Thesis Proposal on the Graph Isomorphism Problem

Grady Ward

November 20, 2015

Background

The majority of attempts at solving the Graph Isomorphism Problem *(GI)* have been based upon *node invariants*, functions which take in a node in a given graph, and calculate a numerical value that describes some aspect of that node, or its position within the graph. These properties are called invariants because they are deterministically calculated, so that an isomorphic pair of vertices in isomorphic graphs will have the same value for any and all invariants calculated over them. Thus, if node n in graph G is isomorphic to node m in graph H, all invariants will return the same numerical value when they are given n and m. Within the context of the *Graph Isomorphism Problem*, we use these functions to narrow down the nodes in one graph could be isomorphic to a given node in the other.

Different invariants have different levels of *discerning power*, and require different levels of computational complexity. For example, the *base invariant*, the degree of a node, certainly can rule out many potential isomorphic mappings, and can be calculated in constant time. However, a second invariant, which deterministically describes the degrees in the level order sets within a breadth first search from the target node (one used in some of my early exploration of this topic), obviously has significantly more discriminatory power in differentiating between nodes that might be isomorphic, but runs in quadratic time. Invariants can be used together to strengthen their discerning power by the fact that that we can calculate many separate invariants on a node, and through combinatorics, combine them into a singular unique, numerical description that encodes all of the calculated invariants. This kind of encoding capabilities might lead one to the thought that some set of polynomial-time invariants might be able to fully discriminate between nodes which are truly isomorphic, and those which have any dissimilarities in their relations within the graph, thus solving the Graph Isomorphism problem by classifying it within P. However, despite many attepts at a solution to GI through these means, none have been successful.

In this proposal, I am first going to lay out what I have done on this problem, then describe a proposed algorithm that I think there is strong evidence to support, and finally outline my thesis goals and fallback strategy. This is an inherently risky project, but firstly, I am not afraid of failing at my thesis (the worst that can come of it is I graduate without departmental honors) and secondly, I think there is a good amount of evidence to support this algorithm's approach and findings, and I would be disappointed in myself if I didn't pursue it.

Quantifying Discerning Power

I began my work on this problem by computationally examining the relative discriminatory power of various sets of invariants. When I say computational, I am referring to finding the least complex graph for which the given invariant set failed to consistently produce a valid isomorphism when one existed. I used SageMath's nauty generator to systematically generate specific types of graphs, and test an isomorphism algorithm against the original graph and randomly scrambled isomorphisms. Each graph was compared against fifty of its scrambled isomorphisms, and if any of them failed to correctly produce a valid mapping, it was labeled a failure.

Though the use of invariants in GI solving is not difficult to explain, the algorithms that invariants feed their data into are more complex and varried. My algorithm used invariants as the basis of a *ranking algorithm*, which aims to rank the nodes in a graph in a deterministic order, based upon invariants over the nodes of the graph, and the relations among already ranked nodes. If you can deterministically rank the nodes of a graph (thus doing it in exactly the same manner for every isomorphic input), constructing an isomorphism becomes a trivial problem of pairing the nodes which have the same ranking.

This ranking algorithm first split the nodes of a graph into different equivalency sets, where each equivalency set (or eSet for short) housed all nodes with the same numerical result(s) from the studied invariant(s). Based on the ranks of its neighbors, an iterative process attempted to extract all meaning out of the eSets, including their relative ranks.

What I found was that using this kind of elaborate ranking scheme allowed very simple invariants to have significant degrees of discriminatory power. The degree of the node, in addition to the level order topological sort (and its rankings), was enough to recognize all isomorphisms in graphs of size 12 and fewer. More complex invariants (as anticipated), had even more descriptive power, but all were shown to fail when tested against large, regular graphs. For each set of invariants, the only real question seemed to be finding the point at which each would fail to find an isomorphism where one exists.

Despite trying dozens of sets of invariants of relatively high polynomial complexity, for each I have been able to find a graph which does not satisfy any set that I have tried. Disproving the validity of larger sets of invariants demanded counterpoint graphs which were larger in size. This result is intiutively pleasing, and if given more time to grapple with this subject, I would like to consider more thorough ways of quantifying the computational discriminatory power of invariants.

