Parameterized Complexity of Small Weight Automorphisms*

Vikraman Arvind[1], **Johannes Köbler**[2], **Sebastian Kuhnert**[3], **and Jacobo Torán**[4]

1 **Institute of Mathematical Sciences (HBNI), Chennai, India**
`arvind@imsc.res.in`
2 **Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany**
`koebler@informatik.hu-berlin.de`
3 **Institut für Informatik, Humboldt-Universität zu Berlin, Berlin, Germany**
`kuhnert@informatik.hu-berlin.de`
4 **Institut für Theoretische Informatik, Universität Ulm, Ulm, Germany**
`toran@uni-ulm.de`

Abstract

We show that checking if a given hypergraph has an automorphism that moves exactly k vertices is fixed parameter tractable, using k and additionally either the maximum hyperedge size or the maximum color class size as parameters. In particular, it suffices to use k as parameter if the hyperedge size is at most polylogarithmic in the size of the given hypergraph.

As a building block for our algorithms, we generalize Schweitzer's FPT algorithm [ESA 2011] that, given two graphs on the same vertex set and a parameter k, decides whether there is an isomorphism between the two graphs that moves at most k vertices. We extend this result to hypergraphs, using the maximum hyperedge size as a second parameter.

Another key component of our algorithm is an orbit-shrinking technique that preserves permutations that move few points and that may be of independent interest. Applying it to a suitable subgroup of the automorphism group allows us to switch from bounded hyperedge size to bounded color classes in the exactly-k case.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Parameterized algorithms, hypergraph isomorphism.

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.7

1 Introduction

The Graph Automorphism problem GA asks whether a given graph has a nontrivial automorphism. We additionally require the automorphism to have small weight. The *weight* of a permutation is the number of its non-fixpoints. We are interested in the following problems:

GA$_{\le k}$: Given a graph $X = (V, E)$ and $k \in \mathbb{N}$, does X have a nontrivial automorphism that moves at most k vertices?

GI$_{\le k}$: Given two graphs $X_1 = (V, E_1)$ and $X_2 = (V, E_2)$ on the same vertex set and $k \in \mathbb{N}$, is there an isomorphism from X_1 to X_2 that moves at most k vertices?

* This work was supported by the Alexander von Humboldt Foundation in its research group linkage program. The second and third authors are supported by DFG grant KO 1053/7-2.

Likewise, let $\mathrm{GA}_{=k}$ and $\mathrm{GI}_{=k}$ denote the exact weight-k automorphism and isomorphism problems, respectively. By $\mathrm{HGA}_{=k}$, $\mathrm{HGA}_{\leq k}$, $\mathrm{HGI}_{=k}$ and $\mathrm{HGI}_{\leq k}$ we denote the hypergraph versions of these problems.

In [12], Schweitzer showed that $\mathrm{GI}_{\leq k}$ parameterized by k is in FPT, giving a $k^{\mathcal{O}(k)}\operatorname{poly}(n)$ time algorithm for it. Schweitzer's algorithm can easily be adapted to also solve $\mathrm{GA}_{\leq k}$.

Our results. In Section 3, we generalize Schweitzer's result [12] to hypergraphs with hyperedge size bounded by d, giving $(dk)^{\mathcal{O}(k^2)}\operatorname{poly}(N)$ time algorithms for $\mathrm{HGI}_{\leq k}$ and $\mathrm{HGA}_{\leq k}$ (throughout the paper we use N to denote the size $|V| \cdot |E|$ of the input hypergraphs). Consequently, for hypergraphs with $\operatorname{poly}(\log N)$ size hyperedges the problems remain in FPT when parameterized only by k. Note that although Hypergraph Isomorphism is known to be reducible to Graph Isomorphism, there is no known reduction from $\mathrm{HGI}_{=k}$ to $\mathrm{GI}_{=k}$ (or from $\mathrm{HGI}_{\leq k}$ to $\mathrm{GI}_{\leq k}$).

In Section 4 we consider $\mathrm{HGI}_{\leq k}$ for vertex-colored hypergraphs of unbounded hyperedge size. For hypergraphs with color classes of size at most b, we obtain a $(kb!)^{\mathcal{O}(k^2)}\operatorname{poly}(N)$ time algorithm.

In Section 6, we use color coding [1] to give FPT algorithms for $\mathrm{HGA}_{=k}$ parameterized by k and additionally either by the maximum hyperedge size d or by the maximum color class size b; the runtime bounds are the same as those mentioned above. In particular, it follows that $\mathrm{GA}_{=k}$ parameterized only by k is in FPT. For general hypergraphs we show that $\mathrm{HGA}_{=k}$ is in $\mathsf{FPT}^{\mathrm{GI}}$.

In contrast to the above results, if X is a colored graph with red and blue vertices and we want to test if X has a nontrivial automorphism π that moves at most k blue vertices then the problem is W[1]-hard. This confirms a claim from [5, Exercise 9.02] (see also [6, Exercise 20.3.2]). As the hint given there does not work out and we require quite different ideas, we have included our proof in Section 7.

Related work. The parametric dual to $\mathrm{GA}_{\leq k}$ asks for an automorphism that has at most k fixpoints. It is NP-complete even when restricted to $k = 0$, where it is known as the *fixpoint free automorphism problem* [11].

A fundamental problem in algorithmic coding theory is the *minimum weight codeword problem*: Given a system of linear equations $Ax = 0$ over $\mathbb{F}_2$ as instance, the problem is to find the minimum weight of a nonzero solution to it. It is known to be NP-hard even to approximate to a constant factor [15, 10]. The parameterized complexity of its decision version (called EVEN) is also well studied [7, 2].

EVEN. Given a binary matrix A defining the linear code $Ax = 0$ over $\mathbb{F}_2$ and a parameter k, is there is a non-zero codeword of weight at most k?

Whether EVEN is in FPT or W[1]-hard remains open. In contrast, EXACTEVEN (which asks for a codeword of weight exactly k) is known to be W[1]-hard [7, 2]. It is interesting to compare this to our result that $\mathrm{GA}_{=k}$ is in FPT. A natural generalization is to consider permutation groups $G \leq S_n$ as input, where G is given by a generating set S, and ask for a permutation in G of minimum weight. Formally, the problem of interest[1] is:

PERMCODE. Given S with $G = \langle S \rangle \leq S_n$ and a parameter $k \in \mathbb{N}$, does $G \setminus \{\mathrm{id}\}$ contain a permutation of weight at most k?

[1] It is called Hamming Distance Minimum Weight Problem in [4].

PermCode generalizes both $\mathrm{GA}_{\leq k}$ and Even. Indeed, an instance (X, k) of $\mathrm{GA}_{\leq k}$ can be transformed into an instance (S, k) of PermCode by computing a generating set S for the automorphism group of X. Computing this efficiently requires a GI oracle.

