Approaches To Solving The Graph Isomorphism Problem

Jordy Eikenberry
College of Computing
Georgia Institute of Technology
Atlanta, GA 30318

September 6, 2004

Abstract

In this paper I propose a polynomial time algorithm for the Graph Isomorphism problem, which always returns a correct answer in the case that the input graphs are non-cospectral or isomorphic. Although I have no correctness proof in the general case, I suspect that most if not all inputs return a correct answer after the execution of my algorithm. This paper assumes no previous knowledge of the problem itself, however a basic understanding of Theoretical Computer Science is assumed. Explained in this paper are the techniques and approaches to breaking down this problem into something that is more manageable. Finally, I discuss problems relating to Graph Isomorphism that remain open at the time of writing this paper.

Contents

1 Introduction

The Graph Isomorphism problem (GI) is an intensively studied problem that has many important applications (i.e. determining whether two chemical compounds are the same). Despite efforts made by researchers in classifying GI, there still remains a number of open problems within the realm of GI. Of these open problems, the big question is where does GI lie? Is GI NP-complete, or is there a polynomial time algorithm? See Definition 1 for a more formal definition of GI. Analogous to GI is the Graph Automorphism (GA) problem (see Definition 2). This problem is also not known to be in P or NP-complete, although it is believed to be easier than GI. For instance, GA is known to be reducible to GI. Currently, however, there is no known reduction from GI to GA.

Definition 1 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs with respective adjacency matrices A_1 and A_2. We say that $G_1 \cong G_2$ (G_1 is **isomorphic** to G_2) if and only if one of the three equivalent conditions hold:*

1. *There exists a permutation $\pi : V_1 \to V_2$ such that for all pairs u, v of vertices $(u, v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$.*

2. *There exists a permutation π that defines a function f on $n \times n$ matrices, such that for $1 \leq i, j \leq n$,*

$$\begin{aligned} A_1 &= [x_{i,j}], \\ A_2 &= [y_{i,j}], \text{ and} \\ f(A_2) &= [y_{\pi(i),\pi(j)}] = [x_{i,j}] = A_1 \end{aligned}$$

3. *There exists a permutation matrix P such that $A_1 = PA_2P^{-1}$.*

We define the Graph Isomorphism problem as follows

Problem:	**GI**
Instance:	Two graphs G_1, G_2.
Question:	Is G_1 isomorphic to G_2?

Definition 2 *Let $G = (V, E)$ be a graph and A be the adjacency matrix for G. We say that G has a non-trivial automorphism if and only if one of the three equivalent conditions hold:*

1. *There exists a permutation, other than the identity permutation, $\pi : V \to V$ such that for all pairs u, v of vertices $(u, v) \in E$ if and only if $(\pi(u), \pi(v)) \in E$.*

2. *There exists a permutation π, other than the identity permutation, that defines a function f on $n \times n$ matrices, such that for $1 \leq i, j \leq n$,*

$$A = [x_{i,j}] \text{ and}$$
$$f(A) = [x_{\pi(i),\pi(j)}] = [x_{i,j}] = A$$

3. *There exists a permutation matrix P, where P is not the identity matrix, such that $A = PAP^{-1}$.*

We define the Graph Automorphism problem as follows

Problem:	**GA**
Instance:	Graph G.
Question:	Does G contain a non-trivial automorphism?

Although we don't know whether $GI \in P$ for the general case, there are polynomial time algorithms for more specific graphs (such as planar graphs, trees, and graphs of bounded degree, genus, and eigenvalue–see Figure 1). Also, there is evidence to support that GI is not NP-complete. If it were, the polynomial hierarchy would collapse to the second level (see for instance [8]). It could be the case that GI is neither in P nor NP-complete. It has been shown that unless $P = NP$, such problems do exist [9, 10]. GI is also not known to be hard for P. In fact, the best known hardness results are still relatively weak [11].

