CS-2008-08

Reductions of Graph Isomorphism Problems

Margareta Ackerman

Technical Report 08

David R. Cheriton School of Computer Science, University of Waterloo.

REDUCTIONS OF GRAPH ISOMORPHISM PROBLEMS

MARGARETA ACKERMAN

David R. Cheriton School of Computer Science, University of Waterloo.
mackerma@uwaterloo.ca

Abstract. We present a reduction between subgraph isomorphism and minimal partial subgraph isomorphism. We also provide a new reduction between graph isomorphism and minimal partial graph isomorphism. The new reduction is more efficient and simpler than the previous reduction and can be generalized to subgraph isomorphism. In addition, we show that a reduction from graph isomorphism that makes only one call to an oracle that finds a minimal partial isomorphism exists only if graph isomorphism is in P. We also provide a generalization of the result.

1. Introduction

An *isomorphism* ϕ between graphs G and H is a set of ordered pairs $(x, y) \in V(G) \times V(H)$ (*isomorphism pairs*) such that for all $x_1, x_2 \in V(G)$, x_1 is adjacent to x_2 if and only if $\phi(x_1)$ is adjacent to $\phi(x_2)$. Given graphs G and H, the function k-GI returns k ordered pairs which are a subset of some isomorphism between G and H, if G and H are isomorphic. If G and H are not isomorphic, it returns an arbitrary subset of k elements in $V(G) \times V(H)$. Gál, Halevi, Lipton, and Petrank proved that GI is polynomial-time reducible to $(3+\epsilon)\log n$-GI for some constant $\epsilon > 0$ where n is the number of vertices in G [1]. Große, Rothe, and Wechsung tightened this result by showing that GI is polynomial-time reducible to 1-GI [2]. We can say that 1-GI returns the minimal (non-empty) partial graph isomorphism (if one exists). Since the latter reduction attaches cliques to vertices, we call it the *clique-padding reduction*.

We present a more efficient and simpler reduction from GI to 1-GI. Our reduction uses colouring and thus is referred to as the *colour reduction*. The colour reduction is $O(n^3)$, improving on the quadratic running time of the clique-padding reduction[1].

[1]Although in [2] it is written that the clique-padding reduction is cubic, the reduction takes $O(n^4)$ due to the expense of writing down the cliques.

We present a reduction from subgraph isomorphism (SI) to 1-SI, the problem of finding an isomorphism pair of a graph and any subgraph of another graph. The reduction for SI is a modification of the colour reduction for GI. The clique-padding reduction from GI to 1-GI does not generalize to a reduction from SI to 1-SI, since a clique of size k is a subgraph of any clique of size greater than k. Moreover, the reduction from SI to 1-SI implies that 1-SI is NP-complete.

The clique-padding reduction and the colour-reduction make a linear number of calls to 1-GI. As pointed out in [2], a reduction that makes only one call to 1-GI is desirable since it would be better suited to fault-tolerance computing, therefore implicitly asking whether such reduction exists. We found that if such a reduction exists, then GI $\in$ P. We also prove a generalization of this result.

2. The Colour Reduction

The clique-padding reduction stores the information that a vertex v is adjacent to certain removed vertices by attaching large cliques to v. The information that certain removed vertices were originally adjacent to non-removed vertices can be stored by colouring vertices instead of attaching cliques. The colour reduction is more efficient and simpler than the previous reduction. We define a colour-preserving function 1-colour-GI which, as we prove, is polynomial-time reducible to 1-GI. Then, we provide a simple polynomial-time reduction from GI to 1-colour-GI. The composition of the reduction from GI to 1-colour-GI with the reduction from 1-colour-GI to 1-GI yields a polynomial-time reduction from GI to 1-GI. With a few minor modifications, we can combine the reduction from GI to 1-colour-GI with the reduction from 1-colour-GI to 1-GI to efficiently reduce GI to 1-GI.