However, the fact remains that nobody (including people much smarter than me) has ever found a static set of invariants that they can prove determines the graph. This has led me to a hard to prove hypothesis about the graph isomorphism problem: that for any finite set of invariants, there exists a graph which those set of invariants cannot process correctly. This hypothesis presents two clear options for further exploration: either embrace that $GI = NP$, and attempt that proof, or try to find a non-static set of invariants which can be used to determine the graph.

Beyond Discerning Power

I decided to follow the second path, and started in the way that many others have, by viewing a graph as an adjacency matrix (or rather many), which describe the interactions (edges) between every pair of vertices within a graph. For any given graph there exist an exponential number of matrices which fully describe it. Precisely, for a graph with a automorphisms, there exist $\frac{n!}{a}$ different adjacency matrices that correctly describe it. The matrix you use to express the graph simply depends on how you order your nodes, and thus the direct comparison of these matrices is no more valuable in the search for an isomorphism than the observation of our graph. However, these matricies allow us to perform deterministic operations on the graph which can prove to be invariant, unaffected by the ordering of the nodes.

The operation that I have been alluding to is raising our square adjacency matrix to some power P, through repeated matrix multiplication. Each one of the entries in each of the matrices in this sequence has a simple conceptual description: if A is our adjacency matrix, an entry (a, b) in matrix A^p describes the number of paths from node a to node b of length p. If we apply this description to the diagonal of the matrix each of these values represents the number of closed paths of size p that travel through the given node. It is also clear that this property is invariant across how you express your graph or order your nodes. The bag of these numbers is a kind of $matrixinvariant$.

Up until this point, what I have described is non-unique to my work. Beyond here, my reasoning differs from that of others, and this is the perspective that I am hoping to write my thesis from.

My Proposal

Rather than viewing the matrix as having a matrix invariant, we can reassign the values in the diagonal back to their referenced node, and use that as a kind of invariant vector. For example, if we take the (k, k) element of of every matrix in the range $[1, P]$, we are left with a vector of size P, which we would assign to node k as its invariant. we can treat these vectors as numbers, as they can be sorted and trivially compared in linear time.

Note that each of these vector invariants has some appealing properties that are either commonly known invariants, or can be used to produce commonly used invariants:

- The first element tells us whether or not the node is self connected.
- The second element can be used to calculate the degree of the node.
- The third element tells us the number of triangles that the node rests on (when taking the first two vector values into account).
- Similarly, higher level elemnts can describe other geometrical shapes (again, when taking the lower level calculations in to account).

Every element within the vector has some kind of graphical description (many of which are commonly used as invariants in GI solvers). This should be our first hint that this kind of analysis may produce an invariant with enough strength to solve GI.

Until now I have glossed over a key question: how large should we make P (i.e. how many times should we exponentiate our adjacency matrix A)? My intuition (though I haven't fully settled on this number) is that it would be best if $P \geq 2 * V$, where V is the number of vertices of the graph. This logic is simple: I think that in order to be maximally effective, the invariant vector must have information from all possible paths within the graph. If we imagine the case where the graph is a continuouslinked list, making P any less than $2 * V$ would mean that the invariant vector associated with an end element in the list would not have any paths described which reach the last element in the list (V transitions traversing to the last element, V transitions back).

This choice of P as a function of V is hugely consequential. It means that we get to use a dynamic number of invariants, a number that scales with the number of vertices we have. If my first hypothesis is true, then this property steers us clear of it, and presents a compelling conceptual case for this algorithm as a polynomial time solution of GI.

Before I describe my first few attempts at a proof, I want to briefly describe the running time of the algorithm I am proposing. My ranking algorithm runs in $O(n^2)$ after the invariants have been calculated. Matrix multiplication can naively be calculated in $O(n^3)$, but a series of clever algorithms have dropped that coefficient dramatically (Strassen, Schonhage, Williams). Modern approaches can run as quickly as $O(n^{2.4})$. Since we are doing a linear number of matrix multiplications ($2V$), the total cost of the matrix multiplication operations is in $O(n^{3.4})$. Thus the overall complexity of this algorithm is $O(n^{3.4})$.

