Finding Small Weight Isomorphisms with Additional Constraints is Fixed-Parameter Tractable[*][†]

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, Germany**
koebler@informatik.hu-berlin.de
3 **Institut für Informatik, Humboldt-Universität zu Berlin, Germany**
kuhnert@informatik.hu-berlin.de
4 **Institut für Theoretische Informatik, Universität Ulm, Germany**
toran@uni-ulm.de

Abstract

Lubiw showed that several variants of Graph Isomorphism are NP-complete, where the solutions are required to satisfy certain additional constraints [12]. One of these, called Isomorphism With Restrictions, is to decide for two given graphs $X_1 = (V, E_1)$ and $X_2 = (V, E_2)$ and a subset $R \subseteq V \times V$ of forbidden pairs whether there is an isomorphism π from X_1 to X_2 such that $i^\pi \neq j$ for all $(i, j) \in R$. We prove that this problem and several of its generalizations are in fact in FPT:

- The problem of deciding whether there is an isomorphism between two graphs that moves k vertices and satisfies Lubiw-style constraints is in FPT, with k and $|R|$ as parameters. The problem remains in FPT even if a CNF of such constraints is allowed. As a consequence of the main result it follows that the problem to decide whether there is an isomorphism that moves exactly k vertices is in FPT. This solves a question left open in [1].
- When the number of moved vertices is unrestricted, finding isomorphisms that satisfy a CNF of Lubiw-style constraints is in $\mathsf{FPT}^{\mathrm{GI}}$.
- Checking if there is an isomorphism between two graphs that has complexity t is also in FPT with t as parameter, where the complexity of a permutation π is the Cayley measure defined as the minimum number t such that π can be expressed as a product of t transpositions.
- We consider a more general problem in which the vertex set of a graph X is partitioned into Red and Blue, and we are interested in an automorphism that stabilizes Red and Blue and moves exactly k vertices in Blue, where k is the parameter. This problem was introduced in [5], and in [1] we showed that it is W[1]-hard even with color classes of size 4 inside Red. Now, for color classes of size at most 3 inside Red, we show the problem is in FPT.

In the non-parameterized setting, all these problems are NP-complete. Also, they all generalize in several ways the problem to decide whether there is an isomorphism between two graphs that moves at most k vertices, shown to be in FPT by Schweitzer [13].

[*] Some proofs are omitted from this extended abstract; see https://arxiv.org/abs/1709.10063 for the full version.
[†] This work was supported by the Alexander von Humboldt Foundation in its research group linkage program.
[‡] The author is supported by DFG grant KO 1053/7-2.
[§] The author is supported by DFG grant KO 1053/7-2.
[¶] The author is supported by DFG grant TO 200/3-2.

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, mislabeled graphs

Digital Object Identifier 10.4230/LIPIcs.IPEC.2017.2

1 Introduction

The Graph Isomorphism problem (GI) consists in deciding whether two given input graphs are isomorphic, i.e., whether there is a bijection between the vertex sets of the two graphs that preserves the adjacency relation. It is an intensively researched algorithmic problem for over four decades, culminating in Babai's recent quasi-polynomial time algorithm [2].

There is also considerable work on the parameterized complexity of GI. For example, already in 1980 it was shown [7] that GI, parameterized by color class size, is fixed-parameter tractable (FPT). It is also known that GI, parameterized by the eigenvalue multiplicity of the input graph, is in FPT [3]. More recently, GI, parameterized by the treewidth of the input graph, is shown to be in FPT [11].

In a different line of research, Lubiw [12] has considered the complexity of GI with additional constraints on the isomorphism. Exploring the connections between GI and the NP-complete problems, Lubiw defined the following version of GI.

Isomorphism With Restrictions: Given two graphs $X_1 = (V_1, E_1)$ and $X_2 = (V_2, E_2)$ and a set of forbidden pairs $R \subseteq V_1 \times V_2$, decide whether there is an isomorphism π from X_1 to X_2 such that $i^\pi \neq j$ for all $(i, j) \in R$.

When $X_1 = X_2$, the problem is to check if there is an automorphism that satisfies these restrictions. Lubiw showed that the special case of testing for *fixed-point-free automorphisms* is NP-complete. Klavík et al. recently reexamined Isomorphism With Restrictions [10]. They show that it remains NP-complete when restricted to graph classes for which GI is as hard as for general graphs. Conversely, they show that it can be solved in polynomial time for several graph classes for which the isomorphism problem is known to be solvable in polynomial time by combinatorial algorithms, e.g. planar graphs and bounded treewidth graphs. However, they also show that the problem remains NP-complete for bounded color class graphs, where an efficient group theoretic isomorphism algorithm is known.

A different kind of constrained isomorphism problem was introduced by Schweitzer [13]. The weight (or support size) of a permutation $\pi \in \mathrm{Sym}(V)$ is $|\{i \in V \mid i^\pi \neq i\}|$. Schweitzer showed that the problem of testing if there is an isomorphism π of weight at most k between two n-vertex input graphs in the same vertex set can be solved in time $k^{\mathcal{O}(k)} \operatorname{poly}(n)$. Hence, the problem is in FPT with k as parameter. Schweitzer's algorithm exploits interesting properties of the structure of an isomorphism π. Based on Lubiw's reductions [12], it is not hard to see that the problem is NP-complete when k is not treated as parameter.

In this paper we consider the problem of finding isomorphisms with additional constraints in the parameterized setting. In our main result we formulate a *graph isomorphism/automorphism problem with additional constraints* that generalizes Lubiw's setting as follows. For a graph $X = (V, E)$, let $\pi \in \mathrm{Aut}(X)$ be an automorphism of X. We say that a permutation $\pi \in \mathrm{Sym}(V)$ *satisfies* a formula F over the variables in $\mathrm{Var}(V) = \{x_{u,v} \mid u, v \in V\}$ if F is satisfied by the assignment that has $x_{u,v} = 1$ if and only if $u^\pi = v$. For example, the conjunction $\bigwedge_{u \in V} \neg x_{uu}$ expresses the condition that π is fixed-point-free. We define:

Exact-CNF-GI: Given two graphs $X_1 = (V, E_1)$ and $X_2 = (V, E_2)$, a CNF formula F over $\mathrm{Var}(V)$, and $k \in \mathbb{N}$, decide whether there is an isomorphism from X_1 to X_2 that has weight exactly k and satisfies F. The parameter is $|F| + k$, where $|F|$ is the number of variables used in F.

