Electronic Colloquium on Computational Complexity, Report No. 157 (2016)

Parameterized Complexity of Small Weight Automorphisms

V. Arvind Johannes Köbler Sebastian Kuhnert
Jacobo Torán

October 13, 2016

We consider the PermCode problem to decide, given a representation of a permutation group G and a parameter k, whether there is a non-trivial element of G with support at most k. This problem generalizes several problems in the literature. We introduce a new method that allows to reduce the maximal orbit size of the group being considered while maintaining elements with small support in the group. Using this technique we provide upper and lower bounds for several variants of the parameterized Hypergraph Isomorphism Problem that extend previous results on parameterized Graph Isomorphism.

1 Introduction

The problem to determine for a given set of linear equations $Ax = 0$ over $\mathbb{F}_2$ the minimum weight of a non-zero solution for the system is known as the minimum weight codeword problem and it is a fundamental and well studied algorithmic problem. It is known to be NP-hard even to approximate to a constant factor [15, 9]. The parameterized complexity of the decision version of this problem (called Even) is also well studied [7, 2].

Even: Given $Ax = 0$ defining a linear code over $\mathbb{F}_2$ where A is a matrix, and a parameter k, the problem is to determine if there is a non-zero codeword of weight at most k.

Whether Even is in FPT or not remains open. In contrast, the exact weight-k codeword problem (determining if there is a codeword with weight exactly k) is known to be W[1]-hard [7, 2].

A natural generalization of these problems is to consider a permutation group $G \leq S_n$, where G is given by a generating set S and ask for a permutation in G with minimum support set. The problem of interest is:

PermCode: Given $G = \langle S \rangle \leq S_n$ and parameter k, is there a permutation $\pi \in G \setminus \{\text{id}\}$ such that $|\text{supp}(g)| \leq k$, where $\text{supp}(\pi) = \{i \in [n] \mid \pi(i) \neq i\}$.

It is readily seen that Even is a special case of PermCode. To wit, we can encode $\mathbb{F}_2$ by a transposition, and a vector $v = (v_1, \dots, v_n) \in \mathbb{F}_2^n$ by a product of disjoint transpositions $(a_{i_1} b_{i_1})(a_{i_2} b_{i_2}) \cdots (a_{i_r} b_{i_r})$, where $v_{i_j} = 1$ for $j = 1, \dots, r$ and v has zeros in all other locations.

Another interesting connection of the minimum weight codeword problem, which is the main topic of this paper, is to the Graph Isomorphism and Graph Automorphism problems. Suppose the underlying permutation group $G \leq S_n$ in which we are to search for a nontrivial weight k

ISSN 1433-8092

element is the automorphism group $\mathrm{Aut}(X)$, where X is an n-vertex graph or hypergraph. What is then the complexity of this problem? Is it in FPT or W[1]-hard? In this setting, the precise analogues of EVEN are:

$\mathrm{GA}_{\leq k}$: Given an undirected simple graph $X = (V, E)$ and parameter k, does $\mathrm{Aut}(X)$ contain a nontrivial automorphism moving at most k vertices?

Likewise, we can define $\mathrm{GA}_{=k}$ as the exact weight-k problem. We denote by $\mathrm{HGA}_{\leq k}$ and $\mathrm{HGA}_{=k}$ the same problems for hypergraphs $X = (V, E)$.

In [12], Schweitzer showed that $\mathrm{GI}_{\leq k}$ is in FPT by giving a $k^{O(k)}\,\mathrm{poly}(n)$ time algorithm for it. The problem $\mathrm{GI}_{\leq k}$ asks for two given graphs X and Y having the same vertex set $V = [n]$ and for a parameter k whether there is an isomorphism between X and Y moving at most k vertices. Schweitzer's algorithm can easily be adapted to also solve the problem $\mathrm{GA}_{\leq k}$.