A *coloured graph* is one in which each vertex is assigned a colour. A *colour-preserving isomorphism* ϕ of graphs G and H is an isomorphism of G and H such that if $\phi(a) = b$, then a and b have the same colour. 1-colour-GI is a 1-GI function that preserves colours.

Definition 1. 1-colour-GI *is a function that takes two coloured graphs G and H and returns a pair from a colour-preserving isomorphism of G and H; if no colour-preserving isomorphism exists, 1-colour-GI returns an arbitrary pair in $V(G) \times V(H)$.*

Lemma 1. *1-colour-GI is polynomial-time reducible to 1-GI using one call to 1-GI.*

Proof. Let G and H be the graphs on which 1-colour-GI is called, where $|V(G)| = |V(H)| = n$ (if the number of vertices in G and H is different,

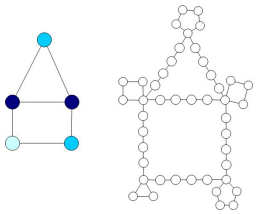

FIGURE 1. An illustration of the conversion from G to G' in the reduction from 1-colour-GI to 1-GI.

then 1-colour-GI returns an arbitrary pair). A graph on n vertices uses at most n colours. Number the used colours from 3 to $k+2$, where the total number of colours used is k (note that $k \leq n$). Create graphs G' from G and H' from H as follows.

- For each vertex v in $G \cup H$ of colour i, extend the vertex to an i-cycle. Designate a unique vertex on the cycle and make all edges that were incident on v incident on this vertex. For every vertex constructed, record in a table the old vertex it was created from.
- Let $j = k + 3$. If j is odd, increase j by 1. For each edge (x, y) in G and H, convert the edge to a path with j vertices. For every new vertex on the path, record in the table the endpoint, x or y, that the new vertex is closer to (this vertex exists since j is even).

Note that cycles of length at most $k + 2$ in G' and H' are in 1-1 correspondence with the vertices of G and H, respectively, as no vertices created from edges of G and H (except the endpoints) can be part of cycles on less than $k + 3$ vertices. Note that the construction is done in polynomial-time.

Run 1-GI on G' and H'. 1-G1 returns a pair $\{x', y'\}$. Look up the old vertices, x and y, that x' and y' are associated with, respectively. Then (x, y) is an isomorphism pair of G and H. □

We now present a simple polynomial-time reduction from GI to 1-colour-GI. Since 1-colour-GI is polynomial-time reducible to 1-GI, this yields a polynomial-time reduction to 1-GI.

Lemma 2. *GI is polynomial-time reducible to 1-colour-GI.*

Proof. At any point in the algorithm, if the algorithm recognizes that a pair of vertices returned by 1-colour-GI is not a colour isomorphism pair, then G and H are not isomorphic. Let $\mathrm{N}(x)$ denote the set of vertices adjacent to a vertex x.

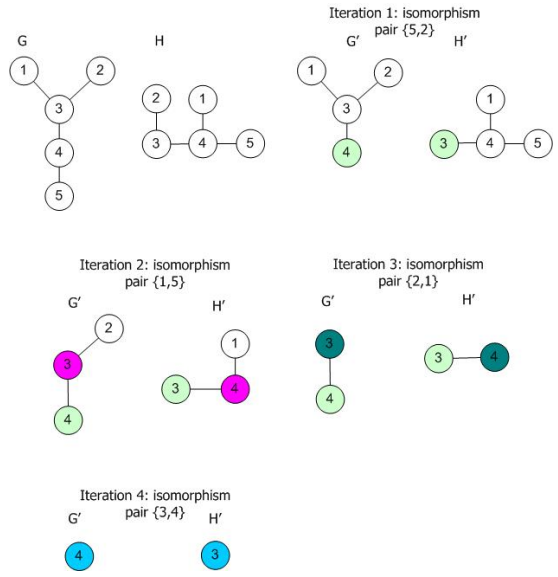

FIGURE 2. An example illustrating the reduction from GI to 1-colour-GI. At each step the isomorphism pair returned by 1-colour-GI is displayed.