Let (A, k) be an instance of Even, where A has n columns. It can be reduced to an instance $(S, 2k)$ of PermCode with $G = \langle S \rangle \leq S_{2n}$ as follows: Using Gaussian elimination, compute from A an $n \times r$ matrix B over $\mathbb{F}_2$ whose columns generate the solution space of $Ax = 0$, where r is the solution-space dimension. For each column $(b_1, \ldots, b_n)^T$ of B, include in S the permutation in S_{2n} that, for $1 \leq i \leq n$, transposes $2i$ and $2i-1$ if $b_i = 1$ and fixes $2i$ and $2i-1$ if $b_i = 0$.

In particular, this shows that the hardness results for Even also apply to PermCode, even when the latter is restricted to groups with orbit size 2. In Section 5, we prove that PermCode for a permutation group G is reducible in polynomial time to PermCode for a subgroup G' of G whose orbits are of size bounded by the parameter k. We use this reduction in our algorithms for $\mathrm{HGA}_{=k}$.

2 Preliminaries

A permutation group G is a subgroup of $\mathrm{Sym}(V)$, where $\mathrm{Sym}(V)$ is the group of all permutations on a finite set V. If $V = [n]$ we denote $\mathrm{Sym}(V)$ by S_n. We denote the image of $v \in V$ under a permutation π by v^π and sometimes also by $\pi(v)$. For a subset $U \subseteq V$, we write $U^\pi = \pi(U) = \{u^\pi \mid u \in U\}$. We apply permutations from left to right so that $v^{\varphi\pi} = (v^\varphi)^\pi$.

We write $H \leq G$ when H is a subgroup of G. If φ is an element of G then $H\varphi$ denotes the coset $\{\pi\varphi \mid \pi \in H\}$. For $S \subseteq \mathrm{Sym}(V)$, the group $\langle S \rangle$ *generated* by S is the smallest subgroup of $\mathrm{Sym}(V)$ containing S. For an element $v \in V$, the set $\{v^\pi \mid \pi \in G\}$ is the *G-orbit* of v. In case $G = \langle\{\pi\}\rangle$ we call the resulting set $\{\pi^i(v) \mid i \in \mathbb{N}\}$ also the *π-orbit* of v.

The analysis of our algorithms relies on some measures on permutations. The *support* of $\pi \in \mathrm{Sym}(V)$ is $\mathrm{supp}(\pi) = \{u \in V \mid u^\pi \neq u\}$, i.e., the set of non-fixpoints of π. Its *complexity* $\mathrm{compl}(\pi)$ is the size of $\mathrm{supp}(\pi)$ minus the number of π-orbits having size at least 2. Equivalently, $\mathrm{compl}(\pi)$ is the minimum number of transpositions whose product is π.

▶ **Definition 2.1.** Let $G \leq \mathrm{Sym}(V)$ and $\pi \in \mathrm{Sym}(V)$; this includes the case $\pi = \mathrm{id}$.

1. A permutation $\sigma \in G\pi \setminus \{\mathrm{id}\}$ has *minimal support* in $G\pi$ if there is no $\varphi \in G\pi \setminus \{\mathrm{id}\}$ with $\mathrm{supp}(\varphi) \subsetneq \mathrm{supp}(\sigma)$.
2. A permutation $\sigma \in G\pi \setminus \{\mathrm{id}\}$ has *minimal complexity* in $G\pi$ if there are no $\varphi \in G \setminus \{\mathrm{id}\}$ and $\psi \in G\pi \setminus \{\mathrm{id}\}$ with $\sigma = \varphi\psi$ and $\mathrm{compl}(\sigma) = \mathrm{compl}(\varphi) + \mathrm{compl}(\psi)$, i.e., if for every way to express σ as the product of a minimum number of transpositions $\sigma = \tau_1 \cdots \tau_{\mathrm{compl}(\sigma)}$ and every $i \in \{2, \ldots, \mathrm{compl}(\sigma)\}$ it holds that $\tau_i \cdots \tau_{\mathrm{compl}(\sigma)} \notin G\pi$.

In particular, these notions apply to elements of the automorphism group $\mathrm{Aut}(X)$ of a graph X and to elements of the coset $\mathrm{Iso}(X, Y)$ of isomorphisms between two graphs X and Y.

▶ **Lemma 2.2.** *Let $G\pi$ be a coset of a permutation group G and let $\sigma \in G\pi \setminus \{\mathrm{id}\}$. Then for some $\ell \geq 1$ there are $\sigma_1, \ldots, \sigma_{\ell-1} \in G$ with minimal complexity in G and $\sigma_\ell \in G\pi$ with minimal complexity in $G\pi$ such that $\sigma = \sigma_1 \cdots \sigma_\ell$ and $\mathrm{supp}(\sigma_i) \subseteq \mathrm{supp}(\sigma)$ for each $i \in \{1, \ldots, \ell\}$. Moreover, all these inclusions become equalities if $\pi = \mathrm{id}$ and σ has minimal support in G.*

Proof. If σ has minimal complexity in $G\pi$, we have $\ell = 1$ and $\sigma_1 = \sigma$. Otherwise, there are $\varphi \in G \setminus \{\mathrm{id}\}$ and $\psi \in G\pi \setminus \{\mathrm{id}\}$ such that $\sigma = \varphi\psi$ and $\mathrm{compl}(\sigma) = \mathrm{compl}(\varphi) + \mathrm{compl}(\psi)$. This implies $\mathrm{supp}(\varphi) \subseteq \mathrm{supp}(\sigma)$ and $\mathrm{supp}(\psi) \subseteq \mathrm{supp}(\sigma)$, and these inclusions become equalities if $\pi = \mathrm{id}$ and σ has minimal support in G. If φ and ψ do not have minimal complexity in G and $G\pi$, respectively, they can be decomposed further. This process terminates, as φ and ψ both have lower complexity than σ. ◀

3 Bounded hyperedge size

Let $X = (V, E)$ and $Y = (V, E')$ be hypergraphs such that for each $e \in E$ we have $|e| \leq d$. We show that a nontrivial element $\pi \in \mathrm{Iso}(X, Y)$ of weight at most k, if it exists, can be found in $(dk)^{\mathcal{O}(k)} \operatorname{poly}(N)$ time.

This generalizes Schweitzer's result [12] shown for usual graphs, to hypergraphs of bounded hyperedge size. In order to find an isomorphism π between two given graphs such that π has support size at most k, Schweitzer's algorithm constructs π by iteratively adding transpositions that bring the input graphs closer to each other. To find suitable transpositions, it explores a search tree of depth k and degree $\binom{2k}{2}$. In each step, it computes a *candidate set* of at most $2k$ vertices and tries all transpositions among them. For each isomorphism π between the input graphs that has support size at most k, this candidate set contains two vertices that are in the same orbit of π. Vertex covers play a crucial role in computing the candidate set. To extend this algorithm to hypergraphs of bounded hyperedge size, we need a generalization of vertex covers.

