On the Hardness of Graph Isomorphism

Jacobo Torán
Abteilung Theoretische Informatik
Universität Ulm
Oberer Eselsberg
89069 Ulm, Germany
toran@informatik.uni-ulm.de

Abstract

We show that the graph isomorphism problem is hard under logarithmic space many-one reductions for the complexity classes NL, PL (probabilistic logarithmic space), for every logarithmic space modular class Mod_kL and for the class DET of problems NC^1 reducible to the determinant. These are the strongest existing hardness results for the graph isomorphism problem, and imply a randomized logarithmic space reduction from the perfect matching problem to graph isomorphism.

1. Introduction

The graph isomorphism problem GI consists in deciding whether two given graphs are isomorphic, or in other words, whether there is a bijection between the nodes of the graphs preserving the edges. This problem has been intensively studied, in part because its many applications, an in part because it is one of the few problems in NP that has resisted all attempts to be classified as NP-complete, or within P. The best existing upper bound for the problem given by Luks and Zemlyachenko is $\exp \sqrt{cn \log n}$ (cf [7]), but there is no evidence of this bound being optimal, and for many restricted graph classes polynomial time algorithms are known. This is for example the case of planar graphs [14], graphs of bounded degree [23] or graphs with bounded eigenvalue multiplicity [6]. In some cases, like trees [22, 10] or graphs with colored vertices and bounded color classes [24], even NC algorithms for isomorphism are known.

Concerning the hardness of GI, there is evidence indicating that the problem is not NP-complete. On the one hand, the counting version of GI is known to be reducible to its decisional version [25], while for all known NP-complete problems the counting version seems to be much harder. On the other hand it has been shown that if GI were NP-complete then the polynomial time hierarchy would collapse to its second level [8, 30]. Because of these facts, there is a common believe that GI does not contain enough structure or redundancy to be hard for NP. The question of whether GI is P-hard is also open, and moreover, the known lower bounds in terms of hardness results for GI are surprisingly weak. It is only known that isomorphism for trees is hard for NC^1 and for L (logarithmic space) depending on the encoding of the input [17].

In this paper we improve the existing hardness results by showing that GI is hard for all complexity classes defined in terms of the number of accepting computations of a nondeterministic logarithmic space machine.

The key ingredient in the proof of our results, is a graph gadget showing that GI has enough structure to encode a modular addition gate. Using this fact, we are able to give for any ($k \in \mathbf{N}$) a logarithmic space many-one reduction from the circuit value problem for addition mod k gates, which is complete for Mod_kL, to GI. Mod_kL is the complexity class corresponding to sets recognized by nondeterministic logarithmic space machines in which the number of accepting computations satisfies a congruence modulo k [9], and it lies within NC^2. We show that a circuit with modular gates can be directly transformed into a graph in which any automorphism of a certain kind maps a special vertex encoding the output gate to a vertex encoding the output of the circuit. The graphs used in the reduction have degree 3 and its vertices can be partitioned (in logarithmic space) into color-classes of size k^2. Luks [24] has given an NC upper bound for the complexity of the isomorphism problem restricted to graphs with bounded color classes. For isomorphism in this class of graphs, the gap between our hardness results and the upper bound given by Luks is therefore small.

By a simple use of the Chinese Reminder Theorem, the hardness results for the modular classes can be transformed into logarithmic space hardness results for NL, and $C_=L$.

0-7695-0850-2/00 $10.00 © 2000 IEEE

It is interesting to observe that the graphs obtained in these reductions have automorphism groups in which the size of the orbits of some of the nodes depend on the input size, and therefore they do not have classes of colored vertices of constant size as in the modular case.

Using the recent surprising result that division can be performed in NC^1 [12], and the fact that an NC^1 circuit can be encoded in an isomorphism problem [17], we can moreover prove that any logarithmic space counting function can be reduced to GI. In particular this implies that GI is many-one hard for probabilistic logarithmic space, PL, and for DET, defined by Cook [11] as the class of problems NC^1 Turing reducible to the determinant.

