Spectral Graph Theory Lecture 9

Testing Isomorphism of Strongly Regular Graphs

Daniel A. Spielman September 26, 2018

9.1 Introduction

In the last lecture we saw how to test isomorphism of graphs in which every eigenvalue is distinct. So, in this lecture we will consider the opposite case: graphs that only have 3 distinct eigenvalues. These are the strongly regular graphs.

Our algorithm for testing isomorphism of these will not run in polynomial time. Rather, it takes time $n^{O(n^{1/2} \log n)}$. This is at least much faster than the naive algorithm of checking all $n!$ possible permutations. In fact, this was the best known running time for general algorithms for graph isomorphism until three years ago.

9.2 Definitions

A graph G is strongly regular if

1. it is d-regular, for some integer d;
2. there exists an integer α such that for every pair of vertices x and y that are neighbors in G, there are exactly α vertices z that are neighbors of *both* x and y;
3. there exists an integer β such that for every pair of vertices x and y that are *not* neighbors in G, there are exactly β vertices z that are neighbors of *both* x and y.

These conditions are very strong, and it might not be obvious that there are any non-trivial graphs that satisfy these conditions. Of course, the complete graph and disjoint unions of complete graphs satisfy these conditions. Before proceeding, I warn you that there is a standard notation in the literature about strongly regular graphs, and I am trying not to use it. In this literature, d becomes k, α becomes λ and β becomes μ. Many other letters are bound as well.

For the rest of this lecture, we will only consider strongly regular graphs that are connected and that are not the complete graph. I will now give you some examples.

9.3 Paley Graphs and The Pentagon

The Paley graphs we encountered are strongly regular. The simplest of these is the pentagon. It has parameters

$$n = 5, \quad d = 2, \quad \alpha = 0, \quad \beta = 1.$$

9.4 Lattice Graphs

For a positive integer n, the *lattice graph* L_n is the graph with vertex set $\{1, \dots n\}^2$ in which vertex (a, b) is connected to vertex (c, d) if $a = c$ or $b = d$. Thus, the vertices may be arranged at the points in an n-by-n grid, with vertices being connected if they lie in the same row or column. Alternatively, you can understand this graph as the product of two complete graphs on n vertices.

The parameters of this graph are:

$$d = 2(n-1), \quad \alpha = n-2, \quad \beta = 2.$$

9.5 Latin Square Graphs

A Latin square is an n-by-n grid, each entry of which is a number between 1 and n, such that no number appears twice in any row or column. For example,

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

are Latin squares. Let me remark that the number of different Latin squares of size n grows very quickly—at least as fast as $n!(n-1)!(n-2)! \dots 2!$. Two Latin squares are said to be *isomorphic* if there is a renumbering of their rows, columns, and entries, or a permutation of these, that makes them the same. As this provides $6(n!)^3$ isomorphisms, and this is much less than the number of Latin squares, there must be many non-isomorphic Latin squares of the same size. The two of the Latin squares above are isomorphic, but one is not.

From such a Latin square, we construct a Latin square graph. It will have n^2 nodes, one for each cell in the square. Two nodes are joined by an edge if

1. they are in the same row,
2. they are in the same column, or
3. they hold the same number.

So, such a graph has degree $d = 3(n-1)$. Any two nodes in the same row will both be neighbors with every other pair of nodes in their row. They will have two more common neighors: the nodes in

their columns holding the other's number. So, they have n common neighbors. The same obviously holds for columns, and is easy to see for nodes that have the same number. So, every pair of nodes that are neighbors have exactly $\alpha = n$ common neighbors.

On the other hand, consider two vertices that are not neighbors, say $(1,1)$ and $(2,2)$. They lie in different rows, lie in different columns, and we are assuming that they hold different numbers. The vertex $(1,1)$ has two common neighbors of $(2,2)$ in its row: the vertex $(1,2)$ and the vertex holding the same number as $(2,2)$. Similarly, it has two common neighbors of $(2,2)$ in its column. Finally, we can find two more common neighbors of $(2,2)$ that are in different rows and columns by looking at the nodes that hold the same number as $(1,1)$, but which are in the same row or column as $(2,2)$. So, $\beta = 6$.

