

# On Graph Isomorphism for Restricted Graph Classes<sup>\*</sup>

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  $\text{Aut}(X)$  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 [Boo78]. 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 [AS05,AT05].

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

---

<sup>\*</sup> 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 [Mat79]. 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 [Tod91]. 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 [Bab85]. As a consequence, GI is not NP-complete unless the polynomial hierarchy collapses to its second level [BHZ87,Sch88].

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 [Luk82] yields the fastest known general graph isomorphism algorithm due to Babai, Luks, and Zemyachenko [Bab81,BL83,ZKT82]. The strongest known hardness result due to Torán [Tor04] 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. [Coo85]). 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  $L(\#L)$  [ÅJ93,BDHM92].

The first significant complexity result for restricted graph classes is the linear time canonization algorithm for trees, designed by Hopcroft and Tarjan [HT74], and independently by Zemyachenko [Zem70]. Miller and Reif [MR91] later gave an NC algorithm for tree canonization, based on tree contraction methods. Then Lindell came up with a logspace algorithm for tree canonization [Lin92]. As shown in [JKMT03], 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 [Bus97] located the canonization problem even in  $NC^1$  (which is also optimal [JKMT03]).

Shortly after the linear time canonization algorithm for trees was found, Hopcroft, Tarjan and Wong designed a linear time canonization algorithm for planar graphs [HT72,HW74]. 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 [Lic80,Mil80,FM80]. In 1991, Miller and Reif [MR91] 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 [FHL80] 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  $\text{Aut}(X)$  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  $\text{Aut}(X)$  in polynomial time for graphs of bounded degree [Luk82]. 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 [ZKT82] (see also [Bab81]) 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 [BL83]). This is the fastest algorithm known for the unrestricted graph isomorphism problem. In [BL83] 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 [Luk86] gave a remarkable NC algorithm for the bounded color class case. Building on [Luk86], 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  $\text{GI}_b$ ) is in the  $\text{Mod}_k\text{L}$  hierarchy (and hence in  $\text{TC}^1$ ), where the constant  $k$  and the level of the hierarchy depend on  $b$  [AKV05]. Prior to this result, Torán showed that  $\text{GI}_{b^2}$  is hard for the logspace counting class  $\text{Mod}_b\text{L}$  [Tor04]. Torán's lower bound has been extended in [AKV05] where it is shown that for each level in the  $\text{Mod}_k\text{L}$  hierarchy there is a constant  $b$  such that  $\text{GI}_b$  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 [BGM82]. 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 [BLS87,Bab86,Luk86]. 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 [Pon88]. In 1990, Bodlaender gave a polynomial-time GI algorithm for graphs of bounded treewidth [Bod90]. 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 [GV06] improved Bodlaender's upper bound by showing that GI for graphs of bounded treewidth is in  $\text{TC}^1$ . 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  $\text{WL}^k$  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  $\text{TC}^1$  algorithm.

As a second ingredient of the proof, Grohe and Verbitsky show that for  $r = O(k \log n)$ , the  $r$ -round  $\text{WL}^{4k+3}$  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  $\text{TC}^1$ ) canonization?*

As Grohe and Verbitsky use the WL algorithm to solve GI for graphs with bounded treewidth, it follows that these graphs have a  $\text{TC}^1$  computable complete normal form (also called invariant). Although, as shown by Gurevich, canonization is polynomial-time reducible to computing a complete normal form [Gur97], 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 [IL90, 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 separate 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 [KST93] or to any textbook on complexity like [BDG95].

We denote the *symmetric group* of all permutations on a set  $A$  by  $\text{Sym}(A)$  and by  $S_n$  in case  $A = \{1, \dots, n\}$ . Let  $G$  be a subgroup of  $\text{Sym}(A)$  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  $\{1, \dots, n\}$ . 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  $c : V \rightarrow [m]$ . We represent colored hypergraphs as triples  $X = (V, E, \mathcal{C})$ , where  $\mathcal{C} = (C_1, \dots, C_m)$  is the color partition induced by  $c$ , i.e.,  $C_i = \{u \in V \mid c(u) = i\}$ .