The perfect matching problem is (as GI) another problem of the few that has resisted classification in terms of completeness. It was shown in [5] that perfect matching can be randomly (or non-uniformly) reducible to Mod_kL for every k. From our results this implies a (random or non-uniform) logarithmic space reduction from matching to GI, which provides the first reduction between the two well studied problems. Moreover, as a consequence of derandomization results from [16, 3, 19], under the natural hypothesis that there is a set in DSPACE(n) with circuit hardness $2^{\delta n}$ for some $\delta > 0$, our reduction implies a many-one logarithmic space (deterministic) reduction from perfect matching to GI.

2. Preliminaries

We assume familiarity with basic notions of complexity theory such as can be found in the standard textbooks in the area. We will prove hardness results for several logarithmic space complexity classes: NL is the class of languages accepted by nondeterministic Turing machines using logarithmic space. The graph accessibility problem GAP (given a directed graph with two designated nodes s and t decide whether there is a path from s to t) is known to be complete for NL, even in the case of acyclic graphs with in-degree at most 2. The accessibility problem for undirected graphs, UGAP, is complete for the class SL, symmetric logarithmic space [21]. This class has different characterizations, but the easiest way to define it is precisely as the class of problems logarithmic space reducible to UGAP.

#L defined by [4] analogously to Valiant's class #P, is the class of functions $f : \Sigma^* \to \mathbb{N}$ that count the number of accepting paths of a nondeterministic Turing machine M on input x. The computation of a #L function on an input x can be reduced to compute the number paths from node s to node t in a directed graph G_x. The complexity classes PL (probabilistic logarithmic space), $\text{C}_=\text{L}$ (exact threshold in logarithmic space), and Mod_kL (modular counting in logarithmic space, $k \geq 2$) can be defined in terms of #L functions:

$$\text{PL} = \{\ A : \exists p \in \text{Poly}, f \in \#\text{L}\ \ x \in A \Leftrightarrow f(x) \geq 2^{p(|x|)}\}\ [13, 29]$$

$$\text{C}_=\text{L} = \{\ A : \exists p \in \text{Poly}, f \in \#\text{L}\ \ x \in A \Leftrightarrow f(x) = 2^{p(|x|)}\}\ [2]$$

$$\text{Mod}_k\text{L} = \{\ A : \exists f \in \#\text{L}\ \ x \in A \Leftrightarrow f(x) = 1 \bmod k\}\ [9]$$

Mod_k circuits ($k \geq 2$), are circuits where the input variables (and the wires) can take values in $\mathbb{Z}_k$, and the gates compute addition in $\mathbb{Z}_k$. The evaluation problem for such circuits (given fixed values for the inputs, decide whether the output value is for example 1) is complete for Mod_kL under logarithmic space many-one reductions. This is because a directed acyclic graph with in-degree at most two, and two designated nodes, s, t, can be easily transformed into a mod_k circuit computing the residue of the number of paths from s to t in G, modulo k.

In some of the proofs we will make use of NC^1 circuits. These are families of logarithmic depth, polynomial size Boolean circuits of bounded fan-in over the basis $\{\wedge, \vee, \neg\}$. We will explicitly mention uniformity conditions for this kind of circuits when needed. DET [11] is the class of problems NC^1 Turing reducible to the determinant, or in other words, the class of problems that can be solved by NC^1 circuits with additional oracle gates that can compute the determinant of integer matrices.

The known relationships among the considered classes are:

$$\text{SL} \subseteq \text{Mod}_k\text{L} \subseteq \text{DET},$$

$$\text{SL} \subseteq \text{NL} \subseteq \text{C}_=\text{L} \subseteq \text{PL} \subseteq \text{DET}.$$

Looking at the known inclusions, the hardness of GI for DET implies hardness with respect to the other classes. We prove however the result for all the classes separately showing how the graphs produced by the reductions increase in complexity with the different classes.

