On Graph Isomorphism for Restricted Graph Classes*

Johannes Köbler

Institut für Informatik, Humboldt-Universität zu Berlin, D-10099 Berlin, Germany
koebler@informatik.hu-berlin.de

Abstract. Graph isomorphism (GI) is one of the few remaining problems in NP whose complexity status couldn't be solved by classifying it as being either NP-complete or solvable in P. Nevertheless, efficient (polynomial-time or even NC) algorithms for restricted versions of GI have been found over the last four decades. Depending on the graph class, the design and analysis of algorithms for GI use tools from various fields, such as combinatorics, algebra and logic.
In this paper, we collect several complexity results on graph isomorphism testing and related algorithmic problems for restricted graph classes from the literature. Further, we provide some new complexity bounds (as well as easier proofs of some known results) and highlight some open questions.

1 Introduction

In this section we briefly review some important complexity results for graph isomorphism as well as for related problems as, e.g., computing the automorphism group $\mathrm{Aut}(X)$ of a given graph X in terms of a generating set of automorphisms (we refer to this problem as AUT) or the canonization problem (i.e., renaming the vertices of a given graph in such a way that all isomorphic graphs become equal). It is easy to see that GI reduces to both problems (in fact, in the unrestricted case, GI and AUT are polynomial-time equivalent, whereas it is open whether canonization reduces to GI). Formal definitions of these and other concepts used in the paper are deferred to the next section. In some sense, graph isomorphism represents a whole class of algorithmic problems; for example, GI is polynomial-time equivalent to the isomorphism problem for semigroups as well as for finite automata [15]. For the interesting relationships between GI and isomorphism testing for other algebraic structures like groups and rings we refer the reader to the excellent surveys [1, 5].

Two graphs X and Y are *isomorphic* (denoted by $X \cong Y$) if there is a bijective mapping g between the vertices of X and the vertices of Y that preserves the adjacency relation, i.e., g relates edges to edges and non-edges to non-edges. *Graph Isomorphism* is the problem of deciding whether two given graphs are isomorphic. The problem has received considerable attention since it is one of

* Work supported by a DST-DAAD project grant for exchange visits.

the few natural problems in NP that are neither known to be NP-complete nor known to be solvable in polynomial time.

There is some evidence that GI is not NP-complete. First of all, GI is polynomial-time equivalent to its counting version #GI which consists in computing the number of isomorphisms between two given graphs [40]. In contrast, the counting versions of NP-complete problems (like #SAT) are typically much harder; in fact they are #P-complete and hence at least as hard as any problem in the polynomial-time hierarchy [46]. More strikingly, the complement of GI belongs to the class AM of decision problems whose positive instances have short membership proofs checkable by a probabilistic verifier [7]. As a consequence, GI is not NP-complete unless the polynomial hierarchy collapses to its second level [16, 45].

A promising approach in tackling the graph isomorphism problem for general graphs is to design efficient algorithms for restricted graph classes. In fact, Luks' efficient GI algorithm for graphs of bounded degree [38] yields the fastest known general graph isomorphism algorithm due to Babai, Luks, and Zemlyachenko [6, 11, 49]. The strongest known hardness result due to Torán [47] says that GI is hard for the class DET of problems that are NC^1 reducible to the computation of the determinant of a given integer matrix (cf. [20]). DET is a subclass of NC^2 (even of TC^1) and contains NL as well as all logspace counting classes like Mod_kL, C=L, PL and L(#L) [2, 17].

The first significant complexity result for restricted graph classes is the linear time canonization algorithm for trees, designed by Hopcroft and Tarjan [29], and independently by Zemlyachenko [48]. Miller and Reif [42] later gave an NC algorithm for tree canonization, based on tree contraction methods. Then Lindell came up with a logspace algorithm for tree canonization [37]. As shown in [34], this upper bound is optimal, since tree isomorphism is also hard for L under AC^0 reductions. If we consider complexity bounds below L, then the representation that we use to encode the input trees becomes important. For trees encoded in the string representation, Buss [18] located the canonization problem even in NC^1 (which is also optimal [34]).

Shortly after the linear time canonization algorithm for trees was found, Hopcroft, Tarjan and Wong designed a linear time canonization algorithm for planar graphs [28, 30]. This line of research has been pursued by Lichtenstein, Miller, Filotti, and Mayer, culminating in a polynomial-time GI algorithm for graphs of bounded genus [36, 41, 33]. In 1991, Miller and Reif [42] designed an AC^1 algorithm for planar graph isomorphism.

Using a group theoretic approach, Babai showed in 1979 that GI is decidable in random polynomial time for the class $\mathcal{CG}_b$ of colored graphs with constant color multiplicity b. More precisely, the vertices of a graph in $\mathcal{CG}_b$ are colored in such a way that at most b vertices have the same color and we are only interested in isomorphisms that preserve the colors. Inspired by Babai's work, Furst, Hopcroft and Luks [22] developed efficient solutions for various permutation group problems and as a byproduct they could eliminate the need for randomness in Babai's algorithm. Both algorithms exploit, in a significant manner, the

fact that the automorphism group $\mathrm{Aut}(X)$ of a graph X with constant color class size, is contained in the product of constant size symmetric groups. For such groups the pointwise stabilizer series can be used to successively compute generators for the groups in the series.

In a breakthrough result, Luks in 1982 was able to design an algorithm for computing $\mathrm{Aut}(X)$ in polynomial time for graphs of bounded degree [38]. To achieve this result, Luks considerably refined the group-theoretic techniques used in earlier algorithms. By combining Luks' algorithm with a preprocessing procedure due to Zemlyachenko [49] (see also [6]) for reducing the color valence of the input graphs, Babai and Luks obtained an $2^{O(\sqrt{n \log n})}$ time-bounded GI algorithm, where n denotes the number of vertices in the input graphs (see [11]). This is the fastest algorithm known for the unrestricted graph isomorphism problem. In [11] it is also shown that for general graphs there is a $2^{O(n^{1/2+o(1)})}$ canonizing algorithm which closely matches the running time of the best known decision algorithm.