▶ **Definition 3.1.** Let X be a hypergraph on a vertex set V, and let $C \subseteq V$. A hyperedge e of X is *q-covered* by C if $|e \cap C| \geq \min(q, |e|)$. The set C is a *q-strong vertex cover* of X if it q-covers every hyperedge of X.

▶ **Definition 3.2.** Let X and Y be hypergraphs on a vertex set V. $X \triangle Y$ is the hypergraph with edge set $\mathrm{E}(X) \triangle \mathrm{E}(Y)$ and having as vertex set the union of all edges in $\mathrm{E}(X) \triangle \mathrm{E}(Y)$.

For the following results, let X and Y be two non-identical hypergraphs on the same vertex set V and let π be an isomorphism from X to Y with $|\mathrm{supp}(\pi)| \leq k$.

▶ **Lemma 3.3.** *Let C be a q-strong vertex cover of $X \triangle Y$ such that no two distinct points in C belong to the same π-orbit. Then for no hyperedge e of $X \triangle Y$ with $|e \cap C| \leq q$ it holds that $e \cap \mathrm{supp}(\pi) \subseteq C$.*

Proof. Let $e = \{u_1, \ldots, u_t\} \in \mathrm{E}(X) \triangle \mathrm{E}(Y)$ be a hyperedge such that $u_1, \ldots, u_s \in C$ and $u_{s+1}, \ldots, u_t \notin C$ for some $s \leq q$. Suppose, to the contrary, that $e \cap \mathrm{supp}(\pi) \subseteq C$, i.e., $u_i^\pi = u_i$ for $i \in \{s+1, \ldots, t\}$. W.l.o.g. let $e \in \mathrm{E}(X) \setminus \mathrm{E}(Y)$. Let ℓ be the length of the π-orbit of e. It is the least positive integer such that $\pi^\ell(e) = e$. We claim that $\pi^{i-1}(e) \in \mathrm{E}(X)$ implies $\pi^i(e) \in \mathrm{E}(X)$, for $1 \leq i < \ell$. This contradicts $\pi^{\ell-1}(e) \notin \mathrm{E}(X)$, which follows from $\pi^\ell(e) = e \notin \mathrm{E}(Y)$ because π is an isomorphism from X to Y.

To prove the claim, fix any $j \in \{1, \ldots, s\}$ with $\pi^i(u_j) \neq u_j$. Such a j must exist because otherwise $\pi^i(e) = e$ for $i < \ell$, contradicting the definition of ℓ. As $u_j \in C$ and no two distinct points in C belong to the same π-orbit, it follows that $\pi^i(u_j) \notin C$, implying that $\pi^i(e)$ is not q-covered by C and thus cannot be contained in $\mathrm{E}(X) \triangle \mathrm{E}(Y)$. Finally, as π is an isomorphism from X to Y, $\pi^{i-1}(e) \in \mathrm{E}(X)$ implies that $\pi^i(e) \in \mathrm{E}(Y)$, but since $\pi^i(e) \notin \mathrm{E}(X) \triangle \mathrm{E}(Y)$, it follows that $\pi^i(e) \in \mathrm{E}(X)$. ◀

▶ **Lemma 3.4.** *If C is a q-strong vertex cover of $X \triangle Y$ such that no two distinct points in C belong to the same π-orbit, then $C \cup \mathrm{supp}(\pi)$ is a $(q+1)$-strong vertex cover of $X \triangle Y$.*

Proof. For all hyperedges e of $X \triangle Y$ with $|e \cap C| \leq q$ we have $e \cap \mathrm{supp}(\pi) \nsubseteq C$ by Lemma 3.3. Hence, $\mathrm{supp}(\pi)$ covers at least one additional vertex in each such hyperedge. ◀

Algorithm 1 $\texttt{CandidateSet}_{k,d}(X \triangle Y)$

```
Input: The symmetric difference $X \triangle Y$ of two hypergraphs $X \neq Y$ on vertex
    set $V = [n]$ containing some hyperedge of size at most $d$
Output: A candidate set $C$ of size at most $dk$ that contains for any isomorphism
    $\pi \in \mathrm{Iso}(X, Y)$ with $|\mathrm{supp}(\pi)| \leq k$ two elements belonging to the same $\pi$-orbit
$C_0 \leftarrow \varnothing$; $q \leftarrow 0$
while $X \triangle Y$ has a $(q+1)$-strong vertex cover $C_{q+1} \supseteq C_q$ with $|C_{q+1}| \leq |C_q| + k$
    and $q < d$ do $q \leftarrow q + 1$
return $C_q$
```

▶ **Lemma 3.5.** *If C is a q-strong vertex cover for $X \triangle Y$ and some hyperedge e of $X \triangle Y$ has size at most q, then C contains two distinct points belonging to the same π-orbit.*

Proof. As C is a q-strong vertex cover and e has size at most q it follows that C contains e. If no two distinct points in C belong to the same π-orbit, then Lemma 3.3 implies that C does not even contain $e \cap \mathrm{supp}(\pi)$, a contradiction. ◀

Lemmas 3.4 and 3.5 suggest the following algorithm for computing a candidate set; it can be plugged into Schweitzer's isomorphism test [12, Algorithm 1], cf. Algorithm 2 below.

Observe that, by construction, Algorithm 1 returns a candidate set C_q of size at most dk. The following lemma shows that C_q also has the other desired property and gives an FPT bound on the running time of the procedure.

▶ **Lemma 3.6.** *If X, Y are hypergraphs such that $X \triangle Y$ has an hyperedge of size at most d, and π is an isomorphism from X to Y with $|\mathrm{supp}(\pi)| \leq k$, then the set C_q returned by $\texttt{CandidateSet}_{k,d}(X \triangle Y)$ contains two vertices in the same π-orbit. Moreover, the running time of the procedure is $\mathcal{O}(d^{k+1}\,\mathrm{poly}(N))$, where N is the length of the encoding of $X \triangle Y$.*

Proof. Observe that $C_0 = \varnothing$ is a 0-strong vertex cover. Lemma 3.4 guarantees that for $q < d$, the condition of the while-loop can only be violated if C_q contains two vertices in the same π-orbit. Furthermore, Lemma 3.5 guarantees that this also holds in the case that q reaches the value d.

It remains to show the bound on the running time. The only critical step is to extend a q-strong vertex cover C_q of $X \triangle Y$ to a $(q+1)$-strong one C_{q+1} in Line 6. This can be reduced to finding a hitting set S of size at most k for the hypergraph $\{e \in \mathrm{E}(X) \triangle \mathrm{E}(Y) \mid e \setminus C_q \neq \varnothing \wedge |e \cap C_q| = q\}$ and taking $C_{q+1} = C_q \cup S$. The latter problem is fixed parameter tractable by the classical bounded search tree technique in time $\mathcal{O}(d^k\,\mathrm{poly}(N))$ (see, e.g., [8, Theorem 1.14]). ◀

The following is a search version of Schweitzer's algorithm adapted to hypergraphs.

