CS5275 Lecture 10: Expander Graphs

Jonathan Scarlett

December 6, 2024

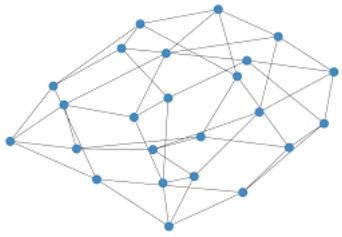
Acknowledgment. The first version of these notes was prepared by Niu Xinyuan and Chen Zhi Liang for a CS6235 assignment.

References:

- Video lecture: Expander Graphs by Ryan O'Donnell [1]
- Lecture notes: 7, 8, 12 by Ryan O'Donnell [2]
- Lecture notes: 3 by Ola Svensson [3]
- Textbook: Expander graphs and their applications, by Hoory, Linial, and Wigderson

1 Introduction

Informally, expander graphs are graphs that are simultaneously sparse and highly connected. This seems like a contradiction at first glance, but we will show formally that there are graphs that fulfill both properties. The following figure gives a simple example – there are relatively few edges, but we can still fairly quickly get from any vertex in the graph to any other:



A direct practical motivation for this would be if we were designing a network (e.g., of inter-connected computing devices) where forming connections (edges) is expensive but we still want a high degree of connectivity. As less obvious applications, we will later see that expander graphs are also useful for reducing the number of random bits needed in derandomization tasks, and for designing error-correcting codes. There are also other more abstract uses in areas like complexity theory (e.g., circuit lower bounds).

One approach to proving the existence of expander graphs is to use the probabilistic method, i.e., showing that a certain random graph gives the desired properties with high probability. On the other hand, it is

[1] https://www.youtube.com/watch?v=bONlIjZRJhA
[2] https://www.cs.cmu.edu/~odonnell/toolkit13/
[3] https://theory.epfl.ch/courses/topicstcs/Lecture3.pdf

often preferable to be able to construct such graphs *explicitly* without randomization. We will study both kinds of constructions.

The outline for the lecture is as follows:

1. Properties and definitions of expanders.
2. Extension to bipartite expanders.
3. Existence of expanders via the probabilistic method.
4. Explicit constructions of expanders.
5. Applications: Error correcting codes and derandomization

2 Definitions of Expander Graphs

We consider a undirected, d-regular graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = \frac{nd}{2}$ edges. A d-regular graph refers to graph where each degree of a vertex is d. In general, expander graphs are constructed by algorithm/network designers, and hence we have the flexibility to only look at d-regular graphs (and d-regularity typically suffices for applications). Further reading regarding irregular graphs of degree at most d can be found under Definition 9.2 in Hart et al. [2013].

2.1 Sparsity

The notion of a graph being "sparse" is relatively straightforward – its number of edges is small. Formally, we consider d-regular graphs such that d is a *constant that does not dependent* n. Thus, the number of edges is Thus, as $n \to \infty$, the number of edges in the graph increases as $O(n)$, which is much smaller than the maximum possible of $O(n^2)$ edges.

While slowly-growing scaling of d may also be of interest, e.g., $d = O(\log n)$, the constant-d regime is the most widespread and broadly applicable, corresponding to being as sparse as reasonably possible. (If we were to have $o(n)$ edges then some vertices would have to have no connections, so the graph certainly wouldn't be "well-connected".)

2.2 Highly connected

The connectivity of a graph can be quantified in several ways. We will look at 3 types of expansions, namely edge, vertex and spectral expansion, which help to formally define the notion of connectivity. We will also briefly discuss (without much detail) how these notions are connected to each other.

2.2.1 Edge expansion

Our first notion of connectivity can concerns edge properties, roughly stating that any given subset of vertices will have "not too few" edges from inside that subset to outside it. This prevents scenarios such as having a subset that is isolated from the rest of the graph, which would certainly not be "well-connected".

Define, for a given set of vertices S in d-regular graph G:

$$\phi[S] = \frac{\text{no. of edges (u,v) with } u \in S, v \notin S}{|S|} \tag{1}$$

This quantity measures how well connected S is to its complement $S^c = V \setminus S$. For a single choice of S this might not tell us much about the connectivity of the entire graph, but we get that by performing a minimization over S to obtain the quantity

$$\phi_G = \min_{0<|S|\leq \frac{n}{2}} \phi[S] \tag{2}$$

This measure is often known as *Cheeger's constant*. Notice that we have limited the size of $|S| \leq \frac{n}{2}$, but for sets bigger than that we can still draw conclusions about the connectivity by considering the complement set (which will have size below $\frac{n}{2}$).

To understand how ϕ_G related to connectivity, suppose that ϕ_G is lower bounded by a constant as $n \to \infty$, say $\phi_G > 0.05$. This means that for any subset S of size at most $\frac{n}{2}$, there are at least $0.05|S|$ edges from S to S^c. This is a constant fraction of the *highest feasible number*, which is $d|S|$ due to d-regularity. Thus, this condition prevents any scenario where some subset is "isolated" or has very low connectivity from the rest of the graph. (The constant 0.05 here is somewhat arbitrary and may seem low; sometimes any constant factor suffices for theoretical purposes, but sometimes we do care about getting a higher constant.)

We briefly note that a probabilistic interpretation is also possible – if we pick a random vertex in S and then traverse a random edge from that vertex, then probability of ending up outside S is at least some (small) constant. With this interpretation, ϕ_G can also be interpreted as measuring to what extent there are "bottlenecks" that prevent random traversals from exiting certain parts of the graph (higher ϕ_G means there are fewer and/or milder bottlenecks).