For this paper, we will assume that all graphs are undirected. We may safely assume this since the GI problem for directed graphs is complete for GI [8]. So, we will only be interested in 0-1 matrices that are symmetric. Furthermore, we shall consider the following definitions (needed for my proposed GI algorithm)

Definition 3 *Let A be the adjacency matrix of a graph. We denote $diag[A]$ to be the vector representation of all elements in the diagonal of A. Similarly, denote $upper[A]$ to be the vector representation of all elements in the upper triangular part of A (not including the diagonal elements).*

In Section 2 we shall discuss some of the known properties of GI that make GI both a unique and interesting problem. Section 3 will mostly be concerned with the following question: what happens if we relax the constraints of GI with respect to the permutation associated with the isomorphism? In Section 4 we will discuss my attempt at showing that $GI \in P$. We will see, however, that this algorithm does not hold for a very rare and specific type of graph, which we will later discuss in Section 5. Section 6 will revisit my initial algorithm, modifying it ever so slightly in the attempt to fix my algorithm for these specific cases. Finally, in Section 7 we will discuss any conclusions that can be made, as well as any problems that remain open.

2 Known Results

First we shall define the class hierarchy structure in terms of the Graph Isomorphism problem, and show where different related problems lie with respect to these classes (See Figure 1). Given the problem GI, we can define the complexity class GI as the set of languages logspace reducible to GI (defined similarly for GA). Now, we can see that GI (GA resp.) also has complete problems, which can defined as the problems equivalent to GI (GA resp.). We notice some interesting properties with this structure, and we will also see in the next section that if we relax the constraints of GI in a particular way, the problem becomes NP-complete (defined as $MaxGI$).

It has been shown that the counting version of GI is just as hard as the decision version of GI [8]. However, the counting version of NP problems appear to be harder than their decision counterpart. In this respect, GI seems to be very different. In fact, there are even problems in P which are conjectured to be intractable. For example, $\#BPM$ is equivalent to solving the permanent problem, which is known to be NP-hard. So, this gives us more evidence towards the conjecture that GI is not NP-complete.

It has also been shown that determining whether two graphs have a unique isomorphism is actually equivalent to GA [8]. Since it is believed that GA is easier than GI, it is believed that $UniqueGI$ is easier than GI. This, again, shows another difference between GI and NP-complete problems. For instance, *co-SAT* (complement of SAT) can be many-one reduced to $UniqueSAT$.

This problem appears to have a very intricate structure, which is what inevitably makes this problem hard. It is known that GI and GA have self-computable functions [8]. So, if one can show that GI (GA resp.) is in BPP, then GI (GA resp.) is in RP, since any self-computable problem in the intersection of BPP and NP is in RP.

We can also think of GI (GA resp.) in terms of it's complement, GNI (GNA resp.). What is the exact complexity of GNI? This is currently not known; in fact, the best known result is relatively weak ($GNI \in AM$ [8]). However, I believe that GI can be done in polynomial time. So, we should be able to show that $GNI \in NP$. But this is also difficult to show, since it is hard to think of a polynomial size proof for Graph Non-Isomorphism. In fact, the best known proof size is exponential. Of course, the next best thing would be to come up with an MA protocol for GNI. It has been shown that if GI (GA resp.) has polynomial size circuits, then GNI (GNA resp.) is in MA [8]. So, this would be one direction in showing that GNI has an MA protocol.

This concludes my section on known results. Note that I have only included known results that I found to be important and/or pertinent towards the discussion of this paper. For more information see [8].

