Algebra and Computation Course Instructor: V. Arvind

Lecture 10: General GRAPH-ISO

Lecturer: V. Arvind *Scribe: Ramprasad Saptharishi*

1 Overview

In the last few classes, we solved some special cases of the graph isomorphism problem. Now we shall use those ideas to give an algorithm for the general GRAPH-ISO problem. This, of course, is not a polynomial time algorithm but is interesting nevertheless.

Since any graph on n vertices has degree bounded by n, just a naive simulation of the bounded degree GRAPH-ISO would only give us a n^{n^2} which is worse than the brute-force approach (which is a $O(n!) = (cn)^n$ algorithm). We shall see a $O(n^{n^{2/3}})$ algorithm for GRAPH-ISO

2 Colourings of Graphs

A *colouring* of a graph is just associating a colour to every vertex of the graph. Formally, it is a map $f : V \longrightarrow \{1, 2, \cdots, |V|\}$. And we can further assume that the range of f is an initial segment of $\{1, 2, \cdots, |V|\}$.

A colouring f_1 is said to be a *refinement* of f, denoted by $f \leq f_1$, if $f_1(x) \leq f_1(y) \implies f(x) \leq f(y)$ for all vertices x, y. A refinement f' of f is said to be *proper* if $f' \neq f$.

2.1 Colour-Degree Refinement

For this lecture, this is the refinement that we would be dealing with. Let (X, f) be a coloured graph. The new refinement is defined as follows:

Define $g(x) = \big(f(x), k_1(x), k_2(x), \cdots, k_{|V|}(x)\big)$ where $k_i(x)$ is the number of neighbours of x that are coloured i. Since we need to map these tuples to $\{1, 2, \cdots, |V|\}$, lexicographically sort the tuples and map them to $\{1, 2, \cdots, |V|\}$. Let us call this induced colouring as f'.

The choice of the tuple, whose first coordinate is $f(x)$, it is clear that f' is a refinement of f.

One can continuously keep refining a colouring, and it spreads out vertices with more refinements. Hence, we can properly refine at most n times. A colouring that cannot be properly refined by the colour-degree refinement is said to be a *stable colouring.*

2.2 Propagating Refinements

The central idea is to keep propagate refinements as much as possible. The problem is that the colour-degree refinements may not be able to spread the vertices enough, it might get stuck in a stable colouring much earlier.

But how does refinements help? Where are we heading?

Proposition 1. *Let (X_1, f_1) and (X_2, f_2) are two coloured graphs with f_1 and f_2 their corresponding refinements. Then*

$$(X_1, f_1) \equiv (X_2, f_2) \iff (X_1, f_1') \equiv (X_2, f_2')$$

Proof. Of course! □

The idea is to get to a stage where we can effeciently solve the problem. But, as remarked earlier, what do we do when we get stuck up at a stable colouring? We *individualize* a vertex.

- Keep refining until stable refinement
- Pick some vertex, and give it a colour that has not been assigned to any vertex.
- Repeat

The trick is to find a good a good vertex to individualize in order to propagate the refinements. We shall see that there is a small sequence of vertices, which on the 'refine until stable, individualize, repeat' gets us where we want.

3 Colour Valence and GRAPH-ISO

Definition 2. *A coloured graph (X, f) is said to have colour valence d if for all vertices x and colours i, either x is adjacent to at most d neighbours of colour i (valence of x is bounded by d) or it is not adjacent to atmost d vertices of colour i (covalence of x is bounded by d).*

The following theorem shows that there exists a small sequence of vertices that can get us to a constant colour valence.

Theorem 3. *If (X, f) has colour valence d, then there exists a sequence of nodes $\{x_1, x_2, \cdots, x_k\}$ with $k \leq 2n/d$ such that (X, f'), (colouring obtained by stabilizing and individualizing these vertices) has colour valence $d/2$.*

Proof. The proof is a greedy algorithm to pick up the vertices. Let $S_i = \{x_1, \cdots, x_{i-1}\}$. If (X, S_i) has colour valence $\leq d/2$, then we are already done, hence stop.

