set itself is called a code).

In general, we define a metric, called Hamming distance, over the set of n-long sequences over Σ. The Hamming distance between y and z, where $y, z \in \Sigma^n$, is defined as the number of locations on which they disagree (i.e., $|\{i : y_i \neq z_i\}|$). The Hamming weight of such sequences is defined as the number of non-zero elements (assuming that one element of Σ is viewed as zero). Typically, Σ is associated with an additive group, and in this case the distance between y and z equals the Hamming weight of $w = y - z$, where $w_i = y_i - z_i$ (for every i).

Asymptotics. We will actually consider infinite families of codes; that is, $\{C_k : \Sigma_k^k \to \Sigma_k^{n(k)}\}_{k \in S}$, where $S \subseteq \mathbb{N}$ (and typically $S = \mathbb{N}$). (N.B., we allow Σ_k to depend on k.) We say that such a family has distance $d : \mathbb{N} \to \mathbb{N}$ if for every $k \in S$ it holds that C_k has distance $d(k)$. Needless to say, both $n = n(k)$ (called the block-length) and $d(k)$ depend on k, and the aim is to have a linear dependence (i.e., $n(k) = O(k)$ and $d(k) = \Omega(n(k))$). In such a case, one talks of the relative rate of the code (i.e., the constant $k/n(k)$) and its relative distance (i.e., the constant $d(k)/n(k)$).

In general, we will often refer to *relative distances* between sequences. For example, for $y, z \in \Sigma^n$, we say that y and z are ε-close (resp., ε-far) if $|\{i : y_i \neq z_i\}| \leq \varepsilon \cdot n$ (resp., $|\{i : y_i \neq z_i\}| \geq \varepsilon \cdot n$).

Explicitness. A mild notion of explicitness refers to constructing the list of all codewords in time that is polynomial in its length (which is exponential in k). A more standard notion of explicitness refers to generating a specific codeword (i.e., producing $C(x)$ when given x), which coincides with the encoding task mentioned next. Stronger notions of explicitness refer to other computational problems concerning codes (see next).

Computational problems. The most basic computational tasks associated with codes are encoding and decoding (under noise). The definition of the encoding task is straightforward (i.e., map $x \in \Sigma_k^k$ to $C_k(x)$), and an efficient algorithm is required to compute each symbol in $C_k(x)$ in $\mathrm{poly}(k, \log|\Sigma_k|)$-time.[1] When defining the decoding task we note that "minimum distance decoding" (i.e., given $w \in \Sigma_k^{n(k)}$, find x such that $C_k(x)$ is closest to y (in Hamming distance)) is just one natural possibility. Two related variants, regarding a code of distance d, are:

Unique decoding: Given $w \in \Sigma_k^{n(k)}$ that is at Hamming distance less than $d(k)/2$ from some codeword $C_k(x)$, retrieve the corresponding decoding of $C_k(x)$ (i.e., retrieve x).

Needless to say, this task is well-defined because there cannot be two different codewords that are each at Hamming distance less than $d(k)/2$ from w.

List decoding: Given $w \in \Sigma_k^{n(k)}$ and a parameter $d' \geq d(k)/2$, output a list of all $x \in \Sigma_k^k$ that are at Hamming distance at most d' from w.

Typically, one considers the case that $d' < d(k)$. See Section E.1.3 for discussion of upper-bounds on the number of codewords that are within a certain distance from a generic sequence.

Two additional computational tasks are considered in Section E.1.2.

Linear codes. Associating Σ_k with some finite field, we call a code $C_k : \Sigma_k^k \to \Sigma_k^{n(k)}$ linear if it satisfies $C_k(x+y) = C_k(x) + C_k(y)$, where x and y (resp., $C_k(x)$ and $C_k(y)$) are viewed as k-dimensional (resp., $n(k)$-dimensional) vectors over Σ_k, and the arithmetic is of the corresponding vector space. A useful property of linear codes is that their distance equals the Hamming weight of the lightest codeword other than $C_k(0^k)$; that is, $\min_{x \neq y}\{|\{i : C_k(x)_i \neq C_k(y)_i\}|\}$ equals $\min_{x \neq 0^k}\{|\{i : C_k(x)_i \neq 0\}|\}$. Another useful property is that the code is fully specified by a k-by-$n(k)$ matrix, called the generating matrix, that consists of the codewords of some fixed basis of Σ_k^k. That is, the set of all codewords is obtained by taking all $|\Sigma_k|^k$ different linear combination of the rows of the generating matrix.

E.1.1 A few popular codes

Our focus will be on explicitly constructible codes; that is, (families of) codes of the form $\{C_k : \Sigma_k^k \to \Sigma_k^{n(k)}\}_{k \in S}$ that are coupled with efficient encoding and decoding algorithms. But before presenting a few such codes, let us consider a non-explicit construction.

Proposition E.1 (random linear codes): *Let $c > 1$ and $n, d : \mathbb{N} \to \mathbb{N}$ be such that, for all sufficiently large k, it holds that $n(k) > \max(c \cdot k/(1 - H_2(d(k)/n(k))), 2d(k))$,*

[1] This formulation is not the one common in coding theory, but it is the most natural one for our applications. On one hand, this formulation is applicable also to codes with super-polynomial block-length. On the other hand, this formulation does not support a discussion of practical algorithms that compute the codeword faster than by computing each of its bits separately.

where $H_2(\alpha) \stackrel{\text{def}}{=} \alpha\log_2(1/\alpha) + (1-\alpha)\log_2(1/(1-\alpha))$. *Then, for all sufficiently large* k*, with high probability, a random linear transformation of* $\{0,1\}^k$ *to* $\{0,1\}^{n(k)}$ *constitutes a code of distance* $d(k)$.

Thus, *for every constant* $\delta \in (0,0.5)$ *there exists a constant* $\rho > 0$ *and an infinite family of codes* $\{C_k : \{0,1\}^k \to \{0,1\}^{k/\rho}\}_{k\in\mathbb{N}}$ *of relative distance* δ. Specifically, $\rho = (1-H_2(\delta))/c$ will do.

Proof: We consider a uniformly selected k-by-$n(k)$ generating matrix over GF(2), and upper-bound the probability that it yields a linear code of distance less than $d(k)$. We use a union bound on all possible 2^k-1 linear combinations of the rows of the generating matrix, where for each such combination we compute the probability that it yields a vector of Hamming weight less than $d(k)$. Observe that the result of each such linear combination is uniformly distributed over $\{0,1\}^{n(k)}$, and thus has Hamming weight less than $d(k)$ with probability $\sum_{i=0}^{d(k)-1}\binom{n(k)}{i}\cdot 2^{-n(k)} < 2^{-(1-H_2(d(k)/n(k)))\cdot n(k))}$. Using $(1-H_2(d(k)/n(k)))\cdot n(k) > c\cdot k$, the proposition follows. ■

E.1.1.1 A mildly explicit version of Proposition E.1

Note that Proposition E.1 yields a (deterministic) $\exp(k\cdot n(k))$-time algorithm that finds a linear code of distance $d(k)$. The time bound can be improved to $\exp(k+n(k))$, by observing that we may choose the rows of the generating matrix one by one, making sure that all non-empty linear combinations of the current rows have weight at least $d(k)$. Note that the proof of Proposition E.1 can be adapted to assert that as long as we have less than k rows a random choice of the next row will do with high probability. Note that in the case that $n(k) = O(k)$, this yields an algorithm that runs in time that is polynomial in the size of the code (i.e., the number of codewords). Needless to say, this mild level of explicitness is inadequate for most coding applications; however, it will be useful to us in §E.1.1.5.

E.1.1.2 The Hadamard Code

The Hadamard code is the longest (non-repetitive) *linear* code over $\{0,1\} \equiv \text{GF}(2)$. That is, $x \in \{0,1\}^k$ is mapped to the sequence of all $n(k) = 2^k$ possible linear combinations of its bits (i.e., bit locations in the codewords are associated with k-bit strings, and location $\alpha \in \{0,1\}^k$ in the codeword of x holds the value $\sum_{i=1}^k \alpha_i x_i$). It can be verified that each non-zero codeword has weight 2^{k-1}, and thus this code has relative distance $d(k)/n(k) = 1/2$ (albeit its block-length $n(k)$ is exponential in k).

Turning to the computational aspects, we note that encoding is very easy. As for decoding, the warm-up discussion at the beginning of the proof of Theorem 7.7 provides a very fast probabilistic algorithm for unique decoding, whereas Theorem 7.8 provides a very fast probabilistic algorithm for list decoding.

We mention that the Hadamard code has played a key role in the proof of the PCP Theorem (Theorem 9.16); see §9.3.2.1.

