Testing isomorphism of finite groups in time polynomial in the group order

Youming Qiao
University of Technology Sydney

Chuanqi Zhang
Monash University

November 20, 2026

https://group-iso.github.io/

Section 1

Background and Main Results

Finite Group Isomorphism

- GroupIso: Given the Cayley (multiplication) tables of two finite groups of order N, decide if they are isomorphic.
- Generation-enumeration algorithm: $N^{\log N + O(1)}$.
  [Tarjan; cf. Miller, STOC '78]
- Previous record: $N^{(\frac{1}{4}+o(1))\log N}$.
  [Rosenbaum '14; cf. Le Gall–Rosenbaum '16]
- An important case is to make progress on Tensor Isomorphism.

Finite Group Isomorphism

- GroupIso: Given the Cayley (multiplication) tables of two finite groups of order N, decide if they are isomorphic.
- Generation-enumeration algorithm: $N^{\log N + O(1)}$.
  [Tarjan; cf. Miller, STOC '78]
- Previous record: $N^{(\frac{1}{4}+o(1))\log N}$.
  [Rosenbaum '14; cf. Le Gall–Rosenbaum '16]
- An important case is to make progress on Tensor Isomorphism.

Finite Group Isomorphism

- GroupIso: Given the Cayley (multiplication) tables of two finite groups of order N, decide if they are isomorphic.
- Generation-enumeration algorithm: $N^{\log N + O(1)}$.
  [Tarjan; cf. Miller, STOC '78]
- Previous record: $N^{(\frac{1}{4}+o(1))\log N}$.
  [Rosenbaum '14; cf. Le Gall–Rosenbaum '16]
- An important case is to make progress on Tensor Isomorphism.

Finite Group Isomorphism

- GroupIso: Given the Cayley (multiplication) tables of two finite groups of order N, decide if they are isomorphic.
- Generation-enumeration algorithm: $N^{\log N + O(1)}$.
  [Tarjan; cf. Miller, STOC '78]
- Previous record: $N^{(\frac{1}{4}+o(1))\log N}$.
  [Rosenbaum '14; cf. Le Gall–Rosenbaum '16]

Theorem (Main result on group isomorphism)

Given two finite groups G and H of order N, there is an algorithm testing whether $G \cong H$ in time $\mathrm{poly}(N)$.

- An important case is to make progress on Tensor Isomorphism.

Finite Group Isomorphism

- GroupIso: Given the Cayley (multiplication) tables of two finite groups of order N, decide if they are isomorphic.
- Generation-enumeration algorithm: $N^{\log N + O(1)}$.
  [Tarjan; cf. Miller, STOC '78]
- Previous record: $N^{(\frac{1}{4}+o(1))\log N}$.
  [Rosenbaum '14; cf. Le Gall–Rosenbaum '16]

Theorem (Main result on group isomorphism)

Given two finite groups G and H of order N, there is an algorithm testing whether $G \cong H$ in time $\mathrm{poly}(N)$.

- An important case is to make progress on Tensor Isomorphism.

Tensor Isomorphism: problem formulation

Definition (3-Tensor Isomorphism, in matrix-tuple formulation)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times m, \mathbb{F})^n$ be two matrix tuples. Decide if there exist $L \in \mathrm{GL}(\ell, \mathbb{F})$, $R \in \mathrm{GL}(m, \mathbb{F})$, and $T = (t_{i,j}) \in \mathrm{GL}(n, \mathbb{F})$,

$$\forall i \in [n], \quad A_i = L\Big(\sum_{j \in [n]} t_{i,j} B_j\Big) R^{\mathrm{t}}.$$

Tensor Isomorphism: problem formulation

Definition (3-Tensor Isomorphism, in matrix-tuple formulation)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times m, \mathbb{F})^n$ be two matrix tuples. Decide if there exist $L \in \mathrm{GL}(\ell, \mathbb{F})$, $R \in \mathrm{GL}(m, \mathbb{F})$, and $T = (t_{i,j}) \in \mathrm{GL}(n, \mathbb{F})$,

$$\forall i \in [n], \quad A_i = L(\sum_{j \in [n]} t_{i,j} B_j) R^{\mathrm{t}}.$$

- (Coordinate-free formulation) Are $\mathbf{A}, \mathbf{B} \in \mathbb{F}^\ell \otimes \mathbb{F}^m \otimes \mathbb{F}^n$ in the same orbit under the natural action of $\mathrm{GL}(\ell, \mathbb{F}) \times \mathrm{GL}(m, \mathbb{F}) \times \mathrm{GL}(n, \mathbb{F})$?
- (Relation with GroupIso) 3-TensorIso over $\mathbb{F}_p$, p odd, in time $p^{O(\ell+m+n)}$ $=_{\mathrm{p}}$ GroupIso for p-groups of class 2 and exponent p. [Grochow–Q]
- (Tensor Isomorphism complexity class) 3-TensorIso is closely related to algebra isomorphism and polynomial isomorphism [Grochow–Q].

Tensor Isomorphism: problem formulation

Definition (3-Tensor Isomorphism, in matrix-tuple formulation)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times m, \mathbb{F})^n$ be two matrix tuples. Decide if there exist $L \in \mathrm{GL}(\ell, \mathbb{F})$, $R \in \mathrm{GL}(m, \mathbb{F})$, and $T = (t_{i,j}) \in \mathrm{GL}(n, \mathbb{F})$,

$$\forall i \in [n], \quad A_i = L(\sum_{j \in [n]} t_{i,j} B_j) R^{\mathrm{t}}.$$

- (Coordinate-free formulation) Are $\mathbf{A}, \mathbf{B} \in \mathbb{F}^\ell \otimes \mathbb{F}^m \otimes \mathbb{F}^n$ in the same orbit under the natural action of $\mathrm{GL}(\ell, \mathbb{F}) \times \mathrm{GL}(m, \mathbb{F}) \times \mathrm{GL}(n, \mathbb{F})$?
- (Relation with GroupIso) 3-TensorIso over $\mathbb{F}_p$, p odd, in time $p^{O(\ell+m+n)}$ $=_{\mathrm{p}}$ GroupIso for p-groups of class 2 and exponent p [†] [Grochow–Q]
- (Tensor Isomorphism complexity class) 3-TensorIso is closely related to algebra isomorphism and polynomial isomorphism [Grochow–Q].

[†] A group G with $|G| = p^s$, $[G, G] \leq Z(G)$, and $\forall g \in G$, $g^p = 1$.

Tensor Isomorphism: problem formulation

Definition (3-Tensor Isomorphism, in matrix-tuple formulation)

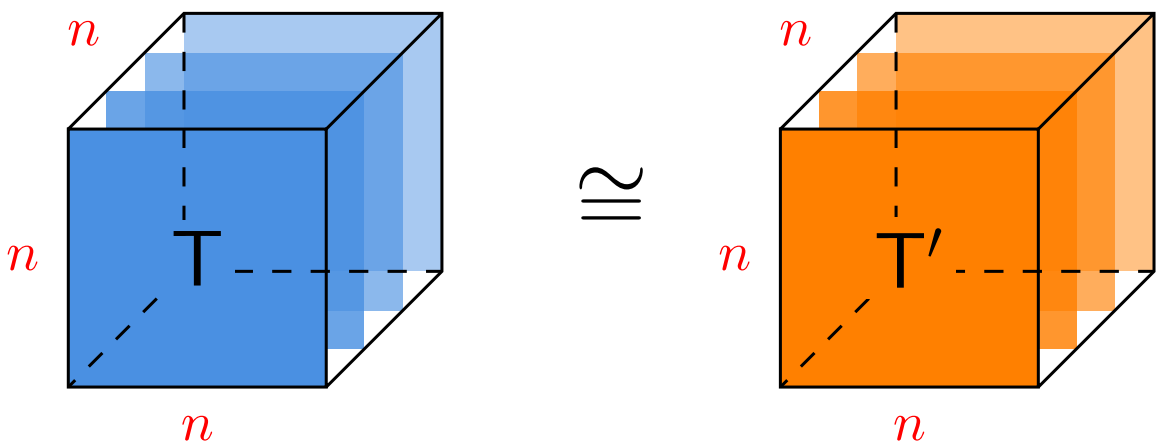
Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times m, \mathbb{F})^n$ be two matrix tuples. Decide if there exist $L \in \mathrm{GL}(\ell, \mathbb{F})$, $R \in \mathrm{GL}(m, \mathbb{F})$, and $T = (t_{i,j}) \in \mathrm{GL}(n, \mathbb{F})$,

$$\forall i \in [n], \quad A_i = L(\sum_{j \in [n]} t_{i,j} B_j) R^{\mathrm{t}}.$$

- (Coordinate-free formulation) Are $\mathbf{A}, \mathbf{B} \in \mathbb{F}^\ell \otimes \mathbb{F}^m \otimes \mathbb{F}^n$ in the same orbit under the natural action of $\mathrm{GL}(\ell, \mathbb{F}) \times \mathrm{GL}(m, \mathbb{F}) \times \mathrm{GL}(n, \mathbb{F})$?
- (Relation with GroupIso) 3-TensorIso over $\mathbb{F}_p$, p odd, in time $p^{O(\ell+m+n)}$ $=_{\mathrm{p}}$ GroupIso for p-groups of class 2 and exponent p [Grochow–Q]
- (Tensor Isomorphism complexity class) 3-TensorIso is closely related to algebra isomorphism and polynomial isomorphism [Grochow–Q].

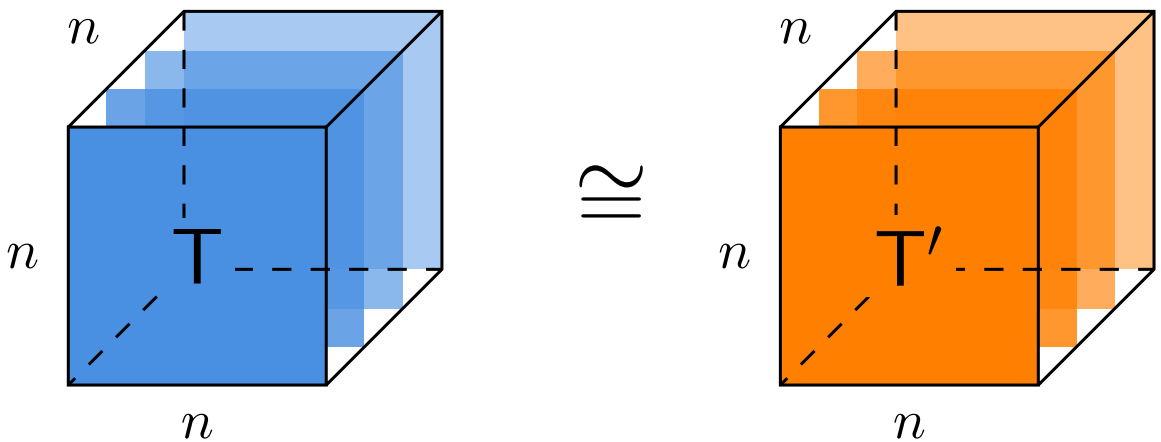
Tensor Isomorphism: main result

- W.l.o.g., let us consider $n \times n \times n$ tensors over $\mathbb{F}_q$.
- Brute-force running time: $q^{O(n^2)}$.
- Sun's breakthrough (STOC '23): $q^{O(n^{1.8} \log q)}$, q odd.
- [Ivanyos–Mendoza–Q–Sun–Z, FOCS '24]: $q^{\tilde{O}(n^{1.5})}$, q odd.



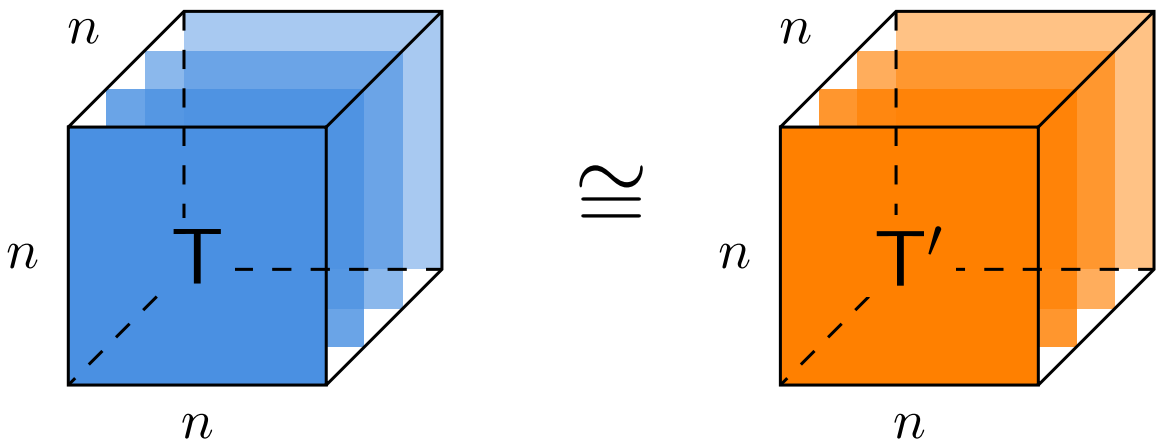
Tensor Isomorphism: main result

- W.l.o.g., let us consider $n \times n \times n$ tensors over $\mathbb{F}_q$.
- Brute-force running time: $q^{O(n^2)}$.
- Sun's breakthrough (STOC '23): $q^{O(n^{1.8}\log q)}$, q odd.
- [Ivanyos–Mendoza–Q–Sun–Z, FOCS '24]: $q^{\tilde{O}(n^{1.5})}$, q odd.