In this paper, we extend Schweitzer's result in several directions. First we generalize it to hypergraphs, using the maximum hyperedge size as a second parameter. It is well known that Hypergraph Isomorphism is reducible to Graph Isomorphism, however we do not know of any FPT reduction achieving this when the support size of the isomorphism is treated as parameter. We also combine this result with the color coding method from [4] in order to provide an *exact* version of the result. With this we can prove that the question of whether a given graph has an automorphism of support size exactly k is also in FPT. The difference between *at most* k and *exactly* k plays an important role in some problems in the area, for which the exact version seems to be computationally harder than the at most version.

Our mains results are the following (throughout the paper we use N to denote the size $|V| \cdot |E|$ of the input hypergraphs).

1. We show that PERMCODE for an arbitrary permutation group G can be reduced in polynomial time to PERMCODE for a subgroup G' of G, where the orbits of G' are of size bounded by the parameter k.

2. For $\mathrm{HGA}_{=k}$ and $\mathrm{HGI}_{=k}$ restricted to hypergraphs of edge size at most d we give $(kd)^{O(k^2)}\,\mathrm{poly}(N)$ time algorithms. This implies that for hypergraphs with hyperedges of $\mathrm{poly}(\log N)$ size the problems are still in FPT. As a consequence, $\mathrm{GA}_{=k}$ is also in FPT (in contrast to EVEN, where the exact version is W[1]-hard).

3. For $\mathrm{HGA}_{=k}$ and $\mathrm{HGI}_{=k}$ restricted to vertex colored hypergraphs having color classes of size at most b we give algorithms running in time $(bk)^{O(k^2)}\,\mathrm{poly}(N)$.

4. For general hypergraphs of unbounded edge size we show the problems $\mathrm{HGA}_{=k}$ and $\mathrm{HGI}_{=k}$ are in $\mathsf{FPT}^{\mathrm{GI}}$.

5. In constrast 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 such that $|\mathrm{supp}(\pi) \cap \mathit{blue}| \leq k$ then the problem is W[1]-hard. This corrects a question from [5, 6].

The key bridge to proving the main results is transforming the given instance hypergraph $X = (V, E)$ into one which has bounded color classes and which still has weight-k automorphisms. Working with bounded color classes enables us to search for exact weight-k solutions which are elusive in a direct application of Schweitzer's technique [12]. We will elaborate this further in Section 3.

2 Preliminaries

We consider only permutation groups and denote them by upper case Roman letters and elements of the groups by lower case Greek letters. 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 write $H \leq G$ when H is a subgroup of G. We apply permutations from left to right so that $(\varphi\pi)(v) = \pi(\varphi(v))$. For a subset $U \subseteq V$, $\pi(U) = \{\pi(u) \mid u \in U\}$. 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 $\{\pi(v) \mid \pi \in G\}$ 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.

Our algorithms rely on different measures on simple permutations. The *support* of $\pi \in \mathrm{Sym}(V)$ is $\mathrm{supp}(\pi) = \{u \in V \mid \pi(u) \neq u\}$. 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 $\pi \in \mathrm{Sym}(V)$.

1. A permutation $\sigma \in G\pi \setminus \{\mathrm{id}\}$ has *minimal support* in $G\pi$ if there is no $\sigma' \in G\pi \setminus \{\mathrm{id}\}$ with $\mathrm{supp}(\sigma') \subsetneq \mathrm{supp}(\sigma)$.
2. A permutation $\sigma \in G\pi \setminus \{\mathrm{id}\}$ has *minimal complexity* in $G\pi$ if there are no $\sigma_1 \in G \setminus \{\mathrm{id}\}$ and $\sigma_2 \in G\pi$ with $\sigma = \sigma_1\sigma_2$ and $\mathrm{compl}(\sigma) = \mathrm{compl}(\sigma_1) + \mathrm{compl}(\sigma_2)$.

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.

Recursively decomposing permutations that do not have minimal complexity yields the following lemma.

