UNIVERSITY
OF OSLO

Bachelor's thesis

Expander graphs from property T groups

Exploration of the explicit construction of expander graphs

Mikkel Gjestrud

Supervisor: Makoto Yamashita
MAT2000 Project work in Mathematics

Department of Mathematics
Faculty of Mathematics and Natural Sciences

Spring 2023

Contents

Abstract

For every connected graph, there is an associated expansion factor. The expansion factor for a graph is a number that measures how much subsets of a graph grows by adding its boundary. We look specifically at graphs with few edges, and high expansion factors (i.e. expander graphs), and especially how they can be constructed. For this, one needs theory from topology, group theory, representation theory, and more. We shall then program an explicit expander graph.

1 Introduction

The central topic of this paper is expander graphs, also referred to as "expanders". While the exact formulation of expanders can vary a little bit, they all aim to describe the same thing: They are highly connected sparse graphs. They are sparce in that they have relatively few edges compared to how many vertices there are, and they are highly connected in that they are hard to separate. The expansion factor captures just how connected a graph is, where a higher number means that it is more connected. A family of expanders is a family of similar graphs, where the graphs increase in size, and all are expanders for the same expansion factor.

Expanders are very useful graphs and there are many real-world applications of their theory. To study the graphs more thoroughly, one would like to know an explicit construction of an expander graph. This turns out to be a difficult problem to solve, and was first shown by Grigorij Margulis.

There are two ways of constructing an expander. One is through Kazhdan property T and simple Lie groups, while the other uses the Ramanujan conjecture. We will use the first method, which was the one Margulis used in his first proof.

After showing that there exists a construction, it is quite natural to use it to construct a specific family of expanders. We will use an example from [3], and go a step further in computing it as well. This allows us to try to estimate the expansion factor for a specific family of expanders.

2 Background and notation

2.1 Notation and some graph theory

A graph in mathematics consists of a set V of *vertices* (also called nodes), and a set E of *edges* between the vertices, where an edge is a pair of vertices (v_1, v_2), where $v_1, v_2 \in V$. An example can be seen in Figure 1. We call a graph connected if for every pair of vertices, there exists a path between those vertices, that is, you can get from one vertex to another by going through edges in E. Equivalently, a graph is connected if one cannot split the graph into two distinct subsets such that there exists no edges between these subsets. The graphs relating to expander graphs are undirected, meaning they travel both ways when connecting vertices. Moreover, relating to expanding graphs we usually look at k-regular graphs. A graph is k-regular when every vertex has exactly k edges, where $k \in \mathbb{N}$.

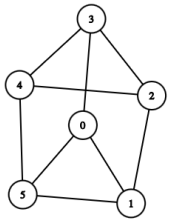

Figure 1: A connected 3-regular graph with 6 vertices and 9 undirected edges. This drawing was made using https://csacademy.com/app/graph_editor/

We want to introduce some of the notation we shall use in the paper.

- Define $\mathbf{1}_V \coloneqq \begin{cases} 1 \text{ if } x \in V \\ 0 \text{ otherwise} \end{cases}$ as *the indicator funtion*
- Define $\mathbb{C}_{\mathbf{1}_V} \coloneqq \{a\mathbf{1}_V \mid a \in \mathbb{C}\}$ as *the constant function over the complex numbers*
- Define $\oplus$ as the *direct sum*

2.2 Expander graphs

There are several different formulations of an expander. We will use one of the formulations from [3] which is as following:

Definition 2.1. Let $G = G(V, E)$ be a finite, k-regular graph, with a set V of n vertices, and a set E of edges (so the set contains $\frac{kn}{2}$ edges). Let $\partial A = \{y \in V \setminus A \mid d(y, A) = 1\}$ be the boundary of A, and d is the distance function on G (Equivalently, $\partial A = \{y \in V \setminus A \mid$ there is an edge between y and an element in V$\}$). G is called an (n, k, c)-expander for an expansion factor $c > 0$ if for every subset $A \subseteq V$,

$$|\partial A| \geq c\left(1 - \frac{|A|}{n}\right)|A|. \tag{1}$$

It is desirable to choose the expansion factor c as large as possible, and so we will work with the assumption that our c is the largest possible value, such that Equation (1) holds for this c, and all subsets A. Granted, every connected graph is a c-expander for some c, because one can always choose c small enough to fit the definition, but we choose to look at graphs with higher factors. It gets more interesting when looking at infinite families of graphs with the same c and k. In these cases we look at what happens when the set of vertices gets larger and larger ($n \to \infty$), but the graphs are still (n, k, c)-expanders for fixed k and c.

2.3 Cheeger constant

The Cheeger constant measures how hard it is to seperate a graph into two disjoint parts by cutting edges.

Definition 2.2. Let $G = G(V, E)$ be a finite graph. Define the Cheeger constanst h(G), by

$$h(G) = \min_{\emptyset \neq A \subseteq V} \frac{|E(A, V \setminus A)|}{\min\{|A|, |V \setminus A|\}},$$

where $E(A, B)$ is the set of edges connecting A to its complement.

The Cheeger constant is strongly connected to the definition of an expander graph, which can be seen in the following proposition, which is proposition 1.1.4 in [3]:

Proposition 2.3. *Let K be a k-regular graph with n vertices. Then*

- *If G is an (n, k, c)-expander, then $h(G) \geq \frac{c}{2}$,*
- *G is an $(n, k, \frac{h(G)}{2})$-expander.*

2.4 Cayley graph

First we define a generating set.

