To appear in Groups and Computation II

The complexity of McKay's canonical labeling algorithm

Takunari Miyazaki[⋆]

Department of Computer and Information Science, University of Oregon, 1477 E. 13th Ave., Eugene, OR 97403-1202 (e-mail: `miyazaki@cs.uoregon.edu`)

June 20, 1996

Abstract. We study the time complexity of McKay's algorithm to compute canonical forms and automorphism groups of graphs. The algorithm is based on a type of backtrack search, and it performs pruning by discovered automorphisms and by hashing partial information of vertex labelings. In practice, the algorithm is implemented in the *nauty* package. We obtain colorings of Fürer's graphs that allow the algorithm to compute their canonical forms in polynomial time. We then prove an exponential lower bound of the algorithm for connected 3-regular graphs of color-class size 4 using Fürer's construction. We conducted experiments with *nauty* for these graphs. Our experimental results also indicate the same exponential lower bound.

1. Introduction

The theoretical complexity status of finding canonical representatives for the isomorphism classes of finite algebraic and combinatorial structures is a long-standing unsolved question in computational complexity theory. Since all such structures can be canonically represented by graphs in polynomial time [10], [19], it suffices to solve the problem for graphs (cf. [2]).

A canonical labeling algorithm maps an input graph X to its canonical representative $\mathrm{CF}(X)$ for the isomorphism class such that, for every graph Y, we have $\mathrm{CF}(X) = \mathrm{CF}(Y)$ if and only if $X \cong Y$. We call $\mathrm{CF}(X)$ the *canonical form* of X and its associated labeling the *canonical labeling* of X. A number of studies on computing canonical forms have shown considerable success. In particular, bounds on certain parameters of a graph make polynomial-time so-

[⋆] This work was partly done as the author's Directed Research Project as a University of Oregon doctoral student under the direction of Professor Eugene M. Luks. This paper represents on-going work with Professor Luks, and the final version will appear elsewhere.

lutions possible. Group-theoretic methods led to the first significant result (cf. [3]; see also [12]):

Theorem 1.1 (Babai–Klingsberg–Luks). *For vertex-colored graphs of bounded color-classes* $($*i.e., color-multiplicities*$)$, *canonical forms can be computed in polynomial time.* □

With considerably deeper use of group theory, Babai and Luks went on to show that canonical forms of graphs of bounded degree can be computed in polynomial time [3], and it was also shown independently by Fürer, Schnyder, and Specker [8]. On the other hand, the fastest known algorithm to compute canonical forms for general graphs runs in $O(\exp(n^{1/2+o(1)}))$ time for n-vertex graphs (Luks, Zemlyachenko, cf. [3]; see also [1], [23]).

Older techniques were often based on a type of brute-force backtrack search. In general, the naive brute-force method searches all possible vertex labelings of an n-vertex input graph, yielding $O(n!)$ time. This naive approach can be improved by classifying the vertices of an input graph into invariant classes during the search in order to avoid all possible $O(n!)$ vertex labelings (Read and Corneil [20]; see also §3.1). This vertex classification method is the basis of McKay's canonical labeling algorithm [15], [16], which canonically colors an input graph and finds its automorphism group to compute its canonical form. However, in addition to vertex classification, this remarkable algorithm extensively utilizes the information of discovered automorphisms of an input graph (cf. §3.2) and hashes partial information of vertex labelings (cf. §7) to keep the search space from becoming impractically large. In practice, the algorithm is implemented in the highly regarded *nauty* package [17], which is widely considered to be the fastest practical graph isomorphism package available.

The generalized naive vertex classification method partitions the set of ordered d-tuples of vertices (Weisfeiler and Lehman, cf. [21]; see also §2 in [5] for historical remarks). This method is sometimes called the d-dimensional Weisfeiler-Lehman method (d-$\dim$ W-L) named by Babai. The naive vertex classification method is then the 1-$\dim$ W-L method. The question is, for what d does this method yield a canonical form in $O(n^{d+1})$ time (cf. [2], [4], [5])? Cai, Fürer, and Immerman constructed a class of pairs of non-isomorphic graphs of bounded color-class that force $d = \Omega(n)$ in order for the d-$\dim$ W-L method to distinguish them (cf. [5]; originally announced in [4]):

Theorem 1.2 (Cai–Fürer–Immerman). *There exists a sequence of pairs of connected graphs* $\{X_n, Y_n,\}_{n\in\mathbf{N}}$ *having the following properties:*

(i) X_n *and* Y_n *have* $O(n)$ *vertices.*
(ii) X_n *and* Y_n *are* 3*-regular and have color-class size* 4.
(iii) $X_n \ncong Y_n$.
(iv) $d = \Omega(n)$ *for the* d-$\dim$ *W-L method to be able to answer* $X_n \ncong Y_n$. □

Recall that canonical forms of the Cai–Fürer–Immerman counterexample can be computed in polynomial time using basic group-theoretic machinery by Theorem 1.1. Their counterexample still leaves many questions on canonical labeling algorithms using combinations of combinatorial and group-theoretic methods. In particular, the question we address is, does the Cai–Fürer–Immerman counterexample still yield exponential time for McKay's algorithm? We have positive and negative answers to this question.

In this paper, for expository purposes, we first consider a simplified version of McKay's algorithm and give partial answers to our question. Our simplified algorithm only performs pruning by partial information of discovered automorphisms of an input graph, and it does not perform pruning by hashing partial information of vertex labelings (cf. §3).

We recall the construction of the *Fürer gadget* [7], a novel technique used to construct the Cai–Fürer–Immerman counterexample, and describe its properties (cf. §4). We then prove our first main result (cf. §5.2).

Theorem 1.3. *For every graph with the Fürer gadgets attached* (*including the Cai–Fürer–Immerman counterexample*), *there exists an ordering of its colors such that McKay's algorithm computes its canonical form and its automorphism group in polynomial time.*

However, for a particular family of graphs with the Fürer gadgets attached, some ordering of the colors leads the algorithm to exponential-time computation. We construct a family of connected 3-regular graphs of color-class size 4 that forces the algorithm to compute their canonical forms in exponential time (cf. §6.2, Proposition 6.2). Here, we would like to point out that, in general, for algorithms that make use of the naive vertex classification method such as *nauty,* smaller the color-class size, smaller the search tree.

We then add *nauty*'s hashing capability into our consideration (cf. §7). We consider the complete algorithm, implemented in *nauty,* that performs pruning both by discovered automorphisms and by hashing partial information of vertex labelings. Given any trivalent graph with the Fürer gadgets attached as an input graph, we prove that graph invariants used in hashing do not distinguish non-equivalent colorings during any stage of the search. Hence, we conclude that our exponential example for the simplified algorithm forces even the complete algorithm to exponential-time computation (cf. §8).

Theorem 1.4. *There exists a sequence of connected graphs* $\{X_n\}_{n\in\mathbf{N}}$ *having the following properties:*

(i) X_n *has* $O(n)$ *vertices.*
(ii) X_n *is* 3*-regular and has color-class size* 4.
(iii) *McKay's algorithm generates* $\Omega(c^n)$ *vertex labelings to compute the canonical form and the automorphism group of* X_n *for some fixed constant* $c > 1$.

We conducted experiments with the *nauty* system for the family of graphs that forces McKay's algorithm to exponential time with some color ordering (cf. §10). Experiments were conducted with three types of color ordering: one that leads McKay's algorithm to polynomial time, one that forces the algorithm to exponential time, and one whose colors are ordered randomly. Our experimental results indicate the same exponential lower bound (obtained in §6.2, Proposition 6.2).

2. Preliminaries

We begin with definitions and describe fundamentals of the naive vertex classification method (see [1], [2], [3]). Throughout, all graphs we consider are finite and simple (i.e., undirected without loops and multiple edges) unless noted, and all colored graphs we consider are vertex-colored graphs defined as follows.

2.1. Vertex coloring

Let $X = (V, E)$ be a graph, where $|V| = n$. A *coloring* of the vertex set V is a function $\varphi : V \to \Omega = \{1, 2, \ldots, n\}$ such that the actual image $\varphi(V)$ is an initial segment of $\{1, 2, \ldots, n\}$. The *color-classes* are the sets $W_i[\varphi] = \{v \in V \mid \varphi(v) = i\}$. A coloring φ is defined by a sequence of non-empty color-classes $\mathcal{W}_\varphi = (W_1[\varphi], W_2[\varphi], \ldots, W_k[\varphi])$, $k \leq n$. The *color-class size* of φ is the cardinality of the largest color-class of φ. We define $\mathcal{C}(V)$ to be the family of all colorings on V.

Let $\mathrm{Sym}(V)$ be the symmetric group of permutations on V. There is a natural action of $\mathrm{Sym}(V)$ on $\mathcal{C}(V)$ defined by $\varphi^g(v) = \varphi(v^{g^{-1}})$ for $\varphi \in \mathcal{C}(V)$, $v \in V$, and $g \in \mathrm{Sym}(V)$. Then we have $\mathcal{W}_{\varphi^g} = (W_1[\varphi^g], W_2[\varphi^g], \ldots, W_k[\varphi^g])$, where each $W_i[\varphi^g] = (W_i[\varphi])^g$ for $1 \leq i \leq k$.