It is straightforward to show that $\phi_G > 0$ if and only if the graph is connected, i.e., there exists a path between any two vertices. (Consider trying this as an exercise.)

2.2.2 Vertex expansion

Along similar lines as edge expansion, one can consider counting the number of neighbors of a subset of vertices, instead of the number of edges directed out of this subset. Thus, we are interested in the following ratio for a given set $S \subset V$:

$$\phi'[S] = \frac{|N(S)|}{|S|}, \tag{3}$$

where $N(S)$ is the set of vertices in $S^c = V \setminus S$ that are connected to one or more vertices in S. Since we are working with d-regular graphs, it is not difficult to see that the vertex expansion is related to edge expansion by a factor of d, so the two at least coincide to within a constant factor.

Like with edge expansion, we could define $\phi'_G = \min_{0<|S|\leq \frac{n}{2}} \phi'[S]$, but the following "reparametrized" version tends to be more common.

Definition 1. *Let G be a d-regular graph with n vertices (let this set be V). Then G is an (n, d, ϵ) expander if for all $S \subseteq V$ where $0 < |S| \leq \frac{n}{2}$, we have:*

$$|N(S)| \geq \epsilon d|S| \tag{4}$$

for $\epsilon < 1$

This definition can be interpreted nicely – clearly for any subset of vertices S, the number of edges going out of S is $d|S|$ (since G is d-regular). However, some of these edges would connect to the same neighboring vertices (i,e., "collisions") and hence we expect the number of neighbors, $|N(S)|$, to be fewer than $d|S|$.

However, if our graph is a good expander we expect that the number of collisions is not too large. We then see that ϵ must lie in $[0, 1]$, with values near 1 indicating always having nearly the highest number of neighbors possible (good expansion), and values near 0 indicating much fewer (poor expansion). At least for theoretical purposes, having ϵ be any constant that doesn't decrease with n (e.g., $\epsilon = 0.01$) is often considered "large enough".

Definition 1 is the main notion of expanders that we will consider throughout the lecture, but it is useful to be aware of the others.

2.2.3 (**Optional**) Spectral expansion

Lastly, it is also possible to define well connectedness via the spectrum of the graph. This is sometimes of interest in its own right, and sometimes used a a stepping stone towards getting results regarding edge and vertex expansion.

Consider a d-regular graph with adjacency matrix $A \in \{0, 1\}^{n \times n}$, i.e., $A_{ij} = 1$ whenever i and j are connected. Instead of working directly with A, spectral properties of graphs are usually characterized by a related matrix called the *(normalized) Laplacian*, $L = I - \frac{1}{d}A$ (for reasons we won't go into). The matrix L has n eigenvalues, namely $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1} \leq 2$.

In turns out that Cheeger's constant ϕ_G (introduced above for edge expansion) can be bounded in terms of λ_1 via a famous result called *Cheeger's inequality*. One reason this is useful is that computation of Cheeger's constant ϕ_G is NP-hard in general, as it involves solving for the sparsest-cut of the graph. However, the computation of the eigenvalues λ_i of L can be done efficiently.

Formally, ϕ_G is related to the second smallest eigenvalue of L, λ_1, as follows:

$$\frac{1}{2}\lambda_1 \leq \phi_G \leq 2\sqrt{\lambda_1}. \tag{5}$$

In particular, we have a direct relation to edge expansion: If λ_1 is "large", then so is ϕ_G.

Definition 2. *A (n, d, ϵ)-spectral expander graph is a d-regular graph with n vertices and $\lambda_1 \leq 2\epsilon$.*

We now have three definitions of expanders, which have clear differences but are all closely related, not only conceptually but also via formal connections such as Cheeger's inequality.

- **(**Optional**)** For further reading on how the expansion measures relate to each other, see, e.g., `https://people.seas.harvard.edu/~salil/pseudorandomness/expanders.pdf`

2.3 Bipartite expanders

For many of the examples in this lecture, we will talk about expanders in the context of bipartite graphs, which are graphs $G = (V, E)$ where V is partitioned into two disjoint sets L and R such that all edges are between L and R (i.e., there are no edges within L nor within R). In this context, we consider *left d-regularity*, which is the property that each vertex in L is has exactly d edges to vertices in R. See Figure 1 for an illustration.

It is then appropriate to define *bipartite expanders*. In particular, we consider expansion from vertices on the *left* of the bipartite graph (instead of on every vertex, as was the case for general expanders above).

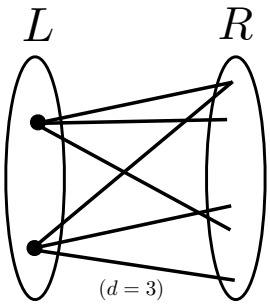


Figure 1: Bipartite graph illustration (left d-regular with $d = 3$).

Definition 3. *A bipartite graph with two disjoint vertex sets L and R, where $|L| = n$ and $|R| = m$ and $\deg(u) = d$ for all $u \in L$, is called a $(n, m, d, \gamma, \epsilon)$ expander if for all $S \subseteq L$ with $0 < |S| \leq \gamma n$, we have:*

$$|N(S)| \geq \epsilon d|S|. \tag{6}$$

Thus, bipartite expanders are defined in the same way as vertex expanders, but we only consider neighbors of vertices in the left side of the bipartite graph. This property is useful in applications where we are interested in mapping a set of objects to another explicitly, and hence the idea of two distinct subsets of vertices becomes relevant.