Definition 2.4. A subset S of a group G is a *generating set* if every element $g \in G$ can be expressed as a combination (under the group operation) of a finite amount of elements $s \in S$, together with their inverses s^{-1}.

Now we can define a Cayley graph (also called a Cayley digraph graph, Cayley directed, Cayley diagram, and more).

Definition 2.5. Let S be a generating set of a group G. Then we define a *Cayley graph* as a graph where every $g \in G$ is a vertex, and $g \in G$ is connected to $gs \in G$ for every $s \in S$.

Remark 2.6. We will in practice deal with undirected Cayley graph. This is because we require our generating set to include all of its inverses as well, and so every edge will necessarily go both ways.

2.5 Unitary representations

A Hilbert space $\mathcal{H}$ is an inner product space that is complete with respect to the norm defined by the inner product. The following definition is from [1].

Definition 2.7. The *unitary group* $\mathcal{U}(\mathcal{H})$ of $\mathcal{H}$ is the group of all invertible bounded linear operators $U \colon \mathcal{H} \to \mathcal{H}$ which are unitary, namely such that $UU^* = U^*U = I$, where U^* is the adjoint of U, and I is the identity operator in $\mathcal{H}$.

Definition 2.8. A *unitary representation* of a group G is a homomorphism $\rho : G \to \mathcal{U}(\mathcal{H})$ to the unitary group which is strongly continuous. That is, such that the map

$$G \to \mathcal{H}, \quad g \mapsto \rho(g)\xi$$

is continuous for every $\xi \in \mathcal{H}$ Such a representation is said to be *irreducible* if $\{0\}$ and $\mathcal{H}$ are the only $\rho(G)$-invariant subspaces of $\mathcal{H}$.

We will see in Section 3 an example of this. There we work with $\mathcal{H} = L^2(\Gamma/N)$ where Γ is a compact group, and N is a family of finite index normal subgroups of Γ. Then the representation $\rho\colon \Gamma \to \mathcal{U}(\mathcal{H})$ is defined by $\rho(\gamma)f(x) = f(x\gamma)$ for $\gamma \in \Gamma$, and $f \in \mathcal{H}$.

2.6 The Kazhdan property T

The Kazhdan property T was first defined in the mid 60's by D. Kazhdan. The first definition used unitary representations, and only required a limited representation theoretic background. There are other equivalent definitions, namely one using the Fell topology, but we do not need this definition for our purposes in this paper.

First, we want to introduce the definitions of an invariant and almost invariant vector.

Definition 2.9. An invariant vector is a vector that is not affected by the group action. That is, a vector ξ is invariant with regards a group G, if for every $g \in G$, $g\xi = \xi$.

A vector is called *almost invariant*, if there exists $\epsilon > 0$, such that for every compact subset $K \subset G$, $||g\xi - \xi|| < \epsilon||\xi||,$ for all $g \in K$.

Now we have what is needed for our formulation of the Kazhdan property T. This formulation is used in both [3] and [1].

Definition 2.10. A locally compact group Γ is called a Kazhdan group or is said to have property T if and only if every unitary representation which has almost invariant vectors, also contains a nonzero invariant vector.

3 Results

We will focus on one main result (Theorem 3.3), which is presented in [3]. Our presentation of the proof will be the same in principle, but we will add some two lemmas before the proof, which Lubotzky has left out. The theorem details how to construct expanders, and is a variant of the proof given by Margulis, which was a breakthrough in the study of expanders. We begin with the first lemma.

Lemma 3.1. *Let Γ be a finitely generated Kazhdan group. Let $\mathscr{L}$ be a family of finite index normal subgroups of Γ. Now let $N \in \mathscr{L}$ and $H = L^2(\Gamma/N)$ be the vector space of the complex valued functions on the finite set $V = \Gamma/N$, with the norm $||f||^2 = \sum_{x\in V} |f(x)|^2$. Let $H_0 = \{f \in H \mid \sum_{x\in V} f(x) = 0\}$. Then for $\gamma \in \Gamma$, Γ acts on H by $(\gamma f)(x) = f(x\gamma)$. Using that $\mathbb{C}_{\mathbf{1}_V}$ is the set of constant functions over V, we get $H = H_0 \oplus \mathbb{C}_{\mathbf{1}_V}$.*

Proof. Let $f \in H$. Then set $a = \frac{\sum_{x\in V} f(x)}{|V|}$. We then let $f' = f - a \cdot \mathbf{1}_V$. Then f' is in H_0, because $||f'|| = \sum_{x\in V}(f(x) - a \cdot \mathbf{1}_V) = 0$. Thus we have shown that we can construct any $f \in H$ as $f = f' + a \cdot \mathbf{1}_V$ for some $f' \in H_0,\ a \cdot \mathbf{1}_V \in \mathbb{C}_{\mathbf{1}_V}$.