9.6 The Eigenvalues of Strongly Regular Graphs

We will consider the adjacency matrices of strongly regular graphs. Let A be the adjacency matrix of a strongly regular graph with parameters (d, α, β). We already know that A has an eigenvalue of d with multiplicity 1. We will now show that A has just two other eigenvalues.

To prove this, first observe that the (a,b) entry of A^2 is the number of common neighbors of vertices a and b. For $a = b$, this is just the degree of vertex a. We will use this fact to write A^2 as a linear combination of A, I and J, the all 1s matrix. To this end, observe that the adjacency matrix of the complement of A (the graph with non-edges where A has edges) is $J - I - A$. So,

$$A^2 = \alpha A + \beta(J - I - A) + dI = (\alpha - \beta)A + \beta J + (d - \beta)I.$$

For every vector $\boldsymbol{v}$ orthogonal to $\mathbf{1}$,

$$A^2\boldsymbol{v} = (\alpha - \beta)A\boldsymbol{v} + (d - \beta)\boldsymbol{v}.$$

So, every eigenvalue λ of A other than d satisfies

$$\lambda^2 = (\alpha - \beta)\lambda + d - \beta.$$

Thus, these are given by

$$\lambda = \frac{\alpha - \beta \pm \sqrt{(\alpha - \beta)^2 + 4(d - \beta)}}{2}.$$

These eigenvalues are traditionally denoted r and s, with $r > s$. By convention, the multiplicty of the eigenvalue r is always denoted f, and the multiplicty of s is always denoted g.

For example, for the pentagon we have

$$r = \frac{\sqrt{5} - 1}{2}, \quad s = -\frac{\sqrt{5} + 1}{2}.$$

For the lattice graph L_n, we have

$$r = n - 2, \quad s = -2.$$

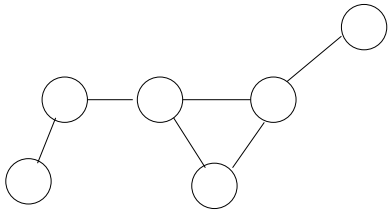
For the Latin square graphs of order n, we have

$$r = n - 3, \quad s = -3.$$

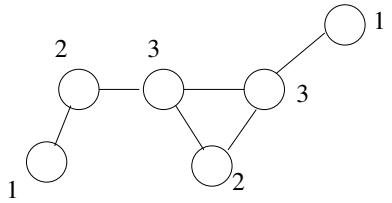
One can prove that every connected regular graph whose adjacency (or Laplacian) matrix has just three distinct eigenvalues is a strongly regular graph.

9.7 Testing Isomorphism by Individualization and Refinement

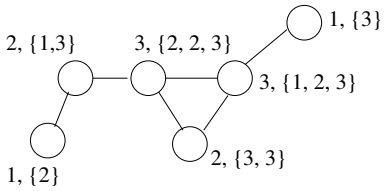
The problem of testing isomorphism of graphs is often reduced to the problem of giving each vertex in a graph a unique name. If we have a way of doing this that does not depend upon the initial ordering of the vertices, then we can use it to test graph isomorphism: find the unique names of vertices in both graphs, and then see if it provides an isomorphism. For example, consider the graph below.



We could begin by labeling every vertex by its degree.



The degrees distinguish between many nodes, but not all of them. We may refine this labeling by appending the labels of every neighbor of a node.



Now, every vertex has its own unique label. If we were given another copy of this graph, we could use these labels to determine the isomorphism between them. This procedure is called *refinement*, and it can be carried out until it stops producing new labels. However, it is clear that this procedure

will fail to produce unique labels if the graph has automorphisms, or if it is a regular graph. In these cases, we need a way to break symmetry.