The parameter $\gamma \in [0, 1]$ gives some possible relaxation where we don't necessarily need the expansion property to hold for all subsets of size $\leq n$ or even $\leq n/2$, but rather only $\leq \gamma n$. This turns out to be sufficient in many applications even when γ is fairly small.

3 Existence via the Probabilistic Method

At this stage, it may not be clear whether the above expansion notions are even attainable. It turns out that they indeed are, and we can show it using the probabilistic method – generate a random graph according to a suitably-defined distribution, and show that it has a positive probability of being an expander graph.

There are many results showing the existence of expanders of various types (edge/vertex/spectral, bipartite vs. non-bipartite) and with various parameters (e.g., ϵ and/or γ). Rather than giving a general statement, for concreteness we focus here on one particular set of parameters for bipartite vertex expanders.

Theorem 3.1. *Consider a randomly constructed bipartite graph $G(L, R, E)$ with two disjoint vertex subsets L and R, where $|L| = n$ and $|R| = m$ and $deg(u) = d$ for all $u \in L$. The random construction is performed as follows:*

- *For each vertex $u \in L$, perform the following independently: Select d vertices in R uniformly at random without replacement, and create an edge from u to each of these vertices.*

For $d \geq 32$, $m \geq 3n/4$, and large enough n, it holds with probability at least $\frac{18}{19}$ that the graph has the following vertex expansion property:

$$|N(S)| \geq \frac{5d}{8}|S|, \quad \forall S \subseteq L : |S| \leq \frac{n}{10d}. \tag{7}$$

That is, the graph is a $\left(n, m, d, \frac{5}{8}, \frac{1}{10d}\right)$ bipartite expander.

Proof. Consider a random d-left regular graph $G(L, R, E)$ with $|L| = n$, $|R| = m$ (with the probability distribution being as given in the theorem statement), and let $S \subset L$ have cardinality $s = |S| \leq \frac{n}{10d}$. Given such an S, the desired expansion probability will be violated if $N(S) \subseteq T$ for some $T \subseteq R$ of size $t = \frac{5d}{8}s$. We call this a "bad event" and denote it by $X_{S,T}$. We wish to show that with positive probability, none of the bad events occur. *(Try to convince yourself the case $t = \frac{5d}{8}s$ suffices, rather than $t \leq \frac{5d}{8}s$.)*

In total, there are sd edges leaving S. We study the probability of one bad event $X_{S,T}$ as follows:

- Interpret the procedure of sampling d vertices without replacement as follows: First pick a vertex uniformly at random from m options (in R), then another uniformly at random from the remaining $m-1$ options, and so on, down to $m-d+1$ options on the d-th selection.
- First consider a single vertex $u \in s$. By the preceding interpretation, the probability that all of its neighbors are in T is given by

$$\frac{t}{m} \cdot \frac{t-1}{m-1} \cdot \ldots \cdot \frac{t-d+1}{m-d+1} \leq \left(\frac{t}{m}\right)^d,$$

  where the inequality follows from $\frac{A-1}{B-1} \leq \frac{A}{B}$ when $A \leq B$ (e.g., $\frac{2}{3} \leq \frac{3}{4}$), and we have $t \leq m$ because $T \subseteq R$. (Note also that $t = \frac{5d}{8}s \leq \frac{5d}{8}\frac{n}{10d} = \frac{n}{16}$, whereas we assume $m \geq 3n/4$.)
- Since the random neighbors selected for each $u \in S$ are independent of each other, it follows that the overall probability of $X_{S,T}$ is at most $(t/m)^{sd}$.

For the desired expansion property to hold, we require that $X_{S,T} = 0$ for all possible choices of S and T. Thus, the probability is *failing* to get the desired expansion condition is

$$\begin{aligned}
\Pr\left[\bigcup_{S,T}\{X_{S,T} > 0\}\right] &\overset{(a)}{\leq} \sum_{S,T} \Pr\left[X_{S,T} = 1\right] \\
&\overset{(b)}{\leq} \sum_{S,T} (t/m)^{sd} \\
&\overset{(c)}{\leq} \sum_{s=1}^{n/10d} \binom{n}{s}\binom{m}{5ds/8}\left(\frac{5ds}{8m}\right)^{sd} \\
&\overset{(d)}{\leq} \sum_{s=1}^{n/10d} \left(\frac{ne}{s}\right)^s \left(\frac{8me}{5ds}\right)^{5ds/8}\left(\frac{5ds}{8m}\right)^{sd} \qquad (8) \\
&\overset{(e)}{\leq} \sum_{s=1}^{n/10d} \left(\frac{1}{20}\right)^s \\
&\leq \sum_{s=1}^{\infty} \left(\frac{1}{20}\right)^s \\
&= \frac{1}{19},
\end{aligned}$$

where (a) uses the union bound, (b) was shown in the dot points above, (c) follows by counting the number of S and T with $|S| = s \leq \frac{n}{10d}$ and $|T| = t = \frac{5d}{8}s$, (d) uses $\binom{n}{k} \leq (ne/k)^k$, and (e) is a nuisance to work out line-by-line but essentially amounts to showing that each summand in the previous line is small (at most $\left(\frac{1}{20}\right)^s$) under the assumed conditions $d \geq 32$, $m \geq 3n/4$, and $s \leq \frac{n}{10d}$. The details:

- Define $a = \frac{ne}{s} \cdot \left(\frac{8me}{5ds}\right)^{5d/8} \cdot \left(\frac{5ds}{8m}\right)^{d}$, so that the summands in (d) are a^s.
- The dependence on s is $\frac{1}{s} \cdot s^{d(1-5/8)}$, so since $d \geq 32$, it is increasing in s. Thus, we may upper bound a by replacing s by its upper bound $\frac{n}{10d}$ to get $a \leq 10de \left(\frac{16me}{n}\right)^{5d/8} \left(\frac{n}{16m}\right)^{d}$.
- Now the dependence on m is $m^{d(5/8-1)}$, which is decreasing, so we can get an upper bound by replacing m by its smallest value $\frac{3n}{4}$: $a \leq 10de \cdot (12e)^{5d/8} \cdot \left(\frac{1}{12}\right)^{d}$.
- A direct calculation gives $(12e)^{5/8} \times \frac{1}{12} < 0.736$, giving $a \leq 10de \cdot (0.736)^d$, which we can directly verify (numerically or analytically) to be below $\frac{1}{20}$ when $d \geq 32$.

For a more general analysis using generic constants instead of $\frac{5}{8}$, $\frac{1}{20}$, etc., refer to Ola Svensson's notes[4].

Since there is at most a $\frac{1}{19}$ chance of at least one bad event occurring, there is at least an $\frac{18}{19}$ chance of none of them occurring, as desired. □

3.1 Bipartite expander graphs from general expander graphs

While we have just proved the existence of bipartite expanders using the probabilistic method, it is also useful to note another general approach to getting bipartite expanders. Specifically, we describe an alternative approach in which a regular graph can be "converted" into a bipartite graph while maintaining its expansion properties, which will generally be easier than performing an entirely separate analysis/construction.

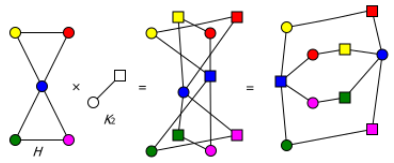


Figure 2: Double cover for a graph

The idea is a notion called the *double cover* of a (non-bipartite) graph G, which is a bipartite graph H with twice the number of vertices and edges as G. To construct H, duplicate vertices in graph G such that for each vertex v_i of G, graph H has two corresponding vertices u_i and w_i. Next, for each edge in G, connect the corresponding vertices across the 2 halves of H, i.e. for an edge (v_i, v_j) in G, replace the edge in H with the edges (u_i, w_j) and (w_i, u_j). Mathematically, for a graph with adjacency matrix A, the adjacency matrix of the double cover is

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}.$$

The resulting bipartite graph H remains d-regular. See Figure 2 for a simple example.

When the original graph is a good expander, the double cover is also a good expander, converting a (n, d, ϵ) expander (Definition 1) into a (n, d, ϵ) bipartite expander. This is because for each vertex and its neighbors in the original graph, the vertex is connected to the same corresponding vertices in the other mirrored half of the graph after the double cover operation.

4 Explicit Constructions

There are two ways to define the explicit construction of expander graphs.

[4] https://theory.epfl.ch/courses/topicstcs/Lecture3.pdf

Definition of explicitness:. A deterministic algorithm outputs the expander graph's entire adjacency matrix in poly(n) time.
Definition of strong explicitness. Given any $u \in [n]$ (a vertex index), $i \in [d]$ (a neighbor index of that vertex), a deterministic algorithm outputs the i-th neighbor of u in poly(log(n)) time.

Observe that in the latter definition, we are not asking for the full adjacency matrix to be formed, but rather, for specific entries of it to be computable *very efficiently* (poly-log instead of poly). As we will see below, this allows working with graphs with a number of vertices that are exponential in n, in which case we would certainly like to avoid constructing the full adjacency matrix.

Of course, strong explicitness implies explictness, because one can just loop over all $n \times n$ entries of the adjacency matrix and compute them one-by-one, incurring a total time of $O(n^2 \text{poly}(\log n))$. On the other hand, explicitness does not imply strong explicitness.

The following subsections give a few well-known (strongly) explicit constructions of expander graphs; their expansion properties will be stated without proof. The theorems all state a spectral expansion property, but these can be related to edge expansion via (5), and to vertex expansion via other tools that we didn't cover (other than a very crude one with a factor of d). These examples follow the ones in the CMU lecture `https://www.youtube.com/watch?v=j6JzqPkvRHM`.

4.1 (**Optional**) Margulis-Gabber-Galil Expanders

Let $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$, and consider mod-$m$ arithmetic over this set. Let $V = \mathbb{Z}_m^2$, where vertices are indexed by $x, y \in \mathbb{Z}_m$ on a two dimensional grid with $m \in \mathbf{Z}^+$ points in each dimension (so m^2 total). Construct the edge set E by connecting each vertex with coordinate (x, y) to the following 8 neighbors:

1. $(x \pm y, y)$
2. $(x \pm (y+1), y)$
3. $(x, y \pm x)$
4. $(x, y \pm (x+1))$

where the $+$ and $-$ operators are performed modulo m [Goldreich, 2011]. Note that this could mean connecting a vertex to itself, or connecting one vertex to another multiple times, meaning it is technically a *multi-graph*, but we won't worry so much about this distinction.

Theorem 4.1. *[Gabber and Galil, 1981] The method described above constructs a $(m^2, 8, \epsilon)$-spectral expander graph for some $\epsilon > 0$ ($\epsilon \approx 0.1$)*

This method is strongly explicit, since finding the i-th neighbor of any vertex can trivially be done in $O(1)$ time. Beyond the fact that it is strongly explicit, we see that it is remarkably simple, being completely described by just 4 extremely basic equations.

