GRAPH ISOMORPHISM PROBLEM

SALLY ZHU

1. Introduction

The Graph Isomorphism Problem (GI) is one of the biggest questions that mathematicians and computer scientists are working on in the modern day. The GI is defined as the computational problem of determining whether two finite graphs are isomorphic. Another way to state the problem is whether or not two different graphs, which are networks of nodes and edges, have the same connectivity.

For example, here are 3 graphs that look different but are actually the same. It can be quite difficult to determine if they are the same or not.

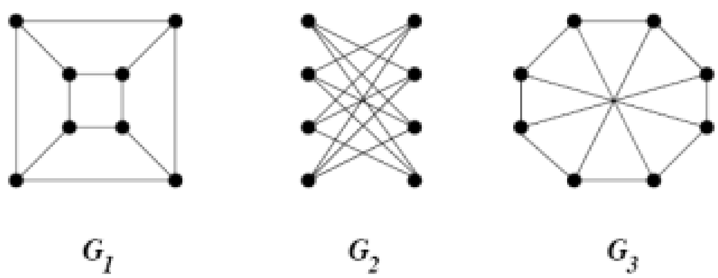

Figure 1. Three graphs

Definition 1.1. An algorithm is said to have a quasipolynomial time complexity if it runs slower than polynomial time but faster than exponential time. In general, it has time complexity of the form $2^{O((logn)^c}$ for some constant c.

Theorem 1.2. *The graph isomorphism problem can be solved with a deterministic algorithm in quasipolynomial time.*

In November of 2015, Laszlo Babai claimed that the Graph Isomorphism Problem could be solved in quasipolynomial time, and he gave a talk about his findings. His talk can be viewed at this link: `http://people.cs.uchicago.edu/~laci/2015-11-10talk.mp4`.

Currently, proving that the GI problem can be solved in polynomial time is still an unsolved problem in computer science.

Another interesting fact about the GI problem is that it is of the class NP, which means that two graphs can be *confirmed* to be isomorphic in polynomial time. While most NP class problems can also be solved in polynomial time, the GI problem can not.

2. String Automorphism Problem

The main result of Babai's proof is a more general problem, the string automorphism problem, which is defined as follows: Given a set of strings X with $x, y \in X$ and a group G acting on X by generating permutations, is there a $\sigma \in G$ such that $x^\sigma = y$? Thus, the actual theorem Babai claims is the following.

Theorem 2.1. *Given a generating set for a group G of permutations of a set X and a string x, there is a quasipolynomial time algorithm which produces a generating set of the subgroup $Aut_G(x)$ of G, i.e. the string automorphisms of x that lie in G.*

Eugene Luks was the first person to incorporate Group theory into the study of graph isomorphism. Group theory helps with this question about graphs, because the GI reduces to this.

Lemma 2.2. *GI is reducible to the problem of computing, given a graph X, a list of generators for the automorphism group of G.*

Proof. Suppose you have two graphs G_1 and G_2. Form the union $G = G_1 \cup G_2$. Then G_1 and G_2 are isomorphic if and only if some automorphism swaps the two groups, so every generating set of $Aut(G)$ must contain one of these automorphisms. ∎

Similarly, the string automorphism problem can be reduced to computing an automorphism group. Computing the automorphism set of a group is reduced to computing the automorphism group of a subgroup, for a particular string. Essentially, one would like to find the subgroup of permutations of $Aut(G)$, and to find its generating set, which would encode the group, which is just the String Automorphism Problem.

Now you just want to compute $Aut(G)$ either by breaking the group into smaller subgroups or by constructing automorphisms of it.

The analogy that Babai used is if you want to know the automorphisms of a graph X, you can partition the vertices by degree, knowing that an automorphism has to preserve the degree of the vertice. Some structures, like an equitable partition, can be used to reduce the steps before brute-force searching.

Definition 2.3. An equitable partition is a partition of vertices by degree in a graph, relative to vertices in other blocks of the partition. (Note: finding equitable partitions of a graph can be done in polynomial time.)

Example: a vertex with degree 3 (the degree is the number of edges on that particular vertex) that has 2 neighboring vertices with degree 2 would be in a different block than a vertex with degree 3 and 1 neighboring vertex with degree 2. (Blocks would have to partitioned separately.)

3. The Johnson Graph

Definition 3.1. A Johnson graph is a special case of undirected graphs, defined from a system of sets. The vertices of a Johnson graph $J(n, k)$ are the k-element subsets of an n-element set.

The Johnson graph is one of the worst-case graphs in the GI problem. This is because it resists being partitioned using an equitable partition. There is a separate algorithm specifically designed for Johnson groups in the Babai result.

Johnson graphs resist being partitioned because it turns out that sometimes it contains part of the automorphism groups that is not yet broken up.

4. Split-or-Johnson

Definition 4.1. A canonical construction of a graph is a labeled graph $Canon(G)$ that is isomorphic to G, such that every graph that is isomorphic to G has the same canonical form as G.

Another simpler definition is that a canonical labeling (or construction) of a graph G is a graph that is isomorphic to the graph and is also to the entire set of graphs that G is isomorphic to.

Here is a canonical ordering of a graph.