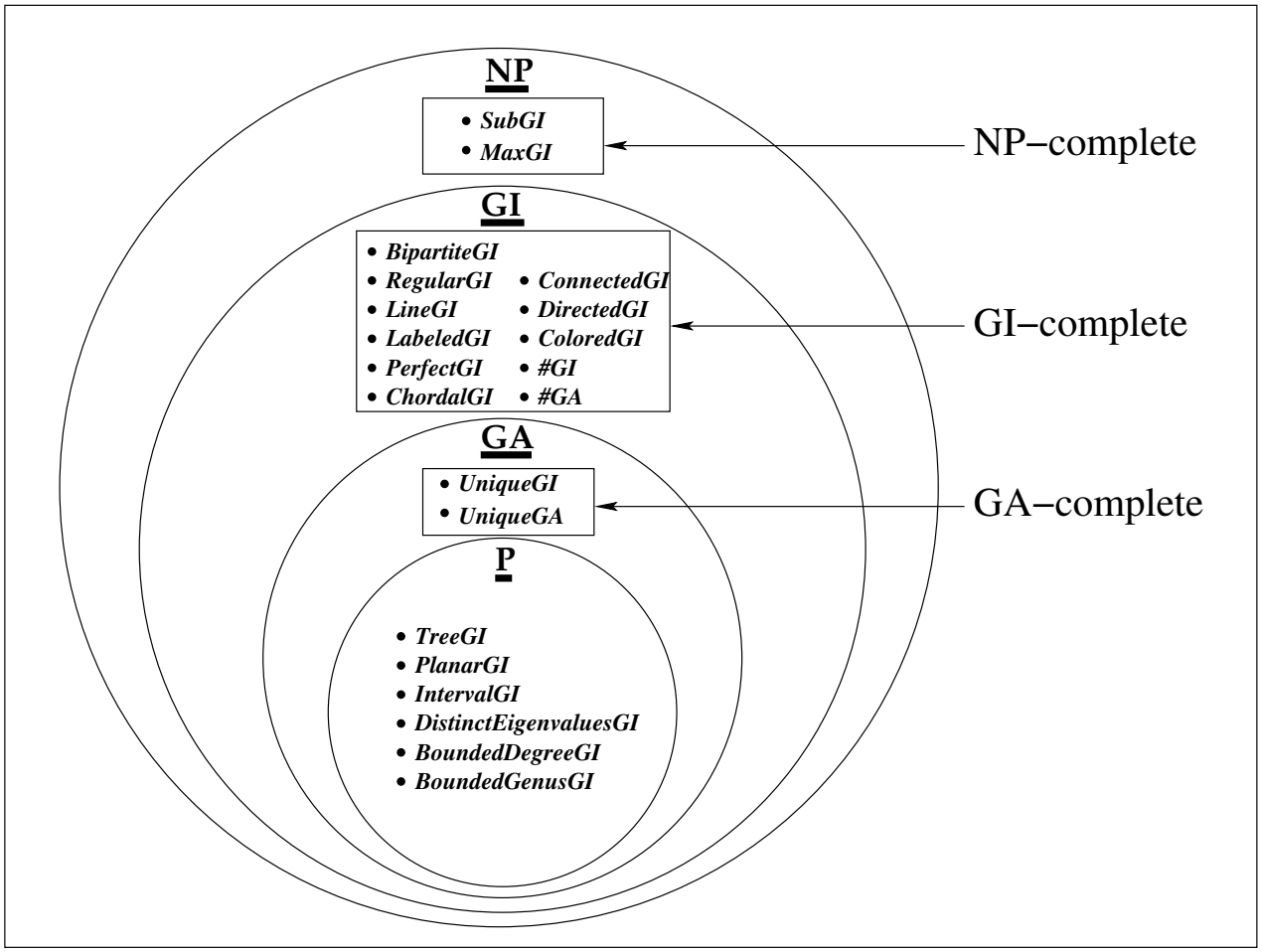

Figure 1: Hypothetical hierarchy of classes and problems relating to Graph Isomorphism [1]. In Section 3, we will define a more general problem of GI ($MaxGI$) and show that it is NP-complete.

3 Harder Problems

Consider the following definitions

Definition 4 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Also, let $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$, where $|S_1| = |S_2|$. A **partial permutation** is a permutation $\pi : S_1 \to S_2$ such that $(u, v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$, for any S_1, S_2. We say that π is a **maximum permutation** if the cardinality of both sets are maximized such that π still holds.*

Now, one can generalize the Graph Isomorphism problem in the following way

Problem:	**MaxGI**
Instance:	Graphs G_1 and G_2. Some integer $k > 0$
Question:	Is there a partial permutation π covering some l vertices in G_1, G_2 (for $l \geq k$) such that their adjacent edges are preserved over those l vertices in G_1, G_2?

