Isomorphism testing problems: in light of Babai's graph isomorphism breakthrough

Youming Qiao

Centre for Quantum Software and Information
University of Technology Sydney

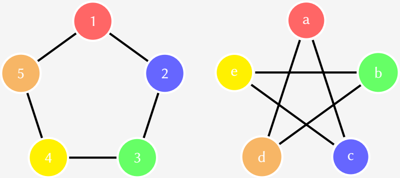
3rd May 2019@ Symmetry in Newcastle, U Newcastle

Based on joint works with Yinan Li and Gábor Ivanyos.

CENTRE FOR QUANTUM SOFTWARE AND INFORMATION

Table of Contents

Graph Isomorphism (GraphIso)



Graph isomorphism problem

Given two graphs $G = (V, E)$ and $H = (U, F)$, decide whether $\exists$ a bijective map $f : V \to U$, such that $v \sim v'$ if and only if $f(v) \sim f(v')$.

A partial review of some results on GrI

1960's	Studied in chemistry; combinatorial methods.
1970's	Received considerable attention; Babai's group-theoretic approach; McKay's NAUTY.
Early 1980's	Luks' algorithm for graphs with bounded degrees; $\exp(\tilde{O}(\sqrt{n}))$-time algorithm by Babai and Luks.
Late 1980's	Unlikely to be NP-complete via interactive proofs.
⋮	RELATIVELY QUIET PERIOD.
2010's	McKay and Piperno, NAUTY and TRACES; Babai's quasipolynomial-time algorithm.

A partial review of some results on GrI

1960's	Studied in chemistry; combinatorial methods.
1970's	Received considerable attention; Babai's group-theoretic approach; McKay's NAUTY.
Early 1980's	Luks' algorithm for graphs with bounded degrees; $\exp(\tilde{O}(\sqrt{n}))$-time algorithm by Babai and Luks.
Late 1980's	Unlikely to be NP-complete via interactive proofs.
⋮	RELATIVELY QUIET PERIOD.
2010's	McKay and Piperno, NAUTY and TRACES; Babai's quasipolynomial-time algorithm.

Theorem (Babai, 2015; cf. arXiv 1710.04574 by Helfgott)

There exists an algorithm that decides whether two graphs of size n are isomorphic in time $\exp(O((\log n)^3))$.

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013.

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013. ✓

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013. ✓

Average-case algorithms An algorithm that works for (G, H) where G is a random graph.

- An efficient algorithm by Babai-Erdős-Selkow in 1980, with follow-up improvements by Karp, Lipton, and Babai-Kučera.

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013. ✓

Average-case algorithms An algorithm that works for (G, H) where G is a random graph.

- An efficient algorithm by Babai-Erdős-Selkow in 1980, with follow-up improvements by Karp, Lipton, and Babai-Kučera. ✓

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013. ✓

Average-case algorithms An algorithm that works for (G, H) where G is a random graph.

- An efficient algorithm by Babai-Erdős-Selkow in 1980, with follow-up improvements by Karp, Lipton, and Babai-Kučera. ✓

Worst-case algorithms An algorithm with rigorous analysis on the running time.

- Poly-time algorithm for graphs of constant degrees [Luks, 1982].
- $\exp(\tilde{O}(\sqrt{n}))$ for general graphs [Babai-Luks, 1983].
- $\exp((\log n)^3)$-time for general graphs by Babai in 2015.

Three types of algorithms for GraphIso

Practical algorithms Implemented software that is effective in practice but with no provable guarantees.

- Nauty by McKay in 1978; Nauty and Traces by McKay and Piperno in 2013. ✓

Average-case algorithms An algorithm that works for (G, H) where G is a random graph.

- An efficient algorithm by Babai-Erdős-Selkow in 1980, with follow-up improvements by Karp, Lipton, and Babai-Kučera. ✓

Worst-case algorithms An algorithm with rigorous analysis on the running time.

- Poly-time algorithm for graphs of constant degrees [Luks, 1982].
- $\exp(\tilde{O}(\sqrt{n}))$ for general graphs [Babai-Luks, 1983].
- $\exp((\log n)^3)$-time for general graphs by Babai in 2015. ✓

Isomorphism testing in light of Babai's breakthrough

- Babai's quasi-polytime algorithm is a cultimation of the journal of graph isomorphism.
  - One cloud: how about improving to polynomial-time?
- It is perhaps time to look further at some other isomorphism testing problems.

Isomorphism testing in light of Babai's breakthrough

- Babai's quasi-polytime algorithm is a cultimation of the journal of graph isomorphism.
  - One cloud: how about improving to polynomial-time?
- It is perhaps time to look further at some other isomorphism testing problems.

(Finite) Group isomorphism problem

"Given" two finite groups $(G, \circ)$ and $(H, *)$, decide whether there exists a bijective map $f : G \to H$, such that $\forall g, g' \in G$, $f(g \circ g') = f(g) * f(g')$.

Some remarks on GroupIso

- GroupIso has been studied in computational group theory (CGT) and theoretical computer science (TCS) communities.
- $\mathfrak{B}(p, 2)$ denotes the class of p-groups of class 2 and exponent p.
- In the following, we assume n represents the group order.

[1] Groups are stored in a data structure with polylogarithmic size.

Some remarks on GroupIso

- GroupIso has been studied in computational group theory (CGT) and theoretical computer science (TCS) communities.
- $\mathfrak{B}(p,2)$ denotes the class of p-groups of class 2 and exponent p.
- In the following, we assume n represents the group order.