Later, Luks in [39] gave a remarkable NC algorithm for the bounded color class case. Building on [39], Arvind, Kurur and Vijayaraghavan further improved Luks' NC upper bound by showing that GI for graphs in $\mathcal{CG}_b$ (we denote this restriction of GI by GI_b) is in the $\mathrm{Mod}_k\mathrm{L}$ hierarchy (and hence in TC^1), where the constant k and the level of the hierarchy depend on b [4]. Prior to this result, Torán showed that GI_{b^2} is hard for the logspace counting class $\mathrm{Mod}_b\mathrm{L}$ [47]. Torán's lower bound has been extended in [4] where it is shown that for each level in the $\mathrm{Mod}_k\mathrm{L}$ hierarchy there is a constant b such that GI_b is hard for this level.

The pointwise stabilizer series approach has also been applied by Babai, Grigoryev and Mount to compute the automorphism group for graphs with bounded eigenvalue multiplicity [10]. By applying group theory to a greater extent, Babai, Luks, and Séress were able to show that isomorphism testing for these graph classes is in NC [12, 8, 39]. However, it is still open whether also Luks' efficient GI algorithm for graphs with bounded degree is parallelizable.

Question 1. *Is GI for graphs with bounded degree in NC?*

Ponomarenko proved that GI for graphs with excluded minors is decidable in P [43]. In 1990, Bodlaender gave a polynomial-time GI algorithm for graphs of bounded treewidth [14]. This class contains all series-parallel graphs, all outerplanar graphs, all graphs with constant bandwidth (or cutwidth) and all chordal graphs with constant clique-size.

Very recently, Grohe and Verbitsky [26] improved Bodlaender's upper bound by showing that GI for graphs of bounded treewidth is in TC^1. This follows by combining the following two results which are interesting on their own.

First, they show that a parallel version of the r-round k-dimensional Weisfeiler-Lehman algorithm (r-round WL^k for short) can be implemented as a logspace uniform family of TC circuits of depth $O(r)$ and polynomial size. As a consequence, for any class $\mathcal{C}$ for which the multidimensional WL algorithm correctly decides GI on $\mathcal{C}$ in $O(\log n)$ rounds, GI on $\mathcal{C}$ is decidable by a TC^1 algorithm.

As a second ingredient of the proof, Grohe and Verbitsky show that for $r = O(k \log n)$, the r-round WL^{4k+3} correctly decides GI on all graphs of treewidth at most k. This latter result is obtained by designing a winning strategy for a suitable Ehrenfeucht-Fraïssé game with $4k + 4$ pebbles and r moves. An interesting question in this context is whether this approach can be extended to the canonization version of WL.

Question 2. *Do graphs of bounded treewidth admit an* NC *(or even* TC^1*) canonization?*

As Grohe and Verbitsky use the WL algorithm to solve GI for graphs with bounded treewidth, it follows that these graphs have a TC^1 computable complete normal form (also called invariant). Although, as shown by Gurevich, canonization is polynomial-time reducible to computing a complete normal form [27], it is not clear whether such a reduction is computable in NC for graphs with bounded treewidth.

Another possibility to answer Question 2 affirmatively may be to use a variation of the WL algorithm to canonize the input graph. For example, in [32, Theorem 1.9.4] Immerman and Lander propose the following procedure: as soon as the refinement process stabilizes choose any vertex (or tuple) from the lexicographically smallest color class of size at least two and individualize it (i.e., give it a new color). Then restart WL and repeat the process until all color classes are singletons. The resulting (total) refinement induces unique names for all the vertices. An interesting question is whether this variant of WL indeed computes a canon for all graphs of bounded treewidth, and, provided the answer is yes, whether this task can be performed in a logarithmic number of rounds.

2 Preliminaries

In this section we fix the notation and give formal definitions for some of the concepts used in this paper. For other basic definitions we refer the reader to [35] or to any textbook on complexity like [13].

We denote the *symmetric group* of all permutations on a set A by $\mathrm{Sym}(A)$ and by S_n in case $A = \{1, \ldots, n\}$. Let G be a subgroup of $\mathrm{Sym}(A)$ and let $a \in A$. Then the set $\{b \in A \mid \exists g \in G : g(a) = b\}$ of all elements $b \in A$ reachable from a via a permutation $g \in G$ is called the *orbit* of a in G.

2.1 Colored graphs

Let $X = (V, E)$ denote a (finite) hypergraph, i.e., E is a subset of the power set $\mathcal{P}(V)$ of V. We always assume that the vertex set is of the form $V = [n]$, where $[n]$ denotes the set $\{1, \ldots, n\}$. For a subset $U \subseteq V$, we use $X[U]$ to denote the *induced subgraph* $(U, E(U))$ of X, where $E(U) = \{e \in E \mid e \subseteq U\}$. For usual graphs, i.e., $E \subseteq \binom{V}{2} = \{e \subseteq v \mid \|e\| = 2\}$, we use $\Gamma_X(u)$ to denote the neighborhood $\{v \in V \mid \{u, v\} \in E\}$ of vertex u in the graph X (if X is clear from the context we omit the subscript). Further, for disjoint subsets $U, U' \subseteq V$, we

use $X[U,U']$ to denote the *induced bipartite subgraph* $(U \cup U', E(U,U'))$, where $E(U,U')$ contains all edges $e \in E$ with $e \cap U \neq \emptyset$ and $e \cap U' \neq \emptyset$.