2.2. Canonical form problem

Let $\mathcal{K}(V) = 2^{\binom{V}{2}}$ be the family of all graphs having a vertex set V. For $X = (V, E)$, there is a natural action of $\mathrm{Sym}(V)$ on $E \subseteq \binom{V}{2}$ given by $\{u, v\}^g = \{u^g, v^g\}$ for $u, v \in V$ and $g \in \mathrm{Sym}(V)$. We define $X^g = (V, E^g)$. Two graphs $X, Y \in \mathcal{K}(V)$ are *isomorphic*, denoted by $X \cong Y$, if $Y = X^g$ for some $g \in \mathrm{Sym}(V)$. The *automorphism group* of $X \in \mathcal{K}(V)$ is $\mathrm{Aut}(X) = \{g \in \mathrm{Sym}(V) \mid X^g = X\}$.

The *canonical form problem* is to find a function, called a *canonical form*, $\mathrm{CF} : \mathcal{K}(V) \to \mathcal{K}(V)$ such that, for all $X \in \mathcal{K}(V)$ and $g \in \mathrm{Sym}(V)$,

(i) $\mathrm{CF}(X) \cong X$, and
(ii) $\mathrm{CF}(X^g) = \mathrm{CF}(X)$.

Given two graphs $X, Y \in \mathcal{K}(V)$, we have $X \cong Y$ if and only if $\mathrm{CF}(X) = \mathrm{CF}(Y)$.

We now extend the above notions to colored graphs. Let $\mathcal{K}(V) \times \mathcal{C}(V)$ be the set of all vertex-colored graphs having a vertex set V. Two colored graphs $(X, \varphi), (Y, \psi) \in \mathcal{K}(V) \times \mathcal{C}(V)$ are *isomorphic*, denoted by $(X, \varphi) \cong (Y, \psi)$, if $Y = X^g$ and $\psi = \varphi^g$ for some $g \in \mathrm{Sym}(V)$. The *color-preserving automorphism group* of (X, φ) is $\mathrm{Aut}(X, \varphi) = \{g \in \mathrm{Sym}(V) \mid X = X^g \text{ and } \varphi = \varphi^g\}$. Now, a *canonical form* is a function $\mathrm{CF} : \mathcal{K}(V) \times \mathcal{C}(V) \to \mathcal{K}(V) \times \mathcal{C}(V)$ such that, for all $X \in \mathcal{K}(V)$, $\varphi \in \mathcal{C}(V)$, and $g \in \mathrm{Sym}(V)$,

(i) $\mathrm{CF}(X, \varphi) \cong (X, \varphi)$,

(ii) $\mathrm{CF}(X^g, \varphi^g) = \mathrm{CF}(X, \varphi)$; and
(iii) if $\mathrm{CF}(X, \varphi^g) = \mathrm{CF}(X, \varphi)$, then $\varphi^g = \varphi^h$ for some $h \in \mathrm{Aut}(X)$.

Fix $g \in \mathrm{Sym}(V)$. Then given two colored graphs $(X, \varphi), (Y, \varphi^g) \in \mathcal{K}(V) \times \mathcal{C}(V)$, we have $(X, \varphi) \cong (Y, \varphi^g)$ if and only if $\mathrm{CF}(X, \varphi) = \mathrm{CF}(Y, \varphi^g)$.

2.3. *Naive vertex classification*

The naive vertex classification method is described by the following color refinement procedure.

Consider a vertex set V and a coloring $\varphi : V \to \Omega$. A coloring φ' is a *refinement* of φ if $\varphi'(u) \leq \varphi'(v)$ implies $\varphi(u) \leq \varphi(v)$ for all $u, v \in V$. A coloring φ is *discrete* if φ is a bijection from V onto Ω.

Let $X = (V, E)$ be a graph, where $|V| = n$, with a coloring $\varphi : V \to \Omega$. We now define a refinement φ' as follows. For $v \in V$, let $d_i(v)$ be the number of neighbors of v having color i, and consider the following tuple:

$$D(v) = (\varphi(v), d_1(v), d_2(v), \ldots, d_n(v)).$$

We sort these new colors lexicographically and define $\varphi'(v)$ to be the ordering number of the new color-class to which v belongs. We keep refining the coloring until at some level $\varphi^{(r)} = \varphi^{(r+1)}$ for some $r < n$. We define $\overline{\varphi} = \varphi^{(r)}$, and we call $\overline{\varphi}$ the *stable refinement* of φ.

3. McKay's algorithm

McKay's algorithm is based on a depth-first search through a tree whose nodes are stable vertex colorings. At each stage, a vertex is chosen and separated as a singleton color-class by assigning a new color. First, for expository purposes, we consider the the simplified version of the algorithm (which only performs pruning by discovered automorphisms) as follows.

Throughout, we denote McKay's algorithm by $\mathsf{nauty}(X, \varphi)$ and the simplified algorithm by $\mathsf{nauty}'(X, \varphi)$.

3.1. *Basic searching method*

Suppose we are given a colored graph (X, π), where $X = (V, E)$ and $|V| = n$. The algorithm computes $\mathrm{CF}(X, \pi)$ by finding the canonical coloring that produces the lexicographically first adjacency matrix and $\mathrm{Aut}(X, \pi)$.

The search begins with the initial coloring π, which is the *root* of the search tree. Let φ be an arbitrary node of the search tree defined by $\mathcal{W}_\varphi = (W_1, W_2, \ldots, W_k)$. If φ is discrete, it is an end-node (leaf) of the tree, and we call it a *terminal coloring*. If φ is not discrete, a vertex is chosen and turned into a singleton color-class as follows. Let W_i be the first non-trivial color-class (i.e., the first color-class having size greater than 1) of the smallest size. Such W_i is called the *target color-class*. For $v \in W_i$, we define the new coloring $\mathrm{ind}(\varphi, v)$ by

$$\mathcal{W}_{\mathrm{ind}(\varphi,v)} = (W_1, \ldots, W_{i-1}, W_i \setminus \{v\}, W_{i+1}, \ldots, W_k, \{v\}).$$

Let φ_v denote the stable refinement of $\mathrm{ind}(\varphi, v)$. We call φ_v the *stabilizer* of v. The vertex v is a singleton under φ_v. For $S = (v_1, v_2, \ldots, v_s)$, an ordered sequence of vertices, we define the stabilizer of S to be $\varphi_S = \varphi_{v_1 v_2 \cdots v_s}$ obtained by repeated application of the vertex refinement algorithm. The coloring φ_S is stable, and the vertices $v_1, v_2, \ldots, v_s$ form singleton color-classes. The successors of φ in the search tree are φ_v for each $v \in W_i$. Thus, the total number of successors of φ is $|W_i|$.

Let $T = \mathrm{Tree}(X, \pi)$ be the search tree generated by the naive method described above. Let $L = \mathrm{Leaf}(X, \pi)$ be the set of terminal colorings in T. Given any $\psi \in L$, we can form the graph $\psi(X)$ by relabeling the vertices of X with the discrete coloring ψ. We define the equivalence relation $\sim$ on L by $\psi_1 \sim \psi_2$ if $\psi_1(X) = \psi_2(X)$. Then for any fixed $\tau \in L$, we have

$$\mathrm{Aut}(X, \pi) = \{g \in \mathrm{Sym}(V) \mid \psi = \tau^g, \psi \in L, \text{ and } \psi \sim \tau\}.$$

By comparing the adjacency matrices lexicographically, we can define an order on $\mathcal{K}(V)$ and define $\mathrm{CF}(X, \pi) = \max\{\psi(X) \mid \psi \in L\}$.

3.2. Automorphism pruning

Since $\mathrm{Aut}(X, \pi)$ acts semiregularly on L, we have $|L| = c\,|\mathrm{Aut}(X, \pi)|$ for some integer $c > 0$. So it is often not feasible to generate the entire search tree. This problem is largely overcome by using discovered automorphisms of X to eliminate sections of the search tree from consideration.

Consider the search tree T. Let $\zeta \in L$ be the first terminal node (i.e., the leftmost leaf). Let ρ be the best candidate[1] for the canonical node we have so far.

3.2.1. Orbit pruning. For a non-terminal node φ in T, let $G_\varphi = \langle A_\varphi \rangle \leq \mathrm{Aut}(X, \pi)$, where A_φ is the set of the generators found so far that stabilize the color-classes of φ (i.e., $\varphi^g = \varphi$ for $g \in A_\varphi$). When the algorithm visits a non-terminal node φ in T, its child is generated according to the orbits of G_φ. Let $\overline{V} = \{\Delta_1, \Delta_2, \ldots, \Delta_r\}$ be the orbits of G_φ, and let u_i be the minimum[2] vertex in Δ_i for $1 \leq i \leq r$. We define $\mathrm{Reps}(\overline{V}) = \{u_1, u_2, \ldots, u_r\}$. At each non-terminal node φ, the algorithm fixes vertices in its target color-class, say $W = \{v_1, v_2, \ldots, v_s\}$, to generate its children. In particular, out of these s choices, it chooses v_i such that $v_i \in \mathrm{Reps}(\overline{V})$ because choosing vertices in the same orbit yields equivalent terminal nodes under some $g \in G_\varphi$.

[1] By "best," we mean the terminal node whose coloring induces the lexicographical leader.
[2] We mean minimum in the lexicographical ordering.