Although it is not known where GI lies exactly, we will see that the more general question ($MaxGI$) is NP-complete. To do this, we must first define the following well known NP-complete problem (see [7])

Problem:	**SubGI**
Instance:	Graphs G_1 and G_2.
Question:	Does there exist a subgraph of G_1, call it S_{G_1}, such that $S_{G_1} \cong G_2$?

Theorem 1 *$MaxGI$ is NP-complete.*

Proof.

($MaxGI \in NP$): On input (G_1, G_2, k):

1. Non-deterministically select k vertices in G_1 and k vertices in G_2 and construct induced subgraphs G_1' and G_2', respectively.
2. Non-deterministically guess an isomorphism π between G_1' and G_2'.
3. Check that $(u, v) \in G_1' \iff (\pi(u), \pi(v)) \in G_2'$.
4. If yes, then *accept*. Otherwise, *reject*.

Steps 1 and 2 are done non-deterministically. Step 3 requires at most $O(n^2)$ time to check whether the isomorphism is valid (i.e. pass over every edge incident on every vertex in the subgraphs). Hence, $MaxGI \in NP$.

($SubGI \leq_m^p MaxGI$): The reduction is the following. Consider the graphs G_1 and G_2. We argue that $(G_1, G_2) \in SubGI \iff (G_1, G_2, |G_2|) \in MaxGI$.

Suppose there exists a subgraph of G_1 (call it S_{G_1}) such that $G_2 \cong S_{G_1}$. Clearly, $|G_2| = |S_{G_1}|$. So, by Definition 1, there must be some permutation $\pi : V_1 \to V_2$ that covers $|G_2|$ vertices in both graphs.

Now, let $S_1 \subseteq G_1$ and $S_2 \subseteq G_2$, where $|S_1| = |S_2| = |G_2|$. Suppose there is a partial permutation mapping vertices in S_1 to vertices in S_2 such that their adjacent edges are preserved. So, $S_1 \cong S_2 \implies S_1 \cong G_2$. ∎

4 My Algorithm (First Attempt)

When I initially encountered this problem, I was hoping that there was some way of deconstructing both graphs to get at a polynomial time algorithm. My initial attempts at solving the graph isomorphism problem was provoked by my intuition of the problem and experimental data I had gathered (i.e. my algorithm was at least correct for graphs with vertices < 10). My algorithm works by taking successive powers of the two adjacency matrices in question A_1, A_2 (up to the number of vertices in both graphs, call it n). For each $1 \leq k \leq n$, let $P^v_{A_1}$ be the number of closed paths of length k on some vertex $v \in A_1$ (define this similarly for $P^v_{A_2}$). For each k, check whether there exists a bijection $\phi^k : V(A_1) \to V(A_2)$ such that $P^{v_i}_{A_1} = P^{\phi(v_i)}_{A_2}$ for all $v_i \in V_1$. More formally,

$GRAPH\text{-}ISOMORPHISM(G_1, G_2)$

1. **if** $|G_1| \neq |G_2|$ **or** $\|G_1\| \neq \|G_2\|$
   then return $false$
2. $A'_1 \leftarrow A_1 \leftarrow adjacency[G_1]$
3. $A'_2 \leftarrow A_2 \leftarrow adjacency[G_2]$
4. $n \leftarrow rows[A_2]$
5. Repeat the following steps n times
   (a) $d_1 \leftarrow QUICKSORT(diag[A'_1], 1, n)$
   (b) $d_2 \leftarrow QUICKSORT(diag[A'_2], 1, n)$
   (c) **if** $d_1 \neq d_2$ **then return** $false$
   (d) $A'_1 \leftarrow A'_1 * A_1$

(e) $A_2' \leftarrow A_2' * A_2$

6. **return** *true*

First let's analyze this algorithm to verify that it runs in polynomial time. Step 1 is a simple check that can be done in constant time. Let $n = |G_1| = |G_2|$. In steps 2 and 3 we construct adjacency matrices from G_1 and G_2, which can take place in $O(n^2)$ time since there are at most $O(n^2)$ edges in both graphs. Step 4 is also a simple check that can be done in constant time. In step 5, we sort the diagonals of A_1 and A_2 ($O(n \log n)$), check equality of the sorted vectors ($O(n)$), and compute the next power of both matrices ($O(n^{2.376})$) [4]. Since this is repeated n times, step 5 will take at most $O(n^{3.376})$ time. Hence, the running time of $GRAPH\text{-}ISOMORPHISM$ is $0(n^{3.376})$, which is polynomial in the number of vertices in both graphs.