In Section 4, we first give an FPT algorithm for Exact-CNF-GA, the automorphism version of this problem. The algorithm uses an orbit shrinking technique that allows us to transform the input graph into a graph with bounded color classes, preserving the existence of an exact weight k automorphism that satisfies the formula F. The bounded color class version is easy to solve using color coding; see Section 3 for details. Building on this, we show that Exact-CNF-GI is also in FPT. In particular, this allows us to efficiently find isomorphisms of weight exactly k, a problem left open in [1], and extends Schweitzer's result mentioned above to the *exact* case. In our earlier paper [1] we have shown that the problem of *exact weight k automorphism* is in FPT using a simpler orbit shrinking technique which does not work for exact weight k isomorphisms. In this paper, we use some extra group-theoretic machinery to obtain a more versatile orbit shrinking.

In Section 5, we consider the problem of computing graph isomorphisms of *complexity* exactly t: The complexity of a permutation $\pi \in \mathrm{Sym}(V)$ is the minimum number of transpositions whose product is π. Checking for automorphisms or isomorphisms of complexity exactly t is NP-complete in the non-parameterized setting. We show that the problem is in FPT with t as parameter. Again, the "at most t" version of this problem was already shown to be in FPT by Schweitzer [13] as part of his algorithmic strategy to solve the weight at most k problem. Our results in Sections 4 and 5 also hold for hypergraphs when the maximum hyperedge size is taken as additional parameter.

In Section 6, we examine a different restriction on the automorphisms being searched for. Consider graphs $X = (V, E)$ with vertex set partitioned into Red and Blue. The *Colored Graph Automorphism* problem (defined in [5]; we denote it Col-GA), is to check if X has an automorphism that respects the partition and moves exactly k Blue vertices. We showed in [1] that this problem is W[1]-hard. In our hardness proof the orbits of the vertices in the Red part of the graph have size at most 4, while the ones for the Blue vertices have size 2. We show here that this cannot be restricted any further: If the input graph has Red further partitioned into color classes of size at most 3 each, then the problem to test whether there is an automorphism moving exactly k Blue vertices can be solved in FPT (with parameter k). The Blue part of the graph remains unconstrained. Observe that Schweitzer's problem [13] coincides with the special case of this problem where there are no Red vertices. This implies that the non-parameterized version of Col-GA is NP-complete (even when X has only Blue vertices). Similarly, finding weight k automorphisms of a hypergraph reduces to Col-GA by taking the incidence graph, where the original vertices become Blue and the vertices for hyperedges are Red; note that this yields another special case, where both Red and Blue induce the empty graph, respectively.

2 Preliminaries

We use standard permutation group terminology, see e.g. [4]. Given a permutation $\sigma \in \mathrm{Sym}(V)$, its *support* is $\mathrm{supp}(\sigma) = \{u \in V \mid u^\sigma \neq u\}$ and its *(Hamming) weight* is $|\mathrm{supp}(\sigma)|$. The *complexity* of σ (sometimes called its *Cayley weight*) is the minimum number t such that σ can be written as the product of t transpositions.

Let $G \leq \mathrm{Sym}(V)$ and $\pi \in \mathrm{Sym}(V)$; this includes the case $\pi = \mathrm{id}$. A permutation $\sigma \in G\pi \setminus \{\mathrm{id}\}$ has *minimal complexity in $G\pi$* 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 $\tau_i \cdots \tau_{\mathrm{compl}(\sigma)} \notin G\pi$. The following lemma observes that every element of $G\pi$ can be decomposed into minimal-complexity factors.

▶ **Lemma 2.1** [1, 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 $\sigma_1, \dots, \sigma_{\ell-1} \in G$ 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 $i \in \{1, \dots, \ell\}$.*

An action of a permutation group $G \leq \mathrm{Sym}(V)$ on a set V' is a group homomorphism $h\colon G \to \mathrm{Sym}(V')$; we denote the image of G under h by $G(V')$. For $u \in V$, we denote its stabilizer by $G_u = \{\pi \in G \mid u^\pi = u\}$. For $U \subseteq V$, we denote its pointwise stabilizer by $G_{[U]} = \{\pi \in G \mid \forall u \in U : u^\pi = u\}$ and its setwise stabilizer by $G_{\{U\}} = \{\pi \in G \mid U^\pi = U\}$. For $S \subseteq \mathcal{P}(V)$, we let $G_S = \{\pi \in G \mid \forall U \in S : U^\pi = U\}$.

A *hypergraph* $X = (V, E)$ consists of a vertex set V and a hyperedge set $E \subseteq \mathcal{P}(V)$. Graphs are the special case where $|e| = 2$ for all $e \in E$. The *degree* of a vertex $v \in V$ is $|\{e \in E | v \in e\}|$. A *(vertex) coloring* of X is a partition of V into color classes $\mathcal{C} = (C_1, \dots, C_m)$. The color classes $\mathcal{C}$ are *b-bounded* if $|C_i| \leq b$ for all $i \in [m]$. An *isomorphism* between two hypergraphs $X = (V, E)$ and $X' = (V', E')$ (with color classes $\mathcal{C} = (C_1, \dots, C_m)$ and $\mathcal{C}' = (C'_1, \dots, C'_m)$) is a bijection $\pi\colon V \to V'$ such that $E' = \big\{\{\pi(v) | v \in e\} \big| e \in E\big\}$ (and $C'_i = \big\{\pi(v) \big| v \in C_i\big\}$). The isomorphisms from X to X' form a coset that we denote by $\mathrm{Iso}(X, X')$. The automorphisms of a hypergraph X are the isomorphisms from X to itself; they form a group which we denote by $\mathrm{Aut}(X)$.

3 Bounded color class size

To show that Exact-CNF-GA for hypergraphs with b-bounded color classes can be solved in FPT, we recall our algorithm for exact weight k automorphisms of bounded color class hypergraphs [1] and show how it can be adapted to the additional constraints given by the input formula.

▶ **Definition 3.1.** Let $X = (V, E)$ be a hypergraph with color class set $\mathcal{C} = \{C_1, \dots, C_m\}$.

(a) For a subset $\mathcal{C}' \subseteq \mathcal{C}$, we say that a color-preserving permutation $\pi \in \mathrm{Sym}(V)$ *$\mathcal{C}'$-satisfies* a CNF formula F over $\mathrm{Var}(V)$ if every clause of F contains a literal $x_{u,v}$ or $\neg x_{u,v}$ with $u \in \bigcup \mathcal{C}'$ that is satisfied by π.

(b) For a color-preserving permutation $\pi \in \mathrm{Sym}(V)$, let $\mathcal{C}[\pi] = \{C_i \in \mathcal{C} \mid \exists v \in C_i : v^\pi \neq v\}$ be the subset of color classes that intersect $\mathrm{supp}(\pi)$. For a subset $\mathcal{C}' \subseteq \mathcal{C}[\pi]$, we define the permutation $\pi_{\mathcal{C}'} \in \mathrm{Sym}(V)$ as

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

Note that $\pi_{\mathcal{C}[\pi]} = \pi$.