□

Lemma 3.2. *Let Γ, V, and the group action of Γ on V be as in Lemma 3.1. Then the only Γ-invariant functions on V are the constants $\mathbb{C}_{\mathbf{1}_V}$.*

Proof. A group action of Γ on V is transitive if we have that for any $x, y \in V$, there exists $\gamma \in \Gamma$ such that $\gamma x = y$. This is true for our group action since for any $x, y \in V$, we can find a $\gamma \in \Gamma$, such that $\gamma^{-1} = y^{-1}x$. Then this lemma can be proven by showing that if a function is Γ-invariant, then it must also be constant.

Assume that $f \in H$ is Γ-invariant. Then for any $f \in H$, $\gamma \in \Gamma$, we have $\gamma f = f$. Since Γ is transitive, a $\gamma \in \Gamma$ such that $\gamma x = y$. Combining these, with the action of Γ on H, we get that for any $x, y \in V$, we can find γ such that

$$\begin{aligned} \gamma f(x) &= f(x) \\ f(x\gamma) &= f(x) \\ f(y) &= f(x), \end{aligned}$$

and thus f must be the constant function, i.e. $f \in \mathbb{C}_{\mathbf{1}_V}$.

□

Theorem 3.3. *Let Γ, $\mathscr{L}$, $\mathcal{H}$ be as in Lemma 3.1. Let S be the (considered as a fixed finite symmetric (i.e., $S^{-1} = S$) set of generators for Γ. Then the family of $X(\Gamma/N, S)$, the Cayley graphs of the finite groups Γ/N, for $N \in \mathscr{L}$, with respect to S (considered as a set of generators of Γ/N) is a family of (n, k, c)-expanders for some $c > 0$, $k = |S|$, and $n = |\Gamma/N|$.*

Proof. From Lemma 3.2 we have that the only Γ-invariant functions are constant, and so the only invariant function in H_0 is the zero function. Then H_0 does not contain any non-zero invariant funtions. Thus, because Γ is Kazhdan, we can use the definition of the Kazhdan property T in Definition 2.10 and get that H_0 does not have any almost invariant functions. In particular, this implies that there are no almost invariant subsets. This is equivalent to saying that we have an expander graph. To be precise, we recall Definition 2.9 and use that every subset is compact since they are finite. Then there exists an $\epsilon > 0$, only dependent on Γ and S, not on N, such that for every $f \in H_0$, $||\gamma f - f > \epsilon||f||$ for some $\gamma \in S$. Now, let A be a subset of V of size a, and B its compliment of size $b = n - a$. Let

$$f(x) = \begin{cases} b \text{ if } x \in A \\ -a \text{ if } x \in B. \end{cases}$$

Then $f \in H_0$, and

$$||f||^2 = ab^2 + ba^2 = nab$$

while for every $\gamma \in S, ||\gamma f - f||^2 = (b + a)^2|E_\gamma(A, B)|$ where $E_\gamma(A, B) = \{x \in V | x \in A \text{ and } x\gamma \in B \text{ or } x \in B \text{ and } \gamma x \in A\}$. Then finally we have that

$$|\partial A| \geq \frac{1}{2}|E_\gamma(A, B)| = \frac{||\gamma f - f||^2}{2n^2} \geq \frac{\epsilon^2||f||^2}{2n^2} = \epsilon^2 \frac{ab}{2n} = \frac{\epsilon^2}{2} \cdot (1 - \frac{|A|}{n})|A|.$$

Thus $X(\Gamma/N, V)$ are expanders with $c \geq \frac{\epsilon^2}{2}$. □

4 The special linear group as an expander

From Theorem 3.3, we have an explicit construction of an expander. Now it is only natural to look at a specific example. We take an example from Lubotzky [3], and take it a step further. While Lubotzky describes the graph, we aim to also program it, so that we can estimate the expansion factor c. First, we want to define the family of expanders we will compute.

We will work with the special linear group $\Gamma_n = SL(n, \mathbb{Z})$ for $n \geq 3$, which is the group of $n \times n$ matrices with determinant 1, equipped with the operation of matrix multiplication. Let $SL_n(p, \mathbb{Z})$ be the group of $n \times n$ matrices over the finite field $\mathbb{Z}/p\mathbb{Z}$ with determinant 1. Now let

$$A_n = \begin{pmatrix} 1 & 1 & \vdots & 0_{2\times(n-2)} \\ 0 & 1 & \vdots & \\ \dots & \dots & \ddots & \dots \\ & 0_{(n-2)\times 2} & \vdots & I_{n-2} \end{pmatrix}, \; B_n = \begin{pmatrix} & 0 & 1 & & & & \\ & 0 & 0 & 1 & & & \\ & & & 0 & 1 & & \\ & & & & \ddots & \ddots & \\ & & & & & \ddots & 1 \\ (-1)^{(n-1)} & & & & & & 0 \end{pmatrix}.$$

Then one can show that $S_n = \{A_n, B_n, A_n^{-1}, B_n^{-1}\}$ is a generator for Γ_n. We do not show this, but a proof is provided in [2]. Thus S_n is also a generator for $SL_n(p, \mathbb{Z})$. Then, we can use the method we saw of drawing a Cayley graph in Definition 2.5. This means that every matrix in $SL_n(p, \mathbb{Z})$ is a vertex in V, and the elements in S_n are the edges. We can calculate the neighbors of a matrix by multiplying it with the edges. Now $X(SL_n(p, \mathbb{Z}), S_n)$ is a family of expanders, when n is fixed and p runs over all prime numbers. The idea of this relates to the finite index normal subgroups $\Gamma_n = \ker(SL_n(\mathbb{Z} \to SL_n(p, \mathbb{Z})$. This means in particular that there will be a constant expansion factor c which holds for all of the graphs when $(n \to \infty)$.

