Electronic Colloquium on Computational Complexity, Report No. 93 (2009)

Colored Hypergraph Isomorphism is Fixed Parameter Tractable

V. Arvind[1], Bireswar Das[1], Johannes Köbler[2], and Seinosuke Toda[3]

[1] The Institute of Mathematical Sciences, Chennai 600 113, India, {arvind,bireswar}@imsc.res.in
[2] Institut für Informatik, Humboldt Universität zu Berlin, Germany, koebler@informatik.hu-berlin.de
[3] Nihon University, Tokyo, Japan, toda@cssa.chs.nihon-u.ac.jp

Abstract. We describe a fixed parameter tractable (fpt) algorithm for Colored Hypergraph Isomorphism which has running time $b!2^{O(b)}N^{O(1)}$, where the parameter b is the maximum size of the color classes of the given hypergraphs and N is the input size. We also describe fpt algorithms for certain permutation group problems that are used as subroutines in our algorithm.

Topic classification: Fixed parameter tractability, fpt algorithms, graph isomorphism, computational complexity.

1 Introduction

A *hypergraph* is an ordered pair $X = (V, E)$ where V is the vertex set and $E \subseteq 2^V$ is the edge set. Two hypergraphs $X = (V, E)$ and $X' = (V', E')$ are said to be *isomorphic*, denoted $X \cong X'$, if there is a bijection $\varphi : V \to V'$ such that for all $e = \{u_1, \cdots, u_l\} \subseteq V$, $e \in E$ if and only if $\varphi(e) = \{\varphi(u_1), \cdots, \varphi(u_l)\} \in E'$. Given two hypergraphs X and X' the decision problem Hypergraph Isomorphism (HI) asks whether $X \cong X'$. Graph Isomorphism (GI) is obviously polynomial-time reducible to HI. Conversely, HI is also known to be polynomial-time reducible to GI: Given a pair of hypergraphs $X = (V, E)$ and $X' = (V', E')$ as instance for HI, the reduced instance of GI consists of two corresponding bipartite graphs Y and Y' defined as follows. The graph Y has vertex set $V \uplus E$ and edge set $E(Y) = \{\{v, e\} \mid v \in V, e \in E \text{ and } v \in e\}$, whereas Y' is defined accordingly. Here, $C \uplus D$ denotes the disjoint union of the sets C and D. It is easy to verify that $Y \cong Y'$ if and only if $X \cong X'$.

The input to Colored Hypergraph Isomorphism (CHI) is a pair of hypergraphs $X = (V, E)$ and $X' = (V', E')$ together with partitions $V = C_1 \uplus \cdots \uplus C_m$ and $V' = C'_1 \uplus \cdots \uplus C'_m$ of their vertex sets into *color classes* C_i and C'_i, respectively. The problem is to decide if there is an isomorphism φ that preserves the colors (meaning that $v \in C_i \Leftrightarrow \varphi(v) \in C'_i$). Colored Graph Isomorphism (CGI) is the analogous problem where instead of hypergraphs we have graphs as inputs.

Furst, Hopcroft and Luks gave an fpt algorithm for CGI [8] with running time $2^{O(b^2)}n^{O(1)}$, where the parameter b is the maximum size of the color classes and n is the number of vertices of the input graphs. Although HI is polynomial time

ISSN 1433-8092

many-one reducible to GI, the reduction we described above does not impose any bound on the size of the color classes of the bipartite graphs Y and Y'. More specifically, if the color classes of the hypergraphs X and X' have maximum size b, then the vertices of the graphs Y and Y' that correspond to the edges of X and X' do not get partitioned into color classes of size bounded by any function of b (since in general, the hyperedges are of *nonconstant* size). Thus, the fpt algorithm for CGI cannot be combined with the above reduction to get an fpt algorithm for CHI. Moreover, even if b is bounded by a constant (say 2), the color classes in the resulting bipartite graphs Y and Y' can have size up to $2^{n/2}$ and hence, this approach would only give a double exponential time algorithm for CHI.

However, an algorithm for CHI with a running time of the form $N^{O(b)}$ was shown in [1], where b bounds the size of the color classes of the given hypergraphs and N is the input size. Hence, if b is bounded by a constant, we have a polynomial-time algorithm for CHI. This algorithm basically applies Luks' seminal result [9] showing that the set stabilizer problem with respect to a class of permutation groups Γ_d can be solved in time $n^{O(d)}$. Recall that a finite permutation group $G < S_n$ is said to belong to the class Γ_d if every *nonabelian* composition factor of G is isomorphic to some subgroup of the symmetric group S_d.