1960's	Studied in CGT with succinct representations[1]; the $n^{\log n+O(1)}$-time algorithm.
1970's	Studied in TCS with Caylay table representations, which reduces to graph isomorphism (GraphIso).
	Realized that $\mathfrak{B}(p,2)$ forms a bottleneck; GraphIso reduces to GroupIso with succinct representations.
1980's to 2010's	Progress in CGT by Cannon, Holt, O'Brien, and others.
	$n^{\frac{1}{4}n+o(\log n)}$-time by Rosenbaum; dynamic programming technique; multilinear algebra perspective; progress on $\mathfrak{B}(p,2)$.

[1]Groups are stored in a data structure with polylogarithmic size.

p-groups of class 2 and exponent p

In the following, p is an odd prime.

- It has been widely regarded that $\mathfrak{B}(p,2)$ is a bottleneck for GroupIso.
- For $G \in \mathfrak{B}(p,2)$, the commutator map gives an alternating bilinear map from $G/[G,G] \times G/[G,G]$ to $[G,G]$.
- Baer's correspondence tells us testing isomorphism of $\mathfrak{B}(p,2)$ is equivalent to the following linear algebraic problem.

p-groups of class 2 and exponent p

In the following, p is an odd prime.

- It has been widely regarded that $\mathfrak{B}(p, 2)$ is a bottleneck for GroupIso.
- For $G \in \mathfrak{B}(p, 2)$, the commutator map gives an alternating bilinear map from $G/[G, G] \times G/[G, G]$ to $[G, G]$.
- Baer's correspondence tells us testing isomorphism of $\mathfrak{B}(p, 2)$ is equivalent to the following linear algebraic problem.

Pseudo-isometry of alternating bilinear maps

Let V, U be linear spaces over $\mathbb{F}_p$. Given two alternating bilinear maps $\alpha, \beta : V \times V \to U$, decide whether $\exists S \in \mathrm{GL}(V), T \in \mathrm{GL}(U)$, such that $T \circ \alpha \circ S = \beta$.

This problem makes sense for any (computable) field; we stick to $\mathbb{F}_p$ and $\mathbb{F}_q$ in this talk.

An linear algebraic analogue of GraphIso

- Let $\Lambda(n,p)$ be the linear space of $n \times n$ alternating matrices over $\mathbb{F}_p$.
- Subspaces of $\Lambda(n,p)$ are called alternating matrix spaces.

We then have an even more concrete formulation.

Alternating matrix space isometry problem (AltSpIso)

Let $A_i, B_i \in \Lambda(n,p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n,p)$, such that $\langle S^t A_1 S, \ldots, S^t A_m S\rangle = \langle B_1, \ldots, B_m\rangle$.

An linear algebraic analogue of GraphIso

- Let $\Lambda(n, p)$ be the linear space of $n \times n$ alternating matrices over $\mathbb{F}_p$.
- Subspaces of $\Lambda(n, p)$ are called alternating matrix spaces.

We then have an even more concrete formulation.

Alternating matrix space isometry problem (AltSpIso)

Let $A_i, B_i \in \Lambda(n, p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n, p)$, such that $\langle S^t A_1 S, \ldots, S^t A_m S\rangle = \langle B_1, \ldots, B_m\rangle$.

$$\text{E.g. } \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \left\langle \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \right\rangle \begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \left\langle \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 3 \\ 0 & -3 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 2 \\ 1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \right\rangle$$

Some facts about AltSpIso

What do we hope to achieve for AltSpIso?

- Brute-force algorithm: $p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$.
- Poly-time algorithm: $\mathrm{poly}(n, m, \log p)$ – polynomial in the finite matrix group model.
- A quite moderate goal: $p^{O(n+m)}$ – polynomial in the group order.
- In $\mathrm{NP} \cap \mathrm{coAM}$, so unlikely to be NP-complete.

Some facts about AltSpIso

What do we hope to achieve for AltSpIso?

- Brute-force algorithm: $p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$.
- Poly-time algorithm: $\mathrm{poly}(n, m, \log p)$ – polynomial in the finite matrix group model.
- A quite moderate goal: $p^{O(n+m)}$ – polynomial in the group order.
- In $\mathrm{NP} \cap \mathrm{coAM}$, so unlikely to be NP-complete.

The relationship between AltSpIso and GraphIso:

- GraphIso reduces to solving AltSpIso in $\mathrm{poly}(n, m, \log p)$[Folklore].
- Solving AltSpIso in time $p^{O(n+m)}$ reduces to solving GraphIso on graphs of size $p^{O(n+m)}$ [Hedrlín-Pultr].
- The current techniques for GraphIso seem not helpful for AltSpIso.
- Achieving a $p^{O(n+m)}$-time algorithm would remove a key bottleneck for getting a poly-time algorithm for GraphIso.

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity		
Average-case Complexity		
Random Model		
Practical		
Group-Theoretic Technique		
Combinatorial Technique		

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity		
Random Model		
Practical		
Group-Theoretic Technique		
Combinatorial Technique		

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity		
Random Model		
Practical		
Group-Theoretic Technique		
Combinatorial Technique		

Seems not helpful to AltSpIso:

- $q^{n^2} \cdot \mathrm{poly}(n, m, \log q)$ is **quasipolynomial in** $q^{O(n+m)}$;
- Not helpful to improve GroupIso [Babai '16, Le Gall-Rosenbaum '16].

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n,p)$
Symmetry	S_n	$\mathrm{GL}(n,p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity	linear time in $\mathrm{ER}(n,m)$ [Babai-Erdős-Selkow '80]	?
Random Model		
Practical		
Group-Theoretic Technique		
Combinatorial Technique		

For most G, test isomorphism with H in linear time [Babai-Erdős-Selkow '80].
Follow-up improved by [Lipton '78], [Karp '79] and [Babai-Kučera '79].