A propos long codes. We note that the longest (non-repetitive) binary code (called the Long-Code and introduced in [27]) is extensively used in the design of "advanced" PCP systems (see, e.g., [112, 113]). In this code, a k-bit long string x is mapped to the sequence of $n(k) = 2^{2^k}$ values, each corresponding to the evaluation of a different Boolean function at x; that is, bit locations in the codewords are associated with Boolean functions such that the location associated with $f:\{0,1\}^k \to \{0,1\}$ in the codeword of x holds the value $f(x)$.

E.1.1.3 The Reed–Solomon Code

A Reed-Solomon code is defined for a non-binary alphabet, which is associated with a finite field of n elements, denoted GF(n). For any $k < n$, we consider the mapping of univariate degree $k-1$ polynomials over GF(n) to their evaluation at all field elements. That is, $p \in \mathrm{GF}(n)^k$ (viewed as such a polynomial), is mapped to the sequence $(p(\alpha_1), ..., p(\alpha_n))$, where $\alpha_1, ..., \alpha_n$ is a canonical enumeration of the elements of GF(n).[2]

The Reed-Solomon code offers infinite families of codes with constant rate and constant relative distance (e.g., by taking $n(k) = 3k$ and $d(k) = 2k$), but the alphabet size grows with k (or rather with $n(k) > k$). Efficient algorithms for unique decoding and list decoding are known (see [208] and references therein). These computational tasks correspond to the extrapolation of polynomials based on a noisy version of their values at all possible evaluation points.

E.1.1.4 The Reed–Muller Code

Reed-Muller codes generalize Reed-Solomon codes by considering multi-variate polynomials rather than univariate polynomials. Consecutively, the alphabet may be any finite field, and in particular the two-element field GF(2). Reed-Muller codes (and variants of them) are extensively used in complexity theory; for example, they underly Construction 7.11 and the PCP constructed at the end of §9.3.2.2. The relevant property of these codes is that, under a suitable setting of parameters that satisfies $n(k) = \mathrm{poly}(k)$, they allow super fast "codeword testing" and "self-correction" (see discussion in Section E.1.2).

For any prime power q and parameters m and r, we consider the set, denoted $P_{m,r}$, of all m-variate polynomials of total degree at most r over GF(q). Each polynomial in $P_{m,r}$ is represented by the $k = \log_q |P_{m,r}|$ coefficients of all relevant monomials, where in the case that $r < q$ it holds that $k = \binom{m+r}{m}$. We consider the code $C : \mathrm{GF}(q)^k \to \mathrm{GF}(q)^n$, where $n = q^m$, mapping m-variate polynomials of total degree at most r to their values at all q^m evaluation points. That is, the m-variate polynomial p of total degree at most r is mapped to the sequence of values $(p(\overline{\alpha}_1), ..., p(\overline{\alpha}_n))$, where $\overline{\alpha}_1, ..., \overline{\alpha}_n$ is a canonical enumeration of all the m-tuples of GF(q). The relative distance of this code is lower-bounded by $(q-r)/q$.

[2] Alternatively, we may map $(v_1, ..., v_k) \in \mathrm{GF}(n)^k$ to $(p(\alpha_1), ..., p(\alpha_n))$, where p is the unique univariate polynomial of degree $k-1$ that satisfies $p(\alpha_i) = v_i$ for $i = 1, ..., k$. Note that this modification amounts to a linear transformation of the generating matrix.

In typical applications one sets $r = \Theta(m^2 \log m)$ and $q = \text{poly}(r)$, which yields $k > m^m$ and $n = \text{poly}(r)^m = \text{poly}(m^m)$. Thus we have $n(k) = \text{poly}(k)$ but not $n(k) = O(k)$. As we shall see in Section E.1.2, the advantage (in comparison to the Reed-Solomon code) is that codeword testing and self-correction can be performed at complexity related to $q = \text{poly}(\log n)$. Actually, in most complexity applications, a variant in which only m-variate polynomials of individual degree $r' = r/m$ are used. In this case, an alternative presentation analogous to the one presented in Footnote 2 is preferred: The information is viewed as a function $f : H^m \to \text{GF}(q)$, where $H \subset \text{GF}(q)$ is of size $r' + 1$, and is encoded by the evaluation at all points in $\text{GF}(q)^m$ of the m-variate polynomial of individual degree r' that extends the function f.

E.1.1.5 Binary codes of constant relative distance and constant rate

Recall that we seek binary codes of constant relative distance and constant rate. Proposition E.1 asserts that such codes exists, but does not provide an explicit construction. The Hadamard code is explicit but does not have a constant rate (to say the least (since $n(k) = 2^k$)).[3] The Reed-Solomon code has constant relative distance and constant rate but uses a non-binary alphabet (which grows at least linearly with k). We achieve the desired construction by using the paradigm of concatenated codes [74], which is of independent interest. (Indeed, concatenated codes may be viewed as a simple version of the proof composition paradigm presented in §9.3.2.2.)

Intuitively, concatenated codes are obtained by first encoding information, viewed as a sequence over a large alphabet, by some code and next encoding each resulting symbol, which is viewed as a sequence of over a smaller alphabet, by a second code. Formally, consider $\Sigma_1 \equiv \Sigma_2^{k_2}$ and two codes, $C_1 : \Sigma_1^{k_1} \to \Sigma_1^{n_1}$ and $C_2 : \Sigma_2^{k_2} \to \Sigma_2^{n_2}$. Then, the concatenated code of C_1 and C_2, maps $(x_1, ..., x_{k_1}) \in \Sigma_1^{k_1} \equiv \Sigma_2^{k_1 k_2}$ to $(C_2(y_1), ..., C_2(y_{n_1}))$, where $(y_1, ..., y_{n_1}) = C_1(x_1, ..., x_{k_1})$.

Note that the resulting code $C : \Sigma_2^{k_1 k_2} \to \Sigma_2^{n_1 n_2}$ has constant rate and constant relative distance if both C_1 and C_2 have these properties. Encoding in the concatenated code is straightforward. To decode a corrupted codeword of C, we view the input as an n_1-long sequence of blocks, where each block is an n_2-long sequence over Σ_2. Applying the decoder of C_2 to each block, we obtain n_1 sequences (each of length k_2) over Σ_2, and interpret each such sequence as a symbol of Σ_1. Finally, we apply the decoder of C_1 to the resulting n_1-long sequence (over Σ_1), and interpret the resulting k_1-long sequence (over Σ_1) as a $k_1 k_2$-long sequence over Σ_2. The key observation is that *if $w \in \Sigma_2^{n_1 n_2}$ is $\varepsilon_1\varepsilon_2$-close to $C(x_1, ..., x_{k_1}) = (C_2(y_1), ..., C_2(y_{n_1}))$ then at least $(1 - \varepsilon_1) \cdot n_1$ of the blocks of w are ε_2-close to the corresponding $C_2(y_i)$.*[4]

We are going to consider the concatenated code obtained by using the Reed-

[3] Binary Reed-Muller codes also fail to simultaneously provide constant relative distance and constant rate.

[4] This observation offers unique decoding from a fraction of errors that is the product of the fractions (of error) associated with the two original codes. Stronger statements regarding unique decoding of the concatenated code can be made based on more refined analysis (cf. [74]).

Solomon Code $C_1 : \mathrm{GF}(n_1)^{k_1} \to \mathrm{GF}(n_1)^{n_1}$ as the large code, setting $k_2 = \log_2 n_1$, and using the mildly explicit version of Proposition E.1, $C_2 : \{0,1\}^{k_2} \to \{0,1\}^{n_2}$ as the small code. We use $n_1 = 3k_1$ and $n_2 = O(k_2)$, and so the concatenated code is $C : \{0,1\}^k \to \{0,1\}^n$, where $k = k_1 k_2$ and $n = n_1 n_2 = O(k)$. The key observation is that C_2 can be constructed in $\exp(k_2)$-time, whereas here $\exp(k_2) = \mathrm{poly}(k)$. Furthermore, both encoding and decoding with respect to C_2 can be performed in time $\exp(k_2) = \mathrm{poly}(k)$. Thus, we get:

Theorem E.2 (an explicit good code): *There exists constants $\delta, \rho > 0$ and an explicit family of binary codes of rate ρ and relative distance at least δ. That is, there exists a polynomial-time* (encoding) *algorithm C such that $|C(x)| = |x|/\rho$* (for every x) *and a polynomial-time* (decoding) *algorithm D such that for every y that is $\delta/2$-close to some $C(x)$ it holds that $D(y) = x$. Furthermore, C is a linear code.*

The linearity of C is justified by using a Reed-Solomon code over the extension field $F = \mathrm{GF}(2^{k_2})$, and noting that this code induces a linear transformation over GF(2). Specifically, the value of a polynomial p over F at a point $\alpha \in F$ can be obtained as a linear transformation of the coefficient of p, when viewed as k_2-dimensional vectors over GF(2).