Parametrized complexity and isomorphism testing

Parametrized complexity is a fundamental strategy for coping with intractability. Pioneered by Downey and Fellows in [4], it is a flourishing area of research (see, e.g. the monographs [5, 7]). Fixed parameter tractability provides a notion of feasible computation less restrictive than polynomial time. It provides a theoretical basis for the design of new algorithms that are efficient and practically useful for small parameter values. We quickly recall the rudiments of this theory relevant for the present paper. For more details see [5, 7].

Computational problems often have inputs consisting of two or more parts where some of these parts typically take only small values. For example, an input instance of the vertex cover problem is (G, k), and the task is to determine if the graph G has a vertex cover of size k. A similar example is the k-clique problem where again an input instance is a pair (G, k) and the problem is to test if the graph G has a clique of size k. For such problems an exhaustive search will take time $O(n^k)$, where n is the number of vertices in G. However, a finer classification is possible. The vertex cover problem has an algorithm running in time $2^k n^{O(1)}$ (even in time $O(1.2738^k + kn)$ [3]), whereas no algorithm is known for the k-clique problem of running time $O(n^{o(k)})$. Thus, if the parameter k is such that $k \ll n$, then we have a faster algorithm for the k-vertex cover problem than is known for the k-clique problem.

Parametrized complexity theory deals with the study and design of algorithms that have a running time of the form $f(b)n^{O(1)}$ where n is the input size, b is the parameter and f is a computable function. If a problem is solvable by such an algorithm it is called *fixed parameter tractable* (fpt). For example, the vertex cover problem has an fpt algorithm, whereas no fpt algorithm is known for the k-clique problem.

Since no polynomial-time algorithm for GI is known, one approach is to design fpt isomorphism testing algorithms with respect to natural graph parameters. For example, the algorithm of Furst et al [8] mentioned above is fpt with respect to the color class size. For isomorphism testing of graphs with eigenvalue multiplicity bounded by k, Evdokimov and Ponomarenko have designed a highly nontrivial fpt algorithm with running time $k^{O(k)}n^{O(1)}$ [6].

Apart from this, fpt algorithms have also been designed with respect to the parameters tree-distance width [16] and the size of the simplicial components of the input graphs[15].

On the other hand, if we use the maximum degree [9], or the treewidth [2], or the genus [12] of the input graphs as parameter b, the best known isomorphism testing algorithms have a worst-case running time bound $n^{O(b)}$. It is an interesting open question if GI has an fpt algorithm with respect to any of these three parameters.

The results

In this paper we present an fpt algorithm for Colored Hypergraph Isomorphism that runs in time $b!2^{O(b)}N^{O(1)}$, where b is the maximum size of the color classes and N is the input size.

We use as subroutines fpt algorithms for certain permutation group problems parametrized by the size of the largest *color class* of the group. More specifically, following Luks' method [10], we design fpt algorithms for the functional problems Colered Coset Intersection and Colered Set Transporter (formal definitions of these problems are given in Section 3).

While the parametrized complexity of permutation group problems, for different parameters, is certainly interesting in its own right, it could also be applicable to GI. For example, an fpt algorithm for Set Transporter with respect to groups in Γ_d (with d as parameter) would result in an fpt algorithm for testing isomorphism of graphs of maximum degree d.

2 Preliminaries

While designing the fpt algorithm for Colored Hypergraph Isomorphism we will have to handle hypergraphs (V, E) with multiple hyperedges (i.e., E is a multiset). Two multihypergraphs $X = (V, E)$ and $X' = (V', E')$ are isomorphic via an isomorphism φ if any hyperedge $e \in E$ has the same multiplicity as $\varphi(e)$ in E'.

Next we recall some basic group theory. Let G be a finite group and let Ω be a finite nonempty set. The *action* of the group G on Ω is defined by a map $\alpha : \Omega \times G \to \Omega$ such that for all $x \in \Omega$, (i) $\alpha(x, id) = x$, i.e., the identity $id \in G$ fixes each $x \in \Omega$, and (ii) $\alpha(\alpha(x, g), h) = \alpha(x, gh)$ for all $g, h \in G$. We write x^g instead of $\alpha(x, g)$ when the group action is clear from the context.

For $x \in \Omega$, its *G-orbit* is the set $x^G = \{y | y \in X, y = x^g \text{ for some } g \in G\}$. When the group is clear from the context, we call x^G the *orbit* of x. Notice that the orbits form a partition of Ω.