Lemma 2.2. *Let $G\pi$ be a coset of a permutation group G and let $\sigma \in G\pi$. Then for some $\ell \geq 1$ there are minimal-complexity permutations $\sigma_i \in G$ (for each $i \in \{1, \ldots, \ell - 1\}$) and a minimal-complexity permutation $\sigma_\ell \in G\pi$ 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.*

3 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 whose orbit sizes are not larger than the parameter k.

Lemma 3.1. *There is a polynomial time algorithm that given an instance (A, k) of* PermCode, *with $G = \langle A\rangle \leq S_n$ and $k \in \mathbb{N}$, computes an instance (B, k) with $\langle B\rangle = G' \leq G$ and the property that every orbit of G' has size bounded by k and $(A, k) \in$* PermCode *if and only if $(B, k) \in$* PermCode. *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 $G_u = \{\varphi \in G \mid \varphi(u) = u\}$.*

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 $\sigma(v) = u$. 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.

Algorithm 1: PermCode reduction

```
Input: A group $G = \langle A \rangle \leq S_n$ and a parameter $k$
Output: 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 $B$
```

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

4 Bounded hyperedge size

In this section we consider the problems $\mathrm{HGA}_{\leq k}$ and $\mathrm{HGI}_{\leq k}$ for hypergraphs with hyperedges of bounded size and give an FPT algorithm for them. More precisely, let $X = (V, E)$ and $Y = (V, E')$ be hypergraphs such that for each $e \in E$ we have $|e| \leq d$. Then we show that a nontrivial element $\pi \in \mathrm{Iso}(X, Y)$ of weight at most k, if it exists, can be found in $(dk)^{O(k)}\,\mathrm{poly}(N)$ time. Notice that even for $d = (\log N)^{O(1)}$ this running time is bounded by $k^{O(k)}\,\mathrm{poly}(N)$.

This generalizes Schweitzer's result [12] shown for usual graphs, to hypergraphs of bounded hyperedge size. In order to find a graph isomorphism π of 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 thus need a generalization of vertex covers.

Definition 4.1. Let X be a hypergraph over 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 4.2. Let X and Y be two hypergraphs over a vertex set V. $X \bigtriangleup Y$ is the hypergraph with edge set $\mathrm{E}(X) \bigtriangleup \mathrm{E}(Y)$ and having as vertex set the union of all edges in $\mathrm{E}(X) \bigtriangleup \mathrm{E}(Y)$.

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

Lemma 4.3. *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 $|e \cap C| \le q$ it holds that $e \cap \operatorname{supp}(\pi) \subseteq C$.*

Proof. Let $e = \{u_1, \dots, u_t\} \in \mathrm{E}(X) \triangle \mathrm{E}(Y)$ be a hyperedge such that $u_1, \dots, u_s \in C$ and $u_{s+1}, \dots, u_t \notin C$ for some $s \le q$. To obtain a contradiction, suppose that $e \cap \operatorname{supp}(\pi) \subseteq C$, i.e., $\pi(u_i) = u_i$ for $i \in \{s+1, \dots, t\}$. W.l.o.g. let $e \in \mathrm{E}(X) \setminus \mathrm{E}(Y)$. Let ℓ be the length of the orbit of e under π, i.e. the smallest 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 $1 \le i < \ell$. This yields a contradiction to $\pi^{\ell-1}(e) \notin \mathrm{E}(X)$, 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 $j \in \{1, \dots, s\}$ 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 4.4. *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 $C \cup \operatorname{supp}(\pi)$ is a $(q+1)$-strong vertex cover of $X \triangle Y$.*

Proof. Applying Lemma 4.3 it follows that for all hyperedges e of $X \triangle Y$ with $|e \cap C| \le q$ we have $e \cap \operatorname{supp}(\pi) \nsubseteq C$ meaning that $\operatorname{supp}(\pi)$ covers at least one additional vertex in each such hyperedge. □

Lemma 4.5. *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 4.3 implies that C does not even contain $e \cap \operatorname{supp}(\pi)$, a contradiction. □

Lemmas 4.4 and 4.5 suggest the following algorithm to compute a candidate set; it can be plugged into Schweitzer's isomorphism test [12, Algorithm 1], cf. Algorithm 3 below.