(c) A color-preserving automorphism $\sigma \neq \mathrm{id}$ of X is said to be *color-class-minimal*, if for every set $\mathcal{C}'$ with $\varnothing \subsetneq \mathcal{C}' \subsetneq \mathcal{C}[\sigma]$, the permutation $\sigma_{\mathcal{C}'}$ is not in $\mathrm{Aut}(X)$.

▶ **Lemma 3.2.** *Let $X = (V, E)$ be a hypergraph with color class set $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$. For $\varnothing \neq \mathcal{C}' \subseteq \mathcal{C}$ and a CNF formula F over $\mathrm{Var}(V)$, the following statements are equivalent:*

- *There is a nontrivial automorphism σ of X with $\mathcal{C}[\sigma] = \mathcal{C}'$ that satisfies F.*
- *$\mathcal{C}'$ can be partitioned into $\mathcal{C}_1, \dots, \mathcal{C}_\ell$ and F (seen as a set of clauses) can be partitioned into CNF formulas $F_0, \dots, F_\ell$ such that F_0 is $(\mathcal{C} \setminus \mathcal{C}')$-satisfied by id and for each $i \in \{1, \dots, \ell\}$ there is a color-class-minimal automorphism σ_i of X with $\mathcal{C}[\sigma_i] = \mathcal{C}_i$ that $\mathcal{C}_i$-satisfies F_i.*

Moreover, the automorphisms σ and σ_i can be chosen to satisfy $\sigma_i = \sigma_{\mathcal{C}_i}$ for $1 \leq i \leq \ell$, respectively.

In [1] an algorithm is presented that, when given a hypergraph X on vertex set V with b-bounded color classes and $k \in \mathbb{N}$, computes all color-class-minimal automorphisms of X that have weight exactly k in $\mathcal{O}\big((kb!)^{\mathcal{O}(k^2)} \operatorname{poly}(N)\big)$ time. We use it as a building block for the following algorithm (see line 5).

Algorithm 1 `Color-Exact-CNF-HGA`$_b(X, \mathcal{C}, k, F)$

```
1  Input: A hypergraph $X = (V, E)$ with $b$-bounded color classes $\mathcal{C} = \{C_1, \dots, C_m\}$,
     a parameter $k \in \mathbb{N}$, and a CNF formula $F$ over $\mathrm{Var}(V)$
2  Output: A color-preserving automorphism $\sigma$ of $X$ with $|\mathrm{supp}(\sigma)| = k$ that satisfies $F$,
     or $\perp$ if none exists
3  $A_0 = \{\mathrm{id}\}$
4  for $i \in \{1, \dots, k\}$ do
5      $A_i \leftarrow \{\sigma \in \mathrm{Aut}(X) \mid \sigma$ is color-class-minimal and has weight $i\}$ // see [1]
6  for $h \in \mathcal{H}_{\mathcal{C},k}$ do // $\mathcal{H}_{\mathcal{C},k}$ is the perfect family of hash functions $h\colon \mathcal{C} \to [k]$ from [6]
7      for $\ell \in \{1, \dots, k\}, h'\colon [k] \to [\ell]$ do
8          for $(k_1, \dots, k_\ell) \in \{0, \dots, k\}^\ell$ with $\sum_{i=1}^{\ell} k_i = k$ do
9              for each partition of the clauses of $F$ into $F_0, \dots, F_\ell$ do
10                 if $\forall i \in \{1, \dots, \ell\} : \exists \sigma_i \in A_{k_i} : \mathrm{supp}(\sigma_i) \subseteq \bigcup (h' \circ h)^{-1}(i)$ and $F_i$ is
                     $\mathcal{C}[\sigma_i]$-satisfied by $\sigma_i$, and $F_0$ is $(\mathcal{C} \setminus \bigcup_{i=1}^{\ell} \mathcal{C}[\sigma_i])$-satisfied by id then
11                     return $\sigma = \sigma_1 \cdots \sigma_\ell$
12 return $\perp$
```

▶ **Theorem 3.3.** *Given a hypergraph $X = (V, E)$ with b-bounded color classes $\mathcal{C}$, a CNF formula F over* $\mathrm{Var}(V)$*, and $k \in \mathbb{N}$, the algorithm* `Color-Exact-CNF-HGA`$_b(X, \mathcal{C}, k, F)$ *computes a color-preserving automorphism σ of X of weight k that satisfies F in $(kb!)^{\mathcal{O}(k^2)} k^{\mathcal{O}(|F|)} \operatorname{poly}(N)$ time (where N is the size of X), or determines that none exists.*

Proof. If the algorithm returns $\sigma = \sigma_1 \cdots \sigma_\ell$, we know $\sigma_i \in A_{k_i}$ and $\mathrm{supp}(\sigma_i) \subseteq \bigcup (h' \circ h)^{-1}(i)$. As these sets are disjoint, we have $|\mathrm{supp}(\sigma)| = \sum_{i=1}^{\ell} |\mathrm{supp}(\sigma_i)| = k$, and Lemma 3.2 implies that σ satisfies F.

We next show that the algorithm does not return $\perp$ if there is an automorphism π of X that has weight k and satisfies F. By Lemma 3.2, we can partition $\mathcal{C}[\pi]$ into $\mathcal{C}_1, \dots, \mathcal{C}_\ell$ and the clauses of F into $F_0 \dots, F_\ell$ such that F_0 is $(\mathcal{C} \setminus \mathcal{C}[\pi])$-satisfied by id and, for $1 \le i \le \ell$, the permutation $\pi_i = \pi_{\mathcal{C}_i}$ is a color-class-minimal automorphism of X that $\mathcal{C}[\pi_i]$-satisfies F. Now consider the iteration of the loop where h is injective on $\mathcal{C}[\pi]$; such an h must exist as it is chosen from a perfect hash family. Now let $h'\colon [k] \to [\ell]$ be a function with $h'\big(h(C)\big) = i$ if $C \in \mathcal{C}[\pi_i]$; such an h' exists because h is injective on $\mathcal{C}[\pi]$. In the loop iterations where h' and the partition of F into $F_0 \dots, F_\ell$ is considered, the condition on line 10 is true (at least) with $\sigma_i = \pi_i$, so the algorithm does not return $\perp$.

Line 5 can be implemented by using the algorithm `ColoredAut`$_{k,b}(X)$ from [1] which runs in $\mathcal{O}\big((kb!)^{\mathcal{O}(k^2)} \operatorname{poly}(N)\big)$ time, and this also bounds $|A_i|$. As $|\mathcal{C}| \le n$, the perfect hash family $\mathcal{H}_{\mathcal{C},k}$ has size $2^{\mathcal{O}(k)} \log^2 n$, and can also be computed in this time. The inner loops take at most k^k, k^k and $(k+1)^{|F|}$ iterations, respectively. Together, this yields a runtime of $(kb!)^{\mathcal{O}(k^2)} k^{\mathcal{O}(|F|)} \operatorname{poly}(N)$. ◀

4 Exact weight

In this section, we show that finding isomorphisms that have an exactly prescribed weight and satisfy a CNF formula is fixed parameter tractable. In fact, we show that this is true even for hypergraphs, when the maximum hyperedge size d is taken as additional parameter.