4.1 Size of $SL_n(p, \mathbb{Z})$

For computational purposes it is useful to know the size of a graph, i.e. $SL_n(p, \mathbb{Z})$, in a faster way than to generate a whole graph and count the components.

The proofs of the formulas are from [4]. They have been slightly altered, but the main ideas remain the same.

To get the formula for $SL_n(p, \mathbb{Z})$, we first look at how big the general linear group $GL_n(p, \mathbb{Z})$ is. This is the group of invertible matrices, together with the ordinary operation of matrix multiplication, i.e. matrices with determinant not equal to 0. Notice that $SL_n(p, \mathbb{Z})$ is the special case where the determinant is 1.

Lemma 4.1. *$GL_n(p, \mathbb{Z})$ contains*

$$(p^n - 1)(p^n - p) \dots (p^n - p^{n-1})$$

matrices.

Proof. The size of the groups are decided by how many matrices of unique inputs we can create. The only requirement is that the determinant cannot be 0. We can go row by row, for n rows.

- row 1 can be any vector but the 0-vector, so we have $(p_n - 1)$ possibilities,

- row 2 can be anything but a multiple of the first row. Since there are p possible multiples, we get $(p^n - p)$ possibilities,
- row 3 cannot be a linear combination of the first two rows. There is p^2 variations of the constants in a linear combination $a_1\mathbf{v}_1 + a_2\mathbf{v}_2$, thus we have $(p^n - p^2)$ possibilities in this row,
- ...
- row n can't be a linear combination of all of the rows above, therefore we have $(p^n - p^{n-1})$ combinations.

Thus $GL_n(p, \mathbb{Z})$ contains $(p^n - 1)(p^n - p)\dots(p^n - p^{n-1})$ matrices. □

We can use this to find the formula for $SL_n(p, \mathbb{Z})$, by creating bijections between matrices with determinant 1, and matrices with determinants different to 1.

Proposition 4.2. *The size of $SL_n(p, \mathbb{Z})$ can be determined by the formula*

$$\frac{(p^n - 1)(p^n - p)\dots(p^n - p^{n-1})}{p - 1}.$$

Proof. First, we fix a matrix in $GL_n(p, \mathbb{Z})$ with determinant 1, i.e. a matrix in $SL_n(p, \mathbb{Z})$. Now we can map this matrix into a matrix with determinant q, for $q \in \mathbb{Z}/p\mathbb{Z},\ q \neq 0$, by multiplying the first row by q. This can be done for any matrix we choose. Next, Fermat's little theorem gives that $q^{n-1} \equiv 1 \bmod p$, and so if we multiply the first row of a matrix with determinant q by q^{n-2}, it is equivalent to multiplying the first row of a matrix with determinant 1 by q^{n-1}, which as we saw is the same as multiplying by 1. Thus we can map from a matrix with determinant q to determinant 1. These two maps are injective, and thus we have a bijection between matrices with determinant 1 and matrices with determinant q. This means that the number of matrices with determinant 1 is equal to the number of matrices of determinant q. Thus $GL_n(p, \mathbb{Z})$ is partitioned into $p - 1$ sets of equal size. Thus, the size of $SL_n(p, \mathbb{Z})$ is $\frac{(p^n-1)(p^n-p)\dots(p^n-p^{n-1})}{p-1}$. □

We can now look at some examples of the sizes we could be working with, to get an idea of the computing needed for later. Looking at $SL_2(p, \mathbb{Z})$ we get 336 matrices for $p = 7$, 1 320 for $p = 11$, and 148 824 for $p = 53$. For $SL_3(p, \mathbb{Z})$ we get a size of 5 630 688 for $p = 7$, 212 427 600 for $p = 11$, and 62 237 108 003 616 for $p = 53$.

The scale of the graphs when $n \geq 3$ means that we will restrict ourselves to the case where $n = 2$ for the code. Strictly speaking, Theorem 3.3 does not prove that $X(SL_2(p, \mathbb{Z}), S_2)$ is a family of expanders, since $SL_2(\mathbb{Z})$ does not have property T, but it is still possible to prove this by other means. We can use property τ, which while it is beyond the scope of this paper, means that we can still look at $X(SL_2(p, \mathbb{Z}), S_2)$ and know that it is a family of expanders, and thus estimate its expansion factor. The code will work for $n \geq 3$, and so it is still relevant when looking at larger matrices.

4.2 Overview of code

Since we know that $X(SL_n(p, \mathbb{Z}), S_n)$ is a family of expanders, we can now describe the graph programmatically, and estimate the expansion factor. We will use the equation in Definition 2.1, and change it to calculate c. That is,

$$c = \min_{A \in V} \frac{|\partial A|}{\left(1 - \frac{|A|}{|V|}\right)|A|}. \tag{2}$$

Thus our aim is to find subsets which yields a small c, for a fixed n and several primes p, and try to look for patterns which could hint at the true minimum subset over all the different p. Therefore our program will consist of several scripts. The different scripts we are using will:

- calculate subsets of a given n and p,
- calculate the boundary of a given subset,
- calculate the expansion factor of a given subset and boundary,
- estimate the expansion factor of $SL_n(p, \mathbb{Z})$ for a given n and p, by utilizing the above scripts,
- plot the estimated expansion factors for several values of p, by the above scripts.

