Lecture notes: Isomorphism testing on cubic graphs

Jiří Fiala, Veronika Slívová and Martin Töpfer

June 14, 2025

The aim of these notes is to survey a part of the Luks's paper from 1982 [3] for educational purposes.

Theorem 1. *The existence of an isomorphisms between cubic graphs can be decided in polynomial time.*

Overview:

1. Given two cubic graphs Y, Y' and create several new graphs X such that Y is isomorphic to Y' if and only if at least one of these X has an automorphism which switches one specified edge $e \in E_X$.

2. To describe automorphisms of X we first find an automorphism on a subgraph of X induced by edges close to e. Then we extend the automorphisms to further distance from e step by step.

3. The extension of automorphisms is reducible to a color automorphism problem. For a colored set A and a group G of permutations on A we want to find generators of automorphisms on G which preserve colors.

4. We split G to two subgroups and/or A to smaller G-orbits so that for each reduced problem we can solve in polynomial time. From the particular solutions we compute a solution for G-orbits and then we compose it to a solution for G.

1 Reduction of isomorphism to automorphism

Assume first that Y and Y' are isomorphic and ψ is such an isomorphism. Choose an arbitrary vertex v of Y and construct a graph X from the disjoint union of Y and Y' by joining v and $v' = \psi(v)$ with a new edge e.

The isomorphism ψ straightforwardly yields an automorphism φ of X that transposes the edge e. If Y and Y' are cubic and we aim to get a cubic X as well we may alter the above by choosing some edge f of Y, and subdivide it by a new vertex v as well as subdivide $\psi(f)$ by v', see Fig. 1.

On the other hand if the two graphs are not isomorphic, then there is no automorphism that transpose the edge e in any possible placement of the vertex v as the subdivision of an edge of Y'.

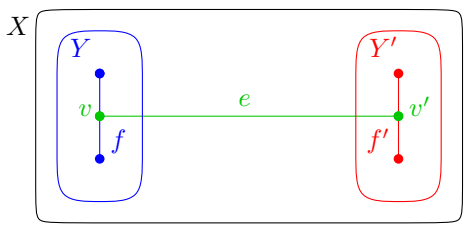
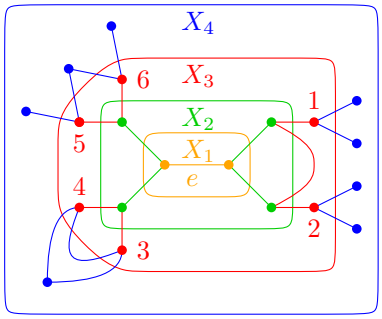


Figure 1: Reduction of graph isomorphism to graph automorphism

a)



b)

$B_e = \{\{3,4\}\}$

$B_1 = \{\{5\},\{6\},\{3,4\},\{5,6\}\}$

$B_2 = \{\{1\},\{2\}\}$

Figure 2: a) The nested subgraphs $X_1, X_2, \ldots$ of a graph yield a partition of the vertex set into layers $V_1, V_2, \ldots$. Notice that the edge $(1,2)$ between two vertices of the layer V_2 is contained in the subgraph X_3 but not in X_2. If we consider the subgraph X_2 then there is an automorphism which switches the edge e, but in X_3 no such automorphism exists.
b) The three color sets B_e, B_1 and B_2 used to determine $\pi_3(\mathrm{Aut}_e(X_4))$.

We reduce a single instance of the graph isomorphism problem to $O(|E_Y|)$ instances of the graph automorphism problem, which for a given graph returns a representation of its automorphism group. A pseudocode of such reduction is described by Algorithm 1.

Note that for bounded degree graphs we have linear dependence between $|V_Y|$ and $|E_Y|$, namely $|E_Y| = \frac{3|V_Y|}{2}$ for cubic graphs.

Algorithm 1: Reduction of graph isomorphism to graph automorphism

Input: Graphs Y and Y'
Query: Is Y isomorphic to Y'?

```
1 begin
2     choose an arbitrary edge f ∈ Y;
3     for each edge f' ∈ Y' do
4         X := Y ∪ Y';
5         subdivide in X the edges f, f' with new vertices v, v';
6         add to X a new edge e between vertices v, v';
7         if there is an automorphism φ of X s.t. φ(v) = v' and φ(v') = v then
8             return "Y and Y' are isomorphic"
9     return "Y and Y' are not isomorphic"
```

2 From automorphism extension to colored subgroups

2.1 Decomposition into layers

For a graph X and en edge $e \in E_X$, let the symbol X_k denotes the subgraph of X induced by the edges *of the paths of length at most k that contain e*. The vertex set V_X is then partitioned into *layers* $V_1, V_2, \ldots$ such that $V_1 = e$ and for $k \geq 2$ we set $V_k = V_{X_k} \setminus V_{X_{k-1}}$.

Notice that for $k \geq 2$, any edge between two vertices from the layer V_k is contained in the graph X_{k+1}, see Fig. 2 a).

Let the symbol $\mathrm{Aut}_e(X_k)$ be the set of automorphisms φ of X_k that fix e. These namely satisfy either $\varphi(u) = u$ & $\varphi(v) = v$ or $\varphi(u) = v$ & $\varphi(v) = u$, where u, v are the vertices of e.

Finally, let π_k be the projection of the automorphisms of $\mathrm{Aut}_e(X_{k+1})$ onto $\mathrm{Aut}_e(X_k)$, namely $\pi_k(\varphi)$ is the restriction of the automorphism φ to the subgraph X_k.