A *coloring* of X is given by a function $c : V \to [m]$. We represent colored hypergraphs as triples $X = (V, E, \mathcal{C})$, where $\mathcal{C} = (C_1, \ldots, C_m)$ is the color partition induced by c, i.e., $C_i = \{u \in V \mid c(u) = i\}$. We denote the class of all colored hypergraphs by $\mathcal{CHG}$ and the class of all colored graphs by $\mathcal{CG}$. Note that the class of uncolored (hyper)graphs can also be seen as a subclass of $\mathcal{CHG}$ where all nodes have color 1. In case $\|C_i\| \leq b$ for all $i \in [m]$, we refer to X as a *b-bounded (hyper)graph*. The class of all b-bounded graphs (hypergraphs) is denoted by $\mathcal{CG}_b$ (respectively, $\mathcal{CHG}_b$).

2.2 Isomorphisms and automorphisms

Let $X = (V, E, \mathcal{C})$ and $Y = (V, E', \mathcal{C})$ be hypergraphs and let g be a permutation on V. We can extend g to a mapping on subsets $U = \{u_1, \ldots, u_k\}$ of V by

$$g(U) = \{g(u_1), \ldots, g(u_k)\}.$$

g is an *isomorphism* between hypergraphs X and Y, if g preserves the edge relation, i.e.,

$$\forall e \subseteq V : e \in E \Leftrightarrow g(e) \in E'$$

as well as the color relation,

$$\forall i \in [m] : g(C_i) = C_i.$$

We also say that g maps X to Y and write $g(X) = Y$. If $g(X) = X$, then g is called an *automorphism* of X. We use $\mathrm{Aut}(X)$ to denote the automorphism group of X. Note that the identity mapping on V is always an automorphism. Any other automorphism is called *nontrivial*.

The decision problem HGI_b consists of deciding whether two given b-bounded hypergraphs X and Y are isomorphic (GI_b denotes the restriction of this problem to graphs). A related problem is the automorphism problem HGA_b (GA_b) of deciding if a given b-bounded hypergraph (respectively, graph) has a nontrivial automorphism. For uncolored (hyper)graphs $X = (V, E)$ we denote these problems by HGI, GI, HGA and GA, respectively.

2.3 Normal forms and canonization

In the following we assume an appropriate binary encoding of colored (hyper)graphs and we identify each graph X with its encoding. Let $\mathcal{D} \subseteq \mathcal{CHG}$ be a graph class and let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. We say that f computes a *normal form* for $\mathcal{D}$, if

$$\forall X, Y \in \mathcal{D} : X \cong Y \Rightarrow f(X) = f(Y).$$

If f also fulfils the backward implication, i.e.

$$\forall X, Y \in \mathcal{D} : X \cong Y \Leftrightarrow f(X) = f(Y),$$

f is called a *complete normal form* for $\mathcal{D}$. A normal form f for $\mathcal{D}$ that computes for any graph $X \in \mathcal{D}$ a graph $f(X)$ that is isomorphic to X, i.e.

$$\forall X, Y \in \mathcal{D} : X \cong f(X) \wedge [X \cong Y \Rightarrow f(X) = f(Y)],$$

is called a *canonization* for $\mathcal{D}$. Note that a canonization for $\mathcal{D}$ is also a complete normal form for $\mathcal{D}$. We call $f(X)$ the *canon* of X (w.r.t. f). Of course, $f(X)$ is uniquely determined by any isomorphism g between X and $f(X)$. We call any such g a *canonical relabeling* of X (w.r.t. f).

2.4 The Weisfeiler-Lehman algorithm

For the history of this approach to GI we refer the reader to [9, 19, 21]. We will abbreviate *k-dimensional Weisfeiler-Lehman algorithm* by WL^k. WL^1 is commonly known as the *canonical labeling* or *color refinement algorithm*. On input a colored graph $X = (V, E, \mathcal{C})$, where $\mathcal{C} = (C_1, \ldots, C_m)$, the algorithm proceeds in rounds starting with the *initial coloring* $\mathcal{C}^0 = \mathcal{C}$, i.e., c^0 assigns to each node $v \in V$ its color $c(v)$. In each round, each node $v \in V$ receives a new color that depends on the previous colors of v and all its neighbors. More precisely, in the $(i+1)$st round, WL^1 assigns to node v the color

$$c^{i+1}(v) = (c^i(v), \{\!\{\, c^i(u) \mid u \in \Gamma(v) \,\}\!\})$$

consisting of the preceding color $c^i(v)$ and the multiset $\{\!\{\, c^i(u) \mid u \in \Gamma(v) \,\}\!\}$ of colors $c^i(u)$ for all $u \in \Gamma(v)$. For example, $c^1(v) = c^1(w)$ if and only if for each color $i \in [m]$, v and w have the same number of neighbors with that color. To keep the color encoding short, after each round the colors are lexicographically sorted and renamed (hence the renamed colors are in the range $[m_i]$, where $m_i = \|\{c^i(v) \mid v \in V\}\| \leq n$). However, the algorithm retains a table that can be used to derive the old color names from the new ones. After r rounds, the r-round WL^1 stops and outputs the multiset $\{\!\{\, c^r(v) \mid v \in V \,\}\!\}$ of colors in the coloring C^r (together with the tables retained at each round). Note that as long as C^{i+1} is a proper refinement of C^i, the number of colors increases. Hence, the coloring stabilizes after at most n rounds, i.e. $C^{s+1} = C^s$ for some $s < n$. We call C^s the WL^1-*stable coloring* of X.

Following the same idea, the k-dimensional version iteratively refines a coloring of V^k. The initial coloring of a k-tuple $\bar{v}$ is the isomorphism type of the subgraph induced by the vertices in $\bar{v}$ (viewed as a labeled graph where each vertex is labeled by its color and by the positions in the tuple where it occurs). The refinement step takes into account the colors of all neighbors of $\bar{v}$ in the Hamming metric (see [19, 26] for details).