Finding *all* isomorphisms of support size at most k is not possible in FPT time; e.g. between two complete graphs there are $\Omega(n^k)$ of them. However, the following lemma shows that Algorithm 2 finds a meaningful subset of them.

▶ **Lemma 3.7.** *Let $X \neq Y$ be two hypergraphs on the vertex set V with hyperedge size bounded by d, and let $c, k \in \mathbb{N}$. Then the set returned by $\texttt{ISO}_{k,d}(X, Y, c)$ is a subset of $\mathrm{Iso}(X, Y)$ containing every complexity-minimal isomorphism φ from X to Y with $|\mathrm{supp}(\varphi)| \leq k$ and $\mathrm{compl}(\varphi) \leq c$. Further, $\texttt{ISO}_{k,d}(X, Y, c)$ runs in time $\mathcal{O}\big((dk)^{\mathcal{O}(ck)}\,\mathrm{poly}(N)\big)$, where N is the length of the encodings of X and Y.*

Algorithm 2 $\texttt{ISO}_{k,d}(X,Y,c)$

```
Input: Two hypergraphs $X$ and $Y$ on vertex set $V=[n]$ with hyperedge size bounded
        by $d$ and a natural number $c \le k$ that bounds the recursion depth
Output: A set $P$ of isomorphisms from $X$ to $Y$
if $X \triangle Y$ is empty then return $\{\mathrm{id}\}$
$P \leftarrow \varnothing$
if $c > 0$ then
    $C \leftarrow \texttt{CandidateSet}_{k,d}(X \triangle Y)$
    foreach $v_1, v_2 \in C$ with $v_1 \neq v_2$ do
        $P' \leftarrow \texttt{ISO}_{k,d}(X, Y^{(v_1v_2)}, c-1)$
        $P \leftarrow P \cup \{\varphi' \cdot (v_1v_2) \mid \varphi' \in P'\}$ // compose with $(v_1v_2) \in \mathrm{Iso}(Y^{(v_1v_2)}, Y)$
return $\{\varphi \in P \mid |\mathrm{supp}(\varphi)| \le k\}$
```

Proof. Clearly, the set returned by $\texttt{ISO}_{k,d}(X,Y,c)$ only contains isomorphisms from X to Y.

It remains to show that every complexity-minimal isomorphism φ from X to Y with $|\mathrm{supp}(\varphi)| \le k$ and $\mathrm{compl}(\varphi) \le c$ is in this set. By Lemma 3.6, the candidate set C contains two vertices v_1 and v_2 that belong to the same orbit of φ. Thus we get $\varphi = \varphi' \cdot (v_1v_2)$ for some φ' with $\mathrm{compl}(\varphi') = \mathrm{compl}(\varphi) - 1$. Note that if $\varphi' \neq \mathrm{id}$ then φ' has minimal complexity in $\mathrm{Iso}(X, Y^{(v_1v_2)})$. Indeed, $\varphi' = \varphi_1\varphi_2'$ with $\varphi_1 \in \mathrm{Aut}(X) \setminus \{\mathrm{id}\}$, $\varphi_2' \in \mathrm{Iso}(X, Y^{(v_1v_2)})$ and $\mathrm{compl}(\varphi') = \mathrm{compl}(\varphi_1) + \mathrm{compl}(\varphi_2')$ would imply $\varphi = \varphi_1\varphi_2$ with $\varphi_1 \in \mathrm{Aut}(X) \setminus \{\mathrm{id}\}$, $\varphi_2 = \varphi_2' \cdot (v_1v_2) \in \mathrm{Iso}(X,Y)$ and $\mathrm{compl}(\varphi) = \mathrm{compl}(\varphi_1) + \mathrm{compl}(\varphi_2)$, contradicting that φ has minimal complexity in $\mathrm{Iso}(X,Y)$.

Now it suffices to show that when v_1 and v_2 are considered in the loop starting in Line 8, the permutation φ' is contained in the set returned by $\texttt{ISO}_{k,d}(X, Y^{(v_1v_2)}, c-1)$. This follows by induction on the complexity of φ, with the base case $\varphi = \mathrm{id}$ taken care of by Line 4.

To show the bound on the running time, it suffices to observe that the depth of the recursion tree is c and, as $|C| \le dk$, there are $\mathcal{O}\big((dk)^{2c}\big)$ recursive calls in total. In each call, it takes $\mathcal{O}(d^{k+1}\,\mathrm{poly}(N))$ time to compute C (Lemma 3.6), and the rest of the work is linear in the size of the recursion tree. ◀

We remark that an isomorphism φ in the set P returned by $\texttt{ISO}_{k,d}$ has minimal complexity if and only if there is no $\sigma \in P$ with $\mathrm{compl}(\sigma) < \mathrm{compl}(\varphi)$ and $\mathrm{compl}(\varphi) = \mathrm{compl}(\varphi\sigma^{-1}) + \mathrm{compl}(\sigma)$; note that this condition can be checked in polynomial time.

▶ **Theorem 3.8.** *Given two hypergraphs $X \neq Y$ with hyperedge size bounded by d, it can be decided in time $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)\big)$ whether there is an isomorphism from X to Y of support size at most k, where N is the length of the encodings of X and Y.*

Proof. The algorithm runs $\texttt{ISO}_{k,d}(X,Y,k)$ and accepts if the returned set is not empty. Every isomorphism φ from X to Y with $|\mathrm{supp}(\varphi)| \le k$ trivially satisfies $\mathrm{compl}(\varphi) \le k$ and can be decomposed by Lemma 2.2, obtaining a minimal-complexity isomorphism φ' from X to Y with $\mathrm{supp}(\varphi') \subseteq \mathrm{supp}(\varphi)$. By Lemma 3.7, φ' is in the set returned by $\texttt{ISO}_{k,d}(X,Y,k)$. ◀

We conclude this section by showing how to decide the problem $\mathrm{HGA}_{\le k}$ for hypergraphs with hyperedge size bounded by d in time $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)\big)$.

▶ **Theorem 3.9.** *Given a hypergraph X on n vertices with hyperedge size bounded by d, the algorithm $\texttt{AUT}_{k,d}(X)$ enumerates all complexity-minimal automorphisms of X with support size at most k (plus possibly some more that do not have minimal complexity) in $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)\big)$ time.*

Algorithm 3 $\texttt{AUT}_{k,d}(X)$

```
Input: A hypergraph $X$ on vertex set $V = [n]$ with hyperedge size bounded by $d$.
Output: A set $P$ of automorphisms of $X$.
$P \leftarrow \varnothing$
foreach $v_1, v_2 \in V$ do
    $P' \leftarrow \texttt{ISO}_{k,d}(X, X^{(v_1 v_2)}, k-1)$
    $P \leftarrow P \cup \{\varphi' \cdot (v_1 v_2) \mid \varphi' \in P'\}$ // compose with the isomorphism $(v_1 v_2)$
                                                  from $X^{(v_1 v_2)}$ to $X$
return $\{\varphi \in P \mid |\mathrm{supp}(\varphi)| \le k\}$
```

Proof. Clearly, all elements in the returned set are automorphisms of X with support size at most k. The fact that every complexity-minimal automorphism φ with $|\mathrm{supp}(\varphi)| \le k$ is found follows from Lemma 3.7 using the same decomposition argument as in the proof of the latter. Also the time bound follows immediately from Lemma 3.7. ◀

We remark that there is no FPT algorithm that enumerates all automorphisms with support size at most k (including those that do not have minimal support), as e.g. K_n has $\sum_{i=1}^{k} \binom{n}{k} k!$ such automorphisms, which is not an FPT number. However, by Lemma 2.2 each $\sigma \in \mathrm{Aut}(X) \setminus \{\mathrm{id}\}$ with support size at most k can be written as a product of minimal-complexity automorphisms of X with support size at most k. Thus $\sigma \in \langle S \rangle$, where S is the set returned by $\texttt{AUT}_{k,d}(X)$.