We write $H \leq G$ when H is a subgroup of G. The *symmetric group* on a finite set Ω consisting of all permutations on Ω is denoted by $\mathrm{Sym}(\Omega)$. If $\Omega = [n] = \{1, \cdots, n\}$, we write S_n instead of $\mathrm{Sym}([n])$. A *finite permutation group* G is a subgroup of $\mathrm{Sym}(\Omega)$ for some finite set Ω.

The permutation group *generated* by a subset $S \subseteq \mathrm{Sym}(\Omega)$ is the smallest subgroup of $\mathrm{Sym}(\Omega)$ containing S and is denoted by $\langle S \rangle$. Each element of the group $\langle S \rangle$ is expressible as a product of elements of S.

The subgroup $G^{(i)}$ of $G \leq S_n$ that fixes each of $\{1, \ldots, i\}$ is called a *pointwise stabilizer* of G. These subgroups form a tower

$$G = G^{(0)} \geq G^{(1)} \geq G^{(2)} \geq \cdots \geq G^{(n-1)} = \{id\}.$$

We notice that by the orbit-stabilizer lemma, the index $[G^{(i-1)} : G^{(i)}]$ is at most n. For each i, let R_i be a set of complete and distinct coset representatives of $G^{(i)}$ in $G^{(i-1)}$. Then $\bigcup_{i=1}^{n-1} R_i$ generates G and is called a *strong generating set* for G. Given a permutation $\pi \in G$ it is easy to check if $\pi \in G^{(i)}$. It is also easy to check if two permutations $\pi, \sigma \in G^{(i)}$ are in the same coset of $G^{(i+1)}$ in $G^{(i)}$. We just have to test if $\pi^{-1}\sigma \in G^{(i+1)}$. These observations yield a polynomial-time algorithm [13, 14, 8] for computing a strong generating set of a permutation group G. This algorithm can also be used to test in polynomial time if $g \in S_n$ is in the group $\langle S \rangle \leq S_n$.

In some applications there is an efficient algorithm for testing membership in a subgroup H of G, where $G \leq S_n$ is given by a generating set but no generating set for H is given. By [13, 14, 8] we can efficiently compute a generating set for H provided that its index in G is polynomially bounded.

Theorem 1 (Schreier Generators). *Let $G = \langle S \rangle \leq S_n$ and $H \leq G$. Then for any set R of coset representatives of H in G, the set $B = \{r'xr^{-1} \mid r, r' \in R, x \in S\} \cap H$ generates H. The generators in B are called* Schreier generators.

The proof of Theorem 1 also provides an algorithm for computing a suitable set R of coset representatives by making $m^2|S|$ tests of membership in H, where $m = [G : H]$. Though the set B of Schreier generators for H can be of size polynomial in m, it is possible to convert it to a strong generating set for H of size $O(n^2)$ [13, 14, 8].

For a permutation $\pi \in \mathrm{Sym}(\Omega)$ and a subset $C \subseteq \Omega$ we use C^π to denote the set $\{x^\pi \mid x \in C\}$. For a set S of permutations, C is called *S-stable* if $C^\pi = C$ for all $\pi \in S$. For a permutation group $G \leq \mathrm{Sym}(\Omega)$, the *stabilizer subgroup* of G is defined as $G_C = \{\pi \in G \mid C^\pi = C\}$.

3 Permutation group problems

In this section we describe fpt algorithms for some permutation group problems with respect to the color class bound as parameter. We say that a group $G \leq \mathrm{Sym}(\Omega)$ has *color class bound* b if Ω can be partitioned into *color classes* $C_1 \uplus \cdots \uplus C_m$ of size $|C_i| \leq b$ such that each C_i is G-stable (equivalently, the maximum orbit size of G is bounded by b). Since the orbits of G can be computed in polynomial time (where G is given by a generating set S), we can also check in polynomial time if G has color class bound b.

First we recall the definition of the set transporter problem for general permutation groups.

Set Transporter (ST)

Input: A generating set for a group $G \leq \mathrm{Sym}(\Omega)$, a permutation $z \in \mathrm{Sym}(\Omega)$ and subsets $\Pi, \Pi' \subseteq \Omega$.
Output: A description of the set $(Gz)_{\Pi \to \Pi'} = \{x \in Gz \mid \Pi^x = \Pi'\}$ which is either empty or a coset of G_Π.

For general permutation groups it is known that GI polynomial-time reduces to ST. Here, we are particularly interested in the following parametrized version of Set Transporter.