The set of generators of $\mathrm{Aut}_e(X_{k+1})$ is constructed as the union of the following sets, see Fig. 3:

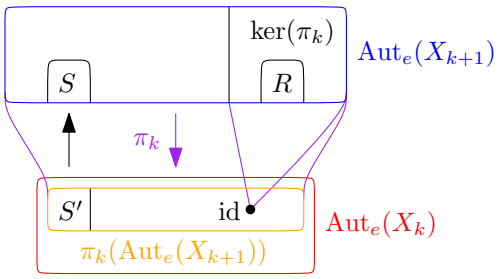


Figure 3: The two sets R and S generating $\mathrm{Aut}_e(X_{k+1})$

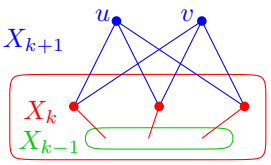


Figure 4: Two vertices u, v in the layer V_{k+1} are twins. The automorphism of X_{k+1} which switches them, while X_k is fixed, is an element of R.

- R containing the generators of the kernel of π_k — notice that these automorphisms only switch vertices from the layer V_{k+1},
- S obtained from the set S' of generators of $\pi_k(\mathrm{Aut}_e(X_{k+1}))$ such that for each $\psi' \in S'$ we insert into S any ψ such that $\pi_k(\psi) = \psi'$. In other words, as each generator of $\pi_k(\mathrm{Aut}_e(X_{k+1}))$ is only an automorphism of X_k, we insert into S any of its extensions onto X_{k+1}.

It holds that $\mathrm{Aut}_e(X_{k+1}) = \langle R \cup S \rangle$, as for every $\varphi \in \mathrm{Aut}_e(X_{k+1})$ we may project φ into $\mathrm{Aut}_e(X_k)$, express $\pi_k(\varphi)$ with respect to the generators of $\mathrm{Aut}_e(X_k)$, say $\pi_k(\varphi) = \psi'_1 \dots \psi'_t$. The corresponding expression with their extensions from S, namely $\psi_1 \dots \psi_t$, agrees with φ on X_k. The difference $\psi_1 \dots \psi_t \varphi^{-1}$ belongs to the kernel of π_k and hence it could be expressed in terms of R.

It is easy to construct the set R, while getting the set S is more complicated.

2.2 Construction of R

We say that vertices u, v from the layer V_{k+1} are *twins* if $N(u) \cap V_k = N(v) \cap V_k$, see Fig. 4.

Since the degree of each vertex is 3, each vertex in V_k has at most two neighbors (possible twins) in V_{k+1} as it must at least one neighbor in the subgraph X_k.

Therefore, the kernel of π_k is generated by the set of transposition of twins, i.e.

$$R = \{(u, v) \colon u, v \text{ are twins in } V_{k+1}\}$$

2.3 Construction of S

Denote by A the set V_{X_k} augmented by all subsets of V_k of size 2 or 3, i.e.: $A = V_{X_k} \cup \binom{V_k}{2} \cup \binom{V_k}{3}$.

Each automorphism $\varphi \in \mathrm{Aut}_e(X_k)$ corresponds uniquely to a permutation $\varphi' \in \mathrm{Sym}(A)$ of elements from A. Consequently the group $\mathrm{Aut}_e(X_k)$ corresponds to some group $G \subseteq \mathrm{Sym}(A)$. To capture the property that φ can be extended into an automorphism of X_{k+1} we introduce three subsets of A as follows, see Fig. 2 b) for an example:

- $B_e = \{\{u, v\} \colon u, v \in V_k \wedge (u, v) \in E(X_{k+1})\}$

  ... these are the new edges inside the layer V_k introduced in the graph X_{k+1},

- $B_1 = \{a \in A \colon \exists! w \in V_{k+1} \colon N(w) = a\}$

  ... for the tuples that form the neighborhood of one vertex from the new layer V_{k+1},

- $B_2 = \{a \in A \colon \exists! w, w' \in V_{k+1} \colon N(w) = N(w') = a\}$

  ... neighborhoods of twins from the ne layer.

The sets B_1 and B_2 are disjoint by the definition. Moreover, B_2 and B_e are disjoint too, due to maximum degree 3. Each element of A hence falls into one of the following five subsets viewed as colors:

$$B_2, \qquad B_1 \cap B_e, \qquad B_1 \setminus B_e, \qquad B_e \setminus B_1, \qquad A \setminus (B_e \cup B_1 \cup B_2).$$

Lemma 2. *A automorphism φ of X_k belongs to the image $\pi_k(\mathrm{Aut}_e(X_{k+1}))$ if and only if φ' belongs to the subgroup of G which respects the three sets B_e, B_1 and B_2.*

Proof. For the forward implication, if $a \in A$ and $\varphi'(a)$ have different colors then φ can not be an automorphism of X_{k+1}.

For the opposite implication consider an automorphism $\varphi \in \mathrm{Aut}_e(X_k)$ such that the corresponding φ' respects the three sets. For a vertex $u \in V_{k+1}$ we denote its neighborhood in the previous layer $N(u) \cap V_k$ by $a_u \in A$.

If $(u, v) \in B_e$ then $(\varphi(u), \varphi(v)) \in B_e$, so φ respects newly added edges between the vertices of the layer V_k.

If $a_u \in B_1$ then there exists exactly one u' such that $a_{u'} = \varphi'(a_u)$. We extend φ on u by setting $\varphi(u) = u'$.

If $a_u \in B_2$ then it has a unique twin $v \in V_{k+1}$ such that $a_v = a_u$. With help of φ' we identify a pair of twins u' and v' such that $a'_v = a'_u = \varphi'(a_u)$. We extend φ on u and v by setting $\varphi(u) = u'$ and $\varphi(v) = v'$. Though there are two possibilities, we choose any of them, since the other could be obtained by the transposition $(u, v) \in R$. □

So the construction of $\mathrm{Aut}_e(X_{k+1})$ now reduces to the problem of how to construct for given generators of $\mathrm{Aut}_e(X_k)$ the set of generators of its subgroup that respects the given coloring constraints.

3 Representations of permutation groups

The goal of this section is to develop methods that would allow us to represent possibly exponentially large permutation groups and their cosets by polynomially many elements, as well as be able to perform membership tests and unions.