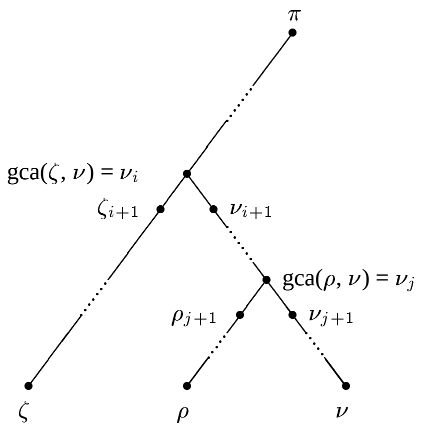

Figure 1. Pruning by automorphism equivalence

3.2.2. Backtracking by automorphism equivalence. Now, for a terminal node ν in T, let ν_i be the ancestor of ν at level i in T.[3] For a node φ in T, let T_φ be the subtree of T rooted at φ. Consider the greatest common ancestor of ζ and ν, denoted by $\mathrm{gca}(\zeta,\nu)$. Here, $\mathrm{gca}(\zeta,\nu) = \zeta_i = \nu_i$ for some $i \geq 0$. Now, ζ_{i+1} is the child of $\mathrm{gca}(\zeta,\nu)$ that is on the unique path down to ζ, and ν_{i+1} is its another child that is on the unique path down to ν. Similarly, we have $\mathrm{gca}(\rho,\nu) = \rho_j = \nu_j$ for some $0 \leq j \leq i$. See Figure 1 for an example.

Automorphism pruning at terminal nodes is performed by considering the following three cases. Suppose the algorithm is generating a terminal node ν.

Case (a). Let $g \in \mathrm{Sym}(V)$ such that $\zeta^g = \nu$. Suppose $g \in \mathrm{Aut}(X,\pi)$. First, g is stored as a new discovered automorphism. Now, the subtrees $T_{\zeta_{i+1}}$ and $T_{\nu_{i+1}}$ are equivalent under g; that is, each node in $T_{\zeta_{i+1}}$ can be relabeled by g to get $T_{\nu_{i+1}}$. Thus, there is no need to search the subtree $T_{\nu_{i+1}}$. So we may immediately return to $\mathrm{gca}(\zeta,\nu)$ to skip the search of $T_{\nu_{i+1}}$ as soon as we find such $g \in \mathrm{Aut}(X,\pi)$.

Case (b). Suppose the condition of Case (a) fails (i.e., $g \notin \mathrm{Aut}(X,\pi)$). Then let $h \in \mathrm{Sym}(V)$ such that $\rho^h = \nu$. Suppose $h \in \mathrm{Aut}(X,\pi)$. Here, $\nu_i = \mathrm{gca}(\zeta,\nu)$, and let $\overline{V}$ denote the orbits of G_{ν_i}. Suppose ν_{i+1} is generated from ν_i by fixing a vertex $v \in \mathrm{Reps}(\overline{V})$. If h induces new orbits of G_{ν_i}, say $\overline{V}'$, and $v \notin \mathrm{Reps}(\overline{V}')$, then we may immediately return to $\mathrm{gca}(\zeta,\nu)$ because $T_{\zeta_{i+1}}$ and $T_{\nu_{i+1}}$ are equivalent under some $t \in G_{\nu_i}$. Otherwise, we may immediately return to $\mathrm{gca}(\rho,\nu)$ since $T_{\rho_{j+1}}$ and $T_{\nu_{j+1}}$ are equivalent under $h \in \mathrm{Aut}(X,\pi)$.

Case (c). If the both conditions of Cases (a) and (b) fail, we simply return to the parent of ν.

[3] The root of T is defined to be at level 0.

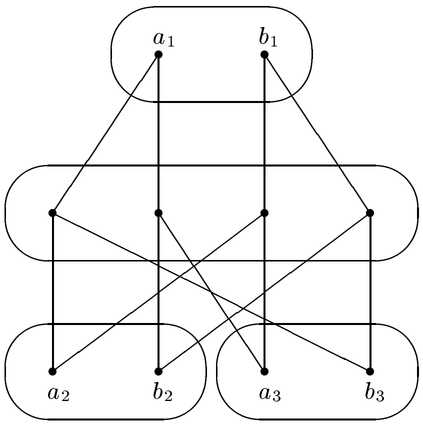

Figure 2. The graph Ξ_3

4. The Fürer gadgets

4.1. Construction

In [7], describing the construction of his counterexample for the d-$\dim$ W-L method for bounded d, Fürer constructs gadgets to form a pair of non-isomorphic graphs. Here, we recall the construction of his gadgets, called the *Fürer gadgets*.

Let Y be a graph of size m (i.e., $|\mathcal{V}(Y)| = m$) such that $\deg(Y, v) \geq 2$ for each $v \in \mathcal{V}(Y)$. WLOG, we assume Y is connected throughout. We replace each vertex $v \in \mathcal{V}(Y)$ of degree k by the graph Ξ_k defined as follows [5]:

(i) $\mathcal{V}(\Xi_k) = A_k \bigcup B_k \bigcup M_k$, where $A_k = \{a_i \mid 1 \leq i \leq k\}$, $B_k = \{b_i \mid 1 \leq i \leq k\}$, and $M_k = \{m_S \mid S \subseteq \{1, 2, \ldots, k\}$ and $|S|$ is even$\}$;
(ii) $\mathcal{E}(\Xi_k) = \{\{m_S, a_i\} \mid i \in S\} \bigcup \{\{m_S, b_i\} \mid i \notin S\}$.

Here, we color each pair of vertices a_i and b_i with a unique color, and we color the middle vertices M_k with a different color from the others. Let θ be such a coloring, then the resulting colored graph is (Ξ_k, θ).

Thus, Ξ_k consists of a set of 2^{k-1} vertices in M_k each connected to one vertex from each of the pairs $\{a_i, b_i\}, 1 \leq i \leq k$. Furthermore, each of the middle vertices is connected to an even number of a_i's.

Lemma 4.1 ([5]**).** *Let $g \in \mathrm{Aut}(\Xi_k, \theta)$, then g stabilizes $\{a_1, b_1\}, \ldots, \{a_k, b_k\}$. So $|\mathrm{Aut}(\Xi_k, \theta)| = 2^{k-1}$. Each $g \in \mathrm{Aut}(\Xi_k, \theta)$ is determined by interchanging a_i and b_i for each i in some subset S of $\{1, 2, \ldots, k\}$ of even cardinality.* □

The graph $\mathcal{X}(Y)$ is defined as follows [5]:

(i) For each vertex $v \in \mathcal{V}(Y)$ of degree k, we replace v by a copy of Ξ_k. We denote such Ξ_k by $\mathcal{X}(v)$, its middle vertex set M_k by $\mathcal{M}(v)$, and its coloring θ by ϑ_v.

(ii) To each edge $e = \{v, w\} \in \mathcal{E}(Y)$, adjacent to v, we associate a pair $\{a_i, b_i\}$ from $\mathcal{X}(v)$. We call this pair $\{a(v, w), b(v, w)\}$.
(iii) We connect the a vertices and the b vertices at each end of each edge. That is, we draw the edges $\{a(v, w), a(w, v)\}$ and $\{b(v, w), b(w, v)\}$. We denote this pair of edges by $\mathcal{X}(e)$.

For each $v \in \mathcal{V}(Y)$, the graph $\mathcal{X}(v)$ is colored uniquely by ϑ_v. The resulting colored graph is denoted by $(\mathcal{X}(Y), \vartheta)$.

Note. The coloring ϑ colors each $\mathcal{X}(v)$ with unique $k+1$ colors, so ϑ colors the entire graph $\mathcal{X}(Y)$ with $\sum_{v\in\mathcal{V}(Y)}(\deg(Y, v)+1)$ colors. If $d = \max\{\deg(Y, v) \mid v \in \mathcal{V}(Y)\}$, then $(\mathcal{X}(Y), \vartheta)$ has color-class size 2^{d-1}.

Now, let $e = \{v, w\} \in \mathcal{E}(Y)$, and consider its corresponding pairs of edges in $\mathcal{X}(e)$, say $\{e_1, e_2\} = \{\{a(v, w), a(w, v)\}, \{b(v, w), b(w, v)\}\}$. Suppose we have $g \in \operatorname{Aut}(\mathcal{X}(Y), \vartheta)$ that swaps e_1 with e_2 (i.e., g swaps $a(v, w)$ with $b(v, w)$ and $a(w, v)$ with $b(w, v)$). Then g *twists* e. For a cycle C in Y, we say g *twists* C if g twists each $e \in \mathcal{E}(C)$.

Observation 4.2. *Let $g \in \operatorname{Aut}(\mathcal{X}(Y), \vartheta)$. Then g is determined by twisting edge-disjoint cycles of Y.* □

Remark. $\operatorname{Aut}(\mathcal{X}(Y), \vartheta)$ is a 2-group.

4.2. Individualization

We now define our notion of individualization of vertices. We then describe some properties of individualization on $(\mathcal{X}(Y), \vartheta)$ when the color refinement algorithm is applied.