A partition  $\mathcal{C}' = (C'_1, \dots, C'_k)$  is a *refinement* of the partition  $\mathcal{C} = (C_1, \dots, C_m)$ , if each color class  $C_i$  of  $\mathcal{C}$  is the union  $C'_i \cup \dots \cup C'_{j_{i+1}-1}$  of one or more consecutive color classes from  $\mathcal{C}'$ . 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  $U = \{u_1, \dots, u_k\}$  of  $V$  by

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

$g$  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  $\text{Aut}(X)$  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  $\text{HGI}_b$  consists of deciding whether two given  $b$ -bounded hypergraphs  $X$  and  $Y$  are isomorphic ( $\text{GI}_b$  denotes the restriction of this problem to graphs). A related problem is the automorphism problem  $\text{HGA}_b$  ( $\text{GA}_b$ ) of deciding if a given  $b$ -bounded hypergraph (respectively, graph) has a non-trivial automorphism. For uncolored (hyper)graphs  $X = (V, E)$  we denote these problems by  $\text{HGI}$ ,  $\text{GI}$ ,  $\text{HGA}$  and  $\text{GA}$ , 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  $\mathcal{G} \subseteq \mathcal{CHG}$  be a

graph class and let  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  be a function. We say that  $f$  computes a *normal form* for  $\mathcal{G}$ , if

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

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

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

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

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

is called a *canonization* for  $\mathcal{G}$ . Note that a canonization for  $\mathcal{G}$  is also a complete normal form for  $\mathcal{G}$ . 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  $f(X)$  (in fact, it seems that  $g$  provides more information than  $f(X)$ ). We call any such  $g$  a *canonical relabeling* of  $X$  (w.r.t.  $f$ ). Another way to represent  $g$  is in terms of the *canonical refinement*  $\mathcal{C}_g = (\{g(1)\}, \dots, \{g(n)\})$  of  $\mathcal{C}$ . Here we assume w.l.o.g. that the coloring  $c$  of the input graph  $X = ([n], E, \mathcal{C})$  is monotone, i.e.,  $c(i) \leq c(i+1)$  for  $i = 1, \dots, n-1$ .

## 2.4 The Weisfeiler-Lehman algorithm

For the history of this approach to GI we refer the reader to [Bab95,CFI92,EKP99]. 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  $\mathcal{C} = (C_1, \dots, C_m)$ , 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  $\mathcal{C}^r$  (together with the tables retained at each round). Note that as long as  $\mathcal{C}^{i+1}$  is a proper refinement of  $\mathcal{C}^i$ , the number of colors increases. Hence, the coloring stabilizes after at most  $n$  rounds, i.e.  $\mathcal{C}^{s+1} = \mathcal{C}^s$  for some  $s < n$ . We call  $\mathcal{C}^s$  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 [CFI92,GV06] 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  $Y \not\cong X$ . It is clear that the  $r$ -round  $WL^k$  computes a complete normal form on a graph class  $\mathcal{G}$ , provided that it works correctly for each graph  $X \in \mathcal{G}$  (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 [IL90]. Thus there was some hope that a low dimensional WL algorithm may work correctly on all graphs. However, in 1990 Cai, Fürer and Immerman [CFI92] 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 [Gro98], graphs of bounded genus [Gro00], and graphs of bounded treewidth [GM99].

### 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  $WL^{o(n)}$ , Cai, Fürer and Immerman used a graph gadget that originally appeared in [Imm81]. This gadget has also been used by Torán in a significant manner to show that GI and GA are hard for various subclasses of  $TC^1$  [Tor04]. Here we use a hypergraph variant of this gadget to show that for any prime  $p$ ,  $HGA_p$  is hard for  $\text{Mod}_p\text{L}$ . The proof given here simplifies a proof of a similar result in [AK06].

It is well-known that the following problem is  $\text{Mod}_p\text{L}$  complete (cf. [BDHM92]). Given a homogenous system

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

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  $\text{Mod}_p\text{L}$  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

$$V = \bigcup_{j=1}^n V_j, \quad E = \bigcup_{j=1}^n Z_j \cup \bigcup_{i=0}^k E_i \quad \text{and} \quad \mathcal{C} = (C_1, C'_1, C''_1, \dots, C_n, C'_n, C''_n),$$

where

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

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

For any vector  $\bar{x} = (x_1, \dots, x_n)$  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  $\bar{x} = (x_1, \dots, x_n)$  of the  $i$ -th equation in (1) the hyperedge  $e(\bar{x}|_{S_i}) = \{u_{x_j}^j \mid j \in S_i\}$ ,  $E_i$  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  $p = 2$  we can simplify  $X$  to the graph  $\hat{X} = X[C_1 \cup \dots \cup C_n]$  since in this case the groups  $\text{Aut}(\hat{X}_j)$  are cyclic anyway. Figure 1 shows the graph  $\hat{X}$  corresponding to the equation  $x_1 + x_2 - x_3 = 0$  over  $\mathbb{Z}_3$ .