Else, there exists an $x \in V(X)$ and a colour m such that both valence and covalence of x in $f^{-1}(m)$ is $> d/2$. Let $N(x)$ be the neighborhood or co-neighbourhood of x (whichever violates the valence bound). Hence we have $d/2 < |N(x)| \leq d$. Pick x as x_i and continue.

Claim: The sets $\{N(x_i)\}$ are pairwise disjoint.

Once we prove the claim,

$$\frac{kd}{2} \leq \sum_{i=1}^{k} |N(x_i)| \leq n$$

which then forces $k \leq dn/2$.

Proof of claim: Suppose $N(x_j) \cap N(x_i) \neq \phi$ for some $j < i$. If m is the colour class that x_i violates, then $N(x_j) \cap f_{S_i}^{-1}(m) \neq 0$.

Now, i is a refinement further away from j, and hence would have further spread the colours of the vertices. Therefore, $N(x_j)$ has to be a union of colour classes in S_i.

Therefore, if $N(x_j) \cap f_{S_i}^{-1}(m) \neq \phi \implies f_{S_i}^{-1}(m) \subseteq N(x_j)$.

Now, since the entire colour class is contained inside $N(x_j)$, both the valence and covalence is contained in $N(x_j)$. And since each of them is atleast $d/2$, forces $|N(x_j)| > d$ contradicting the colour valence of X being bounded by d.

Hence, $\{N(x_j)\}$ are pairwise disjoint, thus proving the claim and the theorem. □

We can now start with a trivial colouring (every vertex given the same colour, and with $k \leq 8n/d$ get to (X, f') with colour valence bounded by d.

3.1 Enter Luks

Let $T(n)$ be the worst case time bound for GRAPH-ISO and $T(n, d)$ be the worst case time bound for d-colourvalence-GRAPH-ISO.

The natural algorithm is the following:

1. Put trivial colouring on both graphs
2. Refine until stable.
3. Pick the set of vertices to individualize on one graph. Try all possibilities on the other.
4. If at any try, if x was the vertex picked with valence greater than $d/2$, but valence $\leq d/2$ on the other graph, stop that try.
5. Once you get to bounded colour valence, do the corresponding algorithm.

Hence, clearly $T(n) \leq n^{8n/d} T(n, d)$.

The claim is that, Luks' algorithm for bounded degree graphs work here!

Let (X, f) be the graph with a stable colouring and colour valence d. Let $C_1, \cdots, C_r$ be the colour classes. Define $X(C_i)$ be the induced subgraph on this colour class (only edges within that class) and $X(C_i, C_j)$ as the induced bipartite graph (only edges from one class to another).

The choice of the stable colouring then forces $X(C_i)$ to be a regular graph (since the colour-degree-refinement would give a proper refinement) and also $X(C_i, C_j)$ is a semi-regular[1] graph.

Now in $X(C_i)$, either the degree is bounded by d or codegree. We can assume that it is the degree, by complementing otherwise (and checking if the other graph also has the same property, they necessarily have to if they are isomorphic). And similarly for $X(C_i, C_j)$, complement if $|E(X(C_i, C_j))| > |C_i||C_j|/2$.

(X, f) now has the property that for all vertices x and colours i, degree of x in C_i is bounded above by d. Hence we now have two graphs (X, f) and (Y, g) with the above property. As in the Luks algorithm, add a distinguished edge, and additionally a vertex on the edge with a new colour. Now layer the graphs based on edges obtained by paths from the new vertex.

Here, the kernel is again $\bigotimes \mathrm{Sym}(\cdot)$ since the colour classes can't move! And each element of the product is bounded since the degree is bounded! Hence the automorphism group is infact in $\mathcal{B}_d$ and hence can be solved by a $O(n^{d^2})$ algorithm.

Now $T(n) \leq n^{8n/d+d^2}$ and an optimal choice for $d$would give us a $O(n^{n^{2/3}})$ algorithm for GRAPH-ISO.

[1] degree of all vertices on left are the same, the same on the right