2.1. Reducibilities

For simplicity we will prove our hardness results for logarithmic space many-one reducibility, but in fact most the results hold for stronger reducibilities, like for example DLOGTIME uniform NC^1 many-one reducibility. This is true with the exception of Theorem 5.5 and Corollaries 5.6 and 5.7. In this cases the proofs make use of the recent NC^1 algorithm for reconstructing a number from its Chinese reminder representation [12], which is only known to

be logarithmic space uniform [1]. Because of this, the (uniform) hardness results in the mentioned cases only hold for logarithmic space reducibilities.

2.2. Graph Isomorphism

An automorphism in an undirected graph $G = (V, E)$ is a permutation φ of the nodes, that preserves adjacency. That is, for every $u, v \in V, (u, v) \in E \Leftrightarrow (\varphi(u), \varphi(v)) \in E$. An isomorphism between two graphs G, H is a bijection between their sets of vertices which preserves the edges. GI is the problem

$$\text{GI} = \{(G, H) \mid G \text{ and } H \text{ are isomorphic graphs}\}$$

Sometimes we will deal with graphs with colored vertices. A coloring with k colors is a function $f : V \to \{1, \ldots k\}$. In an isomorphism between colored graphs, the colors have to be preserved. The isomorphism problem for colored graphs can be easily reduced (by a logarithmic space many-one reductions) to graphs isomorphism without colors (see e.g. [20]).

Sometimes we will consider the following restricted automorphism problem: Given a graph $G = (V, E)$ and two lists of nodes $(x_1, \ldots, x_k), (y_1, \ldots, y_k)$, is there an automorphism in G mapping x_i to y_i for $1 \leq i \leq k$? This problem is also easily (logarithmic space) reducible to GI. In order to check whether there is an automorphism with the desired properties one can make two copies of G, G' and G''. In G' each of the nodes x_i has color i and in G'' node y_i receives this color. All the other nodes are colored with a new color 0, for example. G' and G'' are isomorphic if and only in G has an automorphism with the mentioned properties.

3. Hardness for SL

We start by showing that GI is hard for symmetric logarithmic space SL. Although it is known that this class is included in all the classes that we will consider later, we present the reductions for two reasons. First it is the simpler reduction and helps to understand the others, and secondly, the graphs obtained are the simplest ones since the automorphism orbits of all the nodes have size at most two.

Theorem 3.1 *GI is hard for SL under logarithmic space many-one reductions.*

Proof. We prove the the graph accessibility problem for undirected graphs, UGAP, is reducible to the complement of GI. The result follows since UGAP is logarithmic space complete for SL, and this class is closed under complementation [28].

Let $G = (V, E)$ be an undirected graph with two designated nodes $s, t \in V$. Consider the new graph $G' = G_1 \cup G_2$ where G_1 and G_2 are two copies of G, and for a node $v \in V$ let us call v_1 and v_2 the copies of v in G_1 and G_2 respectively. Furthermore, G' is defined to have node t_1 labeled with color 1, (the rest of the nodes have color 0). We claim that there are not any paths from s to t in G if and only if there is automorphism φ in G' mapping s_1 to s_2. Clearly if there are not any paths between s and t in G, these two nodes belong to different connected components. The desired automorphism can be obtained by mapping the nodes of the connected component of s_1 in G_1 to the corresponding nodes in G_2 and mapping the rest of the nodes in G' (and in particular t_1) to themselves.

Conversely, if there is a path between s and t in G, the mentioned automorphism φ does not exist since the nodes s_2 $(\varphi(s_1))$ and t_1 $(\varphi(t_1))$ should be in the same connected component, but there are no edges between G_1 and G_2 in G'.

The question of whether there is an automorphism in G' with the mentiioned properties, can in turn be reduced to GI as mentioned in the preliminaries.

■

4. Hardness for the modular counting classes

We show now that GI is hard for for all the logarithmic space modular counting classes Mod_kL $(k \geq 2)$. The idea for this proof is to simulate a modular gate with a graph gadget and then combine the gadgets for the different gates into a graph, whose automorphisms simulate the behavior of the modular circuit.

