A short introduction to Babai's quasipolinomial graph isomorphism testing

Ilia Ponomarenko

St.Petersburg Department of V.A.Steklov Institute of Mathematics
of the Russian Academy of Sciences

The International Conference and PhD-Master Summer School
on "Graphs and Groups, Spectra and Symmetries" (G2S2)
Novosibirsk, Russia, August 15-28, 2016.

The main statement

The printed version: `arXiv:1512.03547v2`
The video lectures: at Babai's home page.

The main statement

The printed version: `arXiv:1512.03547v2`
The video lectures: at Babai's home page.

Theorem (Babai, 2015)

The following problems can be solved in quasipolynomial time $\exp((\log n)^{O(1)})$: Graph Isomorphism, String Isomorphism, Coset Intersection.

The main statement

The printed version: `arXiv:1512.03547v2`
The video lectures: at Babai's home page.

Theorem (Babai, 2015)

The following problems can be solved in quasipolynomial time $\exp((\log n)^{O(1)})$: Graph Isomorphism, String Isomorphism, Coset Intersection.

Graph Isomorphism. Known: $\exp(O(\sqrt{n \log n}))$ (Kantor-Luks-Babai, 1983).

The main statement

The printed version: `arXiv:1512.03547v2`
The video lectures: at Babai's home page.

Theorem (Babai, 2015)

The following problems can be solved in quasipolynomial time $\exp((\log n)^{O(1)})$: Graph Isomorphism, String Isomorphism, Coset Intersection.

Graph Isomorphism. Known: $\exp(O(\sqrt{n \log n}))$ (Kantor-Luks-Babai, 1983).
String Isomorphism. For strings s, s' and a group G, find $\mathrm{Iso}_G(s, s')$, where $s, s' : \Omega \to \Sigma$ are strings over the alphabet Σ and $\mathrm{Iso}_G(s, s') = \{f \in G : \ s^f = s'\}$.

The main statement

The printed version: `arXiv:1512.03547v2`
The video lectures: at Babai's home page.

Theorem (Babai, 2015)

The following problems can be solved in quasipolynomial time $\exp((\log n)^{O(1)})$: Graph Isomorphism, String Isomorphism, Coset Intersection.

Graph Isomorphism. Known: $\exp(O(\sqrt{n \log n}))$ (Kantor-Luks-Babai, 1983).
String Isomorphism. For strings s, s' and a group G, find $\mathrm{Iso}_G(s, s')$, where $s, s' : \Omega \to \Sigma$ are strings over the alphabet Σ and $\mathrm{Iso}_G(s, s') = \{f \in G : s^f = s'\}$.
Coset intersection. For $G, G' \leq \mathrm{Sym}(\Omega)$ and $f, f' \in \mathrm{Sym}(\Omega)$, find $Gf \cap G'f'$.

The sense of “quasipolynomial”

- The algorithm splits the problem of size n into $q(n)$ subproblems of size cn, where $0 < c < 1$.

The sense of “quasipolynomial”

- The algorithm splits the problem of size n into $q(n)$ subproblems of size cn, where $0 < c < 1$.
- So for the computational complexity $F(n)$ of the algorithm, $F(n) \leq q(n)F(cn)$.

The sense of "quasipolynomial"

- The algorithm splits the problem of size n into $q(n)$ subproblems of size cn, where $0 < c < 1$.
- So for the computational complexity $F(n)$ of the algorithm, $F(n) \leq q(n)F(cn)$.
- Thus, $F(n) \leq q(n)^{O(\log n)}$.

The sense of “quasipolynomial”

- The algorithm splits the problem of size n into $q(n)$ subproblems of size cn, where $0 < c < 1$.
- So for the computational complexity $F(n)$ of the algorithm, $F(n) \leq q(n)F(cn)$.
- Thus, $F(n) \leq q(n)^{O(\log n)}$.

To get quasipolynomial bound, the function $q(n)$ must be quasipolunomially bounded.