EXACT-CNF-HGI: Given two hypergraphs $X_1 = (V, E_1)$ and $X_2 = (V, E_2)$ with hyperedge size bounded by d, a CNF formula F over $\mathrm{Var}(V)$, and $k \in \mathbb{N}$, decide whether there is an isomorphism from X_1 to X_2 of weight k that satisfies F. The parameter is $|F| + k + d$.

Our approach is to reduce EXACT-CNF-HGI to EXACT-CNF-HGA (the analogous problem for automorphisms), which we solve first.

We require some permutation group theory definitions. Let $G \le \mathrm{Sym}(V)$ be a permutation group. The group G partitions V into orbits: $V = \Omega_1 \cup \Omega_2 \cdots \cup \Omega_r$. On each orbit Ω_i, the group G acts transitively. A subset $\Delta \subseteq \Omega_i$ is a *block* of the group G if for all $\pi \in G$ either $\Delta^\pi = \Delta$ or $\Delta^\pi \cap \Delta = \varnothing$. Clearly, Ω_i is itself a block, and so are all singleton sets. These are *trivial* blocks. Other blocks are *nontrivial.* If G has no nontrivial blocks it is *primitive.* If G is not primitive, we can partition Ω_i into blocks $\Omega_i = \Delta_1 \cup \Delta_2 \cup \cdots \cup \Delta_s$, where each Δ_j is a *maximal nontrivial block.* Then the group G acts primitively on the block system $\{\Delta_1, \Delta_2, \dots, \Delta_s\}$. In this action, a permutation $\pi \in G$ maps Δ_i to $\Delta_i^\pi = \{u^\pi \mid u \in \Delta_i\}$.

The following two theorems imply that every primitive group on a sufficiently large set V contains the alternating group $\mathrm{Alt}(V) = \{\pi \in \mathrm{Sym}(G) \mid \mathrm{compl}(\pi) \text{ is even}\}$.

▶ **Theorem 4.1** [4, Theorem 3.3A]. *Suppose $G \le \mathrm{Sym}(V)$ is a primitive subgroup of $\mathrm{Sym}(V)$. If G contains an element π such that $|\mathrm{supp}(\pi)| = 3$ then G contains $\mathrm{Alt}(V)$. If G contains an element π such that $|\mathrm{supp}(\pi)| = 2$ then $G = \mathrm{Sym}(V)$.*

▶ **Theorem 4.2** [4, Theorem 3.3D]. *If $G \le \mathrm{Sym}(V)$ is primitive with $G \notin \{\mathrm{Alt}(V), \mathrm{Sym}(V)\}$ and contains an element π such that $|\mathrm{supp}(\pi)| = m$ (for some $m \ge 4$) then $|V| \le (m-1)^{2m}$.*

The following lemma implies that the alternating group in a large orbit survives fixing vertices in a smaller orbit.

▶ **Lemma 4.3.** *Let $G \le \mathrm{Sym}(\Omega_1 \cup \Omega_2)$ be a permutation group such that Ω_1 is an orbit of G, and $|\Omega_1| \ge 5$. Recall that $G(\Omega_i)$ denotes the image of G under its action on Ω_i. Suppose $G(\Omega_1) \in \{\mathrm{Alt}(\Omega_1), \mathrm{Sym}(\Omega_1)\}$ and $|G(\Omega_1)| > |G(\Omega_2)|$. Then for some subgroup H of $G(\Omega_2)$, the group G contains the product group $\mathrm{Alt}(\Omega_1) \times H$. In particular, the pointwise stabilizer $G_{[\Omega_2]}$ contains the subgroup $\mathrm{Alt}(\Omega_1) \times \{\mathrm{id}\}$.*

The effect of fixing vertices of some orbit on other orbits of the same size depends on how the group relates these orbits to each other.

▶ **Definition 4.4.** Two orbits Ω_1 and Ω_2 of a permutation group $G \le \mathrm{Sym}(V)$ are *linked* if there is a group isomorphism $\sigma\colon G(\Omega_1) \to G(\Omega_2)$ with $G(\Omega_1 \cup \Omega_2) = \big\{(\varphi, \sigma(\varphi)) \,\big|\, \varphi \in G(\Omega_1)\big\}$. (This happens if and only if both $G(\Omega_1)$ and $G(\Omega_2)$ are isomorphic to $G(\Omega_1 \cup \Omega_2)$.)

We next show that two large orbits where the group action includes the alternating group are (nearly) independent unless they are linked.

▶ **Lemma 4.5.** *Suppose $G \le \mathrm{Sym}(V)$ where $V = \Omega_1 \cup \Omega_2$ is its orbit partition such that $|\Omega_i| \ge 5$ and $G(\Omega_i) \in \{\mathrm{Alt}(\Omega_i), \mathrm{Sym}(\Omega_i)\}$ for $i = 1, 2$. Then either Ω_1 and Ω_2 are linked in G, or G contains $\mathrm{Alt}(\Omega_1) \times \mathrm{Alt}(\Omega_2)$.*

The last ingredient for our algorithm is the observation that when there are two linked orbits where the group action includes the alternating group, fixing a vertex in one orbit is equivalent to fixing some vertex of the other orbit.

▶ **Lemma 4.6** [4, Theorem 5.2A]. *Let $n = |V| > 9$. Suppose G is a subgroup of $\mathrm{Alt}(V)$ of index strictly less than $\binom{n}{2}$. Then, for some point $u \in V$, the group G is the pointwise stabilizer subgroup $\mathrm{Alt}(V)_u$.*

▶ **Corollary 4.7.** *Let Ω_1 and Ω_2 be two linked orbits of a permutation group $G \le \mathrm{Sym}(V)$ with $\mathrm{Alt}(\Omega_1) \le G(\Omega_1)$ and $|\Omega_1| = |\Omega_2| > 9$. Then for each $u \in \Omega_1$ there is a $v \in \Omega_2$ such that $G_u = G_v$.*