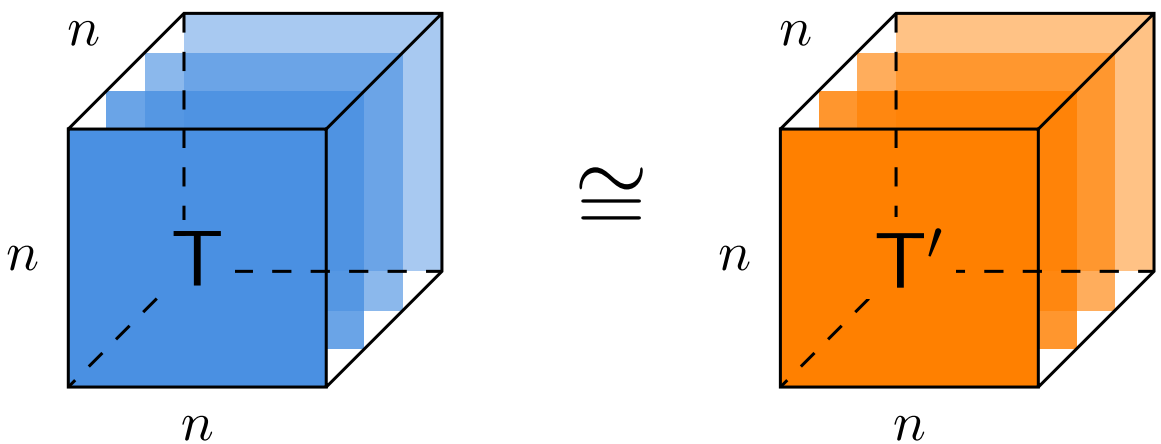
Tensor Isomorphism: main result

- W.l.o.g., let us consider $n \times n \times n$ tensors over $\mathbb{F}_q$.
- Brute-force running time: $q^{O(n^2)}$.
- Sun's breakthrough (STOC '23): $q^{O(n^{1.8} \log q)}$, q odd.
- [Ivanyos–Mendoza–Q–Sun–Z, FOCS '24]: $q^{\tilde{O}(n^{1.5})}$, q odd.



Tensor Isomorphism: main result

- W.l.o.g., let us consider $n \times n \times n$ tensors over $\mathbb{F}_q$.
- Brute-force running time: $q^{O(n^2)}$.
- Sun's breakthrough (STOC '23): $q^{O(n^{1.8} \log q)}$, q odd.
- [Ivanyos–Mendoza–Q–Sun–Z, FOCS '24]: $q^{\tilde{O}(n^{1.5})}$, q odd. †



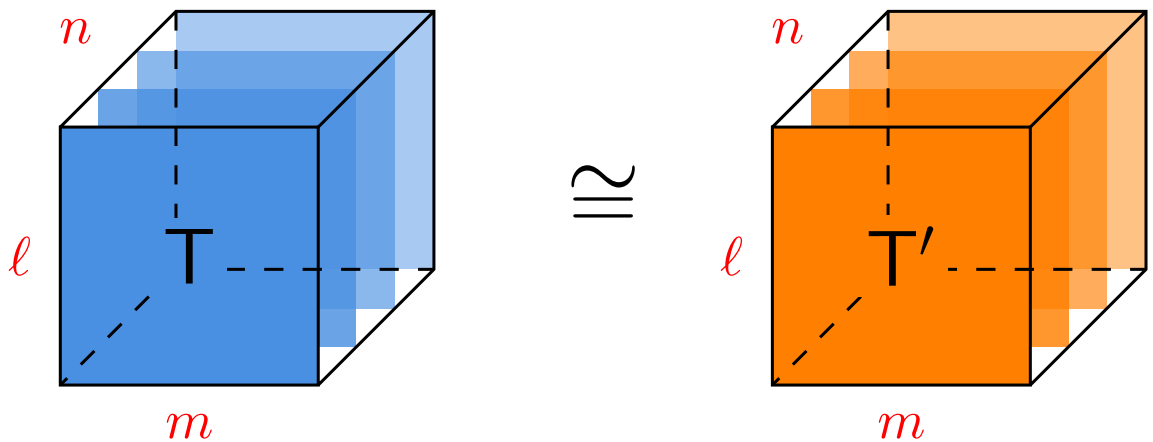
†See [Grochow–Ivanyos–Q–Sun, FOCS '26] for progress on $q = 2$.

Tensor Isomorphism: main result

- W.l.o.g., let us consider $n \times n \times n$ tensors over $\mathbb{F}_q$.
- Brute-force running time: $q^{O(n^2)}$.
- Sun's breakthrough (STOC '23): $q^{O(n^{1.8} \log q)}$, q odd.
- [Ivanyos–Mendoza–Q–Sun–Z, FOCS '24]: $q^{\tilde{O}(n^{1.5})}$, q odd. †

Theorem (Main result on 3-tensor isomorphism)

Given two $\ell \times m \times n$ 3-tensors T, T', there is an algorithm testing whether $\mathsf{T} \cong \mathsf{T}'$ in time $q^{O(\ell+m+n)}$.



†See [Grochow–Ivanyos–Q–Sun, FOCS '26] for progress on $q = 2$.

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Some consequences

The result on 3-TensorIso has the following consequences.

- GroupIso for p-groups of class 2 and exponent p.
- Improve worst-case complexities of algebra isomorphism [Agrawal–Saxena] and some instances of polynomial isomorphism [Patarin].
- The worst-case complexity of 3-TensorIso is much closer to average-case ($q^{O(n)}$, [Li–Q]) and heuristic ($q^{n/2} \cdot \mathrm{poly}(n, \log q)$, [Narayanan–Q–Tang]).

The result on GroupIso has the following consequences.

- GroupIso reduces to GraphIso (quasipolynomial-time, [Babai]).
- GroupIso is viewed as a bottleneck for GraphIso in P [Babai].
- GraphIso in P? Tournament Isomorphism (see e.g. [Raßmann–Schweitzer]).

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

- The main difference with 3-TensorIso:
  - MTC: L acts on the left and right of B_i's.
  - 3-TensorIso: L, R, and T act on left, right, and third direction.
- The automorphism version of MTC: intersection of classical groups [Brooksbank–Wilson].
- [Wilson, Brooksbank–Wilson] introduced the use of $*$-algebras in computing with groups.

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

- The main difference with 3-TensorIso:
  - MTC: L acts on the left and right of B_i's.
  - 3-TensorIso: L, R, and T act on left, right, and third direction.
- The automorphism version of MTC: intersection of classical groups [Brooksbank–Wilson].
- [Wilson, Brooksbank–Wilson] introduced the use of $*$-algebras in computing with groups.

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

- The main difference with 3-TensorIso:
  - MTC: L acts on the left and right of B_i's.
  - 3-TensorIso: L, R, and T act on left, right, and third direction.
- The automorphism version of MTC: intersection of classical groups [Brooksbank–Wilson].
- [Wilson, Brooksbank–Wilson] introduced the use of $*$-algebras in computing with groups.

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

- The main difference with 3-TensorIso:
  - MTC: L acts on the left and right of B_i's.
  - 3-TensorIso: L, R, and T act on left, right, and third direction.
- The automorphism version of MTC: intersection of classical groups [Brooksbank–Wilson].
- [Wilson, Brooksbank–Wilson] introduced the use of $*$-algebras in computing with groups.

Matrix tuple congruence: a key subproblem

Definition (Matrix Tuple Congruence, MTC)

Let $\mathbf{A} = (A_1, \ldots, A_n), \mathbf{B} = (B_1, \ldots, B_n) \in \mathrm{M}(\ell \times \ell, \mathbb{F})^n$ be two matrix tuples. Decide if there exists $L \in \mathrm{GL}(\ell, \mathbb{F})$ such that

$$\forall i \in [n], \quad A_i = L B_i L^{\mathrm{t}}.$$

- The main difference with 3-TensorIso:
  - MTC: L acts on the left and right of B_i's.
  - 3-TensorIso: L, R, and T act on left, right, and third direction.
- The automorphism version of MTC: intersection of classical groups [Brooksbank–Wilson].
- [Wilson, Brooksbank–Wilson] introduced the use of $*$-algebras in computing with groups.

Matrix tuple congruence: main result

Consider matrix tuples in $\mathrm{M}(\ell \times \ell, q)^n$.

- Matrix tuple congruence, brute-force: $q^{\ell^2} \cdot \mathrm{poly}(\ell, n, \log q)$.
- $\mathbb{F} = \mathbb{F}_q$ of char. odd: randomised $\mathrm{poly}(\ell, n, \log q)$ [Ivanyos–Q].
- $\mathbb{F} = \mathbb{F}_2$: $2^{O((\ell+n)^{1.5})}$ [Grochow–Ivanyos–Q–Sun] .

Matrix tuple congruence: main result

Consider matrix tuples in $\mathrm{M}(\ell \times \ell, q)^n$.

- Matrix tuple congruence, brute-force: $q^{\ell^2} \cdot \mathrm{poly}(\ell, n, \log q)$.
- $\mathbb{F} = \mathbb{F}_q$ of char. odd: randomised $\mathrm{poly}(\ell, n, \log q)$ [Ivanyos–Q].
- $\mathbb{F} = \mathbb{F}_2$: $2^{O((\ell+n)^{1.5})}$ [Grochow–Ivanyos–Q–Sun].

Matrix tuple congruence: main result

Consider matrix tuples in $\mathrm{M}(\ell \times \ell, q)^n$.

- Matrix tuple congruence, brute-force: $q^{\ell^2} \cdot \mathrm{poly}(\ell, n, \log q)$.
- $\mathbb{F} = \mathbb{F}_q$ of char. odd: randomised $\mathrm{poly}(\ell, n, \log q)$ [Ivanyos–Q].
- $\mathbb{F} = \mathbb{F}_2$: $2^{O((\ell+n)^{1.5})}$ [Grochow–Ivanyos–Q–Sun] [†].

[†] This is on a closely related problem called quadratic form tuple isometry.

Matrix tuple congruence: main result

Consider matrix tuples in $\mathrm{M}(\ell \times \ell, q)^n$.

- Matrix tuple congruence, brute-force: $q^{\ell^2} \cdot \mathrm{poly}(\ell, n, \log q)$.
- $\mathbb{F} = \mathbb{F}_q$ of char. odd: randomised $\mathrm{poly}(\ell, n, \log q)$ [Ivanyos–Q].
- $\mathbb{F} = \mathbb{F}_2$: $2^{O((\ell+n)^{1.5})}$ [Grochow–Ivanyos–Q–Sun] [†].

Theorem (from this work)

Given two matrix tuples $\mathbf{A}, \mathbf{B} \in \mathrm{M}(\ell \times \ell, \mathbb{F}_2)^n$, *there is an algorithm testing whether they are congruent in time* $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

[†] This is on a closely related problem called quadratic form tuple isometry.

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with
  - MTC: An obstruction module structure. [Bak; Ranicki; Serre]
  - 3-TensorIso: Linear algebraic individualisation [Li–Q, Sun].
  - GroupIso: Characteristic subgroup chain [Canon–Holt], group extensions and cohomology [Grochow–Q].
  - Structures of linearly primitive groups [Jaikin-Zapirain–Pyber].

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with
  - MTC: An obstruction module structure. [Bak; Ranicki; Serre]
  - 3-TensorIso: Linear algebraic individualisation [Li–Q, Sun].
  - GroupIso: Characteristic subgroup chain [Canon–Holt], group extensions and cohomology [Grochow–Q].
  - Structures of linearly primitive groups [Jaikin-Zapirain–Pyber].

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with
  - MTC: An obstruction module structure. [Bak; Ranicki; Serre]
  - 3-TensorIso: Linear algebraic individualisation [Li–Q, Sun].
  - GroupIso: Characteristic subgroup chain [Canon–Holt], group extensions and cohomology [Grochow–Q].
  - Structures of linearly primitive groups [Jaikin-Zapirain–Pyber].

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with
  - MTC: An obstruction module structure. [Bak; Ranicki; Serre]
  - 3-TensorIso: Linear algebraic individualisation [Li–Q, Sun].
  - GroupIso: Characteristic subgroup chain [Canon–Holt], group extensions and cohomology [Grochow–Q].
  - Structures of linearly primitive groups [Jaikin-Zapirain–Pyber].

Summary of the main results

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

Some remarks are as follows.

- Roughly speaking, MTC $\Rightarrow$ 3-TensorIso $\Rightarrow$ GroupIso.
- The organising theme is the Luks–Miyazaki matrix-group divide-and-conquer [Luks, FOCS'92], together with
  - MTC: An obstruction module structure. [Bak; Ranicki; Serre]
  - 3-TensorIso: Linear algebraic individualisation [Li–Q, Sun].
  - GroupIso: Characteristic subgroup chain [Canon–Holt], group extensions and cohomology [Grochow–Q].
  - Structures of linearly primitive groups [Jaikin-Zapirain–Pyber].

Some previous works on special group classes

We list some previous works on special group classes and indicate some techniques useful in this work.

- Groups with trivial solvable radical [Babai–Codenotti–Q]
  - Serves as the induction basis for GroupIso.
- Groups with coprime extensions [Le Gall, Q–Sarma–Tang, Babai–Q]
- Groups with elementary abelian central radical [Grochow–Q]
- Class-2 p-groups with small commutator subgroup [Brooksbank–Wilson, Ivanyos–Q]
- Linear algebraic individualisation [Li–Q, Sun]

Some previous works on special group classes

We list some previous works on special group classes and indicate some techniques useful in this work.

- Groups with trivial solvable radical [Babai–Codenotti–Q]
  - Serves as the induction basis for GroupIso.
- Groups with coprime extensions [Le Gall, Q–Sarma–Tang, Babai–Q]
  - The small set transporter algorithm for module-action compatibility in the semisimple module setting.
- Groups with elementary abelian central radical [Grochow–Q]
- Class-2 p-groups with small commutator subgroup [Brooksbank–Wilson, Ivanyos–Q]
- Linear algebraic individualisation [Li–Q, Sun]

Some previous works on special group classes

We list some previous works on special group classes and indicate some techniques useful in this work.

- Groups with trivial solvable radical [Babai–Codenotti–Q]
  - Serves as the induction basis for GroupIso.
- Groups with coprime extensions [Le Gall, Q–Sarma–Tang, Babai–Q]
  - The small set transporter algorithm for module-action compatibility in the semisimple module setting.
- Groups with elementary abelian central radical [Grochow–Q]
  - The extension and cohomology strategy in GroupIso.
- Class-2 p-groups with small commutator subgroup [Brooksbank–Wilson, Ivanyos–Q]
- Linear algebraic individualisation [Li–Q, Sun]

Some previous works on special group classes

We list some previous works on special group classes and indicate some techniques useful in this work.

- Groups with trivial solvable radical [Babai–Codenotti–Q]
  - Serves as the induction basis for GroupIso.
- Groups with coprime extensions [Le Gall, Q–Sarma–Tang, Babai–Q]
  - The small set transporter algorithm for module-action compatibility in the semisimple module setting.
- Groups with elementary abelian central radical [Grochow–Q]
  - The extension and cohomology strategy in GroupIso.
- Class-2 p-groups with small commutator subgroup [Brooksbank–Wilson, Ivanyos–Q]
  - The $*$-algebra technique and computing algebra structures [Rónyai, Ivanyos].
- Linear algebraic individualisation [Li–Q, Sun]

Some previous works on special group classes

We list some previous works on special group classes and indicate some techniques useful in this work.