From Graph Isomorphism to String Isomorphism

- Let Γ, Γ' be graphs with vertices $\{1, \ldots, n\}$,

From Graph Isomorphism to String Isomorphism

- Let Γ, Γ' be graphs with vertices $\{1, \ldots, n\}$,
- set s, s' to be the strings on $\Omega = \{1, \ldots, n\}^2$ over the alphabet $\Sigma = \{0, 1\}$ obtained from the adjacency matrices of Γ and Γ', respectively,

From Graph Isomorphism to String Isomorphism

- Let Γ, Γ' be graphs with vertices $\{1, \ldots, n\}$,
- set s, s' to be the strings on $\Omega = \{1, \ldots, n\}^2$ over the alphabet $\Sigma = \{0, 1\}$ obtained from the adjacency matrices of Γ and Γ', respectively,
- set G to be the permutation group induced by the action of $\mathrm{Sym}(n)$ on pairs, $G \leq \mathrm{Sym}(\Omega)$.

From Graph Isomorphism to String Isomorphism

- Let Γ, Γ' be graphs with vertices $\{1, \ldots, n\}$,
- set s, s' to be the strings on $\Omega = \{1, \ldots, n\}^2$ over the alphabet $\Sigma = \{0, 1\}$ obtained from the adjacency matrices of Γ and Γ', respectively,
- set G to be the permutation group induced by the action of $\mathrm{Sym}(n)$ on pairs, $G \leq \mathrm{Sym}(\Omega)$.
- Now $\Gamma \cong \Gamma'$ iff $\mathrm{Iso}_G(s, s') \neq \varnothing$.

The Luks Algorithm: preliminaries

For $K \subset \mathrm{Sym}(\Omega)$ and $\Delta \subset \Omega$, set

$$\mathrm{Iso}_K^{\Delta}(s, s') = \{k \in K : \; s(\alpha) = s'(\alpha^k) \text{ for all } \alpha \in \Delta\}.$$

The Luks Algorithm: preliminaries

For $K \subset \mathrm{Sym}(\Omega)$ and $\Delta \subset \Omega$, set

$$\mathrm{Iso}_K^{\Delta}(s, s') = \{k \in K : \ s(\alpha) = s'(\alpha^k) \ \text{for all} \ \alpha \in \Delta\}.$$

Remarks

- The String Isomorphism is to find $\mathrm{Iso}_K^{\Delta}(s, s')$ with $K = G$ and $\Delta = \Omega$.
- In what follows, K is a right coset of G or empty, and Δ is G-invariant.

The Luks Algorithm: preliminaries

For $K \subset \mathrm{Sym}(\Omega)$ and $\Delta \subset \Omega$, set

$$\mathrm{Iso}_K^{\Delta}(s, s') = \{k \in K : \; s(\alpha) = s'(\alpha^k) \; \text{for all} \; \alpha \in \Delta\}.$$

Remarks

- The String Isomorphism is to find $\mathrm{Iso}_K^{\Delta}(s, s')$ with $K = G$ and $\Delta = \Omega$.
- In what follows, K is a right coset of G or empty, and Δ is G-invariant.

Proposition

Let $K = Gf$, where $f \in \mathrm{Sym}(\Omega)$ and $\Delta^G = \Delta$. Then

The Luks Algorithm: preliminaries

For $K \subset \mathrm{Sym}(\Omega)$ and $\Delta \subset \Omega$, set

$$\mathrm{Iso}_K^{\Delta}(s, s') = \{k \in K : \; s(\alpha) = s'(\alpha^k) \text{ for all } \alpha \in \Delta\}.$$

Remarks

- The String Isomorphism is to find $\mathrm{Iso}_K^{\Delta}(s, s')$ with $K = G$ and $\Delta = \Omega$.
- In what follows, K is a right coset of G or empty, and Δ is G-invariant.

Proposition

Let $K = Gf$, where $f \in \mathrm{Sym}(\Omega)$ and $\Delta^G = \Delta$. Then

- $\mathrm{Aut}_G^{\Delta}(s)$ is a subgroup of G,
- $\mathrm{Iso}_{Gf}^{\Delta}(s, s')$ is either empty or a right coset of $\mathrm{Aut}_G^{\Delta}(s)$.