Let (X, π) be a colored graph. A vertex $v \in \mathcal{V}(X)$ is *individualized* if v has its own unique color. We say we *individualize* a vertex $v \in \mathcal{V}(X)$ when we separate v by assigning a unique color to v and applying the color refinement algorithm to stabilize this new coloring. For $(\mathcal{X}(Y), \varphi)$, where φ is arbitrary, a vertex $v \in \mathcal{V}(Y)$ is *$\mathcal{X}$-individualized* if each vertex in $(\mathcal{X}(v), \varphi|_{\mathcal{X}(v)})$ has its own unique color. An edge $e \in \mathcal{E}(Y)$ is *$\mathcal{X}$-individualized* if each vertex in $(\mathcal{X}(e), \varphi|_{\mathcal{X}(e)})$ has its own unique color.

Observation 4.3. *For $(\mathcal{X}(Y), \vartheta)$, let $v \in \mathcal{V}(Y)$ and $k = \deg(Y, v)$.*

(i) *If we individualize a vertex in $\mathcal{M}(v) \subsetneq \mathcal{V}(\mathcal{X}(v))$, then v is $\mathcal{X}$-individualized.*
(ii) *Suppose we only individualize pairs of vertices a_i and b_i of $\mathcal{X}(v)$ in some order. Then v is $\mathcal{X}$-individualized exactly when $k-1$ pairs of vertices a_i and b_i are individualized.*
(iii) *The target color-class has size 2 during any stage of individualization.* □

We define a subgraph of Y induced by a coloring on $\mathcal{X}(Y)$.

Definition. Given $(\mathcal{X}(Y), \vartheta)$, let φ be a stable color refinement of ϑ. For $(\mathcal{X}(Y), \varphi)$, let $\Gamma(Y, \varphi)$ be the subgraph of Y defined by

(i) $\mathcal{V}(\Gamma(Y,\varphi)) = \{v \in \mathcal{V}(Y) \mid\ v$ is not $\mathcal{X}$-individualized with $\varphi\}$, and
(ii) $\mathcal{E}(\Gamma(Y,\varphi)) = \{e \in \mathcal{E}(Y) \mid\ e$ is not $\mathcal{X}$-individualized with $\varphi\}$.

Lemma 4.4. *Given $(\mathcal{X}(Y),\vartheta)$, let φ be a stable color refinement of ϑ. If $\Gamma(Y,\varphi) \neq \varnothing$, then $\Gamma(Y,\varphi)$ contains a cycle.*

Proof. Suppose $\Gamma(Y,\varphi)$ is a forest. Let $U \subseteq \mathcal{V}(\Gamma(Y,\varphi))$ be the set of all vertices of degree 1 in $\Gamma(Y,\varphi)$. Let $v \in U$, and suppose $\deg(Y,v) = k$. There are $k-1$ $\mathcal{X}$-individualized edges adjacent to v. By Observation 4.3 (ii), v must be $\mathcal{X}$-individualized. Thus, $v \notin \mathcal{V}(\Gamma(Y,\varphi))$, a contradiction. □

5. Colorings that allow polynomial-time solutions

5.1. Coloring scheme

We now introduce a coloring scheme that allows the algorithm nauty′, described in §3, to compute the canonical form of $(\mathcal{X}(Y),\vartheta)$ in polynomial time.

Lemma 5.1. *Given $(\mathcal{X}(Y),\vartheta)$, consider applying the depth-first vertex classification algorithm. There exists an ordering of the colors of $(\mathcal{X}(Y),\vartheta)$ such that, for every generated sequence of stable refinements $\vartheta = \varphi_0, \varphi_1, \ldots, \varphi_r = \psi \in \mathrm{Leaf}(\mathcal{X}(Y),\vartheta)$, each graph $\Gamma(Y,\varphi_i), 0 \le i \le r-1$, is connected.*

Proof. We show, given $(\mathcal{X}(Y),\varphi_i)$, where $\Gamma(Y,\varphi_i)$ is connected, the vertex classification algorithm can individualize a vertex in some non-trivial color-class having the smallest size (not necessarily the target color-class) so that $\Gamma(Y,\varphi_{i+1})$ is connected for $0 \le i \le r-1$. Since $\Gamma(Y,\varphi_i)$ is connected, it contains a cycle.

Suppose $\Gamma(Y,\varphi_i)$ itself forms a cycle. Let φ_{i+1} be the resulting stable color refinement after individualizing a vertex in $\mathcal{X}(v)$ for some $v \in \mathcal{V}(\Gamma(Y,\varphi_i))$. Then $\Gamma(Y,\varphi_{i+1}) = \varnothing$.

Suppose $\Gamma(Y,\varphi_i)$ properly contains a cycle, say C. There is an edge $e = \{v,w\} \in \mathcal{E}(C)$ such that $\deg(\Gamma(Y,\varphi_i),v) \ge 3$. Let φ_{i+1} be the resulting stable color refinement after individualizing a vertex in $\mathcal{X}(e)$. Then the remaining components of C keep $\Gamma(Y,\varphi_{i+1})$ connected. □

5.2. Polynomial-time solutions

We prove our first main result. Let ϑ_A be a coloring obtained by permuting the color-classes of ϑ so that it satisfies the condition of Lemma 5.1.

Proposition 5.2. *Given $(\mathcal{X}(Y),\vartheta_A)$, where $n = |\mathcal{V}(\mathcal{X}(Y))|$, the algorithm* nauty′ *generates a search tree having $O(n^2)$ nodes to compute $\mathrm{CF}(\mathcal{X}(Y),\vartheta_A)$ and $\mathrm{Aut}(\mathcal{X}(Y),\vartheta_A)$.*

Proof. We utilize the fact that each automorphism is determined by twisting edge-disjoint cycles of Y (cf. Observation 4.2). At each stage of individualization, we twist a cycle in $\Gamma(Y, \varphi)$, eventually leading to an automorphism when we reach a terminal node. We write $V = \mathcal{V}(\mathcal{X}(Y))$. Let T be the search tree generated by the algorithm. Let ζ be the leftmost leaf of T and ζ_i be the ancestor of ζ at level i. Suppose the height of T_{ζ_i} is h_i. We have the following claim:

Claim. The number of nodes in T_{ζ_i} is $t_1(h_i) = \frac{1}{2}(h_i + 1)(h_i + 2)$.

To confirm our claim, for every leaf ν in T, we show $g \in \mathrm{Sym}(V)$ such that $\zeta^g = \nu$ is an automorphism, so the algorithm returns to the parent of $\mathrm{gca}(\zeta, \nu)$.

We induct from the leftmost leaf ζ. Let ζ_i be the parent of ζ and ν be ζ's another child. Now, $\Gamma(Y, \zeta_i)$ itself forms a cycle. Then $g \in \mathrm{Sym}(V)$ such that $\zeta^g = \nu$ is an automorphism that twists this cycle. Here, $h_i = 1$ and $t_1(h_i) = 3$.

Suppose that the algorithm is visiting some ancestor of ζ, say ζ_i at level i. Now, ζ_{i-1} is the parent of ζ_i. Let ν_i be ζ_{i-1}'s another child and ν be the leftmost leaf generated from ν_i. By Observation 4.3, $\Gamma(Y, \zeta_i)$ is connected and contains a cycle. Now, there is an edge $e \in \mathcal{E}(\Gamma(Y, \zeta_{i-1})) \setminus \mathcal{E}(\Gamma(Y, \zeta_i))$ that gets twisted by any $g \in \mathrm{Sym}(V)$ such that $\zeta_i^g = \nu_i$. The edge e is on a cycle in $\Gamma(Y, \zeta_{i-1})$ because $\Gamma(Y, \zeta_i)$ would be disconnected otherwise. Then we have the following two cases.

(a) e is on a cycle in $\Gamma(Y, \zeta_{i-1}) \setminus \Gamma(Y, \zeta_i)$,[4] or else
(b) e is on a cycle in $\Gamma(Y, \zeta_{i-1})$, and parts of this cycle are in $\Gamma(Y, \zeta_i)$.

In either case, let C be such a cycle. Eventually, ν induces the automorphism that only twists C.

By induction, the number of nodes in T_{ζ_i} is $t_1(h_i)$, and the height of $T_{\zeta_{i-1}}$ is $h_i + 1$. Since the algorithm returns to the parent of ζ_{i-1} immediately after ν is generated, the number of nodes in $T_{\zeta_{i-1}}$ is given by $t_1(h_i) + h_i + 2 = t_1(h_{i-1})$.

The height of T is $O(n)$. Consequently, by our claim, the number of tree nodes generated is $O(n^2)$. $\square$

6. An exponential lower bound

We construct a family of 3-regular graphs of color-class size 4 that forces McKay's algorithm to compute their canonical forms in exponential time.

6.1. Construction

An edge e is a *bridge* (or *isthmus*) if the removal of e increases the number of connected components. First, we construct a multigraph Y_α that has α bridges and $\alpha + 1$ cycles.

Definition. Let $\alpha > 0$ be an integer. Let Y_α be a multigraph defined by $\mathcal{V}(Y_\alpha) = \{v_1, \dots, v_\alpha, w_1, \dots, w_\alpha\}$ and $\mathcal{E}(Y_\alpha) = E_1 \cup E_2 \cup E_3$, where

[4] The graph $X_1 \setminus X_2$ is the subgraph induced by the vertex set $\mathcal{V}(X_1) \setminus \mathcal{V}(X_2)$.

