An algorithm for solving the graph isomorphism problem

By Lucas Allen

Contents

Introduction

In this article I'll present an agorithm for solving the graph isomorphism problem [link to wikipedia page]. I'm not sure if this algorithm is polynomial time for all cases, but I'm sure it will be in some cases.

The problem

The graph isomorphism problem is the problem of determining whether or not two graphs are really the same graph. If they are the same graph then they are called isomorphic. Typically the edges of the graph don't have associated weights (eg, costs, distance, etc.)

Given two graphs, each with n vertices/nodes, we can represent these graphs as two n*n matrices A and B. Where A[i,j] = 1 if there is an edge from the ith vertex to the jth vertex in the first graph and is 0 otherwise, Likewise for B with the second graph.

To solve the graph isomorphism problem, we want to find a permutation of the rows and columns of A which turns it into B. This correpsonds to a permutation of vertices of A. Basically we just want to relabel the numbering of the nodes in one of the graphs to see if we can get the other graph. If there exists a permutation, then the two graphs are isomorphic, otherwise they are not.

So, we want to find a permutation matrix [link to wikipedia page] X which is a solution to the matrix equation:

$XAX^T = B$

Since for a permutation matrix $X^{-1} = X^T$, this equation becomes:

$XA = BX$
$XA – BX = 0$

We can solve this equation by creating an n^2*n^2 matric C and finding it's null space basis. [link to wikipedia page]

The algorithm

If we look at the matrix equation:

XA – BX = 0

And note that the 0 on the right hand side is really an n*n matrix of all zeros, then we can break up X into row (y[i]) and column vectors (z[i]), such that:

$X = [y[1], y[2], \dots , y[n]]^T$
$X = [z[1], z[2], \dots , z[n]]$

And also break A up into column vectors, and B into row vectors:

$A = [a[1], a[2], \dots , a[n]]$
$B = [b[1], b[2], \dots , b[n]]^T$

Then the above equation is a set of n^2 linear equations, because we can write it as:

y[1] · a[1] - b[1] · z[1] = 0
y[1] · a[2] - b[1] · z[2] = 0
…
y[1] · a[n] - b[1] · z[n] = 0
y[2] · a[1] - b[2] · z[1] = 0
…
y[n] · a[n] - b[n] · z[n] = 0

This works because y[i] · a[j] is ith-jth element of XA and b[i] · z[j] is the ith-jth element of BX. Since we're subtracting matrices, we get the ith-jth element of the zero matrix, which is zero.

The y and z vectors have some overlapping elements, so what we can do is create another vector (x) with all the rows of X lined up side by side:

$x = [[y^T[1], y^T[2], \dots , y^T[n]]^T$

This vector has n^2 elements. WE can now create an n^2*n^2 matrix C, which has a paricular structure:

C[1,1] C[1,2] … C[1,n]
C[2,1] C[2,2] … C[2,n]
… …
C[n,1] C[n,2] … C[n,n]

Where all the C[i,j]'s are n*n matrices. If B[i,i] = 0 then C[i,i] = A, if B[i,i] = 1 then C[i,i] = A – I, where I is the n*n identity matrix, and if B[i,j] = 1, when i does not equal j then C[j,i] = -I, if B[i,j] = 0 then C[j,i] is the n*n zero matrix. This leaves us with the set of n^2 linear equations:

Cx = 0

All we need to do now is find the null space [link to wikipedia page] of C. If the dimension of the null space is less than or equal to n, then all we need to do is set y[1] to each binary string of length n with a single 1 in it, plug that into the x vector and solve the null space basis equations to get the resulting matrix. If the resulting matrix is a permutation matrix then the graphs are isomorphic.

However, if the dimension of the null space is less than or equal to 2n, but greater than n, then we need to set both y[1] and y[2] and put them into the x vector. This greatly increases the run time of the algorithm.

Complexity

If the dimension of the null space (dim N(C)) is between kn and mn such that:

$kn < \dim N(C) < mn$

for k and m less than n, then this algorithm will have a run time of O(n!/(n – m)!), which puts it in NP. This is because we need to solve a set of linear equations with dim N(C) unknowns. The right hand side of the linear equations is part of a permutation matrix, so if we only need to know one row of X then we only need to solve n sets of linear equations, whereas if we need to know two rows of X then we need to solve n * (n-1) linear equations, and so on.

However, C has a very particular structure. After thinking about it, I think dim N(C) will never be greater than 2n, but I haven't proved it so I could be wrong.

Examples

Here are a couple of examples with n = 3. The calculations were done using the null space calculator [add link] at gregthatcher.com.

