Automorphism Group of Graphs

Ritam M Mitra

April 2024

1 Introduction

1.1 Graphs

A graph G is a pair of sets (V, E), where V is a finite non-empty set of elements called vertices, and E is a set of unordered pairs of distinct vertices called edges. The sets V and E are the vertex-set and the edge-set of G, and are often denoted by $V(G)$ and $E(G)$, respectively. An example of a graph is shown in Fig. 1.

The number of vertices in a graph is the order of the graph; usually it is denoted by n and the number of edges by m. Standard notation for the vertex-set is $V = \{v_1, v_2, ..., v_n\}$ and for the edge-set is $E = \{e_1, e_2, ..., e_m\}$. Arbitrary vertices are frequently represented by $u, v, w, ...$ and edges by $e, f, ...$.

For convenience, the edge $\{v, w\}$is commonly written as vw. We say that this edge joins v and w and that it is incident with v and w. In this case, v and w are adjacent vertices, or neighbours. The set of neighbours of a vertex v is its neighbourhood $N(v)$. Two edges are adjacent edges if they have a vertex in common. The number of neighbours of a vertex v is called its degree, denoted by deg v. Observe that the sum of the degrees in a graph is twice the number of edges. If all the degrees of G are equal, then G is regular, or is $k - regular$ if that common degree is k. The maximum degree in a graph is often denoted by Δ.

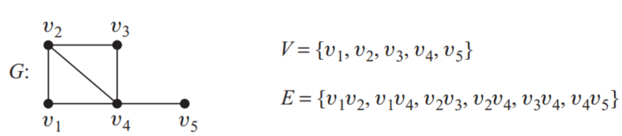

Fig. 1.

Figure 1:

1.2 Groups

A *group* is a set G with a binary operation $\circ$ satisfying the conditions:

- for all $g, h, k \in G$, $(g \circ h) \circ k = g \circ (h \circ k)$ (*associative law*);
- there exists an element $1 \in G$ (the *identity*) such that $1 \circ g = g \circ 1 = g$ for all $g \in G$;
- for each $g \in G$, there is an element $g^{-1} \in G$ (the *inverse* of g) such that $g \circ g^{-1} = g^{-1} \circ g = 1$;
- for all $g, h \in G$, $g \circ h = h \circ g$ (*commutative law*) then the group G is *Abelian* (or *commutative*);

Groups are important here because the set of automorphisms of a graph (with the operation of composition of mappings) is a group. In many cases, the group encodes important information about the graph; and in general, the use of symmetry can be used to do combinatorial searches in the graph more efficiently.

1.2.1 Permutation Group

A *permutation* of the set Ω is a bijective mapping $g : \Omega \to \Omega$. We write the image of the point $v \in \Omega$ under the permutation g as vg, rather than $g(v)$. The composition g_1g_2 of two permutations g_1 and g_2 is the permutation obtained by applying g_1 and then g_2 that is,

$$v(g_1g_2) = (vg_1)g_2 \textit{ for each } v \in \Omega.$$

A *permutation* group on Ω is a set G of permutations of Ω satisfying the following conditions:

- G is closed under composition: if $g_1, g_2 \in G$ then $g_1g_2 \in G$;
- G contains the *identity* permutation 1, defined by $v1 = v$ for $v \in \Omega$;
- G is closed under inversion, where the inverse of g is the permutation g^{-1} defined by the rule that $vg^{-1} = w$ if $wg = v$;

The *degree* of the permutation group G is the cardinality of the set Ω. The simplest example of a permutation group is the set of all permutations of a set Ω. This is the *symmetric* group, denoted by $\mathrm{Sym}(\Omega)$. More generally, an action of G on Ω is a homomorphism from G to $\mathrm{Sym}(\Omega)$. The image of the homomorphism is then a permutation group. The action is faithful if its kernel is $\{1\}$ – that is, if distinct group elements map to distinct permutations. If the action is $faithful$, then G is isomorphic to a permutation group on Ω.

2 Automorphism groups of graphs

Let $G = (V, E)$ be a simple graph, possibly directed and possibly containing loops. An automorphism of G is a permutation g of V with the property that $\{vg, wg\}$ is an edge if and only if $\{v, w\}$ is an edge – or, if G is a digraph, that (vg, wg) is an arc if and only if (v, w) is an arc. Now the set of all automorphisms of G is a permutation group $\mathrm{Aut}(G)$, called the automorphism group of G.

The definition of an automorphism of a multigraph is a little more complicated. The most straightforward approach is to interpret a multigraph as a weighted graph. If $a_{v,w}$ denotes the multiplicity of vw as an edge of G, then an automorphism is a permutation of V satisfying $a_{vg,wg} = a_{v,w}$. Again, the set of automorphisms is a group.