The gadgets are defined by the following graphs (shown in Figure 1 for the case $k = 2$):

Definition 4.1 *Let $k \geq 2$ and denote by $\oplus$ the addition in $\mathbb{Z}_k$. We define the undirected graph $G^k = (V, E)$, given by the set of $k^2 + 3k$ nodes*

$$\begin{aligned} V &= \{x_a, y_a, z_a \mid a \in \{0, \ldots, k-1\} \cup \\ &\quad \{u_{a,b} \mid a, b \in \{0, \ldots, k-1, \}\} \end{aligned}$$

and edges

$$\begin{aligned} E &= \{(x_a, u_{a,b}) \mid a, b \in \{0, \ldots, k-1\}\} \cup \\ &\quad \{(y_b, u_{a,b}) \mid a, b \in \{0, \ldots, k-1\}\} \cup \\ &\quad \{(u_{a,b}, z_{a \oplus b}) \mid a, b \in \{0, \ldots, k-1\}\}. \end{aligned}$$

The graph gadget for a modular gate has nodes encoding the inputs and outputs of the gate. Any automorphism in the graph mapping the input nodes in a certain way, must map the output nodes according to the value of the modular gate being simulated.

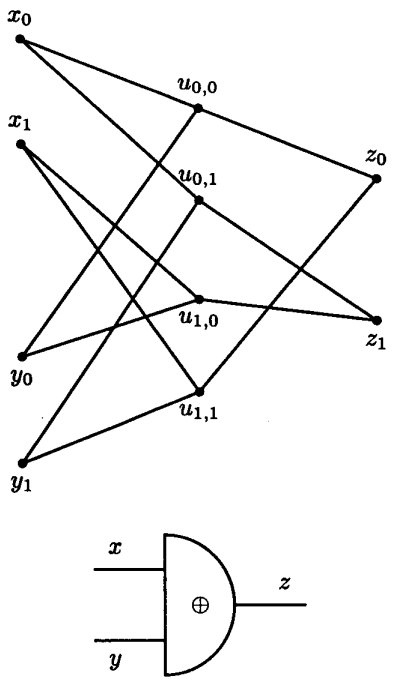

Figure 1. The graph G^2 simulating a parity gate.

Lemma 4.2 *Fix $k \geq 2$, for any $a, b \in \{0, \ldots, k-1\}$,*

1) *there is an automorphism φ in G^k mapping x_0 to x_a and y_0 to y_b, and*

2) *every automorphism φ in G^k mapping x_0 to x_a and y_0 to y_b, maps z_0 to $z_{a \oplus b}$.*

Proof. Let $a, b \in \{0, \ldots, k-1\}$, and denote by $\oplus$ the addition in $\mathbb{Z}_k$. We consider the following function $\varphi : V \to V$ defined as

$$\begin{aligned} \varphi(x_i) &= x_{a \oplus i} \text{ for } i \in 0, \ldots, k-1, \\ \varphi(y_i) &= x_{b \oplus i} \text{ for } i \in 0, \ldots, k-1, \\ \varphi(u_{i,j}) &= u_{a \oplus i, b \oplus j} \text{ for } i, j \in 0, \ldots, k-1, \\ \varphi(z_i) &= z_{a \oplus b \oplus i} \text{ for } i \in 0, \ldots, k-1. \end{aligned}$$

We prove first that φ is an automorphism. For this we have to show that for every pair of nodes v, w, $(v, w) \in E$ if and only if $(\varphi(v), \varphi(w)) \in E$. The nodes in graph G^k can be partitioned in three layers, the x and y nodes, (input layer) the u nodes and the z nodes (output layer). Edges exist only between nodes from first and second layers, or between nodes from second and third layers. We consider first an edge between the first two layers. Let $v = x_i$ and $w = u_{l,m}$ with $i, l, m \in \{0, \ldots, k-1\}$. Then $\varphi(v) = x_{a \oplus i}$ and $\varphi(w) = u_{a \oplus l, b \oplus m}$. By the definition of G^k,

$$\begin{aligned} (x_i, u_{l,m}) \in E &\Leftrightarrow i = l \\ &\Leftrightarrow a \oplus i = a \oplus l \\ &\Leftrightarrow (\varphi(x_i), \varphi(u_{l,m})) \in E. \end{aligned}$$