Since the coloring is stable after at most n^k rounds, WL^k can be implemented in polynomial time for each constant dimension k. Further, since the colorings computed by the WL algorithm in each round only depend on the isomorphism class of X, it is clear that WL computes a normal form on the class of all graphs. We say that the r-round WL^k *works correctly* for a graph X, if the output for X is distinct from all outputs produced for any nonisomorphic graph $Y \ncong X$.

It is clear that the r-round WL^k computes a complete normal form on a graph class $\mathcal{D}$, provided that it works correctly for each graph $X \in \mathcal{D}$ (note that for some graph classes the latter condition might be stronger than the former).

Of course, WL^n needs at most one round to work correctly on all graphs with n vertices. In fact, already WL^1 works correctly on all trees and almost all graphs (in the $\mathcal{G}_{n,1/2}$ model), and WL^2 succeeds on all graphs of color class size 3 [32]. Thus there was some hope that a low dimensional WL algorithm may work correctly on all graphs. However, in 1990 Cai, Fürer and Immerman [19] proved a striking negative result: For any sublinear dimension $k = o(n)$, WL^k does not work correctly even on graphs of vertex degree 3 and color class size 4. Nevertheless, it was realized later that a constant-dimensional WL is still applicable to particular classes of graphs, including planar graphs [23], graphs of bounded genus [24], and graphs of bounded treewidth [25].

3 Hardness of HGA

To show that there are n-vertex graphs of vertex degree 3 and color class size 4 that are hard instances for $\mathrm{WL}^{o(n)}$, Cai, Fürer and Immerman used a graph gadget that originally appeared in [31]. This gadget has also been used by Torán in a significant manner to show that GI and GA are hard for various subclasses of TC^1 [47]. Here we use a hypergraph variant of this gadget to show that for any prime p, HGA_p is hard for $\mathrm{Mod}_{p!}\mathrm{L}$. The proof given here simplifies a proof of a similar result in [3].

It is well-known that the following problem is $\mathrm{Mod}_p\mathrm{L}$ complete (cf. [17]). Given a homogenous system

$$\sum_{j\in[n]} a_{ij}x_j = 0,\ i \in [k] \tag{1}$$

of linear equations over the field $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$, decide whether (1) has a nontrivial solution $\bar{x} \in \mathbb{Z}_p^n$. This problem remains $\mathrm{Mod}_p\mathrm{L}$ complete, if we require that the support $S_i = \{j \in [n] \mid a_{ij} \neq 0\}$ of each equation contains at most three elements and $S_j \neq S_k$ for $j \neq k$ (these restrictions are not really necessary but they simplify the reduction and keep the orbit size of the hyperedges in the reduced hypergraph small). Now consider the following hypergraph $X = (V, E, \mathcal{C})$ with

$$V = \bigcup_{j=1}^{n} V_j,\ \ E = \bigcup_{j=1}^{n} Z_j \cup \bigcup_{i=0}^{k} E_i \text{ and } \mathcal{C} = (C_1, C_1', C_1'', \ldots, C_n, C_n', C_n''),$$

where

$$\begin{aligned}
V_j =& C_j \cup C_j' \cup C_j'',\\
C_j =& \{u_x^j \mid x \in \mathbb{Z}_p\}, C_j' = \{v_x^j \mid x \in \mathbb{Z}_p\}, C_j'' = \{w_x^j \mid x \in \mathbb{Z}_p\},\\
Z_j =& \{\{u_x^j, v_x^j\}, \{v_x^j, w_x^j\}, \{w_x^j, u_{x+1}^j\} \mid x \in \mathbb{Z}_p\}, \text{ and}\\
E_i =& \{\{u_{x_j}^j \mid j \in S_i\} \mid \textstyle\sum_{j\in[n]} a_{ij}x_j = 0\}.
\end{aligned}$$

In the hypergraph X we have for each variable x_j a cycle $X_j = X[V_j]$ such that $\mathrm{Aut}(X_j)$ is isomorphic to the additive group $(\mathbb{Z}_p, +)$. Fix any isomorphism φ between $\mathrm{Aut}(X_j)$ and $\mathbb{Z}_p$ and denote the automorphism $g \in \mathrm{Aut}(X_j)$ with $\varphi(g) = x$ by g^j_x. Then $\mathrm{Aut}(X_j)$ is represented as $\{g^j_x \mid x \in \mathbb{Z}_p\}$ and we have $g^j_x \circ g^j_{x'} = g^j_{x+x'}$.

For any vector $\bar{x} = (x_1, \ldots, x_n)$ we use $\bar{x}|_{S_i}$ to denote the s_i-dimensional projection $(x_j)_{j \in S_i}$ of $\bar{x}$ to S_i. Since E_i contains for each solution $\bar{x} = (x_1, \ldots, x_n)$ of the i-th equation in (1) the hyperedge $e(\bar{x}|_{S_i}) = \{u^j_{x_j} \mid j \in S_i\}$, E_i consists of exactly $p^{s_i - 1}$ hyperedges. We use L_i to denote the set of vectors $\bar{x} \in \mathbb{Z}_p^{s_i}$ with $e(\bar{x}) \in E_i$.

Of course, for $p = 2, 3$ we can simplify X to the graph $\hat{X} = X[C_1 \cup \cdots \cup C_n]$ since in these cases the groups $\mathrm{Aut}(\hat{X}_j)$ are cyclic anyway. Figure 1 shows the graph $\hat{X}$ corresponding to the equation $x_1 + x_2 - x_3 = 0$ over $\mathbb{Z}_3$.