Colored Set Transporter (CST)

Input: A generating set for a group $G \leq \mathrm{Sym}(\Omega)$, a permutation $z \in \mathrm{Sym}(\Omega)$, subsets $\Pi_1, \ldots, \Pi_m, \Pi'_1, \ldots, \Pi'_m \subseteq \Omega$ and a partition $\Omega = C_1 \uplus \cdots \uplus C_m$ such that for each i, C_i is G-stable and $\Pi_i, \Pi'_i \subseteq C_i$.
Parameter: $b = \max\{|C_1|, \cdots, |C_m|\}$.
Output: A description of $(Gz)_{\Pi_1,\ldots,\Pi_m \to \Pi'_1,\ldots,\Pi'_m} = \{x \in Gz \mid \Pi_i^x = \Pi'_i$ for $i = 1, \ldots, m\}$.

The fpt algorithm for solving CST works by restricting the problem to a particular color class, solving it, and then proceeding to the next color class. The restricted version of the problem is defined as follows.

Restricted CST (RCST)

Input: A generating set for a group $G \leq \mathrm{Sym}(C \uplus D)$, a permutation $z \in \mathrm{Sym}(C \uplus D)$ and subsets $\Pi, \Pi' \subseteq C$, where C is G-stable.
Parameter: $b = |C|$.
Output: A description of $(Gz)_{\Pi \to \Pi'} = \{x \in Gz \mid \Pi^x = \Pi'\}$.

Lemma 2. *There is an fpt algorithm for* RCST *running in time* $2^{O(b)}n^{O(1)}$, *where* $b = |C|$ *and* $n = |C| + |D|$.

Proof. Let $G_\Pi = \{x \in G \mid \Pi^x = \Pi\}$ be the stabilizer subgroup of G that stabilizes Π. Since C is G-stable the set Π can only move to a subset Π^x of size $k = |\Pi|$ contained in C. Thus, it follows that

$$[G : G_\Pi] \leq \binom{b}{k} \leq 2^b.$$

Also note that given $x \in G$, it only takes $O(n)$ time to check if $x \in G_\Pi$. Thus using the algorithm given by Theorem 1 we can compute a set $R = \{\rho_1, \cdots, \rho_t\}$ of coset representatives of G_Π in G in time $2^{O(b)}n^{O(1)}$ together with a strong generating set S for G_Π of size at most n^2. Writing

$$Gz = G_\Pi \rho_1 z \uplus \cdots \uplus G_\Pi \rho_t z,$$

the algorithm for RCST picks the uniquely determined coset $G_\Pi \rho_i z$ that sends Π to Π' and outputs the pair $(S, \rho_i z)$ as a description of that coset (if none of the cosets $G_\Pi \rho_i z$, $1 \leq i \leq t$, sends Π to Π', the algorithm outputs the empty set). $\square$

Remark 3. *As the group* G_Π *that is output by the algorithm of Lemma 2 is a subgroup of* G, G_Π *stabilizes any* G*-stable set.*

Theorem 4. *There is an fpt algorithm for* CST *running in time* $2^{O(b)}n^{O(1)}$, *where* $b = \max\{|C_1|, \cdots, |C_m|\}$ *and* $n = |\Omega|$.

Proof. Let $G_0 = G$ and $z_0 = z$ and for $i = 1, \cdots, m$ use the algorithm of Lemma 2 to compute

$$G_i z_i = (G_{i-1} z_{i-1})_{\Pi_i \to \Pi'_i}.$$

Notice that for each $x \in G_m z_m$ we have $\Pi_i^x = \Pi'_i$ for $i = 1, \ldots, m$, implying that $G_m z_m = (Gz)_{\Pi_1, \ldots, \Pi_m \to \Pi'_1, \ldots, \Pi'_m}$.

Furthermore, by Remark 3, each of the subgroups G_i stabilizes the sets C_j, $j = 1, \cdots, m$. Thus, Lemma 2 implies that we can compute $G_i z_i$ from $G_{i-1} z_{i-1}$ in time $2^{O(b)}n^{O(1)}$, implying that the overall running time is also $2^{O(b)}n^{O(1)}$. $\square$

Before we consider the colored version of the coset intersection problem, we state an immediate application of Theorem 4 to the set stabilizer problem.

Colored Set Stabilizer (CSS)

Input: A generating set for a group $G \leq \mathrm{Sym}(\Omega)$, a permutation $z \in \mathrm{Sym}(\Omega)$, subsets $\Pi_1, \ldots, \Pi_m \subseteq \Omega$ and a partition $\Omega = C_1 \uplus \cdots \uplus C_m$ such that for each i, C_i is G-stable and $\Pi_i \subseteq C_i$.
Parameter: $b = \max\{|C_1|, \cdots, |C_m|\}$.
Output: $(Gz)_{\Pi_1, \ldots, \Pi_m} = \{x \in Gz \mid \Pi_i^x = \Pi_i \text{ for } i = 1, \ldots, m\}$.