The proof of Theorem 4.1 is linear algebra based and is less straightforward.

4.2 (**Optional**) Ramanujan graph expanders

The constant $\epsilon \approx 0.1$ in Theorem 4.1 is reasonable, but an analysis of random graphs suggests we can do much better – a random d-regular graph will in fact give a Laplacian matrix L with eigenvalues (except the

one that is zero) very close to one, namely $1 - O\big(\frac{1}{\sqrt{d}}\big)$, suggesting that we could get spectral expansion with $\epsilon \approx 1$.

In this section, we introduce another strongly explicit construction called *Ramanujan graph expanders* that can attain an analogous improvement for certain d values. Since we are interested in eigenvalues of L that are close to one, it is more convenient to work with one minus the eigenvalues of L as follows.

Definition 4. *A Ramanujan graph is a d-regular graph that satisfies*

$$\kappa := \max\{|\kappa_i| : i \in [n-1]\} \leq \frac{2}{\sqrt{d}}\sqrt{1 - \frac{1}{d}} \tag{9}$$

where κ_i are the eigenvalues for the normalised adjacency matrix $K = \frac{1}{d}A$. (Since the normalized Laplacian matrix is $L = I - K$ and has eigenvalues λ_i, we have the relationship $\kappa_i = 1 - \lambda_i$.)

Observe that when d is large, such a graph is an expander (from the definition of spectral expansion in Definition 2), and moreover, the expansion constant ϵ is very close to one. Remarkably, the closeness to one not only matches what random graphs give, but does so in a cleaner non-asymptotic manner.

Theorem 4.2. *There exists a strongly explicit construction of a d-regular Ramanujan graph for any $d \geq 3$ of the form $d = p^k + 1$, where $k \in \mathbb{Z}^+$ and p is a prime number.*[5]

The constructions themselves (which we haven't described) are not especially complicated, but the analysis of the resulting eigenvalues are very advanced based on tools from number theory. While the expansion properties are strongest for large d as mentioned above, the special case of $d = 3$ comes out to be particularly simple as follows.

Corollary 4.3. *Let $V = \mathbb{Z}_n$ with n being a prime number, and let E be the set of edges in which we connect $a \in V$ to $a + 1$, $a - 1$ and a^{-1} in the finite field based on mod-n arithmetic (take 0^{-1} to produce 0). Then this is a $(n, 3, \epsilon)$-spectral expander with $\epsilon \approx 0.01$.*

Intuitively, the edges $(a, a + 1)$ and $(a, a - 1)$ produce a cycle of all the vertices $a \in V$, while the edges (a, a^{-1}) produce "pseudo-random edges" that improve connectivity. As a result, the resultant graph from this construction achieves spectral expansion with similar behavior as a random 3-regular graph.

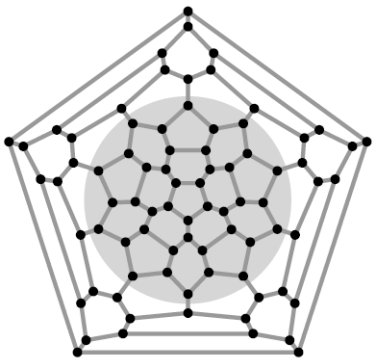


Figure 3: Ramanujan expander graph with 80 vertices [Sarnak, 2004]

[5]Early works in this direction further assumed that $p \equiv 1 \pmod 4$, but that assumption was subsequently dropped.

4.3 (**Optional**) Zig-Zag product expanders

The final explicit construction that we cover is roughly based on iteratively constructing larger and larger expander graphs. This general approach is especially suited to getting very good bipartite expander graphs, e.g., with the constant ϵ in $|N(S)| \geq \epsilon d|S|$ being a "good" constant like 0.8 rather than an "OK" one like 0.05. The specific method that we cover is simpler and perhaps not quite good enough to attain such constants, but it is easier to understand.

We start with the following definition, which we will typically use with G being a "large" graph and H being a "small" one.

Definition 5. *Given two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ where G is an n-vertex, D-regular graph and H is a D-vertex, d-regular graph. We define the **replacement product**, $G \textcircled{r} H$, as a $2d$-regular graph with a vertex set $V_G \times V_H$ in which each vertex in G is replaced by a copy of H and (g, h) has an edge to (g', h') if and only if either (i) $g = g'$ and $(h, h') \in E_H$, or (ii) $g \neq g'$, g' is the h-th neighbor of g in G, and g is the h'-th neighbor of g' in G.*

This definition is a bit complicated, but should become clearer with an example shown below:

- $G = G_1$ has $n = 7$ nodes and degree $D = 6$;
- $H = G_2$ has $D = 6$ nodes and degree $d = 2$;
- The replacement product $G_1 \textcircled{r} G_2$ contains n "copies" of G_2, and there are further edges connecting different copies in a manner that matches the neighbor numberings in the original G_1.