Algorithm 2 `Exact-CNF-HGA`$_d(X, k, F)$

```
Input: A hypergraph $X$ with hyperedge size bounded by $d$, a parameter $k$ and a formula $F$
Output: An automorphism $\sigma$ of $X$ with $|\mathrm{supp}(\sigma)| = k$ that satisfies $F$, or $\perp$ if none exists
$T \leftarrow$ the vertices of $X$ that are mentioned in $F$
$G \leftarrow \langle\{\sigma \in \mathrm{Aut}(X) \mid \sigma$ has minimal complexity in $\mathrm{Aut}(X)$ and $|\mathrm{supp}(\sigma)| \le k\}\rangle$
  // see [1, Algorithm 3]
$b \leftarrow \frac{k}{2} \cdot \max\{(k-1)^{2k}, |T| + k, 9\}$
while $G$ contains an orbit of size more than $b$ do
    repeat
        $\mathcal{O} \leftarrow$ the set of all $G$-orbits
        for $\Omega \in \mathcal{O}$ do
            $\mathcal{B}(\Omega) \leftarrow$ a maximal block system of $\Omega$ in $G$
            if $\exists \Delta \in \mathcal{B}(\Omega) : |\Delta| > \frac{k}{2}$ or $|\mathcal{B}(\Omega)| > (k-1)^{2k} \wedge \mathrm{Alt}(\mathcal{B}(\Omega)) \not\le G(\mathcal{B}(\Omega))$ then
                $G \leftarrow G_{\mathcal{B}(\Omega)}$ // the setwise stabilizer of all $\Delta \in \mathcal{B}(\Omega)$
    until $G$ remains unchanged
    choose $\Omega_{\max} \in \mathcal{O}$ such that $|\mathcal{B}(\Omega_{\max})| \ge |\mathcal{B}(\Omega)|$ for all $\Omega \in \mathcal{O}$
    if $|\mathcal{B}(\Omega_{\max})| > \max\{(k-1)^{2k}, |T| + k, 9\}$ then
        $H \leftarrow G_{[T]}$ // the pointwise stabilizer of $T$
        $\Omega_H \leftarrow$ the largest $H$-orbit that is contained in $\Omega_{\max}$
        $\mathcal{B}_H \leftarrow \{\Delta \in \mathcal{B}(\Omega_{\max}) \mid \Delta \subseteq \Omega_H\}$
        choose $\Delta \in \mathcal{B}_H$
        $G \leftarrow G_{\{\Delta\}}$ // the setwise stabilizer of $\Delta$
$\mathcal{O} \leftarrow$ the set of all $G$-orbits
return Color-Exact-CNF-HGA$_b(X, \mathcal{O}, k, F)$ // see Algorithm 1
```

▶ **Theorem 4.8.** *Algorithm 2 solves* Exact-CNF-HGA *in time* $\big(d(k^k + |F|)!\big)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$.

Proof sketch. Suppose there is some $\pi \in \mathrm{Aut}(X)$ of weight exactly k that satisfies F. By Lemma 2.1, the automorphism π can be decomposed as a product of minimal-complexity automorphisms of weight at most k, which implies $\pi \in G$ after line 4. To show that whenever the algorithm shrinks G, some weight k automorphism of X that satisfies F survives, we first consider the shrinking in line 12: If Ω is an orbit with $|\Delta| > k/2$ for some (and thus all) $\Delta \in \mathcal{B}(\Omega)$, then no block of Ω is moved by π. If $|\mathcal{B}(\Omega)| > (k-1)^{2k}$ and $G\big(\mathcal{B}(\Omega)\big)$ does not contain $\mathrm{Alt}\big(\mathcal{B}(\Omega)\big)$, then Theorems 4.1 and 4.2 imply that π setwise stabilizes all $\Delta \in \mathcal{B}(\Omega)$ and thus survives the shrinking. The other shrinking of G, which occurs in line 20, can only happen if $\mathrm{Alt}\big(\mathcal{B}(\Omega_{\max})\big) \le G\big(\mathcal{B}(\Omega_{\max})\big)$. Let $\mathcal{T} = \bigcup_{\Omega \in \mathcal{O}}\{\Delta \in \mathcal{B}(\Omega) \mid \Delta \cap T \ne \varnothing\}$ be the set of all blocks with vertices from T and let $R = G_{\mathcal{T}}$ be the setwise stabilizer of these blocks. Note that $H \le R \le G$. Using Lemmas 4.3 and 4.5 and Corollary 4.7, it can be shown that a sufficiently large part of $\mathrm{Alt}\big(\mathcal{B}(\Omega_{\max})\big)$ survives in R and also in H.

▶ **Claim.** *$\mathcal{B}_H$ is a maximal block system for the orbit Ω_H in H. Moreover, $|\mathcal{B}_H| > k$ and $\mathrm{Alt}(\mathcal{B}_H) \le H(\mathcal{B}_H)$.*

Building on this, it can be shown that when G and Δ are as in line 20, then for any $\pi \in G$ of weight k that satisfies F, there is a $\pi' \in G_{\{\Delta\}}$ of weight k that satisfies F. ◀

We now turn to Exact-CNF-HGI. Given a formula F over $\mathrm{Var}(V)$ and $\psi \in \mathrm{Sym}(V)$, let $\psi(F)$ denote the formula obtained from F by replacing each variable x_{uv} by $x_{u\psi(v)}$.

▶ **Lemma 4.9.** *A product $\sigma = \varphi\pi \in \mathrm{Sym}(V)$ satisfies a formula F over $\mathrm{Var}(V)$ if and only if φ satisfies $\pi^{-1}(F)$.*

Algorithm 3 `Exact-CNF-HGI`$_d(X_1, X_2, k, F)$

```
Input: Two hypergraphs $X_1$ and $X_2$ on vertex set $V$ with hyperedge size bounded by $d$,
  a parameter $k \in \mathbb{N}$ and a CNF formula $F$ over $\mathrm{Var}(V)$
Output: An isomorphism $\sigma$ from $X_1$ to $X_2$ with $|\mathrm{supp}(\sigma)| = k$ that satisfies $F$,
  or $\bot$ if none exists
$\pi \leftarrow$ some isomorphism from $X_1$ to $X_2$ with $|\mathrm{supp}(\pi)| \leq k$ // see [1, Theorem 3.8]
for $U \subseteq \mathrm{supp}(\pi)$ do // we will force $u \notin \mathrm{supp}(\varphi\pi)$ for $u \in U$
    for $M \subseteq \mathrm{supp}(\pi) \setminus U$ do // we will force $u \in \mathrm{supp}(\varphi) \cap \mathrm{supp}(\varphi\pi)$ for $u \in M$
        $I \leftarrow \mathrm{supp}(\pi) \setminus (U \cup M)$ // we will force $u \notin \mathrm{supp}(\varphi)$ for $u \in I$
        $F' \leftarrow \pi^{-1}(F) \wedge \bigwedge_{u\in U} x_{u\pi^{-1}(u)} \wedge \bigwedge_{u\in M}(\neg x_{u\pi^{-1}(u)} \wedge \neg x_{u,u}) \wedge \bigwedge_{u\in I} x_{uu}$
        $k' \leftarrow k - |I| + |U|$
        $\varphi \leftarrow$ Exact-CNF-HGA$_d(X_1, k', F')$ // see Algorithm 2
        if $\varphi \neq \bot$ then return $\sigma = \varphi\pi$
return $\bot$
```

▶ **Theorem 4.10.** *Algorithm 3 solves* Exact-CNF-HGI *in time* $\big(d(k^k+|F|)!\big)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$.