The next sections will explain each of these scripts, along with giving their pseudocode.

4.3 Script 1: Generating subset

The very first question we encounter is *how* we want the subset to be generated. This is because there are different processes which could be applied, that gives different "shapes" to the subsets. Therefore it is useful to think about what we actually want to achieve in choosing the subset, and what shape would be best for achieving this. Proposition 2.3 implies that there is a connection between the expansion factor, and the Cheeger constant. This means that there is a connection between the expansion factor and how difficult it is to separate the subset from the whole graph. Specifically, the expansion factor will generally be smaller when it is easier to remove the subset from the whole graph. So the fewer nodes you have to remove, relative to the size of the subset and the whole graph minus the subset, the smaller expansion factor you will get.

This means that generating a straight line of vertices (for example by using a depth first search) would yield a very high expansion factor, as it is hard to remove the whole subset from the rest of the graph. Thus this is not the generation we are looking after. An example of this is included in Figure 3.

Our decision stood between simply generating the subset from a specific matrix using a breadth first search, and generating a subset by continually choosing random vertices to continue building from. They both would stop when the subset got to a predetermined size.

The first method works by starting from the edges (we found that the constant did not change depending on which matrices we began calculating from), and using the edges in S_n to calculate all of the neighbours, and then using a first-in first-out approach to continue this process. That is, we use a queue to store all the matrices we shall find the neighbours of. This means that the subset we get is shaped more like a circle than a line, as we can see in Figure 2.

The second method which was considered uses this same principle, but we continually choose a random starting vertex within the subset which we then calculate a smaller subset around. This gives a more random shape.

When running these codes, we found that the first approach always got a smaller expansion factor. One reason could be that we get a structure which looks more like the circle. We know that the circle has the smallest circumference compared to its area. The subset then could be equated to the area of our circle, and the boundary to its circumference, then the boundary is smallest compared to the subset when the subset looks as close to a circle as possible.

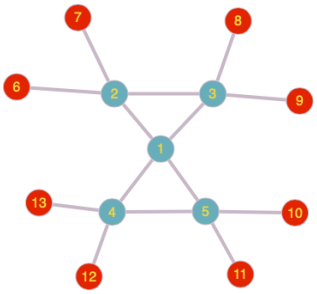

Figure 2: Here we have started from node 1, and generated our subset using breadth first search. The blue nodes are included in the subset, and the red are the boundary of the subset. For this subset, we have 5 nodes in the subset, and 8 nodes in the boundary, which would need to be removed to isolate our subset.

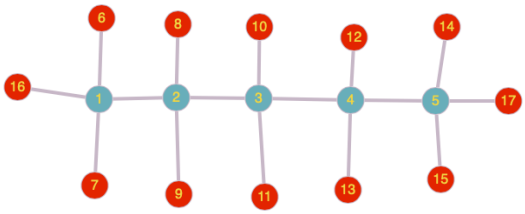

Figure 3: Here we started from node 1, and used a depth first search to generate the subset. The blue nodes are in the subset, and the red are in the boundary. We have 5 nodes in the subset, and 12 in the boundary.

Remark 4.3. Note that we do not start generating many different subsets from completely random starting matrices. This is because many of the subsets would not intersect, greatly increasing the size of the boundary.

These graphs were drawn in https://graphonline.ru/en/#. As we can see, the boundary of the straight subset is larger than that of the more circular subset.

Algorithm 1 An algorithm for generating subset of a given size

Input: Positive integer n, prime p, integer size less than $|SL_n(p, \mathbb{Z})|$
Output: A subset of matrices

```
edges ← edges of the graph, i.e. the generating subset S_n
queue ← queue consisting of the edges
subset ← empty subset
while queue is not empty do
    X ← queue.dequeue()                          ▷ pops first element of queue
    Xes ← list consisting of X × edge for each edge in edges
    Xes ← Xes mod p
    for Xe in Xes do
        if Xe is not in subset then
            add Xe to subset
            add Xe to queue
        end if
        if size of subset = size then
            return subset
        end if
    end for
end while
```

Remark 4.4. This code could also be used to generate the whole set $SL_n(p, \mathbb{Z})$, by not stopping at a given size and instead waiting for the queue to be empty.

In practice we implemented the set as a set of tuples, where a matrix is converted into a tuple when before adding it to the set. We then convert back to a matrix when doing matrix multiplication. This saves a lot of time, as a tuple can be compared to other tuples much faster than a matrix to other matrices. We have done some more such steps to help with the runtime, but we will not go into detail in this paper.

4.4 Script 2: Generating the boundary of a subset

The next script returns the boundary of a given subset. This works by going through all the vertices in the subset, and then calculating its neighbors. If a neighbor is not in the subset, it will be added to the boundary set.