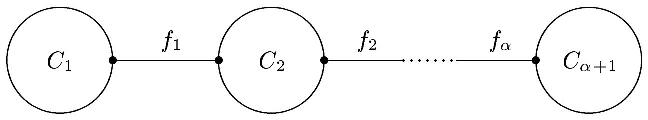

Figure 3. The graph Y_α

(i) $E_1 = \{e_1, e_{\alpha+1} \mid e_1 = \{v_1, v_1\}$ and $e_{\alpha+1} = \{w_\alpha, w_\alpha\}\}$,
(ii) $E_2 = \{e_i, e_i' \mid e_i = e_i' = \{w_{i-1}, v_i\}, 2 \le i \le \alpha\}$, and
(iii) $E_3 = \{f_i \mid \{v_i, w_i\}, 1 \le i \le \alpha\}$.

Let C_1 be the first cycle of Y_α consisting e_1 alone, $C_{\alpha+1}$ be the $(\alpha+1)$st cycle of Y_α consisting $e_{\alpha+1}$ alone, and C_i be the ith cycle of Y_α consisting e_i and e_i' for $2 \le i \le \alpha$.

Now, we apply Fürer's construction to Y_α. We denote the resulting graph by $(\mathcal{X}(Y_\alpha), \vartheta)$. Here, observe that $\mathcal{X}(Y_\alpha)$ is now a simple graph (i.e., a graph without loops and multiple edges) even though Y_α is not.

Remarks. The graph Y_α is a 3-regular graph, so $(\mathcal{X}(Y_\alpha), \vartheta)$ has 8α color-classes and color-class size 4. The automorphism group of $(\mathcal{X}(Y_\alpha), \vartheta)$ is an elementary abelian 2-group. In particular,

$$\mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta) \cong \underbrace{\mathbf{Z}_2 \times \cdots \times \mathbf{Z}_2}_{\alpha+1 \text{ times}},$$

where each $\mathbf{Z}_2$ corresponds to twisting a cycle C_i of Y_α.

Observation 6.1. *Given $(\mathcal{X}(Y_\alpha), \vartheta)$, consider applying the depth-first vertex classification algorithm. Denote a sequence of stable color refinements from the root of the search tree ϑ to a leaf ψ by $\vartheta = \varphi_0, \varphi_1, \ldots, \varphi_t = \psi$. There exists an ordering of the colors of $(\mathcal{X}(Y_\alpha), \vartheta)$ such that $f_1, f_2, \ldots, f_i$ are the only $\mathcal{X}$-individualized edges when φ_i is generated for $1 \le i \le \alpha$.* □

6.2. *Exponential-time solutions*

We prove our second main result. Let ϑ_B be a coloring obtained by permuting the color-classes of ϑ so that it satisfies the condition of Observation 6.1. The algorithm generates exponentially many nodes in the search tree to compute the canonical form of $(\mathcal{X}(Y_\alpha), \vartheta_B)$.

Proposition 6.2. *Given $(\mathcal{X}(Y_\alpha), \vartheta_B)$, where $n = |\mathcal{V}(\mathcal{X}(Y_\alpha))|$, the algorithm* nauty' *generates $\Omega(c^n)$ nodes in the search tree to compute $\mathrm{CF}(\mathcal{X}(Y_\alpha), \vartheta_B)$ and $\mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta_B)$ for some fixed constant $c > 1$.*

Proof. We write $V = \mathcal{V}(\mathcal{X}(Y_\alpha))$. Let T be the search tree generated by the algorithm. First, the height of T is $2\alpha + 1$. Let ζ be the leftmost terminal node of T. WLOG, we assume that ζ induces the canonical labeling.[5] After ζ is generated, the first $\alpha + 1$ terminal nodes generated (excluding ζ) induce automorphisms defined by twisting the cycles $C_1, C_2, \ldots, C_{\alpha+1}$ (cf. Proposition 5.2). Now, no automorphism twists bridges $f_1, f_2, \ldots$, or f_α; thus, no other terminal nodes induce automorphisms during the search.

After generating the first $\alpha+2$ terminal nodes, no branching is performed at any levels below level α because the generators of $\mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta_B)$ are already found and so are its orbits (cf. §3.2.1).

We now prove that T contains the complete binary subtree of height α. Suppose the algorithm is visiting a node φ_k at some level $k < \alpha$ after ζ is generated (i.e., after all the generators of $\mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta_B)$ are found). Let $\zeta_j = \mathrm{gca}(\zeta, \varphi_k)$ at some level $j \leq k$. Let $\psi \neq \zeta$ be a terminal descendant of φ_k. Then $g \in \mathrm{Sym}(V)$ such that $\zeta^g = \psi$ must twist the bridge f_j, so $g \notin \mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta_B)$. Thus, when a terminal node is generated, it fails to satisfy the pruning conditions of Cases (a) and (b) in §3.2.2 except for the first $\alpha + 2$ terminal nodes. Also, note that no orbit pruning is performed above level $\alpha + 1$ (cf. §3.2.1). So T contains the complete binary subtree of height α.

Consequently, for $(\mathcal{X}(Y_\alpha), \vartheta_B)$, the number of tree nodes in T generated by the algorithm is given by $t_2(\alpha) = 2^{\alpha+1} - 1 + 2^\alpha(\alpha+1) + \frac{1}{2}(\alpha^2 + 3\alpha + 2)$. Since $n = |V| = 20\alpha$, the number of tree nodes in n is given by $\mathcal{T}(n) = t_2(\frac{n}{20}) = \Omega(c^n)$, where $c = \sqrt[20]{2} > 1$. $\square$

7. Hashing with a graph invariant

7.1. Graph invariants

While automorphism pruning reduces the size of each equivalence class of terminal colorings of the search tree to a manageable size, it does not reduce the number of classes. To reduce the number of such classes, McKay defines *graph invariants* (called *indicator functions* in [15], [16]) to hash partial information of vertex labelings.

Definition. A function $f : \mathcal{K}(V) \times \mathcal{C}(V) \to \mathbf{N}$ is a *graph invariant* if, for all $X \in \mathcal{K}(V)$, $\varphi \in \mathcal{C}(V)$, and $g \in \mathrm{Sym}(V)$, we have

$$f(X^g, \varphi^g) = f(X, \varphi).$$

Let $\varphi \in \mathcal{C}(V)$. For the sequence of the color-classes $\mathcal{W}_\varphi = (W_1[\varphi], W_2[\varphi], \ldots, W_k[\varphi])$, we write $|\mathcal{W}_\varphi| = k$ for the number of the color-classes and $\|\mathcal{W}_\varphi\| = (|W_1[\varphi]|, |W_2[\varphi]|, \ldots, |W_k[\varphi]|)$ for the sequence of the sizes of the color-classes. The graph invariant in *nauty* is defined according to some

[5] Since no automorphism can twist bridges, having the canonical node (i.e., a terminal node that induces the canonical labeling) elsewhere in the search tree does not make tree pruning more efficient.

canonical information obtained during color refinement steps with some hash function h as follows.

Definition. For $\varphi \in \mathcal{C}(V)$, suppose the stable coloring $\overline{\varphi}$ is obtained by the sequence of refinements $\varphi, \varphi', \ldots, \varphi^{(r)} = \overline{\varphi}$. We then define

$$\Lambda(X, \varphi) = h(\|\mathcal{W}_{\varphi}\|, \|\mathcal{W}_{\varphi'}\|, \ldots, \|\mathcal{W}_{\varphi^{(r)}}\|),$$

where h is some hash function.

7.2. Canonical forms with a graph invariant

For $X = (V, E)$, let $\varphi \in \mathcal{C}(V)$ and $S = (v_1, v_2, \ldots, v_s)$ be a sequence of vertices. Consider the sequence of stabilizers $\varphi_{v_1}, \varphi_{v_1 v_2}, \ldots, \varphi_{v_1 v_2 \cdots v_s} = \varphi_S$. We define

$$\tilde{\Lambda}(X, \varphi, S) = (\Lambda(X, \varphi), \Lambda(X, \operatorname{ind}(\varphi, v_1)), \ldots, \Lambda(X, \operatorname{ind}(\varphi_{v_1 v_2 \cdots v_{s-1}}, v_s))).$$

Remark. $\tilde{\Lambda}(X^g, \varphi^g, S^g) = \tilde{\Lambda}(X, \varphi, S)$ for all $X \in \mathcal{K}(V)$, $\varphi \in \mathcal{C}(V)$, sequences of vertices S, and $g \in \operatorname{Sym}(V)$.

For a graph (X, φ), let $L = \operatorname{Leaf}(X, \varphi)$ be the set of all leaves of the search tree generated by the depth-first vertex classification algorithm. Terminal colorings φ_{S_1} and φ_{S_2} cannot be equivalent unless $\tilde{\Lambda}(X, \varphi, S_1) = \tilde{\Lambda}(X, \varphi, S_2)$. Further efficiency can be achieved by ordering the vectors $\tilde{\Lambda}(X, \varphi, S)$ lexicographically and redefining $\operatorname{CF}(X)$ to be $\max\{\varphi_S(X) \mid \varphi_S \in L$ and $\tilde{\Lambda}(X, \varphi, S) = \Lambda^*\}$, where $\Lambda^* = \max\{\tilde{\Lambda}(X, \varphi, S) \mid \varphi_S \in L\}$. By this means, McKay's algorithm nauty eliminates sections of the search tree that cannot contain either new automorphisms or $\operatorname{CF}(X)$.