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity	linear time in $\mathrm{ER}(n, m)$ [Babai-Erdős-Selkow '80]	?
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	?
Practical		
Group-Theoretic Technique		
Combinatorial Technique		

Erdős-Rényi model: Randomly choose a graph with n vertices and m edges with probability $1/\binom{\binom{n}{2}}{m}$.

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity	linear time in $\mathrm{ER}(n, m)$ [Babai-Erdős-Selkow '80]	?
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	?
Practical	Nauty & Traces[1]	Magma & Gap[2]
Group-Theoretic Technique		
Combinatorial Technique		

[1]Developed by McKay & Piperno.

[2]We thank James B. Wilson for communicating his hands-on experience to us.

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, p)$
Symmetry	S_n	$\mathrm{GL}(n, p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity	linear time in $\mathrm{ER}(n, m)$ [Babai-Erdős-Selkow '80]	?
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	?
Practical	Nauty & Traces	Magma & Gap
Group-Theoretic Technique	Permutation group algorithm	Matrix group algorithm
Combinatorial Technique		

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n,p)$
Symmetry	S_n	$\mathrm{GL}(n,p)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$p^{n^2} \cdot \mathrm{poly}(n, m, \log p)$
Average-case Complexity	linear time in $\mathrm{ER}(n,m)$ [Babai-Erdős-Selkow '80]	?
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	?
Practical	Nauty & Traces	Magma & Gap
Group-Theoretic Technique	Permutation group algorithm	Matrix group algorithm
Combinatorial Technique	Individualization and refinement	?

AltSpIso and other isomorphism testing problems

Some other isomorphism testing problems have been studied.

- Linear code equivalence: whether two linear subspaces are the same up to permuting coordinates. Studied in coding theory since 1990's.
- Polynomial map isomorphism: whether two polynomial maps from $\mathbb{F}_q^n \to \mathbb{F}_q^m$, defined by quadratic polynomials, are the same up to $\mathrm{GL}(n,q) \times \mathrm{GL}(m,q)$. Studied in cryptography since 1990's.
- Cubic form equivalence: whether two cubic forms in $\mathbb{F}_q[x_1, \dots, x_n]$ are the same to $\mathrm{GL}(n,q)$. Studied in TCS in early 2000's.

AltSpIso and other isomorphism testing problems

Some other isomorphism testing problems have been studied.

- Linear code equivalence: whether two linear subspaces are the same up to permuting coordinates. Studied in coding theory since 1990's.
- Polynomial map isomorphism: whether two polynomial maps from $\mathbb{F}_q^n \to \mathbb{F}_q^m$, defined by quadratic polynomials, are the same up to $\mathrm{GL}(n, q) \times \mathrm{GL}(m, q)$. Studied in cryptography since 1990's.
- Cubic form equivalence: whether two cubic forms in $\mathbb{F}_q[x_1, \ldots, x_n]$ are the same to $\mathrm{GL}(n, q)$. Studied in TCS in early 2000's.

Theorem (Grochow-Q, 2019)

All these problems reduce to AltSpIso.

This suggests that AltSpIso captures the difficulties of all these problems. Perhaps it is even difficult enough to be used for cryptographic purposes.

Two concrete results on AltSpIso

- In the following, I will introduce two concrete results on AltSpIso, based on joint works with Gábor Ivanyos and Yinan Li.
- These are algorithms with rigorous (worst-case or average-case) analyses.
- Thanks to the great works of Peter Brooksbank and James Wilson, they are also implemented in Magma, and shown to be helpful for practical computations.
- One algorithm heavily depends on the $*$-algebra technique first developed by James Wilson.
- AltSpIso is too difficult in both theoretical and practical senses, so an interaction between CGT and TCS will be helpful.

A similar problem

Recall that the key problem is:

Alternating matrix space isometry problem (AltSpIso)

Let $A_i, B_i \in \Lambda(n, p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n, p)$, such that $\langle S^t A_1 S, \ldots, S^t A_m S \rangle = \langle B_1, \ldots, B_m \rangle$.

A similar problem

Recall that the key problem is:

Alternating matrix space isometry problem (AltSpIso)

Let $A_i, B_i \in \Lambda(n, p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n, p)$, such that $\langle S^t A_1 S, \ldots, S^t A_m S \rangle = \langle B_1, \ldots, B_m \rangle$.

How about the following similar problem?

Alternating matrix *tuple* isometry problem (AltTpIso)

Let $A_i, B_i \in \Lambda(n, p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n, p)$, such that $(S^t A_1 S, \ldots, S^t A_m S) = (B_1, \ldots, B_m)$.

A similar problem

Recall that the key problem is:

Alternating matrix space isometry problem (AltSpIso)

Let $A_i, B_i \in \Lambda(n,p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n,p)$, such that $\langle S^t A_1 S, \ldots, S^t A_m S \rangle = \langle B_1, \ldots, B_m \rangle$.

How about the following similar problem?

Alternating matrix *tuple* isometry problem (AltTpIso)

Let $A_i, B_i \in \Lambda(n,p)$, $i = 1, \ldots, m$. Decide whether there exists $S \in \mathrm{GL}(n,p)$, such that $(S^t A_1 S, \ldots, S^t A_m S) = (B_1, \ldots, B_m)$.

- This problem was thought to be difficult in cryptography in the 1990's.
- A poly-time algorithm for AltTpIso implies a $p^{m^2} \cdot \mathrm{poly}(n, m, \log p)$-time algorithm for AltSpIso.

