Colored Hypergraph Isomorphism is Fixed Parameter Tractable

V. Arvind[1], Bireswar Das[2], Johannes Köbler[3], and Seinosuke Toda[4]

[1] The Institute of Mathematical Sciences, Chennai 600 113, India
arvind@imsc.res.in
[2] Indian Institute of Technology, Gandhinagar, India
bireswar@iitgn.ac.in
[3] Institut für Informatik, Humboldt Universität zu Berlin, Germany
koebler@informatik.hu-berlin.de
[4] 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 $2^{O(b)}N^{O(1)}$, 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.

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 $E(Y) = \{\{v, e\} \mid v \in V, e \in E \text{ and } v \in e\}$, and Y' is defined similarly. 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 $X \cong X'$ assuming that V can be mapped only to V' and E can be mapped only to E'. This latter condition is easy to enforce.

However, since the above reduction blows up the size of the vertex set in the bipartite encoding, the Zemlyachenko-Luks-Babai graph isomorphism algorithm [3, 5, 6, 25] that runs in time $c^{\sqrt{n \log n}}$, where n is the size of the vertex set of the graph, does not yield a similar algorithm for hypergraph isomorphism. We note here that the best known hypergraph isomorphism test due to Luks [16] has running time c^n. Recently, Babai and Codenotti [4] have shown a $2^{\tilde{O}(k^2\sqrt{n})}$

isomorphism testing algorithm for hypergraphs with hyperedges of size bounded by k.

Motivated by this situation, we explore the same algorithmic problem for bounded color class hypergraphs. The input to Colored Hypergraph Isomorphism (CHI) is a pair of hypergraphs $X = (V, E)$ and $X' = (V', E')$ together with partitions $V = C_1 \uplus \cdots \uplus C_k$ and $V' = C'_1 \uplus \cdots \uplus C'_k$ 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.

CGI with color classes of size bounded by a constant is the first special case of GI shown to be in polynomial time [2, 12] and which brought in the application of permutation group theory to the problem. In fact, Babai [2] and Furst, Hopcroft and Luks [12] even gave an fpt algorithm for CGI with running time $O(b!)n^{O(1)}$, where the parameter b is the maximum size of the color classes and n is the number of vertices of the input graphs. By using the halving technique as introduced in [5] (see also [16]), the running time can be improved to $2^{O(b)}n^{O(1)}$.

In [13] a *complexity-theoretic* study of some special cases of bounded color class Graph Isomorphism has been done in connection to logarithmic space-bounded complexity classes. This line of research is continued in [1], where special cases of bounded color class Graph Isomorphism as well as Hypergraph Isomorphism are studied from a complexity theory perspective.

In this paper our focus is on designing an efficient algorithm for CHI. Although HI is polynomial time 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 size at most 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. 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 can have size exponential in n where n is the number of vertices and hence, this approach would not even give a polynomial time isomorphism algorithm for hypergraphs with color class bound 2.

However, an algorithm for CHI running in time $N^{O(b)}$ was shown in [19], 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 already a polynomial-time algorithm for CHI. This algorithm basically applies Luks's seminal result [15] showing that the set stabilizer problem with respect to a class of permutation groups Γ_d can be solved in time $n^{O(d)}$.

Parametrized complexity and isomorphism testing

Parametrized complexity is a fundamental strategy for coping with intractability. Pioneered by Downey and Fellows in [8], it is a flourishing area of research (see, e.g. the monographs [9, 11]). 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.

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).

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 Babai and Furst et al [2, 12] 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 $k^{O(k)}n^{O(1)}$ [10].

Apart from this, fpt algorithms have also been designed with respect to the parameters tree-distance width [24] and the size of the simplicial components of the input graphs [23]. Very recently, it is shown in [14] that Graph Isomorphism for graphs with feedback vertex sets of size k is fixed parameter tractable, with k as the parameter.

On the other hand, if we use the maximum degree [15], or the treewidth [7], or the genus [18] 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.