Algorithm 2: $\texttt{CandidateSet}_{k,d}(X \triangle Y)$

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

Observe that, by construction, Algorithm 2 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 4.6. *If $X \neq Y$ are hypergraphs on vertex set V with hyperedge size bounded by 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 4.4 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 4.5 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., [10, Theorem 1.14]). □

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

Algorithm 3: $\texttt{ISO}_{k,d}(X, Y, c)$

```
Input: Two hypergraphs $X$ and $Y$ on vertex set $V = [n]$ with hyperedge size bounded
         by $d$ and a natural number $c \leq k$ 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 $v_1, v_2 \in C$ do
        $P' \leftarrow \texttt{ISO}_{k,d}(X, Y^{(v_1 v_2)}, c-1)$
        $P \leftarrow P \cup \{\varphi' \cdot (v_1 v_2) \mid \varphi' \in P'\}$ // compose with the isomorphism $(v_1 v_2)$ from
                                                   $Y^{(v_1 v_2)}$ to $Y$
return $\{\varphi \in P \mid |\mathrm{supp}(\varphi)| \leq k\}$
```

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 3 finds a meaningful subset of them.

Lemma 4.7. *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.*

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 $|\mathrm{supp}(\varphi)| \leq k$ and $\mathrm{compl}(\varphi) \leq c$ is in this set. By Lemma 4.6, the candidate set C contains two vertices v_1 and v_2 that belong to the same orbit of φ. Thus we get $\varphi = \varphi' \cdot (v_1 v_2)$

for some φ' with $\operatorname{compl}(\varphi') = \operatorname{compl}(\varphi) - 1$. Note that if $\varphi' \neq \mathrm{id}$ then φ' has minimal complexity in $\operatorname{Iso}(X, Y^{(v_1v_2)})$. Indeed, $\varphi' = \varphi_1\varphi_2'$ with $\varphi_1 \in \operatorname{Aut}(X) \setminus \{\mathrm{id}\}$, $\varphi_2' \in \operatorname{Iso}(X, Y^{(v_1v_2)})$ and $\operatorname{compl}(\varphi') = \operatorname{compl}(\varphi_1) + \operatorname{compl}(\varphi_2')$ would imply $\varphi = \varphi_1\varphi_2$ with $\varphi_1 \in \operatorname{Aut}(X) \setminus \{\mathrm{id}\}$, $\varphi_2 = \varphi_2' \cdot (v_1v_2) \in \operatorname{Iso}(X, Y)$ and $\operatorname{compl}(\varphi) = \operatorname{compl}(\varphi_1) + \operatorname{compl}(\varphi_2)$, contradicting that φ has minimal complexity in $\operatorname{Iso}(X, Y)$.

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 $|C| \leq dk$, there are $\mathcal{O}\big((dk)^{2c}\big)$ recursive calls in total. In each call, it takes $\mathcal{O}(d^{k+1} \operatorname{poly}(N))$ time to compute C (Lemma 4.6), 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 $\operatorname{compl}(\sigma) < \operatorname{compl}(\varphi)$ and $\operatorname{compl}(\varphi) = \operatorname{compl}(\varphi\sigma^{-1}) + \operatorname{compl}(\sigma)$; note that this property can be checked in polynomial time.

Theorem 4.8. *Given two hypergraphs $X \neq Y$ with hyperedge size bounded by d, it can be decided in time $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)} \operatorname{poly}(N)\big)$ 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 $|\operatorname{supp}(\varphi)| \leq k$ trivially satisfies $\operatorname{compl}(\varphi) \leq k$ and can be decomposed by Lemma 2.2, obtaining a minimal-complexity isomorphism φ' from X to Y with $\operatorname{supp}(\varphi') \subseteq \operatorname{supp}(\varphi)$. By Lemma 4.7, 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 $\mathrm{HGA}_{\leq k}$ for hypergraphs with hyperedge size bounded by d in time $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)} \operatorname{poly}(N)\big)$.