- Let $G' := G$. Let $H' := H$. Colour in white all the vertices of G' and H'.
- Let $\phi = \emptyset$ be a partial isomorphism from G to H.
- Repeat the following sequence of operations n times.
  - Call 1-colour-GI(G', H') getting (x, y).
  - Let $\phi := \phi \cup (x, y)$.
  - Let $r = deg(x) = deg(y)$ (else G' and H' are not isomorphic).
  - If G and H are isomorphic then x and y should have the same number of neighbours coloured i for each colour i. Let S be the set of colours appearing as colours of neighbours of x and y in G' and H'. Note that $|S| \leq r$. Select a set, T, of $|S|$ new colours that have not yet been used by the algorithm and define a bijection, $\rho : S \to T$.
  - For each vertex $u \in N(x) \cup N(y)$ with colour c, colour u with $\rho(c)$.
  - Remove x from G' and y from H' and the edges adjacent to them.
- Check whether ϕ is an isomorphism from G to H. If so, return ϕ. Otherwise, return that G and H are not isomorphic.

There are n iterations. At each iteration the colours of neighbours of some vertices $x \in V(G)$ and $y \in V(H)$ are updated. Since the degree of each vertex is bounded by n, each iteration has $O(n)$ operations. Therefore, the runtime of the reduction is $O(n^2)$. □

We now present a reduction from GI to 1-GI, based on the composition of the reduction from GI to 1-colour-GI and the reduction from 1-colour-GI to 1-GI, with a few modifications that make the reduction run in cubic time.

Theorem 1. *GI is reducible to 1-GI in time $O(n^3)$.*

Proof. To reduce GI to 1-GI, perform the reduction from GI to 1-colour-GI with the following modifications:

- Let G^* and H^* be graphs obtained from G and H, respectively, using the conversion described in the reduction from 1-colour-GI to 1-GI, except that edges are converted to paths of length $n+3$ (or $n+4$ if n is even). As described in the reduction from 1-colour-GI to 1-GI, create a lookup table from vertices in $G^* \cup H^*$ to vertices in $G \cup H$.
- Instead of calling 1-colour-GI on G' and H' (as in Lemma 2), update G^* and H^* so that G^* and H^* are the graphs we would get using the conversion from 1-colour-GI on G' and H' to 1-GI, except that edges are converted to paths of length $n+3$ (or $n+4$ if n is even). To do so, remove all cycles in G^* and H^* corresponding to vertices of G and H and place new ones in the way described in Lemma 1. Then call 1-GI on G^* and H^* getting a pair of vertices (x', y'). Look up in the table the vertices x and y associated with x' and y' respectively and continue the algorithm as if the colour-isomorphism pair (x, y) was returned by 1-colour-GI.

In the construction of G^* and H^*, each vertex in G and H is extended to a cycle on at most $n+2$ vertices and each edge is subdivided to a path of length at most $n+4$. Therefore constructing G^* and H^* takes time $O(n^3)$.

Consider the runtime of the modifications of G^* and H^*. The only operation outside 1-GI that is done with paths on G^* and H^* created from edges of G and H is their removal. Therefore their total contribution to the running time over all iterations is $O(n^3)$. At every iteration, old cycles corresponding to vertices in G and H are replaced by new ones. Since the maximum size of such a cycle is $n+2$ and there are at most $2n$ cycles, the cycle replacement takes $O(n^2)$ steps per iteration. When a vertex corresponding to a cycle is removed from G and H,

the cycle is removed. This adds $O(n)$ work to each iteration. Therefore, aside from removal of vertices corresponding to edges of G and H, each iteration takes time $O(n^2)$. Thus, the runtime of the algorithm is $O(n^3 + n^3 + n(n^2)) = O(n^3)$. □

The constant in the running time of the algorithm can be improved by finding an isomorphism between the complement of G and the complement of H whenever the number of edges in G and H reaches a certain threshold.