The Luks Algorithm: intransitive case

Input: strings s, s' on Ω, group $G \leq \mathrm{Sym}(\Omega)$, a G-invariant set $\Delta \subset \Omega$, and $f \in \mathrm{Sym}(\Omega)$.

The Luks Algorithm: intransitive case

Input: strings s, s' on Ω, group $G \leq \mathrm{Sym}(\Omega)$, a G-invariant set $\Delta \subset \Omega$, and $f \in \mathrm{Sym}(\Omega)$.
Output: $\mathrm{Iso}^{\Delta}_{Gf}(s, s')$.

The Luks Algorithm: intransitive case

Input: strings s, s' on Ω, group $G \leq \mathrm{Sym}(\Omega)$, a G-invariant set $\Delta \subset \Omega$, and $f \in \mathrm{Sym}(\Omega)$.
Output: $\mathrm{Iso}^{\Delta}_{Gf}(s, s')$.

Step 1 (G is intransitive on Δ). Let $\Delta = \Delta_1 \cup \Delta_2$, where Δ_i is G-invariant nonempty set. Then

$$\mathrm{Iso}^{\Delta_1 \cup \Delta_2}_{Gf}(s, s') = \mathrm{Iso}^{\Delta_2}_{G_1 f_1}(s, s'),$$

where $G_1 f_1 = \mathrm{Iso}^{\Delta_1}_{Gf}(s, s')$.

The Luks Algorithm: intransitive case

Input: strings s, s' on Ω, group $G \leq \mathrm{Sym}(\Omega)$, a G-invariant set $\Delta \subset \Omega$, and $f \in \mathrm{Sym}(\Omega)$.
Output: $\mathrm{Iso}_{Gf}^{\Delta}(s, s')$.

Step 1 (G is intransitive on Δ). Let $\Delta = \Delta_1 \cup \Delta_2$, where Δ_i is G-invariant nonempty set. Then

$$\mathrm{Iso}_{Gf}^{\Delta_1 \cup \Delta_2}(s, s') = \mathrm{Iso}_{G_1 f_1}^{\Delta_2}(s, s'),$$

where $G_1 f_1 = \mathrm{Iso}_{Gf}^{\Delta_1}(s, s')$.

The time bound is $F(n) \leq F(n_1) + F(n_2)$, where $n = n_1 + n_2$ and $n_i = |\Delta_i|$.

The Luks Algorithm: transitive case

Step 2 (*G* is "imprimitive" on Δ). Let $\Delta = \Delta_1 \cup \cdots \cup \Delta_m$ is a *G*-invariant partition with the maximum possible $|\Delta_1| < |\Omega|$ (it is possible that $|\Delta_1| = 1$ and *G* is primitive).

The Luks Algorithm: transitive case

Step 2 (G is "imprimitive" on Δ). Let $\Delta = \Delta_1 \cup \cdots \cup \Delta_m$ is a G-invariant partition with the maximum possible $|\Delta_1| < |\Omega|$ (it is possible that $|\Delta_1| = 1$ and G is primitive).

Set $\psi : G \to \mathrm{Sym}(m)$ to be the induced homomorphism.

$$N = \ker(\psi), \quad \overline{G} = \mathrm{im}(\psi), \quad k = |\overline{G}|.$$

Then

$$\mathrm{Iso}^{\Delta}_{Gf}(s, s') = \bigcup_{i=1}^{k} \mathrm{Iso}^{\Delta}_{Ng_i f}(s, s'),$$

where $\{g_1, \ldots, g_k\}$ is a full set of distinct representatives of the family $\{\psi^{-1}(\overline{g}) : \overline{g} \in \overline{G}\}$. Here,

$$\mathrm{Orb}(N, \Delta) = \{\Delta_1 \ldots, \Delta_m\} \quad \text{and} \quad |\Delta_i| = n/m.$$

The Luks Algorithm: transitive case

Step 2 (G is "imprimitive" on Δ). Let $\Delta = \Delta_1 \cup \cdots \cup \Delta_m$ is a G-invariant partition with the maximum possible $|\Delta_1| < |\Omega|$ (it is possible that $|\Delta_1| = 1$ and G is primitive).