4 Bounded color class size

In this section we consider vertex colored hypergraphs using the maximum color class size b as an additional parameter. We call a colored hypergraph *b-bounded* if the size of each of its color classes is bounded by b. We give an FPT algorithm for $\mathrm{HGI}_{\le k}$ for b-bounded hypergraphs. More precisely, given hypergraphs $X = (V, E)$ and $Y = (V, E')$ along with a partition $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of V into (pairwise disjoint) color classes C_i with $|C_i| \le b$, our algorithm will compute in time $\mathcal{O}\big((kb!)^{\mathcal{O}(k^2)} \mathrm{poly}(N)\big)$ a color-preserving isomorphism from X to Y of weight at most k (if it exists). For a permutation $\pi \in \mathrm{Sym}(V)$ let $\mathcal{C}[\pi] = \{C_i \in \mathcal{C} \mid \exists v \in C_i : v^\pi \ne v\}$ be the subset of color classes that intersect $\mathrm{supp}(\pi)$. Suppose $\pi \in \mathrm{Iso}(X, Y)$ is an isomorphism of weight at most k and that $\mathcal{C}[\pi] = \{C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}\}$, $\ell \le k/2$. In order to search for the color classes in $\mathcal{C}[\pi]$ we will apply the color-coding method of Alon-Yuster-Zwick [1]. Consider the FKS [9] family $\mathcal{H}$ of perfect hash functions $h\colon [m] \to [\ell]$. We can use each $h \in \mathcal{H}$ to partition the color classes $C_1, C_2, \ldots, C_m$ into ℓ bags $\mathcal{B}_1, \ldots, \mathcal{B}_\ell$, where $\mathcal{B}_j$ contains all color classes C_i labeled with $h(i) = j$. Since $\mathcal{H}$ is a perfect family of hash functions, some $h \in \mathcal{H}$ is good for π in the sense that the color classes $C_{i_1}, \ldots, C_{i_\ell}$ all have distinct labels in $[\ell]$, i.e., $\{h(i_1), h(i_2), \ldots, h(i_\ell)\} = [\ell]$.

For $j \in [\ell]$, we define the hypergraphs $X_j = (V_j, E_j)$ and $Y_j = (V_j, E'_j)$ as follows: $V_j = \bigcup \mathcal{B}_j$, $E_j = \{e \cap V_j \mid e \in E\}$, and $E'_j = \{e \cap V_j \mid e \in E'\}$.

Notice that if $h \in \mathcal{H}$ is good for the target isomorphism $\pi \in \mathrm{Iso}(X, Y)$, then the restriction of π to V_j is an isomorphism from X_j to Y_j that moves only vertices of exactly one color class in $\mathcal{B}_j$. We say that a color class $C_i \in \mathcal{B}_j$ *allows to move* a hyperedge $e \in E$ if there is an isomorphism $\pi \in \mathrm{Iso}(X_j, Y_j)$ that moves only vertices of color class C_i and $(e \cap V_j)^\pi \ne e \cap V_j$. We denote the set of all color classes $C_i \in \mathcal{B}_j$ that allow to move e by $\mathrm{Movers}_j(e)$ and define

$\text{Movers}(e) = \bigcup_{j=1}^{\ell} \text{Movers}_j(e)$. The next claim shows that the size of $\text{Movers}(e)$ is bounded by $\sum_{j=1}^{\ell} \log|E'_j| \le \ell \log|E'| = \ell \log|E|$.

▶ **Claim.** *For each $j \in [\ell]$ and each hyperedge $e \in E$, there at most $\log|E'_j|$ many color classes $C_i \in \mathcal{B}_j$ that allow to move e.*

Proof of the claim. Suppose there are $t > \log|E'_j|$ many color classes $C_{j_1}, \dots, C_{j_t} \in \mathcal{B}_j$ that allow to move e. Let $\pi_{j_r} \in \text{Iso}(X_j, Y_j)$, for $r \in [t]$, be corresponding isomorphisms such that π_{j_r} fixes all color classes in $\mathcal{B}_j$ except C_{j_r} and $(e \cap V_j)^{\pi_{j_r}} \neq e \cap V_j$. Clearly, for each subset $T \subseteq [t]$, the product $\prod_{r \in T} \pi_{j_r}$ is in $\text{Iso}(X_j, Y_j)$ and all the images $\left(\prod_{r \in T} \pi_{j_r}\right)(e \cap V_j)$, for $T \subseteq [t]$, are distinct. Hence the total number of distinct edges in E'_j, generated as such images of $(e \cap V_j)$, will be $2^t > |E'_j|$ which is impossible. ◀

Now we show the main result of this section. Before proceeding we introduce a definition that is important for the rest of the paper.

▶ **Definition 4.1.** Let $X = (V, E)$ and $Y = (V, E)$ be two hypergraphs with color class set $\mathcal{C} = \{C_1, \dots, C_m\}$ and let $\pi \in \text{Sym}(V)$. For a subset $\mathcal{C}' \subseteq \mathcal{C}[\pi]$ define the permutation $\pi_{\mathcal{C}'}$ as

$$\pi_{\mathcal{C}'}(v) = \begin{cases} v^{\pi}, & \text{if } v \in \bigcup \mathcal{C}', \\ v, & \text{if } v \notin \bigcup \mathcal{C}'. \end{cases}$$

An isomorphism $\pi \neq \text{id}$ from X to Y is said to be *color-class-minimal*, if for any set $\mathcal{C}'$ with $\varnothing \subsetneq \mathcal{C}' \subsetneq \mathcal{C}[\pi]$, the permutation $\pi_{\mathcal{C}'}$ is not in $\text{Iso}(X, Y)$.

We notice that all isomorphisms having minimal support are also color-class-minimal. Another immediate consequence of Definition 4.1 is the following lemma for decomposing automorphisms of hypergraphs.