We assume the permutation group $\mathrm{Sym}(A)$ acting on a set A with the composition operation defined for $\alpha, \beta \in \mathrm{Sym}(A)$ as follows: $\forall a \in A : (\alpha\beta)(a) = \beta(\alpha(a))$ where the left to right order of permutations on the composition corresponds to the order of application of them on A. We adopt the notation that Latin letters like a, b or a_i stand for the elements of the ground set A, while Greek minuscules represents group elements, i.e. the permutations.

For a group $G \subseteq \mathrm{Sym}(A)$ and an arbitrary order of the elements of $A = \{a_1, \dots a_n\}$, we denote by G_i the subgroup of G which fixes each of $a_1, \dots, a_i$.

Immediately, we get a chain of subgroups:

$$\{id\} = G_n = G_{n-1} \subseteq G_{n-2} \subseteq \ldots \subseteq G_2 \subseteq G_1 \subseteq G_0 = G$$

Definition. *Let G be a group, H be a subgroup and β be any element of G. Then $\beta H = \{\beta\alpha,\ \alpha \in H\}$ is the left* coset *of H in the group G.*

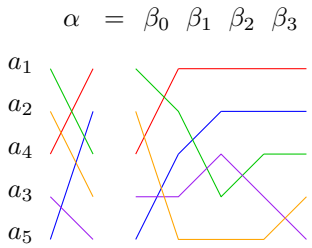


Figure 5: Composing α on $n = 5$ elements from coset representatives $\beta_0 \cdots \beta_3$.

We aim to represent G with sets C_i for $i \in \{0, \ldots, n-2\}$, where each such C_i contains representatives of left cosets of the subgroup G_{i+1} in the group G_i. In other words G_i is the union of left cosets of G_{i+1}, formally written as: $G_i = \bigcup_{\beta \in C_i} \beta G_{i+1}$. Immediately we get that $|G| = \prod_{i=0}^{n-2} |C_i|$.

Our goal is to represent any $\alpha \in G$ by a chain of permutations $\beta_0\beta_1 \cdots \beta_{n-2}$, where each β_i is a coset representative from C_i. As C_i contains only permutations in which the elements a_j for $j < i$ are fixed, the chain $\beta_0\beta_1 \cdots \beta_{n-2}$ can be viewed as a sequence of permutations where one-by-one β_i selects the correct *preimage* of a_{i+1} when $\beta_0, \ldots, \beta_{i-1}$ are already fixed, see Fig. 5.

For this purpose we use the following filtering procedure which for a (not necessarily complete set) of coset representatives either finds such a decomposition of a given α, or extends some of the sets so that it exists.

Algorithm 2: Filter(α)

Input: $\alpha \in \mathrm{Sym}(A)$, $G \subseteq \mathrm{Sym}(A)$ given by sets $C_0, \ldots, C_{n-2}$
Output: $\beta_0, \beta_1, \ldots, \beta_{n-2}$ so that $\beta_i \in C_i$ and $\alpha = \beta_0\beta_1 \cdots \beta_{n-2}$ if $\alpha \in G$; otherwise one C_i is in addition extended to generate $\langle G \cup \alpha \rangle$

```
1  begin
2      α_0 := α;
3      for i = 0 to n − 2 do
4          foreach β_i ∈ C_i do
5              if α_i^{-1}(a_{i+1}) = β_i^{-1}(a_{i+1}) then
6                  α_{i+1} := β_i^{-1} α_i;
7                  keep the current value of β_i and exit to the next round of the for loop
               // no coset representative of G_{i+1} was found
8          add α_i to C_i;
9          β_i := α_i;
10         α_{i+1} := id
11     return β_0, β_1, ..., β_{n-2}
```

Observe that the test $\alpha_i^{-1}(a_{i+1}) = \beta_i^{-1}(a_{i+1})$ is equivalent to $\beta_i^{-1}\alpha_i \in G_{i+1}$, which could be rephrased that α_i belongs to the coset of G_{i+1} in G_i represented by β_i. So if no β_i represents such coset, it is appropriate to add at line 8 the present value of α_i as a new representative. Then as every subgroup contains the identity all further findings of $\beta_i \in C_i$ succeed and no more coset representatives will be added.

From the assignments at line 6, namely $\alpha_1 = \beta_0^{-1}\alpha_0$, $\alpha_2 = \beta_1^{-1}\alpha_1, \ldots$ follows: $\alpha_0 = \beta_0\alpha_1$, $\alpha_1 = \beta_1\alpha_2, \ldots$ and in consequence also:

$$\alpha = \alpha_0 = \beta_0\alpha_1 = \beta_0\beta_1\alpha_2 = \cdots = \beta_0\beta_1 \cdots \beta_{n-2}$$

Moreover, for a representative β_i the only relevant information was the value of $\beta_i^{-1}(a_{i+1})$. By keeping only one representative for each coset, i.e. representatives that differ on $\beta_i^{-1}(a_{i+1})$, the size of

each C_i is at most $n - i$.

Claim 3. *The time complexity of Filter(α) is $O(n^2)$, where $n = |A|$.*

Proof. Assume first that each permutation α is represented as two arrays of images $[\alpha(a_1), \alpha(a_2), \dots]$, and $[\alpha^{-1}(a_1), \alpha^{-1}(a_2), \dots]$. When α has to be stored, the inverse α^{-1} can be computed in $O(n)$ time.

The **for** loop at line 3 is executed $O(n)$ times. Each its iteration requires only $O(n)$ time, because even though the **foreach** loop at line 4 is iterated $O(n)$ times, the **if** test at line 5 requires only $O(1)$ time, and if it succeeds, the assignment at line 6 will be performed only once as it quits the **foreach** loop.

Analogously, when no coset representative was found, the commands at lines 8–10 require $O(n)$ time, but are performed at most once in the whole execution of Filter(α). □

In the next step we show how to assure that each element of G has a unique expression with respect to the given coset representatives.