Now it is easy to see that for each  $i \in [k]$  the automorphism group  $\text{Aut}(Y_i)$  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

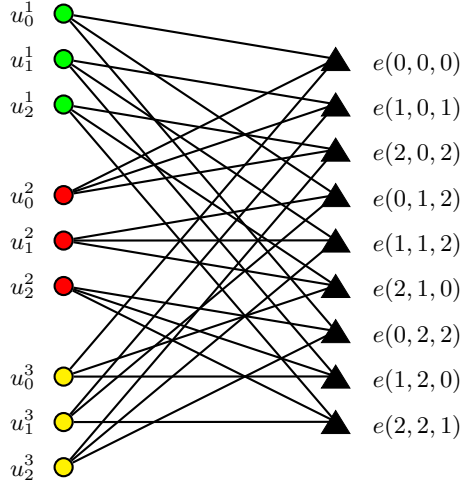
$$\text{Aut}(Y_i) = \{(g_{x_1}^1, g_{x_2}^2, g_{x_3}^3) \mid x_1 + x_2 - x_3 = 0\}.$$

Hence, a permutation  $g = (g_{x_1}^1, \dots, g_{x_n}^n) \in \text{Aut}(X_1) \times \dots \times \text{Aut}(X_n)$  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

$$\text{Aut}(X) = \{(g_{x_1}^1, \dots, g_{x_n}^n) \mid (x_1, \dots, x_n) \text{ is a solution of (1)}\}.$$

This shows that  $X \in \text{HGA}_p$  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  $\text{AC}^0$ , it follows that for any prime  $q \leq p$ ,  $\text{HGA}_p$  is hard for the class  $\text{Mod}_q\text{L}$  under  $\text{AC}^0$  many-one reductions.

Moreover, since  $\text{HGA}_p$  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



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

colors in common) and since any set in the class  $\text{Mod}_m\mathbb{L}$  can be represented as the union  $A_1, \dots, A_k$  of sets  $A_i$  in  $\text{Mod}_{p_i}\mathbb{L}$ , where  $p_1, \dots, p_k$  are the prime factors of  $m$  [BDHM92], it immediately follows that  $\text{HGA}_p$  is even hard for  $\text{Mod}_{p!}\mathbb{L}$ . Since the orbit size of the hyperedges in the reduced hypergraph is bounded by  $p^2$ , we also get that  $\text{GA}_{p^2}$  is  $\text{Mod}_{p!}\mathbb{L}$  hard.

**Theorem 3.**  $\text{HGA}_p$  and  $\text{GA}_{p^2}$  are hard for  $\text{Mod}_{p!}\mathbb{L}$ .

In [AK06] it is shown that  $\text{GA}_4$  (as well as  $\text{GA}_5$  and  $\text{HGA}_2$ ) in fact is complete for the class  $\text{Mod}_2\mathbb{L} = \oplus\mathbb{L}$ . The best known upper bound for  $\text{HGA}_b$ ,  $b > 2$ , is P [AK06]. We remark that if the hyperedges are all of constant size, i.e.,  $\|e\| \leq k$  for all  $e \in E$ , then  $\text{HGA}_b$  is reducible to  $\text{GA}_{b^k}$  for  $b' = b^k$  which is known to be in  $\text{TC}^1$  [Luk86, AKV05]. However, when hyperedges are of unbounded size, it is not clear whether  $\text{HGA}_b$  is reducible to  $\text{GA}_{b'}$  for any constant  $b'$ .

**Question 4.** Is  $\text{HGA}_b$  in  $\text{NC}$  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  $\text{HGA}_b$  (or  $\text{HGI}_b$ ) 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  $\text{HGA}_b$  (or  $\text{HGI}_b$ ) is NL hard?