Set $\psi : G \to \mathrm{Sym}(m)$ to be the induced homomorphism.

$$N = \ker(\psi), \quad \overline{G} = \mathrm{im}(\psi), \quad k = |\overline{G}|.$$

Then

$$\mathrm{Iso}^{\Delta}_{Gf}(s, s') = \bigcup_{i=1}^{k} \mathrm{Iso}^{\Delta}_{Ng_i f}(s, s'),$$

where $\{g_1, \dots, g_k\}$ is a full set of distinct representatives of the family $\{\psi^{-1}(\overline{g}) : \overline{g} \in \overline{G}\}$. Here,

$$\mathrm{Orb}(N, \Delta) = \{\Delta_1 \dots, \Delta_m\} \quad \text{and} \quad |\Delta_i| = n/m.$$

Thus, the time bound here is $F(n) \leq mkF(n/m)$.

The Luks Algorithm: remarks

- If the group G is solvable, then $k = m^{O(1)}$ and $F(n) \leq n^{O(1)}$.

The Luks Algorithm: remarks

- If the group G is solvable, then $k = m^{O(1)}$ and $F(n) \leq n^{O(1)}$.
- If all non-abelian composition factors of G are bounded by a constant, say d, then $k = m^{g(d)}$ and $F(n) \leq n^{g(d)}$, where g is a function.

The Luks Algorithm: remarks

- If the group G is solvable, then $k = m^{O(1)}$ and $F(n) \leq n^{O(1)}$.
- If all non-abelian composition factors of G are bounded by a constant, say d, then $k = m^{g(d)}$ and $F(n) \leq n^{g(d)}$, where g is a function.
- If $k = |\overline{G}| = n^{O(\log n)}$ for all primitive groups $\overline{G}$ occuring in the Luks algorithm, then

$$F(n) \leq n^{(\log n)^{O(1)}}.$$

The Luks Algorithm: remarks

- If the group G is solvable, then $k = m^{O(1)}$ and $F(n) \leq n^{O(1)}$.
- If all non-abelian composition factors of G are bounded by a constant, say d, then $k = m^{g(d)}$ and $F(n) \leq n^{g(d)}$, where g is a function.
- If $k = |\overline{G}| = n^{O(\log n)}$ for all primitive groups $\overline{G}$ occuring in the Luks algorithm, then

$$F(n) \leq n^{(\log n)^{O(1)}}.$$

In all the above cases the Luks algorithm is quasipolynomial.

An obstacle to the Luks approach

Let $A_m^{(t)}$ and $S_m^{(t)}$ be the actions of $\mathrm{Alt}(m)$ and $\mathrm{Sym}(m)$ on $\binom{m}{t}$.

An obstacle to the Luks approach

Let $A_m^{(t)}$ and $S_m^{(t)}$ be the actions of Alt(m) and Sym(m) on $\binom{m}{t}$.

Definition

The Cameron group $G \leq \mathrm{Sym}(n)$ with parameters $s, t \geq 1$ and $m \geq \max\{2t, 5\}$ is defined by the following conditions:

$$n = \binom{m}{t}^s \quad \text{and} \quad (A_m^{\{t\}})^s \leq G \leq S_m^{\{t\}} \wr \mathrm{Sym}(s)$$

and the action of G on the factors of Soc(G) is transitive.

An obstacle to the Luks approach

Let $A_m^{(t)}$ and $S_m^{(t)}$ be the actions of $\mathrm{Alt}(m)$ and $\mathrm{Sym}(m)$ on $\binom{m}{t}$.