In the case $v = y_j$ the proof is analogous. For an edge (v, w) between the second and third layers, let $(v, w) = (u_{i,j}, z_l)$ with $i, j, l \in \{0, \ldots, k-1\}$. Then $\varphi(v) = u_{a \oplus i, b \oplus j}$ and $\varphi(w) = z_{a \oplus b \oplus l}$. By the definition of G^k,

$$\begin{aligned} (u_{i,j}, z_l) \in E &\Leftrightarrow l = i \oplus j \\ &\Leftrightarrow a \oplus b \oplus l = a \oplus b \oplus i \oplus j \\ &\Leftrightarrow (\varphi(u_{i,j}), \varphi(z_l)) \in E. \end{aligned}$$

For the proof of 2), observe that for any automorphism ϕ with the restriction $\phi(x_0) = x_a$ and $\phi(y_0) = y_b$ it holds $\phi(u_{0,0}) = u_{a,b}$ since this is the only node in the second layer connected to both nodes $\phi(x_0)$, $\phi(y_0)$. This implies $\phi(z_0) = z_{a \oplus b}$ since $\phi(z_0)$ must be connected to $\varphi(u_{0,0})$. ■

Theorem 4.3 *For any $k \geq 2$, GI is hard for Mod_kL under logarithmic space many-one reductions.*

Proof. Let $k \geq 2$. We reduce the mod_k circuit value problem to GI. We transform an instance C of the circuit value problem for mod_k circuits into a graph G_C by constructing for every modular gate g_j of C a subgraph like the one described in Lemma 4.2. Moreover, we color the x, y, u and z nodes of the j-th gadget respectively with one of the colors $(x, j), (y, j), (u, j)$ and (z, j). Connections between gates are translated in the following way: If the output z of a gate in the circuit is connected to one of the inputs x of another gate, the reduction puts k additional edges connecting (for $i \in \{0, \ldots, k-1\}$) node z_i from the first gate to node x_i from the second gate. For an input variable v^j, k nodes $v_0^j, \ldots, v_{k-1}^j$ are considered in the reduction. The coloring implies that in any automorphism the nodes corresponding to a gate are mapped to nodes from the same gate. If the input variables of the circuit, $v^1, \ldots, v^n$ take values $a_1, \ldots, a_n$, it follows from Lemma 4.2, by induction on the circuit depth, that the output gate z takes value $b \in \{0, \ldots, k-1\}$ if an only if there in an automorphism in G_C mapping v_0^i to $v_{a_i}^i$ for $i = 1, \ldots, n$, and mapping z_0 to z_b.

All the steps in the reduction can be done locally in logarithmic space. The value of the circuit is 1 if an only if there is an automorphism in G_C with the restriction that some nodes w_j have to be mapped to some specific nodes

w'_j. This question can be easily reduced to GI as explained in the preliminaries. ■

Observe that the graphs obtained in the reduction, have at most k^2 nodes with the same color (the nodes $u_{i,j}$ in any of the gate gadgets). The maximum degree can be reduced to three. In the above description this does not necessarily hold because of the connection between gates. However, the reduction can be easily modified to achieve degree 3 by adding some extra nodes and arranging the fan-out connections of the gates in a tree-like fashion.

5. Hardness for other complexity classes

In this section we show the hardness of GI for nondeterministic logarithmic space, for C$_=$L, and for probabilistic logarithmic space. For the first two cases the proof follows by the modular results, using the Chinese Reminder Theorem (CRT).

A Chinese reminder representation base is a set $m_1, \ldots, m_n$ of pairwise coprime integers. Let $M = \prod_{i=1}^{n} m_n$. By the CRT, every integer $0 \leq x \leq M$ is uniquely represented by its Chinese reminder representation $(x_1, \ldots, x_n)$, where $0 \leq x_i < m_i$ and $x_i = x \bmod m_i$. We will consider the base B_n formed by the first n prime numbers.