Theoretical Support for Proposed Algorithm

Defining the Proof Task

For the purposes of simplification, lets assume that the graph is not self-connected: no vertices in the graph are connected to themselves. Then our original adjacency matrix will have zeroes on the diagonal. Lets also assume that we write our adjacency matrices with the standard convention of 1's representing edges. Finally, lets assume we are only examining undirected graphs (so our matrix is symmetric). These are all simplifying assumtions that can easily be reversed if we can solve this base level problem.

Under these assumptions, if our graph had four nodes, we would be examining a sequence of matrices that look something like this:

$$A = \begin{bmatrix} 0 & x_1 & x_2 & x_3 \\ x_1 & 0 & x_4 & x_5 \\ x_2 & x_4 & 0 & x_6 \\ x_3 & x_5 & x_6 & 0 \end{bmatrix}, \quad A^2 = \begin{bmatrix} k_{2,1} & \dots & \dots & \dots \\ \dots & k_{2,2} & \dots & \dots \\ \dots & \dots & k_{2,3} & \dots \\ \dots & \dots & \dots & k_{2,4} \end{bmatrix}$$

$$A^3 = \begin{bmatrix} k_{3,1} & \dots & \dots & \dots \\ \dots & k_{3,2} & \dots & \dots \\ \dots & \dots & k_{3,3} & \dots \\ \dots & \dots & \dots & k_{3,4} \end{bmatrix}, \quad A^P = \begin{bmatrix} k_{P,1} & \dots & \dots & \dots \\ \dots & k_{P,2} & \dots & \dots \\ \dots & \dots & k_{P,3} & \dots \\ \dots & \dots & \dots & k_{P,4} \end{bmatrix}$$

Our task is to prove that if you are given each of the elements $k_{i,j}, i \in [1, V], j \in [1, P]$, you can solve for each of the variables x_i in the original adjacency matrix, and come to a unique solution, if one exists. *If this property holds, it would prove the validity of my algorithm, as it would prove that a* *set of vectors specifying the diagonal of a matrix across repeated self multiplication necessarily specifies the matrix itself*, showing that if we are given the diagonals of all matrices from 1 to P, we can deduce the full adjacecny matrix. In graphical terms, this translates to saying: knowing the number of closed paths of sizes $[1, P]$ through every given node is enough information to describe the full structure of the graph.

How do we go about proving this property? I am not fully certain, but examination of the first few cases leads one to believe it can be done through induction. Constructing this proof is where I think my thesis could lie, and is the important piece of an otherwise unremarkable idea. I am going to briefly go through proofs which show that (for the first three sized graphs $V \in 2, 3, 4$, a linear number of diagonals are enough to solve for a singular solution for our adjacency matrix.

Two-Node Graph

In the two node case, we have a very simple adjacency matrix. When we exponentiate it once, we get following matrix with one unknown.

$$\begin{bmatrix} 0 & x_1 \\ x_1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & x_1 \\ x_1 & 0 \end{bmatrix} = \begin{bmatrix} x_1^2 & 0 \\ 0 & x_1^2 \end{bmatrix}$$

If $k_{2,1} = 0$ then $x_1 = 0$. if $k_{2,1} = 1$, then $x_1 = 1$. Note that with $V = 2$, it takes $p = 2$ exponentiations to fully determine the matrix. Notice that here we are using little p to indicate that this is fewer than the $P = 2 * V$ that was described earlier.

Three-Node Graph

Now lets examine the case with three variables:

$$\begin{bmatrix} 0 & x_1 & x_2 \\ x_1 & 0 & x_3 \\ x_2 & x_3 & 0 \end{bmatrix} * \begin{bmatrix} 0 & x_1 & x_2 \\ x_1 & 0 & x_3 \\ x_2 & x_3 & 0 \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 & x_2 * x_3 & x_1 * x_3 \\ x_2 * x_3 & x_1^2 + x_3^2 & x_1 * x_2 \\ x_1 * x_3 & x_1 * x_2 & x_2^2 + x_3^2 \end{bmatrix}$$