This completes the time analysis of the algorithm. Now with regards to the proof of correctness we start to run into problems. Clearly, we maintain correctness up until the beginning of the loop, since two graphs with an unequal number of vertices or edges cannot be isomorphic. But, what can we say about the remaining part of the algorithm? Ideally we would like to show that $A_1 \cong A_2$ if and only if there exists a permutation ϕ^k for all $1 \leq k \leq n$, such that $diag(A_1^k) = \phi^k(diag(A_2^k))$. In Section 4, however, we will see that this is not exactly the case. The forward direction is necessarily true, which will be proved in Theorem 2. However, it turns out the reverse direction is in fact false for seemingly rare cases (cospectral regular graphs). Despite this roadblock, we will prove that the reverse direction is true under the assumption that the graphs are non-cospectral. In fact, my algorithm seems to work for most cospectral graphs as well, although I am unable to prove this; the only counterexamples I have managed to find was in the case when G_1 and G_2 were non-isomorphic k-regular cospectral graphs.

Lemma 1 *Let A_1, A_2 be any two matrices and P be some permutation matrix. Then $A_1 = PA_2P^{-1} \Rightarrow A_1^k = PA_2^kP^{-1}$ for all $k \geq 1$.*

Proof. (By induction on k)

$\underline{k = 1}$: This follows from the assumption.

$\underline{k \geq 1}$: Suppose the statement is true for $k - 1$.

$$A_1 = PA_2P^{-1} \Rightarrow A_1^{k-1} = PA_2^{k-1}P^{-1} \tag{1}$$

$$\equiv A_1 = PA_2P^{-1} \Rightarrow A_1^{k-1}A_1 = PA_2^{k-1}P^{-1}PA_2P^{-1} \tag{2}$$

$$\equiv A_1 = PA_2P^{-1} \Rightarrow A_1^{k} = PA_2^{k-1}A_2P^{-1} \tag{3}$$

$$\equiv A_1 = PA_2P^{-1} \Rightarrow A_1^{k} = PA_2^{k}P^{-1} \tag{4}$$

∎

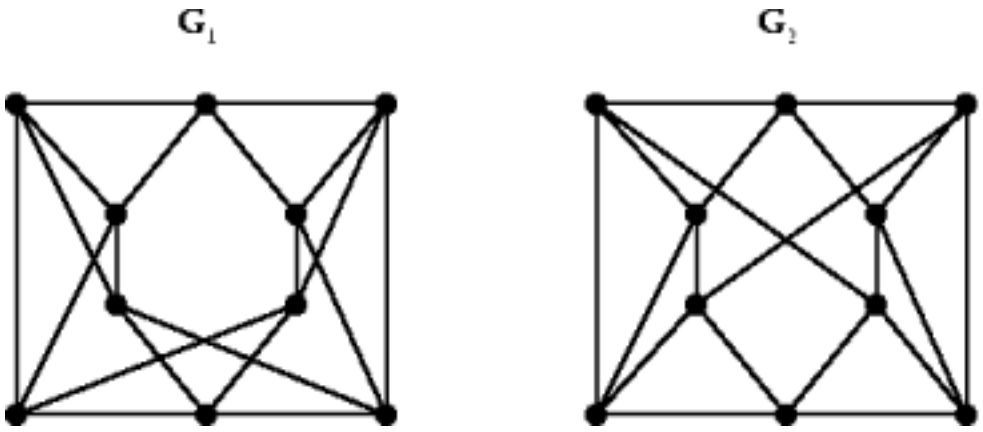

Figure 2: Two k-regular cospectral graphs [6].