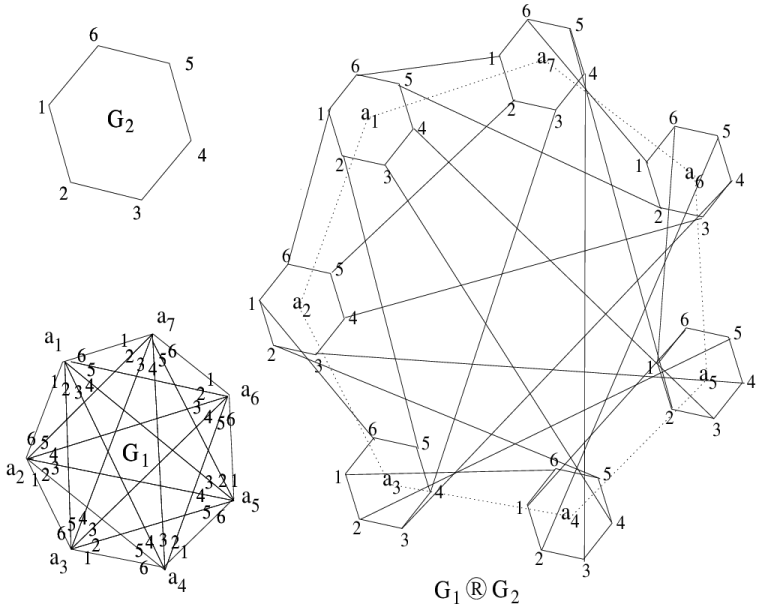


Figure 4: Replacement product example from the paper "Zig-zag and replacement product graphs and LDPC codes" (Kelley/Sridhara/Rosenthal, 2006).

As it turns out, if G and H are both spectral expanders, then so is their replacement product.

Theorem 4.4. *Consider the case that G is a (n, D, ϵ_G)-spectral expander and H is a (D, d, ϵ_H)-spectral expander. Then, the replacement product $G \textcircled{r} H$ is a $(Dn, 2d, \frac{\epsilon_H \epsilon_G}{16})$-spectral expander.*

Notice that the number of vertices has increased (the sizes of G and H get multiplied), the degree has changed from D to $2d$ which is typically a decrease (due to G being larger than H), and the spectral expansion constant has decreased. The last of these is not desirable, but there is a way to increase the expansion factor. We only provide a brief overview of the idea as follows.

Given the graph G, consider G^t, which is a new graph with edges added based on the t-step connectivity (if there exists a t-step path from edges u to v in G, add an edge to them). Then G_t has a degree of d^t (including multi-edges). It can be shown that after performing this idea on the replacement product, the resulting graph is "approximately" a $(Dn, (2d)^t, t\frac{\epsilon_H \epsilon_G}{16})$-expander (this statement is a bit imprecise, as the expansion factor is only an approximate expression).

By carefully interleaving the two steps (replacement product and $(\cdot)^t$), we can get a good expander graph – the replacement product helps keep the degree from getting too large, and the $(\cdot)^t$ step prevents the spectral expansion parameter from getting too small. See the CMU video lecture for somewhat more detail (though it is still on the brief side).

5 Applications

5.1 Error correcting codes

Consider the case where a sender must send a message to a receiver over a noisy channel that can flip an arbitrary fraction of the k bits of the original message. Knowing that the channel is noisy, the sender adds redundancy by encoding the message into $n > k$ bits. The hope is that this can be done in a manner that attains multiple goals:

- Send information at a high rate (i.e., keep $\frac{k}{n}$ as high as possible);
- Achieve resilience to errors (i.e., correct decoding is guaranteed even when there are δn bit flips; the higher δ the better);
- Maintain low computational complexity at the encoder and decoder.

We will show how to use expander graphs to build such a code.

The notions of high rate and resilience to errors are formalized as follows.

Definition 6. ***(Code rate)****. The rate of a code $C \subset \{0,1\}^n$ is $R = \frac{k}{n}$.*

Definition 7. ***(Minimum distance)****. The normalized minimum distance (or "distance" for short) D of a code $C \subset \{0,1\}^n$ is $\min_{c_1 \neq c_2} \frac{1}{n} d_H(c_1, c_2)$, where d_H is the Hamming distance.*

The higher the D, the more tolerant is our coding method to errors during transmission. In particular, it can be shown that an optimal decoder is always able to uniquely recover the message when there are $\frac{D-1}{2}$ or fewer bit flips.

In general, when the rate is high, the distance tends to be small (and vice versa). A sequence of codes (indexed by k) is said to be *asymptotically good* the rate and distance are both lower bounded by positive constants as $k \to \infty$ (e.g., $R \geq 0.01$ and $D \geq 0.01$).

- Note: Even better would be to get the rate and distance both as high as possible (instead of just "any constant value"), and expanders can be good for that too, as well as being extremely efficient computationally (in particular, $O(n)$ decoding time is attainable). However, we'll only focus on this more modest goal here, and we won't place much emphasis on computation.

We can obtain a asymptotically good code using bipartite expanders. We give a specific example using specific constants, but this can be generalized to get a more general trade-off between rate and distance.

To represent the error correcting code as a bipartite expander, we consider codewords of length n, with $m = \frac{3}{4}n$ parity check constraints. The dimension of the code is $k = n - m = \frac{1}{4}n$, representing a constant rate of $\frac{1}{4}$. We represent the error correcting code as a bipartite graph with $|L| = n$ and $|R| = m$. Such a bipartite graph is called the *Tanner graph* of C. The vertices in the left subgraph L represent the bits of the code, while the vertices in the right subgraph R represent the parity checks. For each parity check, the bits connected to it are required to consist of an even number of 1s. It turns out to work well to let this graph be a bipartite expander graph, and we will specifically adopt the choice from Theorem 3.1.