Definition. *We say that $C_0, \dots C_{n-2}$ are the sets of* strong generators *iff*

$$\forall \alpha \in G \; \exists! \beta_0, \beta_1, \dots, \beta_{n-2} \colon \beta_i \in C_i \;\wedge\; \alpha = \beta_0 \cdots \beta_{n-2}.$$

Lemma 4. *Let $C_0, \dots C_{n-2}$ be the sets of coset representatives for a chain of subgroups of G. If for each $i, j \in \{0, \dots n-2\}$ with $i \leq j$ and each $\sigma \in C_i$ and $\tau \in C_j$ holds that $\tau\sigma$ could be expressed as $\beta_0 \cdots \beta_{n-2}$ with $\beta_i \in C_i$ then any $\alpha \in \langle C_0 \cup \cdots \cup C_{n-2} \rangle$ has such expression as well.*

Proof. Assume $\alpha = \gamma_1 \cdots \gamma_m$ and for $j \in \{1, \dots, m\}$ choose i_j so that $\gamma_j \in C_{i_j}$.

We first determine the set $\{l \colon i_{l-1} \geq i_l\}$. If it is empty then the sequence $\gamma_1 \cdots \gamma_m$ (perhaps padded with the identity maps when necessary) is the desired expression of α.

The we select the subset with the smallest value of i_l, and finally, out of these let k be the maximum of this set, see Fig. 6. By the choice, this k has the following property:

- $i_{k-1} \geq i_k$
- $\forall l > k \colon i_l > i_k$
- $\forall i_l < i_k \colon i_{l-1} < i_l$

We apply the lemma assumption for $\sigma = \gamma_k$ and $\tau = \gamma_{k-1}$. Thus $\gamma_{k-1}\gamma_k = \tau\sigma = \beta_{i_k} \cdots \beta_{n-2}$ and if we substitute this term in the expression of α, we obtain $\alpha = \gamma_1 \cdots \gamma_{k-2}\beta_{i_k} \cdots \beta_{n-2}\gamma_{k+1} \cdots \gamma_m$.

Observe that after this replacement each i_l with $l \geq k$ is strictly greater than the former value of i_k. Therefore either the maximum of its subset attaining the minimal value i_l, i.e. the value of k has decreased, see Fig. 6 a) or the minimum of $\{l \colon i_{l-1} \geq i_l\}$ has increased, see Fig. 6 b).

As each index i_l is bounded by $n - 2$ and k is non-negative, after finitely many iterations of the above argument (in each round we express the same α but as a different sequence of γ's) we obtain a sequence where the set $\{l \colon i_{l-1} \geq i_l\}$ is empty as wanted. □

Now we show how a set S generating a group G can be transformed to a set of strong generators. The number of coset representatives might be larger than the size of S. For instance consider the cyclic subgroup of S_5 generated by the cyclic permutation $(2, 3, 4, 5, 1)$. This solely permutation generates the cyclic subgroup, but each of its five elements form a unique coset of the (only) subgroup $\{\mathrm{id}\}$, so $|C_1| = 5 > 1 = S = \{(2, 3, 4, 5, 1)\}$.

To obtain a set of strong generators we shall not only filter the generators of the subgroup to get coset representatives but also all their possible compositions of to assure that assumptions of Lemma 4 are satisfied. A naïve approach is summarized in the Algorithm 3.

The correctness of the algorithms follows from the fact that all elements of S were filtered, so $G = \langle C_0 \cup \cdots \cup C_{n-2} \rangle$ and by Lemma 4.

Claim 5. *The time complexity of Strong generators is $O(|S|n^2 + n^7)$, where $n = |A|$.*

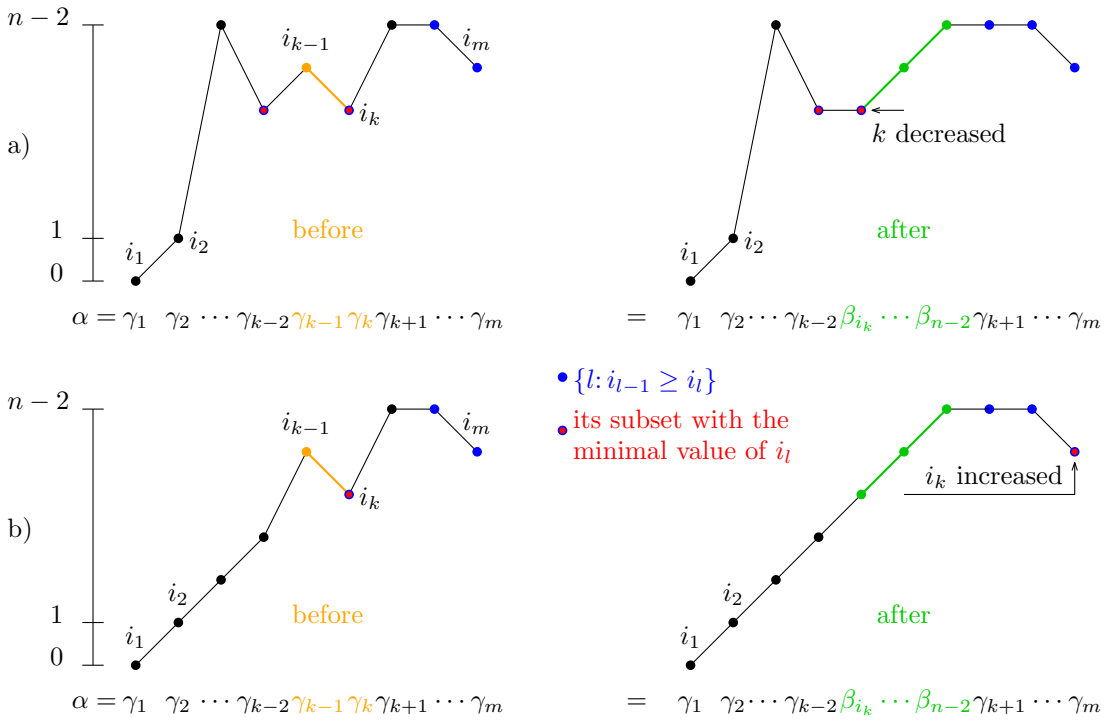
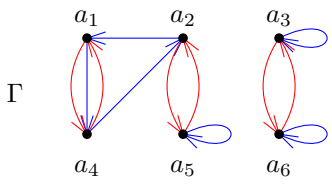


Figure 6: Example of the choice of k and the corresponding i_k.
a) The replacement of $\gamma_{k-1}\gamma_k$ with $\beta_{i_k} \cdots \beta_{n-2}$ decreased k. b) The replacement increased i_k.