Our result

In this paper we present an fpt algorithm for Colored Hypergraph Isomorphism that runs in time $2^{O(b)}N^{O(1)}$, where b is the maximum size of the color classes and N is the input size (which we can define as $N = mn$, where n is the number of vertices and m is the number of hyperedges).

Broadly speaking, our algorithm is a combination of divide and conquer with dynamic programming. We adapt ideas from [5, 16] which applies the halving technique in combination with dynamic programming. Luks [16] gives a $2^{O(n)}$ time algorithm for Hypergraph Isomorphism. Our algorithm can be seen as a generalization of Luks's result.

We use as subroutines fpt algorithms for certain permutation group problems (mainly, the coset intersection problem) parametrized by the size of the largest *color class* of the group. 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 w.r.t. groups in Γ_d (with d as parameter) would result in an fpt algorithm for testing isomorphism of graphs of degree $\leq d$.

2 Preliminaries

In this section 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 [21, 22, 12] 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 [21, 22, 12] 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 $B = \{r' x r^{-1} \mid r, r' \in R, x \in S\} \cap H$ 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 $O(n^2)$ [21, 22, 12].

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. These algorithms are useful subroutines for our main algorithm which will be described in the next section.

A permutation group $G \leq \mathrm{Sym}(\Omega)$ has *color class bound* b if Ω is a colored set partitioned into *color classes* $\Omega = C_1 \uplus \cdots \uplus C_k$ such that $|C_i| \leq b$ for each i and 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 $|\Omega|^{O(1)}$ time (for G given by a generating set S), we can determine in $|\Omega|^{O(1)}$ time if G has color class bound b. We first consider the following parametrized version of the set transporter problem.

Set Transporter

Input: A generating set for a group $G \leq \mathrm{Sym}(\Omega)$, a permutation $z \in \mathrm{Sym}(\Omega)$, subsets $\Pi_1, \ldots, \Pi_k, \Pi'_1, \ldots, \Pi'_k \subseteq \Omega$ and a partition $\Omega = C_1 \uplus \cdots \uplus C_k$ such that for each i, C_i is G-stable and $\Pi_i, \Pi'_i \subseteq C_i$.

Parameter: $b = \max\{|C_1|, \cdots, |C_k|\}$.

Output: A description of $(Gz)_{\Pi_1,\ldots,\Pi_k \to \Pi'_1,\ldots,\Pi'_k} = \{x \in Gz \mid \Pi_i^x = \Pi'_i \text{ for } i = 1, \ldots, k\}$.

The simple fpt algorithm for Set Transporter works by solving the problem for the first color class C_1 by computing the subcoset $G_1 z_1$ of Gz that maps Π_1 to Π'_1, then computing the subcoset $G_2 z_2$ of $G_1 z_1$ that maps Π_2 to Π'_2 and so on until all the color classes are dealt with. The following lemma shows how to compute $G_i z_i$ from $G_{i-1} z_{i-1}$.

Lemma 1. *There is an fpt algorithm running in time $2^{O(b)} n^{O(1)}$ that computes the subcoset $H'y'$ of the coset Hy that maps Π_i to Π'_i, where $\Pi_i, \Pi'_i \subseteq C_i$.*

Proof. Let $H_{\Pi_i} = \{x \in H \mid \Pi_i^x = \Pi_i\}$ be the subgroup of H that stabilizes Π_i. Let $|\Pi_i| = \ell$. Since C_i is H-stable the set Π_i^x is also a size ℓ subset of C_i. It follows that

$$[H : H_{\Pi_i}] \leq \binom{b}{\ell} \leq 2^b.$$

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

$$Hy = H_{\Pi_i}\rho_1 y \uplus \cdots \uplus H_{\Pi_i}\rho_t y,$$

the algorithm picks the uniquely determined coset $H_{\Pi_i}\rho_i y$ that sends Π_i to Π'_i and outputs the pair $(S, \rho_i z)$ as a description of the coset $H_{\Pi_i}\rho_i y$. If none of the cosets $H_{\Pi_i}\rho_i z$ maps Π_i to Π'_i, the algorithm outputs the empty set.