Since CSS is the special case of CST where Π'_i is set to Π_i, Theorem 4 shows that CSS is solvable by an fpt algorithm.

Corollary 5. *There is an fpt algorithm for* CSS *running in time* $2^{O(b)}poly(n)$.

The coset intersection problem is defined as follows.

Coset Intersection (CI)

Input: Generating sets for two groups $G, H \leq \mathrm{Sym}(\Omega)$ and $x, y \in \mathrm{Sym}(\Omega)$.
Output: $Gx \cap Hy$.

GI is known to be polynomial-time reducible to CI, since the latter is polynomial time many-one equivalent to Set Transporter [11]. However, the known polynomial-time reduction between these problems is not an fpt reduction [5]. Therefore, we cannot invoke the algorithm for CST to get an fpt algorithm for the parametrized version CCI of CI. Nevertheless we design an fpt algorithm for CCI that we will use in the next section to solve Colored Hypergraph Isomorphism.

Colored Coset Intersection (CCI)

Input: Generating sets for groups $G, H \leq \mathrm{Sym}(\Omega)$, permutations $x, y \in \mathrm{Sym}(\Omega)$ and a partition $\Omega = C_1 \uplus \cdots \uplus C_m$ such that for each i, C_i is $G \cup H \cup \{x, y\}$-stable.
Parameter: $b = \max\{|C_1|, \cdots, |C_m|\}$.
Output: $Gx \cap Hy$.

In the proof of the next theorem which provides an fpt algorithm for CCI we make use of the following lemma where we consider a suitably chosen restricted version of Colored Set Stabilizer. Here, the set Θ is only used to facilitate an inductive proof. Later, we will apply Lemma 6 only with Θ set to $C \times D$.

Restricted CSS (RCSS)

Input: A generating set for a group $L \leq \mathrm{Sym}(\Omega_1) \times \mathrm{Sym}(\Omega_2)$, a permutation $z \in \mathrm{Sym}(\Omega_1 \times \Omega_2)$ and subsets $\Pi, \Theta = \Phi \times \Psi \subseteq C \times D$, where $\Omega_1 = C \uplus U$, $\Omega_2 = D \uplus V$ and the two sets $C \times D$ and Θ are L-stable.
Parameter: $b = \max\{|C|, |D|\}$.
Output: $(Lz)_{\Pi}[\Theta] = \{x \in Lz \mid (\Pi \cap \Theta)^x = \Pi \cap \Theta^x\}$.

Lemma 6. *There is an fpt algorithm for* RCSS *running in time* $2^{O(b)}n^{O(1)}$, *where* $b = \max\{|C|, |D|\}$ *and* $n = |\Omega_1| + |\Omega_2|$.

Proof. We use ideas from [10, Proposition 3.1] where the author describes an algorithm for a version of the set transporter problem that can be easily adapted to solve RCSS. We only have to slightly modify Luks' proof to suit the parametrized setting.

We can assume that $|\Phi|$ and $|\Psi|$ are powers of 2 since otherwise we can add some points to Φ and Ψ (as well as to C and D) and let L act trivially on these points. This will increase the size of b and of the input only by a factor of 4. Further, these extra points can be easily removed from the algorithm's output.

Observe that since $L_\Theta = L$, we have $\Theta^x = \Theta^z$ for all $x \in Lz$. If $(Lz)_\Pi[\Theta]$ is not empty then for $x, y \in (Lz)_\Pi[\Theta]$ we have $(\Pi \cap \Theta)^x = \Pi \cap \Theta^z = (\Pi \cap \Theta)^y$ and hence $(Lz)_\Pi[\Theta]$ is a coset of $L_{\Pi\cap\Theta}$.

Clearly, if $|\Pi \cap \Theta| \neq |\Pi \cap \Theta^z|$ then $(Lz)_\Pi[\Theta]$ is empty. Next we consider the case that $|\Pi \cap \Theta| = |\Pi \cap \Theta^z| = 1$. Let $\Pi \cap \Theta = \{u\}$ and $\Pi \cap \Theta^z = \{v\}$. Let L_u be the stabilizer of the point u which can be computed using the Schreier-Sims method. Then we can express L as the disjoint union of cosets

$$L = L_u x_1 \uplus \cdots \uplus L_u x_t$$

and consequently Lz as $L_u x_1 z \uplus \cdots \uplus L_u x_t z$. Hence, it suffices to pick the uniquely determined coset $L_u x_i z$ that maps u to v (if there is any).