Algorithm 3: Strong generators (S)

Input: A generating set S of a group G
Output: Sets of strong generators $C_0, \ldots, C_{n-2}$ for G

```
1  begin
2      for i := 0 to n − 2 do C_i := {id};
3      foreach α ∈ S do Filter(α);
4      for i := 0 to n − 2 do
5          repeat
6              foreach σ ∈ C_i do
7                  foreach τ ∈ C_j, j ≥ i do
8                      Filter(τσ)
9          until no coset representative was added by filtration at line 8;
10     return C_0, ..., C_{n−2}
```



Generators of G (i.e. the set S)

in the cyclic notation:

$\alpha : (a_1, a_4, a_2)$

$\beta : (a_1, a_4), (a_2, a_5), (a_3, a_6)$

Figure 7: Example of the graph Γ constructed by Algorithm 4. Here $A = \{a_1, \ldots, a_6\}$, the group G is represented by two generators α (blue) and β (red). The group G has two orbits: $\{a_1, a_2, a_4, a_5\}$ and $\{a_3, a_6\}$. The orbit of a_1 contains a_5, because for $\gamma = \alpha^2\beta \in G$ we have $\gamma(a_1) = a_5$.

Proof. The term $O(|S|n^2)$ stands for filtering the generators from the set S at line 3.

The addition tested at line 9 may happen at most $O(n^2)$ times; for each there are $O(n)$ choices of σ, $O(n^2)$ choices of τ and for each such combination $O(n^2)$ time spent on filtration. □

We could get more efficient code if a queue of the newly added elements is used instead the repeat–until loop.

Recall that the *index* of a subgroup H in a group G is the ratio $\frac{|G|}{|H|}$.

Lemma 6. *Let H be a subgroup of $G \subseteq \mathrm{Sym}(A)$. If H has a polynomial index in G and the membership test for H can be performed in polynomial time then from any set S of polynomially many generators of G we can construct the set of strong generators for H in polynomial time.*

Proof. We alter slightly Algorithm 3 for finding strong generators by adding the set C_{-1} of coset representatives of H in G. This can be done by adding one more filtration step to the beginning of the Algorithm 2. We start with filtration whether $\beta^{-1}\alpha \in H$ for some $\beta \in C_{-1}$ and if not, then we add α to C_{-1}.

From assumptions we know that the additional filtration step can be done in polynomial time. Also because H has a polynomial index in G we add some α to C_{-1} at most polynomially many times.

The resulting set of strong generators for H is $C_0, \ldots C_{n-2}$. □

4 Concepts from group theory

We say that G *acts* on A, or that G is an *action* on A, if $G \subseteq \mathrm{Sym}(A)$. (More properly, an action of a group G on a set A is a homomorphism $G \to \mathrm{Sym}(A)$, but we keep our actions faithful, i.e. injective, and such could be seen just as subgroups of $\mathrm{Sym}(A)$.)

We say that $G \subseteq \mathrm{Sym}(A)$ *stabilizes* $B \subseteq A$, or equivalently that B is *G-stable*, if

$$\forall \alpha \in G\colon \alpha(B) = B.$$

The *G-orbit* of an element $a \in A$ is the set $\{\alpha(a)\colon \alpha \in G\}$. A group G is a *transitive* action, or equivalently that G acts *transitively*, if it has exactly one orbit.

The following Algorithm 4 splits a group G into orbits. See Fig. 7 for an example. We later use it for branching to smaller problems by divide & conquer technique.

Claim 7. *The time complexity of Orbits is $O(|S| \cdot |A|)$. In particular, for $n = |A|$ and $|S| = O(n^2)$ the complexity is $O(n^3)$.*

The *order* of an element α is the smallest $k \in \mathbb{N}$ such that $\alpha^k = \mathrm{id}$. We call a group G a *p-group*, where p is a prime, if each element of G has order p^i for some i. Consequently, the order of G is also a power of p.

Since each subgroup of a p-group is also a p-group [5], then by the construction of sets R and S from Sections 2.2 and 2.3 we have:

Fact 8. *For each k, the group $\mathrm{Aut}_e(X_k)$ is a 2-group.*

Algorithm 4: Orbits (S, A)

Input: Action G on a set A given by a set S of generators, i.e. $G = \langle S \rangle$
Output: Orbits of G on A

```
1 begin
2     create the empty graph Γ on the vertex set A;
3     foreach α ∈ S, a ∈ A do
4         add the edge (a, α(a));
5     return connected components of Γ
```