The procedure called *individualization* breaks symmetry arbitrarily. It chooses some nodes in the graph, arbitrarily, to give their own unique names. Ideally, we pick one vertex to give a unique name, and then refine the resulting labeling. We could then pick another troubling vertex, and continue. We call a set of vertices $S \subset V$ a *distinguishing set* if individualizing this set of nodes results in a unique name for every vertex, after refinement. How would we use a distinguishing set to test isomorphism? Assume that S is a distinguishing set for $G = (V, E)$. To test if $H = (W, F)$ is isomorphic to G, we could enumerate over *every* possible set of $|S|$ vertices of W, and check if they are a distinguishing set for H. If G and H are isomorphic, then H will also have an isomorphic distinguishing set that we can use to find an isomorphism between G and H. We would have to check $\binom{n}{|S|}$ sets, and try $|S|!$ labelings for each, so we had better hope that S is small.

9.8 Distinguishing Sets for Strongly Regular Graphs

We will now prove a result of Babai [Bab80] which says that every strongly regular graph has a distinguishing set of size $O(\sqrt{n}\log n)$. Babai's result won't require any refinement beyond naming every vertex by the set of individualized nodes that are its neighbors. So, we will prove that a set of nodes S is a distinguishing set by proving that for every pair of distinct vertices a and b, either there is an $s \in S$ that is a neighbor of a but not of b, or the other way around. This will suffice to distinguish a and b. As our algorithm will work in a brute-force fashion, enumerating over all sets of a given size, we merely need to show that such a set S exists. We will do so by proving that a random set of vertices probably works.

I first observe that it suffices to consider strongly-regular graphs with $d < n/2$, as the complement of a strongly regular graph is also a strongly regular graph (that would have been too easy to assign as a homework problem). We should also observe that every strongly-regular graph has diameter 2, and so $d \geq \sqrt{n-1}$.

Lemma 9.8.1. *Let $G = (V, E)$ be a connected strongly regular graph with n vertices and degree $d < n/2$. Then for every pair of vertices a and b, there are at least $d/3$ vertices that are neighbors of a but not b.*

Before I prove this, let me show how we may use it to prove the theorem. This lemma tells us that there are at least $\sqrt{n-1}/3$ nodes that are neighbors of a but not of b. Let T be the set of nodes that are neighbors of a but not neighbors of b. So, if we choose a vertex at random, the probability that it is in T is at least

$$\frac{|T|}{n} \geq \frac{\sqrt{n-1}}{3n} \geq \frac{1}{3\sqrt{n+2}}.$$

If we choose a set S of $3\sqrt{n+2}\ln n^2$ vertices at random, the probability that none of them is in T is

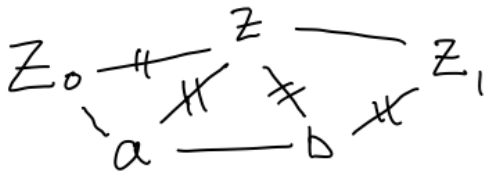
$$\left(1 - \frac{1}{3\sqrt{n+2}}\right)^{3\sqrt{n+2}\ln n^2} \leq \frac{1}{n^2}.$$

So, the probability that a random set of this many nodes fails to distinguish all $\binom{n}{2}$ pairs is at most $1/2$.

Proof of Lemma 9.8.1. Write $a \sim b$ if a is a neighbor of b, and $a \not\sim b$ otherwise. If $a \sim b$, then the number of nodes that are neighbors of a but not of b is $d-1-\alpha$, and if $a \not\sim b$ the number if $d-\beta$. So, we need to prove that neither α nor β is too close to d.

We will do this by establishing some elementary relations between these parameters. First, consider the case in which $a \sim b$. Let z be any vertex such that $a \not\sim z$ and $b \not\sim z$. We will use z to prove an upper bound on the number of vertices w that are neighbors of a but not of b (I know this looks like the wrong direction, but be patient). Let

$$Z_0 = \{w : w \sim a, w \not\sim z\}, \quad \text{and} \quad Z_1 = \{w : w \not\sim b, w \sim z\}.$$



Clearly, every w that is a neighbor of a but not of b lies in either Z_0 or Z_1. As z is neither a neighbor of a nor of b,

$$|Z_0| = |Z_1| = d - \beta.$$

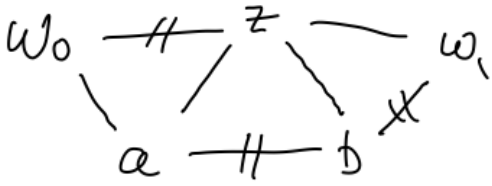
So,

$$d - \alpha - 1 \leq 2(d-\beta) \quad \Longrightarrow \quad 2\beta \leq d + \alpha + 1. \tag{9.1}$$

So, if β is close to d, α must also be close to d.