3. Subgraph Isomorphism

The colour reduction for GI can be modified and applied to the subgraph isomorphism problem. The subgraph isomorphism search problem is the following: given graphs G and H, find a subgraph of H that is isomorphic to G or determine that no such subgraph exists.

Definition 2. *A* subgraph-isomorphism pair *is a pair of vertices* $(x, y) \in V(G) \times V(H)$ *such that* y *is a vertex of some subgraph* H^* *of* H *that is isomorphic to* G *and* (x, y) *is an isomorphism pair of some isomorphism from* G *to* H^**.*

The function 1-SI returns the minimal (non-empty) partial subgraph-isomorphism (if one exists). That is, 1-SI is a function that takes graphs G and H, where if G is isomorphic to a subgraph of H, 1-SI returns a subgraph-isomorphism pair $(x, y) \in V(G) \times V(H)$; otherwise, 1-SI returns an arbitrary pair of vertices in $V(G) \times V(H)$.

We explain how to reduce SI to 1-SI using a reduction analogous to the colour reduction from GI to 1-GI. Instead of assigning unique colours to vertices, each vertex is assigned a subset of colours from $\{1, 2, ..., n\}$, where $n = |V(G)|$. A graph coloured using this scheme is called a *list-coloured* graph. Let $colours(v)$ denote the set of colours assigned to a vertex v.

Definition 3. *Given list-coloured graphs* G *and* H*, let a* list-colour subgraph-isomorphism pair *be a pair* $(x, y) \in V(G) \times V(H)$ *such that* $colours(x) \subseteq colours(y)$ *and* (x, y) *is an isomorphism pair of* G *and some subgraph of* H*.*

Definition 4. *Given list-coloured graphs* G *and* H*,* 1-colour-SI *returns a list-colour subgraph-isomorphism pair. If no such pair exists, it returns an arbitrary pair in* $V(G) \times V(H)$*.*

Lemma 3. *1-colour-SI is polynomial-time reducible to 1-SI.*

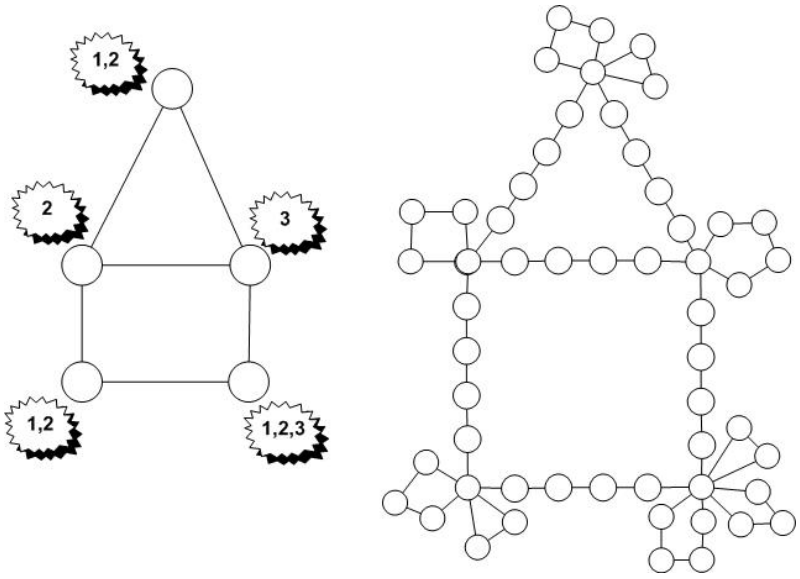

FIGURE 3. An illustration of the conversion of G to G^* in the reduction from 1-colour-SI to 1-SI.

Proof. Let $G^* = G$, $H^* = H$. Build a table that represents a surjection $\psi : V(G^*) \cup V(H^*) \mapsto V(G) \cup V(H)$. Initially the table is empty. Recall that the lists of colours are represented by subsets of $\{1, 2, ..., n\}$. For each vertex $u \in G^* \cup H^*$ do the following: if $colours(u) \neq \emptyset$, for each colour k in $colours(u)$ attach a cycle of length $k + 3$ to u, otherwise, attach a cycle of size 3 to u. For each vertex v on a cycle attached to u, set $\psi(v) = u$ and store this in the table.