Setting each of the diagonals to its known value, $k_{2,j}$ yields us a perfectly specified system of linear equations:

$$\begin{cases} x_1^2 + x_2^2 = k_{2,1} \\ x_1^2 + x_3^2 = k_{2,2} \\ x_2^2 + x_3^2 = k_{2,3} \end{cases}$$

Though these equations are not strictly linear, remember that we know that each one of our $x_i's$ is either 0 or 1. Thus, the above equations can easily be simplifed by removing the squares, as both zero and 1 are their own square roots:

$$\begin{cases} x_1 + x_2 \qquad\quad = k_{2,1} \\ x_1 \qquad\; + x_3 = k_{2,2} \\ \qquad x_2 + x_3 = k_{2,3} \end{cases}$$

This system of equations is perfectly specified, meaning that it will either yield a single solution to the problem, or will show that no solution to the problem exists for the given k-vector. This is ideal, because it proves that if we are given $k_{1,j}$ for any three graph, we can fully determine the adjacency matrix. Note that with $V = 3$, it takes $p = 2$ exponentiations to fully determine the matrix.

Four-Node Graphs

As a final case for why this should be able to work, examine the four-node scenario (note that we immediately make the substitution of the variable for all squared variables, as we know they are binary):

$$A = \begin{bmatrix} 0 & x_1 & x_2 & x_3 \\ x_1 & 0 & x_4 & x_5 \\ x_2 & x_4 & 0 & x_6 \\ x_3 & x_5 & x_6 & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} x_1 + x_2 + x_3 & x_2x_4 + x_3x_5 & x_1x_4 + x_3x_6 & x_1x_5 + x_2x_6 \\ x_2x_4 + x_3x_5 & x_1 + x_4 + x_5 & x_1x_2 + x_5x_6 & x_1x_3 + x_4x_6 \\ x_1x_4 + x_3x_6 & x_1x_2 + x_5x_6 & x_2 + x_4 + x_6 & x_2x_3 + x_4x_5 \\ x_1x_5 + x_2x_6 & x_1x_3 + x_4x_6 & x_2x_3 + x_4x_5 & x_3 + x_5 + x_6 \end{bmatrix}$$

Setting up the system of equations:

$$\begin{cases} x_1 + x_2 + x_3 = k_{2,1} \\ x_1 + x_4 + x_5 = k_{2,2} \\ x_2 + x_4 + x_6 = k_{2,3} \\ x_3 + x_5 + x_6 = k_{2,4} \end{cases}$$

Which is underspecified (i.e. we cannot deduce a single solution from it). However, if we look at the second exponentiation:

$$A^3 = 2\begin{bmatrix} x_1x_3x_5 + x_1x_2x_4 + x_2x_3x_6, \ \ldots, \ \ldots, \ \ldots \\ \ldots, x_1x_3x_5 + x_1x_2x_4 + x_2x_3x_6, \ \ldots, \ \ldots \\ \ldots, \ \ldots, x_1x_3x_5 + x_1x_2x_4 + x_2x_3x_6, \ \ldots \\ \ldots, \ \ldots, \ \ldots, x_1x_3x_5 + x_1x_2x_4 + x_2x_3x_6 \end{bmatrix}$$

This may not look any better, but in fact, it has a beautiful symmetry that is exposed if we make the following substitutions: $y_1 = 2x_1x_2x_4, \ \ y_2 = 2x_1x_3x_5, \ \ y_3 = 2x_2x_3x_6, \ \ y_4 = 2x_4x_5x_6$, we get back the following perfectly specified system of equations when we set the diagonal to our k vector.:

$$\begin{cases} y_1 + y_2 + y_3 \qquad = k_{3,1} \\ y_1 + y_2 \qquad + y_4 = k_{3,2} \\ y_1 \qquad + y_3 + y_4 = k_{3,3} \\ \qquad y_2 + y_3 + y_4 = k_{3,4} \end{cases}$$

Which (as it is perfectly sepcified, will have a single solution, or find that no such solution exists. This means that we can solve for our y's, but doesn't nescessarily specify our x's. That transformation requires us to utilize the fact that our $x_i's$ are binary. Based on that knowledge, we know each of the y_i's could only take on two possible values (zero or two). We will finish this examination by examining two cases, based on whether any of the y's are equal to two:

- If any of the y_i's are equal to two, we know that each of their component parts is equal to 1. This would give us three of the six x_i's we are attempting to solve for, and, in combination with the system of four linear equations we gleaned from the first exponentiation, we have already overspecified our system, leading to either zero or one solution. Thus, in this case, we can immediately solve for our adjacency matrix.