Algorithm 4: $\texttt{AUT}_{k,d}(X)$

```
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
    $P' \leftarrow \texttt{ISO}_{k,d}(X, X^{(v_1v_2)}, k-1)$
    $P \leftarrow P \cup \{\varphi' \cdot (v_1v_2) \mid \varphi' \in P'\}$ // compose with the isomorphism $(v_1v_2)$
                                          from $X^{(v_1v_2)}$ to $X$
return $\{\varphi \in P \mid |\operatorname{supp}(\varphi)| \leq k\}$
```

Theorem 4.9. *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 $\mathcal{O}\big((dk)^{\mathcal{O}(k^2)} \operatorname{poly}(N)\big)$ time.*

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 $|\operatorname{supp}(\varphi)| \leq k$ is found follows from Lemma 4.7 using the same decomposition argument as in the proof of the latter. Also the time bound follows immediately from Lemma 4.7. □

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 $\sigma \in \mathrm{Aut}(X)$ with support size at most k can be written as a product of minimal-complexity automorphisms of X with support size at most k, so $\sigma \in \langle S\rangle$, where S is the set returned by $\mathtt{AUT}_{k,d}(X)$.

5 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 $\mathrm{HGI}_{\leq k}$ for b-bounded hypergraphs. More precisely, given hypergraphs $X = (V, E)$ and $Y = (V, E')$ and a partition $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of V into (pairwise disjoint) color classes C_i with $|C_i| \leq b$, our algorithm will compute in time $\mathcal{O}\big((kb!)^{O(k^2)}\,\mathrm{poly}(N)\big)$ 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

$$\mathcal{C}[\pi] = \{C_i \in \mathcal{C} \mid \exists v \in C_i : \pi(v) \neq v\}$$

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 $\mathcal{C}[\pi] = \{C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}\}$, $\ell \leq k/2$. In order to search for the color classes in $\mathcal{C}[\pi]$ we will apply the color-coding method of Alon-Yuster-Zwick [4]. Consider the FKS 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:

$$V_j = \bigcup \mathcal{B}_j, E_j = \{e \cap V_j \mid e \in E\} \text{ and } E'_j = \{e \cap V_j \mid e \in E'\}.$$

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 $\mathcal{B}_j$ (namely C_{i_j}). We say that a color class $C_i \in \mathcal{B}_j$ *moves* 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 $(e \cap V_j)^\pi \neq e \cap V_j$. We denote the set of all color classes C_i that move e by $\mathrm{Move}(e)$. The next claim shows that the size of $\mathrm{Move}(e)$ is bounded by $\sum_{j=1}^{\ell} \log|E'_j| \leq \ell \log|E|$.

Claim. *Each hyperedge $e \in E$ is moved by at most $\log|E'_j|$ many color classes $C_i \in \mathcal{B}_j$.*

Proof of the claim. Suppose that e is moved by $t > \log|E_j|$ many color classes $C_{j_1}, \ldots, C_{j_t} \in \mathcal{B}_j$. Let $\pi_{j_r} \in \mathrm{Iso}(X_j, Y_j)$, 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 $\mathrm{Iso}(X_j, Y_j)$ and all the images $\big(\prod_{r\in T} \pi_{j_r}\big)(e \cap V_j)$, 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. $\square$

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

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

An isomorphism $\pi \neq \mathrm{id}$ 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 $\mathrm{Iso}(X, Y)$.

We notice that all isomorphisms having minimal support are also color-class-minimal. Another immediate consequence of Definition 5.1 is the following lemma for decomposing automorphisms of hypergraphs.

Lemma 5.2. *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 $1 \leq i \leq \ell$, are pairwise disjoint and form a partition of the support color class set $\mathcal{C}[\pi]$.*

The following algorithm 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 $\pi \in \mathrm{Sym}(V)$ $(k_1, \dots, k_\ell)$-*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 $|\mathrm{supp}(\pi) \cap C_{i_j}| = k_j$ for $j = 1, \dots, \ell$.