- Groups with trivial solvable radical [Babai–Codenotti–Q]
  - Serves as the induction basis for GroupIso.
- Groups with coprime extensions [Le Gall, Q–Sarma–Tang, Babai–Q]
  - The small set transporter algorithm for module-action compatibility in the semisimple module setting.
- Groups with elementary abelian central radical [Grochow–Q]
  - The extension and cohomology strategy in GroupIso.
- Class-2 p-groups with small commutator subgroup [Brooksbank–Wilson, Ivanyos–Q]
  - The $*$-algebra technique and computing algebra structures [Rónyai, Ivanyos].
- Linear algebraic individualisation [Li–Q, Sun]
  - Examining the effect of contracting constant number of slices [Li–Q]
  - Use of the low-rank matrix space [Sun], connections b/w commutative and non-commutative ranks [Ivanyos–Mendoza–Q–Sun–Z, Haramaty–Shpilka]

Some previous works on Group and TensorIso

- GroupIso has been studied in theoretical computer science and computational group theory.
  - Computational group theory focuses on practical algorithms, usually implemented in GAP or Magma
- Some complexity works on Group and TensorIso.
- Some cryptographic works on Group and TensorIso.

Some previous works on Group and TensorIso

- GroupIso has been studied in theoretical computer science and computational group theory.
  - Computational group theory focuses on practical algorithms, usually implemented in GAP or Magma
- Some complexity works on Group and TensorIso.
  - Tensor Isomorphism complexity class [Grochow–Q]
  - Proof complexity of TensorIso [Galesi–Grochow–Pitassi–She]
  - No AC^0 reduction from GraphIso to GroupIso [Chattopadhyay–Torán–Wagner]
- Some cryptographic works on Group and TensorIso.

Some previous works on Group and TensorIso

- GroupIso has been studied in theoretical computer science and computational group theory.
  - Computational group theory focuses on practical algorithms, usually implemented in GAP or Magma
- Some complexity works on Group and TensorIso.
  - Tensor Isomorphism complexity class [Grochow–Q]
  - Proof complexity of TensorIso [Galesi–Grochow–Pitassi–She]
  - No AC^0 reduction from GraphIso to GroupIso [Chattopadhyay–Torán–Wagner]
- Some cryptographic works on Group and TensorIso.
  - Digital signature schemes based on isomorphism problems [Goldreich–Micali–Wigderson, Fiat–Shamir]
  - TensorIso complete problems as the security bases: MEDS[1] and ALTEQ[2]

[1] `https://www.meds-pqc.org/`
[2] `https://pqcalteq.github.io/`

Section 2

Matrix Tuple Congruence

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = L B_i L^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$L B_i = A_i R \quad \text{and} \quad L {B_i}^{\mathrm{t}} = {A_i}^{\mathrm{t}} R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{ (U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; U B_i = B_i V \text{ and } U {B_i}^{\mathrm{t}} = {B_i}^{\mathrm{t}} V \text{ for all } i \in [m] \right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$.
- $F^{\times} := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^{\times}$ such that

$$x^* x = h, \quad \text{where } h := c^* c = (R^{\mathrm{t}} L, L^{\mathrm{t}} R) \in F^{\times}.$$

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$. Then $(F, *)$ is a $*$-*algebra* since $F^* = F$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

The $*$-operation is defined as $(U, V)^* := (V^{\mathrm{t}}, U^{\mathrm{t}})$.

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$. Then $(F, *)$ is a $*$-*algebra* since $F^* = F$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

The $*$-operation is defined as $(U, V)^* := (V^{\mathrm{t}}, U^{\mathrm{t}})$.

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$. Then $(F, *)$ is a $*$-*algebra* since $F^* = F$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$. Then the solution to (1) is $cF^\times$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

The $*$-operation is defined as $(U, V)^* := (V^{\mathrm{t}}, U^{\mathrm{t}})$.

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$. Then $(F, *)$ is a $*$-*algebra* since $F^* = F$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$. Then the solution to (1) is $cF^\times$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

The $*$-operation is defined as $(U, V)^* := (V^{\mathrm{t}}, U^{\mathrm{t}})$.

MTC: reducing to problems on $*$-algebras

Problem (Matrix Tuple Congruence, MTC)

Given $n \times n$ matrices $(A_1, \ldots, A_m)$ and $(B_1, \ldots, B_m)$ over $\mathbb{F}_q$, find $L \in \mathrm{GL}(n, q)$ such that

$$A_i = LB_iL^{\mathrm{t}} \quad \textit{for all} \quad i \in [m].$$

By [Ivanyos–Q '19], we can efficiently find $c := (L, R) \in \mathrm{GL}(n, q)^2$ such that

$$LB_i = A_iR \quad \text{and} \quad LB_i{}^{\mathrm{t}} = A_i{}^{\mathrm{t}}R, \quad \text{for all} \quad i \in [m]. \tag{1}$$

Define a *matrix algebra*:

$$F := \left\{(U, V) \in \mathrm{M}(n, q)^2 \;\middle|\; UB_i = B_iV \text{ and } UB_i{}^{\mathrm{t}} = B_i{}^{\mathrm{t}}V \text{ for all } i \in [m]\right\}.$$

- $F^* := \{(U, V)^* : (U, V) \in F\}$. Then $(F, *)$ is a $*$-*algebra* since $F^* = F$.
- $F^\times := \{(U, V) \in \mathrm{GL}(n, q)^2 : (U, V) \in F\}$. Then the solution to (1) is $cF^\times$.

By [Ivanyos–Q '19], MTC reduces to finding $x \in F^\times$ such that

$$x^*x = h, \quad \text{where } h := c^*c = (R^{\mathrm{t}}L, L^{\mathrm{t}}R) \in F^\times.$$

The $*$-operation is defined as $(U, V)^* := (V^{\mathrm{t}}, U^{\mathrm{t}})$.

MTC: computing on the semisimple quotient

Problem (∗-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a ∗-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the ∗-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$. [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal:** find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$.
  [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$.
  [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$.
  [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$.
  [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$. [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^* x = h$.

- $U(F) = \{u \in F^\times : u^* u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^* x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^* x - h \in J$ and generators of $U(F/J)$. [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^* x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^* x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^* x = h}_{J^t = 0}.$$

MTC: computing on the semisimple quotient

Problem ($*$-Symmetric Decomposition Problem, [Ivanyos-Q '19])

Given a $*$-algebra $(F, *)$ and an element $h \in F^\times$ that satisfies $h^* = h$, the $*$-symmetric decomposition problem is to find $x \in F^\times$ such that $x^*x = h$.

- $U(F) = \{u \in F^\times : u^*u = 1_F\}$.
- **Goal**: find a solution $x \in F^\times$ such that $x^*x = h$ and generators of $U(F)$.
- If a solution exists, the full solution set is $U(F)x$.

Denote by $J = \mathrm{rad}(F)$ the Jacobson radical.

- The quotient F/J is semisimple.
- Find a solution $x \in F^\times$ such that $x^*x - h \in J$ and generators of $U(F/J)$. [Ivanyos-Q '19, with appropriate adaptation to char. 2]
- Since $J^t = 0$ for some t, we lift the solution x through:

$$\underbrace{x^*x - h \in J}_{\text{base case}} \xrightarrow{\text{lift}} x^*x - h \in J^2 \xrightarrow{\text{lift}} \cdots \xrightarrow{\text{lift}} \underbrace{x^*x = h}_{J^t = 0}.$$

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \tag{1}$$

$$x \equiv x' \pmod{J^i}. \tag{2}$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.

- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \tag{1}$$

$$x \equiv x' \pmod{J^i}. \tag{2}$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).
  - This only means some x cannot be lifted; this doesn't mean (1) has no solutions.
  - We can check whether there exists solution d for every fixed x by linear algebra.
  - Main challenge: find which x can be lifted to x'.

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).
  - This only means some x cannot be lifted; this doesn't mean (1) has no solutions.
  - We can check whether there exists solution d for every fixed x by linear algebra.
  - Main challenge: find which x can be lifted to x'.

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).
  - This only means some x cannot be lifted; this doesn't mean (1) has no solutions.
  - We can check whether there exists solution d for every fixed x by linear algebra.
  - Main challenge: find which x can be lifted to x'.

MTC: lifting the solutions over char. odd

- At layer J^i, assume we have $x \in F^\times$ such that $x^*x - h \in J^i$.
- **Goal**: find a lifted solution $x' \in F^\times$ s.t.

$$(x')^*x' - h \in J^{i+1}, \qquad (1)$$

$$x \equiv x' \pmod{J^i}. \qquad (2)$$

- By (2), write $x' = x + d$, where $d \in J^i$.
- Substitute $x' = x + d$ into (1):

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

- Over char. odd, we can let $x^*d \equiv d^*x \equiv \frac{1}{2}(h - x^*x) \pmod{J^{i+1}}$.
  - x is invertible, so we can always construct such d to lift every x to x'.
- Over char. 2, not every x can be lifted to x' satisfying both (1) and (2).
  - This only means some x cannot be lifted; this doesn't mean (1) has no solutions.
  - We can check whether there exists solution d for every fixed x by linear algebra.
  - Main challenge: find which x can be lifted to x'.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \iff \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^*g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \,\mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\iff$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^* d + d^* x \equiv h - x^* x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \iff \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^* g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \, \mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\iff$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \iff \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^*g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \,\mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\iff$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \iff \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^*g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \,\mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\iff$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \iff \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^*g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \,\mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\iff$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

MTC: lifting the solutions over char. 2

For fixed $x \in F^\times$, we can check whether there exists $d \in J^i$ such that

$$x^*d + d^*x \equiv h - x^*x \pmod{J^{i+1}}.$$

For example,

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\text{Coefficient matrix after Gaussian elimination}} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \boxed{b_1} \\ \boxed{b_2} \end{pmatrix}.$$

If $b_1 = b_2 = 0$, we can solve d; otherwise, the lifting equation has no solution.

This illustrates the obstruction vector $\mathrm{obs}_i(x)$ represented by $(b_1, b_2)^{\mathrm{t}}$. Thus,

$$x \text{ can be lifted} \quad \Longleftrightarrow \quad \mathrm{obs}_i(x) = 0$$

Construct an affine action of $H_i = \{g \in F^\times : g^*g \equiv 1_F \pmod{J^i}\}$ such that

$$\mathrm{obs}_i(gx) = L_g \,\mathrm{obs}_i(x) + c_g.$$

Therefore, finding which x can be lifted $\Longleftrightarrow$ solving $\mathrm{Trans}_{H_i}(\mathrm{obs}_i(x), 0)$.

From MTC to quotient tensor module transporter

Let N be the error space and $P \leq N$ the space of solvable errors.

Our example illustrates the quotient $Q = N/P$:

$$\underbrace{\begin{pmatrix} 1 \\ 0 \\ b_1 \\ b_2 \end{pmatrix}}_{\text{error in } N} \xrightarrow{\text{quotient by } P} \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{\text{obstruction in } Q} .$$

Tensor module: an ambient space Z given by direct sums of tensor products.

We solve MTC and degree-1 QTMT together.

We also solve degree-2 QTMT for 3-TensorIso and degree-2 QTMT for GroupIso.

From MTC to quotient tensor module transporter

Let N be the error space and $P \leq N$ the space of solvable errors.

Our example illustrates the quotient $Q = N/P$:

$$\underbrace{\begin{pmatrix} 1 \\ 0 \\ b_1 \\ b_2 \end{pmatrix}}_{\text{error in } N} \xrightarrow{\text{quotient by } P} \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{\text{obstruction in } Q} .$$

Tensor module: an ambient space Z given by direct sums of tensor products.

We solve MTC and degree-1 QTMT together.

We also solve degree-2 QTMT for 3-TensorIso and degree-2 QTMT for GroupIso.

From MTC to quotient tensor module transporter

Let N be the error space and $P \leq N$ the space of solvable errors.

Our example illustrates the quotient $Q = N/P$:

$$\underbrace{\begin{pmatrix} 1 \\ 0 \\ b_1 \\ b_2 \end{pmatrix}}_{\text{error in } N} \xrightarrow{\text{quotient by } P} \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{\text{obstruction in } Q} .$$

Tensor module: an ambient space Z given by direct sums of tensor products.

Problem (Quotient Tensor Module Transporter Problem, informal)

Given a group H, H-invariant subspaces $P \leq N \leq Z$, an affine action $z \mapsto L_g z + c_g$ on $Q = N/P$, with L_g induced by the H-action on Z, and two points $u, v \in Q$, compute

$$\mathrm{Trans}_H(u, v) := \{g \in H : L_g u + c_g = v\}.$$

We solve MTC and degree-1 QTMT together.

We also solve degree-2 QTMT for 3-TensorIso and degree-2 QTMT for GroupIso.

From MTC to quotient tensor module transporter

Let N be the error space and $P \leq N$ the space of solvable errors.

Our example illustrates the quotient $Q = N/P$:

$$\underbrace{\begin{pmatrix} 1 \\ 0 \\ b_1 \\ b_2 \end{pmatrix}}_{\text{error in } N} \xrightarrow{\text{quotient by } P} \underbrace{\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}}_{\text{obstruction in } Q} .$$

Tensor module: an ambient space Z given by direct sums of tensor products.

Problem (Quotient Tensor Module Transporter Problem, informal)

Given a group H, H-invariant subspaces $P \leq N \leq Z$, an affine action $z \mapsto L_g z + c_g$ on $Q = N/P$, with L_g induced by the H-action on Z, and two points $u, v \in Q$, compute

$$\mathrm{Trans}_H(u, v) := \{g \in H : L_g u + c_g = v\}.$$

We solve MTC and degree-1 QTMT together.

We also solve degree-2 QTMT for 3-TensorIso and degree-2 QTMT for GroupIso.