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

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

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

Notice that for each $x \in G_k z_k$ we have $\Pi_i^x = \Pi'_i$ for $i = 1, \ldots, k$, implying that $G_k z_k = (Gz)_{\Pi_1,\ldots,\Pi_k \to \Pi'_1,\ldots,\Pi'_k}$.

Furthermore, each of the subgroups G_i stabilizes the sets C_j, $j = 1, \cdots, k$. Thus, Lemma 1 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 $2^{O(b)}n^{O(1)}$.

Next we consider the following parametrized version of the coset intersection problem.

Coset Intersection (Coset-Inter)

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

Applying well-known techniques from [5] we will design an fpt algorithm for Coset-Inter. We will use this as a subroutine in the next section to solve Colored Hypergraph Isomorphism. Our fpt algorithm for Coset-Inter will require solving a subproblem which is a restricted version of the set stabilizer problem.

Restricted Set Stabilizer (RSS)

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 2. *There is an fpt algorithm for* RSS *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 [16, Proposition 3.1] where the author describes an algorithm for a version of the set transporter problem that can be easily adapted to solve RSS. These ideas were first applied in [5]. We only have to slightly modify Luks's 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

$$L = My_1 \uplus \cdots \uplus My_s$$

M, yielding $Lz = My_1 z \uplus \cdots \uplus My_s z$. 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, we can use the equality

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

to set up 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 3. *There is an fpt algorithm for* Coset-Inter *running in time* $2^{O(b)} n^{O(1)}$*, where* $b = \max\{|C_1|, \cdots, |C_k|\}$ *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 $(Lz)_{\Pi_1, \ldots, \Pi_k} = \{x \in Lz \mid \Pi_i^x = \Pi_i \text{ for } i = 1, \ldots, k\}$ projected to the first (or second) coordinate is $Gx \cap Hy$. Hence, it suffices to prove the following claim.

Claim 4. *$(Lz)_{\Pi_1,\ldots,\Pi_k}$ is computable in time $2^{O(b)}n^{O(1)}$.*

We will repeatedly use Lemma 2 to solve the problem in time $2^{O(b)}n^{O(1)}$ as in the above claim. 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 $i = 1, \cdots, k$. 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 $L_kz_k = (Lz)_{\Pi_1,\ldots,\Pi_k}$. Furthermore, by Lemma 2 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 Colored Hypergraph Isomorphism

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 $2^{O(b)}N^{O(1)}$.

Theorem 5. *Let $X = (V, E)$ be a colored hypergraph of size N with $V = C_1 \uplus \cdots \uplus C_k$ where $|C_i| \leq b$ for all i. Given X as input there is an algorithm that computes $Aut(X)$ in time $2^{O(b)}N^{O(1)}$.*

Proof. The algorithm first partitions the hyperedges into different subsets 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 $i = k, k-1, \ldots, 0$, 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 coset $ISO(Y, Y')$ of all isomorphisms between the hypergraphs Y and Y' induced by A and A', respectively, on $V_i = C_i \cup \cdots \cup C_k$ and stores this coset in a table T.

Stage k: Let A, A' be two (k)-blocks and let Y, Y' be the corresponding hypergraphs on the vertex set C_k as defined above. Since A and A' are (k)-blocks, the sets $E(Y) = \{e \cap C_k \mid e \in A\}$ and $E(Y') = \{e \cap C_k \mid e \in A'\}$ only contain a single hyperedge a and a', respectively.
Clearly, $ISO(Y, Y') = \emptyset$ if $|a| \neq |a'|$. Otherwise, $ISO(Y, Y') \subseteq \mathrm{Sym}(C_k)$ is the coset of $\mathrm{Aut}(Y) = \mathrm{Sym}(C_k)_a$ that maps a to a' which can be easily computed in time $O(N)$ and stored in the table entry $T[A, A']$.