Definition

The Cameron group $G \le \mathrm{Sym}(n)$ with parameters $s, t \ge 1$ and $m \ge \max\{2t, 5\}$ is defined by the following conditions:

$$n = \binom{m}{t}^s \quad \text{and} \quad (A_m^{\{t\}})^s \le G \le S_m^{\{t\}} \wr \mathrm{Sym}(s)$$

and the action of G on the factors of $\mathrm{Soc}(G)$ is transitive.

Theorem (Cameron, 1981)

For $n \ge 25$, if G is primitive and $|G| \ge n^{1+\log_2 n}$, then G is a Cameron group.

Using CFSG were removed by Pyber (2016).

Reduction to Johnson scheme

The Luks algorithm stops if $|\overline{G}|$ is larger than quasipolynomial, i.e. if $|\overline{G}|$ is a Cameron group.

Reduction to Johnson scheme

The Luks algorithm stops if $|\overline{G}|$ is larger than quasipolynomial, i.e. if $|\overline{G}|$ is a Cameron group.

In fact, one can continue until $\overline{G}$ becomes a Johnson group, i.e., the Cameron group with $s = 1$:

$$A_m^{\{t\}} \leq G \leq S_m^{\{t\}}$$

Reduction to Johnson scheme

The Luks algorithm stops if $|\overline{G}|$ is larger than quasipolynomial, i.e. if $|\overline{G}|$ is a Cameron group.

In fact, one can continue until $\overline{G}$ becomes a Johnson group, i.e., the Cameron group with $s = 1$:

$$A_m^{\{t\}} \leq G \leq S_m^{\{t\}}$$

It is well known that

$$\bigcap_{i=0}^{t} \mathrm{Aut}(R_i) = S_m^{\{t\}},$$

where $t \geq 2$, $m \geq 2t + 1$, and

$$R_i = \{(S, T) \in \Omega^2 : |S \cap T| = t - i\}.$$

with $\Omega = \binom{m}{t}$.

Reduction to Johnson scheme

The Luks algorithm stops if $|\overline{G}|$ is larger than quasipolynomial, i.e. if $|\overline{G}|$ is a Cameron group.

In fact, one can continue until $\overline{G}$ becomes a Johnson group, i.e., the Cameron group with $s = 1$:

$$A_m^{\{t\}} \leq G \leq S_m^{\{t\}}$$

It is well known that

$$\bigcap_{i=0}^{t} \mathrm{Aut}(R_i) = S_m^{\{t\}},$$

where $t \geq 2$, $m \geq 2t + 1$, and

$$R_i = \{(S, T) \in \Omega^2 : |S \cap T| = t - i\}.$$

with $\Omega = \binom{m}{t}$. Here $J(m, t) = (\Omega, \{R_i\})$ is the Johnson scheme.

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c \log_2 n$.

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c\log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$;

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c \log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$; replace G by $\ker(\varphi)$.

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c\log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$; replace G by $\ker(\varphi)$.
- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$, or vice versa;

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c \log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$; replace G by $\ker(\varphi)$.
- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$, or vice versa; here $\mathrm{Iso}_G(s, s') = \varnothing$.

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c\log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$; replace G by $\ker(\varphi)$.
- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$, or vice versa; here $\mathrm{Iso}_G(s, s') = \varnothing$.
- $\mathrm{Aut}_G(s)^\varphi < \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$;

Removing obstackles in Luks's algorithm

Since G is a Cameron group, one can find an explicitly given homomorphism

$$\varphi : G \to \mathrm{Alt}(\Gamma),$$

where Γ is a set of cardinality $m = c \log_2 n$.

Final steps of Babai's algorithm:

- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi \geq \mathrm{Alt}(\Gamma)$;
  replace G by $\ker(\varphi)$.
- $\mathrm{Aut}_G(s)^\varphi \geq \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$, or vice versa;
  here $\mathrm{Iso}_G(s, s') = \varnothing$.
- $\mathrm{Aut}_G(s)^\varphi < \mathrm{Alt}(\Gamma)$ and $\mathrm{Aut}_G(s')^\varphi < \mathrm{Alt}(\Gamma)$;
  replace G by $H^{\varphi^{-1}}$, where $H = \langle \mathrm{Aut}_G(s)^\varphi, \mathrm{Aut}_G(s')^\varphi \rangle$.