Section 3

3-Tensor Isomorphism

3-TensorIso: algorithmic overview

Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

Group structure
subspaces and blocks

3-TensorIso: algorithmic overview

Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

Group structure
subspaces and blocks

3-TensorIso: algorithmic overview

Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

Group structure
subspaces and blocks

3-TensorIso: algorithmic overview

Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

Group structure
subspaces and blocks

3-TensorIso: algorithmic overview

Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

when decomposition stops

Group structure
subspaces and blocks

Tensor structure
contractions and rank

new constraints and new progress

3-TensorIso: algorithmic overview

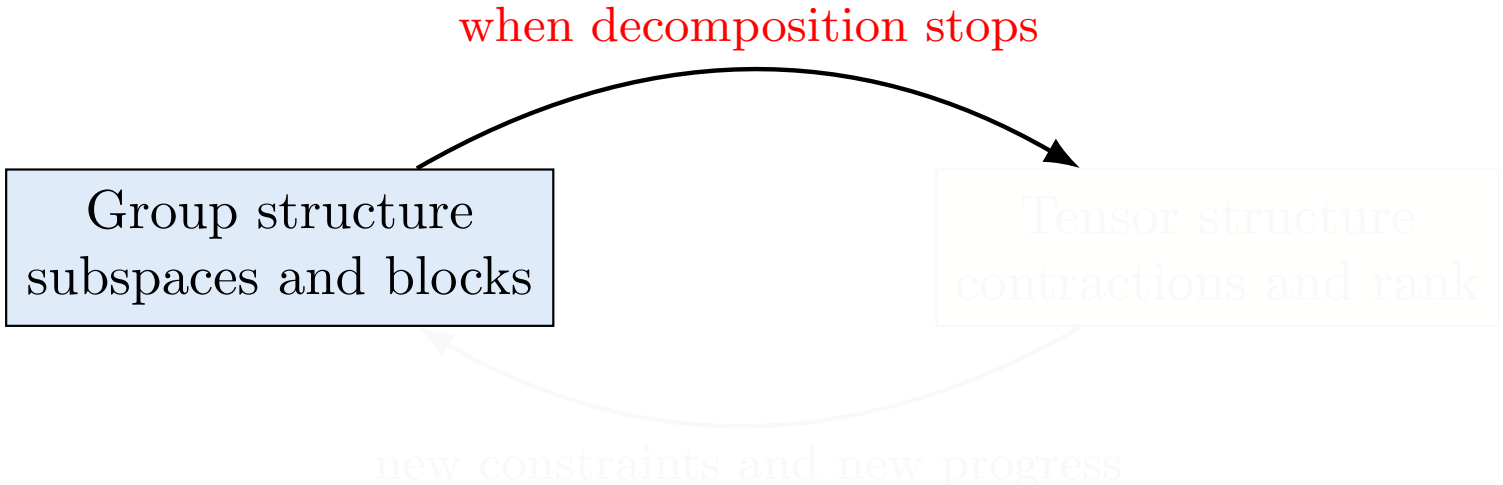
Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$



3-TensorIso: algorithmic overview

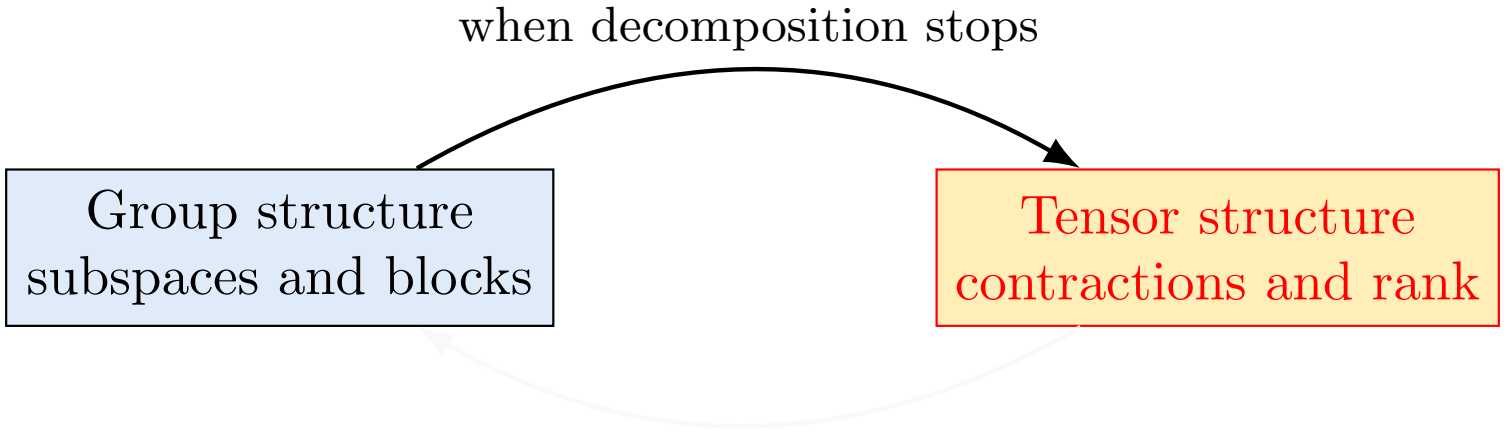
Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$



3-TensorIso: algorithmic overview

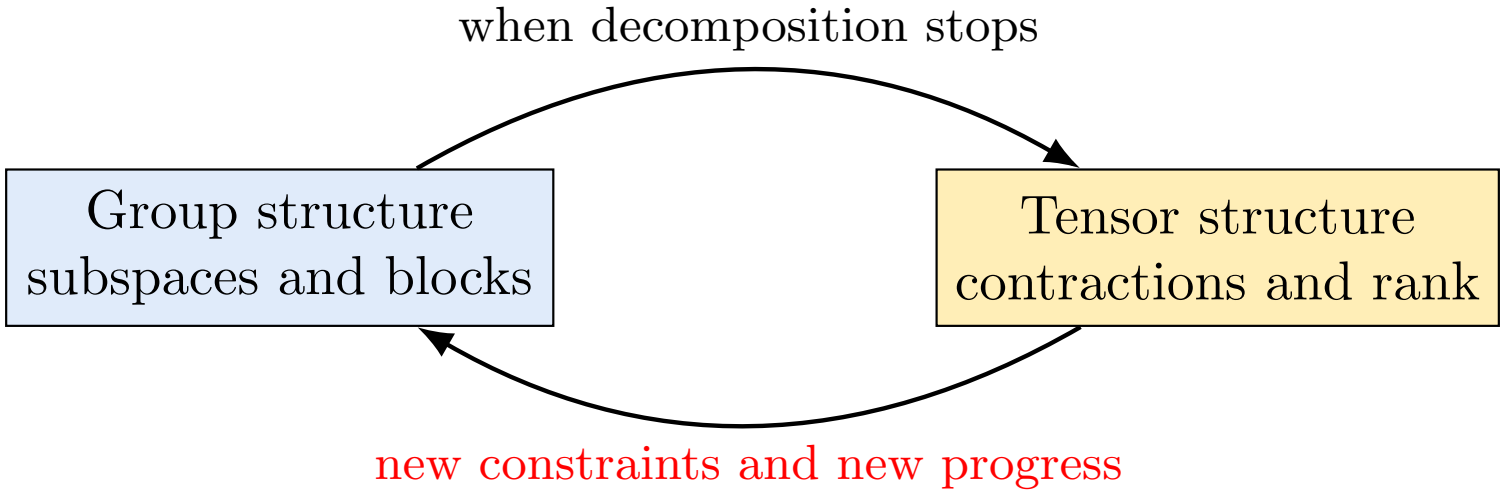
Problem (3-Tensor Isomorphism, in trilinear-form formulation)

Let A_1, A_2, A_3 *be finite-dimensional vector spaces over* $\mathbb{F}_q$. *Given two trilinear forms* $\mathsf{T}, \mathsf{T}' : A_1 \times A_2 \times A_3 \to \mathbb{F}_q$ *and a subgroup* $H \leq \mathrm{GL}(A_1) \times \mathrm{GL}(A_2) \times \mathrm{GL}(A_3)$, *find the transporter*

$$\mathrm{Trans}_H(\mathsf{T}, \mathsf{T}') := \{h \in H : h \cdot \mathsf{T} = \mathsf{T}'\},$$

where $h = (h_1, h_2, h_3)$ *acts by*

$$(h \cdot \mathsf{T})(v_1, v_2, v_3) := \mathsf{T}(h_1^{-1} v_1, h_2^{-1} v_2, h_3^{-1} v_3).$$

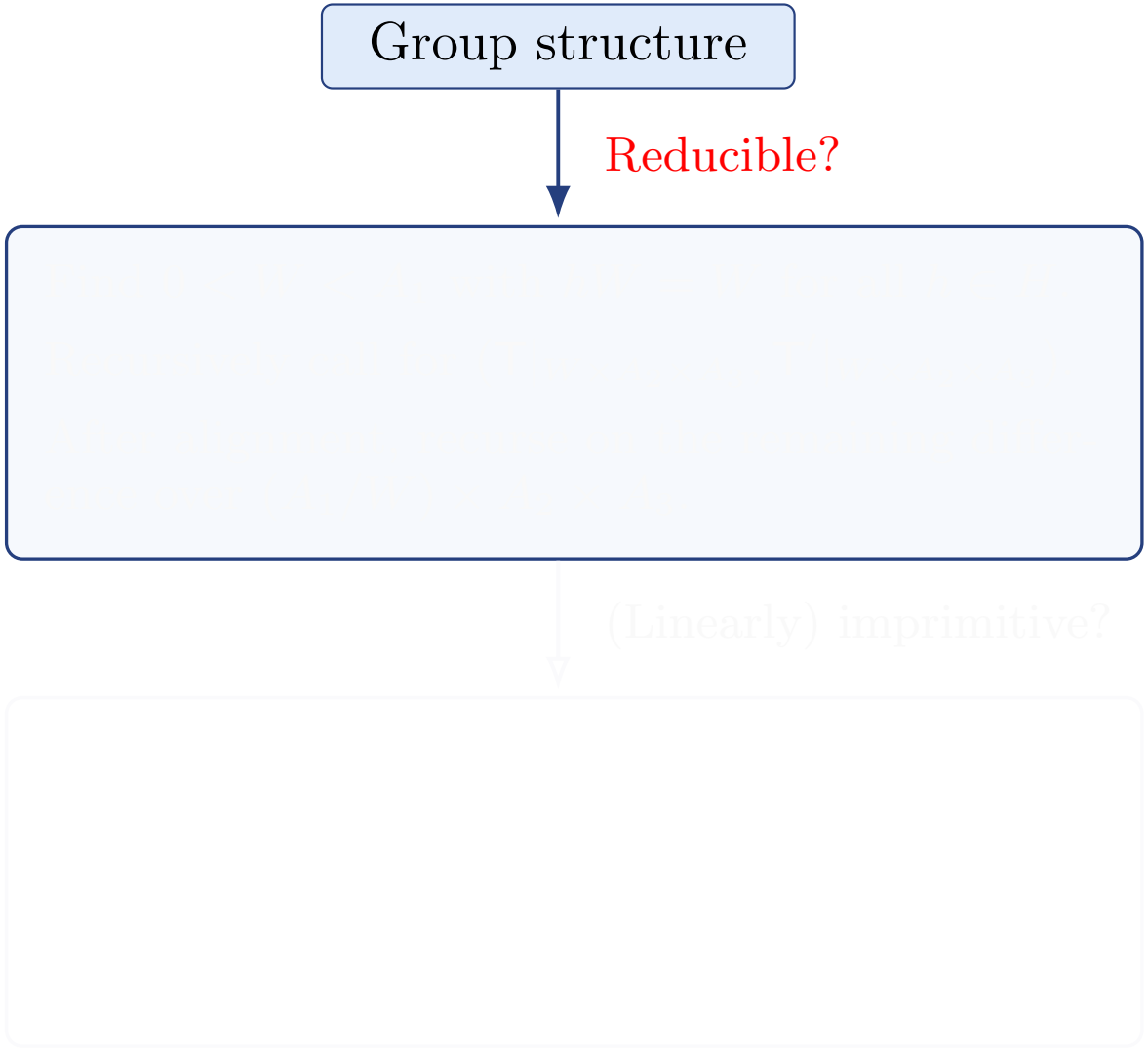


3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].



Candidate action illlustraion

Each $*$ denotes a matrix block.

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

(Linearly) imprimitive?

Candidate action illlustraion

Each $*$ denotes a matrix block.

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

↓ Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

Recursively call for $(\mathsf{T}|_{W \times A_2 \times A_3}, \mathsf{T}'|_{W \times A_2 \times A_3})$.

↓ (Linearly) imprimitive?

Candidate action illlustraion

Each $*$ denotes a matrix block.

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

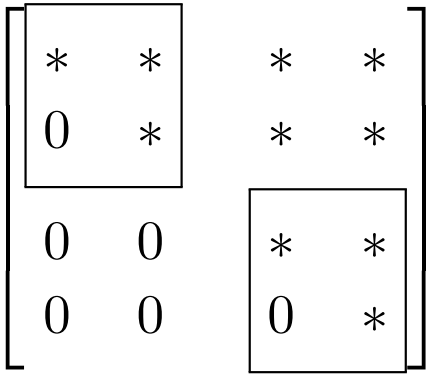
Recursively call for $(\mathsf{T}|_{W \times A_2 \times A_3}, \mathsf{T}'|_{W \times A_2 \times A_3})$.

After alignment, recurse on the remaining difference over $(A_1/W) \times A_2 \times A_3$.

(Linearly) imprimitive?

Candidate action illlustraion

Each $*$ denotes a matrix block.



3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

↓ Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

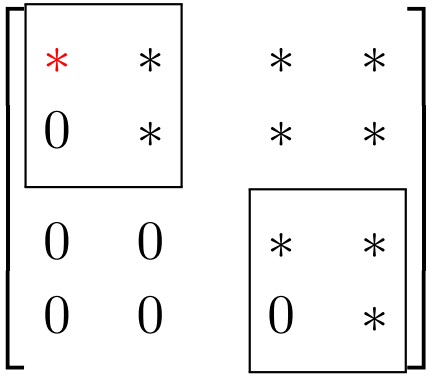
Recursively call for $(\mathsf{T}|_{W \times A_2 \times A_3}, \mathsf{T}'|_{W \times A_2 \times A_3})$.

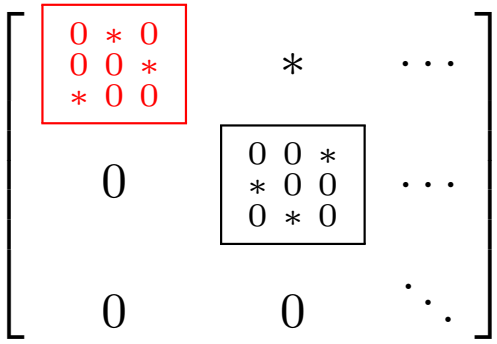
After alignment, recurse on the remaining difference over $(A_1/W) \times A_2 \times A_3$.

↓ (Linearly) imprimitive?

Candidate action illlustraion

Each $*$ denotes a matrix block.





3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

↓ Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

Recursively call for $(\mathsf{T}|_{W \times A_2 \times A_3}, \mathsf{T}'|_{W \times A_2 \times A_3})$.