▶ **Lemma 4.2.** *Let $X = (V, E)$ be a hypergraph with color class set $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$. Each nontrivial automorphism π of X can be written as a product of nontrivial color-class-minimal automorphisms $\pi_1, \pi_2, \dots, \pi_\ell$ of X, where the support color class sets $\mathcal{C}[\pi_i]$, for $1 \le i \le \ell$, are pairwise disjoint and form a partition of the support color class set $\mathcal{C}[\pi]$.*

We now describe an algorithm that computes isomorphisms between hypergraphs by building them color class by color class. Let $\{\mathcal{B}_1, \dots, \mathcal{B}_\ell\}$ be a partition of the color class set $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ and let $k = k_1 + \cdots + k_\ell$. Then we call a permutation $\pi \in \text{Sym}(V)$ $(k_1, \dots, k_\ell)$-*good* for $\{\mathcal{B}_1, \dots, \mathcal{B}_\ell\}$, if each bag $\mathcal{B}_j$ contains exactly one of the color classes in $\mathcal{C}[\pi]$ (say C_{i_j}) and $|\text{supp}(\pi) \cap C_{i_j}| = k_j$ for $j = 1, \dots, \ell$.

The following lemma shows that Algorithm 4 finds a meaningful set of isomorphisms.

▶ **Lemma 4.3.** *Let $X \neq Y$ be two b-bounded hypergraphs. Then the set returned by the algorithm $\texttt{ColoredIso}_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \text{id})$ contains all color-class-minimal isomorphisms φ from X to Y that are $(k_1, \dots, k_\ell)$-good for $\mathcal{B}_1, \dots, \mathcal{B}_\ell$. Furthermore, Algorithm 4 runs in time $\mathcal{O}\big((b!)^k \operatorname{poly}(N)\big)$.*

Proof. Let φ be such an isomorphism, and let $\pi = \varphi_{\mathcal{C}'}$ for some subset $\mathcal{C}' \subsetneq \mathcal{C}[\varphi]$ of the color classes that intersect $\text{supp}(\varphi)$. We show by induction on the number of color classes in $\mathcal{C}[\varphi] \setminus \mathcal{C}'$ touched by $\text{supp}(\varphi)$ but not by $\text{supp}(\pi)$ that $\texttt{ColoredIso}_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \pi)$ finds φ. If this number is 0, we have $\varphi = \pi$, and Line 5 ensures that φ is found. Otherwise, the color-class-minimality of φ implies that $\pi \notin \text{Iso}(X, Y)$. Since φ is $(k_1, \dots, k_\ell)$-good for $\mathcal{B}_1, \dots, \mathcal{B}_\ell$, for any hyperedge $e \in \text{E}(X)$ with $e^{\pi} \notin \text{E}(Y)$, there is a bag $\mathcal{B}_j$ with $\text{supp}(\pi) \cap \bigcup \mathcal{B}_j = \varnothing$

Algorithm 4 `ColoredIso`$_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \pi)$

```
Input: Hypergraphs $X$ and $Y$ on vertex set $V = C_1 \cup \dots \cup C_m$ with color classes
        $C_i$ of size $|C_i| \le b$ and a permutation $\pi \in \mathrm{Sym}(V)$
Output: A set $P$ of color-preserving isomorphisms from $X$ to $Y$
if $\pi$ is a $(k_1,\dots,k_\ell)$-good isomorphism from $X$ to $Y$ for $\mathcal{B}_1,\dots,\mathcal{B}_\ell$ then
    return $\{\pi\}$
else
    $P \leftarrow \varnothing$
    pick a hyperedge $e \in \mathrm{E}(X)$ with $e^\pi \notin \mathrm{E}(Y)$
    foreach bag $\mathcal{B}_j$ with $\mathrm{supp}(\pi) \cap \bigcup \mathcal{B}_j = \varnothing$ do
        foreach color class $C \in \mathrm{Movers}_j(e)$ do
            foreach $\sigma \in \mathrm{Sym}(V)$ with $\mathrm{supp}(\sigma) \subseteq C$ and $|\mathrm{supp}(\sigma)| = k_j$ do
                $P \leftarrow P \cup$ ColoredIso$_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \pi\sigma)$
    return $P$
```

and a color class $C \in \mathrm{Movers}_j(e)$ such that $|C \cap \mathrm{supp}(\varphi)| = k_j$. By the inductive hypothesis, φ is found in the iteration of the inner loop where $\sigma = \varphi_{\{C\}}$.

To show the bound on the running time, it suffices to observe that the recursion tree has degree bounded by $|\mathrm{Movers}(e)| \cdot |\mathrm{Sym}(C)| \le k \log|E| \cdot b!$ and depth bounded by $k/2$, and that all steps in each recursive call can be implemented in $\mathcal{O}(N)$ time. ◀

To compute all color-class-minimal isomorphisms between two b-bounded hypergraphs $X \ne Y$ of support size exactly k, we add an initial branching over all $\ell \in [k]$ and all FKS partitions $\mathcal{B}_1, \dots, \mathcal{B}_\ell$ of $\mathcal{C}$ and trying all partitions $k = k_1 + \dots + k_\ell$. This adds only an extra $k^{\mathcal{O}(k)}$ factor and yields the algorithm `ColoredIso`$_{k,b}(X, Y)$ for computing all color-class-minimal isomorphisms from X to Y of support size exactly k. Further, we can also handle the case $X = Y$, i.e., computing all color-class-minimal automorphisms of support size exactly k, by adding another initial branching over all color classes to choose the first one that is permuted. This adds the number of vertices n as an additional factor to the running time and yields the algorithm `ColoredAut`$_{k,b}(X)$ for computing all color-class-minimal automorphisms of X with support size exactly k.

▶ **Theorem 4.4.** *Given two b-bounded hypergraphs X and Y on vertex set $V = [n]$ and $k \in \mathbb{N}$, the set of all color-class-minimal isomorphisms from X to Y with support size exactly k can be computed in $\mathcal{O}\big((kb!)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)\big)$ time, where N is the size of the input hypergraphs.*

As each isomorphism having minimum support size k is also color-class-minimal, we can state the following corollary.

▶ **Corollary 4.5.** *There is an algorithm for $\mathrm{HGI}_{\le k}$ that decides for two given b-bounded hypergraphs X and Y in time $\mathcal{O}\big((kb!)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)\big)$ if there is an isomorphism from X to Y of weight at most k and computes such an isomorphism if it exists.*

5 Shrinking orbits while preserving small weight permutations

In this section, we show how to reduce instances of the PermCode problem to other instances on subgroups with orbit sizes bounded by the parameter k.

Algorithm 5 PermCode reduction

```
Input: A generating set $A$ of a group $G = \langle A \rangle \leq S_n$ and a parameter $k$
Output: A generating set $B$ for a subgroup $\langle B \rangle \leq G$ with orbits of size bounded by $k$
while $G$ has an orbit $O$ of size more than $k$ do
    pick $u \in O$
    compute a generating set $B$ for $G_u$ using the Schreier-Sims algorithm [13]
    $G \leftarrow \langle B \rangle$
return $B$
```

▶ **Lemma 5.1.** *There is a polynomial-time algorithm that on input an instance (A, k) of* PermCode, *where $G = \langle A \rangle \leq S_n$ and $k \in \mathbb{N}$, computes an equivalent instance (B, k) with $G' = \langle B \rangle \leq G$ and the property that every orbit of G' has size bounded by k. Moreover, for every $\pi \in G$ with $|\mathrm{supp}(\pi)| \leq k$ there is a $\pi' \in G'$ with $|\mathrm{supp}(\pi')| = |\mathrm{supp}(\pi)|$.*

Proof. The reduction is an application of the following simple group-theoretic observation.