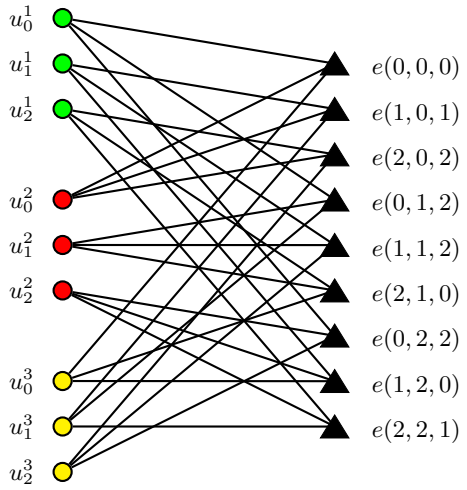

Fig. 1. The hypergraph gadget for the equation $x_1 + x_2 - x_3 = 0$ over $\mathbb{Z}_3$.

Now it is easy to see that for each $i \in [k]$ the automorphism group $\mathrm{Aut}(Y_i)$ of the hypergraph $Y_i = (W_i, F_i, \mathcal{C}_i)$ where $W_i = \bigcup_{j \in S_i} V_j$, $F_i = E_i \cup \bigcup_{j \in S_i} Z_j$ and $\mathcal{C}_i$ is the restriction of the coloring $\mathcal{C}$ to W_i, is isomorphic to the solution space L_i of the equation $\sum_{j \in S_i} a_{ij} x_j = 0$. For example, if $S_i = \{1, 2, 3\}$ and the i-th equation of (1) is $x_1 + x_2 - x_3 = 0$, then

$$\mathrm{Aut}(Y_i) = \{(g^1_{x_1}, g^2_{x_2}, g^3_{x_3}) \mid x_1 + x_2 - x_3 = 0\}.$$

Hence, a permutation $g = (g^1_{x_1}, \ldots, g^n_{x_n}) \in \mathrm{Aut}(X_1) \times \cdots \times \mathrm{Aut}(X_n)$ is an automorphism of X if and only if for all $i \in [k]$, the restriction of g to W_i is an automorphism of Y_i, implying that

$$\mathrm{Aut}(X) = \{(g^1_{x_1}, \ldots, g^n_{x_n}) \mid (x_1, \ldots, x_n) \text{ is a solution of (1)}\}.$$

This shows that $X \in \mathrm{HGA}_p$ if and only if the system (1) has a nontrivial solution. Since the reduction from the given homogenous system (1) to the hypergraph X can be performed in AC^0, it follows that for any prime $q \leq p$, HGA_p is hard for the class $\mathrm{Mod}_q\mathrm{L}$ under AC^0 many-one reductions.

Moreover, since HGA_p has an easily computable or-function (just take the union of the graphs where we assume w.l.o.g. that the input graphs have no colors in common) and since any set in the class $\mathrm{Mod}_m\mathrm{L}$ can be represented as the union $A_1, \ldots, A_k$ of sets A_i in $\mathrm{Mod}_{p_i}\mathrm{L}$, where $p_1, \ldots, p_k$ are the prime factors of m [17], it immediately follows that HGA_p is even hard for $\mathrm{Mod}_{p!}\mathrm{L}$. Since the orbit size of the hyperedges in the reduced hypergraph is bounded by p^2, we also get that GA_{p^2} is $\mathrm{Mod}_{p!}\mathrm{L}$ hard.

Theorem 3. *HGA_p and GA_{p^2} are hard for $\mathrm{Mod}_{p!}\mathrm{L}$.*

In [3] it is shown that GA_4 (as well as GA_5 and HGA_2) in fact is complete for the class $\mathrm{Mod}_2\mathrm{L} = \oplus\mathrm{L}$. The best known upper bound for HGA_b, $b > 2$, is P [3]. We remark that if the hyperedges are all of constant size, i.e., $\|e\| \leq k$ for all $e \in E$, then HGA_b is reducible to $\mathrm{GA}_{b'}$ for $b' = b^k$ which is known to be in TC^1 [39, 4]. However, when hyperedges are of unbounded size, it is not clear whether HGA_b is reducible to $\mathrm{GA}_{b'}$ for any constant b'.

Question 4. *Is HGA_b in NC for some constant $b > 2$?*

Torán's proof that GI and GA are hard for NL crucially hinges on the fact that the produced graphs have unbounded color classes. Since already in the 2-bounded case the orbits of the edges of a hypergraph can have exponential size it might be possible to reduce NL to HGA_b (or HGI_b) for a constant b. Note that the orbit size of the edges of a b-bounded graph is at most b^2.

Question 5. *Is there any constant b for which HGA_b (or HGI_b) is NL hard?*

4 Logspace canonization of 3-bounded graphs

In this section we improve the result from [34] that GI for 2-bounded as well as for 3-bounded graphs is equivalent to undirected graph reachability (and therefore complete for L [44]). We first describe a logspace canonization algorithm for 2-bounded graphs. This algorithm performs a 1-round WL^1 and uses individualization to refine the remaining size two color classes. We also sketch how this algorithm can be improved to handle the 3-bounded case. For the complete proof we refer the reader to the journal version of [3] (in preparation).

Let $X = (V, E, \mathcal{C})$ be a b-bounded graph and let $\mathcal{C} = (C_1, \ldots, C_m)$. We use X_i to denote the graph $X[C_i]$ induced by C_i and X_{ij} to denote the bipartite graph $X[C_i, C_j]$ induced by the pair of color classes C_i and C_j. Since it suffices to compute a canonical relabeling for X we can assume that all vertices in the same color class C_i have the same degree and each graph X_i is regular of degree at most $(\|C_i\| - 1)/2$. Otherwise we can either canonically split C_i into smaller color classes or we can replace X_i by the complement graph. Further, we assume

that the edge set E_{ij} of X_{ij} is of size at most $\|C_i\| \cdot \|C_j\|/2$, since otherwise, we can replace X_{ij} by the complement bipartite graph.

We say that two color classes C_i, C_j with $\|C_i\| = \|C_j\|$ are *directly linked*, if E_{ij} is a perfect matching in X_{ij}. C_i and C_j are *linked*, if C_i is reachable from C_j by a chain of directly linked color classes. We make use of some basic facts from [3].