Stage $i < k$: Let A, A' be two (i)-blocks and let Y, Y' be the corresponding hypergraphs on the vertex set V_i. We explain how to compute the entry $T[A, A'] = ISO(Y, Y')$.
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 sets $E(Y) = \{e \cap V_i \mid e \in A\}$ and $E(Y') = \{e \cap V_i \mid e \in A'\}$ 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' and let S_2 be the set of all permutations on V_{i+1} that map Y to Y' isomorphically *when restricted* to V_{i+1}. 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 above. The crux of the algorithm is in computing the set S_2. We first explain a naive method that takes time $(b!)2^{O(b)}N^{O(1)}$ (later we will explain the more complicated $2^{O(b)}N^{O(1)}$ algorithm for computing S_2).

To compute S_2, we partition the (i)-blocks A and A' into $(i+1)$-blocks $A_1, \cdots, A_\ell$ and $A'_1, \cdots, A'_{\ell'}$, respectively. Since S_2 is empty if $\ell \neq \ell'$ we assume $\ell = \ell'$. For each $j = 1, \ldots, \ell$, let Z_j and Z'_j be the hypergraphs induced by the $(i+1)$-blocks A_j and A'_j, respectively, on the vertex set V_{i+1}. Now it is easy to see that

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

where the sets $ISO(Z_j, Z'_{\pi(j)})$ are already stored in the table T. Now, observe that instead of cycling through all $\pi \in S_\ell$ 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 induced permutation $\pi \in S_\ell$ 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 can apply Theorem 3 to compute for each $\rho \in \mathrm{Sym}(C_{i+1})$ which corresponds to some $\pi \in S_\ell$ as explained above the coset intersection $H_\rho\sigma_\rho = \bigcap_{j=1}^{\ell} ISO(Z_j, Z'_{\pi(j)})$ which is either empty or a coset. As $\ell \leq 2^b$, this takes time bounded by $2^{O(b)}N^{O(1)}$. Now the algorithm can compute

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

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 $\rho \in \mathrm{Sym}(C_{i+1})$.

In order to obtain the improved $2^{O(b)}N^{O(1)}$ time bound, it suffices to give a $2^{O(b)}N^{O(1)}$ time algorithm for computing the coset S_2 of all permutations on V_{i+1} that map Y to Y' isomorphically *when restricted* to V_{i+1}.

Claim 6. *There is a $2^{O(b)}N^{O(1)}$ time algorithm for computing S_2.*

We will compute S_2 with a dynamic programming strategy that will involve solving $2^{O(b)}$ many subproblems and $2^{O(b)}$ many coset intersection instances for

which we can invoke Theorem 3. We use ideas from Luks's dynamic programming algorithm in [16]. For each subset $\Delta \subseteq C_{i+1}$ and $\Sigma \subseteq C_{i+1} \setminus \Delta$ we define hypergraphs

$$Y^{\Delta,\Sigma} = \{e \cap V_{i+1} \mid e \in Y, e \cap (C_{i+1} \setminus \Delta) = \Sigma\}, \text{ and}$$
$$Y'^{\Delta',\Sigma'} = \{e' \cap V_{i+1} \mid e' \in Y', e' \cap (C_{i+1} \setminus \Delta') = \Sigma'\}.$$

Notice that Y projected on V_{i+1} is $Y^{C_{i+1},\emptyset}$ and that Y' projected on V_{i+1} is $Y'^{C_{i+1},\emptyset}$, and we are interested in computing $S_2 = ISO(Y^{C_{i+1},\emptyset}, Y'^{C_{i+1},\emptyset})$. Furthermore, notice that for different subsets Σ and Σ' the hypergraphs $Y^{\emptyset,\Sigma}$ and $Y^{\emptyset,\Sigma'}$ are the hypergraphs induced by the different $(i+1)$-blocks. Observe that in the $(i+1)^{st}$ stage we have already computed the cosets $ISO(Y^{\emptyset,\Sigma}, Y'^{\emptyset,\Sigma'})$ for different Σ and Σ' (as these correspond to the different $(i+1)$-blocks). Our goal is to compute all the cosets