Example 1

For the first example the graphs have all undirected edges, they look like:

A: (1)---(2)---(3) B: (1)---(3)---(2)

Where the numbers in the brackets are vertices and the dashes are edges. The matrices are:

A:

0	1	0
1	0	1
0	1	0

B:

0	0	1
0	0	1
1	1	0

The C matrix is:

0	1	0	0	0	0	-1	0	0
1	0	1	0	0	0	0	-1	0

0	1	0	0	0	0	0	0	-1
0	0	0	0	1	0	-1	0	0
0	0	0	1	0	1	0	-1	0
0	0	0	0	1	0	0	0	-1
-1	0	0	-1	0	0	0	1	0
0	-1	0	0	-1	0	1	0	1
0	0	-1	0	0	-1	0	1	0

The reduced row echalon form of C is:

1	0	0	0	0	-1	0	0	0
0	1	0	0	0	0	0	0	-1
0	0	1	0	0	1	0	-1	0
0	0	0	1	0	1	0	-1	0
0	0	0	0	1	0	0	0	-1
0	0	0	0	0	0	1	0	-1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

And the null space basis of C is:

a[1 0 -1 -1 0 1 0 0 0] + b[0 0 1 1 0 0 0 1 0] + c[0 1 0 0 1 0 1 0 1]

Where a, b and c are real numbers. Because there are only 3 vectors in the null space basis we only need to solve for the top row of X. The first set of equations are:

a = 1
c = 0
-a + b = 0

The solution is clearly a = 1, b = 1, c =0, the corresponding X matrix is:

1	0	0
0	0	1
0	1	0

This is a permutation matrix, so A and B are isomorphic graphs. The second set of equations are:

a = 0
c = 1
-a + b = 0

The solution is a = 0, b = 0, c = 1, The corresponding X matrix is:

0	1	0
0	1	0

1	0	1

This is not a permutation matrix, and is therefore not a solution. The third set of equations are:

a = 0
c = 0
-a + b = 1

The solution is a = 0, b = 1, c = 0, The corresponding X matrix is:

0	0	1
1	0	0
0	1	0

This is a permutation matrix. The graphs A and B are isomorphic with two sets of permutations linking the graphs.

Example 2

For the first example the graphs have all undirected edges, the matrices are:

A:

0	1	0
1	0	1
0	1	0

B:

0	1	1
1	0	1
1	1	0

The A graph is the same as in the first example, the B graph now has an edge going from the first vertex to the second. The C matrix is:

0	1	0	-1	0	0	-1	0	0
1	0	1	0	-1	0	0	-1	0
0	1	0	0	0	-1	0	0	-1
-1	0	0	0	1	0	-1	0	0
0	-1	0	1	0	1	0	-1	0
0	0	-1	0	1	0	0	0	-1
-1	0	0	-1	0	0	0	1	0
0	-1	0	0	-1	0	1	0	1
0	0	-1	0	0	-1	0	1	0

The reduced row echalon form of C is:

1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0

0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

Because the rows are independent, the null space contains only the zero vector. This means there is no solution to the equation and the graphs are not isomorphic.

Example 3

For the third example the graphs have all directed edges, they look like:

A: (1)--->(2)--->(3) B: (3)--->(1)--->(2)

Where the numbers in the brackets are vertices and the dashes with arrows are directed edges. The matrices are:

A:

0	1	0
0	0	1
0	0	0

B:

0	1	0
0	0	0
1	0	0

The C matrix is:

0	1	0	0	0	0	-1	0	0
0	0	1	0	0	0	0	-1	0
0	0	0	0	0	0	0	0	-1
-1	0	0	0	1	0	0	0	0
0	-1	0	0	0	1	0	0	0
0	0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0

The reduced row echalon form of C is:

1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	-1	0	0
0	0	1	0	0	0	0	-1	0

0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	-1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

And the null space basis of C is:

a[0 1 0 0 0 1 1 0 0] + b[0 0 1 0 0 0 0 1 0] + c[0 0 0 0 0 0 0 0 1]

Where a, b and c are real numbers. Because there are only 3 vectors in the null space basis we only need to solve for the top row of X. The first set of equations are:

0 = 1
a = 0
b = 0

This set of equations is internally inconsistent and doesn't have a solution. The second set of equations are:

0 = 0
a = 1
b = 0

The solution is a = 1, b = 0, c = 0, The corresponding X matrix is:

0	1	0
0	0	1
1	0	0

This is a permutation matrix, so A and B are isomorphic. The third set of equations are:

0 = 0
a = 0
b = 1

The solution is a = 0, b = 1, c = 0, The corresponding X matrix is:

0	0	1
0	0	0
0	1	0

This is not a permutation matrix. The graphs A and B are isomorphic with one set of permutations linking the graphs.

Example 4

For the forth example the graphs have all directed edges, the matrices are:

A:

0	1	1
0	0	1
0	0	0

B:

0	1	0
0	0	1
1	0	0

The C matrix is:

0	1	1	0	0	0	-1	0	0
0	0	1	0	0	0	0	-1	0
0	0	0	0	0	0	0	0	-1
-1	0	0	0	1	1	0	0	0
0	-1	0	0	0	1	0	0	0
0	0	-1	0	0	0	0	0	0
0	0	0	-1	0	0	0	1	1
0	0	0	0	-1	0	0	0	1
0	0	0	0	0	-1	0	0	0

The reduced row echalon form of C is:

1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

Because the rows are independent, the null space contains only the zero vector. This means there is no solution to the equation and the graphs are not isomorphic.

Conclusion

This algorithm needs to be tested on large graphs to make sure it works. I know I should do it, but I want to avoid coding if I can and move on to other projects.

The obvious question, for me anyway, is whether this method might work for the sub-graph isomorphism problem, [link to Wikipedia] where A is an n*n matrix and B is an m*m matrix. I've tried to get it to work without any success, but I think if we fill up B with unknown variables so that it's an n*n matrix, then maybe the we can throw away the unknown variables once we have X. I don't know though and I think it's a long shot.