Algorithm 5: $\texttt{ColoredIso}_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \pi)$

```
Input: Hypergraphs $X$ and $Y$ on vertex set $V = C_1 \cup \cdots \cup C_m$ with color classes
        $C_i$ of size $|C_i| \leq b$ 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 $\pi(e) \notin \mathrm{E}(Y)$
    foreach bag $\mathcal{B}_j$ with $\mathrm{supp}(\pi) \cap \bigcup \mathcal{B}_j = \varnothing$ do
        foreach color class $C \in \mathrm{Move}(e) \cap \mathcal{B}_j$ do
            foreach $\sigma \in \mathrm{Sym}(V)$ with $\mathrm{supp}(\sigma) \subseteq C$ and $|\mathrm{supp}(\sigma)| = k_j$ do
                $P \leftarrow P \cup \texttt{ColoredIso}_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \pi\sigma)$
    return $P$
```

The following lemma shows that Algorithm 5 allows to find a meaningful set of isomorphisms.

Lemma 5.3. *Let $X \neq Y$ be two b-bounded hypergraphs. Then the set returned by the algorithm $\texttt{ColoredIso}_{k_1,\dots,k_\ell,b,\mathcal{B}_1,\dots,\mathcal{B}_\ell}(X, Y, \mathrm{id})$ 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 5 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 $\mathrm{supp}(\varphi)$. We show by induction on the number of color classes in $\mathcal{C}[\varphi] \setminus \mathcal{C}'$ touched by $\mathrm{supp}(\varphi)$ but not by $\mathrm{supp}(\pi)$ 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 $\pi \notin \mathrm{Iso}(X, Y)$. Since φ is $(k_1, \dots, k_\ell)$-good for $\mathcal{B}_1, \dots, \mathcal{B}_\ell$, for any hyperedge $e \in \mathrm{E}(X)$ with $\pi(e) \notin \mathrm{E}(Y)$, there is a bag $\mathcal{B}_j$ with $\mathrm{supp}(\pi) \cap \bigcup \mathcal{B}_j = \varnothing$ and a color class $C \in \mathrm{Move}(e) \cap \mathcal{B}_j$ 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 $|\mathrm{Move}(e)| \cdot |\mathrm{Sym}(C)| \leq k \log|E| \cdot b!$ 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 $X \neq Y$ 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 $k = k_1 + \cdots + k_\ell$. This adds only an extra $k^{O(k)}$ factor and yields the algorithm $\texttt{ColoredIso}_{k,b}(X, Y)$ 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 $\texttt{ColoredAut}_{k,b}(X)$ for computing all color-class-minimal automorphisms of X with support size exactly k.

Theorem 5.4. *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 $\mathcal{O}\big((kb!)^{O(k^2)} \operatorname{poly}(N)\big)$ 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 5.5. *There is an algorithm for $\mathrm{HGI}_{\leq k}$ that decides for two given b-bounded hypergraphs X and Y in time $\mathcal{O}\big((kb!)^{O(k^2)} \operatorname{poly}(N)\big)$ if there is an isomorphism from X to Y of weight at most k and computes such an isomorphism if it exists.*

6 Exact support size

In this section we show that the problem $\mathrm{HGA}_{=k}$ 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 $\mathrm{HGA}_{=k}$ problem for such graphs is FPT reducible to the $\mathrm{HGA}_{=k}$ 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 4 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 4.9.