7.3. Local equivalence

For stabilizers $\varphi_{S_1}, \varphi_{S_2}$, what are conditions on φ_{S_1} and φ_{S_2} in order to get the equality $\tilde{\Lambda}(X, \varphi, S_1) = \tilde{\Lambda}(X, \varphi, S_2)$ independent of h? To answer this question, we introduce our notion of local equivalence of colorings.

A *semiregular bipartite graph* is a bipartite graph with a given chromatic partition $V = V_1 \cup V_2$ such that all vertices in the same color-class have equal degrees. For V_1, V_2, disjoint subsets of the vertex set of the graph $X = (V, E)$, let $\mathcal{B}(V_1, V_2)$ denote the bipartite subgraph of X induced between V_1 and V_2. If $|V_1| = |V_2| = 1$, then the bipartite subgraph $\mathcal{B}(V_1, V_2)$ is called *trivial*. We first characterize stable colorings as follows (cf. Proposition 2.1 in [1]).

Observation 7.1. *For $X = (V, E)$, let $\varphi \in \mathcal{C}(V)$. The coloring φ is stable if and only if every subgraph induced by $W_i[\varphi]$ is regular and every induced bipartite subgraph $\mathcal{B}(W_i[\varphi], W_j[\varphi])$ is semiregular for $1 \leq i, j \leq |\mathcal{W}_{\varphi}|$.* $\square$

Now, we define our notion of local equivalence. For $X = (V, E)$, let $\varphi, \psi \in \mathcal{C}(V)$. The colorings φ and ψ are *locally equivalent*, denoted by $\varphi \sim_\ell \psi$, if every non-trivial bipartite subgraph $\mathcal{B}(W_i[\varphi], W_j[\varphi])$ is color-preserving isomorphic to the corresponding subgraph $\mathcal{B}(W_i[\psi], W_j[\psi])$ for $1 \le i, j \le |\mathcal{W}_\varphi|$.

Let $S_1 = (v_1, v_2, \ldots, v_s)$ and $S_2 = (w_1, w_2, \ldots, w_s)$ be sequences of vertices of the same length, where each $v_i, w_i \in V, 1 \le i \le s$. Let $K_1 = (v_1, v_2, \ldots, v_k)$ and $K_2 = (w_1, w_2, \ldots, w_k)$ be the subsequences of length $k \le s$. Now, for $\varphi \in \mathcal{C}(V)$, define $\Phi_{1,k} = \operatorname{ind}(\varphi_{K_1}, v_{k+1})$ and $\Phi_{2,k} = \operatorname{ind}(\varphi_{K_2}, w_{k+1})$. If $\Phi_{1,k}^{(i)} \sim_\ell \Phi_{2,k}^{(i)}$ for all i and k, then $\tilde{\Lambda}(X, \varphi, S_1) = \tilde{\Lambda}(X, \varphi, S_2)$. That is, having the local equivalence of refinements at all levels is a sufficient condition to have the values of $\tilde{\Lambda}$ equal.

8. Still exponential

Let Y be a 3-regular graph. For $(\mathcal{X}(Y), \vartheta)$, let T be the search tree generated by the depth-first vertex refinement algorithm. Then all colorings in T at the same level are locally equivalent as follows.

Proposition 8.1. *For $(\mathcal{X}(Y), \vartheta)$, let $\varphi = \vartheta_{S_1}$ and $\psi = \vartheta_{S_2}$ be locally equivalent stabilizers. Let $\Phi = \operatorname{ind}(\varphi, v)$ and $\Psi = \operatorname{ind}(\psi, w)$, where v and w are in the target color-classes of φ and ψ, respectively. Then $\Phi^{(r)}$ and $\Psi^{(r)}$ are locally equivalent for $0 \le r < n$.*

Proof. Let $W_t[\varphi]$ and $W_t[\psi]$ be the target color-classes of φ and ψ, respectively. First, we show $\Phi \sim_\ell \Psi$. Suppose $W_t[\Phi] = \{v, v'\}$ and $W_t[\Psi] = \{w, w'\}$. For all non-trivial color-classes $W_j[\Phi]$ that are adjacent to v, we have $\mathcal{B}(\{v\}, W_j[\Phi]) \cong \mathcal{B}(\{w\}, W_j[\Psi])$ and $\mathcal{B}(\{v'\}, W_j[\Phi]) \cong \mathcal{B}(\{w'\}, W_j[\Psi])$. Consequently, $\Phi \sim_\ell \Psi$.

Suppose $\Phi^{(r-1)} \sim_\ell \Psi^{(r-1)}$ for some $r \ge 1$. Here, it suffices to show the isomorphisms of all pairs of non-trivial bipartite subgraphs $\mathcal{B}(W_i[\Phi^{(r)}], W_j[\Phi^{(r)}])$ and $\mathcal{B}(W_i[\Psi^{(r)}], W_j[\Psi^{(r)}])$, where color-classes $W_i[\Phi^{(r)}]$ and $W_i[\Psi^{(r)}]$ are properly contained in color-classes of $\Phi^{(r-1)}$ and $\Psi^{(r-1)}$, respectively.

First, fix $W_i[\Phi^{(r)}] \subsetneq W_k[\Phi^{(r-1)}]$. Suppose $|W_k[\Phi^{(r-1)}]| = 2$, then $|W_k[\Psi^{(r-1)}]| = 2$. Let $x \in W_k[\Phi^{(r-1)}]$ and $\{y, y'\} = W_k[\Psi^{(r-1)}]$. Let $W_j[\Phi^{(r)}]$ be a non-trivial color-class adjacent to x. Since $|W_j[\Phi^{(r)}]| = |W_j[\Psi^{(r)}]| = 2$ or 4, we have

$$\begin{aligned} \mathcal{B}(\{x\}, W_j[\Phi^{(r)}]) &\cong \mathcal{B}(\{y\}, W_j[\Psi^{(r)}]) \\ &\text{and} \\ \mathcal{B}(\{x\}, W_j[\Phi^{(r)}]) &\cong \mathcal{B}(\{y'\}, W_j[\Psi^{(r)}]). \end{aligned}$$

Suppose $|W_k[\Phi^{(r-1)}]| = 4$. Then $W_k[\Phi^{(r-1)}] = \mathcal{M}(v)$ for some $v \in \mathcal{V}(\mathcal{X}(Y))$. That is, $|W_i[\Phi^{(r)}]| = 2$ and $W_k[\Phi^{(r-1)}] = W_k[\Psi^{(r-1)}]$. Let $W_{j_1}[\Phi^{(r)}]$ and $W_{j_2}[\Phi^{(r)}]$ be the color-classes that form two outer pairs of $\mathcal{X}(v)$.

Here, we observe that $W_{j_1}[\Phi^{(r)}] = W_{j_1}[\Psi^{(r)}]$ and $W_{j_2}[\Phi^{(r)}] = W_{j_2}[\Psi^{(r)}]$. The color-classes of $\Phi^{(r)}$ adjacent to $W_i[\Phi^{(r)}]$ are $W_{j_1}[\Phi^{(r)}], W_{j_2}[\Phi^{(r)}]$, and a singleton $W_{j_3}[\Phi^{(r)}]$ for some j_3. We then have the three required pairs of color-preserving isomorphic semiregular bipartite subgraphs. □

We conclude this section with two corollaries.

Corollary 8.2. *Given $(\mathcal{X}(Y), \vartheta)$, consider applying the depth-first vertex classification algorithm. Let ϑ_{S_1} and ϑ_{S_2} be stabilizers at the same level in the search tree. Then $\tilde{\Lambda}(\mathcal{X}(Y), \vartheta, S_1) = \tilde{\Lambda}(\mathcal{X}(Y), \vartheta, S_2)$ for any hash function h.* □

Corollary 8.3. *Given $(\mathcal{X}(Y_\alpha), \vartheta_B)$, where $n = |\mathcal{V}(\mathcal{X}(Y_\alpha))|$, the algorithm* `nauty` *generates $\Omega(c^n)$ nodes in the search tree to compute $\mathrm{CF}(\mathcal{X}(Y_\alpha), \vartheta_B)$ and $\mathrm{Aut}(\mathcal{X}(Y_\alpha), \vartheta_B)$ for some fixed constant $c > 1$.* □

9. Extending the distance

For a coloring $\varphi \in \mathcal{C}(V)$ and a vertex $v \in V$, the standard stable refinement $\overline{\varphi}$ is based on a lexicographic ordering of the tuple $D(v) = (\varphi(v), d_1(v), \ldots, d_n(v))$. We can extend this method by counting the numbers of neighbors of distance δ in each color-class. For vertices $v, w \in V = \mathcal{V}(X)$, the number of edges traversed in the shortest path joining v and w is called the *distance* in X between v and w and is denoted by $\partial(v, w)$. For $v \in V$ and $U \subseteq V$, we define $N_\delta(v, U) = \{w \in U \mid \partial(v, w) = \delta\}$ and $n_\delta(v, U) = |N_\delta(v, U)|$. For a coloring φ, where $\mathcal{W}_\varphi = (W_1, W_2, \ldots, W_k)$, and a vertex $v \in V$, consider the tuple

$$D_\delta(v) = (\varphi(v), n_\delta(v, W_1), n_\delta(v, W_2), \ldots, n_\delta(v, W_k)).$$