It remains to consider the case that $|\Pi \cap \Theta| = |\Pi \cap \Theta^z| > 1$. If $|\Phi| > 1$ we partition Φ in two subsets Φ_1 and Φ_2 of equal size and let $\Theta_1 = \Phi_1 \times \Psi$. Otherwise, $|\Psi_i| > 1$ and we partition Ψ in two subsets Ψ_1 and Ψ_2 of equal size and let $\Theta_1 = \Phi \times \Psi_1$. In both cases we let $\Theta_2 = \Theta \setminus \Theta_1$.

Let $k = \max\{|\Phi|, |\Psi|\}$ and let $M = L_{\Theta_1}$. Notice that $[L : M] \leq \binom{k}{k/2} \leq 2^b$, no matter which of the two sets Φ or Ψ we divide into two parts. Now we write L as the disjoint union of cosets

$$L = My_1 \uplus \cdots \uplus My_t$$

of M, yielding $Lz = My_1 z \uplus \cdots \uplus My_t z$. As mentioned in the preliminary section, this decomposition of Lz can be computed in time $2^{O(b)} n^{O(1)}$. Since M stabilizes Θ_1, $(My_i z)_\Pi[\Theta_1]$ is a coset of M. Moreover, we can use the equality

$$(My_i z)_\Pi[\Theta] = ((My_i z)_\Pi[\Theta_1])_\Pi[\Theta_2]$$

to setup the recursive calls. Finally we paste the answers to the subproblems $(My_i z)_\Pi[\Theta]$ together to get

$$(Lz)_\Pi[\Theta] = \cup_{i=1}^t (My_i z)_\Pi[\Theta].$$

It is easy to verify that the overall run-time of the algorithm is bounded by $2^{O(b)} poly(n)$. □

Theorem 7. *There is an fpt algorithm for* CCI *running in time* $2^{O(b)} n^{O(1)}$*, where* $b = \max\{|C_1|, \cdots, |C_m|\}$ *and* $n = |\Omega|$.

Proof. Let $L = G \times H \leq \mathrm{Sym}(\Omega) \times \mathrm{Sym}(\Omega)$ and let $z = (x, y) \in \mathrm{Sym}(\Omega) \times \mathrm{Sym}(\Omega)$. Further, let $\Pi_i = \{(a, a) \mid a \in C_i\}$ and notice that $(Lz)_{\Pi_1,\ldots,\Pi_m} = \{x \in Lz \mid \Pi_i^x = \Pi_i \text{ for } i = 1, \ldots, m\}$ projected to the first (or second) coordinate is $Gx \cap Hy$. Hence, it suffices to prove the following claim.

Claim. $(Lz)_{\Pi_1,\ldots,\Pi_m}$ is computable in time $2^{O(b)}n^{O(1)}$.

Since computing $(Lz)_{\Pi_1,\ldots,\Pi_i}$ is an instance of CSS we could use Corollary 5 to do it. Unfortunately, this approach would take time $2^{O(b^2)}n^{O(1)}$, since the group L has color class bound b^2 (and not b). Nevertheless, by repeatedly using Lemma 6 we can still solve the problem in time $2^{O(b)}n^{O(1)}$.

To start off we let $L_0z_0 = Lz$. Then we compute $L_iz_i = (L_{i-1}z_{i-1})_{\Pi_i}$ from $L_{i-1}z_{i-1}$ for $i = 1, \cdots, m$.

We claim that for all i, $L_iz_i = (Lz)_{\Pi_1,\ldots,\Pi_i}$. This follows from the fact that $((Lz)_{\Pi_1,\ldots,\Pi_{i-1}})_{\Pi_i} = (Lz)_{\Pi_1,\ldots,\Pi_i}$. Thus at the end of the computation we have $L_mz_m = (Lz)_{\Pi_1,\ldots,\Pi_m}$.

Furthermore, by Lemma 6 it follows that the time needed for computing L_iz_i from $L_{i-1}z_{i-1}$ is $2^{O(b)}n^{O(1)}$, implying that the overall running time is also $2^{O(b)}n^{O(1)}$. □

4 Fpt algorithms for the colored hypergraph automorphism and isomorphism problems

In this section, we use a dynamic programming approach to design an fpt algorithm for finding the automorphism group $\mathrm{Aut}(X)$ (i.e., a set of generators for $\mathrm{Aut}(X)$) of a given hypergraph X which has running time $(b!)2^{O(b)}N^{O(1)}$, where b is the color class bound and N is the input size. The subproblems of this dynamic programming algorithm will involve multi-hypergraphs. For this reason we begin with a multihypergraph $X = (V, E)$ as input, where V is partitioned into color classes $C_1, \cdots, C_m$ such that $|C_i| \leq b$ for all i.