## 4 Logspace canonization of 3-bounded graphs

In this section we improve the result from [JKMT03] that GI for 2-bounded as well as for 3-bounded graphs is equivalent to undirected graph reachability

(and therefore complete for L [Rei05]). We first describe a logspace canonization algorithm for 2-bounded graphs. This algorithm performs a 1-round WL<sup>1</sup> 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 [AK06] (in preparation).

Let  $X = (V, E, \mathcal{C})$  be a  $b$ -bounded graph and let  $\mathcal{C} = (C_1, \dots, C_m)$ . 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 [AK06].

**Lemma 6.** [AK06] *For any directly linked pair  $C_i, C_j$  of color classes there is a bijection  $\pi_{ij} : \text{Sym}(C_i) \rightarrow \text{Sym}(C_j)$  such that for any automorphism  $g = (g_i, g_j) \in \text{Aut}(X_{ij})$  it holds that  $g_j = \pi_{ij}(g_i)$ .*

Let  $G_i$  be the intersection of  $\text{Aut}(X_i)$  with the projections of  $\text{Aut}(X_{ij})$  on  $C_i$  for all  $j \neq i$ . Any subgroup of the symmetric group  $\text{Sym}(C_i)$  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.** [AK06] *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.** [AK06] *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  $\text{SL} = \text{L}$  [Rei05], this can be done in deterministic logspace.  $\square$

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  $\text{Sym}(C_i)$ . 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  $\text{WL}^1$  will never refine any of the whole color classes in  $X$ . Let  $W$  be the union of all whole color classes. Then  $\text{Aut}(X[W])$  is computable in logspace.

**Lemma 10.** [JKMT03,AK06] *A generating set for  $\text{Aut}(X[W])$  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  $g_i \in \text{Aut}(X_i)$ . 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  $\text{Aut}(X[W])$  and since  $\text{SL} = \text{L}$  [Rei05], this can be done in deterministic logspace.  $\square$

It is easy to see that computing the  $\text{WL}^1$ -stable coloring of a 2-bounded graph is hard for NL. Nevertheless, we can implement a canonization algorithm for all 2-bounded graphs in logspace. For this, we first show that the first round of  $\text{WL}^1$  can be simulated in logspace.

**Lemma 11.** *For any graph  $X$  in  $\mathcal{CG}_2$ , the coloring  $\mathcal{C}^1$  of the 1-round  $\text{WL}^1$  algorithm on input  $X$  is computable in FL.*

*Proof.* Let  $X = (V, E, \mathcal{C})$  be a 2-bounded graph with coloring  $\mathcal{C} = (C_1, \dots, C_m)$ . The only way that a color class  $C_i = \{a, b\}$  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 in the first round and they are clearly computable in logspace.  $\square$

For the sake of completeness we also mention that for 2-bounded graphs, the  $WL^1$ -stable coloring is computable in logspace relative to an NL oracle.

**Theorem 12.** *For any graph in  $\mathcal{CG}_2$ , the  $WL^1$ -stable coloring is computable in  $FL(NL)$ .*

*Proof (sketch).* Let  $X = (V, E, \mathcal{C})$  be a 2-bounded graph with coloring  $\mathcal{C} = (C_1, \dots, C_m)$ . As shown in Lemma 11 the coloring  $\mathcal{C}^1$  of the 1-round  $WL^1$  algorithm on input  $X$  is 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  $(C_j, \dots, C_i)$  be the lexicographically smallest chain of directly linked color classes from some directly split color class  $C_j$  to  $C_i$ . Then  $WL^1$  transposes the refinement of  $C_j$  to  $C_i$  via the chain  $(C_j, \dots, C_i)$ . Clearly, these refinements are computable in logspace with the help of an NL oracle (here, the NL oracle is used for computing the lexicographically smallest chain between two color classes). Finally, observe that the whole color classes never get refined by  $WL^1$ . In fact, they form orbits in  $\text{Aut}(X)$ .  $\square$

**Theorem 13.**  *$\mathcal{CG}_2$  admits a logspace canonization.*

*Proof (sketch).* Let  $X = (V, E, \mathcal{C})$  be a 2-bounded graph with coloring  $\mathcal{C} = (C_1, \dots, C_m)$ . First we use Lemma 11 to compute the coloring  $\mathcal{C}^1$  of the 1-round  $WL^1$  algorithm on input  $X$  in logspace. If  $\mathcal{C}^1$  assigns unique colors to all vertices, then a canonical labeling is determined.