We sort these new colors lexicographically and define $\varphi'(v)$ to be the ordering number of the new color-class to which v belongs. We keep refining the coloring until at some level $\varphi^{(r)} = \varphi^{(r+1)}$ for some $r < n$. We define $\mathcal{S}_\delta(\varphi) = \varphi^{(r)}$, and we call $\mathcal{S}_\delta(\varphi)$ the *δ-stable refinement* of φ. Note here that $\overline{\varphi} = \mathcal{S}_1(\varphi)$. Furthermore, we define

$$\overline{\mathcal{S}}_\delta(\varphi) = \begin{cases} \mathcal{S}_\delta(\varphi) & \text{if } \delta = 1, \\ \mathcal{S}_\delta(\overline{\mathcal{S}}_{\delta-1}(\varphi)) & \text{if } \delta > 1. \end{cases}$$

We call $\overline{\mathcal{S}}_\delta(\varphi)$ the *strongly δ-stable refinement* of φ. If $\varphi = \overline{\mathcal{S}}_\delta(\varphi)$, then we call φ is *strongly δ-stable*.

In theory, for a fixed $\delta > 0$, both δ-stable and strongly δ-stable refinements can be computed in polynomial time. However, for practical purposes, there is a weaker color-refinement heuristic.[6] If φ is strongly δ-stable, we define $\mathrm{dist}_\delta(\varphi) = \varphi$. Otherwise, for some $\varepsilon \leq \delta$, φ is strongly $(\varepsilon - 1)$-stable but not ε-stable. Then there exists a non-trivial color-class $W_i[\varphi]$ such that $D_\varepsilon(v_1) \neq$

[6] This is defined in *nauty* as an optional vertex-refinement heuristic.

$D_\varepsilon(v_2)$ for some $v_1, v_2 \in W_i[\varphi]$.[7] Now, let φ' be the new coloring defined by lexicographically ordering the tuples $D_\varepsilon(v)$ for $v \in V$. Here, the color-class $W_i[\varphi]$ splits as $W_i[\varphi] = W_{i_1}[\varphi'] \cup W_{i_2}[\varphi'] \cup \cdots \cup W_{i_\ell}[\varphi']$ for some $\ell > 1$. We then define $\mathrm{dist}_\delta(\varphi)$ by the sequence of color-classes

$$\begin{aligned}\mathcal{W}_{\mathrm{dist}_\delta(\varphi)} = (&W_1[\varphi], \ldots, W_{i-1}[\varphi], W_{i_1}[\varphi'], \ldots, W_{i_\ell}[\varphi'], W_{i+1}[\varphi],\\ &\ldots, W_k[\varphi]).\end{aligned}$$

We say $\mathrm{dist}_\delta(\varphi)$ *splits* the color-class $W_i[\varphi]$, and we call $\mathrm{dist}_\delta(\varphi)$ the *δ-distance refinement* of φ.

For the family of graphs $(\mathcal{X}(Y_\alpha), \vartheta_B)$, applying $\mathrm{dist}_n(\varphi)$ at each level improves the performance of the algorithm only by a constant factor (i.e., it still remains exponential) for the following reasons.

Proposition 9.1. *Let Y be a 3-regular graph. Given $(\mathcal{X}(Y), \vartheta)$, let φ be a stable refinement of ϑ. For all $\delta > 0$, the refinement $\mathrm{dist}_\delta(\varphi)$ does not split any color-class of φ of size 2.*

Proof. Suppose φ is defined by $\mathcal{W}_\varphi = (W_1, W_2, \ldots, W_k)$. Let W_i be a color-class of size 2 and $v, w \in W_i$. Then

(i) for $\delta = 1, 2, 3, 4$, and 5, we have $n_\delta(v, W_j) = n_\delta(w, W_j)$; and
(ii) for $\delta \geq 6$, we have $N_\delta(v, W_j) = N_\delta(w, W_j)$

for all $W_j, 1 \leq j \leq k$. □

Corollary 9.2. *For $(\mathcal{X}(Y_\alpha), \vartheta)$, let $\varphi = \vartheta_{S_1}$ and $\psi = \vartheta_{S_2}$ be stabilizers. If φ and ψ are locally equivalent, then for all $1 \leq \delta \leq n$,*

(i) *$\mathrm{dist}_\delta(\varphi)$ and $\mathrm{dist}_\delta(\psi)$ are locally equivalent, and*
(ii) *the stable refinements of $\mathrm{dist}_\delta(\varphi)$ and $\mathrm{dist}_\delta(\psi)$ are locally equivalent.* □

10. Experiments with *nauty*

We first conducted experiments with the *nauty* system using its default parameters to compute the canonical form of $(\mathcal{X}(Y_\alpha), \vartheta)$. We refer to [17] for details on the implementation of the algorithm. We used the program *dreadnaut* [17] for *nauty* (version 1.7) running on a Sun SPARCstation-1 computer under the operating system SunOS (version 4.1.3). The experiments were conducted for the following three types of colorings on $(\mathcal{X}(Y_\alpha), \vartheta)$.

(i) $(\mathcal{X}(Y_\alpha), \vartheta_A)$, where ϑ_A is defined in §5.1. We call this Type A.
(ii) $(\mathcal{X}(Y_\alpha), \vartheta_B)$, where ϑ_B is defined in §6.1. We call this Type B.
(iii) $(\mathcal{X}(Y_\alpha), \vartheta_C)$, where ϑ_C is a coloring defined by randomly permuting all the color-classes. We call this Type C.

For Type A, we tested for $1 \leq \alpha \leq 31$, and for Type B, we tested for $1 \leq \alpha \leq 23$. For Type C, we took the average of 16 randomly generated ϑ_C for each size, where $1 \leq \alpha \leq 31$. We collected the numbers of tree nodes generated and the CPU execution time measured by *dreadnaut*. Table 1 shows the number of

Table 1. The number of tree nodes generated and the CPU time for $(\mathcal{X}(Y_\alpha), \vartheta_B)$, where $1 \leq \alpha \leq 23$.

Graph	$\lvert\mathcal{V}(\mathcal{X}(Y_\alpha))\rvert$	Search tree size	CPU time (seconds)
$(\mathcal{X}(Y_1), \vartheta_B)$	20	10	0.01
$(\mathcal{X}(Y_2), \vartheta_B)$	40	25	0.03
$(\mathcal{X}(Y_3), \vartheta_B)$	60	57	0.07
$(\mathcal{X}(Y_4), \vartheta_B)$	80	126	0.21
$(\mathcal{X}(Y_5), \vartheta_B)$	100	276	0.50
$(\mathcal{X}(Y_6), \vartheta_B)$	120	603	1.25
$(\mathcal{X}(Y_7), \vartheta_B)$	140	$1,315$	3.18
$(\mathcal{X}(Y_8), \vartheta_B)$	160	$2,860$	8.04
$(\mathcal{X}(Y_9), \vartheta_B)$	180	$6,198$	20.23
$(\mathcal{X}(Y_{10}), \vartheta_B)$	200	$13,377$	51.91
$(\mathcal{X}(Y_{11}), \vartheta_B)$	220	$28,749$	139.43
$(\mathcal{X}(Y_{12}), \vartheta_B)$	240	$61,530$	306.55
$(\mathcal{X}(Y_{13}), \vartheta_B)$	260	$131,176$	729.66
$(\mathcal{X}(Y_{14}), \vartheta_B)$	280	$278,647$	$1,499.58$
$(\mathcal{X}(Y_{15}), \vartheta_B)$	300	$589,959$	$3,395.27$
$(\mathcal{X}(Y_{16}), \vartheta_B)$	320	$1,245,336$	$7,737.59$
$(\mathcal{X}(Y_{17}), \vartheta_B)$	340	$2,621,610$	$16,379.63$
$(\mathcal{X}(Y_{18}), \vartheta_B)$	360	$5,505,213$	$20,261.31$
$(\mathcal{X}(Y_{19}), \vartheta_B)$	380	$11,534,545$	$44,285.77$
$(\mathcal{X}(Y_{20}), \vartheta_B)$	400	$24,117,478$	$177,259.39$
$(\mathcal{X}(Y_{21}), \vartheta_B)$	420	$50,331,900$	$211,538.75$
$(\mathcal{X}(Y_{22}), \vartheta_B)$	440	$104,857,875$	$460,035.89$
$(\mathcal{X}(Y_{23}), \vartheta_B)$	460	$218,104,107$	$994,427.10$

tree nodes generated and the CPU time of Type B for $1 \leq \alpha \leq 23$ obtained by *nauty*. Figure 4 shows the numbers of tree nodes generated for Type A, Type B, and the average of 16 randomly generated Type C's for each size. Here, the curves in Figure 4 appear to show that the number of tree nodes generated for Type A has a polynomial upper bound, and for Type B, it has an exponential lower bound. For the average of Type C's, its behavior is not quite clear from our experiments, which leaves some questions in studying average time complexity of Type C's.