Consider the adjacency matrix of the bipartite expander graph $\boldsymbol{H} \in \{0,1\}^{n \times m}$, where $H_{i,j}$ (with $i \in L$ and $j \in R$) is the indicator for when there is an edge from vertex i in the left subgraph to vertex j in the right subgraph. Then, a string $\boldsymbol{x} \in \{0,1\}^n$ is a valid codeword if and only if all the parity checks are satisfied:

$$\boldsymbol{x} \in C \iff \bigoplus_{i=1}^{n} H_{i,j} x_i = 0, \ \forall j \in \{1 \cdots m\} \tag{10}$$

where $\oplus$ is mod-2 addition. We make use of the properties of the bipartite expander graph (Theorem 3.1) to analyse the Hamming distance of the code C.

Lemma 5.1. *Consider the bipartite expander graph from Theorem 3.1. For any subset $S \subseteq L$ with $|S| \leq \frac{n}{10d}$, there exists a vertex $v \in N(S)$ with exactly one neighbor in S.*

Proof. Assume, for the sake of contradiction that for all $v \in N(S)$, it holds that $|N(v) \cap S| \geq 2$. Then,

$$(\#\text{edges from } S \text{ to } N(S)) \geq 2|N(S)| \geq 2 \cdot \frac{5d}{8}|S| > d|S|,$$

where the $\frac{5d}{8}$ term comes from the expansion property in Theorem 3.1. This contradicts with the fact that the graph is d left regular, meaning there are only $d|S|$ edges containing vertices from S. □

Corollary 5.2. *The minimum Hamming distance of the code C is greater than $\frac{n}{10d}$, which is linear in n.*

Proof. Let $\boldsymbol{x}$ be any valid codeword, and let $\boldsymbol{x'}$ be any other sequence whose Hamming distance to $\boldsymbol{x}$ is at most $\frac{n}{10d}$. Let S be the set of indices at which $\boldsymbol{x}$ and $\boldsymbol{x'}$ differ. From Lemma 5.1, there exists some $v_i \in N(S)$ with exactly one neighbor in S.

Since $\boldsymbol{x}\boldsymbol{H} = \boldsymbol{0}$ (due to $\boldsymbol{x}$ being a codeword) and $\boldsymbol{x}, \boldsymbol{x'}$ differ in position i (since $i \in S$), it follows that the i-th entry of $\boldsymbol{x'}\boldsymbol{H}$ is 1, meaning $\boldsymbol{x'}$ is not a codeword.

Thus, for any codeword $\boldsymbol{x}$, there are no other codewords within Hamming distance $\frac{n}{10d}$. □

Next, we proceed to consider the decoding step, which seeks to recover $\boldsymbol{x}$ from its corrupted version $\boldsymbol{y} = \boldsymbol{x} \oplus \boldsymbol{z}$ (with $\boldsymbol{z} \in \{0,1\}^n$ containing a 1 wherever a bit is flipped).[6] We will focus only on a very simple decoder, described as follows.

Similarly to the proof for Corollary 5.2, it is evident that at each iteration of the loop, there exists at least one constraint violation associated with exactly one bit in x_i ($i \in S$), where S is the index set of parity checks that are violated. Flipping that bit would reduce the number of constraint violations by one. There may be better choices that reduce by more than one, but even so, we can conclude that the number of

[6] Actually the goal is to recover the original message bits, but that's straightforward once $\boldsymbol{x}$ is recovered.

Input: $\boldsymbol{y}$ such that $\boldsymbol{yH} \neq \boldsymbol{0} \pmod 2$
Output: A codeword $\boldsymbol{x}$ (desired to be as close to $\boldsymbol{y}$ as possible)

$\boldsymbol{x} \leftarrow \boldsymbol{y}$
while $\boldsymbol{xH} \neq \boldsymbol{0} \pmod 2$ **do**
    Flip any x_i that decreases the number of constraint violations in $\boldsymbol{xH}$

constraint violations will *strictly decrease* on each iteration. Thus, this number will eventually decrease to zero, meaning we end up with a valid codeword.

The preceding argument does not establish that we will end up with the *closest* codeword, but this turns out to also be true. For the proof of that, see Lecture 8 by Venkatesan Guruswami[7], which also describes other decoding algorithms with a *linear* runtime of $O(n)$. (The above algorithm has polynomial runtime, but not linear.)

5.2 Error reduction in randomized algorithms

Expanders are also useful in area of error reduction (and derandomization more broadly). The goal in this problem is to reduce the error probability of a randomized algorithm without using too many extra random bits – random bits are often viewed as a scarce resource, so the fewer that are needed, the better.

Let $\mathcal{A}$ be a randomized algorithm for solving a decision problem (i.e., something with a YES/NO answer). Suppose that we require a *one-sided* error guarantee:

- If the correct answer is YES, the output must be YES with probabiltiy one;
- If the correct answer is NO, the output must be NO with some specified probability (e.g., 2/3 or 0.95).

For example, the algorithm might be for checking whether an input number is prime: If the number is prime, $\mathcal{A}$ returns 1 with probability 1, and if the number is non-prime, $\mathcal{A}$ returns 1 with probability ≤ 0.05 - for example, the Miller-Rabin primality test has such properties.

Assume that this algorithm makes use of a random n-bit string from $\{0,1\}^n$ and makes a mistake over at most a fraction 0.05 (say) of all n-bit random strings.