AltTpIso can be efficiently solved

Theorem (Ivanyos-Q)

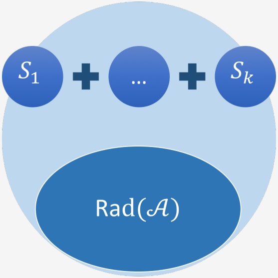
There exists a randomized polynomial-time algorithm for AltTpIso.

- One key ingredient is the $*$-algebra technique, first introduced for computing with p-groups by J. B. Wilson.
- The other key ingredient is the solution to the module isomorphism problem.
- Overall, the algorithm can be viewed as a reduction from alternating matrix tuples, to *single* classical forms.

Structure of algebras

Let $\mathcal{A}$ be a finite dimensional associative algebra over $\mathbb{F}$.

- $\mathrm{Rad}(\mathcal{A})$: the radical, e.g. the largest nilpotent ideal.
- $\mathcal{A}/\mathrm{Rad}(\mathcal{A})$: semisimple, that is, isomorphic to a direct sum of simple algebras.
- $S_i \cong M(n_i, \mathbb{F}_i)$: a full matrix algebra over $\mathbb{F}_i$, an extension field of $\mathbb{F}$.



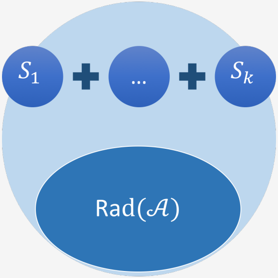
Theorem ([Rónyai 90])

Over $\mathbb{F}_q$, the above structural information of $\mathcal{A}$ can be computed in randomized polynomial time.

Structure of $*$-algebras

Let $* : \mathcal{A} \to \mathcal{A}$ be an involution, e.g. an anti-automorphism such that $\forall a \in \mathcal{A}$, $(a^*)^* = a$.

- $\mathrm{Rad}(\mathcal{A})$ is invariant under $*$: $*$ induces an involution on $\mathcal{A}/\mathrm{Rad}(\mathcal{A})$.
- Recall that $S_i \cong M(n_i, \mathbb{F}_i)$.
  1. $S_i^* = S_j$, $i \neq j$. Then $S_i \cong S_j$, and $(a, b)^* = (b, a)$, $(a, b) \in S_i \oplus S_j$.
  2. $S_i^* = S_i$. There is a classical form $F \in M(n_i, \mathbb{F}_i)$, such that $A^* = F^{-1} A^t F$ for $A \in S_i$.



Theorem ([Wilson 09])

Over $\mathbb{F}_q$, the above structural information can be computed in randomized polynomial time.

Module isomorphism problem

Module isomorphism problem

Given $n \times n$ matrices $A_1, \ldots, A_m$, and $B_1, \ldots, B_m$, decide whether there exist an invertible C, such that for all $i \in [m]$, $CA_i = B_iC$.

Theorem ([Chistov-Ivanyos-Karpinski 97, Brooksbank-Luks 08])

There are deterministic efficient algorithms for the module isomorphism problem over any field.

Module isomorphism problem

Module isomorphism problem

Given $n \times n$ matrices $A_1, \ldots, A_m$, and $B_1, \ldots, B_m$, decide whether there exist an invertible C, such that for all $i \in [m]$, $CA_i = B_iC$.

Theorem ([Chistov-Ivanyos-Karpinski 97, Brooksbank-Luks 08])

There are deterministic efficient algorithms for the module isomorphism problem over any field.

- It allows an easy linearisation, i.e. set up $XA_i = B_iX$, and search for an invertible matrix in the solution space.
- Can be solved very efficiently in practice by MeatAxe.
- To the contrary, AltTpIso does not allow for such a straightforward linearisation.

Isometry testing algorithm outline

Given $A_1, \ldots, A_m, B_1, \ldots, B_m$, $n \times n$ alternating matrices over $\mathbb{F}$, do the following:

1. Compute invertible D, E, such that $\forall i$, $D^t A_i = B_i E$, by reducing to module isomorphism problem.

Isometry testing algorithm outline

Given $A_1, \ldots, A_m, B_1, \ldots, B_m, n \times n$ alternating matrices over $\mathbb{F}$, do the following:

1. Compute invertible D, E, such that $\forall i$, $D^t A_i = B_i E$, by reducing to module isomorphism problem.
2. Compute a linear basis for the algebra
$$\mathcal{A} = \{F : \exists! F', \forall i, F^t B_i = B_i F'\} \subseteq M(n, \mathbb{F}).$$
   - $\mathcal{A}$ is a $*$-algebra: $F^* = F'$, because of the alternating condition.

Isometry testing algorithm outline

Given $A_1, \ldots, A_m, B_1, \ldots, B_m$, $n \times n$ alternating matrices over $\mathbb{F}$, do the following:

1. Compute invertible D, E, such that $\forall i$, $D^t A_i = B_i E$, by reducing to module isomorphism problem.
2. Compute a linear basis for the algebra
$$\mathcal{A} = \{F : \exists! F', \forall i, F^t B_i = B_i F'\} \subseteq M(n, \mathbb{F}).$$
   - $\mathcal{A}$ is a $*$-algebra: $F^* = F'$, because of the alternating condition.
3. $F = D^{-1} E^{-1} \in \mathcal{A}$, $F^* = F$. The problem then boils down to compute $X \in \mathcal{A}$, such that $X^* X = F$.
   1. Reduce to semisimple $\mathcal{A}$.
   2. Reduce to simple $S_i \cong M(n_i, \mathbb{F}_i)$ and $S_i^* = S_i$.