Relative distance approaching one half. Starting with a Reed-Solomon code of relative distance δ_1 and a smaller code C_2 of relative distance δ_2, we obtain a concatenated code of relative distance $\delta_1\delta_2$. Note that, for any constant $\delta_1 < 1$, there exists a Reed-Solomon code $C_1 : \mathrm{GF}(n_1)^{k_1} \to \mathrm{GF}(n_1)^{n_1}$ of relative distance δ_1 and constant rate (i.e., $1 - \delta_1$). Giving up on constant rate, we may start with a Reed-Solomon code of block-length $n_1(k_1) = \mathrm{poly}(k_1)$ and distance $n_1(k_1) - k_1$ over $[n_1(k_1)]$, and use a Hadamard code (encoding $[n_1(k_1)]$ by $\{0,1\}^{n_1(k_1)}$) in the role of the small code C_2. This yields a (concatenated) binary code of block length $n(k) = n_1(k)^2$ and distance $(n_1(k) - k) \cdot n_1(k)/2$. Thus, *the resulting explicit code has relative distance approximately* $(1/2) - (k/\sqrt{n(k)})$.

E.1.2 Two additional computational problems

In this section we briefly review relaxations of two traditional coding theoretic tasks. The purpose of these relaxations is enabling super-fast (randomized) algorithms that provide meaningful information. Specifically, these algorithms may run in sub-linear (e.g., poly-logarithmic) time, and thus cannot possibly solve the unrelaxed version of the problem.

Local testability. This task refers to testing whether a given word is a codeword (in a predetermine code), based on (randomly) inspecting few locations in the word. Needless to say, we can only hope to make an approximately correct decision; that is, accept each codeword and reject with high probability each word that is *far* from the code. (Indeed, this task is within the framework of property testing; see Section 10.1.2.)

Local decodability. Here the task is to recover a specified bit in the plaintext by (randomly) inspecting few locations in a mildly corrupted codeword. This task is somewhat related to the task of self-correction (i.e., recovering a specified bit in the codeword itself, by inspecting few locations in the mildly corrupted codeword).

Note that the Hadamard code is both locally testable and locally decodable as well as self-correctable (based on a constant number of queries into the word); these facts were demonstrated and extensively used in §9.3.2.1. However, the Hadamard code has an exponential block-length (i.e., $n(k) = 2^k$), and the question is whether one can achieve analogous results with respect to a shorter code (e.g., $n(k) = \text{poly}(k)$). As hinted in §E.1.1.4, the answer is positive (when we refer to performing these operations in time that is poly-logarithmic in k):

Theorem E.3 *For some constant $\delta > 0$ and polynomials $n, q : \mathbb{N} \to \mathbb{N}$, there exists an explicit family of codes $\{C_k : [q(k)]^k \to [q(k)]^{n(k)}\}_{k\in\mathbb{N}}$ of relative distance δ that can be locally testable and locally decodable in $\text{poly}(\log k)$-time. That is, the following three conditions hold.*

1. Encoding: *There exists a polynomial time algorithm that on input $x \in [q(k)]^k$ returns $C_k(x)$.*
2. Local Testing: *There exists a probabilistic polynomial-time oracle machine T that given k (in binary)[5] and oracle access to $w \in [q(k)]^{n(k)}$ distinguishes the case that w is a codeword from the case that w is $\delta/2$-far from any codeword. Specifically:*
   (a) *For every $x \in [q(k)]^k$ it holds that $\mathsf{Pr}[T^{C_k(x)}(k)=1] = 1$.*
   (b) *For every $w \in [q(k)]^{n(k)}$ that is $\delta/2$-far from any codeword of C_k it holds that $\mathsf{Pr}[T^w(k)=1] \leq 1/2$.*

   As usual, the error probability can be reduced by repetitions.
3. Local Decoding: *There exists a probabilistic polynomial-time oracle machine D that given k and $i \in [k]$ (in binary) and oracle access to any $w \in [q(k)]^{n(k)}$ that is $\delta/2$-close to $C_k(x)$ returns x_i; that is, $\mathsf{Pr}[D^w(k,i)=x_i] \geq 2/3$.*

   Self correction *holds too: there exists a probabilistic polynomial-time oracle machine M that given k and $i \in [n(k)]$ (in binary) and oracle access to any $w \in [q(k)]^{n(k)}$ that is $\delta/2$-close to $C_k(x)$ returns $C_k(x)_i$; that is, $\mathsf{Pr}[D^w(k,i)= C_k(x)_i] \geq 2/3$.*

We stress that all these oracle machines work in time that is polynomial in the binary representation of k, which means that they run in time that is poly-logarithmic in k. The code asserted in Theorem E.3 is a (small modification of a) Reed-Muller code, for $r = m^2 \log m < q(k) = \text{poly}(r)$ and $[n(k)] \equiv \text{GF}(q(k))^m$ (see §E.1.1.4).[6]

[5] Thus, the running time of T is $\text{poly}(|k|) = \text{poly}(\log k)$.

[6] The modification is analogous to the one presented in Footnote 2: For a suitable choice of k points $\overline{\alpha}_1, ..., \overline{\alpha}_k \in \text{GF}(q(k))^m$, we map $v_1, ..., v_k$ to $(p(\overline{\alpha}_1), ..., p(\overline{\alpha}_n))$, where p is the unique m-variate polynomial of degree at most r that satisfies $p(\overline{\alpha}_i) = v_i$ for $i = 1, ..., k$.

The aforementioned oracle machines query the oracle $w : [n(k)] \to \mathrm{GF}(q(k))$ at a non-constant number of locations. Specifically, self-correction for location $i \in \mathrm{GF}(q(k))^m$ is performed by selecting a random line (over $\mathrm{GF}(q(k))^m$) that passes through i, recovering the values assigned by w to all $q(k)$ points on this line, and performing univariate polynomial extrapolation (under mild noise). Local testability is easily reduced to self-correction, and (under the aforementioned modification) local decodability is a special case of self-correction.

Constant number of queries. The local testing and decoding algorithms asserted in Theorem E.3 make a polylogarithmic number of queries into the oracle. In contrast, the Hadamard code supports these operation using a *constant number of queries. Can this be obtained with much shorter codewords?* For local testability the answer is definitely positive. One can obtain such locally testable codes with length that is nearly linear (i.e., linear up to polylogarithmic factors; see [34, 63]). For local decodability based on a constant number of queries, the shortest known code has super-polynomial length (see [231]). In light of this state of affairs, we advocate a relaxation of the local decodability task (e.g., the one studied in [33]).

The interested reader is referred to [89], which includes more details on locally testable and decodable codes as well as a wider perspective. (Note, however, that this survey was written prior to [63] and [231], which address two major open problems discussed in [89].)

E.1.3 A list decoding bound

A necessary condition for the feasibility of the list decoding task is that the list of codewords that are close to the given word is short. In this section we present an upper-bound on the length of such lists, noting that this bound has found several applications in complexity theory (and specifically to studies related to the contents of this book). In contrast, we do not present far more famous bounds (which typically refer to the relation among the main parameters of codes (i.e., k, n and d)), because they seem irrelevant to the contents of this book.

We start with a general statement that refers to any alphabet $\Sigma \equiv [q]$, and later specialize it to the case that $q = 2$. Especially in the general case, it is natural and convenient to consider the agreement (rather than the distance) between sequences over $[q]$. Furthermore, it is natural to focus on agreement rate of at least $1/q$, and it is convenient to state the following result in terms of the "excessive agreement rate" (i.e., the excess beyond $1/q$).[7]

Lemma E.4 (Part 2 of [101, Thm. 15]): *Let $C : [q]^k \to [q]^n$ be an arbitrary code of distance $d \leq n - (n/q)$, and let $\eta_C \stackrel{\mathrm{def}}{=} (1 - (d/n)) - (1/q) \geq 0$ denote the corresponding upper-bound on the excessive agreement rate between codewords.*

[7] Indeed, we only consider codes with distance $d \leq (1 - 1/q) \cdot n$ and words that are at distance at most d from the code. Note that $1/q$ is a natural threshold for an upper-bound on the relative agreement between sequences over $[q]$, because a random sequence is expected to agree with any fixed sequence on a $1/q$ fraction of the locations.