Theorem 5.1 *GI is hard for NL under logarithm space many-one reductions.*

Proof. The graph accessibility problem for acyclic directed graphs with fan-in at most 2 is complete for the class NL. We reduce the complementary of this set (non-reachability) to GI. The result follows by the closure of NL under complementation [15, 31]. Let $G = (V, E)$ be such a graph, with $|V| = n$ and with two designated nodes s and t. Let P the number of paths from s to t in G, clearly $P \leq 2^n$. Let $p_1, \ldots, p_m$ be the m smallest prime numbers, and let m be the smallest integer such that $\prod_{i=1}^{m} p_i > 2^n$. By the CRT $P = 0$ if and only if $P \bmod p_1 = \ldots = P \bmod p_m = 0$. In order to check whether $P = 0$ we just have to check that the residues of P modulo the first primes is 0. This can be done composing two logarithmic space reductions. In the first reduction, on input G, the reduction machine computes p_i, for each of the first $m = \log n$ prime numbers, and transforms G into a circuit C_i with addition modulo p_i gates, with the property that the output of the circuit coincides with $P \bmod p_i$ (see the preliminaries). The second reduction transforms the sequence of C_i circuits into a a sequence of graphs G_{C_i} (as in the proof of Theorem 4.3) in which there is an automorphism (with some restrictions encoding the input conditions) mapping z_0^i (the node corresponding to the output gate of G_{C_i}) to z_j^i if and only if $P \bmod p_i = j$. The number of paths from s to t in G is then 0 if and only for all $i \leq m$ there is an automorphism in G_{C_i} encoding the input values of C_i and mapping z_0^i to itself. This can be easily reduced to GI as explained in the preliminaries.

By fixing $m = \log n$, by the Prime Number Theorem, p_m is bounded by $\log^2 n$, and the computation of the small primes p_i can be done in logarithmic space. ■

Observe that in the graphs obtained in this reduction, the size of the classes of the nodes with the same color are not bounded by a constant as before, but by p_m^2.

Basically the same proof holds for proving hardness for the class C$_=$L. Here instead of checking that the number of paths from s to t is 0, we have to check that this number coincides with some exact threshold $f(G) \leq 2^n$. For this the reduction machine has to compute for each small prime p_i the residue $r_i = f(G) \bmod p_i$ (this can be done in NC^1 [26]), and then check whether there is an automorphism that for all i maps z_0^i to $z_{r_i}^i$.

Corollary 5.2 *GI is hard for C$_=$L under logarithm space many-one reductions.*

In fact, we can reduce any logarithmic space counting function to GI. We understand by this that for any function $f \in$ #L the set

$$A_f = \{\langle x, 0^i \rangle \mid \text{the } i\text{-th bit of } f(x) \text{ is } 1\}$$

is many-one reducible to GI.

For proving this reduction, we need two known results. On the one hand we need the surprising fact that division can be computed by NC^1 circuits [12]. More precisely we need the following part of the mentioned result:

Theorem 5.3 *[12] There is a logarithmic space uniform family of NC^1 circuits that on input the Chinese reminder representation $(x_1, \ldots, x_n)$ in base B_n of a number x, outputs the binary representation of x.*

We also need (the proof of) the following result, basically saying that the computation of an NC^1 circuit can be encoded into a single instance of GI.

Theorem 5.4 *[17] GI is hard for NC^1 under DLOGTIME many-one reductions.*

The proof of this result shows using some of the closure properties of GI that an NC^1 circuit with input values can be reduced to a pair of graphs that are isomorphic iff the value of the output gate of the circuit is 1. The inputs of the circuit are represented by pairs of graphs that are isomorphic if and only if the value of the input was a 1, and inner gates are transformed in a bottom up fashion into pairs of graphs. Having this in mind, we can now show the hardness of GI with respect to #L.