Isometry testing algorithm outline

Given $A_1, \ldots, A_m, B_1, \ldots, B_m$, $n \times n$ alternating matrices over $\mathbb{F}$, do the following:

1. Compute invertible D, E, such that $\forall i$, $D^t A_i = B_i E$, by reducing to module isomorphism problem.
2. Compute a linear basis for the algebra
   $$\mathcal{A} = \{F : \exists! F', \forall i, F^t B_i = B_i F'\} \subseteq M(n, \mathbb{F}).$$
   - $\mathcal{A}$ is a $*$-algebra: $F^* = F'$, because of the alternating condition.
3. $F = D^{-1}E^{-1} \in \mathcal{A}$, $F^* = F$. The problem then boils down to compute $X \in \mathcal{A}$, such that $X^* X = F$.
   1. Reduce to semisimple $\mathcal{A}$.
   2. Reduce to simple $S_i \cong M(n_i, \mathbb{F}_i)$ and $S_i^* = S_i$.
      - Let F_i be the classical form from the action of $*$ on S_i. The question then becomes whether two *single* forms FF_i and F_i are isometric.

GraphIso and AltSpIso

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, q)$
Symmetry	S_n	$\mathrm{GL}(n, q)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$q^{n^2} \cdot \mathrm{poly}(n, m, \log q)$
Average-case Complexity	linear time in $\mathrm{ER}(n, m)$ [Babai-Erdős-Selkow '80]	?
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	?
Practical	Nauty & Traces	Magma & Gap
Group-Theoretic Technique	Permutation group algorithm	Matrix group algorithm
Combinatorial Technique	Individualization and refinement	?

An attempt to address the challenges [Li-Q]

	GraphIso	AltSpIso
Objects	$G, H \subseteq \Lambda_n$	$\mathcal{G}, \mathcal{H} \leq \Lambda(n, q)$
Symmetry	S_n	$\mathrm{GL}(n, q)$
Worst-case Complexity	$\exp((\log n)^{O(1)})$ [Babai '16]	$q^{n^2} \cdot \mathrm{poly}(n, m, \log q)$
Average-case Complexity	linear time in $\mathrm{ER}(n, m)$ [Babai-Erdős-Selkow '80]	$q^{O(n)}$ **in LinER**(n, m, q)
Random Model	Erdős-Rényi model [Erdős-Rényi '59]	**Linear algebraic analogue of Erdős-Rényi model**
Practical	Nauty & Traces	Magma & Gap
Group-Theoretic Technique	Permutation group algorithm	Matrix group algorithm
Combinatorial Technique	Individualization and refinement	**Linear-algebraic analogue of individualization and refinement**

From graphs to alternating matrix spaces

- Vector $v \Longleftarrow$ Vertex i.
- Alternating matrix $H \Longleftarrow$ Edge $\{i, j\}$.
- Alternating matrix space $\mathcal{G} \Longleftarrow$ Graph G.

From graphs to alternating matrix spaces

- Vector $v \Longleftarrow$ Vertex i.
- Alternating matrix $H \Longleftarrow$ Edge $\{i, j\}$.
- Alternating matrix space $\mathcal{G} \Longleftarrow$ Graph G.

- The Erdős-Rényi Model (ER(n, m)): Randomly choose a graph with vertex set $[n]$ and m edges. Each graph appears with probability $1/\binom{\binom{n}{2}}{m}$.
- Linear algebraic analogue of the Erdős-Rényi Model (LinER(n, m, q)): Randomly choose a dim-m alternating matrix space $\mathcal{G} \leq \Lambda(n, q)$ with probability $1/\begin{bmatrix}\binom{n}{2} \\ m\end{bmatrix}_q$.

From graphs to alternating matrix spaces

- Vector $v \Longleftarrow$ Vertex i.
- Alternating matrix $H \Longleftarrow$ Edge $\{i, j\}$.
- Alternating matrix space $\mathcal{G} \Longleftarrow$ Graph G.

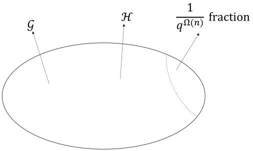
- The Erdős-Rényi Model (ER(n, m)): Randomly choose a graph with vertex set $[n]$ and m edges. Each graph appears with probability $1/\binom{\binom{n}{2}}{m}$.
- Linear algebraic analogue of the Erdős-Rényi Model (LinER(n, m, q)): Randomly choose a dim-m alternating matrix space $\mathcal{G} \leq \Lambda(n, q)$ with probability $1/\begin{bmatrix}\binom{n}{2} \\ m\end{bmatrix}_q$.

Previous works with a similar strategy:

- Linear algebraic analogue of the perfect matching problem on bipartite graphs [Garg-Gurvits-Oliveira-Wigderson '16, Ivanyos-Q-Subrahmanyam '17].
- Zero-error capacity of quantum channels $\Rightarrow$ Non-commutative graph [Duan-Severini-Winter '13].

AltSpIso in the LinER(n, m) setting

Theorem (Li-Q)



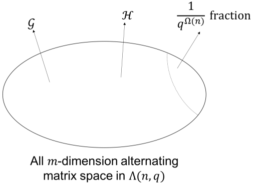
Let $m = cn$ for some constant c.

For most $\mathcal{G} \in$ LinER(n, m, q) (all but $\frac{1}{q^{\Omega(n)}}$ fraction),

Test isometry with any $\mathcal{H} \leq \Lambda(n, q)$ in time $q^{O(n)}$.

AltSpIso in the LinER(n, m) setting

Theorem (Li-Q)



Let $m = cn$ for some constant c.