Let m be the maximal value of a colour that occurs in a list in G^* or H^*. Let $j = m + 4$ if m is even, and $j = m + 5$ if m is odd. Subdivide each edge in G^* and in H^* into a path with j vertices. For each vertex v on a path created through subdivision of an edge of G^* or H^*, associate v with the closest endpoint of the path. More precisely, if the closest endpoint to v is u, then $\psi(v) = u$ is stored in the table.

Call 1-SI(G^*, H^*) getting the pair (u, v). Look up the vertex associated with u and the vertex associated with v. Say $\psi(u) = x$ and $\psi(v) = y$. Then (x, y) is returned by 1-SI.

The pair (x, y) is a valid list-colour subgraph-isomorphism pair because cycles attached to the vertices in G^* and H^* preserve the lists of colours. Edges are subdivided so that the cycles attached to vertices to represent the colour lists are too small to occur in any other way than through the cycle attachment procedure. Thus G^* and H^* preserve both the structure of G and H and the lists of colours.

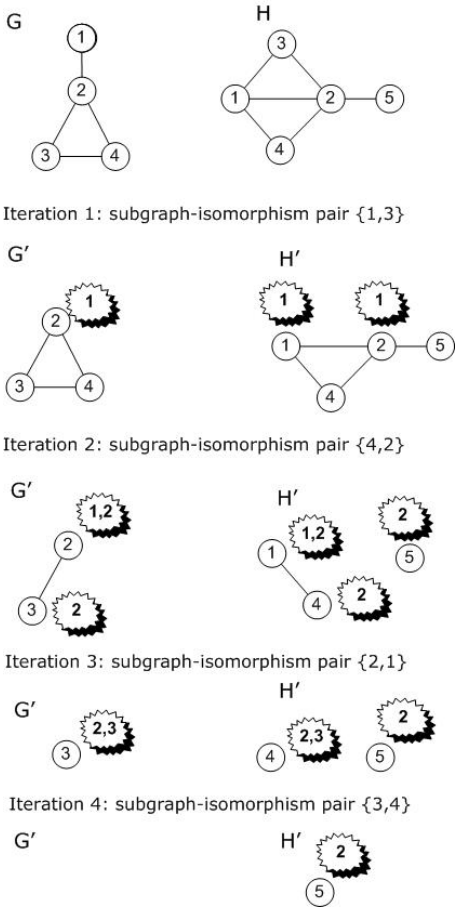

FIGURE 4. An example illustrating the reduction from SI to 1-colour-SI. At each step the list-colour subgraph-isomorphism pair returned by 1-colour-SI is displayed.

The lists of colours are of size at most n and contain colours from 1 to n. Therefore, each attached cycle is of size at most $n+3$ and there are at most n cycles that need to be attached to each vertex in G and H. Each edge is subdivided into a path of size at most $n+5$. Therefore the reduction is polynomial in the size of the input. □

Lemma 4. *SI is polynomial-time reducible to 1-colour-SI.*

Proof. The following is an algorithm that reduces SI to 1-colour-SI.

- Given graphs G and H construct G' by associating each vertex in G with an empty list of colours. Similarly, construct H' from H by associating each vertex in H with an empty list of colours.
- Let colourNumber = 1.

- Let $\phi = \emptyset$.
- Repeat the following iteration n times.
  - Call 1-colour-SI(G',H') getting a pair of vertices (x, y). Set $\phi = \phi \cup (x, y)$.
  - Add colourNumber to the list of colours of each vertex in $N(x) \cup N(y)$.
  - Remove x and all edges adjacent to x from G'. Similarly, remove y and all edges adjacent to y from H'.
  - Let colourNumber = colourNumber+1.
- Check if ϕ is an isomorphism between G and a subgraph of H. If it is, then return ϕ. Otherwise, G is not isomorphic to a subgraph of H.