Proof. Suppose Algorithm 3 returns a permutation $\sigma = \varphi\pi$. Then π is an isomorphism from X_1 to X_2 and φ is an automorphism of X_1 that satisfies F' and has weight k'. As φ satisfies $\pi^{-1}(F)$, Lemma 4.9 implies that σ satisfies F. The additional literals in F' ensure $\mathrm{supp}(\sigma) = (\mathrm{supp}(\varphi) \setminus U) \cup I$ and thus $\big|\mathrm{supp}(\sigma)\big| = k' - |U| + |I| = k$.

Now suppose there is an isomorphism σ from X_1 to X_2 that satisfies F and has weight k. Let π be the isomorphism computed on line 3. Then $\varphi = \sigma\pi^{-1}$ is an automorphism of X_1; it satisfies $\pi^{-1}(F)$ by Lemma 4.9. In the iteration of the loops where $U = \big\{u \in \mathrm{supp}(\pi)\cap\mathrm{supp}(\varphi) \,\big|\, u^{\varphi\pi} = u\big\}$ and $M = \big(\mathrm{supp}(\pi)\cap\mathrm{supp}(\varphi)\big)\setminus U$, it holds that φ has weight k' and satisfies F'. Thus `Exact-CNF-HGA`$_d(X_1, k', F')$ does not return $\bot$.

The isomorphism π can be found in $(dk)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ time [1, Theorem 3.8]. The loops have at most 3^k iterations, and `Exact-CNF-HGA`$_d$ takes $\big(d(k^k+|F|)!\big)^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ time. ◀

5 Exact complexity

The complexity of a permutation $\pi \in \mathrm{Sym}(V)$ can be bounded by functions of its weight: $\big|\mathrm{supp}(\pi)\big| - 1 \leq \mathrm{compl}(\pi) \leq 2\cdot\big|\mathrm{supp}(\pi)\big|$. However, there is no direct functional dependence between these two parameters. And while the algorithms of Sections 3 and 4 can be modified to find isomorphisms of exactly prescribed complexity, we give an independent and more efficient algorithm in this section.

The main ingredient is an analysis of decompositions $\sigma = \sigma_1 \cdots \sigma_\ell$ of $\sigma \in \mathrm{Sym}(V)$ into $\sigma_i \in \mathrm{Sym}(V) \setminus \{\mathrm{id}\}$ (for $1 \leq i \leq \ell$) with $\mathrm{compl}(\sigma) = \sum_{i=1}^{\ell} \mathrm{compl}(\sigma_i)$; we call such decompositions *complexity-additive*. For example, the decomposition into complexity-minimal permutations provided by Lemma 2.1 is complexity-additive.

For a sequence of permutations $\sigma_1, \ldots, \sigma_\ell \in \mathrm{Sym}(V)$ and a coloring $c\colon V \to [k]$, its *colored cycle graph* $\mathrm{CG}_c(\sigma_1, \ldots, \sigma_\ell)$ is the incidence graph between $\bigcup_{i=1}^{\ell} \mathrm{supp}(\sigma_i)$ and the σ_i-orbits

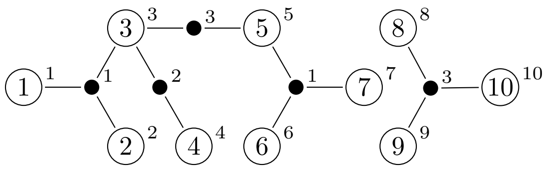

Figure 1 The colored cycle graph $\mathrm{CG}_{\mathrm{id}}\big((1,2,3)(5,6,7),(3,4),(3,5)(8,9,10)\big)$; the colors are depicted next to the vertices.

of size at least 2, i.e., the cycles of σ_i, for $1 \le i \le \ell$. We call the former *primal vertices* and the latter *cycle-vertices*. Each primal vertex $v \in V$ is colored by $c(v)$, and each cycle-vertex that corresponds to a cycle of σ_i is colored by i. (Note that a vertex of this graph is a cycle-vertex if and only if it has odd distance to some leaf.) See Figure 1 for an example.

▶ **Lemma 5.1.** *Let $\sigma \in \mathrm{Sym}(V)$, let $\sigma = \sigma_1 \cdots \sigma_\ell$ be a complexity-additive decomposition, and let $c\colon V \to [k]$ be a coloring. Then $\mathrm{CG}_c(\sigma_1,\dots,\sigma_\ell)$ is a forest.*

A *cycle pattern* P is a colored cycle graph $\mathrm{CG}_c(\sigma_1,\dots,\sigma_\ell)$ where all primal vertices have different colors. A complexity-additive decomposition $\sigma' = \sigma'_1 \cdots \sigma'_\ell$ of a permutation $\sigma' \in \mathrm{Sym}(V)$ *weakly matches* P if there is a coloring $c'\colon V \to [k]$ and a surjective color-preserving homomorphism φ from $\mathrm{CG}_{c'}(\sigma'_1,\dots,\sigma'_\ell)$ to P where $\varphi(u) = \varphi(v)$ for $u \ne v$ implies that u and v are both primal vertices and belong to different σ'-orbits.

▶ **Lemma 5.2.** *For any $t \in \mathbb{N}$, there is a set $\mathcal{P}_t$ of $t^{\mathcal{O}(t)}$ cycle patterns such that a permutation $\sigma \in \mathrm{Sym}(V)$ has complexity t if and only if it has a complexity-additive decomposition $\sigma = \sigma_1 \cdots \sigma_\ell$ that weakly matches a pattern in $\mathcal{P}_t$. Moreover, $\mathcal{P}_t$ can be computed in $t^{\mathcal{O}(t)}$ time.*

For a pattern P, let P_i denote the subgraph of P induced by the cycle-vertices of color i and their neighbors. A permutation $\sigma \in \mathrm{Sym}(V)$ and a coloring $c\colon V \to [k]$ *realize color i of P* if there is an isomorphism φ from $\mathrm{CG}_c(\sigma)$ to P_i that preserves colors of primal vertices.