After alignment, recurse on the remaining difference over $(A_1/W) \times A_2 \times A_3$.

↓ (Linearly) imprimitive?

$$A_1 = D_1 \oplus \cdots \oplus D_s$$

$\{hD_1, \ldots, hD_s\} = \{D_1, \ldots, D_s\}$ for all $h \in H$

Candidate action illlustraion

Each $*$ denotes a matrix block.

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$$

$$\begin{bmatrix} 0 & * & 0 & * & \cdots \\ 0 & 0 & * & * & \\ * & 0 & 0 & & \\ 0 & & * & 0 & \cdots \\ & 0 & 0 & * & \\ & & * & 0 & \\ 0 & & 0 & & \ddots \end{bmatrix}$$

3-TensorIso: progress on the group structure

The main idea is to follow the divide-and-conquer framework of [Luks, FOCS'92].

Group structure

↓ Reducible?

Find $0 < W < A_1$ with $hW = W$ for all $h \in H$.

Recursively call for $(\mathsf{T}|_{W \times A_2 \times A_3}, \mathsf{T}'|_{W \times A_2 \times A_3})$.

After alignment, recurse on the remaining difference over $(A_1/W) \times A_2 \times A_3$.

↓ (Linearly) imprimitive?

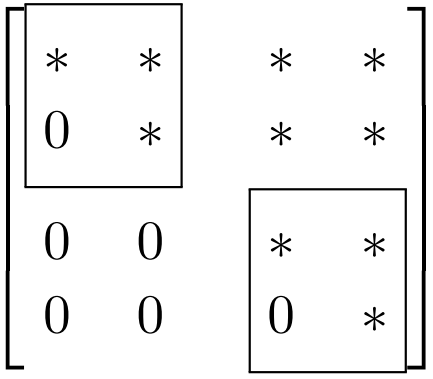
$$A_1 = D_1 \oplus \cdots \oplus D_s$$

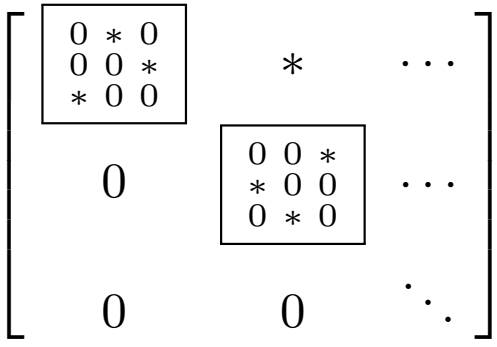
$$\{hD_1, \ldots, hD_s\} = \{D_1, \ldots, D_s\} \text{ for all } h \in H$$

Recursively call for $(\mathsf{T}|_{D_i \times A_2 \times A_3}, \mathsf{T}'|_{D_j \times A_2 \times A_3})$, among all $h \in H$ s.t. $hD_i = D_j$, for $1 \le i, j \le s$.

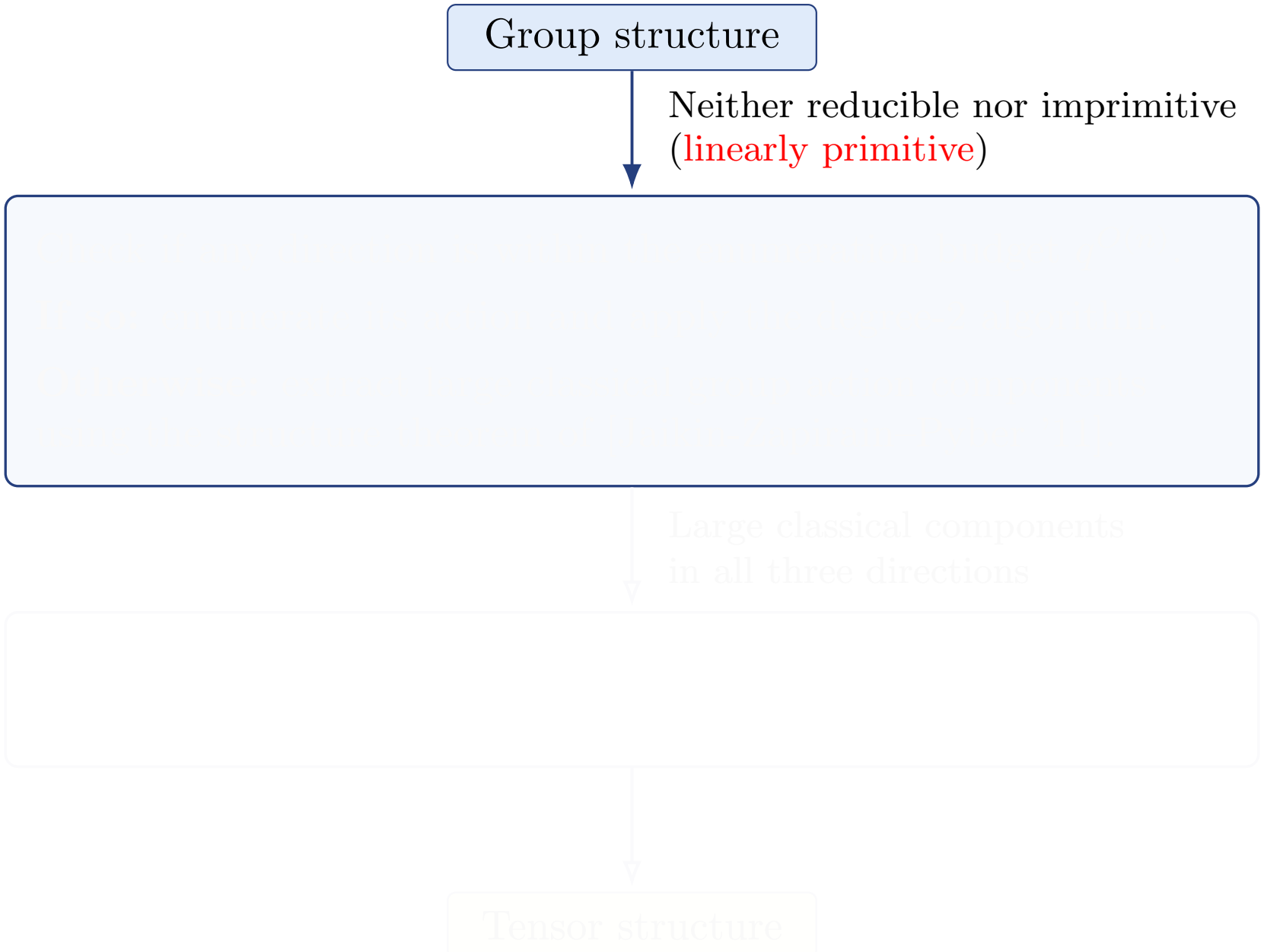
Candidate action illlustraion

Each $*$ denotes a matrix block.





3-TensorIso: progress on the group structure



3-TensorIso: progress on the group structure

Group structure

Neither reducible nor imprimitive
(linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

Large classical components
in all three directions

Tensor structure

3-TensorIso: progress on the group structure

Group structure

Neither reducible nor imprimitive
(linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

If so: enumerate its action and apply the degree-2 algorithm. †

Large classical components
in all three directions

Tensor structure

† An extension of matrix tuple isomorphism to affine group actions, solvable in $q^{O(n)}$ time.

3-TensorIso: progress on the group structure

Group structure

Neither reducible nor imprimitive
(linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

If so: enumerate its action and apply the degree-2 algorithm.

Otherwise: extract large classical group action components using the structure theorem of [Jaikin-Zapirain–Pyber '11].

Large classical components
in all three directions

Tensor structure

3-TensorIso: progress on the group structure

Group structure

Neither reducible nor imprimitive
(linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

If so: enumerate its action and apply the degree-2 algorithm.

Otherwise: extract large classical group action components using the structure theorem of [Jaikin-Zapirain–Pyber '11].

Large classical components
in all three directions

Tensor structure

3-TensorIso: progress on the group structure

Group structure

↓ Neither reducible nor imprimitive (linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

If so: enumerate its action and apply the degree-2 algorithm.

Otherwise: extract large classical group action components using the structure theorem of [Jaikin-Zapirain–Pyber '11].

↓ Large classical components in all three directions

Handle repeated copies, quotient spaces, and translations.

The remaining input is an ordinary 3-TensorIso instance again.

↓

Tensor structure

3-TensorIso: progress on the group structure

Group structure

↓ Neither reducible nor imprimitive (linearly primitive)

Check if any direction is within the enumeration budget $q^{O(n)}$.

If so: enumerate its action and apply the degree-2 algorithm.

Otherwise: extract large classical group action components using the structure theorem of [Jaikin-Zapirain–Pyber '11].

↓ Large classical components in all three directions

Handle repeated copies, quotient spaces, and translations.

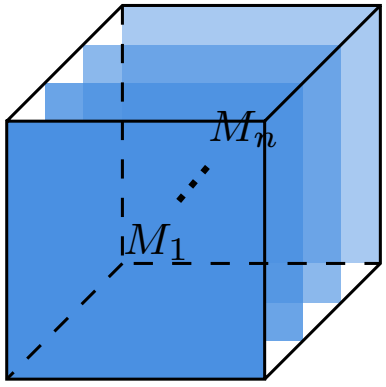
The remaining input is an ordinary 3-TensorIso instance again.

↓

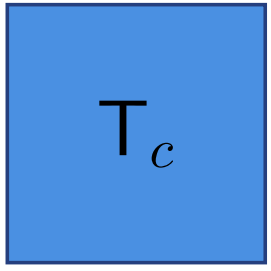
Tensor structure

3-TensorIso: main ideas for tensor structure

1. Contraction. [Li-Q]



$c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$



$\mathsf{T}_c = \sum_{i=1}^{n} c_i M_i$

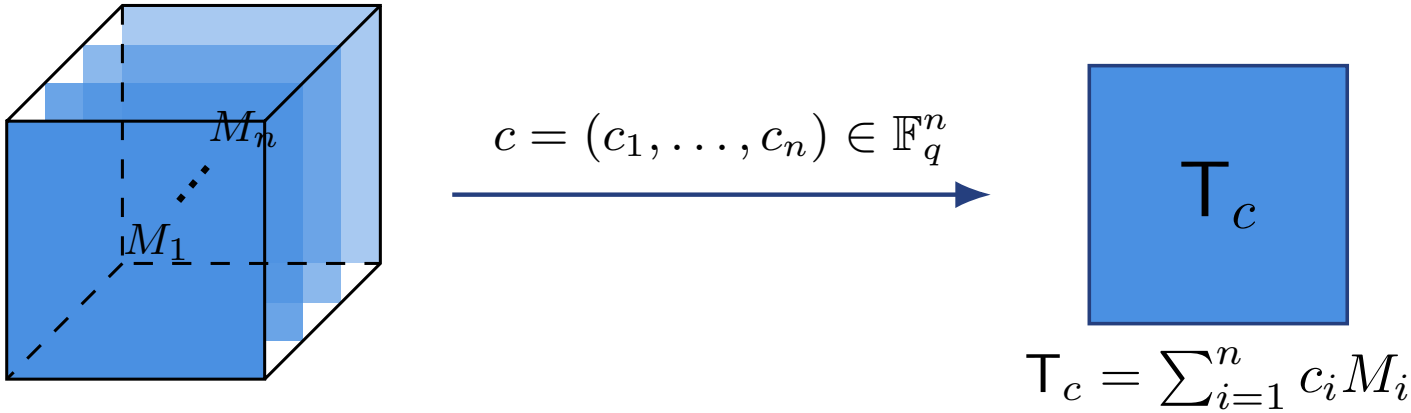
Matching the contractions adds new constraints on the allowed group actions.

2. Exploiting low-rank structures. [Sun; Ivanyos-Mendoza-Q-Sun-Z]

A low-rank matrix subspace gives a zero block.

3-TensorIso: main ideas for tensor structure

1. Contraction. [Li-Q]



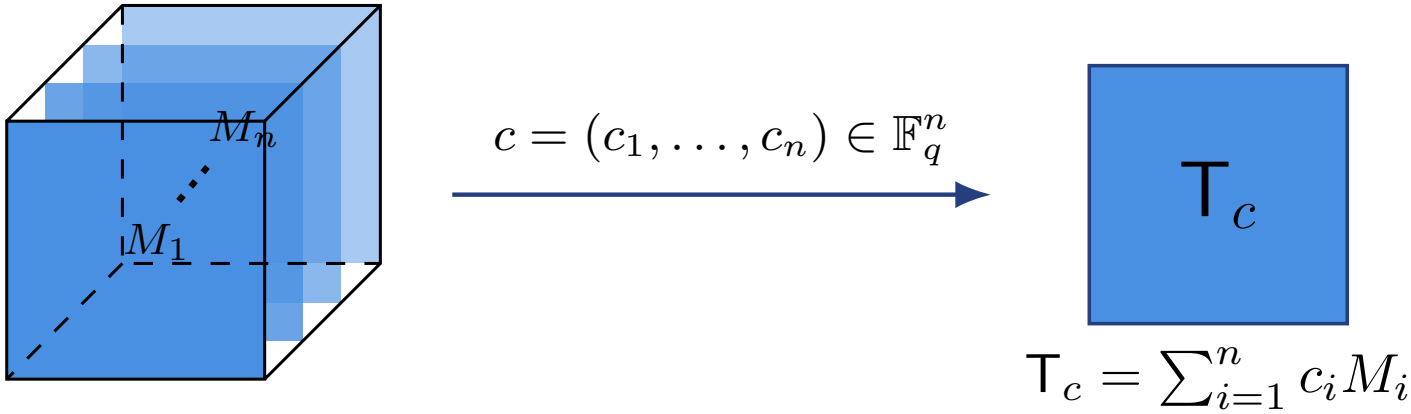
Matching the contractions adds new constraints on the allowed group actions.

2. Exploiting low-rank structures. [Sun; Ivanyos-Mendoza-Q-Sun-Z]

A low-rank matrix subspace gives a zero block.