Local certificates

A set $\Delta \subset \Gamma$ is said to be full if

$$(G_\Delta \cap \mathrm{Aut}_G(s))^\varphi \geq \mathrm{Alt}(\Delta),$$

where $G_\Delta = ((G^\varphi)_{\{\Delta\}})^{\varphi^{-1}}$.

Local certificates

A set $\Delta \subset \Gamma$ is said to be full if

$$(G_\Delta \cap \mathrm{Aut}_G(s))^\varphi \geq \mathrm{Alt}(\Delta),$$

where $G_\Delta = ((G^\varphi)_{\{\Delta\}})^{\varphi^{-1}}$.

1. The non-fullness certificate is a group $M = M(\Delta) \leq \mathrm{Sym}(\Delta)$ such that

$$M \not\geq \mathrm{Alt}(\Delta) \quad \text{and} \quad \mathrm{Aut}_G(s)^\varphi \leq M.$$

Local certificates

A set $\Delta \subset \Gamma$ is said to be full if

$$(G_\Delta \cap \mathrm{Aut}_G(s))^\varphi \geq \mathrm{Alt}(\Delta),$$

where $G_\Delta = ((G^\varphi)_{\{\Delta\}})^{\varphi^{-1}}$.

1. The non-fullness certificate is a group $M = M(\Delta) \leq \mathrm{Sym}(\Delta)$ such that

$$M \ngeq \mathrm{Alt}(\Delta) \quad \text{and} \quad \mathrm{Aut}_G(s)^\varphi \leq M.$$

2. The fullness certificate is a group $K = K(\Delta) \leq \mathrm{Sym}(\Omega)$ such that

$$K \leq \mathrm{Aut}_G(s) \quad \text{and} \quad \Delta^{K^\varphi} = \Delta \quad \text{and} \quad (K^\varphi)^\Delta \geq \mathrm{Alt}(\Delta).$$

Local certificates theorem

Theorem

Let $\Delta \subset \Gamma$, $k = |\Delta|$, and

$$\max\{8, 2 + \log_2 n\} < k \leq c|\Gamma|.$$

Then by making at most $k!\, n^2$ calls to String Isomorphism problems on domains of size $\leq n/k$ and performing $k! n^{O(1)}$ computation, one can decide whether or not Δ is full and

Local certificates theorem

Theorem

Let $\Delta \subset \Gamma$, $k = |\Delta|$, and

$$\max\{8, 2 + \log_2 n\} < k \leq c|\Gamma|.$$

Then by making at most $k!\, n^2$ calls to String Isomorphism problems on domains of size $\leq n/k$ and performing $k! n^{O(1)}$ computation, one can decide whether or not Δ is full and

- if Δ is full, then find a certificate $K(\Delta) \leq \mathrm{Aut}_G(s)$ of fullness of Δ,
- if Δ is not full, then find a certificate $M(\Delta) \leq \mathrm{Sym}(\Delta)$ of non-fullness.

Local certificates theorem

Theorem

Let $\Delta \subset \Gamma$, $k = |\Delta|$, and

$$\max\{8, 2 + \log_2 n\} < k \leq c|\Gamma|.$$

Then by making at most $k!\,n^2$ calls to String Isomorphism problems on domains of size $\leq n/k$ and performing $k!n^{O(1)}$ computation, one can decide whether or not Δ is full and

- if Δ is full, then find a certificate $K(\Delta) \leq \mathrm{Aut}_G(s)$ of fullness of Δ,
- if Δ is not full, then find a certificate $M(\Delta) \leq \mathrm{Sym}(\Delta)$ of non-fullness.

Moreover, the families $\{(\Delta, K(\Delta)) : \Delta \text{ is full}\}$ and $\{(\Delta, M(\Delta)) : \Delta \text{ is not full}\}$ are canonical.