$$ISO(\Delta, \Sigma, \Delta', \Sigma'),$$

consisting of all isomorphisms π from the hypergraph $Y^{\Delta,\Sigma}$ to the hypergraph $Y'^{\Delta',\Sigma'}$ that map Δ to Δ' and Σ to Σ'. To this end we actually compute for different subsets $\Gamma \subseteq \Delta$ and $\Gamma' \subseteq \Delta'$ the cosets

$$ISO(\Delta, \Sigma, \Gamma, \Delta', \Sigma', \Gamma') = ISO(\Delta, \Sigma, \Delta', \Sigma') \cap \text{Coset}(\Gamma, \Gamma'), \tag{1}$$

where $\text{Coset}(\Gamma, \Gamma')$ denotes the coset of all permutations on V_{i+1} that map Γ to Γ'. Notice that $\text{Coset}(\Gamma, \Gamma')$ can be easily computed in time $O(N)$. To complete the description of the dynamic programming algorithm, we consider different cases for $|\Gamma|$ and $|\Gamma'|$. Clearly, if $|\Gamma| \neq |\Gamma'|$ then the corresponding coset intersection of Equation 1 is the empty set.

Suppose $|\Gamma| = |\Gamma'| = \ell > 1$. In this case, we fix a subset Γ_1 of Γ of size $\lceil \ell/2 \rceil$ and cycle through all possible subsets Γ'_1 of Γ' of size $\lceil \ell/2 \rceil$. Clearly, we can write

$$ISO(\Delta, \Sigma, \Gamma, \Delta', \Sigma'\Gamma') =$$
$$\bigcup_{\Gamma'_1 \subset \Gamma'} (ISO(\Delta, \Sigma, \Gamma_1, \Delta', \Sigma', \Gamma'_1) \cap ISO(\Delta, \Sigma, \Gamma \setminus \Gamma_1, \Delta', \Sigma', \Gamma' \setminus \Gamma'_1)),$$

where the union runs over subsets of size $\lceil \ell/2 \rceil$. Computing this union as a coset essentially involves solving at most 2^b many coset intersections, each of which takes $2^{O(b)}N^{O(1)}$ time, assuming that the dynamic programming table entries for Γ_1 and Γ'_1 are already there. Finally, we turn to the case when $|\Gamma| = |\Gamma'| = 1$. Let $\Gamma = \{\gamma\}$ and $\Gamma' = \{\gamma'\}$. Let $\Delta_1 = \Delta \setminus \{\gamma\}$ and $\Delta'_1 = \Delta' \setminus \{\gamma'\}$. It is easy to see that

$$ISO(\Delta, \Sigma, \{\gamma\}, \Delta', \Sigma', \{\gamma'\}) = \text{Coset}(\{\gamma\}, \{\gamma'\})$$
$$\cap\, ISO(\Delta_1, \Sigma \cup \{\gamma\}, \Delta_1, \Delta'_1, \Sigma' \cup \{\gamma'\}, \Delta'_1) \cap ISO(\Delta_1, \Sigma, \Delta_1, \Delta'_1, \Sigma', \Delta'_1),$$

which is again a coset intersection instance for table entries already computed since they correspond to smaller size sets Δ_1 and Δ'_1.

To complete the proof (of both the claim and the theorem), notice that we compute the table entries for increasing sizes of Δ. For each Δ we compute the entries for different Σ and increasing sizes of Γ. Finally, the base case for which the cosets in the table are already computed is when Δ is the empty set. For different subsets Σ these correspond to the $(i+1)$-blocks. This proves the correctness and the running time bound follows from the fact that the number of subproblems is $2^{O(b)}N^{O(1)}$, each of which involves $2^{O(b)}N^{O(1)}$ many coset intersections which takes $2^{O(b)}N^{O(1)}$ time by Theorem 3.