For most $\mathcal{G} \in$ LinER(n, m, q) (all but $\frac{1}{q^{\Omega(n)}}$ fraction),

Test isometry with any $\mathcal{H} \leq \Lambda(n, q)$ in time $q^{O(n)}$.

Why $m = cn$? $(m \leq \binom{n}{2})$

- For $m = \Omega(n^2)$, the brute-force algorithm runs in time $q^{O(n+m)}$.
- For $m = O(1)$, AltSpIso can be solved in randomized $\mathrm{poly}(n, m, \log q)$ by the last result.

Individualisation and Refinement in GraphIso

Aim: For most graphs G, $|\mathrm{Iso}(G, H)| \leq |\mathrm{Aut}(G)| \leq n^{O(\log n)}$ [BES80].

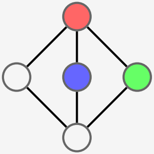
Individualisation and Refinement in GraphIso

Aim: For most graphs G, $|\mathrm{Iso}(G, H)| \leq |\mathrm{Aut}(G)| \leq n^{O(\log n)}$ [BES80].
View $\sigma \in S_n$ as bijective map $\sigma : [n] \to [n]$

k-individualization:

Fix the image of $1, \ldots, k$.

Enumeration cost n^k.



[1]When $k = \lceil 3 \log n \rceil$, most graphs satisfy this property.

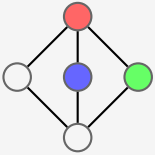
Individualisation and Refinement in GraphIso

Aim: For most graphs G, $|\mathrm{Iso}(G,H)| \leq |\mathrm{Aut}(G)| \leq n^{O(\log n)}$ [BES80].
View $\sigma \in S_n$ as bijective map $\sigma : [n] \to [n]$

k-individualization:

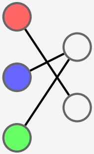
Fix the image of $1, \ldots, k$.

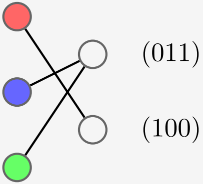
Enumeration cost n^k.



Refinement: Focus on the induced Bipartite Graph: $\forall\, j \in [n] \setminus [k]$, the adjacency relation with $[k]$ are distinct[1].

At most one way to extend σ to automorphism.



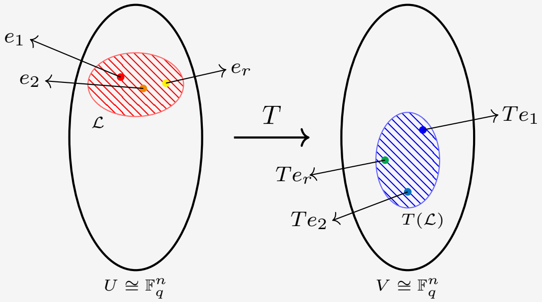


[1]When $k = \lceil 3 \log n \rceil$, most graphs satisfy this property.

The Linear Algebraic Analogue of Individualization

Recall: vertex $i \Longrightarrow$ vector v

Bij. Map	$\sigma \in S_n$	$T \in \mathrm{GL}(n, q)$
Ind.	Fix the image of $1, \ldots, k$	Fix the image, $\mathcal{L}$, of $e_1, \ldots e_r$ [1]
Cost		

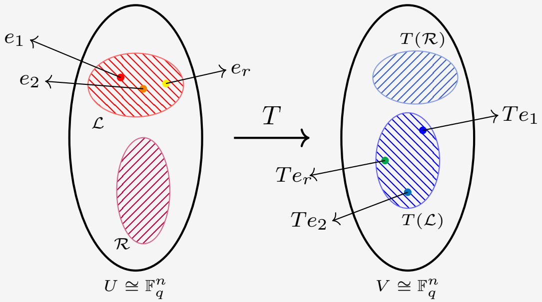


[1] r is a constant decided by m and n.

The Linear Algebraic Analogue of Individualization

Recall: vertex $i \Longrightarrow$ vector v

Bij. Map	$\sigma \in S_n$	$T \in \mathrm{GL}(n, q)$
Ind.	Fix the image of $1, \ldots, k$	Fix the image, $\mathcal{L}$, of $e_1, \ldots e_r$ [1] Fix a complement subspace $\mathcal{R}$ $\mathcal{L} \cap \mathcal{R} = \{0\}$, $\langle \mathcal{L}, \mathcal{R} \rangle = \mathbb{F}_q^n$.
Cost		

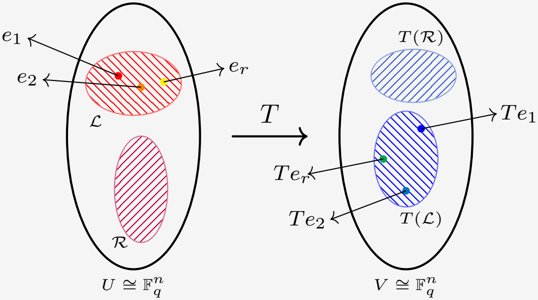


[1] r is a constant decided by m and n.