Lemma 6. [3] *For any directly linked pair C_i, C_j of color classes there is a bijection $\pi_{ij} : \mathrm{Sym}(C_i) \to \mathrm{Sym}(C_j)$ such that for any automorphism $g = (g_i, g_j) \in \mathrm{Aut}(X_{ij})$ it holds that $g_j = \pi_{ij}(g_i)$.*

Let G_i be the intersection of $\mathrm{Aut}(X_i)$ with the projections of $\mathrm{Aut}(X_{ij})$ on C_i for all $j \neq i$. Any subgroup of the symmetric group $\mathrm{Sym}(C_i)$ of all permutations on C_i is called a *constraint* for C_i. We call G_i the *direct constraint* for C_i.

Lemma 7. [3] *For a given b-bounded graph, the direct constraints of each color class can be determined in deterministic logspace.*

We use Lemma 6 to define a symmetric relation on constraints. Let G_i and G_j be constraints of two directly linked classes C_i and C_j, respectively, and let g_{ij} be the bijection provided by Lemma 6. We say that G_i is *directly induced by* G_j, if g_{ij} is an isomorphism between the groups G_i and G_j. Further, a constraint G *is induced by* a constraint H, if G is reachable from H via a chain of directly induced constraints. Note that the latter relation is an equivalence on the set of all constraints. We call the intersection of all constraints of C_i that are induced by some direct constraint the *induced constraint* of C_i and denote it by G_i'.

Lemma 8. [3] *For a given b-bounded graph, the induced constraints of each color class can be determined in deterministic logspace.*

Proof. Consider the undirected graph $X' = (V', E')$ where V' consists of all constraints G in X and $E' = \{(G, H) \mid G \text{ is directly induced by } H\}$. In this graph we mark all direct constraints computed by Lemma 7 as special nodes. Now, the algorithm outputs for each color class C_i the intersection of all constraints for C_i that are reachable from some special node, and since SL = L [44], this can be done in deterministic logspace. □

We define two special types of constraints. We say that C_i is *split*, if its induced constraint G_i' has at least two orbits, and we call the partition of C_i in the orbits of G_i' the *splitting partition* of C_i. Further, a class C_i of size b is called *whole*, if its induced constraint G_i' is the whole group $\mathrm{Sym}(C_i)$. The following lemma summarizes some properties of whole color classes.

Lemma 9. *Let C_i be a whole color class in a b-bounded graph X and let C_j be a color class such that $E_{ij} \neq \emptyset$. Then the following holds.*

- *$X[C_i, \Gamma_{X_{ij}}(C_i)]$ is semiregular,i.e., the degree of any node u in the bipartite graph only depends on its (non)membership to C_i.*

– *If also C_j is whole, then $\|C_i\| = \|C_j\|$ and C_i, C_j are directly linked.*
– *If C_j is split or $\|C_j\| < b$, then all vertices in C_i have the same neighborhood in X_{ij}.*

Lemma 9 tells us that the action of an automorphism on a whole color class C is not influenced by its action on color classes that are either smaller or split, i.e., only other whole color classes can influence C. Similarly, it follows that WL^1 will never refine any of the whole color classes in X. Let W be the union of all whole color classes. Then $\mathrm{Aut}(X[W])$ is computable in logspace.

Lemma 10. [34, 3] *A generating set for* $\mathrm{Aut}(X[W])$ *is computable in* FL.

Proof. The algorithm works by reducing the problem to reachability in undirected graphs. For each whole color class C_i we create a set P_i of $b!$ nodes (one for each permutation of C_i). Recall that if C_i and C_j are directly linked, then each $g \in P_i$ induces a unique permutation $h = \pi_{ij}(g)$ on C_j and hence, we put an undirected edge between g and h. This gives an undirected graph G with $(b-1)!\|W\|$ nodes.

A connected component P in G that picks out at most one element g_i from each set P_i defines a valid automorphism g for the graph $X[W]$, if P contains only elements $g_i \in \mathrm{Aut}(X_i)$. On the color classes C_i, for which P contains an element $g_i \in P_i$, g acts as g_i, and it fixes all nodes of the other color classes. By collecting these automorphisms we get a generating set for $\mathrm{Aut}(X[W])$ and since $\mathrm{SL} = \mathrm{L}$ [44], this can be done in deterministic logspace. □

Now we prove that WL^1 on 2-bounded graphs can be implemented in logspace.

Theorem 11. *For graphs in $\mathcal{CG}_2$ the* WL^1*-stable coloring is computable in* FL.

Proof (sketch). Let $X = (V, E, \mathcal{C})$ be a 2-bounded graph with coloring $\mathcal{C} = (C_1, \ldots, C_m)$. The only way that a color class C_i gets directly split (i.e. by its direct constraint G_i) is that one node $a \in C_i$ is incident to some color class C_j whereas the other node $b \in C_i$ is not. Let C_j be the lexicographically smallest color class with this property. Then WL^1 refines C_i into $(\{b\}, \{a\})$. These are exactly the refinements that WL^1 performs by the initial coloring and they are clearly computable in logspace.

If C_i gets refined in a later round, then this refinement is caused by a direct link to a color class that has been refined earlier. Let C_j be the lexicographically smallest directly split color class which is linked to C_i by a chain $(C_j, \ldots, C_i)$ of directly linked color classes of minimal length. Then WL^1 transposes the refinement of C_j to C_i via the chain $(C_j, \ldots, C_i)$. Clearly, also these refinements are computable in logspace. Finally, observe that the whole color classes never get refined by WL^1. In fact, they form orbits in $\mathrm{Aut}(X)$. □

As an easy consequence we get a logspace canonization for all 2-bounded graphs.