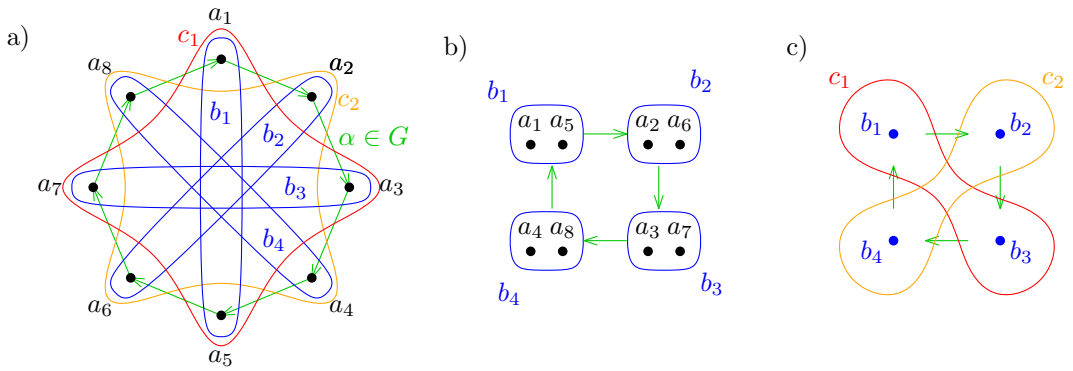


Figure 8: a) Example of two block systems for $G = \langle \alpha \rangle$ on $A = \{a_1, \ldots, a_8\}$, where $\{b_1, \ldots, b_4\}$ is not minimal and $\{c_1, c_2\} = \{\{a_1, a_3, a_5, a_7\}, \{a_2, a_4, a_6, a_8\}\}$ is minimal. b) The action of α on the blocks $b_1, \ldots, b_4$. c) The block system obtained in the second iteration of Algorithm 5 corresponds to the minimal block system $\{c_1, c_2\}$ on A.

We say that a set $B \subseteq A$ is a *G-block* if $\forall \alpha \in G$: $B = \alpha(B)$ or $B \cap \alpha(B) = \emptyset$. If B is a G-block, then the set $\{\alpha(B) \colon \alpha \in G\}$ is a partition of A into disjoint sets of equal size.

Observation 9. *If a group G is transitive on A, then G is transitive also on G-blocks.*

A transitive group G acts *primitively* on A if it does not have blocks of size greater than 1.

Definition. *A nontrivial block system $\mathcal{B} = \{B_1, \ldots, B_k\}$, $k \geq 2$ is* minimal *if G acts primitively on its blocks $B_1, \ldots, B_k$.*

It means that it has the minimal number of blocks and we cannot merge any remaining blocks, see Fig. 8 a).

The following lemma is well known:

Lemma 10 ([5])**.** *If G is a transitive p-group action on A, $|A| > 1$ then each nontrivial minimal G-block system has exactly p blocks.*

Proof. Let $\mathcal{B}$ be a minimal G-block system. In this proof all actions of G and its subgroups are considered on $\mathcal{B}$.

Denote by G_1 the subgroup of G that stabilizes the first block $B_1 \in \mathcal{B}$, namely $G_1 = \{\alpha \in G : \alpha(B_1) = B_1\}$. Consider a subgroup H of G where $G_1 \subseteq H$.

We claim that $C = \{\sigma B_1 \colon \sigma \in H\}$ is a G-block of $\mathcal{B}$. Whenever $B \in C \cap \alpha(C)$, then $B = \sigma B_1 = \alpha\tau B_1$ for some $\sigma, \tau \in H$. Thus $B_1 = \sigma^{-1}\alpha\tau B_1$, so $\sigma^{-1}\alpha\tau \in G_1$ and therefore $\alpha = \sigma\sigma^{-1}\alpha\tau\tau^{-1} \in H$. Consequently $\alpha(C) = C$, so C is a G-block as it was claimed.

If there was an H strictly between G_1 and G, then $\mathcal{B}$ would not be nontrivial and minimal, which contradicts our assumptions. So G_1 is a maximal subgroup of G. Each maximal subgroup of a p-group has index p. Finally, as G is transitive on $\mathcal{B}$, we get that $|\mathcal{B}| = \frac{G}{G_1} = p$. □

Algorithm 5: Block system(A, S)

Input: A set S generating a group action G on $A = \{a_1, \ldots, a_n\}$
Output: A G-block system on A

```
1 begin
2     for i = 2 to n do
3         create the empty graph Γ_i on the vertex set A;
4         add (a_1, a_i) ∈ E(Γ_i);
5         foreach (a, a') ∈ E(Γ_i), α ∈ S do
6             add (α(a), α(a')) ∈ E(Γ_i);
7         if Γ_i is not connected then
8             return connectivity components of Γ_i
```

Algorithm 6: Minimal block system (A, S)

Input: A set S generating a transitive p-group action G on A
Output: A minimal G-block system on A

```
1 begin
2     B = A;
3     repeat
4         B := Block system(B, S)
5     until |B| = p;
6     return B
```

Our goal is to find a minimal block system. We use the following procedure to make from a given block system a new block system with fewer blocks.

Correctness of Algorithm 6 follows from the Lemma 10: When the block system has more than p blocks, there exists i such that a_1, a_i are in the same block of the minimal block system, hence at least one graph Γ_i constructed by Algorithm 5 is disconnected. The corresponding block system need not to be minimal, so it is necessary to iterate Algorithm 6 until exactly p blocks are obtained, see Fig. 8 b-c).

Claim 11. *The time complexity of Minimal block system is $O(|A|^3|S|)$. In particular for $n = |A|$ and $|S| = O(n^2)$ the complexity is $O(n^5)$.*

Proof. The construction of each Γ_i needs $O(|A|^2|S|)$ time as there are $|A|^2$ pairs (a, a') and for each $|S|$ possible shifts. This bound already covers the cost $O(|A|^2)$ of connectivity test and finding components.

In each loop at line 4 of Algorithm 6 the size of ground set of the block system $\mathcal{B}$ is made p times smaller. Together with the line 2 of Algorithm 5 the graph Γ_i is generated at most $\sum_{i=0}^{\lfloor \log_p n \rfloor} \frac{n}{p^i} < \frac{pn}{p-1} = O(n)$ times. □

5 Finding color preserving subgroups

5.1 Cosets of permutations preserving colors on a block

For elements $a, b \in A$ we write $a \sim b$ when a has the same color as b.