The Linear Algebraic Analogue of Individualization

Recall: vertex $i \Longrightarrow$ vector v

Bij. Map	$\sigma \in S_n$	$T \in \mathrm{GL}(n, q)$
Ind.	Fix the image of $1, \dots, k$	Fix the image, $\mathcal{L}$, of $e_1, \dots e_r$ [1] Fix a complement subspace $\mathcal{R}$ $\mathcal{L} \cap \mathcal{R} = \{0\}, \langle \mathcal{L}, \mathcal{R} \rangle = \mathbb{F}_q^n$.
Cost	n^k	$q^r \times q^{r(n-r)} = q^{O(n)}$

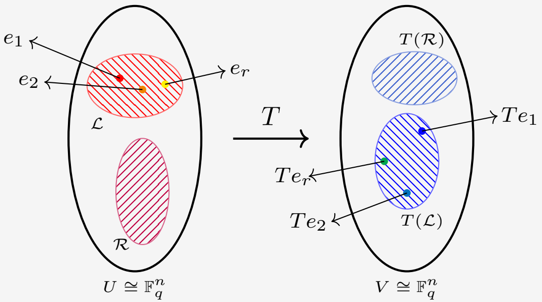


[1] r is a constant decided by m and n.

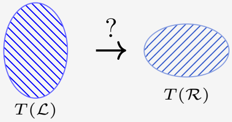
The Linear Algebraic Analogue of Individualization

Recall: vertex $i \Longrightarrow$ vector v

Bij. Map	$\sigma \in S_n$	$T \in \mathrm{GL}(n, q)$
Ind.	Fix the image of $1, \ldots, k$	Fix the image, $\mathcal{L}$, of $e_1, \ldots e_r$[1] Fix a complement subspace $\mathcal{R}$ $\mathcal{L} \cap \mathcal{R} = \{0\}$, $\langle \mathcal{L}, \mathcal{R} \rangle = \mathbb{F}_q^n$.
Cost	n^k	$q^r \times q^{r(n-r)} = q^{O(n)}$



"Induced Bipartite Graph"



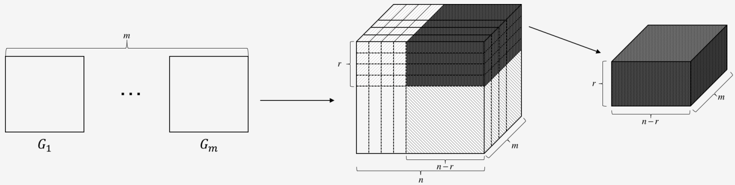
[1] r is a constant decided by m and n.

The Linear Algebraic Analogue of Individualization

Recall: vertex $i \Longrightarrow$ vector v

Bij. Map	$\sigma \in S_n$	$T \in \mathrm{GL}(n, q)$
Ind.	Fix the image of $1, \ldots, k$	Fix the image, $\mathcal{L}$, of $e_1, \ldots e_r$[1] Fix a complement subspace $\mathcal{R}$ $\mathcal{L} \cap \mathcal{R} = \{0\}, \langle \mathcal{L}, \mathcal{R} \rangle = \mathbb{F}_q^n$.
Cost	n^k	$q^r \times q^{r(n-r)} = q^{O(n)}$

The "Induced Bipartite Graph"



- Apply the chosen ind. to $\mathcal{G}$, representing its linear basis as a 3-tensor.
- Take the upper-right subtensor of size $r \times (n - r) \times m$
  $\Rightarrow$ "induced bipartite graph" $\mathcal{B}_{\mathcal{G}}$.

[1] r is a constant decided by m and n.

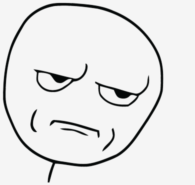
The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?

The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?

$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.



The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?
$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.

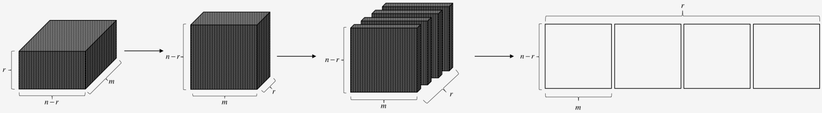
Aim: upper bound
$|\{P \in \mathrm{GL}(n, q) : \mathcal{B}_{\mathcal{G}} P = \mathcal{B}_{\mathcal{G}}\}|$.

The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?
$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.

Aim: upper bound
$|\{P \in \mathrm{GL}(n,q) : \mathcal{B}_\mathcal{G} P = \mathcal{B}_\mathcal{G}\}|$.

Flip $\mathcal{B}_\mathcal{G} \Rightarrow \mathcal{B}'_\mathcal{G} = \langle B_1, \dots, B_r \rangle \leq M((n-r) \times m, q)$.

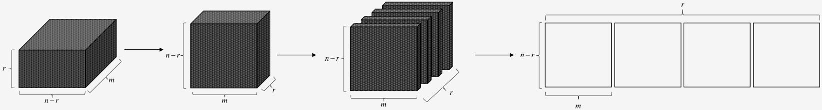


The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?
$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.

Aim: upper bound
$|\{P \in \mathrm{GL}(n,q) : \mathcal{B}_{\mathcal{G}} P = \mathcal{B}_{\mathcal{G}}\}|$.

Flip $\mathcal{B}_{\mathcal{G}} \Rightarrow \mathcal{B}'_{\mathcal{G}} = \langle B_1, \ldots, B_r \rangle \leq M((n-r) \times m, q)$.



$$\mathrm{Adj}(\mathcal{B}'_{\mathcal{G}}) = \{(A, D) \in M(n-r, q) \oplus M(m, q) : AB_i = B_i D \; \forall \, i = 1, \ldots, r\}.$$

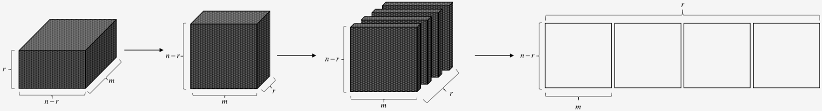
$$\mathcal{B}_{\mathcal{G}} P = \mathcal{B}_{\mathcal{G}} \;\Rightarrow\; \exists \, Q \text{ s.t. } (P, Q) \in \mathrm{Adj}(\mathcal{B}'_{\mathcal{G}})$$

The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?
$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.