Lemma 2 $A_1 \cong A_2 \Rightarrow \exists \phi$ *s.t.* $diag(A_1) = \phi(diag(A_2))$

Proof. Consider order n graphs A_1, A_2. Suppose $A_1 \cong A_2$, which means $A_1 = f(A_2)$ for some permutation π. So, $diag(A_1) = [x_{1,1}, \ldots, x_{n,n}] = [y_{\pi(1),\pi(1)}, \ldots, y_{\pi(n),\pi(n)}] = diag(f(A_2))$, by Definition 1. But, $diag(A_2) = [y_{1,1}, \ldots, y_{n,n}]$. Now, since π is a bijection, there must also be a bijection $\phi : diag(A_2) \rightarrow diag(A_1)$. ∎

Theorem 2 $A_1 \cong A_2 \Rightarrow \forall 1 \leq k \leq n, \exists \phi$ *s.t.* $diag(A_1^k) = \phi(diag(A_2^k))$

Proof. Suppose that $A_1 \cong A_2$. Then there exists some permutation matrix P such that $A_1 = PA_2P^{-1}$. By Lemma 1, this means that for all k, $A_1^k = PA_2^kP^{-1} \overset{(1)}{\Rightarrow} A_1^k \cong A_2^k \overset{(2)}{\Rightarrow} \exists \phi$ s.t. $diag(A_1^k) = \phi(diag(A_1^k))$. (1) follows directly from Definition 1. (2) follows from Lemma 2. ∎

5 Cospectral Graphs

In this section we will explore cospectral graphs and see what about cospectral graphs make finding an isomorphism hard. In general we don't know of any method of generating cospectral graphs efficiently; nor do we know how often cospectral graphs can appear. For more information regarding this, refer to [6] and [3]; both papers discuss this issue in great detail.

The following theorem is what breaks my algorithm defined in Section 4.

Theorem 3 *Let G_1 and G_2 be k-regular graphs. G_1 and G_2 are cospectral if and only if, for any l, the number of closed paths of of length l are the same for G_1 and G_2.*

Proof. See [3] ∎

Theorem 4 *If A_1 and A_2 are non-cospectral graphs, then $A_1 \ncong A_2$.*

Proof. Let's suppose that A_1 and A_2 are non-cospectral graphs. This is equivalent to saying that $tr(A_1^k) \neq tr(A_2^k)$, for some $1 \leq k \leq n$.

$$\Rightarrow \exists 1 \leq k \leq n, \forall \phi \text{ s.t. } diag(A_1^k) \neq \phi(diag(A_2^k))$$
$$\Rightarrow A_1 \ncong A_2 \qquad \text{(follows from Theorem 2)}$$

∎

Theorem 5 *Let A_1 and A_2 be non-cospectral graphs. Then $A_1 \ncong A_2 \Rightarrow \exists 1 \leq k \leq n, \forall \phi$ s.t. $diag(A_1^k) \neq \phi(diag(A_2^k))$.*

Proof. Suppose that $A_1 \ncong A_2$ with A_1 and A_2 being non-cospectral. Then this is equivalent to saying

$$tr(A_1^k) \neq tr(A_2^k), \text{ for some } 1 \leq k \leq n.$$
$$\Rightarrow \exists 1 \leq k \leq n, \forall \phi \text{ s.t. } diag(A_1^k) \neq \phi(diag(A_2^k))$$

∎

Theorem 5 completes the other direction of Theorem 2. Hence, my algorithm clearly works correctly on all non-cospectral graphs. It also appears to work for most cospectral graphs, although this seems much more difficult to prove.

6 My Algorithm (Second Attempt)

In this section we attempt to improve the original algorithm by adding steps (c) and (d) and modifying (e); this will (hopefully) fix the problems we encountered with cospectral k-regular graphs from the previous section. You can easily see that if we input (to the modified algorithm shown below) the two graphs shown in Figure 2, we now return the correct output for these two specific cospectral k-regular graphs. But, does this fix our problems for all cases? Although we can't answer this question for sure, we can at least show that this algorithm returns at least as many "correct" answers as the previous algorithm.