Algorithm 2 An algorithm for returning the boundary of a subset

Input: A set of matrices called subset
Output: The boundary of a subset
  boundary ← empty set
  edges ← edges of the graph, i.e. the generating subset S_n
  for X in subset **do**
    Xes ← list consisting of $X \times edge$ for each edge in edges
    Xes ← Xes mod p
    for Xe in Xes **do**
      if Xe is not in subset **then**
        add Xe to boundary
      end if
    end for
  end for
  return boundary

4.5 Script 3: Calculating the expansion factor of a subset

This script will calculate the expansion factor for a single subset, by using Equation (3) and removing the restriction of the minimum subset, so

$$c = \frac{|\partial A|}{\left(1 - \frac{|A|}{|V|}\right)|A|}, \tag{3}$$

for $A \in V$.

The subset only varies by the integer size. This is because as we recall from Section 4.3, it is the only variable which changes the factor.

Algorithm 3 An algorithm for returning the expansion factor c

Input: An integer n, integer p, integer size
Output: A float c
  degree ← size of $SL_n(p, \mathbb{Z})$
  subset ← subset returned from Algorithm 1 with inputs n, p, and size
  boundary ← boundary returned from Algorithm 2 with input subset
  $c \leftarrow \frac{|boundary|}{(1-\frac{|subset|}{degree})|subset|}$
  return c

4.6 Script 4: Getting the expansion factors over several graphs

This script aims to gather the expansion factors of different graphs and different subsets within those graphs. This is so that we can later look at the differences in the expansion factor depending on the primes we look at, and depending on the how big the subset is, relative to the entire graph. We have chosen to gather the data in a nested list, so that the first index relates to the size of the subset, while the second index relates to the prime we are looking at.

Algorithm 4 An algorithm for plotting expansion factors over several graphs

Input: An integer n, integer p, integer l, float ratiostart, float ratiostop, $0 <$ ratiostart $<$ ratiostop < 1

Output: A nested list containing floats c

```
ratios ← list of equally spaced ratios, from ratiostart to ratiostop, containing l floats
primes ← list of primes from 7 to p
factors ← empty list
for ratio in ratios do
    temporary ← empty list
    for p in primes do
        size ← ratio times size of SL_n(p, ℤ), converted to integer value
        c ← c returned from Algorithm 3 with inputs n, p, and size
        add c to temporary
    end for
    add temporary to factors
end for
return factors
```

We can now plot the result gathered from this. We have opted to do this by plotting with matplotlib.pyplot, and also by printing a table with pandas. We will go through the results in the next section.

4.7 Figures and conclusion

Below are figures which present the data gathered. In Figure 4, we have chosen to compute to the prime p=97, and chosen 30 equally spaced sizes of the subsets, ranging from $\frac{|A|}{|V|} = 0.01$, to $\frac{|A|}{|V|} = 0.5$. As we can tell from the table, the smallest overall value of c is at $p = 43$, and $\frac{|A|}{|V|} = 0.29$. We can also see that the expansion factor is generally smaller in $p = 43$, especially around $\frac{|A|}{|V|} = 0.3$.