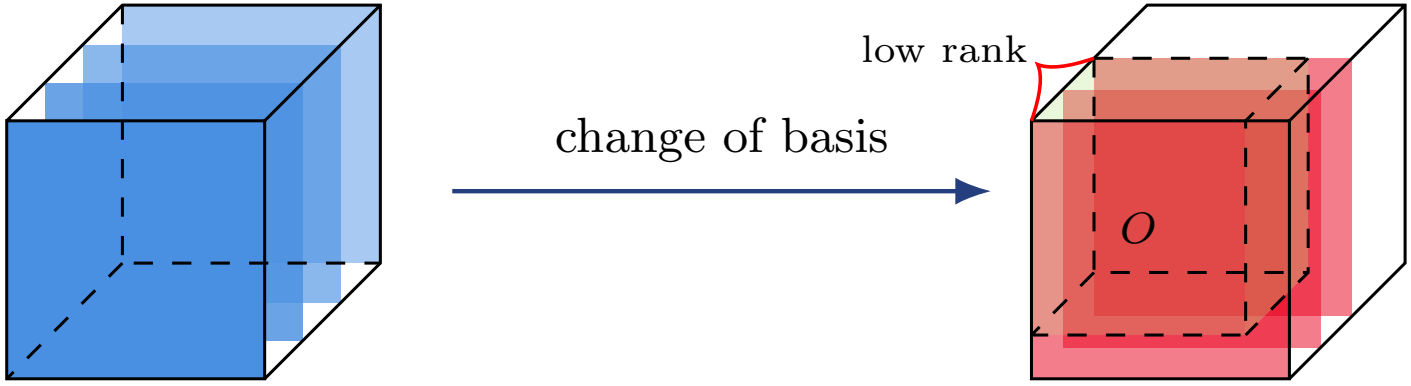
3-TensorIso: main ideas for tensor structure

1. Contraction. [Li-Q]



Matching the contractions adds new constraints on the allowed group actions.

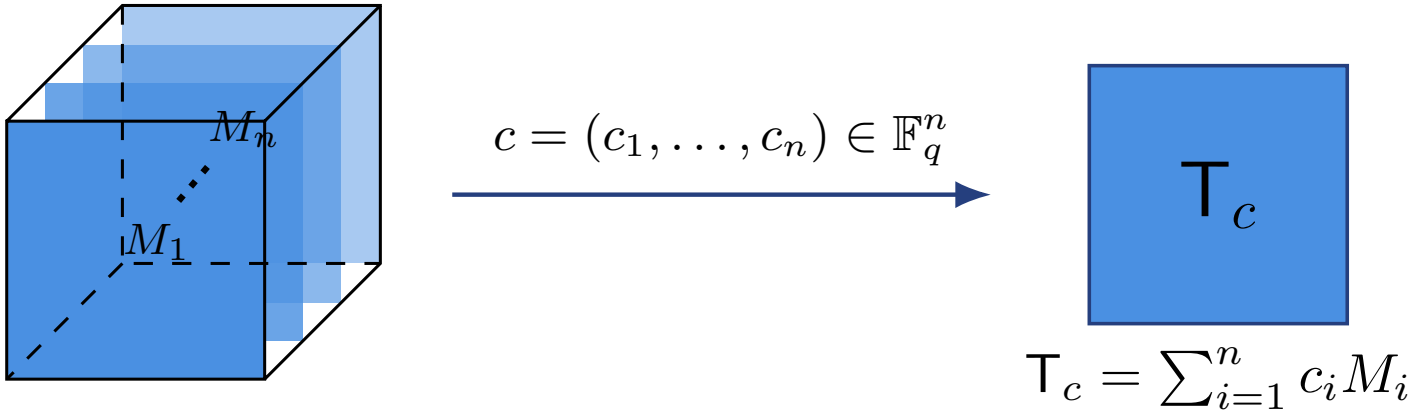
2. Exploiting low-rank structures. [Sun; Ivanyos-Mendoza-Q-Sun-Z]



A low-rank matrix subspace gives a zero block.

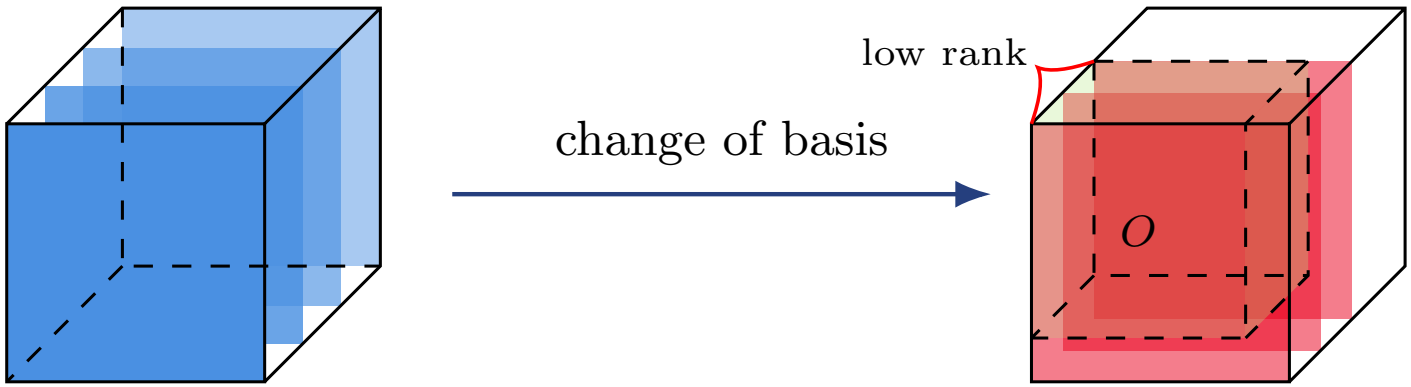
3-TensorIso: main ideas for tensor structure

1. Contraction. [Li-Q]



Matching the contractions adds new constraints on the allowed group actions.

2. Exploiting low-rank structures. [Sun; Ivanyos-Mendoza-Q-Sun-Z]



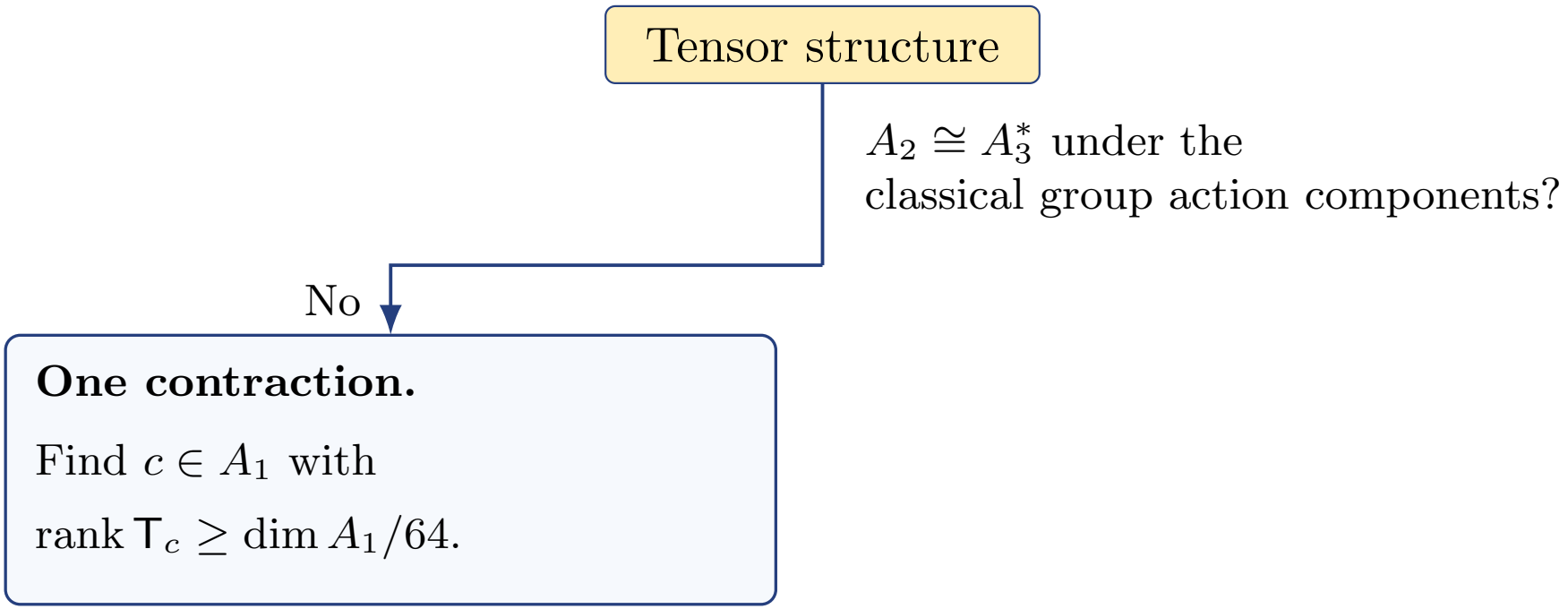
A low-rank matrix subspace gives a zero block.

3-TensorIso: progress on the tensor structure

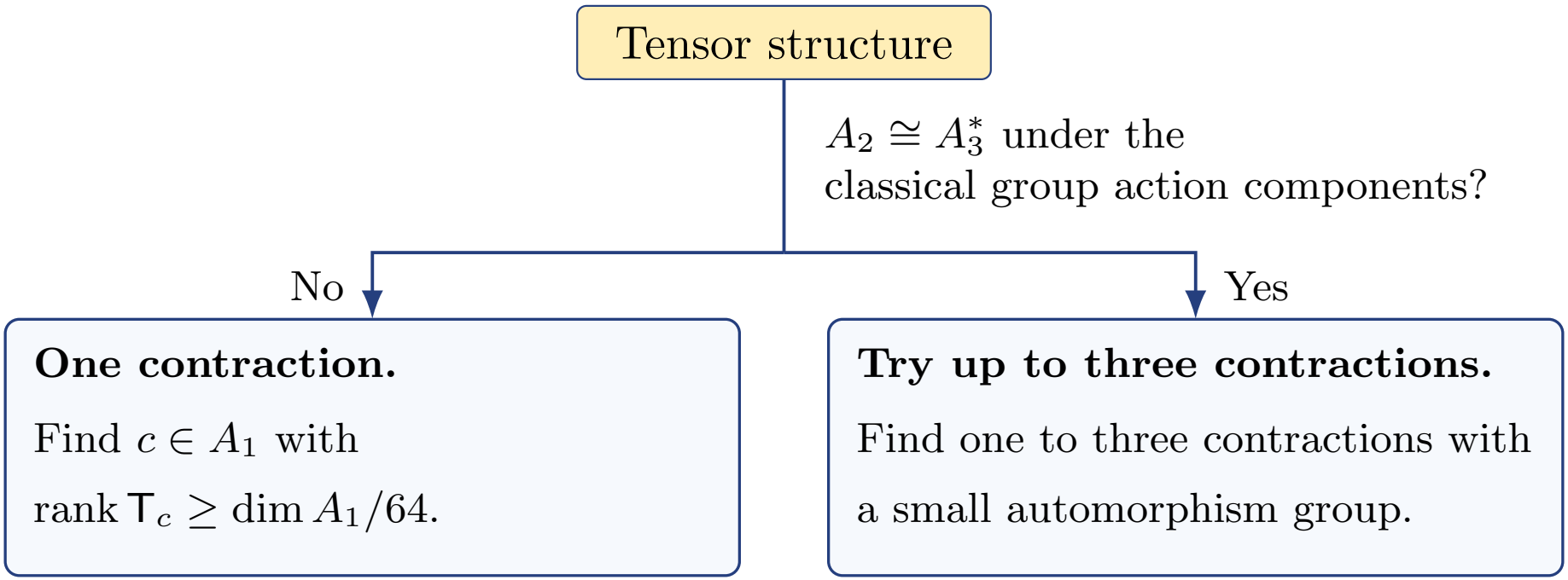
Tensor structure

$A_2 \cong A_3^*$ under the
classical group action components?

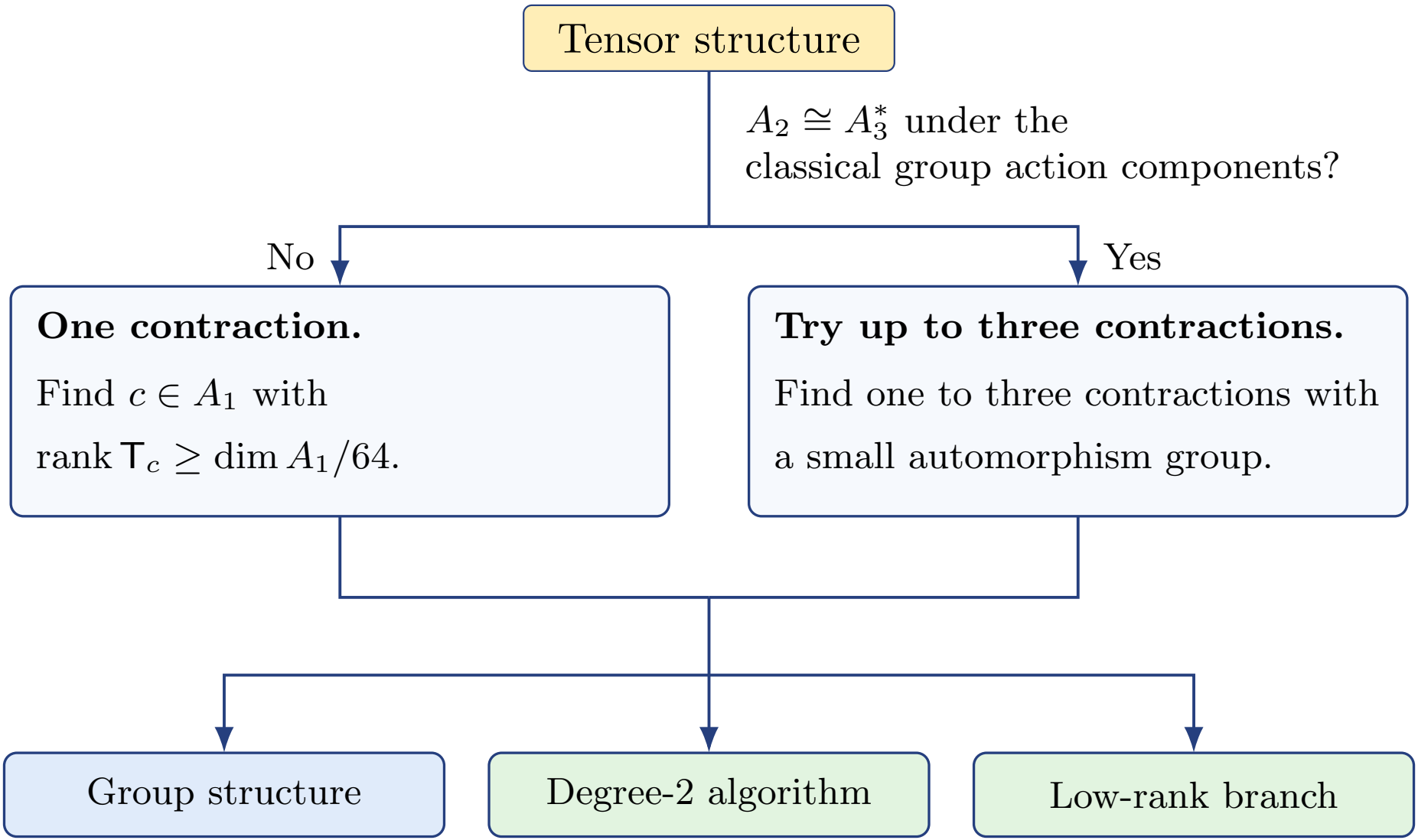
3-TensorIso: progress on the tensor structure