Suppose that $\eta \in (0,1)$ *satisfies*

$$\eta > \sqrt{\left(1-\frac{1}{q}\right)\cdot\eta_{\mathrm{C}}} \tag{E.1}$$

Then, for any $w \in [q]^n$*, the number of codewords that agree with* w *on at least* $((1/q)+\eta)\cdot n$ *positions* (i.e., are at distance at most $(1-((1/q)+\eta))\cdot n$ from w) *is upper-bounded by*

$$\frac{(1-(1/q))^2-(1-(1/q))\cdot\eta_{\mathrm{C}}}{\eta^2-(1-(1/q))\cdot\eta_{\mathrm{C}}} \tag{E.2}$$

In the binary case (i.e., $q=2$), Eq. (E.1) requires $\eta > \sqrt{\eta_{\mathrm{C}}/2}$ and Eq. (E.2) yields the upper-bound $(1-2\eta_{\mathrm{C}})/(4\eta^2-2\eta_{\mathrm{C}})$. We highlight two specific cases:

1. At the end of §D.4.2.2, we refer to this bound (for the binary case) while setting $\eta_{\mathrm{C}} = (1/k)^2$ and $\eta = 1/k$. Indeed, in this case $(1-2\eta_{\mathrm{C}})/(4\eta^2-2\eta_{\mathrm{C}}) = O(k^2)$.

2. In the case of the Hadamard code, we have $\eta_{\mathrm{C}} = 0$. Thus, for every $w \in \{0,1\}^n$ and every $\eta > 0$, the number of codewords that are $(0.5-\eta)$-close to w is at most $1/(4\eta^2)$.

In the general case (and specifically for $q \gg 2$) it is useful to simplify Eq. (E.1) by $\eta > \min\{\sqrt{\eta_{\mathrm{C}}}, (1/q)+\sqrt{\eta_{\mathrm{C}}-(1/q)}\}$ and Eq. (E.2) by $\frac{1}{\eta^2-\eta_{\mathrm{C}}}$.

E.2 Expander Graphs

Loosely speaking, expander graphs are graphs of small degree that exhibit various properties of cliques. In particular, we refer to properties such as the relative sizes of cuts in the graph, and the rate at which a random walk converges to the uniform distribution (relative to the logarithm of the graph size to the base of its degree).

Some technicalities. Typical presentations of expander graphs refer to one of several variants. For example, in some sources, expanders are presented as bipartite graphs, whereas in others they are presented as ordinary graphs (and are in fact very far from being bipartite). We shall follow the latter convention. Furthermore, at times we implicitly consider an augmentation of these graphs where self-loops are added to each vertex. For simplicity, we also allow parallel edges.

We often talk of expander graphs while we actually mean an infinite collection of graphs such that each graph in this collection satisfies the same property (which is informally attributed to the collection). For example, when talking of a d-regular expander (graph) we actually refer to an infinite collection of graphs such that each of these graphs is d-regular. Typically, such a collection (or family) contains a single N-vertex graph for every $N \in \mathbb{S}$, where $\mathbb{S}$ is an infinite subset of $\mathbb{N}$. Throughout this section, we denote such a collection by $\{G_N\}_{N\in\mathbb{S}}$, with the understanding that G_N is a graph with N vertices and $\mathbb{S}$ is an infinite set of natural numbers.

E.2.1 Definitions and Properties

We consider two definitions of expander graphs, two different notions of explicit constructions, and two useful properties of expanders.

E.2.1.1 Two Mathematical Definitions

We start with two different definitions of expander graphs. These definitions are qualitatively equivalent and even quantitatively related. We start with an algebraic definition, which seems technical in nature but is actually the definition typically used in complexity theoretic applications, since it directly implies various "mixing properties" (see §E.2.1.3). We later present a very natural combinatorial definition (which is the source of the term "expander").

The algebraic definition (spectral gap). Identifying graphs with their adjacency matrix, we consider the eigenvalues (and eigenvectors) of a graph (or rather of its adjacency matrix). Any d-regular graph $G = (V, E)$ has the uniform vector as an eigenvector corresponding to the eigenvalue d, and if G is connected and not bipartite then (the absolute values of) all other eigenvalues are strictly smaller than d. The second eigenvalue, denoted $\lambda_2(G) < d$, of such a graph G is thus a tight upper-bound on the *absolute value* of all the other eigenvalues. Using the connection to the combinatorial definition, it follows that $\lambda_2(G) < d - \Omega(1/|V|^2)$ holds (for every connected non-bipartite d-regular graph G). The algebraic definition of expanders refers to an infinite family of d-regular graphs and requires the existence of a *constant* eigenvalue bound that holds for all the graphs in the family.

Definition E.5 *An infinite family of d-regular graphs, $\{G_N\}_{N\in\mathbb{S}}$, where $\mathbb{S} \subseteq \mathbb{N}$,* satisfies the eigenvalue bound *λ if for every $N \in \mathbb{S}$ it holds that $\lambda_2(G_N) \leq \lambda$.*

In such a case we say that the family has spectral gap $d - \lambda$. It will be often convenient to consider relative (or normalized) versions of these quantities, obtained by division by d.

The combinatorial definition (expansion). Loosely speaking, expansion requires that any (not too big) set of vertices of the graph has a relatively large set of neighbors. Specifically, a graph $G = (V, E)$ is c-expanding if, for every set $S \subset V$ of cardinality at most $|V|/2$, it holds that

$$\Gamma_G(S) \stackrel{\text{def}}{=} \{v : \exists u \in S \text{ s.t. } \{u, v\} \in E\} \tag{E.3}$$

has cardinality at least $(1 + c) \cdot |S|$. Equivalently (assuming the existence of self-loops on all vertices), we may require that $|\Gamma_G(S) \setminus S| \geq c \cdot |S|$. Clearly, every connected graph $G = (V, E)$ is $(1/|V|)$-expanding. The combinatorial definition of expanders refers to an infinite family of d-regular graphs and requires the existence of a *constant* expansion bound that holds for all the graphs in the family.

Definition E.6 *An infinite family of d-regular graphs, $\{G_N\}_{N\in\mathbb{S}}$* is c-expanding *if for every $N \in \mathbb{S}$ it holds that G_N is c-expanding.*

The two definitions of expander graphs are related (see [10, Sec. 9.2] or [120, Sec. 4.5]).

Theorem E.7 *Let G be a non-bipartite d-regular graph.*

1. *The graph G is c-expanding for $c \geq (d - \lambda_2(G))/2d$.*
2. *If G is c-expanding then $d - \lambda_2(G) \geq c^2/(4 + 2c^2)$.*

Thus, any non-zero bound on the combinatorial expansion of a family of d-regular graphs yields a non-zero bound on its spectral gap, and vice versa. Note, however, that the back-and-forth translation between these definitions is not tight. The applications presented in the main text refer to the algebraic definition, and the loss incurred in Theorem E.7 is immaterial for them.

Amplification. The quality of expander graphs improves by raising them to any power $t > 1$ (i.e., raising their adjacency matrix to the t^{th} power), which corresponds to considering graphs in which t-paths are replaced by edges. Using the algebraic definition, we have $\lambda_2(G^t) = \lambda_2(G)^t$, but indeed the degree also gets raised to the power t. Still, the ratio $\lambda_2(G^t)/d^t$ deceases with t. An analogous phenomenon occurs also under the combinatorial definition, provided that some suitable modifications are applied. For example, if $G = (V, E)$ is c-expanding (i.e., for every $S \subseteq V$ it holds that $|\Gamma_G(S)| \geq \min((1 + c) \cdot |S|, |V|/2)$), then for every $S \subseteq V$ it holds that $|\Gamma_{G^t}(S)| \geq \min((1 + c)^t \cdot |S|, |V|/2)$.

The optimal eigenvalue bound. For every d-regular graph $G = (V, E)$, it holds that $\lambda_2(G) \geq 2\gamma_G \cdot \sqrt{d - 1}$, where $\gamma_G = 1 - O(1/\log_d |V|)$. Thus, $2\sqrt{d - 1}$ is a lower-bound on the eigenvalue bound of any infinite family of d-regular graphs.

E.2.1.2 Two levels of explicitness

A mild level of explicit constructiveness refers to the complexity of constructing the entire object (i.e., graph). Thus, an infinite family of graphs $\{G_N\}_{N \in \mathbb{S}}$ is said to be explicitly constructible if there exists a *polynomial-time algorithm that, on input* 1^N (where $N \in \mathbb{S}$), *outputs the list of the edges in the N-vertex graph G_N*.

The aforementioned level of explicitness suffices when the application requires holding the entire graph and/or when the running-time of the application is lower-bounded by the size of the graph. In contrast, other applications only refer to a huge virtual graph (which is much bigger than their running time), and only require the computation of the neighborhood relations in such a graph. In this case, the following stronger level of explicitness is relevant.

A strongly explicit construction of an infinite family of (d-regular) graphs $\{G_N\}_{N \in \mathbb{S}}$ is a *polynomial-time algorithm that on input N* (in binary), *a vertex v in the N-vertex graph G_N and an index i ($i \in \{1, ..., d\}$), returns the i^{th} neighbor of v*. That is, the neighbor is determined in time that is polylogarithmic in the size of the graph. Needless to say, the strong level of explicitness implies the basic level.