Algorithm 4 `Exact-Complexity-HGI`$_d(X,Y,t)$

```
Input: Two hypergraphs $X$ and $Y$ on vertex set $V$ with hyperedge size bounded by $d$,
  and $t \in \mathbb{N}$
Output: An isomorphism $\sigma$ from $X$ to $Y$ with $\mathrm{compl}(\sigma) = t$, or $\perp$ if none exists
$A \leftarrow \{\sigma \in \mathrm{Aut}(X) \mid \sigma$ has minimal complexity in $\mathrm{Aut}(X)$ and $|\mathrm{supp}(\sigma)| \le 2t\}$
  // see [1, Algorithm 3]
if $X = Y$ then $I \leftarrow A$ else
    $I \leftarrow \{\sigma \in \mathrm{Iso}(X,Y) \mid \sigma$ has minimal complexity in $\mathrm{Iso}(X,Y)$ and $|\mathrm{supp}(\sigma)| \le 2t\}$
      // see [1, Algorithm 2]
for $P \in \mathcal{P}_t$ do // see Lemma 5.2
    $k \leftarrow$ the number of primal vertices in $P$
    $\ell \leftarrow$ the number of colors of cycle-vertices in $P$
    for $h \in \mathcal{H}_{V,k}$ do // $\mathcal{H}_{V,k}$ is the perfect family of hash functions $h\colon V \to [k]$ from [6]
        if there are $\sigma_1,\dots,\sigma_{\ell-1} \in A$ and $\sigma_\ell \in I$ s.t. $(\sigma_i, h)$ realize color $i$ of $P$ then
            return $\sigma = \sigma_1 \cdots \sigma_\ell$
return $\perp$
```

▶ **Theorem 5.3.** *Given two hypergraphs X and Y of hyperedge size at most d and $t \in \mathbb{N}$, the algorithm* `Exact-Complexity-HGI`$_d(X,Y,t)$ *finds $\sigma \in \mathrm{Iso}(X,Y)$ with $\mathrm{compl}(\sigma) = t$ (or determines that there is none) in $\mathcal{O}\big((dt)^{\mathcal{O}(t^2)}\,\mathrm{poly}(N)\big)$ time.*

Proof. Suppose there is some $\sigma \in \mathrm{Iso}(X, Y)$ with $\mathrm{compl}(\sigma) = t$. Lemma 2.1 gives the complexity-additive decomposition $\sigma = \sigma_1 \cdots \sigma_\ell$ into minimal-complexity permutations $\sigma_1, \ldots, \sigma_{\ell-1} \in \mathrm{Aut}(X)$ and $\sigma_\ell \in \mathrm{Iso}(X, Y)$; all of them have complexity at most t. By the correctness of the algorithms from [1], we have $\sigma_1, \ldots, \sigma_{\ell-1} \in A$ and $\sigma_\ell \in I$. As $\mathcal{H}_{V,k}$ is a perfect hash family, it contains some function h whose restriction to $\mathrm{supp}(\sigma)$ is injective. Then $\mathrm{CG}_h(\sigma_1, \ldots, \sigma_\ell)$ is isomorphic to some $P \in \mathcal{P}_t$ by Lemma 5.2. Thus (σ_i, h) realize color i of P, for $1 \le i \le \ell$, so the algorithm does not return $\perp$.

Now suppose that the algorithm returns $\sigma = \sigma_1 \cdots \sigma_\ell$ with $\sigma_1, \ldots, \sigma_{\ell-1} \in A \subseteq \mathrm{Aut}(X)$ and $\sigma_\ell \in I \subseteq \mathrm{Iso}(X, Y)$. This clearly implies $\sigma \in \mathrm{Iso}(X, Y)$. To show $\mathrm{compl}(\sigma) = t$, we observe that the algorithm only returns σ if there is a pattern $P \in \mathcal{P}_t$ whose cycle-vertices have ℓ colors and which contains k primal vertices such that there is a hash function $h \in \mathcal{H}_{V,k}$ with the property that σ_i and h realize color i of P, for $i \in [\ell]$. In particular, there is an isomorphism φ_i from $\mathrm{CG}_h(\sigma_i)$ to P_i that preserves colors of primal vertices. As the primal vertices of P all have different colors and as P is a forest by Lemma 5.1, it follows that the decomposition $\sigma = \sigma_1 \cdots \sigma_\ell$ is complexity-additive. Now consider the function $\varphi = \bigcup_{i=1}^{\ell} \varphi_i$; it is well-defined, as $v \in \mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j)$ implies $\varphi_i(v) = \varphi_j(v)$ because P contains only one primal vertex of color $h(v)$. It is surjective, as every vertex of P occurs in at least one P_i. It is a homomorphism from $P_\sigma = \mathrm{CG}_h(\sigma_1, \ldots, \sigma_\ell)$ to P, as every edge occurs in the support of one of the isomorphisms φ_i. Also, $\varphi(u) = \varphi(v)$ for $u \neq v$ implies that u and v are in different connected components of P_σ, as P is a forest; consequently u and v are in different orbits of σ. Thus $\sigma = \sigma_1 \cdots \sigma_\ell$ weakly matches P. By Lemma 5.2 it follows that $\mathrm{compl}(\sigma) = t$.

It remains to analyze the runtime. The algorithms used to compute A and I each take $\mathcal{O}\big((dt)^{\mathcal{O}(t^2)} \mathrm{poly}(N)\big)$ time [1]. The pattern set $\mathcal{P}_t$ can be computed in $t^{\mathcal{O}(t)}$ time by Lemma 5.2. As $k \le 2t$, the perfect hash family $\mathcal{H}_{V,k}$ has size $2^{\mathcal{O}(t)} \log^2 n$. As $\ell \le t$, this gives a total runtime of $\mathcal{O}\big((dt)^{\mathcal{O}(t^2)} \mathrm{poly}(N)\big)$. ◀

6 Colored Graph Automorphism

In [1] we showed that the following parameterized version of Graph Automorphism is W[1]-hard. It was first defined in [5] and is a generalization of the problem studied by Schweitzer [13].

Col-GA: Given a graph X with its vertex set partitioned as $\textsc{Red} \cup \textsc{Blue}$, and a parameter k, decide if there is a partition-preserving automorphism that moves exactly k Blue vertices.

For an automorphism $\pi \in \mathrm{Aut}(X)$, we will refer to the number of Blue vertices moved by π as the Blue weight of π. In this section, we show that Col-GA is in FPT when restricted to colored graphs where the color classes inside Red have size at most 3.

Given an input instance $X = (V, E)$ with vertex partition $V = \textsc{Red} \cup \textsc{Blue}$ such that Red is refined into color classes of size at most 3 each, our algorithm proceeds as follows.

Step 1: color-refinement. X already comes with a color classification of vertices (Red and Blue, and within Red color classes of size at most 3 each; within Blue there may be color classes of arbitrary size). The color refinement procedure keeps refining the coloring in steps until no further refinement of the vertex color classes is possible. In a refinement step, if two vertices have identical colors but differently colored neighborhoods (with the multiplicities of colors counted), then these vertices get new different colors.
At the end of this refinement, each color class C induces a regular graph $X[C]$, and each pair (C, D) of color classes induces a semiregular bipartite graph $X[C, D]$.