It is easy to modify the algorithm in the above theorem to compute all isomorphisms between two colored hypergraphs $X = (V, E)$ and $X' = (V', E')$ without changing the running time. Clearly, we can assume that $V = V' = C_1 \uplus \cdots \uplus C_k$. The new algorithm computes for each pair of (i)-blocks A, A' the set $ISO(Y, Y')$, where Y and Y' are the hypergraphs 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 all isomorphisms from X to X'.

Corollary 1. *Let $X = (V, E)$ and $X' = (V, E')$ be two colored hypergraphs of size N with $V = C_1 \uplus \cdots \uplus C_k$ where $|C_i| \leq b$ for all i. 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 $2^{O(b)}N^{O(1)}$.*

5 Discussion

We now briefly address the complexity of the canonization problem associated with CHI. We first recall the definition of canonization. Let $\mathcal{K}$ denote the set of all instances of CHI. A mapping $f : \mathcal{K} \to \mathcal{K}$ is a *canonizing function* for $\mathcal{K}$ if for all pairs of isomorphic instances X and X' in $\mathcal{K}$, $f(X) = f(X')$ and $f(X) \cong X$. We say that f assigns a *canonical form* to each isomorphism class of $\mathcal{K}$.

It is often the case that canonization and isomorphism testing for a class of structures have the same complexity. However, for CHI we do not know a canonization procedure even with running time $(b!)2^{O(b)}n^{O(1)}$. Indeed, we do not know if the problem is fixed parameter tractable. The following result is the best we know which follows easily by applying known techniques [6].

Theorem 7. *The canonization problem associated with* CHI *has an $N^{O(b)}$ time algorithm, where N is the input size and b bounds the size of the color classes.*

Proof Sketch. Let $X = (V, E)$ be an input instance of CHI, where $|E| = m$ and $|V| = n$. Then, by definition, the size of X is $N = mn$. Let $V = \bigcup_{i=1}^{k} C_i$ be the partition of the vertex set into color classes C_i, where $|C_i| \leq b$ for each i. Let $X_i = (V_i, E_i)$ denote the multi-hypergraph obtained from X by projecting the hyperedges $e \in E$ to the set $V_i = C_i \cup \ldots \cup C_k$. The canonization algorithm proceeds inductively. Suppose we have computed the canonical labeling coset $G\sigma$ of the multi-hypergraph X_{i+1}. It suffices to give an $m^{O(b)}$ algorithm for

canonizing the multi-hypergraph X_i obtained by projecting E on $V_i = C_i \cup V_{i+1}$, given the canonical labeling coset $G\sigma$ for X_{i+1}. Clearly, it suffices to canonize X_i under the action of the coset $\mathrm{Sym}(C_i) \times G\sigma$, where $\mathrm{Sym}(C_i)$ is the group of the (at most $b!$ many) permutations acting on the color class C_i. Applying the standard orbit finding algorithm for permutation groups [17, 20] we can compute the hypergraph X_i' with vertex set V_i and multiset E_i' consisting of all hyperedges $E_i' = \{\{e^\pi \mid e \in E_i \text{ and } \pi \in \mathrm{Sym}(C_i) \times G\sigma\}\}$. Since $G\sigma$ canonizes X_{i+1}, it follows that $|E_i'| \leq 2^b \cdot |E_i|$. Thus, X_i' can be easily computed in time $\mathrm{poly}(2^b, m, n)$. Notice that every permutation $\pi \in \mathrm{Sym}(C_i) \times G\sigma$ maps X_i to a subgraph X_i^π of the hypergraph X_i'. Furthermore, notice that the automorphism group $\mathrm{Aut}(X_i')$ of X_i' is precisely $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$. Define $Y_i = X_i^{(id,\sigma)}$, where id is the identity permutation in $\mathrm{Sym}(C_i)$. Then, Y_i is clearly a subgraph of X_i', and canonizing X_i under the action of the coset $\mathrm{Sym}(C_i) \times G\sigma$ is equivalent to canonizing Y_i under the action of $\mathrm{Aut}(X_i') = \mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$. Now, we write the multiset E_i' as