Otherwise, for each connected component  $\mathcal{D}$  of directly linked color classes (using direct links as the edge relation) the algorithm computes a spanning tree  $\mathcal{T}$  for  $\mathcal{D}$  (in a canonical way). If  $\mathcal{D}$  contains at least one directly split color class, then the refinement of the lexicographically smallest directly split color class in  $\mathcal{D}$  is propagated along  $\mathcal{T}$  to all the other color classes in  $\mathcal{D}$ . If  $\mathcal{D}$  consists only of whole color classes, then the algorithm arbitrarily refines any color class  $C$  of  $\mathcal{D}$  and propagates this refinement to the other color classes in  $\mathcal{D}$ . Since  $\mathcal{D}$  contains only whole color classes, flipping all color classes of  $\mathcal{D}$  is an automorphism of  $X$ , and hence, the resulting graph will neither depend on the choice of  $C$  nor on the way how the coloring  $\mathcal{C}$  is refined.  $\square$

We notice that similar to the proof above it can be shown that the canonization version of  $WL^1$  (as proposed in [IL90, Theorem 1.9.4]) succeeds on the class  $\mathcal{CG}_2$  (despite the fact that  $WL^1$  does not work correctly on  $\mathcal{CG}_2$  [IL90, Corollary 1.6.2]).

**Question 14.** *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 12 it can be shown that also for graphs in  $\mathcal{CG}_3$  the  $WL^1$ -stable coloring is computable in  $FL(NL)$  but it is not clear whether this generalizes.

**Question 15.** *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  $\mathcal{CG}_3$  [IL90]. Here we give a logspace canonization algorithm for this class.

**Theorem 16.**  *$\mathcal{CG}_3$  admits a logspace canonization.*

*Proof (sketch).* Let  $X = (V, E, C)$  be a 3-bounded graph with coloring  $C = (C_1, \dots, C_m)$ . Similarly to the proof of Theorem 12 it can be shown that the coloring  $C^1$  of the 1-round  $WL^1$  algorithm on input  $X$  is computable in logspace. If  $X' = (V, E, C^1)$  is 2-bounded, then the algorithm of Theorem 13 can be used to compute a canonical form  $CF(X')$  for  $X'$  which will also be a canonical form for  $X$ .

Otherwise, we refine  $X'$  in a canonical way into a 2-bounded graph. For each connected component  $\mathcal{D}$  of directly linked color classes of size 3 (using direct links as the edge relation) the algorithm computes a spanning tree  $\mathcal{T}$  for  $\mathcal{D}$ . If  $\mathcal{D}$  contains at least one directly split color class, then the refinement of the lexicographically smallest directly split color class in  $\mathcal{D}$  is propagated along  $\mathcal{T}$  to all the other color classes in  $\mathcal{D}$ . If  $\mathcal{D}$  consists only of whole color classes, then for each vertex  $v_i$  in the lexicographically smallest color class  $C = \{v_1, v_2, v_3\}$  of  $\mathcal{D}$ , the algorithm individualizes  $v_i$  in  $C$  (i.e., refines  $C$  to the two color classes  $\{v_i\}$  and  $C - \{v_i\}$ ) and propagates this refinement of  $C$  to the other color classes of  $\mathcal{D}$  along the edges of  $\mathcal{T}$ . This results in three 2-bounded graphs  $\mathcal{D}_i$ ,  $i = 1, 2, 3$ . In order to refine  $\mathcal{D}$  in a canonical way to either  $\mathcal{D}_1$ ,  $\mathcal{D}_2$  or  $\mathcal{D}_3$ , the algorithm computes the canons  $CF(\mathcal{D}_i)$  (by using the algorithm of Theorem 13) and refines  $\mathcal{D}$  to the  $\mathcal{D}_i$  with the smallest canonical form. Thus, in each case, we have transformed  $X'$  into a 2-bounded graph in a canonical way and hence we can invoke Theorem 13.  $\square$

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  $FL(\oplus L)$ ?

**Question 17.** *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 [AKV05] uses the group theoretic approach to compute a generating set for  $Aut(X)$ . 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, T. Tantau and O. Verbitsky.