3-TensorIso: progress on the tensor structure



3-TensorIso: progress on the tensor structure



3-TensorIso: progress on the tensor structure

Tensor structure

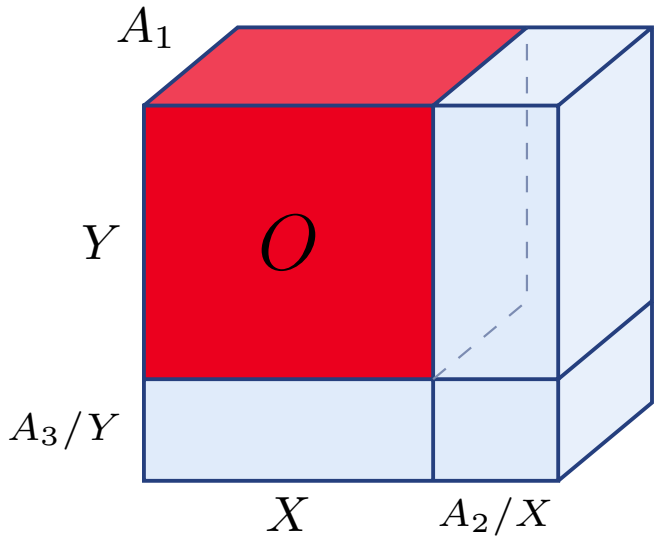
Low-rank branch

1. Find and align large zero blocks.

For $X \leq A_2$, $Y \leq A_3$, arrange that

$$\mathsf{T}|_{A_1 \times X \times Y} = \mathsf{T}'|_{A_1 \times X \times Y} = 0$$

Tensor illustration



Align the zero block.

3-TensorIso: progress on the tensor structure

Tensor structure

Low-rank branch

1. Find and align large zero blocks.

For $X \leq A_2$, $Y \leq A_3$, arrange that

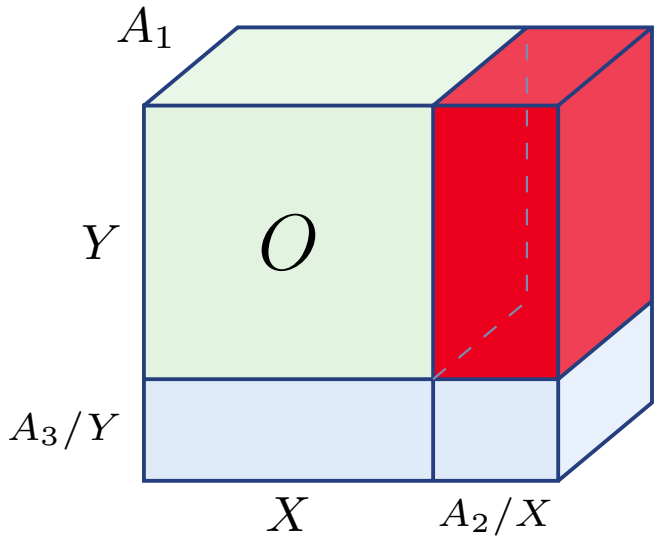
$$\mathsf{T}|_{A_1 \times X \times Y} = \mathsf{T}'|_{A_1 \times X \times Y} = 0$$

2. Recurse on the right strip.

$$A_1 \times (A_2/X) \times Y$$

At each step, keep only solutions that matched all earlier parts.

Tensor illustration



Match the right strip.

3-TensorIso: progress on the tensor structure

Tensor structure

Low-rank branch

1. Find and align large zero blocks.

For $X \le A_2$, $Y \le A_3$, arrange that

$$\mathsf{T}|_{A_1 \times X \times Y} = \mathsf{T}'|_{A_1 \times X \times Y} = 0$$

2. Recurse on the right strip.

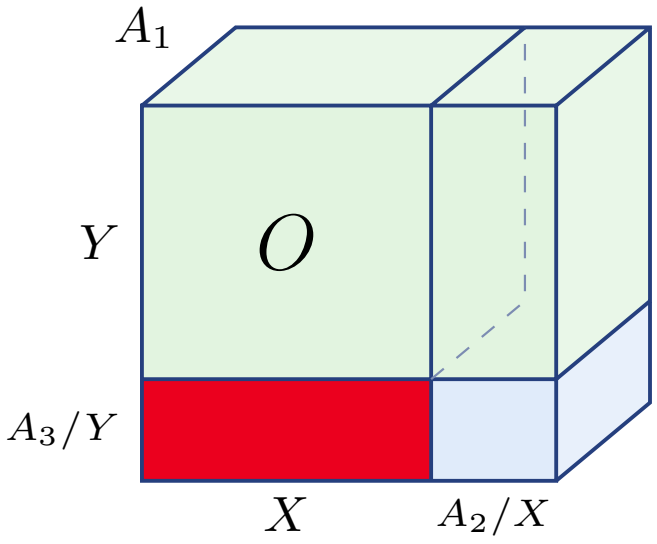
$$A_1 \times (A_2/X) \times Y$$

3. Recurse on the lower strip.

$$A_1 \times X \times (A_3/Y)$$

At each step, keep only solutions that matched all earlier parts.

Tensor illustration



Match the lower strip.

3-TensorIso: progress on the tensor structure

Tensor structure

Low-rank branch

1. Find and align large zero blocks.

For $X \le A_2$, $Y \le A_3$, arrange that

$$\mathsf{T}|_{A_1 \times X \times Y} = \mathsf{T}'|_{A_1 \times X \times Y} = 0$$

2. Recurse on the right strip.

$$A_1 \times (A_2/X) \times Y$$

3. Recurse on the lower strip.

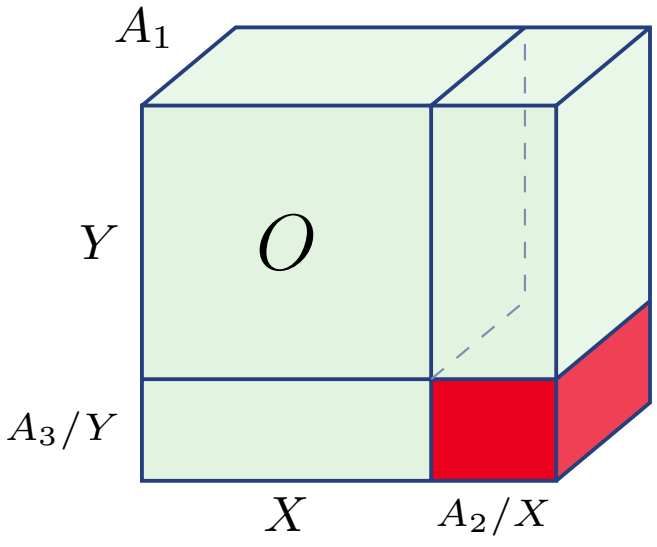
$$A_1 \times X \times (A_3/Y)$$

4. Match the remaining difference.

$$A_1 \times (A_2/X) \times (A_3/Y)$$

At each step, keep only solutions that matched all earlier parts.

Tensor illustration



Only the lower-right difference remains.

3-TensorIso: running time bound

Let $d = \sum_{i=1}^{3} \dim A_i$. Count a new recursive call only after a tensor size reduction:

- Non-recursive cost per call: $q^{O(d)}$
  - E.g., degree-2 solving, group-structure computations, and linear algebra
- Number of recursive calls (for one fixed enumeration): $\mathrm{poly}(d, \log q)^{O(\log d)}$
- Number of enumerations: $q^{O(d)}$

Therefore, the total cost is

$$\underbrace{q^{O(d)}}_{\text{local cost}} \underbrace{\mathrm{poly}(d, \log q)^{O(\log d)}}_{\text{recursion number}} \underbrace{q^{O(d)}}_{\text{enumeration cost}} = q^{O(d)}$$

3-TensorIso: running time bound

Let $d = \sum_{i=1}^{3} \dim A_i$. Count a new recursive call only after a tensor size reduction:

- Non-recursive cost per call: $q^{O(d)}$
  - E.g., degree-2 solving, group-structure computations, and linear algebra
- Number of recursive calls (for one fixed enumeration): $\mathrm{poly}(d, \log q)^{O(\log d)}$
  - Number of sub-problems: $\mathrm{poly}(d, \log q)$
  - Depth of recursive tree: $O(\log d)$
- Number of enumerations: $q^{O(d)}$

Therefore, the total cost is

$$\underbrace{q^{O(d)}}_{\text{local cost}} \underbrace{\mathrm{poly}(d, \log q)^{O(\log d)}}_{\text{recursion number}} \underbrace{q^{O(d)}}_{\text{enumeration cost}} = q^{O(d)}$$

3-TensorIso: running time bound

Let $d = \sum_{i=1}^{3} \dim A_i$. Count a new recursive call only after a tensor size reduction:

- Non-recursive cost per call: $q^{O(d)}$
  - E.g., degree-2 solving, group-structure computations, and linear algebra
- Number of recursive calls (for one fixed enumeration): $\mathrm{poly}(d, \log q)^{O(\log d)}$
  - Number of sub-problems: $\mathrm{poly}(d, \log q)$
  - Depth of recursive tree: $O(\log d)$
- Number of enumerations: $q^{O(d)}$
  - Key lemma: dimension loss $\Delta \Rightarrow$ enumeration cost at most $q^{C\Delta}$

Therefore, the total cost is

$$\underbrace{q^{O(d)}}_{\text{local cost}} \underbrace{\mathrm{poly}(d, \log q)^{O(\log d)}}_{\text{recursion number}} \underbrace{q^{O(d)}}_{\text{enumeration cost}} = q^{O(d)}$$

3-TensorIso: running time bound

Let $d = \sum_{i=1}^{3} \dim A_i$. Count a new recursive call only after a tensor size reduction:

- Non-recursive cost per call: $q^{O(d)}$
  - E.g., degree-2 solving, group-structure computations, and linear algebra
- Number of recursive calls (for one fixed enumeration): $\mathrm{poly}(d, \log q)^{O(\log d)}$
  - Number of sub-problems: $\mathrm{poly}(d, \log q)$
  - Depth of recursive tree: $O(\log d)$
- Number of enumerations: $q^{O(d)}$
  - Key lemma: dimension loss $\Delta \Rightarrow$ enumeration cost at most $q^{C\Delta}$

Therefore, the total cost is

$$\underbrace{q^{O(d)}}_{\text{local cost}} \underbrace{\mathrm{poly}(d, \log q)^{O(\log d)}}_{\text{recursion number}} \underbrace{q^{O(d)}}_{\text{enumeration cost}} = q^{O(d)}$$

Section 4

Finite Group Isomorphism

GroupIso: characteristic subgroup chains

To test if two groups G and H of order N are isomorphic.

- Solvable radical of G, $\mathrm{srad}(G)$: the largest solvable normal subgroup.
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are trivial, use [Babai–Codenotti–Q].
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are nontrivial, build a characteristic subgroup chain (see also [Cannon–Holt]).

- $G \cong H \Rightarrow \mathrm{AC}(G) \cong \mathrm{AC}(H) \cong \mathbb{Z}_p^a$.
- By induction, assume $G/\mathrm{AC}(G) \cong H/\mathrm{AC}(H)$, identified as Q.

GroupIso: characteristic subgroup chains

To test if two groups G and H of order N are isomorphic.

- Solvable radical of G, $\mathrm{srad}(G)$: the largest solvable normal subgroup.
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are trivial, use [Babai–Codenotti–Q].
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are nontrivial, build a characteristic subgroup chain (see also [Cannon–Holt]).
  1. Compute the solvable radical $\mathrm{srad}(G) \leq G$, and the derived series of $\mathrm{srad}(G)$

$$1 = S_0 \trianglelefteq S_1 \trianglelefteq S_2 \trianglelefteq \ldots \trianglelefteq S_\ell = \mathrm{srad}(G),$$

     where $S_{i-1} = [S_i, S_i]$ for $i \in [\ell]$.
  2. p: smallest prime dividing $|S_1|$. D: unique p-Sylow subgroup of S_1.
  3. $\mathrm{AC}(G) := \{x \in D \mid x^p = 1\}$ is elementary abelian.
  4. Inductively, construct a chain of characteristic subgroups

$$1 = T_0 \trianglelefteq T_1 \trianglelefteq T_2 \trianglelefteq \ldots \trianglelefteq T_s = \mathrm{srad}(G) \trianglelefteq G,$$

     where every T_i/T_{i-1} is elementary abelian.

- $G \cong H \Rightarrow \mathrm{AC}(G) \cong \mathrm{AC}(H) \cong \mathbb{Z}_p^a$.
- By induction, assume $G/\mathrm{AC}(G) \cong H/\mathrm{AC}(H)$, identified as Q.

GroupIso: characteristic subgroup chains

To test if two groups G and H of order N are isomorphic.

- Solvable radical of G, $\mathrm{srad}(G)$: the largest solvable normal subgroup.
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are trivial, use [Babai–Codenotti–Q].
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are nontrivial, build a characteristic subgroup chain (see also [Cannon–Holt]).
  1. Compute the solvable radical $\mathrm{srad}(G) \leq G$, and the derived series of $\mathrm{srad}(G)$

     $$1 = S_0 \trianglelefteq S_1 \trianglelefteq S_2 \trianglelefteq \ldots \trianglelefteq S_\ell = \mathrm{srad}(G),$$

     where $S_{i-1} = [S_i, S_i]$ for $i \in [\ell]$.
  2. p: smallest prime dividing $|S_1|$. D: unique p-Sylow subgroup of S_1.
  3. $\mathrm{AC}(G) := \{x \in D \mid x^p = 1\}$ is elementary abelian.
  4. Inductively, construct a chain of characteristic subgroups

     $$1 = T_0 \trianglelefteq T_1 \trianglelefteq T_2 \trianglelefteq \ldots \trianglelefteq T_s = \mathrm{srad}(G) \trianglelefteq G,$$

     where every T_i/T_{i-1} is elementary abelian.