Theorem 12. *$\mathcal{CG}_2$ admits a logspace canonization.*

Proof (sketch). Let $X = (V, E, \mathcal{C})$ be a 2-bounded graph with coloring $\mathcal{C} = (C_1, \ldots, C_m)$. By Theorem 11 the WL^1-stable coloring X' of X is computable in logspace. If X' assigns unique colors to all vertices, then a canonical labeling is determined.

Otherwise, for each connected component of linked color classes (of size 2), the algorithm determines the lexicographically smallest color class C in that component. Since the nodes of different C's can be flipped independently, the algorithm can select in each such color class an arbitrary node and give it a new color. Now it suffices to run WL^1 once more to compute the stable coloring for the modified graph which will provide unique colors for all vertices. □

We notice that the above proof also shows that the canonization version of WL^1 (as proposed in [32, Theorem 1.9.4]) succeeds on the class $\mathcal{CG}_2$ (despite the fact that WL^1 does not work correctly on $\mathcal{CG}_2$ [32, Corollary 1.6.2]).

Question 13. *For which values of k and b does the canonization version of WL^k succeed in canonizing the graphs in $\mathcal{CG}_b$?*

Similar to the proof of Theorem 11 it can be shown that also for graphs in $\mathcal{CG}_3$ the WL^1-stable coloring is computable in logspace but it is not clear whether this generalizes.

Question 14. *What is the complexity of computing the WL^k-stable coloring for graphs in $\mathcal{CG}_b$?*

Immerman and Lander have shown that WL^2 works correctly on all 3-bounded graphs, implying that the canonizing version of WL^2 succeeds on the class $\mathcal{CG}_3$ [32]. Here we give a logspace canonization algorithm for this class.

Theorem 15. *$\mathcal{CG}_3$ admits a logspace canonization.*

Proof (sketch). Let $X = (V, E, \mathcal{C})$ be a 3-bounded graph with coloring $\mathcal{C} = (C_1, \ldots, C_m)$. Let $\mathcal{C}_w$ denote the subclass of $\mathcal{C}$ containing all whole color classes of size 3 that are linked to the lexicographically smallest whole class C_i and let W be the set of vertices in these color classes. W.l.o.g. let $i = 1$ and $\mathcal{C}_w = (C_1, \ldots, C_l)$ for some $l \leq m$.

We first show how to refine the color classes in $\mathcal{C}_w$ in a canonical way. We define a (canonical) reflexive, transitive and connex relation $\preceq$ on C_1 such that u and v are in the same orbit of $\mathrm{Aut}(X[W])$ if and only if $u \preceq v$ as well as $v \preceq u$. To define $\preceq$, for $u \in C_1$ consider the set $Z(u)$ of all cycles of color classes starting (and ending) at C_1 such that starting from vertex $u \in C_1$ it is possible to follow this cycle along the edges in E and come back to u. Now define $u \preceq v$ if $Z(u) = Z(v)$ or the lexicographically smallest cycle in $Z(u) \Delta Z(v)$ is in $Z(v)$. Then we can proof the following claim.

Claim. If the three nodes u_1, u_2, u_3 in C_1 are cycle-equivalent (i.e. $Z(u_1) = Z(u_2) = Z(u_3)$), then the permutation $g_1 : u_1 \mapsto u_2 \mapsto u_3$ is extendible to an automorphism of $X[W]$.

Proof of Claim. The permutation g_1 uniquely extends to a permutation $g = (g_1, \ldots, g_l) \in \mathrm{Aut}(X_1) \times \cdots \times \mathrm{Aut}(X_l)$ on $X[W]$, where we extend g successively by the lexicographically smallest color class that is linked to a color class on which g is already defined. If $g \notin \mathrm{Aut}(X[W])$, then there must exist two vertices u, v in two color classes C_i, C_j, respectively, such that

$$\{u, v\} \in E_{ij} \Leftrightarrow \{g_i(u), g_j(v)\} \notin E_{ij}.$$

Now let u' be the vertex in C_j that is linked to u in the spanning tree T along which g has been extended.

In case $u' = v$ and $\{u, v\} \in E_{ij}$ it follows that there is a cycle starting at u following some path in T to $u' = v$ and then back to u. Starting at $g_i(u)$ we reach $g_j(v)$ following the same path through T but proceeding further to C_i we don't come back to $g_i(u)$, implying that u and $g_i(u)$ (and hence also the corresponding vertices in C_1) are not cycle-equivalent.

The other cases are similar. This completes the proof of the claim. ◁

A similar argument shows that if exactly two of the three vertices u_1, u_2, u_3 are cycle-equivalent, then there is an automorphism flipping them. Now, we select any vertex $u \in C_1$ with $u \preceq v$ for all $v \in C_1$ and give it a new color. As in the proof of Theorem 12, this can be done in parallel for all connected components of linked color classes. Running WL^1 again on the graph with the individualized vertices will now refine all whole color classes. Thus we have transformed X into a canonical 2-bounded refinement and hence we can invoke Theorem 12. □

It follows that for the graph classes GA_2 and GA_3 all problems related to GI are complete for L: GA, #GA, #GI, AUT, computing a complete normal form and canonization. Is this also true for the class of 2-bounded hypergraphs (or for GA_4), i.e., is the canonization problem for these graphs solvable in $\mathrm{FL}(\oplus\mathrm{L})$?

Question 16. *What is the complexity of computing a canonizing function for the graph classes $\mathcal{CG}_b$ and $\mathcal{CHG}_b$?*

We remark that the TC^1 upper bound for GI_b given in [4] uses the group theoretic approach to compute a generating set for $\mathrm{Aut}(X)$. Can this approach be adapted to give also an NC upper bound on the canonization problem for graphs with bounded color classes?

Acknowledgements

For helpful conversations and suggestions on this work I'm very grateful to V. Arvind, O. Beyersdorff and O. Verbitsky.

References

1. M. Agrawal and N. Saxena. Automorphisms of finite rings and applications to complexity of problems. In *Proc. 22nd Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2005.