Notation. *For $B \subseteq A$, $K \subseteq \mathrm{Sym}(A)$ we denote by $\mathcal{C}_B(K)$ is the subset of permutations from K preserving colors on elements of B (note that we do not require $\alpha(b) \in B$, see Fig. 9), formally:*

$$\mathcal{C}_B(K) = \{\alpha \in K \colon \forall b \in B \colon b \sim \alpha(b)\}.$$

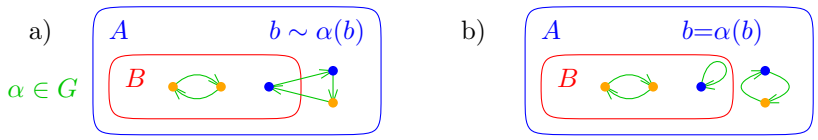


Figure 9: a) Example of a color preserving action α on B. b) A color preserving action α stabilizing B.

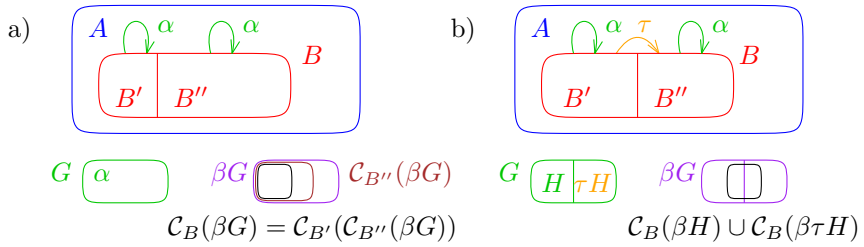


Figure 10: a) Case 1 — G does not act transitively on B. b) Case 2 — G acts transitively on B.

The goal is to determine $\mathcal{C}_A(G)$. We solve this problem in more general setting, namely, we determine $\mathcal{C}_B(\beta G)$, where B is G-stable, $G \subseteq \mathrm{Sym}(A)$ and $\beta \in \mathrm{Sym}(A)$. Then for $B = A$ and $\beta = \mathrm{id}$ we get $\mathcal{C}_A(G)$.

The two following observations are immediate:

Observation 12. $\mathcal{C}_B(K \cup K') = \mathcal{C}_B(K) \cup \mathcal{C}_B(K')$

Observation 13. $\mathcal{C}_{B \cup B'}(K) = \mathcal{C}_B(\mathcal{C}_{B'}(K))$

Lemma 14. *If B is G-stable then either $\mathcal{C}_B(\beta G) = \emptyset$ or $\mathcal{C}_B(\beta G)$ is a coset of a subgroup $\mathcal{C}_B(G)$ of G.*

Proof. As B is G-stable then $\forall \sigma, \tau \in \mathcal{C}_B(G)$ we have that σ and τ preserve colors on B, namely $\forall b \in B\colon \sigma(b) \sim \tau(b) \sim b$. Consequently, their composition $\sigma\tau$ preserves colors as well, i.e. $\sigma\tau(b) \sim b$ and we get that $\sigma\tau \in \mathcal{C}_B(G)$. Hence $\mathcal{C}_B(G)$ is a subgroup of $\mathrm{Sym}(A)$.

If $\mathcal{C}_B(\beta G) \neq \emptyset$ then there exists $\gamma \in \mathcal{C}_B(\beta G)$. Especially $\gamma \in \beta G$ and so $\beta G = \gamma G$, as we may change the coset representative from β to γ. Because γ is color preserving on B, the following implication holds:

$$\forall \alpha \in G, \forall b \in B : \alpha(b) \in B \Rightarrow \gamma\alpha(b) \sim \alpha(b),$$

The assumption of the implication is valid for all $\alpha \in G$ as B is G-stable, so we obtain:

$$\alpha \in \mathcal{C}_B(G) \Leftrightarrow \gamma\alpha \in \mathcal{C}_B(\gamma G)$$

because either both $\alpha(b)$ and $\gamma\alpha(b)$ have the same color as b or none of them.

Therefore $\mathcal{C}_B(\gamma G) = \gamma \mathcal{C}_B(G)$, in other words $\mathcal{C}_B(\gamma G)$ is a coset of $\mathcal{C}_B(\beta G)$. □

5.2 Recursive coset calculation

In order to determine $\mathcal{C}_B(\beta G)$ we distinguish two cases:

Case 1. *The group G does not act transitively on B, see Fig. 10 a).*

There is more than one G-orbit on B, so there exist blocks B', B'' such that $B = B' \cup B''$ and both B', B'' are G-stable. We use the Observation 13 and get:

$$\mathcal{C}_B(\beta G) = \mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta G))$$

Case 2. *The group G acts transitively on B, see Fig. 10 b).*

Due to Fact 8 and Lemma 10 we use Algorithm 6 and split B into two G-blocks B' and B'' such that $B = B' \cup B''$.

In the next step we determine generators of a subgroup H, which stabilizes B' as follows: The membership of any permutation α in H can be decided in polynomial time by checking whether $\forall b \in B' \colon \alpha(b) \in B'$. Also as $G = H \cup \tau H$, where τ is a permutation which switches blocks B' and B'', we get that the index of H in G is 2. So we can compute generators of the subgroup H from generators of G in polynomial time due to Lemma 6. Note that H stabilizes B'' as well.

Due to Observation 12 (for $G = H \cup \tau H$) and Observation 13 (for $B = B' \cup B''$) the following continued equality holds:

$$\mathcal{C}_B(\beta G) = \mathcal{C}_B(\beta H \cup \beta\tau H) = \mathcal{C}_B(\beta H) \cup \mathcal{C}_B(\beta\tau H) = \mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta H)) \cup \mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta\tau H))$$

By Lemma 14 the first set $\mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta H))$ is either empty or could be expressed as $\gamma\mathcal{C}_B(H)$ for a suitable $\gamma \in G$. Similarly, the second set is either empty or could be written as $\delta\mathcal{C}_B(H)$ for a $\delta \in G$.