|A|/|V|	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97
0.010000	3.363363	2.019893	1.202034	0.841584	0.772385	0.717897	0.665071	0.659783	0.646782	0.640115	0.638623	0.637031	0.632670	0.630759	0.629476	0.627699	0.627572	0.627514	0.626676	0.626521	0.626010	0.625764
0.027500	2.511723	0.913811	0.822599	0.721232	0.700093	0.673112	0.651679	0.651150	0.644602	0.641722	0.640965	0.639379	0.638273	0.634887	0.637729	0.637062	0.636820	0.636483	0.635505	0.629288	0.631905	0.628234
0.045000	1.814330	0.851624	0.769210	0.704378	0.682080	0.671203	0.659222	0.657671	0.652084	0.651446	0.650232	0.650155	0.635746	0.640239	0.648514	0.647107	0.642499	0.628798	0.631680	0.624747	0.636479	0.626971
0.062500	1.275949	0.793175	0.744916	0.699301	0.691905	0.675402	0.667178	0.665567	0.663667	0.662195	0.661561	0.639500	0.635571	0.646382	0.655402	0.643189	0.648183	0.624166	0.637457	0.623622	0.634808	0.624795
0.080000	1.083871	0.765667	0.730623	0.697660	0.695471	0.685053	0.677371	0.677730	0.674881	0.667951	0.640512	0.636829	0.639296	0.642642	0.650627	0.640108	0.653723	0.612174	0.643529	0.623807	0.631202	0.621092
0.097500	0.898026	0.744023	0.720923	0.706110	0.703648	0.694390	0.656886	0.689784	0.687047	0.666956	0.634320	0.634506	0.647255	0.629570	0.651075	0.639488	0.652897	0.605354	0.648317	0.626801	0.627366	0.617951
0.115000	0.979159	0.747795	0.729226	0.716494	0.707223	0.707392	0.651491	0.701994	0.681836	0.661901	0.632112	0.636740	0.655356	0.620985	0.653412	0.636337	0.651314	0.601955	0.651057	0.627195	0.626716	0.613735
0.132500	0.889166	0.754649	0.741752	0.725675	0.706112	0.720008	0.647969	0.715668	0.669699	0.666676	0.626727	0.636477	0.656602	0.618401	0.654618	0.637040	0.653183	0.597987	0.651418	0.627484	0.623467	0.611757
0.150000	0.916364	0.748663	0.740901	0.730816	0.712074	0.735171	0.641038	0.723698	0.668499	0.671358	0.611771	0.637813	0.657035	0.615090	0.653111	0.641966	0.653814	0.594435	0.649347	0.628191	0.619324	0.609558
0.167500	0.900000	0.739133	0.740133	0.735355	0.706996	0.736421	0.644164	0.733615	0.669598	0.669886	0.604750	0.635496	0.661701	0.618850	0.650043	0.648083	0.651283	0.589316	0.649596	0.630442	0.616700	0.610156
0.185000	0.870261	0.749132	0.753187	0.740123	0.719655	0.730358	0.642614	0.739076	0.667931	0.671939	0.597964	0.632794	0.665726	0.618729	0.647021	0.653396	0.644708	0.587985	0.646264	0.632621	0.612442	0.611970
0.202500	0.774363	0.732418	0.754508	0.740121	0.725177	0.723069	0.647777	0.744515	0.675334	0.671367	0.593797	0.625940	0.670227	0.618529	0.647247	0.653913	0.639504	0.585621	0.645436	0.634303	0.613158	0.613768
0.220000	0.787541	0.733579	0.771684	0.748733	0.720193	0.728571	0.652143	0.751169	0.676846	0.669065	0.592477	0.623050	0.672565	0.625045	0.645775	0.655008	0.634054	0.584666	0.641398	0.636835	0.611390	0.614913
0.237500	0.777816	0.728701	0.774407	0.753768	0.735615	0.731209	0.660141	0.758904	0.679025	0.669406	0.587628	0.617152	0.675974	0.627348	0.645771	0.654865	0.629358	0.582478	0.641108	0.640480	0.610892	0.616103
0.255000	0.708695	0.750581	0.801054	0.769990	0.739611	0.736115	0.672300	0.765222	0.682839	0.672129	0.589361	0.615692	0.672873	0.632092	0.646294	0.653783	0.628047	0.583576	0.638614	0.643979	0.613661	0.616262
0.272500	0.723391	0.753741	0.820050	0.785139	0.746545	0.741891	0.671958	0.769901	0.685475	0.672151	0.582372	0.617280	0.670165	0.634743	0.645146	0.653439	0.624787	0.586624	0.638909	0.646552	0.613364	0.617781
0.290000	0.724669	0.751515	0.831972	0.782945	0.759887	0.738362	0.682078	0.770308	0.688188	0.675701	0.577090	0.616358	0.671421	0.641253	0.644080	0.651948	0.622728	0.588125	0.641299	0.648131	0.616241	0.620731
0.307500	0.700029	0.776523	0.851896	0.798179	0.765576	0.738240	0.693652	0.773221	0.684623	0.677151	0.580098	0.615339	0.671200	0.643348	0.649140	0.652962	0.626395	0.591647	0.640367	0.649513	0.616891	0.623309
0.325000	0.733298	0.784858	0.877129	0.803546	0.763911	0.724151	0.695034	0.774021	0.689673	0.677489	0.579906	0.613420	0.669597	0.644558	0.651323	0.654164	0.626847	0.593254	0.640126	0.650931	0.617179	0.625351
0.342500	0.753571	0.800742	0.874309	0.801980	0.770727	0.715996	0.706939	0.774452	0.696081	0.678863	0.579782	0.609509	0.672790	0.646650	0.652317	0.653063	0.626780	0.594833	0.645484	0.651883	0.619941	0.629309
0.360000	0.725926	0.812308	0.884470	0.805049	0.766583	0.701280	0.710216	0.779548	0.696639	0.681171	0.587159	0.606360	0.672609	0.649711	0.657603	0.653174	0.630532	0.596401	0.646482	0.653847	0.622964	0.632392
0.377500	0.761905	0.799695	0.873101	0.817057	0.767732	0.692090	0.710489	0.787037	0.698821	0.686506	0.586932	0.607670	0.670723	0.649757	0.660407	0.654614	0.633535	0.599479	0.646188	0.656115	0.622558	0.635357
0.395000	0.798574	0.824447	0.870101	0.824907	0.772139	0.685751	0.708422	0.793605	0.708918	0.693237	0.590368	0.606538	0.669279	0.653463	0.661723	0.654855	0.635108	0.601661	0.650396	0.656932	0.624974	0.638790
0.412500	0.811594	0.844262	0.876926	0.836136	0.773434	0.683667	0.716197	0.797820	0.716752	0.697793	0.593848	0.604906	0.672066	0.656458	0.663872	0.655192	0.638340	0.603248	0.655962	0.658772	0.629785	0.642010
0.430000	0.802083	0.859490	0.892988	0.840021	0.776042	0.682712	0.719535	0.799828	0.721272	0.702683	0.597898	0.606399	0.673745	0.658873	0.667733	0.656956	0.639986	0.608066	0.658311	0.663274	0.632867	0.644917
0.447500	0.818925	0.867332	0.883424	0.847651	0.767569	0.675441	0.724740	0.807838	0.728440	0.706633	0.598718	0.606051	0.672388	0.659168	0.670677	0.656532	0.636834	0.610676	0.661103	0.667633	0.635586	0.648304
0.465000	0.837607	0.877176	0.883514	0.859636	0.767522	0.669302	0.729357	0.814748	0.740631	0.713777	0.599435	0.607085	0.671003	0.661356	0.672784	0.654577	0.635419	0.612240	0.666142	0.669837	0.639486	0.651234
0.482500	0.870158	0.901192	0.883911	0.857267	0.778737	0.670786	0.742127	0.812421	0.757448	0.718070	0.604889	0.611373	0.675208	0.663525	0.675409	0.654312	0.635834	0.614816	0.670255	0.673131	0.643030	0.655697
0.500000	0.857143	0.933333	0.890110	0.859477	0.779532	0.683794	0.751396	0.814247	0.765924	0.721951	0.606564	0.613398	0.677767	0.663959	0.678724	0.655985	0.638766	0.617847	0.673215	0.676406	0.647293	0.660342

Figure 4: Pandas dataframe of the result, with the columns signifying which prime we use, and the rows signifying how big our subset is compared to the whole graph. The smallest values of c for each p is highlighted in red.