Theorem 8. *Let $X = (V, E)$ be a colored multi-hypergraph with $V = C_1 \uplus \cdots \uplus C_m$ where $|C_i| \leq b$ for all i. Let N be the total number of vertices and edges of X. Given X as input there is an algorithm that computes $\mathit{Aut}(X)$ in time $b!2^{O(b)}N^{O(1)}$.*

Proof. The algorithm first partitions the hyperedges into different multisets that we call blocks. More formally, we say that two hyperedges $e_1, e_2 \in E$ are *i-equivalent* and write $e_1 \equiv_i e_2$, if

$$e_1 \cap C_j = e_2 \cap C_j \text{ for } j = 0, \ldots, i,$$

where we let $C_0 = \emptyset$. We call the corresponding equivalence classes *(i)-blocks.*

Notice that for $i \geq j$, i-equivalence is a refinement of j-equivalence. Thus, if e_1 and e_2 are in the same (i)-block then they are in the same (j)-block for all $j = 0, 1, \ldots, i-1$. The algorithm proceeds in stages $i = m, m-1, \ldots, 0$, where in stage i the algorithm considers (i)-blocks. More precisely, in stage i the algorithm computes for each pair of (i)-blocks A, A' the set $\mathit{ISO}(Y, Y')$ of all isomorphisms between the multihypergraphs Y and Y' induced by A and A', respectively, on the vertex set $C_i \cup \cdots \cup C_m$ and stores this set in a table T.

Stage m: Let A, A' be two (m)-blocks and let Y, Y' be the corresponding multihypergraphs on the vertex set C_m as defined above. Since A and A' are (m)-blocks, the multisets $E(Y) = \{\{e \cap C_m \mid e \in A\}\}$ and $E(Y') = \{\{e \cap C_m \mid e \in A'\}\}$ only contain a single hyperedge a and a' with multiplicity $|A|$ and $|A'|$, respectively.

Clearly, *ISO*$(Y, Y') = \emptyset$ if $|A| \neq |A'|$ or $|a| \neq |a'|$. Otherwise, *ISO*$(Y, Y') \subseteq \mathrm{Sym}(C_m)$ is the coset of $\mathrm{Aut}(Y) = \mathrm{Sym}(C_m)_a$ that maps a to a' wich can be easily computed in time $2^{O(b)}$ by Theorem 4 and stored in $T[A, A']$.

Stage $i < m$: Let A, A' be two (i)-blocks and let Y, Y' be the corresponding multihypergraphs on the vertex set $C_i \cup \cdots \cup C_m$. We explain how the computation of $T[A, A'] =$ *ISO*(Y, Y') is done.

Let a and a' be the unique subsets of C_i such that for all $e \in A$, $e \cap C_i = a$ and for all $e' \in A'$, $e' \cap C_i = a'$. Clearly *ISO*(Y, Y') is empty if the sizes of a and a' or the sizes of the hyperedge multisets $E(Y) = \{\{e \cap (C_i \cup \cdots \cup C_m) \mid e \in A\}\}$ and $E(Y') = \{\{e \cap (C_i \cup \cdots \cup C_m) \mid e \in A'\}\}$ differ. Otherwise, let $S_1 = \{\varphi \in \mathrm{Sym}(C_i) \mid a^\varphi = a'\}$ be the set containing all permutations in $\mathrm{Sym}(C_i)$ that map a to a'. If S is empty then so is *ISO*(Y, Y'). Otherwise, let S_2 be the set of all permutations on $C_{i+1} \cup \cdots \cup C_m$ that map Y to Y' isomorphically *when restricted* to the color classes $C_{i+1}, \cdots, C_m$. Crucially, since A and A' are both (i)-blocks it follows that *ISO*$(Y, Y') = S_1 \times S_2$.