Theorem 5.5 *Every #L function*[1] *is logarithmic space many-one reducible to GI.*

Proof. Let $f \in$ #L. For some polynomial q, it is possible to construct in logarithmic space for each $x \in \Sigma^*$ a graph G_x with at most $q(|x|)$ nodes so that $f(x)$ is the number of $s-t$ paths in G_x. Let i be the bit of $f(x)$ we want to reduce to GI, and let $m = \log(q(|x|))$. By Theorem 5.3, in order to compute $f(x)$, is suffices to compute its Chinese reminder representation $(f(x)_1, \ldots, f(x)_m)$ in B_m. Once this is done, $f(x)$ can be computed by an NC^1 circuit.

To obtain the Chinese reminder representation the reduction machine computes prime number p_i, for every $1 \leq i \leq m$, and reduces G_x to a circuit with addition gates in $\mathbb{Z}_{p_i}$, as in the proof of Theorem 5.1. It then produces p_i pairs of graphs with the property that in the j-th pair the graphs are isomorphic iff $f(x) = j - 1 \bmod p_i$. These form a list of $\sum_{i=1}^{m} p_i$ pairs of graphs, and can be considered as an encoding of the CRR of $f(x)$ $(f(x)_1, \ldots, f(x)_m)$ of the form $(w_1, \ldots, w_m)$ where each $w_i \in \{0,1\}^{p_i}$ is formed by 0's with a 1 in position $f(x)_i + 1$. The 0's and 1's in the w_i's are encoded by pairs of non-isomorphic and isomorphic pairs respectively.

By Theorem 5.3 it is possible to construct in logarithmic space an NC^1 circuit that having as inputs the CRR of $f(x)$, outputs the i-th bit of $f(x)$. We can consider the list of pairs of graphs as the inputs of this circuit, and as done in the proof of Theorem 5.4, the reduction machine can transform the whole circuit into a single pair of graphs. These graphs are isomorphic depending on the output of the circuit, which coincides with the i-th bit of $f(x)$. ■

Observe that since Theorem 5.3 is only known to hold for logarithmic space uniform NC^1, the hardness result in Theorem 5.5 is not known to be true for uniform reductions using less resources than logarithmic space, as do the results in previous sections.

As mentioned in the preliminaries, for a set $L \in$ PL, there is a function $f \in$ #L such that for any input x, $x \in L$ iff the most significant bit of $f(x)$ is 1.

The next result follows then directly from Theorem 5.5.

Corollary 5.6 *GI is hard for the class PL under logarithmic space many-one reductions.*

The class DET of problems NC^1 Turing reducible to the determinant coincides with $NC^1(\#L)$ (see e.g. [2]). Combining (the proof of) Theorem 5.4 and Theorem 5.5, we obtain the strongest known hardness result for GI:

Corollary 5.7 *GI is hard for the class DET under logarithmic space many-one reductions.*

[1] In fact this result also holds for the more powerful class of GapL functions defined in [2].

5.1. Matching is reducible to GI

We finish by mentioning an interesting connection between the perfect matching problem and GI. The perfect matching problem consist in deciding whether a given undirected graph has a perfect matching, that is, a set of edges that contain all the vertices, and such that no two of these edges share a vertex. This problem has been intensively studied, but like GI, it has resisted all classification attempts in terms of completeness in a class. The problem has polynomial time algorithms, and it is known to be in random NC [18, 27]. In [5] it has been proved that for any $k \geq 2$, the perfect matching problem can be randomly reducible to a set in Mod_kL. Together with Theorem 4.3 this implies:

Corollary 5.8 *Matching is reducible to GI under logarithmic space randomized reductions.*

Since the reduction works correctly with probability exponentially close to 1, for each input size n there is a sequence of random choices that can be taken as correct advice in the reduction of all instances of size n. This implies a non-uniform reduction from Matching to GI. Moreover as noted in [3], under a natural hardness hypothesis, the reduction from Matching to Mod_kL can be derandomized using techniques from [16, 19]. This yields:

Corollary 5.9 *If there is a set A in DSPACE(n) and $\delta > 0$ with the property that, for all large n, no circuit of size less than $2^{\delta n}$ accepts exactly the strings of length n in A, then perfect matching is reducible to GI under logarithmic space many-one reductions.*

Acknowledgments: I would like to thank V. Arvind and J. Köbler for interesting discussions that clarified the results of this paper, and E. Allender to pointing reference [12] to me. I also would like to thank D. Therien for organizing the McGill Invitational Workshop where this research started.