An additional requirement, which is often forgotten but is very important, refers to the "tractability" of the set $\mathbb{S}$. Specifically, we require the existence of an *efficient algorithm that given any $n \in \mathbb{N}$ finds an $s \in \mathbb{S}$ such that $n \leq s < 2n$*. Corresponding to the foregoing definitions, efficient may mean either running in time $\text{poly}(n)$ or running in time $\text{poly}(\log n)$. The requirement that $n \leq s < 2n$ suffices in most applications, but in some cases a smaller interval (e.g., $n \leq s < n+\sqrt{n}$) is required, whereas in other cases a larger interval (e.g., $n \leq s < \text{poly}(n)$) suffices.

Greater flexibility. In continuation to the foregoing paragraph, we comment that expanders can be combined in order to obtain expanders for a wider range of sizes. For example, two d-regular c-expanding graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where $|V_1| \leq |V_2|$ and $c \leq 1$, can be combined into a $(d+1)$-regular $c/2$-expanding graph on $|V_1|+|V_2|$ vertices by connecting the two graphs with a perfect matching of V_1 and $|V_1|$ of the vertices of V_2 (and adding self-loops to the remaining vertices of V_2). More generally, the d-regular c-expanding graphs, $G_1 = (V_1, E_1)$ through $G_t = (V_t, E_t)$, where $N \stackrel{\text{def}}{=} \sum_{i=1}^{t-1} |V_i| \leq |V_t|$, yield a $(d+1)$-regular $c/2$-expanding graph on $\sum_{i=1}^{t} |V_i|$ vertices by using a perfect matching of $\cup_{i=1}^{t-1} V_i$ and N of the vertices of V_t.

E.2.1.3 Two properties

The following two properties provide a quantitative interpretation to the statement that expanders approximate the complete graph. The deviation from the latter is represented by an error term that is linear in λ/d.

The mixing lemma. The following lemma is folklore and has appeared in many papers. Loosely speaking, the lemma asserts that expander graphs (for which $d \gg \lambda$) have the property that the fraction of edges between two large sets of vertices approximately equals the product of the densities of these sets. This property is called *mixing*.

Lemma E.8 (Expander Mixing Lemma): *For every d-regular graph $G = (V, E)$ and for every two subsets $A, B \subseteq V$ it holds that*

$$\left| \frac{|(A \times B) \cap E_2|}{|E_2|} - \frac{|A|}{|V|} \cdot \frac{|B|}{|V|} \right| \leq \frac{\lambda_2(G)\sqrt{|A| \cdot |B|}}{d \cdot |V|} \leq \frac{\lambda_2(G)}{d} \tag{E.4}$$

where E_2 denotes the set of directed edges that correspond to the undirected edges of G (i.e., $E_2 = \{(u,v) : \{u,v\} \in E\}$ and $|E_2| = d|V|$).

Proof: Let $N \stackrel{\text{def}}{=} |V|$ and $\lambda \stackrel{\text{def}}{=} \lambda_2(G)$. For any subset of the vertices $S \subseteq V$, we denote its density in V by $\rho(S) \stackrel{\text{def}}{=} |S|/N$. Hence, Eq. (E.4) is restated as

$$\left| \frac{|(A \times B) \cap E_2|}{d \cdot N} - \rho(A) \cdot \rho(B) \right| \leq \frac{\lambda\sqrt{\rho(A) \cdot \rho(B)}}{d} .$$

We proceed by providing bounds on the value of $|(A \times B) \cap E_2|$. To this end we let $\overline{a}$ denote the N-dimensional Boolean vector having 1 in the i^{th} component if and only if $i \in A$. The vector $\overline{b}$ is defined similarly. Denoting the adjacency matrix of the graph G by $M = (m_{i,j})$, we note that $|(A \times B) \cap E_2|$ equals $\overline{a}^\top M \overline{b}$ (because $(i,j) \in (A \times B) \cap E_2$ if and only if it holds that $i \in A$, $j \in B$ and $m_{i,j} = 1$). We consider the *orthogonal eigenvector basis*, $\overline{e_1}, ..., \overline{e_N}$, where $\overline{e_1} = (1, ..., 1)^\top$ and $\overline{e_i}^\top \overline{e_i} = N$ for each i, and write each vector as a linear combination of the vectors in this basis. Specifically, we denote by a_i the coefficient of $\overline{a}$ in the direction of $\overline{e_i}$; that is, $a_i = (\overline{a}^\top \overline{e_i})/N$ and $\overline{a} = \sum_i a_i \overline{e_i}$. Note that $a_1 = (\overline{a}^\top \overline{e_1})/N = |A|/N = \rho(A)$ and $\sum_{i=1}^N a_i^2 = (\overline{a}^\top \overline{a})/N = |A|/N = \rho(A)$. Similarly for $\overline{b}$. It now follows that

$$\begin{aligned} |(A \times B) \cap E_2| &= \overline{a}^\top M \left(b_1 \overline{e_1} + \sum_{i=2}^N b_i \overline{e_i} \right) \\ &= \rho(B) \cdot \overline{a}^\top M \overline{e_1} + \sum_{i=2}^N b_i \cdot \overline{a}^\top M \overline{e_i} \\ &= \rho(B) \cdot d \cdot \overline{a}^\top \overline{e_1} + \sum_{i=2}^N b_i \lambda_i \cdot \overline{a}^\top \overline{e_i} \end{aligned}$$

where λ_i denotes the i^{th} eigenvalue of M (and indeed $\lambda_1 = d$). Thus,

$$\begin{aligned} \frac{|(A \times B) \cap E_2|}{dN} &= \rho(B)\rho(A) + \sum_{i=2}^N \frac{\lambda_i b_i a_i}{d} \\ &\in \left[\rho(B)\rho(A) \pm \frac{\lambda}{d} \cdot \sum_{i=2}^N a_i b_i \right] \end{aligned}$$

Using $\sum_{i=1}^N a_i^2 = \rho(A)$ and $\sum_{i=1}^N b_i^2 = \rho(B)$, and applying Cauchy-Schwartz Inequality, we bound $\sum_{i=2}^N a_i b_i$ by $\sqrt{\rho(A)\rho(B)}$. The lemma follows. ■

The random walk lemma. Loosely speaking, the first part of the following lemma asserts that, as far as remaining trapped in some subset of the vertex set is concerned, a random walk on an expander approximates a random walk on the complete graph.

Lemma E.9 (Expander Random Walk Lemma): *Let $G = ([N], E)$ be a d-regular graph, and consider walks on G that start from a uniformly chosen vertex and take $\ell - 1$ additional random steps, where in each such step we uniformly selects one out of the d edges incident at the current vertex and traverses it.*

Theorem 8.28 (restated): *Let W be a subset of $[N]$ and $\rho \stackrel{\text{def}}{=} |W|/N$. Then the probability that such a random walk stays in W is at most*

$$\rho \cdot \left(\rho + (1 - \rho) \cdot \frac{\lambda_2(G)}{d} \right)^{\ell - 1} \tag{E.5}$$

Exercise 8.38 (restated): *For any $W_0, ..., W_{\ell-1} \subseteq [N]$, the probability that a random walk of length ℓ intersects $W_0 \times W_1 \times \cdots \times W_{\ell-1}$ is at most*

$$\sqrt{\rho_0} \cdot \prod_{i=1}^{\ell-1} \sqrt{\rho_i + (\lambda/d)^2}, \tag{E.6}$$

where $\rho_i \stackrel{\text{def}}{=} |W_i|/N$.

The basic principle underlying Lemma E.9 was discovered by Ajtai, Komlos, and Szemerédi [4], who proved a bound as in Eq. (E.6). The better analysis yielding Theorem 8.28 is due to Kahale [129, Cor. 6.1]. A more general bound that refer to the probability of visiting W for a number of times that approximates $|W|/N$ is given in [116], which actually considers an even more general problem (i.e., obtaining Chernoff-type bounds for random variables that are generated by a walk on an expander).

Proof of Equation (E.6): The basic idea is to view the random walk as the evolution of a corresponding probability vector under suitable transformations. The transformations correspond to taking a random step in G and to passing through a "sieve" that keeps only the entries that correspond to the current set W_i. The key observation is that the first transformation shrinks the component that is orthogonal to the uniform distribution, whereas the second transformation shrinks the component that is in the direction of the uniform distribution. Details follow.