Remark 4.5. When investigating the data further, we found no indications of a new minimum being reached around this before $\frac{|A|}{|V|} = 0.01$, or after $\frac{|A|}{|V|} = 0.5$. Thus the range we are using should be big enough to include our minimum.

In Figure 5, we have chosen to use the maximum prime as $p = 101$, and looked at 24 equally spaced sizes of the subsets, ranging from $\frac{|A|}{|V|} = 0.01$, to $\frac{|A|}{|V|} = 0.9$. In this case, one can get a better overwiev of how the expansion factor developes, and try to notice any lower points on the plot.

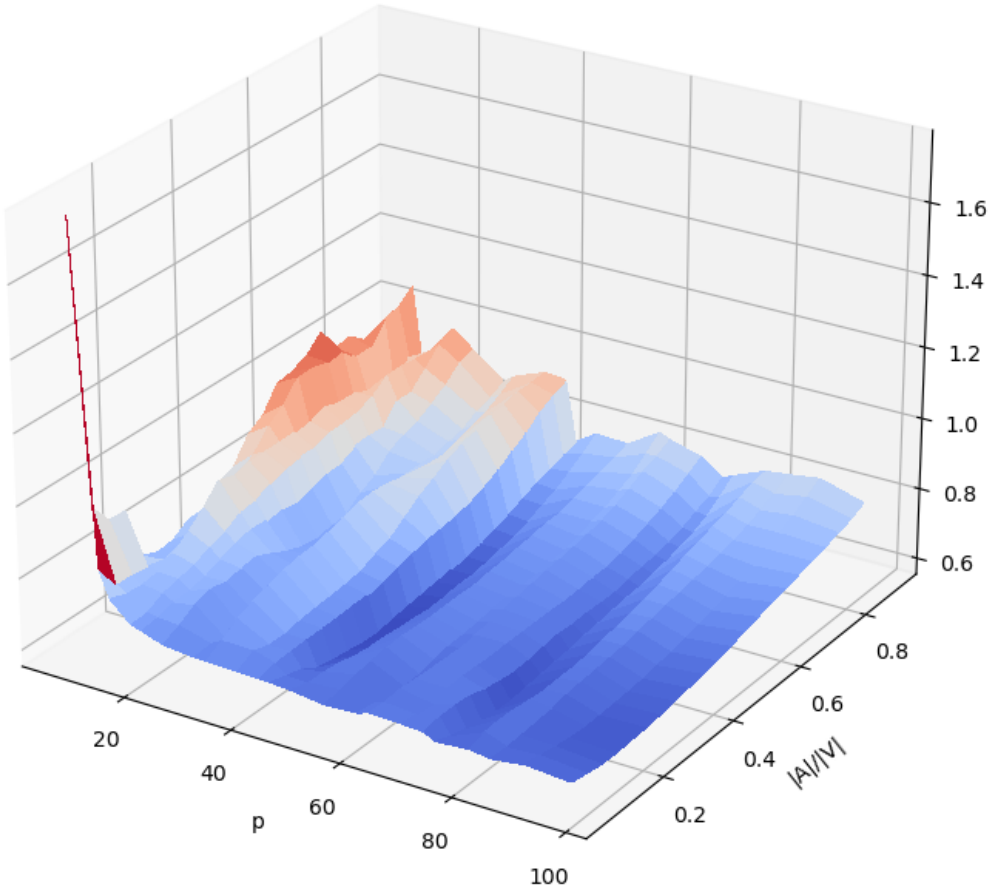

Figure 5: 3d plot of the expansion factor on the z-axis. The primes are on the x-axis, and sizes of the subset are on the y-axis.

Here, one can more easily see what was found in Figure 4. The factors seem to be generally smaller in $p = 43$, and also in $p = 73$. There is no way to be sure that $p = 43$ contains a global minimum, but the plot gives no clear indication of getting smaller for bigger primes, especially since $p = 43$ gave the biggest drop in expansion factors from the previous primes, and the slope seemed to get much straighter as p got larger.

References

[1] Bekka, B., Harpe, P. de la and Valette, A. *Kazhdan's Property (T)*. New Mathematical Monographs. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511542749.

[2] Eitzen, H. von. *About the special linear group*. 2015. URL: https://math.stackexchange.com/questions/1234631/ (visited on 24/05/2023).

[3] Lubotzky, A. *Discrete Groups, Expanding Graphs and Invariant Measures*. Birkhäuser Verlag, 1994, pp. ix+195.

[4] Nicolas, A. *Order of general- and special linear groups over finite fields*. 2011. URL: https://math.stackexchange.com/questions/34271 (visited on 24/05/2023).