- If all of the y_i's are equal to zero, that tells us that the graph is triangleless. Very few graphs with four vertices are triangleless, so we can use a proof by cases to prove uniquenes in this case. Using the first vector of $v_{2,j}$, there are four possible sets of vertex degrees, each of which translate into a uniquely defined matrix by trivial transitions:

$$\{0,0,0,0\}, \{1,1,0,0\}, \{1,1,1,1\}, \{2,1,1,0\}, \{2,2,1,1\}, \text{and}\{2,2,2,2\}$$

.

Through a much more complicated *(and ad hoc)* proof, we have shown that for graphs of four vertices, we can always solve for the original component $x_i's$ given the diagonals of the exponentiated matrix through three exponentiations.

Summary

Through these three ad-hoc proofs, we have shown an interesting property. For $V = 2$, we require p be greater than or equal to 1 in order to fully deterimine the original matrix. For $V = 3$, we require $p \geq 2$ to solve for a unique solution. For $V = 4$, we require $p \geq 3$ to fully solve for the original adjacency matrix.

There is more than a hint of linearity in these numbers. I think that it is very possible that the diagonals of a linear number of matrix exponentiations could fully deterimine the original matrix, and I want to use this as my thesis topic.

Higher Order Graphs

I am close to a generalized proof, but there are requirements on domains that I haven't quite figured out yet. The beauty of solving this kind of problem comes from the fact that all of our x_i's are binary. This has the useful property that

$$x_i^p = x_i \ \ \forall p \geq 1$$

This might not sound too important, but when we have a complicated polynomial expression in terms of x_i's, we will be able to simplify it to a solution in terms of the given x_i for classical substitution without any difficulty. This is not nescessarily possible (and likely yields multiple solutions) if you have a a degree on your variable other than 1. For example, this complex function (found in the anlysis of a fourth exponentiation of a four element matrix)

$$x_1^4 + 2x_1^2x_2^2 + 2x_1^2x_3^2 + x_1^2x_4^2 + x_1^2x_5^2 + 2x_1x_2x_5x_6 + 2x_1x_3x_4x_6+$$

$$x_2^4 + 2x_2^2x_3^2 + x_2^2x_4^2 + x_2^2x_6^2 + 2x_2x_3x_4x_5 + x_3^4 + x_3^2x_5^2 + x_3^2x_6^2 = k_{4,1}$$

can be made useful by our ability to reduce polynomial degrees. We can simplify that beastly equation to the following:

$$x_1 + x_2 + x_3 + 2x_1x_2 + 2x_1x_3 + x_1x_4 + 2x_2x_3 + x_1x_5 + x_2x_4 + x_2x_6+$$

$$x_3x_5 + x_3x_6 + 2x_1x_2x_5x_6 + 2x_1x_3x_4x_6 + 2x_2x_3x_4x_5 = k_{4,1}$$

Note that if we had tried to solve the above equation without a polynomial reduction, it would have resulted in multiple answers, and there may have not been an efficent way of solving for an answer. However when we solve any equation with no self-multiplied terms, we are guarenteed to come to a singular solution for that variable by using the elementary technique of seperation and division. In this case, when solving for x_2, we get the following result:

$$x_2 = \frac{k_{4,1} - x_1 - x_4 - x_5 - x_1x_3 - 2x_1x_4 - 2x_1x_5 - x_3x_5 - 2x_4x_5 - x_4x_6 - x_5x_6 - 2x_1x_3x_4x_6)}{(x_1 + x_4 + 2x_1x_5x_6 + 2x_3x_4x_5)}$$