Theorems

- A simple undirected graph and its complement have the same automorphism group.
- The automorphism group of the complete graph K_n or the null graph N_n is the symmetric group S_n.
- The 5-cycle C_5 has ten automorphisms, realized geometrically as the rotations and reflections of a regular pentagon.

This last group is the *dihedral group* D_{10}. More generally, $\mathrm{Aut}(C_n)$ is the dihedral group D_{2n}, for $n \geq 3$.

2.1 Algorithmic aspects

Two algorithmic questions that arise from the above definitions are graph isomorphism and finding the automorphism group. The first is a decision problem.

Graph isomorphism
Instance: Graphs G and H
Question: Is $G \cong H$?

The second problem requires output. Note that a subgroup of S_n may be superexponentially large in terms of n, but that any subgroup has a generating set of size $O(n)$, which specifies it in polynomial space.

Automorphism group
Instance: A graph G
Output: generating permutations for $\mathrm{Aut}(G)$.

These two problems are closely related: indeed, the first has a polynomial reduction to the second. For, suppose that we are given two graphs G and H. By taking complements if necessary, we may assume that both G and H are connected.

Now suppose that we can find generating permutations for $\mathrm{Aut}(K)$, where K is the disjoint union of G and H. Then G and H are isomorphic if and only if some generator interchanges the two connected components.

Conversely, if we can solve the graph isomorphism problem, we can at least check whether a graph has a non-trivial automorphism, by attaching distinctive 'gadgets' at each vertex and checking whether any pair of the resulting graphs are isomorphic.

3 Graph Isomorphism and Automorphism Groups

Recall that two graphs G_1 and G_2 are isomorphic if there is a re-numbering of vertices of one graph to get the other, or in other words, there is an automorphism of one graph that sends it to the other. And clearly, $\text{Aut}(G) \leq S_n$, the symmetric group on n objects, which represent the permutation group on the vertices. And since it is a subgroup of the permutation group, $|Aut(G)| \leq n!$

Of course, providing the entire automorphism group as output would take exponential time but what about a small generating set? Which then leads us to, does there exist a small generating set?

Theorem. With **Graph-Iso** as an oracle, there is a polynomial time algorithm for **Graph-Aut** and vice-versa.

First we shall show that we can solve **Graph-Iso** with **Graph-Aut** as an oracle. We are given two graphs G_1 and G_2 and we need to create a graph G using the two such that the generating set of the automorphism group should tell us if they are isomorphic or not.

Let $G = G_1 \cup G_2$. Suppose additionally we knew that G_1 and G_2 are connected, then a single oracle query would be sufficient: if any of the generators of $\text{Aut}(G)$ interchanged a vertex in G_1 with one in G_2, then connnectivity should force $G_1 \cong G_2$.

But what if they are not connected? We then have this very neat trick: $G_1 \cong G_2 \iff \overline{G_1} \cong \overline{G_2}$. As either G_1 or $\overline{G_1}$ has to be connected, one can check for connectivity and then ask the appropriate query.

The other direction is a bit more involved. The idea is to see that any group is a union of cosets. Suppose

$$H = a_1K \cup a_2K \cup \ldots a_nK.$$

then $\{a_1, a_2, \ldots, a_n\}$ along with a generating set for K form a generating set for H. Hence once we have a subgroup K with small index, we can then recurse on K.

Hence we are looking for a tower of subgroups.

$$Aut(G) = H \geq H_1 \geq H_2 \geq \ldots \geq H_m = \{e\}$$

such that $[H_i : H_{i+1}]$ is polynomially bounded.

For our graph G, let $Aut(G) = H \leq S_n$. We shall use Weilandt's notation where i^π denotes the image of i under π. In this notation, composition becomes simpler: $(i^\pi)^\tau = i^{\pi.\tau}$.

Define $H_i = \{\pi \in H : 1^\pi = 1, 2^\pi = 2, \ldots i^\pi = i\}$. And this gives the tower

$$H_0 = H \geq H_1 \geq H_2 \geq \ldots \geq H_{n-1} = \{e\}$$

with the additional property that $[H_i : H_{i+1}] \leq n - i$ since there are at most $n - i$ places that $i + 1$ can go to when the first i are fixed. We need to find to find the coset representatives.

As H is $\mathrm{Aut}(G)$, we can find the coset representatives using queries to the **Graph-Iso** subroutine: to find a representative for $[H^{(i)} : H^{(i+1)}]$, make two copies of G, force the first i vertices to be fixed (by putting identical gadgets on them in each copy), and for each place j' that $i + 1$ might go to, force $i + 1$ to go to j', test if a graph isomorphism exists, and continue till an isomorphism is found.

4 References

[1] Algebraic Graph Theory Book by Chris Godsil and Gordon Royle

[2] Topics in Algebraic Graph Theory Lowell W. Beineke, Robin J. Wilson Cambridge University Press