Aim: upper bound
$|\{P \in \mathrm{GL}(n,q) : \mathcal{B}_\mathcal{G} P = \mathcal{B}_\mathcal{G}\}|$.

Flip $\mathcal{B}_\mathcal{G} \Rightarrow \mathcal{B}'_\mathcal{G} = \langle B_1, \dots, B_r \rangle \leq M((n-r) \times m, q)$.



$$\mathrm{Adj}(\mathcal{B}'_\mathcal{G}) = \{(A, D) \in M(n-r, q) \oplus M(m, q) : AB_i = B_i D \;\forall\, i = 1, \dots, r\}.$$

$$\mathcal{B}_\mathcal{G} P = \mathcal{B}_\mathcal{G} \;\Rightarrow\; \exists\, Q \text{ s.t. } (P, Q) \in \mathrm{Adj}(\mathcal{B}'_\mathcal{G})$$

Theorem: For most $\mathcal{G} \in$ LinER(n, m, q) ($1/q^{\Omega(n)}$ fraction), $|\mathrm{Adj}(\mathcal{B}'_\mathcal{G})| \leq q^{O(n)}$.

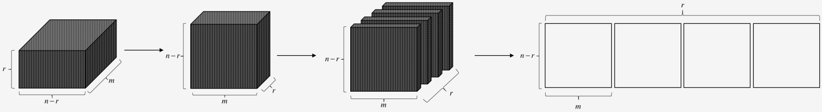
- The proof is inspired by the stable concept from geometric invariant theory.
- Plus basic algebraic results and probability calculations.

The Linear Algebraic Analogue of Refinement

Linear Algebraic "Labeling"?
$\#(v \in \mathcal{R}) = q^{(n-r)^2}$. Cost $q^{O(n^2)}$.

Aim: upper bound
$|\{P \in \mathrm{GL}(n, q) : \mathcal{B}_{\mathcal{G}} P = \mathcal{B}_{\mathcal{G}}\}|$.

Flip $\mathcal{B}_{\mathcal{G}} \Rightarrow \mathcal{B}'_{\mathcal{G}} = \langle B_1, \ldots, B_r \rangle \leq M((n-r) \times m, q)$.



$$\mathrm{Adj}(\mathcal{B}'_{\mathcal{G}}) = \{(A, D) \in M(n-r, q) \oplus M(m, q) : AB_i = B_i D \; \forall\, i = 1, \ldots, r\}.$$

$$\mathcal{B}_{\mathcal{G}} P = \mathcal{B}_{\mathcal{G}} \;\Rightarrow\; \exists\, Q \text{ s.t. } (P, Q) \in \mathrm{Adj}(\mathcal{B}'_{\mathcal{G}})$$

Theorem: For most $\mathcal{G} \in$ LinER(n, m, q) ($1/q^{\Omega(n)}$ fraction), $|\mathrm{Adj}(\mathcal{B}'_{\mathcal{G}})| \leq q^{O(n)}$.

- The proof is inspired by the stable concept from geometric invariant theory.
- Plus basic algebraic results and probability calculations.

For most $\mathcal{G} \in$ LinER(n, m, q) ($1/q^{\Omega(n)}$ fraction), $|\mathrm{Aut}(\mathcal{G})| \leq q^{O(n)}$.

1 The journey of graph isomorphism

2 Isomorphism testing after graph isomorphism

3 Concrete result one: tuples instead of spaces

4 Concrete result two: an average-case algorithm

5 Conclusion

Isomorphism testing and cryptography

AltSpIso seems to be much more difficult than the graph isomorphism problem. Given its (current) difficulty, one may hope to use it for cryptographic purposes [Brassard-Yung, Patarin].

- One-way function: for G action on S, $f_s(g) = g \cdot s$;
- Identification: Alice proves to Bob that this is the real Alice;
- Signature: Alice proves to Bob that the message is from Alice.

Isomorphism testing and cryptography

AltSpIso seems to be much more difficult than the graph isomorphism problem. Given its (current) difficulty, one may hope to use it for cryptographic purposes [Brassard-Yung, Patarin].

- One-way function: for G action on S, $f_s(g) = g \cdot s$;
- Identification: Alice proves to Bob that this is the real Alice;
- Signature: Alice proves to Bob that the message is from Alice.

Post-quantum security: the negative evidence for the hidden subgroup approach on graph isomorphism is the strongest known theoretical limitation on a class of quantum algorithms [Hallgren, Moore, ...].

Summary

A bit summary of the main messages:

- Despite Babai's recent progress on GraphIso, certain isomorphism testing problems still pose a great challenge for algorithm design.
- A key problem is AltSpIso, which captures the difficulties of many other isomorphism testing problems.
- The research into AltSpIso has lead a nice interaction among combinatorics, algebra, and algorithm design.
- Despite the progress, AltSpIso still stands as a difficult problem – both in theory and in practice.

A future direction?

Alternating matrix spaces as a linear algebraic analogue of graphs?

- Structures: perfect matchings [Lovász], cuts and connectivities [Li-Q], independent sets and vertex colorings [Bei-Chen-Guan-Q-Sun];
- Techniques: the augmenting path [Ivanyos-Karpinski-Q-Santha], individualisation and refinement [Li-Q];
- Questions: enumeration [BCGQS], probabilistic [LQ], and extremal [Turán, Buhler-Gupta-Harris].

Thank you for your attention!