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 $\texttt{AUT}_{k,d}(X)$. 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 4.9, 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 3.1 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. $\square$

Theorem 6.3. *There is an algorithm for $\mathrm{HGA}_{=k}$ for b-bounded hypergraphs $X = (V, E)$ that decides in time $(kb!)^{O(k^2)}\,\mathrm{poly}(N)$ 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 5.4 to enumerate the set A of all color-class-minimal automorphisms of X of weight at most k in time $(kb!)^{O(k^2)}\,\mathrm{poly}(N)$. By Lemma 5.2, we know that for any exact weight k color-preserving automorphism π of X there is an $\ell \le k$ such that π is expressible as $\pi = \pi_1\pi_2 \dots \pi_\ell$, where each π_s is a color-class-minimal automorphism of X of weight at most k and $\mathcal{C}[\pi_s]$, for $1 \le s \le \ell$, forms a partition of $\mathcal{C}[\pi]$. I.e. each candidate π_s is in the enumerated set A. Let $\mathcal{C}[\pi] = \{C_{j_1}, C_{j_2}, \dots, C_{j_r}\}$, where $r \le k$ (because in all only k vertices are moved by π).

We will again use color coding [4] to search for the $\pi_i, 1 \le i \le \ell$. 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 $r \le k$, as above, is the number of color classes in $\mathcal{C}[\pi]$. For each $i \in [r]$ define the bags $\mathcal{B}_i = \{C_j \mid h(j) = i\}$. By the property of the FKS family, there is some h such that for $i = 1, \dots, r$, each bag $\mathcal{B}_i$ contains exactly one color class from $\mathcal{C}[\pi]$ (namely C_{j_i}). Notice the following simple claim.

Claim. *The total number of partitions of the set $\{j_1, j_2, \dots, j_r\}$ into ℓ sets is bounded by $2^{r\ell}$ and we can cycle through all such partitions in time $2^{O(r\ell)}$.*

One of the $2^{r\ell}$ many partitions, say $I_1 \sqcup I_2 \sqcup \dots \sqcup I_\ell$ of $\{j_1, j_2, \dots, j_r\}$ will be the partition such that $\mathcal{C}[\pi_s] = \{C_j \mid j \in I_s\}, 1 \le s \le \ell$. Assume we are considering this partition $I_1 \sqcup I_2 \sqcup \dots \sqcup I_\ell$. Now, let A denote the set of all color-class-minimal automorphisms of X that can be enumerated applying the algorithm in Theorem 5.4.

In $|A|$ time we partition A into subsets A_s, with $1 \le s \le \ell$, defined by

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

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

Clearly, by Theorem 5.4, if there exists a color-preserving exact weight k automorphism of X, then for some $h \in \mathcal{H}$ and some partition the search will succeed.

This concludes the proof. □

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* $d^{O(k)}2^{O(k^2)}\operatorname{poly}(N)$ *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 $(dk)^{O(k)}\operatorname{poly}(|V|, |E|)$ 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 6.5. For the general case of hypergraphs of unbounded edge size it is easy to see that we have an $\mathsf{FPT}^{\mathrm{GI}}$ algorithm for the problem $\mathrm{HGA}_{=k}$ 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 $\mathrm{Aut}(X)$ and then using Lemma 3.1 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, defined in the book [5], asks to decide, given a red/blue graph $X = (\mathcal{R}, \mathcal{B}, E)$ and the parameter $k \in \mathbb{N}$, whether X has a color-preserving automorphism whose support contains exactly k of the "blue" vertices in $\mathcal{B}$.

Exercise 9.0.2 in [5] (also Exercise 20.3.2 in [6]) asks to prove that this problem is W[1]-hard. The authors give as a hint a reduction from Weighted Antimonotone 2-CNF-SAT transforming a formula into a graph, and it should be proven that the formula has a weight k satisfying assignment if and only if the graph has an automorphism that moves exactly $2k$ blue vertices. Unfortunately this statement is not correct. It can be seen that the constructed graph is rigid and therefore it does not have any non-trivial automorphisms.

We present here a correct proof of this result by giving an FPT reduction from the ExactEvenSet problem, shown to be W[1]-hard in [7], to ColoredGraphAutomorphism.

ExactEvenSet: Given a system S of m linear equations in $\mathbb{F}_2$ of the form $e_{i,1} \oplus \cdots \oplus e_{i,k_i} = 0$, over a set of n variables and a positive integer k as parameter, decide if there is an assignment of the variables of weight k satisfying all the equations in the system.