Clearly, S_1 can be easily computed as explained in stage m. To compute S_2, we partition the (i)-blocks A and A' into $(i+1)$-blocks $A_1, \cdots, A_k$ and $A'_1, \cdots, A'_{k'}$, respectively. Since S_2 is empty if $k \neq k'$ we assume $k = k'$. For each $j = 1, \ldots, k$, let Z_j and Z'_j be the multihypergraphs induced by the $(i+1)$-blocks A_j and A'_j, respectively, on the vertex set $C_{i+1}, \cdots, C_m$. Now it is easy to see that

$$S_2 = \bigcup_{\pi \in S_k} \bigcap_{j=1}^{k} \mathit{ISO}(Z_j, Z'_{\pi(j)}),$$

where the sets *ISO*$(Z_j, Z'_{\pi(j)})$ are already stored in the table T. Notice that instead of cycling through all $\pi \in S_k$ it suffices to cycle through all $\rho \in \mathrm{Sym}(C_{i+1})$ and check whether $\{\{e \cap C_{i+1} \mid e \in A\}\}^\rho = \{\{e' \cap C_{i+1} \mid e' \in A'\}\}$. For each such ρ the corresponding permutation $\pi \in S_k$ with $\{\{e \cap C_{i+1} \mid e \in A_j\}\}^\rho = \{\{e' \cap C_{i+1} \mid e' \in A'_{\pi(j)}\}\}$ can be easily derived.

Now we apply Theorem 7 to compute for each $\rho \in \mathrm{Sym}(C_{i+1})$ which corresponds to some $\pi \in S_k$ as explained above the intersection $H_\rho \sigma_\rho = \bigcap_{j=1}^{k} \mathit{ISO}(Z_j, Z'_{\pi(j)})$ which is either empty or a coset. As $k \leq 2^b$, this takes time bounded by $2^{O(b)} N^{O(1)}$. Next the algorithm computes

$$S_2 = \bigcup_{\rho \in \mathrm{Sym}(C_{i+1})} H_\rho \sigma_\rho$$

which again is either empty or a coset and stores the set $S_1 \times S_2$ in $T[A, A']$.

Since there is a single (0)-block E, we can find $\mathrm{Aut}(X) = T(E, E)$ in the table. It remains to analyze the running time of the algorithm. The number of blocks at any stage is bounded by the number of edges of X. Thus, the i-th stage takes time bounded by $b!2^{O(b)}N^{O(1)}$, where the $b!$ factor is because we cycle through all the $\rho \in \mathrm{Sym}(C_{i+1})$. □

Next we indicate how the above algorithm can be easily modified to give an isomorphism algorithm for colored hypergraphs without changing the running time. Let $X = (V, E)$ and $X' = (V', E')$ be two colored hypergraphs. Without loss of generality we assume $V = V' = C_1 \uplus \cdots \uplus C_m$. As before the algorithm computes for each pair of (i)-blocks A, A' the set $ISO(Y, Y')$, where Y and Y' are the multihypergraphs induced by A and A', respectively, with the only difference that now the block A comes from the hypergraph X and A' comes from X'. Thus, in stage 0 the algorithm computes the set $ISO(X, X')$ of isomorphisms from X to X'.

Corollary 9. *Let $X = (V, E)$ and $X' = (V, E')$ be two colored hypergraphs with $V = C_1 \uplus \cdots \uplus C_m$ where $|C_i| \leq b$ for all i. Let N be the total number of vertices and edges of X. Given X and X' as input there is an algorithm that computes the set $ISO(X, X')$ of isomorphisms from X to X' in time $b!2^{O(b)}N^{O(1)}$.*

References

1. 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.
2. H. Bodlaender. Polynomial algorithm for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990.
3. J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. In *Proc. 31st Symposium on Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 238–249. Springer-Verlag, 2006.
4. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.
6. S. Evdokimov and I. Ponomarenko. Isomorphism of colored graphs with slowly increasing multiplicity of Jordan blocks. *Combinatorica*, 19(3):321-333, 1999.
7. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.
8. 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.
9. E. Luks. Isomorphism of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.
10. E. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In *Proc. 31st ACM Symposium on Theory of Computing*, pages 652–658. ACM Press, 1999.
11. E. M. Luks. Permutation groups and polynomial time computations. In L. Finkelstein and W. M. Kantor, editors, *Groups and Computation*, volume 11 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 139–175. American Mathematical Society, 1993.
12. 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.

13. C. C. Sims. Computational methods in the study of permutation groups. In J. Leech, editor, *Computational problems in abstract algebra, Proc. Conf. Oxford, 1967*, pages 169–183. Pergamon Press, 1970.
14. C. C. Sims. Some group theoretic algorithms. In A. Dold and B. Eckmann, editors, *Topics in Algebra*, volume 697 of *Lecture Notes in Mathematics*, pages 108–124. Springer-Verlag, 1978.
15. S. Toda. Computing automorphism groups of chordal graphs whose simplicial components are of small size. *IEICE Transactions*, 89-D(8):2388–2401, 2006.
16. K. Yamazaki, H. L. Bodlaender, B. de Fluiter, and D. M. Thilikos. Isomorphism for graphs of bounded distance width. *Algorithmica*, 24(2):105–127, 1999.