so long as $x_1 + x_4 + 2x_1x_5x_6 + 2x_3x_4x_5 \neq 0$. If we for a moment ignore the entire idea of domains, orthogonality and complications of variable sets, and simply think about what this kind of substitut-ability means, we would be able to systematically remove variables from our system of equations, even when those equations are non-linear. Again, ignoring the pragmatic concerns of this approach, this would confirm our intutition of a linear number of exponentiations + matrix diagonals determining the original matrix. Since we need as many equations as variables (and the number of unknown values in our matrix $U = \frac{V(V-1)}{2}$), and each exponentiation of the matrix (starting at 2) yeilds us V different equations (setting each member of the diagonal equal to its $k_{i,j}$ value), we can quickly deduce that if all equations are non-paralell, and all equations can yeild us a valid substitution domain and that each one can be used to substitute for a different variable, then:

$P = 2 + \frac{ciel(V-1)}{2}$, the number of exponentiations that *might* determine the matrix.

There are three nescessary conditions for this to be a valid approach. Each requires further investigation and research.

- We don't know if the domain of such a substitution is valid until we have our individual k_i's. It is very possible that we will be able to construct valid substitutions if we are given enough equations, but some might pose insurmountable domain restrictions. Luckily there is a tool in our back pocket for any difficulties we have with this element. If we are not able to find a sutible nubmer of substituitons that satisfy our domain restrictions, we can simply invert the graph, and compare the inverted graphs for ismorphisms. The more ones in our adjacency matrix, the fewer domain restrictions we will have. I have already began work on proving why the symmetry in matrix multiplicaiton mandates that a matrix with half or more 1's will result in enough valid substitutions. I think this proof might be the bulk of the overall proof.

- We need to be judicious about ensuring that a given number of equations gives us enough substituting power to solve the given system of equations. Particularly we need to be contientious of sub-groups of variables (if they exist), and need to proove systematically that the symmetry of matrix multiplication ensures valid and non-sparse substitutions.

- We need to be sure that any set of equations we set up maintains enough non-paralellism to avoid duplicate equations. For example, if we look at the third exponentiation of a 3x3 matrix, all three equations one could generate from that exponentiation are the same, and mandate that $k_{3,1} = k_{3,2} = k_{3,3}$. (Intuitively, this makes lots of sense, in a three node graph, if there is one triangle, it must include all of the vertices and all of the edges). Detrmining exactly when this kind of paralellism happens is critical, because substitutions from paralell equations gleans us no new information. Luckily, it appears to happen inconsistently, and only in lower order graphs. I will quantitatively make sure that is true.

Proposal

I have presented you with three ad-hoc proofs which clearly not valuable in and of themselves. They lack refinement, symmetry, and any hint of induction, all of which would be key to a correct proof of the desired property. However, they shed light on the potential for our generalized proof, grounded in substitution and polynomial reduction. I think there is a good chance that such an approach will work, and it is important to note that our theorized general proof, our concrete ad-hoc proofs, and the intution behind a variable sized invariant all point to the same possibility: the diagonal of the repeatedly squared adjacency matrix can likely give us enough information to solve the graph isomorphism problem in polynomial time.

For my thesis, I want to pursue this question, with three large sub-questions. I think that a proof of a solution to any of these hypotheses (or their disproof) would be a signifigant piece of work, and that they can be accomplished over the course of several months.

- Can any static set of invariants satisfy all graphs? (Including expontentially calculated ones?) For any polynomial time invariants, can we describe in a rigorous way why they must nescessarily be incomplete?

- Given a series of matrix diagonals from exponentiated graphs, how many do we need to specify the original matrix? Is that question even possible to determine for any value of N? What can we say about the domain of substitutions within this approach?

- Is the graph isomorphism problem in P? If this approach does not work, that doesn't disqualify this question...

I think that this project has potential, and I haven't seen any other attempts at solving it in a similar way. But I want to reiterate: I don't mind being wrong. I don't care if this doesn't turn up anything conclusive, and I don't mind if I don't graduate with honors. In the last two years, what has made me academically happy (not grades, grants or jobs) has been pursuing what interests me. I don't mind being wrong because I would rather know that my hypothesis failed than remain unsure of whether or not it is true.