$GRAPH\text{-}ISOMORPHISM(G_1, G_2)$

1. **if** $|G_1| \neq |G_2|$ **or** $||G_1|| \neq ||G_2||$
   then return $false$
2. $A_1' \leftarrow A_1 \leftarrow adjacency[G_1]$
3. $A_2' \leftarrow A_2 \leftarrow adjacency[G_2]$
4. $n \leftarrow rows[A_2]$
5. Repeat the following steps n times
   (a) $d_1 \leftarrow QUICKSORT(diag[A_1'], 1, n)$
   (b) $d_2 \leftarrow QUICKSORT(diag[A_2'], 1, n)$
   (c) $u_1 \leftarrow QUICKSORT(upper[A_1'], 1, n)$
   (d) $u_2 \leftarrow QUICKSORT(upper[A_2'], 1, n)$
   (e) **if** $d_1 \neq d_2 \vee u_1 \neq u_2$ **then return** $false$
   (f) $A_1' \leftarrow A_1' * A_1$
   (g) $A_2' \leftarrow A_2' * A_2$
6. **return** $true$

Clearly the correctness up until the beginning of the loop shall not change, since we did not make any modifications to that part of the algorithm. As for the analysis, the runtime of the algorithm will also not change, since the changes we made take at most $O(n^2 \log n^2) = O(n^2 \log n)$ time, which is dominated by steps (f) and (g) (known to take $O(n^{2.376})$ time).

Now, we will show that my modified algorithm is still correct (in one direction).

Lemma 3 $A_1 \cong A_2 \Rightarrow \exists \phi \ s.t. \ upper(A_1) = \phi(upper(A_2))$

Proof. Consider order n graphs A_1, A_2. Suppose $A_1 \cong A_2$, which means $A_1 = f(A_2)$ for some permutation π. So, $upper(A_1) = upper(f(A_2))$. Consider some $y_{\pi(i),\pi(j)} \in upper(f(A_2))$ for some i, j. Now it is possible that $\pi(i) > \pi(j)$ (but $\pi(i) \neq \pi(j)$ by Definition 3). Hence some elements in $upper(f(A_2))$ may map to elements that are actually in the lower triangular part of A_2. But, since A_1 and A_2 are undirected graphs, $y_{\pi(i),\pi(j)} = y_{\pi(j),\pi(i)}$. And so every lower triangular element is equal to an upper triangular element such that the one-to-one mapping is still preserved. Hence, there must be some bijection ϕ that maps elements of $upper(A_2)$ to elements of $upper(A_1)$. ∎

Theorem 6 $A_1 \cong A_2 \Rightarrow \forall 1 \leq k \leq n, \exists \phi, \pi \ s.t. \ upper(A_1^k) = \pi(upper(A_2^k))$

Proof. Suppose that $A_1 \cong A_2$. Then there exists some permutation matrix P such that $A_1 = PA_2P^{-1}$. By Lemma 1, this means that for all k, $A_1^k = PA_2^kP^{-1} \overset{(1)}{\Rightarrow} A_1^k \cong A_2^k \overset{(2)}{\Rightarrow} \exists\phi$ s.t. $upper(A_1^k) = \phi(upper(A_1^k))$. (1) follows directly from Definition 1. (2) follows from Lemma 3. ∎

Theorem 7 $A_1 \cong A_2 \Rightarrow \forall 1 \leq k \leq n, \exists\phi, \pi \; s.t. \; diag(A_1^k) = \phi(diag(A_2^k)) \wedge upper(A_1^k) = \pi(upper(A_2^k))$

Proof. This follows immediately from Theorem 2 and Theorem 6. ∎

Open Problem 1 *Is the following statement true?*

$$\forall 1 \leq k \leq n, \exists\phi, \pi \; s.t. \; \underbrace{diag(A_1^k) = \phi(diag(A_2^k))}_{(1)} \wedge \underbrace{upper(A_1^k) = \pi(upper(A_2^k))}_{(2)} \Rightarrow A_1 \cong A_2$$

I had attempted to prove the above claim, but unfortunately I didn't get very far. The proof attempt was this. Assume A_1 and A_2 are regular graphs. Clearly, if we prove this statement holds for regular graphs, then we have shown that $GI \in P$, since the Graph Isomorphism problem is GI-complete for regular graphs [2, 5]. So, the proof is by contradiction. Suppose $A_1 \ncong A_2$. Also, let's suppose A_1 and A_2 are not cospectral. Now, condition (1) must have been violated by Theorem 3, giving us our contradiction. But, suppose A_1 and A_2 are cospectral. Now, condition (1) will always hold true (by Theorem 3), which means we will need to show that condition (2) must be violated to reach a contradiction. But, how do we do this?