▶ **Claim.** *Let O be a G-orbit of size more than k and let u be any point in O. Then, for any element $\pi \in G$ of support size $|\mathrm{supp}(\pi)| \leq k$, there is an element $\pi' \in G_u$ with $|\mathrm{supp}(\pi')| = |\mathrm{supp}(\pi)|$ where $G_u = \{\varphi \in G \mid u^\varphi = u\}$.*

To prove the claim, we only have to consider the case that $u \in \mathrm{supp}(\pi)$, since otherwise $\pi \in G_u$. Let v be a point in $O \setminus \mathrm{supp}(\pi)$ and let $\sigma \in G$ such that $v^\sigma = u$. Then it follows immediately that the permutation $\pi' = \sigma^{-1}\pi\sigma$ belongs to G_u and has the same support size as π. The claimed reduction is given by the following simple algorithm that stabilizes points in orbits of size larger than k until no such orbits exist anymore.

The correctness of the reduction is a direct consequence of the claim above. Further, the reduction is polynomial-time computable as the Schreier-Sims algorithm has polynomial running time and the while loop runs for at most n iterations. ◀

6 Exact support size

In this section we show that the problem HGA$_{=k}$ of computing automorphisms of support size exactly k is also solvable in FPT for hypergraphs having hyperedges or color classes of bounded size. We will first focus on hypergraphs having hyperedges of size bounded by d and show that the HGA$_{=k}$ problem for such graphs is FPT reducible to the HGA$_{=k}$ problem for k-bounded hypergraphs.

We stress that Schweitzer's algorithm [12], for ordinary graphs, cannot guarantee finding isomorphisms of weight exactly k. This is because exact weight k isomorphisms (and automorphisms), unless of minimal complexity, may not get enumerated in either Schweitzer's algorithm [12] or our generalization in Section 3 to bounded hyperedge size hypergraphs.

However, each weight k automorphism is expressible as a product of automorphisms enumerated by the search. We state this as a simple corollary of Lemma 2.2 and Theorem 3.9.

▶ **Corollary 6.1.** *Let X be a hypergraph with hyperedges of size bounded by d and let S be the set returned by the algorithm $\mathtt{AUT}_{k,d}(X)$. Then the subgroup $G = \langle S \rangle$ of $\mathrm{Aut}(X)$ contains all automorphisms of X of weight at most k. In particular, G includes all automorphisms of weight exactly k.*

▶ **Lemma 6.2.** *Let $X = (V, E)$ be a hypergraph with hyperedges of size bounded by d. In FPT time we can reduce the search for an exact weight k automorphism of X to finding an exact weight k automorphism of X that is vertex colored with k-bounded color classes.*

Proof. Let $X = (V, E)$ be a hypergraph with hyperedges of size bounded by d. Applying Theorem 3.9, we can enumerate the set B of all complexity-minimal automorphisms of X that are of weight at most k. Clearly, any weight at most k automorphism (including the exact weight k ones) of X is in the subgroup $G = \langle B\rangle$ of $\mathrm{Aut}(X)$. Next, we can apply Lemma 5.1 to the permutation group G and replace it by the subgroup G' whose orbits are of size at most k such that if G has an exact weight k automorphism then G' also has an exact weight k automorphism.

We can designate the orbits of G' (which are all of size at most k) as color classes of X to obtain a k-bounded hypergraph. I.e. we assign different colors to vertices that are in different orbits of G' and consider only color-preserving automorphisms. Clearly, the exact weight k automorphisms of X that survive in G' are also color-preserving automorphisms of this k-bounded colored version of hypergraph X. ◀

▶ **Theorem 6.3.** *There is an algorithm for* $\mathrm{HGA}_{=k}$ *for b-bounded hypergraphs $X = (V, E)$ that decides in time $(kb!)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ if there is an exact weight k automorphism of X and computes such an automorphism if it exists.*

Proof. We apply the algorithm of Theorem 4.4 to enumerate the set A of all color-class-minimal automorphisms of X of weight at most k in time $(kb!)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$. By Lemma 4.2, we know that for any exact weight k color-preserving automorphism π of X there is an $\ell \leq k$ such that π is expressible as $\pi = \pi_1\pi_2\ldots\pi_\ell$, where each π_s is a color-class-minimal automorphism of X of weight at most k and $\mathcal{C}[\pi_s]$, for $1 \leq s \leq \ell$, forms a partition of $\mathcal{C}[\pi]$. I.e. each candidate π_s is in the enumerated set A. Let $\mathcal{C}[\pi] = \{C_{i_1}, C_{i_2}, \ldots, C_{i_r}\}$, where $r \leq k$ (because only k vertices are moved by π).

We will again use color coding [1] to search for the π_s, $1 \leq s \leq \ell$. Let $\mathcal{H}$ be the FKS family of perfect hash functions $h\colon [m] \to [r]$, where m is the total number of color classes in X, and $r \leq k$, as above, is the number of color classes in $\mathcal{C}[\pi]$. For each $j \in [r]$ define the bags $\mathcal{B}_j = \{C_i \mid h(i) = j\}$. By the property of the FKS family, there is some h such that for $j = 1, \ldots, r$, each bag $\mathcal{B}_j$ contains exactly one color class from $\mathcal{C}[\pi]$. Notice the following simple claim.

▶ **Claim.** *The total number of partitions of the set $\{i_1, i_2, \ldots, i_r\}$ into ℓ sets is bounded by $2^{r\ell}$ and we can cycle through all such partitions in time $2^{\mathcal{O}(r\ell)}$.*

One of the $2^{r\ell}$ many partitions, say $I_1 \sqcup I_2 \sqcup \cdots \sqcup I_\ell$ of $\{i_1, i_2, \ldots, i_r\}$ will be the partition such that $\mathcal{C}[\pi_s] = \{C_i \mid i \in I_s\}, 1 \leq s \leq \ell$. Assume we are considering this partition $I_1 \sqcup I_2 \sqcup \cdots \sqcup I_\ell$. Then in $|A|$ time we partition A into subsets A_s, with $1 \leq s \leq \ell$, defined by

$$A_s = \{\psi \in A \mid C_i \in \mathcal{C}[\psi] \text{ iff } i \in I_s\}.$$

Finally, we can try all partitions $k = k_1 + k_2 + \cdots + k_\ell$ (there are at most 2^{2k} many). For each partition we look for an element π_s of weight exactly k_s in A_s in time $|A_s|\,\mathrm{poly}(N)$. ◀