Let A be a matrix representing the random walk on G (i.e., A is the adjacency matrix of G divided by d), and let $\hat{\lambda}$ denote the absolute value of the second largest eigenvalue of A (i.e., $\hat{\lambda} \stackrel{\text{def}}{=} \lambda_2(G)/d$). Note that the uniform distribution, represented by the vector $\overline{u} = (N^{-1}, ..., N^{-1})^\top$, is the eigenvector of A that is associated with the largest eigenvalue (which is 1). Let P_i be a 0-1 matrix that has 1-entries only on its diagonal, and furthermore entry (j,j) is set to 1 if and only if $j \in W_i$. Then, the probability that a random walk of length ℓ intersects $W_0 \times W_1 \times \cdots \times W_{\ell-1}$ is the sum of the entries of the vector

$$\overline{v} \stackrel{\text{def}}{=} P_{\ell-1} A \cdots P_2 A P_1 A P_0 \overline{u}. \tag{E.7}$$

We are interested in upper-bounding $\|\overline{v}\|_1$, and use $\|\overline{v}\|_1 \leq \sqrt{N} \cdot \|\overline{v}\|$, where $\|\overline{z}\|_1$ and $\|\overline{z}\|$ denote the L_1-norm and L_2-norm of $\overline{z}$, respectively (e.g., $\|\overline{u}\|_1 = 1$ and $\|\overline{u}\| = N^{-1/2}$). The key observation is that the linear transformation $P_i A$ shrinks every vector.

Main Claim. For every $\overline{z}$, it holds that $\|P_i A \overline{z}\| \leq (\rho_i + \hat{\lambda}^2)^{1/2} \cdot \|\overline{z}\|$.

Proof. Intuitively, A shrinks the component of $\overline{z}$ that is orthogonal to $\overline{u}$, whereas P_i shrinks the component of $\overline{z}$ that is in the direction of $\overline{u}$. Specifically, we decompose $\overline{z} = \overline{z_1} + \overline{z_2}$ such that $\overline{z_1}$ is the projection of $\overline{z}$ on $\overline{u}$ and $\overline{z_2}$ is the component orthogonal to $\overline{u}$. Then, using the triangle inequality and other obvious facts (which

imply $\|P_iA\overline{z_1}\| = \|P_i\overline{z_1}\|$ and $\|P_iA\overline{z_2}\| \leq \|A\overline{z_2}\|$), we have

$$\begin{aligned}\|P_iA\overline{z_1} + P_iA\overline{z_2}\| &\leq \|P_iA\overline{z_1}\| + \|P_iA\overline{z_2}\| \\ &\leq \|P_i\overline{z_1}\| + \|A\overline{z_2}\| \\ &\leq \sqrt{\rho_i}\cdot\|\overline{z_1}\| + \hat{\lambda}\cdot\|\overline{z_2}\|\end{aligned}$$

where the last inequality uses the fact that P_i shrinks any uniform vector by eliminating $1-\rho_i$ of its elements, whereas A shrinks the length of any eigenvector except $\overline{u}$ by a factor of at least $\hat{\lambda}$. Using the Cauchy-Schwartz inequality[8], we get

$$\begin{aligned}\|P_iA\overline{z}\| &\leq \sqrt{\rho_i+\hat{\lambda}^2}\cdot\sqrt{\|\overline{z_1}\|^2+\|\overline{z_2}\|^2} \\ &= \sqrt{\rho_i+\hat{\lambda}^2}\cdot\|\overline{z}\|\end{aligned}$$

where the equality is due to the fact that $\overline{z_1}$ is orthogonal to $\overline{z_2}$. □

Recalling Eq. (E.7) and using the Main Claim (and $\|\overline{v}\|_1 \leq \sqrt{N}\cdot\|\overline{v}\|$), we get

$$\begin{aligned}\|\overline{v}\|_1 &\leq \sqrt{N}\cdot\|P_{\ell-1}A\cdots P_2AP_1AP_0\overline{u}\| \\ &\leq \sqrt{N}\cdot\left(\prod_{i=1}^{\ell-1}\sqrt{\rho_i+\hat{\lambda}^2}\right)\cdot\|P_0\overline{u}\|.\end{aligned}$$

Finally, using $\|P_0\overline{u}\| = \sqrt{\rho_0 N\cdot(1/N)^2} = \sqrt{\rho_0/N}$, we establish Eq. (E.6). ■

Rapid mixing. A property related to Lemma E.9 is that a random walk starting at any vertex converges to the uniform distribution on the expander vertices after a logarithmic number of steps. Using notation as in the proof of Eq. (E.6), we *claim that for every starting distribution* $\overline{s}$ (including one that assigns all weight to a single vertex), *it holds that* $\|A^\ell\overline{s}-\overline{u}\|_1 \leq \sqrt{N}\cdot\hat{\lambda}^\ell$, which is meaningful for any $\ell > 0.5\cdot\log_{1/\hat{\lambda}} N$. The claim is proved by recalling that $\|A^\ell\overline{s}-\overline{u}\|_1 \leq \sqrt{N}\cdot\|A^\ell\overline{s}-\overline{u}\|$ and using the fact that $\overline{s}-\overline{u}$ is orthogonal to $\overline{u}$ (because the former is a zero-sum vector). Thus, $\|A^\ell\overline{s}-\overline{u}\| = \|A^\ell(\overline{s}-\overline{u})\| \leq \hat{\lambda}^\ell\|\overline{s}-\overline{u}\|$ and using $\|\overline{s}-\overline{u}\| < 1$ the claim follows.

E.2.2 Constructions

Many explicit constructions of expanders were discovered, starting in [157] and culminating in the optimal construction of [153] where $\lambda = 2\sqrt{d-1}$. Most of these constructions are quite simple (see, e.g., §E.2.2.1), but their analysis is based on non-elementary results from various branches of mathematics. In contrast, the construction of Reingold, Vadhan, and Wigderson [184], presented in §E.2.2.2,

[8] That is, we get $\sqrt{\rho_i}\|z_1\| + \hat{\lambda}\|z_2\| \leq \sqrt{\rho_i+\hat{\lambda}^2}\cdot\sqrt{\|z_1\|^2+\|z_2\|^2}$, by using $\sum_{i=1}^n a_i\cdot b_i \leq \left(\sum_{i=1}^n {a_i}^2\right)^{1/2}\cdot\left(\sum_{i=1}^n {b_i}^2\right)^{1/2}$, with $n=2$, $a_1=\sqrt{\rho_i}$, $b_1=\|z_1\|$, etc.

is based on an iterative process, and its analysis is based on a relatively simple algebraic fact regarding the eigenvalues of matrices.

Before turning to these explicit constructions we note that it is relatively easy to prove the existence of 3-regular expanders, by using the Probabilistic Method (cf. [10]) and referring to the combinatorial definition of expansion.

Theorem E.10 *For some constant $\lambda < 3$ there exists a family of $(3,\lambda)$-expanders for any even graph size.*

Proof Sketch:[9] As a warm-up, one may establish the existence of d-regular expanders, for some constant d. In particular, foreseeing the case of $d = 3$, consider a random graph G on the vertex set $V = \{0,...,n-1\}$ constructed by augmenting the fixed edge set $\{\{i, i+1 \bmod n\} : i = 0,...,n-1\}$ with $d-2$ uniformly (and independently) chosen perfect matchings of the vertices of $F \stackrel{\text{def}}{=} \{0,...,(n/2)-1\}$ to the vertices of $L \stackrel{\text{def}}{=} \{n/2,...,n-1\}$. For a sufficiently small universal constant $\varepsilon > 0$, we upper-bound the probability that such a random graph is not ε-expanding. Noting that for every set S it holds that $|\Gamma_G(S\cap F)\cap F| \geq |S\cap F|-1$ (and similarly for L), we focus on the sizes of $|(\Gamma_G(S \cap F) \cap L) \setminus \Gamma_G(S \cap L)|$ and $|(\Gamma_G(S \cap L) \cap F) \setminus \Gamma_G(S \cap F)|$. Assuming without loss of generality that $|S \cap F| \geq |S \cap L|$, we upper-bound the probability that there exists a set $S \subset V$ of size at most $n/2$ such that $|(\Gamma_G(S \cap F) \cap L) \setminus \Gamma_G(S \cap L)| < \varepsilon|S|$. Fixing a set S, the corresponding probability is upper-bounded by p_S^{d-2}, where p_S denotes the probability that a uniformly selected matching of F to L matches $S \cap F$ to a set that contains less than $\varepsilon|S|$ elements in $L \setminus \Gamma_G(S \cap L)$. That is,

$$p_S \stackrel{\text{def}}{=} \sum_{i=0}^{\varepsilon|S|-1} \frac{\binom{|L|-\ell}{i} \cdot \binom{\ell}{|S\cap F|-i}}{\binom{|L|}{|S\cap F|}} \leq \frac{\binom{(n/2)-\ell}{\varepsilon|S|} \cdot \binom{\ell+\varepsilon|S|}{|S\cap F|}}{\binom{n/2}{|S\cap F|}}$$

where $\ell = |\Gamma_G(S\cap L)\cap L|$. Indeed, we may focus on the case that $|S\cap F| \leq \ell+\varepsilon|S|$ (because in the other case $p_S = 0$), and observe that for every $\alpha < 1/2$ there exists a sufficiently small $\varepsilon > 0$ such that $p_S < \binom{n}{|S|}^{-\alpha}$. The claim follows for $d \geq 5$, by using a union bound on all sets (and setting $\alpha = 1/3$).