The number of iterations is n. At each iteration, a colour is added to at most $2(n-1)$ lists of colours, x is removed from G and y is removed from H. Therefore, the runtime of the reduction is $O(n^2)$. □

The composition of the reduction from SI to 1-colour-SI with the reduction from SI to 1-SI yields a polynomial-time reduction from SI to 1-SI. We show how to compose the reductions to yield a more efficient reduction from SI to 1-SI.

Theorem 2. *SI is reducible to 1-SI in time $O(en + m + n^3)$, where $e = |E(H)|$, $m = |V(H)|$, and $n = |V(G)|$.*

Proof. To reduce SI to 1-SI, perform the reduction from SI to 1-colour-SI with the following modifications;

- Let G^* and H^* be graphs obtained from G and H respectively using the conversion described in the reduction from 1-colour-SI to 1-SI, except that edges are converted to paths of length $n + 4$ (or $n + 5$ if n is odd). As described in the reduction from 1-colour-SI to 1-SI, create a lookup table from vertices in $G^* \cup H^*$ to vertices in $G \cup H$.
- Instead of calling 1-colour-SI on G' and H' (as in Lemma 4), update G^* and H^* so that G^* and H^* are the graphs that we would get using the conversion from 1-colour-SI on G' and H' to 1-SI except that edges are converted to paths of length $n+4$ (or $n + 5$ if n is odd). To do so, remove all cycles of length $\leq n+3$ attached to vertices and replace them by new cycles as in the reduction from 1-colour-SI to 1-SI. Since a single cycle of length 3 indicates an empty colour list, do not remove a cycle of length 3 when it is the only cycle attached to a vertex unless the corresponding vertex in G' or H' has been assigned a colour in the given iteration. Run 1-SI getting a pair of vertices

(x', y'). Look up the vertices x and y associated with x' and y' respectively in the lookup table and continue the reduction as in the reduction from SI to 1-colour-SI using the pair (x, y) as if it were returned by 1-colour-SI.

Let $e = |E(H)|$, $f = |E(G)|$, $m = |V(H)|$ and $n = |V(G)|$. The construction of G^* and H^* takes $O(en+m)$, en to convert the edges to paths and m to convert the vertices to 3-cycles (note that $m \geq n$ and $e \geq f$, unless G is not isomorphic to a subgraph of H). The removal of paths corresponding to edges of G or H takes at most $O(en)$ time in total. Since up to n vertices in G and in H are attached up to n cycles of length between 4 and $n+3$, and a cycle of length 3 is not removed when it is the only cycle attached to a vertex unless its associated vertex in G' or H' has been assigned a colour in the given iteration, the addition of cycles in G^* and H^* takes $O(n^3)$. The assignment of colours and removal of x from G and y from H takes $O(n)$ time per iteration. The corresponding cycle removal in G^* and H^* is $O(n^2)$ per iteration. Therefore the total running time of the reduction is $O(en+m+n^3)$. □

Since SI is NP-complete, Theorem 2 implies that 1-SI is also NP-complete.

Corollary 1. *1-SI is NP-complete.*

Proof. Since SI is NP-complete and, by Theorem 2, SI is polynomial-time reducible to 1-SI, 1-SI is NP-hard. By providing an isomorphism from G to a subgraph of H, we can verify in polynomial-time that a given pair of vertices is a valid subgraph-isomorphism pair. □

4. Comparison of the reductions

The main idea in the clique-padding reduction, the colour reduction, and the list-colour reduction is that all necessary information about the removed vertices is preserved. It is necessary that a (subgraph-) isomorphism pair found in step i is compatible with the partial (subgraph-) isomorphism found up to step i. In order to do that, it is necessary and sufficient to know which removed vertices were adjacent to which remaining vertices. In the earlier clique-padding reduction, this information is retained by attaching large cliques. A clique of a unique size is associated with each isomorphism pair (x, y) and attached to all vertices adjacent to x or y. Then a pair (u, v) of old vertices can be an isomorphism pair only if u and v are adjacent to isomorphic cliques of new vertices. The cliques of new vertices need to be sufficiently large so that they are different from all cliques that could potentially be subgraphs of G and H.