If both are nonempty then Lemma 14 guarantees that $\mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta H))$ and $\mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta\tau H))$, or equivalently $\gamma\mathcal{C}_B(H)$ and $\delta\mathcal{C}_B(H)$, "must paste together neatly to a single coset"[3], namely $\gamma \langle \mathcal{C}_B(H) \cup \gamma^{-1}\delta \rangle$.

It remains to describe the situation when the recursion stops, i.e. when $|B| = 1$. Then we have directly

$$\mathcal{C}_B(\beta G) = \mathcal{C}_{\{b\}}(\beta G) = \begin{cases} \beta G & \text{if } \beta(b) \sim b, \\ \emptyset & \text{otherwise.} \end{cases}$$

Algorithm 7: $\mathcal{C}_B(\beta G)$

Input: A G-stable set $B \subseteq A$, $\beta \in Sym(A)$ and a 2-group G given by a set S of generators
Output: Coset $\mathcal{C}_B(\beta G)$ w.r.t. $\sim$ on A, described by generators and a representative

```
1  begin
2      if |B| = 1 with B = {b} then
3          if β(b) ∼ b then return βG ;
4          else return ∅;
5      else if |Orbits(S, B)| > 1 then
           // G does not act transitively on B
6          B' := any component of Orbits(S, B);
7          B'' := B \ B';
8          return C_B'(C_B''(βG))
9      else
           // G acts transitively on B
10         {B', B''} := Minimal block system(B, S);
11         S' := Strong generators(S) of a subgroup H ⊂ G that stabilizes B';
12         τ := any element of G \ H;
13         if C_B'(C_B''(βH)) = ∅ then return C_B'(C_B''(βτH));
14         else
15             if C_B'(C_B''(βτH)) = ∅ then return C_B'(C_B''(βH));
16             else
17                 γC_B(H) := C_B'(C_B''(βH));
18                 δ := any element of C_B'(C_B''(βτH));
19                 return γ⟨C_B(H) ∪ γ^{-1}δ⟩
```

Claim 15. *The time complexity of Algorithm 7 for computing $\mathcal{C}_B(\beta G)$ is $O(|B|^3|S| + |B|^7)$. In particular for $n = |B|$ and $|S| = O(n^2)$ the complexity is $O(n^7)$.*

Proof. Let $T(b,s)$ be the running time of $\mathcal{C}_B(\beta G)$ for $b = |B|$ and $s = |S|$ generating G.

Line 5 requires $O(bs)$ time by Claim 7.

The recursive call at line 8 requires $T(b',s) + T(b'',s)$ time with $b = b' + b''$ and $b', b'' \geq 1$.

In the other branch, line 10 requires $O(b^3 s)$ time by Claim 11. Line 11 requires $O(b^2 s + b^7)$ time by Claim 5. At lines 13–19 there are two recursive calls of $\mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta H))$ and $\mathcal{C}_{B'}(\mathcal{C}_{B''}(\beta\tau H))$, both of which require $2T(\frac{b}{2}, s-1)$ time.

In total we have:

$$T(b,s) \leq O(bs) + \max\{T(b',s) + T(b'',s), 4T(\tfrac{b}{2}, s-1) + O(b^3 s + b^7)\}$$

By Master theorem, the overall complexity is driven by the term $O(b^3 s + b^7)$. It outperforms the $O(b^2)$ complexity of the $4T(\frac{b}{2}, s-1)$ in the recursion. □

6 Finalizing the algorithm

The following algorithm summarizes all the steps described so far.

Algorithm 8: Generators of $\mathrm{Aut}_e(X)$

Input: A cubic graph X and $e \in E_X$
Output: Generators of the group $\mathrm{Aut}_e(X)$

```
1  begin
2      k := 1;
3      S_1 := the only generator of Sym(e) transposing e;
4      repeat
5          A := V_{X_k} ∪ (V_k choose 2) ∪ (V_k choose 3);
6          determine ∼ on A according to the three sets B_e, B_1 and B_2;
7          S' := generators of C_A(⟨S_k⟩);
8          k++;
9          R := {(u,v): u,v are twins in V_k};
10         S_k := R ∪ {π_{k-1}^{-1}(φ): φ' ∈ S'};
11     until X = X_k;
12     return S_k
```

Claim 16. *The time complexity of Algorithm 8 for computing generators of* $\mathrm{Aut}_e(X)$ *is* $O(n^{21})$*, where* $n = |V_X|$.

Proof. The sum s of the sizes of the sets A constructed at line 5 is $s = O(n^3)$. The most demanding computation is performed at line 7 which by Claim 15 requires $O(s^7) = O((n^3)^7) = O(n^{21})$ operations, when summarized over all iterations of the **repeat**–**until** cycle. □

6.1 Concluding remarks

Isomorphism of cubic graphs can indeed be tested in $O(n^3 \log n)$ time [1]. For further reading see also monograph [2] and PhD thesis [4].

References

[1] Zvi Galil, Christoph M. Hoffmann, Eugene M. Luks, Claus P. Schnorr, and Andreas Weber. *An $O(n^3 \log n)$ deterministic and an $O(n^3)$ Las Vegas isomorphism test for trivalent graphs. Journal of the ACM*, 34(3):513–531, July 1987.

[2] Christoph M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism*, volume 136 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin-Heidelberg-New York, 1982.

[3] Luks, Eugene M. *Isomorphism of graphs of bounded valence can be tested in polynomial time*. Journal of computer and system sciences, 1982.

[4] Adria Alcala Mena. *Trivalent Graph isomorphism in polynomial time*. PhD thesis, Universidad de Cantabria, 2012.

[5] Wielandt, Helmut. *Finite permutation groups*. New York: Academic Press, 1964.