To deal with the case $d = 3$, we use a more sophisticated union bound. Specifically, fixing an adequate constant $t > 6$ (e.g., $t = 1/\sqrt{\varepsilon}$), we decompose S into S' and S'', where S' contains the elements of S that reside on t-long arithmetic subsequences of S that use an step increment of either 1 or 2, and $S'' = S \setminus S'$. It can be shown that $|\Gamma_G(S'') \setminus S| > |S''|/2t$ (hint: an arithmetic subsequence has neighborhood greater than itself whereas a suitable partition of the elements to such subsequences guarantees that the overall excess is at least half the individual

[9] The proof is much simpler in the case that one refers to the alternative definition of combinatorial expansion in which for each relevant set S it holds that $|\Gamma_G(S) \setminus S| \geq \varepsilon \cdot |S|$. In this case, for a sufficiently small $\varepsilon > 0$ and all sufficiently large n, a random 3-regular n-vertex graph is ε-expanding with overwhelmingly high probability. The proof proceeds by considering a (not necessarily simple) graph G generated by three perfect matchings of the elements of $[n]$. For every $S \subseteq [n]$ of size at most $n/2$ and for every set T of size $\varepsilon|S|$, we consider the probability that $\Gamma_G(S) \subseteq S \cup T$. The argument is concluded by applying a union bound.

count). Thus, if $|S''| > 2|S|/t$ then $|\Gamma_G(S'') \setminus S| > |S|/t^2$. Hence, it suffices to consider the case $|S''| \leq 2|S|/t$ (and $t \geq 6$) and prove that $|\Gamma_G(S')| > (1+(4/t)) \cdot |S'|$. The gain is that, when applying the union bound, it suffices to consider less than $\sum_{j=1}^{n'/t} 2^j \cdot \binom{n}{2j} < \binom{n}{3n'/t}$ possible sets S' of size n', which are each a union of at most n'/t arithmetic sequences that use an step increment of either 1 or 2. □

E.2.2.1 The Margulis–Gabber–Galil Expander

For every natural number m, consider the graph with vertex set $\mathbb{Z}_m \times \mathbb{Z}_m$ and edge set in which every $\langle x, y \rangle \in \mathbb{Z}_m \times \mathbb{Z}_m$ is connected to the vertices $\langle x \pm y, y \rangle$, $\langle x \pm (y+1), y \rangle$, $\langle x, y \pm x \rangle$, and $\langle x, y \pm (x+1) \rangle$, where the arithmetic is modulo m. This yields an extremely simple and explicit 8-regular graph with second eigenvalue that is bounded by a constant $\lambda < 8$ that is independent of m. Thus we get:

Theorem E.11 *For some constant $\lambda < 8$ there exists a strongly explicit construction of a family of $(8, \lambda)$-expanders for graph sizes $\{m^2 : m \in \mathbb{N}\}$. Furthermore, the neighbors of a vertex can be computed in logarithmic-space.*[10]

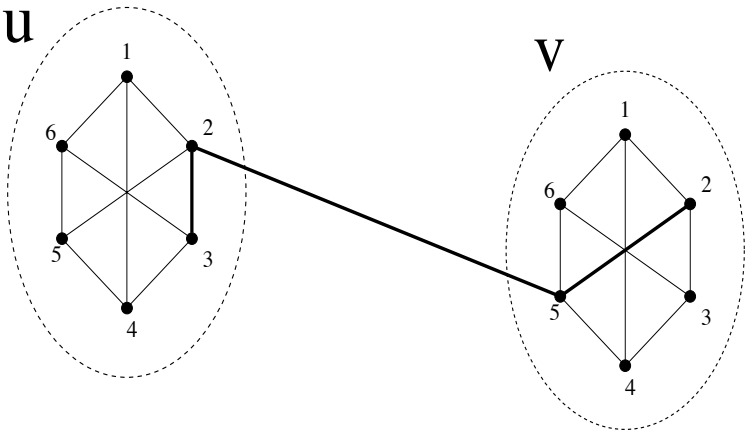
An appealing property of Theorem E.11 is that, for every $n \in \mathbb{N}$, it directly yields expanders with vertex set $\{0,1\}^n$. This is obvious in case n is even, but can be easily achieved also for odd n (e.g., use two copies of the graph for $n-1$, and connect the two copies by the obvious perfect matching).

Theorem E.11 is due to Gabber and Galil [80], building on the basic approach suggested by Margulis [157]. We mention again that the optimal construction of [153] achieves $\lambda = 2\sqrt{d-1}$, but there are annoying restrictions on the degree d (i.e., $d-1$ should be a prime congruent to 1 modulo 4) and on the graph sizes for which this construction works.

E.2.2.2 The Iterated Zig-Zag Construction

The starting point of the following construction is a very good expander G of *constant size*, which may be found by an exhaustive search. The construction of a large expander graph proceeds in iterations, where in the i^{th} iteration the current graph G_i and the fixed graph G are combined, resulting in a larger graph G_{i+1}. The combination step guarantees that the expansion property of G_{i+1} is at least as good as the expansion of G_i, while G_{i+1} maintains the degree of G_i and is a constant times larger than G_i. The process is initiated with $G_1 = G^2$ and terminates when we obtain a graph G_t of approximately the desired size (which requires a logarithmic number of iterations).

[10] In fact, under a suitable encoding of the vertices and for m that is a power of two, the neighbors can be computed by a on-line algorithm that uses a constant amount of space. The same holds also for a variant in which each vertex $\langle x, y \rangle$ is connected to the vertices $\langle x \pm 2y, y \rangle$, $\langle x \pm (2y+1), y \rangle$, $\langle x, y \pm 2x \rangle$, and $\langle x, y \pm (2x+1) \rangle$. (This variant yields a better known bound on λ, i.e., $\lambda \leq 5\sqrt{2} \approx 7.071$.)



In this example G' is 6-regular and G is a 3-regular graph having six vertices. In the graph G' (not shown), the 2nd edge of vertex u is incident at v, as its 5th edge. The wide 3-segment line shows one of the corresponding edges of $G'ⓩG$, which connects the vertices $\langle u,3\rangle$ and $\langle v,2\rangle$.

Figure E.1: Detail of the zig-zag product of G' and G.

The Zig-Zag product. The heart of the combination step is a new type of "graph product" called *Zig-Zag product*. This operation is applicable to any pair of graphs $G = ([D], E)$ and $G' = ([N], E')$, provided that G' (which is typically larger than G) is D-regular. For simplicity, we assume that G is d-regular (where typically $d \ll D$). The Zig-Zag product of G' and G, denoted $G'ⓩG$, is defined as a graph with vertex set $[N] \times [D]$ and an edge set that includes an edge between $\langle u, i\rangle \in [N] \times [D]$ and $\langle v, j\rangle$ if and only if $(i,k), (\ell, j) \in E$ and the k^{th} edge incident at u equals the ℓ^{th} edge incident at v. See Figure E.1 as well as further clarification that follows.

Teaching note: The following paragraph, which provides a formal description of the zig-zag product, can be ignored in first reading but is useful for more advanced discussion.

It will be convenient to represent graphs like G' by their edge rotation function, denoted $R' : [N]\times[D] \to [N]\times[D]$, such that $R'(u,i) = (v,j)$ if (u,v) is the i^{th} edge incident at u as well as the j^{th} edge incident at v. For simplicity, we assume that G is edge-colorable with d colors, which in turn yields a natural edge rotation function (i.e., $R(i,\alpha) = (j,\alpha)$ if the edge (i,j) is colored α). We will denote by $E_\alpha(i)$ the vertex reached from $i \in [D]$ by following the edge colored α (i.e., $E_\alpha(i) = j$ iff $R(i,\alpha) = (j,\alpha)$). The Zig-Zag product of G' and G, denoted $G'ⓩG$, is then defined as a graph with the vertex set $[N] \times [D]$ and the edge rotation function

$$(\langle u,i\rangle, \langle \alpha,\beta\rangle) \mapsto (\langle v,j\rangle, \langle \beta,\alpha\rangle) \quad \text{if } R'(u, E_\alpha(i)) = (v, E_\beta(j)). \tag{E.8}$$

That is, edges are labeled by pairs over $[d]$, and the $\langle\alpha,\beta\rangle^{\text{th}}$ edge out of vertex $\langle u,i\rangle \in [N]\times[D]$ is incident at the vertex $\langle v,j\rangle$ (as its $\langle\beta,\alpha\rangle^{\text{th}}$ edge) if $R(u,E_\alpha(i)) = (v,E_\beta(j))$. (That is, based on $\langle\alpha,\beta\rangle$, we take a G-step from $\langle u,i\rangle$ to $\langle u,E_\alpha(i)\rangle$, then viewing $\langle u,E_\alpha(i)\rangle \equiv (u,E_\alpha(i))$ as an edge of G' we rotate it to $(v,j') \stackrel{\text{def}}{=} R'(u,E_\alpha(i))$, and take a G-step from $\langle v,j'\rangle$ to $\langle v,E_\beta(j')\rangle$, while defining $j = E_\beta(j')$ and using $j' = E_\beta(E_\beta(j')) = E_\beta(j)$.)