The advantage of the colour reduction for graph isomorphism over the clique-padding reduction is that it stores less information with each vertex. This reduces the complexity of the reduction. The colour-reduction has runtime of $O(n^3)$ while the runtime of the clique-padding reduction is $O(n^4)$.

Note that the clique-padding reduction from GI to 1-GI does not generalize to a reduction from SI to 1-SI since a clique of size k is a subgraph of any clique of size greater than k.

5. Reductions with a restricted number of calls

The clique-padding reduction as well as the colour reduction call 1-GI n times. The function 1-GI can be viewed as an oracle. A referee in [2] observed that in fault-tolerant computing where one tries to recover a solution to a hard problem, parts of which have been lost through transmission, a single call to an oracle is more realistic. This observation presents the question of whether such a reduction exists. The following simple algorithm shows that such a reduction would imply that GI is in P.

Theorem 3. *If there exists a polynomial-time reduction from GI to 1-GI that makes only one call to 1-GI, then GI is in P.*

Proof. Let A be such a reduction. Let the graphs on which 1-GI is called be G' and H'. Consider the following algorithm;

Run A **until the call to 1-GI on graphs** G' **and** H'
Fix a vertex u **in** G'
For all vertices v **in** H'
    Assume that 1-GI returned $\{u, v\}$
    Run the rest of A
    Check if the result is an isomorphism
    If it is an isomorphism, return it and terminate
If this point is reached, G **and** H **are not isomorphic**

Since G' and H' are constructed in A, constructing them takes polynomial time. Since $P \subseteq PSPACE$, the size of G' and H' is bounded by some polynomial function $f(n)$. Therefore the loop runs $O(f(n))$ times. Running the remainder of A is done in polynomial-time as is checking if the result is an isomorphism. Therefore the above is a polynomial-time algorithm for GI. □

By modifying the argument above, we can get the following.

Theorem 4. *If GI is polynomial-time reducible to k-GI, for some constant k, where the reduction makes a constant number of parallel calls to k-GI, then GI is in P.*

Proof. Assume such a reduction, A, exists. k-GI is called on $(G_1, H_1), (G_2, H_2), ..., (G_c, H_c)$ in parallel. Then the following algorithm solves GI in polynomial time;

Run A until the c parallel calls to k-GI
Fix any sequence of distinct vertices $(u_1^j, u_2^j, ..., u_k^j)$ in G_j for each j
For all sequences $(v_1^i, v_2^i, ..., v_k^i) \subseteq V(H_i)$ over all i
    Assume k-GI gives:
        $\{[(u_1^1, v_1^1), (u_2^1, v_2^1), ..., (u_k^1, v_k^1)], ..., [(u_1^c, v_1^c), (u_2^c, v_2^c), ..., (u_k^c, v_k^c)]\}$
    Run the rest of A
    Check if the result is an isomorphism
    If it is an isomorphism, return it and terminate
If this point is reached, G and H are not isomorphic

□

6. Acknowledgements

I would also like to thank Jonathan Buss, Eugene Eisenstein, and Matei Zaharia for useful discussion. I would also like to thank an anonymous referee for helpful suggestions.

References

[1] A. Gál, S. Halevi, R. Lipton, and E. Petrank. "Computing from partial solutions." *Proceedings of the 14th Annual IEEE Conference on Computational Complexity*, 34-45. IEEE Computer Society Press, 1999.

[2] A. Große, J. Rothe and, G. Wechsung. "Computing complete graph isomorphisms and hamiltonian cycles from partial ones." *Theory of Computing Systems*, 35(1):81-93, 2002.