Reframing the Graph Isomorphism Problem using Permutation Groups

By Michael Almandeel

This paper is not meant to be a rigorous proof, and as such, it is assumed that the reader is familiar with modern algebra , linear algebra, boolean algebra and basic computer science theory. With that out of the way, lets get started.

Graph Isomorphism(GI) is an open problem in computer science. It is currently unknown whether or not this problem is np-complete. In this paper I am going to demonstrate a method of mapping a certain class of isomorphism problems(in which GI is included) into a homogeneous system of equations. Further, I will explain how in the special case of GI , this homogeneous system is mappable(or reducible using cs jargon) to a boolean satisfiability problem which is np-complete.

The specific kinds of isomorphisms we will investigate are those in which the transformation matrices that satisfy them are members of the permutation group(PG(I,N)) of the identity matrix for some n. To be clear, the transformations these matrices perform is that of a permutation function, which effectively permutes either the rows(TxM) or the columns(MxT) of a matrix M.

PG(I,N) of the N by N identity matrix has several properties, but we are only interested in a few:

1- For all Y in PG(I,N) , Y x Y(inverse) = I
2- Y(inverse) = Y(transpose)
3- For any matrix M, and any Y in PG(I,N) , the entries of Y that contain a value of 1 at row i and column j permutes row j in M to row I in M as a result of the expression Y x M.
4- For any matrix M, and any Y in PG(I,N) , the entries of Y that contain a value of 1 at row i and column j permutes column i in M to column j in M as a result of the expression
M x Y.

GI is typically posed as an algebra problem by mapping the two graphs we want an isomorphism for into adjacency matrices M1 and M2. Then the following expression is set up:

T x M1 x T = M2 , where T is a transformation matrix. What's interesting is that the set of all possible solutions of T is equal the set of all matrices in PG(I,N) , because any valid solution to GI consists of a permutation of M1 such that M1 = M2.

Consider the properties of PG(I,N) mentioned earlier. In the expression above, we are first permuting the rows of M1 using T, and then permuting the columns of M1 using T, which is

equivalent to M2. It stands to reason then, that the matrix given by permuting the rows of M1 with T is equivalent to the matrix given by permuting the columns of M2 using T(inverse). The above statement is also true for the columns of M1 and the rows of M2. Therefore, we find the following equivalency:

(T x M1 x T = M2) = (T x M1 = M2 x T(transpose)) = (M1 x T = T(transpose) x M2)

The expression (T x M1 = M2 x T(transpose)) is interesting because when you multiply out T x M1 and M2 x T(transpose) what you get is a system of homogeneous equations where the variables are the entries of T , and each homogeneous equation represents the equivalency of the corresponding entries of both resultant matrices.

While the above holds true for all transformation matrices in PG(I,N) , the fact that GI uses adjacency matrices allows us to map this into a boolean satisfiability problem. Let's examine why. The entries of an adjacency matrix are elements of the Galois field with two elements, a boolean algebra. There are several properties of boolean algebra, but we are interested in the following one:

(A = B) = (or(and(A,B) , and(not A , not B))) . Consider applying the homogeneous equations yielded by T x M1 = M2 x T(transpose) to the above theorem where A and B are the corresponding entries of T x M1 and M2 x T(transpose).

The solution set of this expression however, can contain values for the entries of T such that T is not an element of PG(I,N). To restrict the solution set to correspond solely to matrices in PG(I,N), we form a conjunction of the preceding expression with a conjunction of two expressions which restricts all elements in the solution set to only have a single value of 1 in any given row or column of T.

It's important to note that the resulting boolean SAT expression is np-complete by Schafer's dichotomy theorem, as it fails all of the specified tests.