References

[1] E. Allender and D. Mix Barrington, Uniform circuits for division: consequences and problems, personal communication, June 2000.

[2] E. Allender and M. Ogihara, Relationships among PL, #L, and the determinant, *RAIRO Theoretical Information and Applications* 30:1–21, 1996.

[3] E. Allender, K. Rheinhardt and S. Zhou. Isolation, matching and counting: uniform and nonuniform upper bounds. To appear in Journal of Computer and System Science, 2000.

[4] C. Àlvarez and B. Jenner. A very hard logspace counting class. *Theoretical Computer Science*, 107:3–30, 1993.

[5] L. Babai, A. Gál and A. Widgerson. Superpolynomial lower bounds for monotone span programs. *Combinatorica* 19, 1999.

[6] L. Babai, D. Grigoryev, and D. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proc. 14th ACM Symp on Theory of Computing* pp. 310–324, 1982.

[7] L. Babai and E. Luks. Canonical labeling of graphs. In *Proc. 15th ACM Symp on Theory of Computing* pp. 171–183, 1983.

[8] R. Boppana J. Hastad and S. Zachos. Does co-NP have short interactive proofs? In *Information Processing Letters* 25, pp. 27–32, 1987.

[9] G. Buntrock, C. Damm, U. Hertrampf and C. Meinel. Structure and importance of logspace-MOD-classes. In *Math. System Theory* 25: 223–237, 1992.

[10] S. R. Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium'97*, Lecture Notes in Computer Science #1289, Springer-Verlag, 1997, pp. 18-33.

[11] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1):2–22, 1985.

[12] A. Chiu, G. Davida and B. Litow. NC^1 division. Preliminary version, http://www.cs.jcu.edu.au/~bruce/papers/ cr00.ps.gz

[13] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal con Computing* 6:675–695, 1977.

[14] J. E. Hopcroft and R. E. Tarjan. A V^2 algorithm for determining isomorphism of planar graphs. *Information Processing Letters*, 32–34, 1971.

[15] N. Immerman. Nondeterministic space is closed under complement. *SIAM Journal on Computing*, 17:935–938, 1988.

[16] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. *Proc. 29th ACM Symp. on Theory of Computing* 220–229, 1997.

[17] B. Jenner, P.McKenzie and J. Torán. *A note on the hardness of tree isomorphism.* In Proc. 13th IEEE Computational Complexity Conference 101–106, 1998.

[18] R. Karp, E. Upfal and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica* 6:35–48, 1986.

[19] A. Klivans and D. van Melkebeek. Graph isomorphism has subexponential size proofs unless PH collapses. In *Proc. 31st Symp. on Theory of Computing* 659–667, 1999.

[20] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem — Its Structural Complexity.* Progress in Theoretical Computer Science. Birkhäuser, Boston, 1993.

[21] H. Lewis and C. Papadimitriou. Symmetric space-bounded computation. *Theor. Computer Science* 19:161–187, 1982.

[22] S. Lindell. A logspace algorithm for tree canonization. In *Proc. of the 24th STOC*, 400–404. ACM, 1992.

[23] E. Luks. Isomorphism of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.

[24] E. Luks. Parallel algorithms for permutation groups and graph isomorphism. *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 292–302, 1986.

[25] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–132, 1979.

[26] P. McKenzie and S. Cook. The parallel complexity of Abelian permutation group problems. *SIAM Journal on Computing* 19:880–909, 1987.

[27] K. Mulmuley, U. Vazirani and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica* 7:105–113, 1987.

[28] N. Nisan and A. Ta-Shma. Symmetric logspace is closed under complement. *Proc. 27th Symp. on Theory of Computing*, 140–146, 1995.

[29] W. Ruzzo, J. Simon and M. Tompa. Space bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28:216–230, 1984.

[30] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.

[31] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica* 26:279–284, 1988.