▶ **Corollary 6.4.** *There is an algorithm for* $\mathrm{HGA}_{=k}$ *for hypergraphs X with hyperedges of size bounded by d that decides in time $d^{\mathcal{O}(k)}2^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ if there is an exact weight k automorphism of X and computes such an automorphism if it exists.*

Proof. By Lemma 6.2 we can transform $X = (V, E)$ into a hypergraph with k-bounded color classes in time $(dk)^{\mathcal{O}(k)}\,\text{poly}(|V|, |E|)$ such that X has an exact weight k automorphism if and only if there is an exact weight k color-preserving automorphism of X. Hence, we can apply the algorithm of Theorem 6.3 to solve the problem. ◀

▶ Remark. For the general case of hypergraphs of unbounded edge size it is easy to see that we have an FPT^{GI} algorithm for the problem $\text{HGA}_{=k}$ to find an exact weight k automorphism (and hence it is unlikely to be W[1]-hard). We use the GI oracle in order to first compute a generating set for $\text{Aut}(X)$ and then using Lemma 5.1 we can reduce the instance to a k-bounded hypergraph to which Theorem 6.3 can be applied.

7 The Colored Graph Automorphism problem is W[1]-hard

The ColoredGraphAutomorphism problem [5, p. 460] is defined as follows: given a red/blue graph $X = (\mathcal{R}, \mathcal{B}, E)$ and the parameter $k \in \mathbb{N}$, does X have a color-preserving automorphism whose support contains exactly k vertices in $\mathcal{B}$ (the blue vertices)?

Exercise 9.0.2 in [5] (Exercise 20.3.2 in [6]) is to show that this problem is W[1]-hard. The book gives a reduction, as hint, from the Weighted Antimonotone 2-CNF-Sat problem. The exercise is to show that the input formula has a weight k satisfying assignment if and only if the constructed graph has an automorphism that moves exactly $2k$ blue vertices. This hint does not work: the resulting graph has no nontrivial automorphisms. However, we now present a proof for the W[1]-hardness of ColoredGraphAutomorphism by giving an FPT reduction from ExactEven, which is known to be W[1]-hard [7].

▶ **Theorem 7.1.** ExactEven *is FPT reducible to* ColoredGraphAutomorphism.

Proof. The reduction is based on a gadget in [3, 14] for simulating a circuit with parity gates as an instance of Graph Isomorphism. Let $\oplus$ denote the addition in $\mathbb{F}_2$. We define the undirected graph $X_2 = (V, E)$ by $V = \{x_a, y_a, z_a \mid a \in \{0,1\}\} \cup \{u_{a,b} \mid a, b \in \{0,1\}\}$ and

$$E = \{(x_a, u_{a,b}), (y_b, u_{a,b}), (u_{a,b}, z_{a\oplus b}) \mid a, b \in \{0,1\}\}.$$

This graph gadget simulates a fan-in 2 parity gate. The x and y vertices encode the inputs of the gate while the z vertices encode the output. Any automorphism in the graph mapping the input nodes corresponding to any 0-1 input values for the gate, must map the output nodes according to the value of the parity gate being simulated.

▶ **Lemma 7.2** ([14]). *For any $a, b \in \{0,1\}$, there is a unique automorphism φ of X_2 that maps x_i to $x_{a\oplus i}$ and y_i to $y_{b\oplus i}$ for $i \in \{0,1\}$. Moreover, this automorphism maps z_i to $z_{a\oplus b\oplus i}$.*

For the simulation of a circuit with fan-in 2 parity gates, one has to construct a parity gadget for each gate, and connect by an edge the output nodes of the gadgets (z nodes) with the input nodes (x and y nodes) of the corresponding gadget as indicated in the circuit description. Any automorphism of the constructed graph mapping the input nodes as the input values of the circuit (x_0 to x_a if the input value of the x gate is $a \in \{0,1\}$) must map node z_0 from the output gate to z_b, where b is the output value of the circuit.

Now for the reduction from ExactEven: given a system of equations S, we construct a red/blue graph $X = (\mathcal{R}, \mathcal{B}, E)$ in the following way: For every variable e_i in the system we define two blue vertices e_i^0 and e_i^1 in $\mathcal{B}$. These are the only blue vertices in the construction. For every equation $e_{i,1} \oplus \cdots \oplus e_{i,k_i} = 0$ in S we want to translate the property that the number of variables being set to 1 in this equation has to be even. For this, we can consider the circuit (formula) of fan-in 2 parity gates that computes the addition modulo 2 of the variables in the equation, and transform this circuit into a graph as described above. To ensure that no automorphism permutes the vertices z_0 and z_1 of the output gate, we add an additional neighbor to z_0. To ensure that any automorphism stabilizes the set $\{e_i^0, e_i^1\}$ for each i, we add a vertex e_i adjacent to both e_i^0 and e_i^1, and also add i additional neighbors to e_i. Now there is an assignment with exactly k ones satisfying all the equations in the system if and only if there is an automorphism in X moving exactly $2k$ blue vertices. ◀

Acknowledgments. We thank the anonymous referees for their valuable comments.

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42:844–856, 1995. `doi:10.1145/210332.210337`.

2 V. Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Solving linear equations parameterized by Hamming weight. *Algorithmica*, 75(2):322–338, 2016. `doi:10.1007/s00453-015-0098-3`.

3 Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. `doi:10.1007/BF01305232`.

4 Peter J. Cameron and Taoyang Wu. The complexity of the weight problem for permutation and matrix groups. *Discrete Mathematics*, 310(10):408–416, 2010. `doi:10.1016/j.disc.2009.03.005`.

5 Rod G. Downey and Michael R. Fellows. *Parameterized complexity.* Springer, New York, 1999.

6 Rod G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity.* Springer, London, 2013.

7 Rod G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999. `doi:10.1137/S0097539797323571`.

8 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, 2006.

9 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $\mathcal{O}(1)$ worst case access time. *J. ACM*, 31(3):538–544, 1984. `doi:10.1145/828.1884`.

10 Daniele Micciancio Ilya Dumer and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003. `doi:10.1109/TIT.2002.806118`.

11 Anna Lubiw. Some NP-complete problems similar to Graph Isomorphism. *SIAM Journal on Computing*, 10(1):11–21, 1981. `doi:10.1137/0210002`.

12 Pascal Schweitzer. Isomorphism of (mis)labeled graphs. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Proc. 19th ESA*, pages 370–381, Berlin, 2011. Springer. `doi:10.1007/978-3-642-23719-5_32`.

13 Charles C. Sims. Computational methods in the study of permutation groups. In John Leech, editor, *Computational problems in abstract algebra*, pages 169–183. Pergamon Press, 1970.

14 Jacobo Torán. On the hardness of graph isomorphism. *SIAM Journal on Computing*, 33(5):1093–1108, 2004. `doi:10.1137/S009753970241096X`.

15 Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997. `doi:10.1109/18.641542`.