5.2.1 Naive approach

One naive approach for reducing the error of our random algorithm is to repeat it d times with a different random n-bit string each time. This incurs:

1. dn random bits.
2. at most 0.05^d chance of being wrong. (Hence, we need $d = O\big(\log \frac{1}{\delta}\big)$ to get down to some target δ.)
3. algorithm runtime of $dT + O(n)$ (including $O(n)$ time for generating the random bits), where T is the time for a single invocation.

This approach requires us to regenerate a new random n-bit string on each invocation of the algorithm, and it turns out that this can be avoided/alleviated using expanders.

[7] https://www.cs.cmu.edu/ venkatg/teaching/codingtheory/notes/notes8.pdf

5.2.2 Improved approach using expanders

Suppose that we have a strongly explicit algorithm to generate a bipartite expander that has the properties in Theorem 3.1 with $|L| = |R| = 2^n$. For both L and R, we represent any given vertex via a unique n-bit string, for a total of 2^n vertices on each side. We will interpret these strings as choices of the "random seed" for the randomized algorithm.

Now, consider the algorithm described as follows. First pick a random vertex ℓ from L; this is equivalent to picking a n-bit random string (just like in the naive approach). Next, use the strongly explicit algorithm to generate the d neighboring vertices of $\ell : r_1, r_2, \ldots, r_d$ that form $N(\ell) \subseteq R$ in the bipartite expander. Due to the strongly explicit property, this takes $\text{poly}(\log(2^n)) = \text{poly}(n)$ time. We now have d n-bit strings which we run $\mathcal{A}$ with, and we return 1 if the answer is positive for all d trials, and 0 otherwise .

Claim: The preceding algorithm incurs:

1. n random bits.
2. at most $\frac{0.1}{d}$ chance of being wrong (proved below).
3. a runtime of $dT + \text{poly}(n)$, where T is the time for a single invocation.

Notice that compared to just running $\mathcal{A}$ once, the algorithm uses the *same* number of random bits but has a smaller error rate ($\frac{0.1}{d} \leq 0.05$ for $d \geq 2$), albeit at the cost of a higher runtime (at least a factor d larger).

Proof of error rate. Since the algorithm makes use of n-bit strings on R (right partition) of our bipartite expander, let $B_x \subseteq R$ denote "bad" n-bit strings which cause $\mathcal{A}$ to give the wrong answer. Note that $|B_x|$ consists of at most a 0.05 fraction of the number of n-bit strings i.e., $|B_x| \leq 0.05(2^n)$. Correspondingly, define $S \subseteq L$ to contain the "bad" choices in L such that if we choose any $\ell \subseteq S$ in the algorithm, we will have all of its neighbors in B_x (i.e., $N(\ell) \subseteq B_x$ – this is the only case where the algorithm makes an error; if some vertices in $N(\ell)$ lies outside of B_x, then the algorithm will output the correct answer). The key observation is stated in the following lemma.

Lemma 5.3. *The number of bad choices $|S|$ in L is such that $|S| < \frac{0.1}{d}2^n$*

Proof. For the sake of contradiction, assume that $|S| \geq \frac{0.1}{d}2^n$. Then, consider $S' \subseteq S$ such that $|S'| = \frac{0.1}{d}2^n$. The bipartite expansion property from Theorem 3.1 gives the following:

$$|N(S')| \geq \frac{5}{8}d|S'| \geq \frac{5}{8}d\frac{0.1}{d}2^n = \frac{1}{16}(2^n) > |B_x|, \tag{11}$$

where the last step follows by recalling that $|B_x| \leq 0.05(2^n)$. This implies that there exist some choices in S' such that the algorithm does not produce the wrong answer (since the algorithm will only give a wrong answer if all vertices chosen in R are in B_x). This leads to a contradiction, because S' can only contain "bad" choices. Thus, our original assumption $|S| \geq \frac{0.1}{d}2^n$ must have been incorrect, meaning we indeed have $|S| < \frac{0.1}{d}2^n$. □

Therefore, the probability of us picking one of these "bad" choices in L is at most $\frac{|S|}{2^n} = \frac{0.1}{d}$, which completes the claim.

Overall, the expansion property is a useful sufficient condition for ensuring that this error reduction technique works well. In principle we could use any bipartite graph, but graphs with poor expansion

properties may be significantly less useful. This is because vertices in L might map to many identical vertices in R, implying that the n-bit strings in R are not "random enough". The properties of expanders prevent such undesirable scenarios.

5.3 (**Optional**) Other applications

We briefly note that expanders also arise in many other topics in theoretical computer science and applied domains. Some examples and corresponding paper titles are as follows:

- Cryptography (e.g., "Cryptographic hash functions from expander graphs")
- Sparse estimation (e.g., "Efficient and robust compressed sensing using high-quality expander graphs")
- Group testing (e.g., "Derandomization and group testing")
- Circuit complexity lower bounds (e..g, "Poly-logarithmic Frege depth lower bounds via an expander switching lemma")
- Network design (e.g., "Optimal network topologies: Expanders, cages, Ramanujan graphs, entangled networks and all that")
- Deep neural networks (e.g., "Deep expander networks: Efficient deep networks from graph theory")

References

Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981.

Oded Goldreich. *Basic Facts About Expander Graphs*, pages 451–464. Springer Berlin Heidelberg, 2011.

K.P. Hart, J. van Mill, and P. Simon. *Recent Progress in General Topology III*. SpringerLink : Bücher. Atlantis Press, 2013.

Peter Clive Sarnak. What is . . . an expander? *Notices of the American Mathematical Society*, 51(7):762–763, August 2004.