7 Open Problems

With respect to the Graph Isomorphism problem, the biggest question that remains open today asks whether $GI \in P$. I have tried to resolve this issue, but I am currently at an impasse. The question is whether my algorithm precisely determines whether two graphs are isomorphic. Clearly, I have proved that if my algorithm returns *false*, then the two graphs in question are indeed not isomorphic. However, what if my algorithm returns *true*? Can we arrive at the conclusion that the two graphs in question are indeed isomorphic? The answer to this question remains open. Despite this, we can still ask questions regarding the complexity of GI. Bipartite perfect matching (BPM) seems to share similar properties with GI. The hardness results for both problems are currently weak [11]. It is known that BPM is randomly reducible to GI [11]. But, is $BPM \leq_m^p GI$ (or $BPM \leq_m^p GA$)? Also, is the complexity class GI closed under complementation?

GNI is another interesting and important point of discussion. Clearly, $GNI \in$ *co-*NP, but is $GI \in NP$? It has been shown that $GI \in AM$ [8], but it seems like we should be able to do better

than this. Can we at least show that $GI \in MA$? Clearly, if we showed that $GNI \in NP$, this would be a monumental step towards showing that $GI \in P$; this would imply that $GI \in NP \bigcap co\text{-}NP$, which would give us further evidence showing that GI is probably not NP-complete. If it were, $NP = co\text{-}NP$.

Also, going back to cospectral graphs, how often do cospectral graphs occur? If we can prove that cospectral graphs only occur with probability $\leq 1/2$, then my algorithm would at least be an RP algorithm. Now, I suspect that the only case in which my original algorithm failed was when the two graphs in question were non-isomorphic k-regular cospectral graphs. If we could somehow prove this, then to get an RP algorithm, we would only have to show that the probability of randomly producing two k-regular cospectral graphs is $\leq 1/2$. Moreover, if we could prove that the second proposed algorithm fixes the case for all cospectral k-regular graphs, then the entire problem is solved and $GI \in P$.

8 Acknowledgments

I wish to acknowledge H. Venkateswaran for his helpful insight and guidance throughout my research on this problem.

References

[1] B. Jenner, J. Köbler, P. M. J. T. Completeness results for graph isomorphism. Journal of Computer and System Sciences 66 (2003), 549–566.

[2] Booth, K. Problems polynomially equivalent to graph isomorphism. Tech. Rep. CS-77-04, University of Waterloo, Ontario, Canada, 1979.

[3] Brooks, R. Isospectral graphs and isospectral surfaces.

[4] D. Coppersmith, S. W. Matrix multiplication via arithmetic progressions. J. Symbolic Computation 9 (1990), 251–280.

[5] D. Corneil, D. K. A theoretical analysis of various heuristics for the graph isomorphism problem. SIAM Journal of Computing 9 (1980), 281–297.

[6] E. R. van Dam, W. H. H. Which graphs are determined by their spectrum? Linear Algebra Appl. (in press).

[7] Garey, M. S., and Johnson, D. S. Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Company, New York, 1979.

[8] J. Köbler, U. Schöning, J. T. The Graph Isomorphism Problem: Its Structural Complexity. Birkháuser, Boston, 1993.

[9] Ladner, R. E. On the struture of polynomial-time reducibilities. Journal of the ACM 22 (1975), 155–171.

[10] Schöning, U. A uniform approach to obtain diagonal sets in complexity classes. Theoretical Computer Science 18 (1982), 95–103.

[11] Torán, J. On the hardness of graph isomorphism. FOCS (2000), 180–186.