## References

- [ÀJ93] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107(1):3–30, 1993.
- [AK06] 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.
- [AKV05] 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.
- [AS05] 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.
- [AT05] V. Arvind and J. Torán. Isomorphism testing: Perspective and open problems. *Bulletin of the European Association of Theoretical Computer Science (BEATCS)*, 86:66–84, 2005.
- [Bab81] 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.
- [Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symposium on Theory of Computing*, pages 421–429. ACM Press, 1985.
- [Bab86] 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.
- [Bab95] 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.
- [BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, second edition, 1995.
- [BDHM92] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of logspace-MOD classes. *Mathematical Systems Theory*, 25:223–237, 1992.
- [BGM82] L. Babai, D.Y. Grigoryev, and D.M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proc. 14th ACM Symposium on Theory of Computing*, pages 310–324. ACM Press, 1982.
- [BHZ87] R. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):27–32, 1987.

- [BL83] L. Babai and E. Luks. Canonical labeling of graphs. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 171–183, 1983.
- [BLS87] L. Babai, E.M. Luks, and Á. Seress. Permutation groups in NC. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 409–20. ACM Press, 1987.
- [Bod90] H. Bodlaender. Polynomial algorithm for graph isomorphism and chromatic index on partial  $k$ -trees. *Journal of Algorithms*, 11(4):631–643, 1990.
- [Boo78] K.S. Booth. Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM Journal on Computing*, 7(3):273–279, 1978.
- [Bus97] 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.
- [CFI92] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound for the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.
- [Coo85] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.
- [EKP99] S. Evdokimov, M. Karpinski, and I. Ponomarenko. On a new high dimensional Weisfeiler-Lehman algorithm. *Journal of Algebraic Combinatorics*, 10:29–45, 1999.
- [FHL80] 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.
- [FM80] I. S. Filotti and J. N. Mayer. 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.
- [GM99] 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.
- [Gro98] M. Grohe. Fixed-points logics on planar graphs. In *Proceedings of the 13th Symposium on Logic in Computer Science*, pages 6–15, 1998.
- [Gro00] M. Grohe. Isomorphism testing for embeddable graphs through definability. In *Proc. 32th ACM Symposium on Theory of Computing*, pages 63–72, 2000.
- [Gur97] Yuri Gurevich. From invariants to canonization. *Bulletin of the European Association of Theoretical Computer Science (BEATCS)*, 63:115–119, 1997.
- [GV06] M. Grohe and O. Verbitsky. Testing graph isomorphism in parallel by playing a game. In *Proc. 33rd International Colloquium on Automata, Languages, and Programming*, volume 4051 of *Lecture Notes in Computer Science*, pages 3–14. Springer-Verlag, 2006.
- [HT72] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs (working paper). In R.E. Miller and J.W. Thatcher, editors, *Complexity of computer computations*, pages 131–152. Plenum Press, New York-London, 1972.
- [HT74] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21:549–568620, 1974.
- [HW74] 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.

- [IL90] 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.
- [Imm81] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
- [JKMT03] B. Jenner, J. Köbler, P. McKenzie, and J. Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66:549–566, 2003.
- [KST93] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1993.
- [Lic80] 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.
- [Lin92] S. Lindell. A logspace algorithm for tree canonization. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 400–404. ACM Press, 1992.
- [Luk82] E. Luks. Isomorphism of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.
- [Luk86] 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.
- [Mat79] R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8:131–132, 1979.
- [Mil80] G. L. Miller. Isomorphism testing for graphs of bounded genus. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 225–235. ACM Press, 1980.
- [MR91] G. L. Miller and J. H. Reif. Parallel tree contraction. Part 2: Further applications. *SIAM Journal on Computing*, 20(6):1128–1147, 1991.
- [Pon88] I. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 174(Teor. Slozhn. Vychisl. 3):147–177, 1988. In Russian. English translation in: *Journal of Mathematical Sciences* 55(2):1621–1643 (1991).
- [Rei05] O. Reingold. Undirected st-connectivity in log-space. In *Proc. 37th ACM Symposium on Theory of Computing*, pages 376–385. ACM Press, 2005.
- [Sch88] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.
- [Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.
- [Tor04] J. Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004.
- [Zem70] V. N. Zemlyachenko. Canonical numbering of trees (Russian). *Proc. Seminar on Comb. Anal. at Moscow State University*, 1970.
- [ZKT82] V. N. Zemlyachenko, N. Konienko, and R. I. Tyshkevich. Graph isomorphism problem (Russian). *The Theory of Computation I, Notes Sci. Sem. LOMI 118*, 1982.