We also conducted experiments using *nauty*'s optional vertex-refinement heuristics (called *vertex invariants* in [17]). Except for $\mathrm{dist}_n(\varphi)$, which showed some improvement by a constant factor (cf. §9), no other heuristics helped the performance on our exponential examples.

11. Remarks

11.1. Other classes of graphs

In general, if we ignore pruning by hashing partial information of vertex labelings, it is not so difficult to construct examples that can force the algorithm to

[7] We take the minimum of all such i's determined uniquely by some ordering of color-classes.

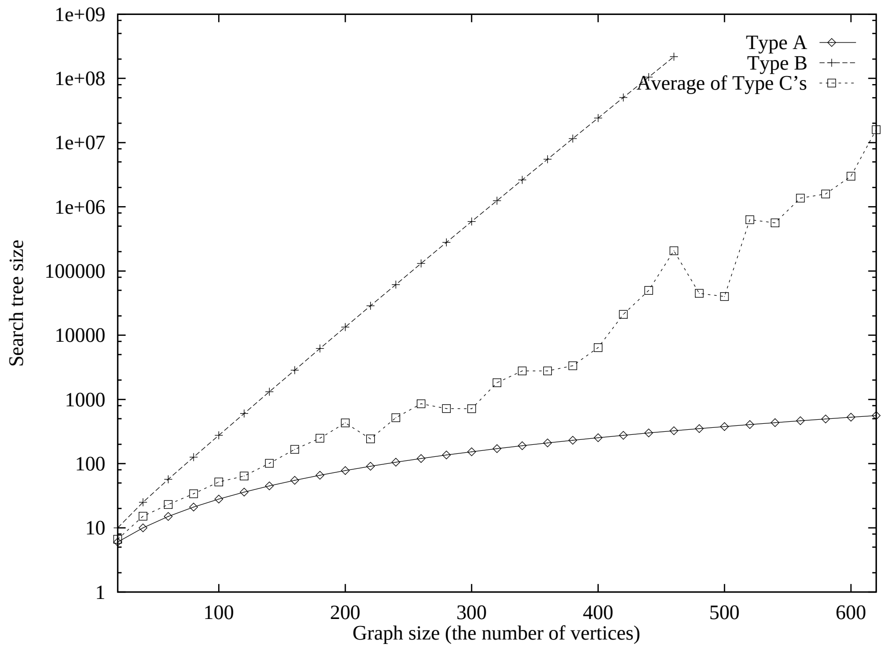

Figure 4. The number of tree nodes generated

exponential time. In fact, a referee pointed out that the family of graphs consisting of a disjoint union of triangles and squares will do the job. For such graphs, the naive vertex classification method (described in §2.3) cannot detect the difference between triangles and squares; and thus, we are forced to backtrack. The size of their search trees becomes exponential if we arrange vertex labelings so that triangles are matched with squares. It still remains exponential even if each pair of triangles and squares are colored uniquely with initial color-class size 7.

However, we note that if we extend the distance of neighbors to $\delta = 2$ during the refinement step (as described in §9), the naive vertex classification method computes the orbit partitions of such graphs in one refinement step, and their canonical forms can be computed in polynomial time. The referee's idea of triangles and squares can be pushed further by taking a sequence of polygons at the cost of increasing color-class size. Consider an n-vertex graph consisting of a sequence of m-gons and $(m+1)$-gons, where $m = \Theta(\sqrt{n})$, with vertex labelings arranged as in the triangle-square example. With δ being a fixed constant, if we ignore pruning by hashing, such an example gives a search tree of size $\Omega(c^{\sqrt{n}})$ for the method described in §9. Here, note that its color-class size is unbounded; in particular, it has size $\Omega(\sqrt{n})$.

Another example was suggested by McKay [18]. He pointed out that the family of graphs consisting of a disjoint union of strongly regular graphs, non-isomorphic but having the same parameters, would be a good candidate to force his algorithm to exponential time. But such an example also requires increasing color-class size.

11.2. On-going work

McKay's algorithm is based on coloring the vertex set V into invariant color-classes according to the number of neighbors in each color-class. This method can be extended by coloring V according to some invariant information obtained by partitioning the set V^d of ordered d-tuples in an analogous way (i.e., by the d-$\dim$ W-L method). This is a generalized, stronger version of the extended-distance algorithm described in §9. This algorithm not only searches out up to distance d but also considers all possible colorings of d-tuples. Since the present work was completed, by extending our results, we have constructed a family of connected 3-regular graphs of color-class size 4 that forces $d = \Omega(\sqrt{n})$ in order for the extended algorithm to work. That is, with d being a fixed constant, this example yields search trees of size $\Omega(c^{\sqrt{n}})$ for some fixed constant $c > 1$. This result is stronger than what is stated in §9 in the sense that the algorithm we consider employs the more generalized d-$\dim$ W-L method. This result will appear elsewhere.

Acknowledgements

It is a pleasure to express many thanks to my advisor Professor Eugene M. Luks for suggesting this problem to me and for helpful guidance, suggestions, and encouragement. I would also like to acknowledge the referees for their effort, interest, and suggestions.

References

1. L. BABAI, *Moderately exponential bound for graph isomorphism*, Fundamentals of Computation Theory, Proceedings of the 1981 International FCT-Conference, Szeged, Aug. 24–28, 1981 (F. Gécseg, ed.), Lecture Notes in Comput. Sci., vol. 117, Springer, New York, 1981, pp. 34–50.
2. ———, *Automorphism groups, isomorphism, reconstruction*, Handbook of Combinatorics, vol. 2 (L. Lovász, R. L. Graham, and M. Grötschel, eds.), Elsevier Science B.V., Amsterdam, 1995, pp. 1447–1540.
3. L. BABAI and E. M. LUKS, *Canonical labeling of graphs*, Proceedings of the Fifteenth Annual ACM Symposium on the Theory of Computing, Boston, Apr. 25–27, 1983, ACM, New York, 1983, pp. 171–183.
4. J. CAI, M. FÜRER, and N. IMMERMAN, *An optimal lower bound on the number of variables for graph identification*, 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, N.C., Oct. 30–Nov. 1, 1989, IEEE Computer Soc., Los Alamitos, Calif., 1989, pp. 612–617.
5. ———, *An optimal lower bound on the number of variables for graph identification*, Combinatorica **12** (1992), 389–410.
6. J. D. DIXON and B. MORTIMER, *Permutation groups*, Graduate Texts in Math., vol. 163, Springer, New York, 1996.
7. M. FÜRER, *A counterexample in graph isomorphism testing*, Tech. Rep. CS-87-36, Department of Computer Science, The Pennsylvania State University, University Park, Penna., 1987.

8. M. FÜRER, W. SCHNYDER, and E. SPECKER, *Normal forms for trivalent graphs and graphs of bounded valence*, Proceedings of the Fifteenth Annual ACM Symposium on the Theory of Computing, Boston, Apr. 25–27, 1983, ACM, New York, 1983, pp. 161–170.
9. M. HALL, Jr., *The theory of groups*, 2nd ed., Chelsea, New York, 1976.
10. Z. HEDRLÍN and A. PULTR, *On full embeddings of categories of algebras*, Illinois J. Math. **10** (1966), 392–406.
11. N. IMMERMAN and E. S. LANDER, *Describing graphs: a first-order approach to graph canonization*, Complexity Theory Retrospective (A. Selman, ed.), Springer, New York, 1990, pp. 59–81.
12. P. KLINGSBERG and E. M. LUKS, *Succinct certificates for a class of graphs*, manuscript, 1981.
13. E. M. LUKS, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, J. Comput. System Sci. **25** (1982), 42–65.
14. R. MATHON, *Sample graphs for isomorphism testing*, Congr. Numer. **21** (1978), 499–517.
15. B. D. MCKAY, *Computing automorphisms and canonical labellings of graphs*, Combinatorial Mathematics, Proceedings of the International Conference on Combinatorial Theory, Canberra, Aug. 16–27, 1977 (D. A. Holton and J. Seberry, eds.), Lecture Notes in Math., vol. 686, Springer, New York, 1978, pp. 223–232.
16. ———, *Practical graph isomorphism*, Congr. Numer. **30** (1981), 45–87.
17. ———, *nauty user's guide*, version 1.5, Tech. Rep. TR-CS-90-02, Computer Science Department, Australian National University, Canberra, 1990.
18. ———, Personal communication with L. Babai, 1995.
19. G. L. MILLER, *Graph isomorphism, general remarks*, J. Comput. System Sci. **18** (1979), 128–142.
20. R. C. READ and D. G. CORNEIL, *The graph isomorphism disease*, J. Graph Theory **1** (1977), 339–363.
21. B. WEISFEILER, *On construction and identification of graphs*, Lecture Notes in Math., vol. 558, Springer, New York, 1976.
22. H. WIELANDT, *Finite permutation groups*, Academic Press, New York, 1964.
23. V. N. ZEMLYACHENKO, N. M. KORNEENKO, and R. I. TYSHKEVICH, *Graph isomorphism problem*, Zap. Nauchn. Sem. Leningradskogo Otdel. Mat. Inst. Steklov. (LOMI) **118** (1982), 83–158 (Russian); English transl., J. Soviet Math. **29** (1985), 1426–1481.