Theorem 7.1. ExactEvenSet *is FPT reducible to* ColoredGraphAutomorphism*.*

Proof. The base of the reduction is a gadget given in [14] for simulating a circuit with parity gates as an instance of the Graph Isomorphism problem. Let $\oplus$ denote the addition in $\mathbb{F}_2$. We define the undirected graph $X^2 = (V, E)$, given by the set of 10 nodes $V = \{x_a, y_a, z_a \mid a \in \{0,1\} \cup \{u_{a,b} \mid a, b \in \{0,1\}\}\}$ and edges

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

This graph gadget simulates a parity fan-in 2 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 ExactEvenSet, given a system of equations S, we construct a red/blue graph $X = (\mathcal{R}, \mathcal{B}, \mathcal{E})$ 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 the system 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 computing the addition modulo 2 of the variables in the equation, transforming this circuit into a graph as described above. We call this subgraph a pyramid for the equation. All the vertices in this pyramid graph are red, except those corresponding to the e variables (inputs), at the bottom level, which are blue. This is done for all the equations separately. The vertices e_i^0 and e_i^1 corresponding to a variable e_i are connected to the pyramids of all the equations in which e_i appears. There is an assignment of the e variables with exactly k ones satisfying all the equations in the system if and only if there is an automorphism mapping exactly k of the e_i^0 vertices to the corresponding e_i^1 vertices (and vice versa) and fixing the output vertex of every pyramid (this assures that the equations are satisfied). In order to force this last property, we connect each one of the output vertices z_j^0 and z_j^1 of a pyramid to a different rigid subgraph. In order to assure that for each i, an automorphism can map e_i^0 only to itself or to e_i^1, we connect these two vertices by an edge to a new vertex e_i and connect e to a rigid gadget (different for each i). The vertices e_i as well as the vertices in the gadgets are red. 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. □

References

[1] V. Arvind, Bireswar Das, Johannes Köbler and Seinosuke Toda, Colored Hypergraph Isomorphism is Fixed Parameter Tractable. *FSTTCS 2010,* 327–337, 2010.

[2] V. Arvind, Johannes Köbler, Sebastian Kuhnert and Jacobo Torán, Solving Linear Equations Parameterized by Hamming Weight. *Algorithmica,* 75(2): 322-.338, 2016.

[3] V. Arvind and Johannes Köbler, The Parallel Complexity of Graph Canonization Under Abelian Group Action. *Algorithmica,* 67(2): 247–276, 2013.

[4] Noga Alon, Raphael Yuster and Uri Zwick, Color-coding. *J. ACM* 42: 844–856, 1995.

[5] Rod G. Downey and Michael R. Fellows, *Parameterized Complexity.* Springer, 1999.

[6] Rod G. Downey and Michael R. Fellows, *Fundamentals of Parameterized Complexity.* Springer, 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.

[8] John D. Dixon and Brian Mortimer, *Permutation groups.* Springer, 1996.

[9] Ilya Dumer, Daniele Micciancio and Madhu Sudan, Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1), 22–37, 2003.

[10] Jörg Flum and Martin Grohe, *Parameterized complexity theory.* Springer, 2006.

[11] M. Furst, J. E. Hopcroft, and E. M. Luks, Polynomial-time algorithms for permutation groups. Technical report, Cornell University, 10 1980.

[12] Pascal Schweitzer, Isomorphism of (mis)labeled graphs. *ESA 2011*, 370–381, 2011.

[13] Charles C. Sims, Computational methods in the study of permutation groups. Computational Problems in Abstract Algebra, 169–183, Pergamon, Oxford, 1970.

[14] Jacobo Torán, On the hardness of Graph Isomorphism. *SIAM Journal on Computing*, 33(5): 1093–1108, 2004.

[15] Alexander Vardy, The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory* 43, 1757–1766, 1997.