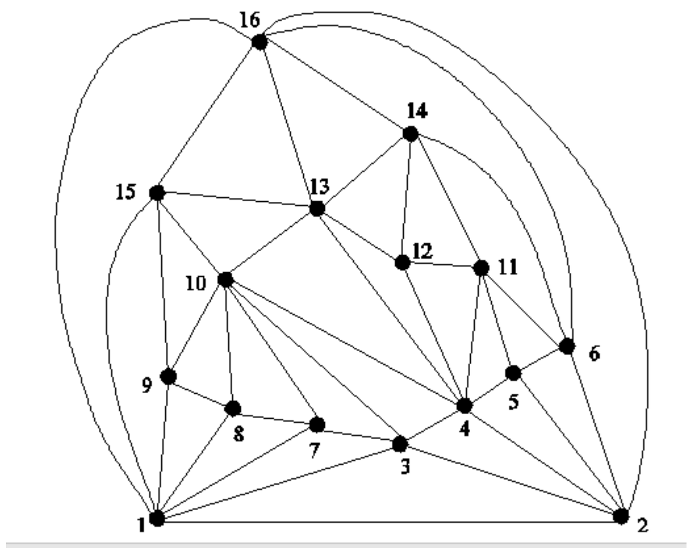

Figure 2. Canonical ordering

Definition 4.2. An algorithm is said to run in polylogarithmic time if the time complexity is of the form $O((logn)^c$ for some constant c.

Babai's algorithm uses a "divide and conquer" method by breaking the problem into many pieces, in this case pieces with 9/10 the size of the original. Recursively, one could make progress on each piece until the problem size is polylogarithmic enough to brute force. Then one could put the pieces back in quasipolynomial time.

One can individualize a vertex by removing it from its equitable partition, then re-defining the partition. For all the vertices in a block, one can individualize them and form smaller blocks. By performing each of these possibilities, one can find a "canonical labeling" of the graph. Two different graphs with different canonical labelings must have started with two graphs that are not isomorphic.

This results in a major theorem of Babai's result.

Theorem 4.3. *If X is a regular graph on m vertices, then by individualizing a number of vertices we can find one of the following three things:*

(1) *A canonical coloring with each color class having at most 90% of all the nodes.*
(2) *A canonical equipartition of some subset of of the vertices that has at least 90% of the nodes.*
(3) *A canonically embedded Johnson graph on at least 90$ of the nodes.*

The above method is known as the "split-or-Johnson" routine. Another method is by finding the $Aut_G(x)$ without breaking it into smaller pieces.

The first two cases can be solved "easily" with a simple recursion. Babai comments that the last case, Johnson groups, are an "unspeakable misery" when it comes to solving the graph isomorphism problem.

Note that the split-or-Johnson routine solves at $log^3(n)$ time, so this would be another point of difficulty in trying to solve the GI problem in polynomial time.

5. Giant Homomorphisms

Definition 5.1. A homomorphism $\varphi : G \to S_m$ is called the *giant* if the image of G is either the alternating group or all of S_m. Let $Stab_G(x)$ denote the stabilizer subgroup of $x \in G$. Then x is called "affected" by φ if $\varphi(Stab_G(x))$ is not giant.

Theorem 5.2. *Let $\varphi : G \to S_m$ be a giant homomorphism and $U \subset G$ the set of unaffected elements. Let $G_{(U)}$ be the point-wise stabilizer of U, and suppose that $m > \max(8, 2+\log_2 n)$. Then the restriction $\varphi : G_{(U)} \to S_m$ is still giant.*

Apparently this is a non-trivial theorem because it depends on the classification of finite simple groups.

The assumption that the $Aut_G(x)$ was the entire symmetrical group of G is an assumption that $\varphi \to S_m$ is giant. By refuting this without having to check all sets and subsets of S, one can pick a test set $A \subset [m]$ where A is polylogarithmic in size and test whether the restriction of φ to the test set is giant in $Sym(A)$. If it is giant, A is known as *full*.

Theorem 5.3. *One can test the property of a polylogarithmic size test set being full in quasipolynomial time in m.*

Babai claimed that in a non-full group, one could construct a subgroup with the image of the automorphism group onto the symmetric group.

If the group if full, then one must find automorphisms to show that the group if giant.
The algorithm to test fullness is called the *Local Certificates Algorithm*. The termination is when W, defined to be the group of elements affected by φ_A stops growing or ceases to be giant. Then, the subgroup generated by these layers of the elements contain the automorphism group.

6. Conclusion

There are many uses in all types of fields for telling whether two graphs are isomorphic. Crytographists use graph isomorphism in zero knowledge proofs with two given graphs. In biometrics, people use it for fingerprint scanners, facial recognition, and retina scanners.

Chemists use it for compound recognition, where atoms are indicated by vertices and covalent bonds are represented by edges. Even more uses are for studying social networks, searching databases, recognizing plagiarism, and more.

7. References

László Babai. *Graph Isomorphism in Quasipolynomial Time.* URL: `http://people.cs.uchicago.edu/~laci/17groups/version2.1.pdf`.

Chris Coyle, Austin Wyer, Elvis Offor. *Graph Isomorphism.* URL: `http://web.eecs.utk.edu/~cphill25/cs594_spring2017/presentations/graph-isomorphism.pdf`.

Jeremy Kun. *A Quasipolynomial Time Algorithm for Graph Isomorphism: The Details.* URL: `https://jeremykun.com/2015/11/12/a-quasipolynomial-time-algorithm-for-graph-isomorp`

Johnson graph. URL: `https://en.wikipedia.org/wiki/Johnson_graph`.

Graph isomorphism problem. URL: `https://en.wikipedia.org/wiki/Graph_isomorphism_problem`.

Erica Klarreich. *Graph Isomorphism Vanquished–Again.* URL: `https://www.quantamagazine.org/graph-isomorphism-vanquished-again-20170114/`.

László Babai. URL: `http://people.cs.uchicago.edu/~laci/2015-11-10talk.mp4`.

Finding Canonical Ordering of a planar graph. URL: `https://math.stackexchange.com/questions/2049380/finding-canonical-ordering-of-a-planar-graph`.