We will obtain an inequality in the other direction when $a \not\sim b$ by exploiting a z such that $z \sim a$ and $z \sim b$. Now, for any $w \sim a$ but $w \not\sim b$, we have either

$$(w \sim a \text{ and } w \not\sim z) \text{ or } (w \sim z \text{ and } w \not\sim b).$$

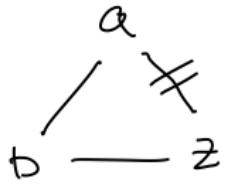


So,

$$d - \beta \leq 2(d - \alpha - 1) \quad \Longrightarrow \quad 2(\alpha+1) \leq d + \beta. \tag{9.2}$$

This tells us that if α is close to d, then β is also.

We require one more relation between α and β. We obtain this relation by picking any vertex a, and counting the pairs b, z such that $b \sim z$, $a \sim b$ and $a \not\sim z$.



Every node b that is a neighbor of a has α neighbors in common with a, and so has $d - \alpha - 1$ neighbors that are not neighbors of a. This gives

$$|\{(b, z) : b \sim z, a \sim b, a \not\sim z\}| = d(d - \alpha - 1).$$

On the other hand, there are $n - d - 1$ nodes z that are not neighbors of a, and each of them has β neighbors in common with a, giving

$$|\{(b, z) : b \sim z, a \sim b, a \not\sim z\}| = (n - d - 1)\beta.$$

Combining, we find

$$(n - d - 1)\beta = d(d - \alpha - 1). \tag{9.3}$$

As $d < n/2$, this equation tells us

$$d(d - \alpha - 1) \geq d\beta \quad \Longrightarrow \quad d - \alpha - 1 \geq \beta. \tag{9.4}$$

Adding inequality 9.1 to (9.4) gives

$$2d \geq 3\beta \quad \Longrightarrow \beta \leq \frac{2}{3}d,$$

while adding inequality 9.2 to (9.4) gives

$$\alpha + 1 \leq \frac{2}{3}d.$$

Thus, for every $a \neq b$ the number of vertices that are neighbors of a but not of b is at least $\min(d - \alpha - 1, d - \beta) \geq d/3$.

□

9.9 Notes

You should wonder if we can make this faster by analyzing refinement steps. In, [Spi96], I improved the running time bound to $2^{O(n^{1/3} \log n)}$ by analyzing two refinement phases. The algorithm required us to handle certain special families of strongly regular graphs separately: Latin square graphs and Steiner graphs. Algorithms for testing isomorphism of strongly regular graphs were recently improved by Babai, Chen, Sun, Teng, and Wilmes [BCS+13, BW13, SW15]. The running times of all these algorithms are subsumed by that in Babai's breakthrough algorithm for testing graph isomorphism [Bab16].

References

[Bab80] László Babai. On the complexity of canonical labeling of strongly regular graphs. *SIAM Journal on Computing*, 9(1):212–216, 1980.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697. ACM, 2016.

[BCS+13] László Babai, Xi Chen, Xiaorui Sun, Shang-Hua Teng, and John Wilmes. Faster canonical forms for strongly regular graphs. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 157–166. IEEE, 2013.

[BW13] László Babai and John Wilmes. Quasipolynomial-time canonical form for steiner designs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 261–270. ACM, 2013.

[Spi96] Daniel A. Spielman. Faster isomorphism testing of strongly regular graphs. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 576–584, New York, NY, USA, 1996. ACM.

[SW15] Xiaorui Sun and John Wilmes. Faster canonical forms for primitive coherent configurations. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 693–702. ACM, 2015.