$$E_i' = \{(e_1, n_1), (e_2, n_2), \ldots, (e_r, n_r)\},$$

where the edges e_j are the distinct edges (with corresponding multiplicity n_j), lexicographically ordered. Since $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$ is $\mathrm{Aut}(X_i')$, each permutation in $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$ uniquely defines a permutation on the set $\{e_1, e_2, \ldots, e_r\}$. Thus $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$ gives rise to a subgroup H_i contained in $\mathrm{Sym}(\{e_1, \ldots, e_r\})$. Let $E(Y_i) = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$. The hypergraph Y_i, as a subgraph of X_i', can be represented by a *colored* binary string $x \in \{0,1\}^r$, whose j^{th} bit $x_j = 1$ iff $e_j \in E(Y_i)$, and x_j is colored by its multiplicity n_j.

The problem of canonizing X_i under $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$ action reduces to canonize the binary string $x \in \{0,1\}^r$ under the action of the group H_i. Since $\mathrm{Sym}(C_i) \times \sigma^{-1}G\sigma$ is a group with *composition width* [6] bounded by b, it follows that H_i also has composition width bounded by b. Hence, by invoking the Babai-Luks canonization procedure [6] we can compute the canonical form for X_i and the canonical labeling coset in $N^{O(b)}$ time. This completes the proof sketch. ■

References

1. V. Arvind and J. Köbler. On Hypergraph and Graph Isomorphism with 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. L. Babai. Monte Carlo algorithms for graph isomorphism testing. Technical Report 79-10, Dép. Math. et Stat., Univ. de Montréal, 1979.
3. 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.
4. L. Babai and P. Codenotti. Isomorphism of Hypergraphs of Low Rank in Moderately Exponential Time. In *Proc. 39th Ann. IEEE Symposium on the Foundations of Computer Science*, pages 667–676, IEEE Computer Society Press, 2008.

5. L. Babai, W. Kantor, and E. M. Luks. Computational complexity and the classification of finite simple groups. In *Proc. 24th IEEE Symposium on the Foundations of Computer Science*, pages 162–171. IEEE Computer Society Press, 1983.
6. L. Babai and E. M. Luks. Canonical labeling of graphs. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 171–183, 1983.
7. H. Bodlaender. Polynomial algorithm for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11(4):631–643, 1990.
8. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.
9. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
10. S. Evdokimov and I. Ponomarenko. Isomorphism of colored graphs with slowly increasing multiplicity of Jordan blocks. *Combinatorica*, 19(3):321–333, 1999.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
12. M. Furst, J. Hopcroft, and E. M. 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.
13. B. Jenner, J. Köbler, P. McKenzie, and J. Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66:549–566, 2003.
14. S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number is fixed-parameter tractable, 2009.
15. E. M. Luks. Isomorphism of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.
16. E. M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. In *Proc. 31st ACM Symposium on Theory of Computing*, pages 652–658. ACM Press, 1999.
17. 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.
18. 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.
19. G. L. Miller. Isomorphism of k-contractible graphs. A generalization of bounded valence and bounded genus. *Information and Computation*, 56(1/2):1–20, 1983.
20. Á. Seress. *Permutation Group Algorithms*. Cambridge University Press, 2003.
21. 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.
22. C. C. Sims. Some group theoretic algorithms. In A. Dold and B. Eckmann, editors, *Topics in Algebra*, volume 697 of *Lecture Notes in Mathematics*, 108–124. Springer 1978.
23. S. Toda. Computing automorphism groups of chordal graphs whose simplicial components are of small size. *IEICE Transactions*, 89-D(8):2388–2401, 2006.
24. 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.
25. V. N. Zemlyachenko, N. Konienko, and R. I. Tyshkevich. Graph isomorphism problem (Russian). *The Theory of Computation I, Notes Sci. Sem. LOMI 118*, 1982.