Clearly, the graph $G'ⓩG$ is d^2-regular and has $D\cdot N$ vertices. The key fact, proved in [184] (using techniques as in §E.2.1.3), is that the relative eigenvalue of the zig-zag product is upper-bounded by the sum of the relative eigenvalues of the two graphs (i.e., $\bar\lambda_2(G'ⓩG) \leq \bar\lambda_2(G') + \bar\lambda_2(G)$, where $\bar\lambda_2(\cdot)$ denotes the relative eigenvalue of the relevant graph). The (qualitative) fact that $G'ⓩG$ is an expander if both G' and G are expanders is very intuitive (e.g., consider what happens if G' or G is a clique). Things are even more intuitive if one considers the (related) replacement product of G' and G, denoted $G'ⓡG$, *where there is an edge between* $\langle u,i\rangle \in [N]\times[D]$ *and* $\langle v,j\rangle$ *if and only if either* $u = v$ *and* $(i,j) \in E$ *or the* i^{th} *edge incident at* u *equals the* j^{th} *edge incident at* v.[11]

The iterated construction. The iterated expander construction uses the aforementioned zig-zag product as well as graph squaring. Specifically, the construction starts with the d^2-regular graph $G_1 = G^2 = ([D],E^2)$, where $D = d^4$ and $\bar\lambda_2(G) < 1/4$, and proceeds in iterations such that $G_{i+1} = G_i^2ⓩG$ for $i = 1,2,...,t-1$. That is, in each iteration, the current graph is first squared and then composed with the fixed (d-regular D-vertex) graph G via the zig-zag product. This process maintains the following two invariants:

1. The graph G_i is d^2-regular and has D^i vertices.

   (The degree bound follows from the fact that a zig-zag product with a d-regular graph always yields a d^2-regular graph.)

2. The relative eigenvalue of G_i is smaller than one half.

   (Here we use the fact that $\bar\lambda_2(G_{i-1}^2ⓩG) \leq \bar\lambda_2(G_{i-1}^2) + \bar\lambda_2(G)$, which in turn equals $\bar\lambda_2(G_{i-1})^2 + \bar\lambda_2(G) < (1/2)^2 + (1/4)$. Note that graph squaring is used to reduce the relative eigenvalue of G_i before increasing it by zig-zag product with G.)

To ensure that we can construct G_i, we should show that we can actually construct the edge rotation function that correspond to its edge set. This boils down to showing that, given the edge rotation function of G_{i-1}, we can compute the edge rotation function of G_{i-1}^2 as well as of its zig-zag product with G. Note that this computation amounts to two recursive calls to computations regarding G_{i-1} (and two computations that correspond to the constant graph G). But since the recursion depth is logarithmic in the size of the final graph, the time spend in the recursive computation is polynomial in the size of the final graph. This suffices for the minimal notion of explicitness, but not for the stronger one.

[11] As an exercise, the reader is encouraged to show that if both G' and G are expanders according to the combinatorial definition then so is $G'ⓡG$.

The strongly explicit version. To achieve a *strongly explicit construction*, we slightly modify the iterative construction. Rather than letting $G_{i+1} = G_i^2 ⓩ G$, we let $G_{i+1} = (G_i \times G_i)^2 ⓩ G$, where $G' \times G'$ denotes the *tensor product of G' with itself*; that is, if $G' = (V', E')$ then $G' \times G' = (V' \times V', E'')$, where

$$E'' = \{(\langle u_1, u_2\rangle, \langle v_1, v_2\rangle) : (u_1, v_1), (u_2, v_2) \in E'\}$$

with an edge rotation function

$$R''(\langle u_1, u_2\rangle, \langle i_1, i_2\rangle) = (\langle v_1, v_2\rangle, \langle j_1, j_2\rangle)$$

where $R'(u_1, i_1) = (v_1, j_1)$ and $R'(u_2, i_2) = (v_2, j_2)$. (We still use $G_1 = G^2$.) Using the fact that tensor product preserves the relative eigenvalue (while squaring the degree) and using a d-regular $G = ([D], E)$ with $D = d^8$, we note that the modified $G_i = (G_{i-1} \times G_{i-1})^2 ⓩ G$ is a d^2-regular graph with $(D^{2^{i-1}-1})^2 \cdot D = D^{2^i-1}$ vertices, and $\bar{\lambda}_2(G_i) < 1/2$ (because $\bar{\lambda}_2((G_{i-1} \times G_{i-1})^2 ⓩ G) \leq \bar{\lambda}_2(G_{i-1})^2 + \bar{\lambda}_2(G)$). Computing the neighbor of a vertex in G_i boils down to a constant number of such computations regarding G_{i-1}, but due to the tensor product operation the depth of the recursion is only double-logarithmic in the size of the final graph (and hence logarithmic in the length of the description of vertices in it).

Digest. In the first construction, the zig-zag product was used both in order to increase the size of the graph and to reduce its degree. However, as indicated by the second construction (where the tensor product of graphs is the main vehicle for increasing the size of the graph), the primary effect of the zig-zag product is to reduce the degree, and the increase in the size of the graph is merely a side-effect (which is actually undesired in Section 5.2.4). In both cases, graph squaring is used in order to compensate for the modest increase in the relative eigenvalue caused by the zig-zag product. In retrospect, the second construction is the "correct" one, because it decouples three different effects, and uses a natural operation to obtain each of them: Increasing the size of the graph is obtained by tensor product of graphs (which in turn increases the degree), a degree reduction is obtained by the zig-zag product (which in turn increases the relative eigenvalue), and graph squaring is used in order to reduce the relative eigenvalue.

Stronger bound regarding the effect of the zig-zag product. In the foregoing description we relied on the fact, proved in [184], that the relative eigenvalue of the zig-zag product is upper-bounded by the sum of the relative eigenvalues of the two graphs. Actually, a stronger upper-bound is proved in [184]: For $g(x, y) = (1 - y^2) \cdot x/2$, it holds that

$$\begin{aligned} \bar{\lambda}_2(G' ⓩ G) &\leq g(\bar{\lambda}_2(G'), \bar{\lambda}_2(G)) + \sqrt{g(\bar{\lambda}_2(G'), \bar{\lambda}_2(G))^2 + \bar{\lambda}_2(G)^2} \\ &\leq 2g(\bar{\lambda}_2(G'), \bar{\lambda}_2(G)) + \bar{\lambda}_2(G) \\ &= (1 - \bar{\lambda}_2(G)^2) \cdot \bar{\lambda}_2(G') + \bar{\lambda}_2(G). \end{aligned} \tag{E.9}$$

Thus, we get $\bar{\lambda}_2(G'ⓩG) \leq \bar{\lambda}_2(G') + \bar{\lambda}_2(G)$. Furthermore, Eq. (E.9) yields a non-trivial bound for any $\bar{\lambda}_2(G'), \bar{\lambda}_2(G) < 1$, even in case $\bar{\lambda}_2(G')$ is very close to 1 (as in the proof of Theorem 5.6). Specifically, Eq. (E.9) is upper-bounded by

$$\begin{aligned} & g(\bar{\lambda}_2(G'), \bar{\lambda}_2(G)) + \sqrt{\left(\frac{1-\bar{\lambda}_2(G)^2}{2}\right)^2 + \bar{\lambda}_2(G)^2} \\ = \quad & \frac{(1-\bar{\lambda}_2(G)^2)\cdot\bar{\lambda}_2(G')}{2} + \frac{1+\bar{\lambda}_2(G)^2}{2} \\ = \quad & 1 - \frac{(1-\bar{\lambda}_2(G)^2)\cdot(1-\bar{\lambda}_2(G'))}{2} \end{aligned} \tag{E.10}$$

Thus, $1-\bar{\lambda}_2(G'ⓩG) \geq (1-\bar{\lambda}_2(G)^2)\cdot(1-\bar{\lambda}_2(G'))/2$. In particular, if $\bar{\lambda}_2(G) < 1/\sqrt{3}$ then $1-\bar{\lambda}_2(G'ⓩG) > (1-\bar{\lambda}_2(G'))/3$. This fact plays an important role in the proof of Theorem 5.6.