2. C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107(1):3–30, 1993.
3. V. Arvind and J. Köbler. Hypergraph isomorphism testing for bounded color classes. In *Proc. 23rd Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 384–395. Springer-Verlag, 2006.
4. V. Arvind, P. Kurur, and T. Vijayaraghavan. Bounded color multiplicity graph isomorphism is in the #L hierarchy. In *Proc. 20th Annual IEEE Conference on Computational Complexity*, pages 13–27. IEEE Computer Society Press, 2005.
5. V. Arvind and J. Torán. Isomorphism testing: Pespective and open problems. *Bulletin of the European Association of Theoretical Computer Science (BEATCS)*, 86, 2005.
6. L. Babai. Moderately exponential bounds for graph isomorphism. In *Proc. International Symposium on Fundamentals of Computing Theory 81*, volume 117 of *Lecture Notes in Computer Science*, pages 34–50. Springer-Verlag, 1981.
7. L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symposium on Theory of Computing*, pages 421–429. ACM Press, 1985.
8. L. Babai. A Las Vegas-NC algorithm for isomorphism of graphs with bounded multiplicity of eigenvalues. In *Proc. 27th IEEE Symposium on the Foundations of Computer Science*, pages 303–312. IEEE Computer Society Press, 1986.
9. L. Babai. Automorphism groups, isomorphism, reconstruction. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, pages 1447–1540. Elsevier Science Publishers, 1995.
10. L. Babai, D. Grigoryev, and D. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proc. 14th ACM Symposium on Theory of Computing*, pages 310–324. ACM Press, 1982.
11. L. Babai and E. Luks. Canonical labeling of graphs. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 171–183, 1983.
12. L. Babai, E. Luks, and Á. Seress. Permutation groups in NC. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 409–20. ACM Press, 1987.
13. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, second edition, 1995.
14. H. Bodlaender. Polynomial algorithm for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11:631–643, 1990.
15. K. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7(3):273–279, 1978.
16. R. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):27–32, 1987.
17. G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-MOD classes. *Mathematical Systems Theory*, 25:223–237, 1992.
18. S. Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium'97*, volume 1289 of *Lecture Notes in Computer Science*, pages 18–33. Springer-Verlag, 1997.
19. J. Cai, M. Fürer, and N. Immerman. An optimal lower bound for the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.
20. S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.
21. S. Evdokimov, M. Karpinski, and I. Ponomarenko. On a new high dimensional Weisfeiler-Lehman algorithm. *Journal of Algebraic Combinatorics*, 10:29–45, 1999.

22. M. Furst, J. Hopcroft, and E. Luks. Polynomial time algorithms for permutation groups. In *Proc. 21st IEEE Symposium on the Foundations of Computer Science*, pages 36–41. IEEE Computer Society Press, 1980.
23. M. Grohe. Fixed-points logics on planar graphs. In *Proceedings of the 13th Symposium on Logic in Computer Science*, pages 6–15, 1998.
24. M. Grohe. Isomorphism testing for embeddable graphs through definability. In *Proc. 32th ACM Symposium on Theory of Computing*, pages 63–172, 2000.
25. M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In C. Beeri and P. Bunemann, editors, *Proceedings of the 7th Conference on Database Theory*, volume 1540, pages 70–82. Springer-Verlag, 1999.
26. M. Grohe and O. Verbitsky. Testing graph isomorphism in parallel by playing a game. Manuscript, 2006.
27. Y. Gurevich. From invariants to canonization. *Bulletin of the European Association of Theoretical Computer Science (BEATCS)*, 63, 1997.
28. J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs (working paper). In R. Miller and J. Thatcher, editors, *Complexity of computer computations*, pages 131–152. Plenum Press, New York-London, 1972.
29. J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21:549–568620, 1974.
30. J. E. Hopcroft and J. Wong. Linear time algorithm for isomorphisms of planar graphs. *Proc. 6th ACM Symposium on Theory of Computing*, pages 172–184, 1974.
31. N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
32. N. Immerman and E. Lander. Describing graphs: a first order approach to graph canonization. In A. L. Selman, editor, *Complexity Theory Retrospective*, pages 59–81. Springer-Verlag, 1990.
33. J. M. I.S. Filotti. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 236–243. ACM Press, 1980.
34. B. Jenner, J. Köbler, P. McKenzie, and J. Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66:549–566, 2003.
35. J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1993.
36. D. Lichtenstein. Isomorphism for graphs embaddable on the projective plane. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 218–224. ACM Press, 1980.
37. S. Lindell. A logspace algorithm for tree canonization. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 400–404. ACM Press, 1992.
38. E. Luks. Isomorphism of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.
39. E. Luks. Parallel algorithms for permutation groups and graph isomorphism. In *Proc. 27th IEEE Symposium on the Foundations of Computer Science*, pages 292–302. IEEE Computer Society Press, 1986.
40. R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–132, 1979.
41. G. Miller. Isomorphism testing for graphs of bounded genus. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 225–235. ACM Press, 1980.
42. G. Miller and J. Reif. Parallel tree contraction, part 2: Further applications. *SIAM Journal on Computing*, 20:1128–1147, 1991.

43. I. Ponomarenko. The isomorphism problem for classes of graphs that are invariant with respect to contraction (Russian). *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 174:147–177, 1988.
44. O. Reingold. Undirected st-connectivity in log-space. In *Proc. 37th ACM Symposium on Theory of Computing*, pages 376–385. ACM Press, 2005.
45. U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.
46. S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.
47. J. Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004.
48. V. N. Zemlyachenko. Canonical numbering of trees (Russian). *Proc. Seminar on Comb. Anal. at Moscow State University*, 1970.
49. V. N. Zemlyachenko, N. Konienko, and R. I. Tyshkevich. Graph isomorphism problem (Russian). *The Theory of Computation I, Notes Sci. Sem. LOMI 118*, 1982.