Step 2: local complementation. We complement the graph induced by a color class if this reduces the number of its edges; this does not change the automorphism group of X. Similarly, we complement the induced bipartite graph between two color classes if this reduces the number of its edges.

Now each color class within RED induces the empty graph. Similarly, for $b \in \{2, 3\}$, the bipartite graph between any two color classes of size b is empty or a perfect matching. (Note that this does not necessarily hold for $b \geq 4$.) Color refinement for graphs of color class size at most 3 has been used in earlier work [8, 9].

Let $C \subseteq$ RED and $D \subseteq$ BLUE be color classes after Step 1. Because of the complementations we have applied, $|C| = 1$ implies that $X[C, D]$ is empty, and if $|C| \in \{2, 3\}$ then $X[C, D]$ is either empty or the degree of each D-vertex in $X[C, D]$ is 1.

Step 3: fix vertices that cannot move. For a color class $C \subseteq$ RED whose elements have more than k BLUE neighbors, give different new colors to each vertex in C (because of Step 2, each non-isolated RED vertex is in a color class with more than one vertex). Afterwards, rerun Steps 1 and 2 so we again have a stable coloring.

Fixing the vertices in C does not lose any automorphism of X that has BLUE-weight at most k. Indeed, as every BLUE vertex has at most one neighbor in C, any automorphism that moves some $v \in C$ has to move all (more than k) BLUE neighbors of v.

Step 4: remove edges in the red part. We already observed that each color class in RED induces the empty graph. Let $\mathcal{X}$ be the graph whose vertices are the color classes in RED, where two of them are adjacent iff there is a perfect matching between them in X. For each $b \in \{1, 2, 3\}$, the color classes in RED of size b get partitioned into components of $\mathcal{X}$. We consider each connected component $\mathcal{C}$ of $\mathcal{X}$ that consists of more than one color class. Let X' be the subgraph of X induced by vertices in $\bigcup \mathcal{C}$ and their neighbors in BLUE. Because of Step 3, the graph X' has color class size at most $3k$, so we can compute its automorphism group $H = \mathrm{Aut}(X')$ in $2^{\mathcal{O}(k^2)} \operatorname{poly}(N)$ time [7]. We distinguish several cases based on the action of H on an arbitrary color class $C \in \mathcal{C}$:

Case 1: If $H(C)$ is not transitive, we split the color class C into the orbits of $H(C)$ and start over with Step 1.

Case 2: If $H(C) = \mathrm{Sym}(C)$, we drop all vertices in $(\bigcup \mathcal{C}) \setminus C$ from X. And for each color class D within BLUE that has neighbors in at least one $C' \in \mathcal{C}$, we replace the edges between a vertex $u \in D$ and $\bigcup \mathcal{C}$ by the single edge (u, v), where v is the vertex in C that is reachable via the matching edges from the neighbor of u in C'.

Case 3: If $H(C)$ is generated by a 3-cycle $(v_1 v_2 v_3)$, we first proceed as in Case 2. Additionally, we add directed edges within each color class D within BLUE that now has neighbors in C. Let $D_i \subseteq D$ be the neighbors of v_i. We add *directed edges* from all vertices in D_i to all vertices in $D_{(i+1) \bmod 3}$ and color these directed edges by C.

After this step, there are no edges induced on the RED part of X. Moreover, we have not changed the automorphisms on the induced subgraph, so the modified graph X still has the same automorphism group as before.

Step 5: turn red vertices into hyperedges. We encode X as a hypergraph $X' = (\text{BLUE} \cup \text{NEW}, E')$ in which each vertex in RED is encoded as a hyperedge on the vertex set BLUE $\cup$ NEW. Let NEW $= \{v_C \mid C \subseteq \text{RED}$ is a color class$\}$. Let $v \in C \subseteq$ RED be any red vertex. We encode v as the hyperedge $e_v = \{v_C\} \cup \{u \in \text{BLUE} \mid (v, u) \in E(X)\}$.

In the hypergraph X' we give distinct colors to each vertex in NEW in order to ensure that each color class $\{v_{C,1}, v_{C,2}, v_{C,3}\}$ in RED is preserved by the automorphisms of X'. Clearly, there is a 1-1 correspondence between the color-preserving automorphisms of X and those of X'. Note that the hyperedges of X' have size bounded by $k + 1$, as each RED vertex in X has at most k BLUE neighbors after Step 3.

Step 6: bounded hyperedge size automorphism. We seek a weight k automorphism of X' using the algorithm of [1, Corollary 6.4];[1] this is possible in $d^{\mathcal{O}(k)}2^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ time.

This algorithm gives us the following.

▶ **Theorem 6.1.** *The above algorithm solves* Col-GA *when the* Red *part of the input graph is refined into color classes of size at most 3. It runs in* $d^{\mathcal{O}(k)}2^{\mathcal{O}(k^2)}\,\mathrm{poly}(N)$ *time.*

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

References

1 Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Jacobo Torán. Parameterized complexity of small weight automorphisms. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 7:1–7:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.7.

2 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

3 László Babai, D. Yu. Grigoryev, and David M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 310–324. ACM, 1982. doi:10.1145/800070.802206.

4 John D. Dixon and Brian Mortimer. *Permutation groups*. Springer, 1996. doi:10.1007/978-1-4612-0731-3.

5 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

6 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984. doi:10.1145/828.1884.

7 Merrick L. Furst, John E. Hopcroft, and Eugene M. Luks. Polynomial-time algorithms for permutation groups. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 36–41. IEEE Computer Society, 1980. doi:10.1109/SFCS.1980.34.

8 Neil Immerman and Eric Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer, 1990. doi:10.1007/978-1-4612-4478-3_5.

9 Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *J. Comput. Syst. Sci.*, 66(3):549–566, 2003. doi:10.1016/S0022-0000(03)00042-4.

10 Pavel Klavík, Dušan Knop, and Peter Zeman. Graph Isomorphism restricted by lists, 2016. URL: https://arxiv.org/abs/1607.03918.

11 Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM J. Comput.*, 46(1):161–189, 2017. doi:10.1137/140999980.

12 Anna Lubiw. Some np-complete problems similar to graph isomorphism. *SIAM J. Comput.*, 10(1):11–21, 1981. doi:10.1137/0210002.

[1] There is a caveat that in addition to hyperedges in the graph $X'[\textsc{Blue}]$ we also have colored directed edges. However, the algorithm of [1, Corollary 6.4] needs only minor changes to handle this.

13 Pascal Schweitzer. Isomorphism of (mis)labeled graphs. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2011. `doi:10.1007/978-3-642-23719-5_32`.