- $G \cong H \Rightarrow \mathrm{AC}(G) \cong \mathrm{AC}(H) \cong \mathbb{Z}_p^a$.
- By induction, assume $G/\mathrm{AC}(G) \cong H/\mathrm{AC}(H)$, identified as Q.

GroupIso: characteristic subgroup chains

To test if two groups G and H of order N are isomorphic.

- Solvable radical of G, $\mathrm{srad}(G)$: the largest solvable normal subgroup.
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are trivial, use [Babai–Codenotti–Q].
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are nontrivial, build a characteristic subgroup chain (see also [Cannon–Holt]).
  1. Compute the solvable radical $\mathrm{srad}(G) \leq G$, and the derived series of $\mathrm{srad}(G)$

     $$1 = S_0 \trianglelefteq S_1 \trianglelefteq S_2 \trianglelefteq \ldots \trianglelefteq S_\ell = \mathrm{srad}(G),$$

     where $S_{i-1} = [S_i, S_i]$ for $i \in [\ell]$.
  2. p: smallest prime dividing $|S_1|$. D: unique p-Sylow subgroup of S_1.
  3. $\mathrm{AC}(G) := \{x \in D \mid x^p = 1\}$ is elementary abelian.
  4. Inductively, construct a chain of characteristic subgroups

     $$1 = T_0 \trianglelefteq T_1 \trianglelefteq T_2 \trianglelefteq \ldots \trianglelefteq T_s = \mathrm{srad}(G) \trianglelefteq G,$$

     where every T_i/T_{i-1} is elementary abelian.
- $G \cong H \Rightarrow \mathrm{AC}(G) \cong \mathrm{AC}(H) \cong \mathbb{Z}_p^a$.
- By induction, assume $G/\mathrm{AC}(G) \cong H/\mathrm{AC}(H)$, identified as Q.

GroupIso: characteristic subgroup chains

To test if two groups G and H of order N are isomorphic.

- Solvable radical of G, $\mathrm{srad}(G)$: the largest solvable normal subgroup.
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are trivial, use [Babai–Codenotti–Q].
- If both $\mathrm{srad}(G)$ and $\mathrm{srad}(H)$ are nontrivial, build a characteristic subgroup chain (see also [Cannon–Holt]).
  1. Compute the solvable radical $\mathrm{srad}(G) \leq G$, and the derived series of $\mathrm{srad}(G)$

     $$1 = S_0 \trianglelefteq S_1 \trianglelefteq S_2 \trianglelefteq \ldots \trianglelefteq S_\ell = \mathrm{srad}(G),$$

     where $S_{i-1} = [S_i, S_i]$ for $i \in [\ell]$.
  2. p: smallest prime dividing $|S_1|$. D: unique p-Sylow subgroup of S_1.
  3. $\mathrm{AC}(G) := \{x \in D \mid x^p = 1\}$ is elementary abelian.
  4. Inductively, construct a chain of characteristic subgroups

     $$1 = T_0 \trianglelefteq T_1 \trianglelefteq T_2 \trianglelefteq \ldots \trianglelefteq T_s = \mathrm{srad}(G) \trianglelefteq G,$$

     where every T_i/T_{i-1} is elementary abelian.
- $G \cong H \Rightarrow \mathrm{AC}(G) \cong \mathrm{AC}(H) \cong \mathbb{Z}_p^a$.
- By induction, assume $G/\mathrm{AC}(G) \cong H/\mathrm{AC}(H)$, identified as Q.

GroupIso: group extensions and cohomology

- We have two groups G and H as extensions of $A \cong \mathbb{Z}_p^a$ by Q.
- An extension of A by Q is determined by two pieces of correlated data.

GroupIso: group extensions and cohomology

- We have two groups G and H as extensions of $A \cong \mathbb{Z}_p^a$ by Q.
- An extension of A by Q is determined by two pieces of correlated data.
  - Group action: $\theta : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
  - 2-cocycle: $f_s : Q \times Q \to A$, built from a section $s : Q \to G$.

GroupIso: group extensions and cohomology

- We have two groups G and H as extensions of $A \cong \mathbb{Z}_p^a$ by Q.
- An extension of A by Q is determined by two pieces of correlated data.
  - Group action: $\theta : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
  - 2-cocycle: $f_s : Q \times Q \to A$, built from a section $s : Q \to G$.

Lemma (Criterion lemma, [Holt–Eick–O'Brien, Grochow–Q])

$G \cong H$ if and only if there is a pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ satisfying both conditions.

- *It matches the actions, meaning $\alpha\theta_G(q)\alpha^{-1} = \theta_H(\sigma(q))$ for every $q \in Q$.*
- *On cohomology, the same pair carries the extension class $[f_G]$ to $[f_H]$.*

GroupIso: group extensions and cohomology

- We have two groups G and H as extensions of $A \cong \mathbb{Z}_p^a$ by Q.
- An extension of A by Q is determined by two pieces of correlated data.
  - Group action: $\theta : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
  - 2-cocycle: $f_s : Q \times Q \to A$, built from a section $s : Q \to G$.

Lemma (Criterion lemma, [Holt–Eick–O'Brien, Grochow–Q])

$G \cong H$ if and only if there is a pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ satisfying both conditions.

- *It matches the actions, meaning $\alpha\theta_G(q)\alpha^{-1} = \theta_H(\sigma(q))$ for every $q \in Q$.*
- *On cohomology, the same pair carries the extension class $[f_G]$ to $[f_H]$.*

GroupIso: a closer look at action compatibility

- Consider two actions of Q on $A \cong \mathbb{Z}_p^a$ by $\theta, \theta' : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
- List the images of θ, θ' as $(A_1, \ldots, A_{|Q|})$ and $(B_1, \ldots, B_{|Q|})$.
- Two actions are compatible if there exists $\sigma \in \mathrm{Aut}(Q)$, $T \in \mathrm{GL}(a, p)$, such that for any $i \in [|Q|]$,

$$A_i = T B_{\sigma(i)} T^{-1}.$$

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_p Q$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.

GroupIso: a closer look at action compatibility

- Consider two actions of Q on $A \cong \mathbb{Z}_p^a$ by $\theta, \theta' : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
- List the images of θ, θ' as $(A_1, \ldots, A_{|Q|})$ and $(B_1, \ldots, B_{|Q|})$.
- Two actions are compatible if there exists $\sigma \in \mathrm{Aut}(Q)$, $T \in \mathrm{GL}(a, p)$, such that for any $i \in [|Q|]$,

$$A_i = T B_{\sigma(i)} T^{-1}.$$

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_p Q$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.

GroupIso: a closer look at action compatibility

- Consider two actions of Q on $A \cong \mathbb{Z}_p^a$ by $\theta, \theta' : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
- List the images of θ, θ' as $(A_1, \ldots, A_{|Q|})$ and $(B_1, \ldots, B_{|Q|})$.
- Two actions are compatible if there exists $\sigma \in \mathrm{Aut}(Q)$, $T \in \mathrm{GL}(a, p)$, such that for any $i \in [|Q|]$,
$$A_i = T B_{\sigma(i)} T^{-1}.$$
- The module viewpoint: θ and θ' give rise to $\mathbb{F}_p Q$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.

GroupIso: a closer look at action compatibility

- Consider two actions of Q on $A \cong \mathbb{Z}_p^a$ by $\theta, \theta' : Q \to \mathrm{Aut}(A) \cong \mathrm{GL}(a, p)$.
- List the images of θ, θ' as $(A_1, \ldots, A_{|Q|})$ and $(B_1, \ldots, B_{|Q|})$.
- Two actions are compatible if there exists $\sigma \in \mathrm{Aut}(Q)$, $T \in \mathrm{GL}(a, p)$, such that for any $i \in [|Q|]$,
$$A_i = T B_{\sigma(i)} T^{-1}.$$
- The module viewpoint: θ and θ' give rise to $\mathbb{F}_p Q$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
  - Decompose into simple summands, compare multiplicities, use a singly exponential set-transporter algorithm.

Action compatibility: from semisimple to general

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_p Q$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
- M is not semisimple $\iff$ the radical of M, $\mathrm{rad}(M)$, is non-trivial and proper.
- Let $U = \mathrm{rad}(M)$ and $V = M/\mathrm{rad}(M)$. Then actions on M are of the format
$$\begin{bmatrix} M_U & T_{U,V} \\ 0 & M_V \end{bmatrix}.$$
- A natural strategy:

Action compatibility: from semisimple to general

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_pQ$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
- M is not semisimple $\iff$ the radical of M, $\mathrm{rad}(M)$, is non-trivial and proper.
  - $\mathrm{rad}(M)$: the intersection of all maximal proper $\mathbb{F}_pQ$ invariant subspaces.
  - $\mathrm{rad}(M) = J(\mathbb{F}_pQ)M$, computable by [Chistov–Ivanyos–Karpinski].
- Let $U = \mathrm{rad}(M)$ and $V = M/\mathrm{rad}(M)$. Then actions on M are of the format

$$\begin{bmatrix} M_U & T_{U,V} \\ 0 & M_V \end{bmatrix}.$$

- A natural strategy:

Action compatibility: from semisimple to general

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_pQ$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
- M is not semisimple $\iff$ the radical of M, $\mathrm{rad}(M)$, is non-trivial and proper.
  - $\mathrm{rad}(M)$: the intersection of all maximal proper $\mathbb{F}_pQ$ invariant subspaces.
  - $\mathrm{rad}(M) = J(\mathbb{F}_pQ)M$, computable by [Chistov–Ivanyos–Karpinski].
- Let $U = \mathrm{rad}(M)$ and $V = M/\mathrm{rad}(M)$. Then actions on M are of the format

$$\begin{bmatrix} M_U & T_{U,V} \\ 0 & M_V \end{bmatrix}.$$

- A natural strategy:

Action compatibility: from semisimple to general

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_pQ$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
- M is not semisimple $\iff$ the radical of M, $\mathrm{rad}(M)$, is non-trivial and proper.
  - $\mathrm{rad}(M)$: the intersection of all maximal proper $\mathbb{F}_pQ$ invariant subspaces.
  - $\mathrm{rad}(M) = J(\mathbb{F}_pQ)M$, computable by [Chistov–Ivanyos–Karpinski].
- Let $U = \mathrm{rad}(M)$ and $V = M/\mathrm{rad}(M)$. Then actions on M are of the format
$$\begin{bmatrix} M_U & T_{U,V} \\ 0 & M_V \end{bmatrix}.$$
- A natural strategy:
  1. Solve the radical part by restricting to U.
  2. Compare the semisimple quotients using the Babai–Qiao method.
  3. How about the glue ($T_{U,V}$) part?

Action compatibility: from semisimple to general

- The module viewpoint: θ and θ' give rise to $\mathbb{F}_pQ$ modules M and M'.
- If M is semisimple, use the Babai–Qiao method.
- M is not semisimple $\iff$ the radical of M, $\mathrm{rad}(M)$, is non-trivial and proper.
  - $\mathrm{rad}(M)$: the intersection of all maximal proper $\mathbb{F}_pQ$ invariant subspaces.
  - $\mathrm{rad}(M) = J(\mathbb{F}_pQ)M$, computable by [Chistov–Ivanyos–Karpinski].
- Let $U = \mathrm{rad}(M)$ and $V = M/\mathrm{rad}(M)$. Then actions on M are of the format
$$\begin{bmatrix} M_U & T_{U,V} \\ 0 & M_V \end{bmatrix}.$$
- A natural strategy:
  1. Solve the radical part by restricting to U.
  2. Compare the semisimple quotients using the Babai–Qiao method.
  3. How about the glue ($T_{U,V}$) part?

The glue part: cohomology transporter problem

- Choose a linear section $s : V \to M$. The glue block gives

$$z : Q \to X = \mathrm{Hom}_{\mathbb{F}_p}(V, U), \qquad z(q) = T_{U,V}(q) M_V(q)^{-1}.$$

- Multiplying the block matrices gives the 1-cocycle identity

$$z(qr) = z(q) + q \cdot z(r), \qquad q \cdot b = M_U(q) b M_V(q)^{-1} \quad (b \in X).$$

- Replacing s by $s + b$ adds the coboundary $q \mapsto q \cdot b - b$ to z. Thus the glue determines a class $[z] \in H^1(Q, X)$. Define $[z']$ similarly for M'.
- Among the layer isomorphisms $u : U \to U'$ and $v : V \to V'$ compatible with the same $\sigma \in \mathrm{Aut}(Q)$, find all triples carrying $[z]$ to $[z']$.

$$[\, q \mapsto u\, z(\sigma^{-1}(q))\, v^{-1} \,] = [z'].$$

Exactly these triples lift to compatible maps $M \to M'$.

The glue part: cohomology transporter problem

- Choose a linear section $s : V \to M$. The glue block gives

$$z : Q \to X = \mathrm{Hom}_{\mathbb{F}_p}(V, U), \qquad z(q) = T_{U,V}(q) M_V(q)^{-1}.$$

- Multiplying the block matrices gives the 1-cocycle identity

$$z(qr) = z(q) + q \cdot z(r), \qquad q \cdot b = M_U(q) b M_V(q)^{-1} \quad (b \in X).$$

- Replacing s by $s + b$ adds the coboundary $q \mapsto q \cdot b - b$ to z. Thus the glue determines a class $[z] \in H^1(Q, X)$. Define $[z']$ similarly for M'.
- Among the layer isomorphisms $u : U \to U'$ and $v : V \to V'$ compatible with the same $\sigma \in \mathrm{Aut}(Q)$, find all triples carrying $[z]$ to $[z']$.

$$[q \mapsto u\, z(\sigma^{-1}(q))\, v^{-1}] = [z'].$$

Exactly these triples lift to compatible maps $M \to M'$.

The glue part: cohomology transporter problem

- Choose a linear section $s : V \to M$. The glue block gives

$$z : Q \to X = \mathrm{Hom}_{\mathbb{F}_p}(V, U), \qquad z(q) = T_{U,V}(q) M_V(q)^{-1}.$$

- Multiplying the block matrices gives the 1-cocycle identity

$$z(qr) = z(q) + q \cdot z(r), \qquad q \cdot b = M_U(q) b M_V(q)^{-1} \quad (b \in X).$$

- Replacing s by $s + b$ adds the coboundary $q \mapsto q \cdot b - b$ to z. Thus the glue determines a class $[z] \in H^1(Q, X)$. Define $[z']$ similarly for M'.

- Among the layer isomorphisms $u : U \to U'$ and $v : V \to V'$ compatible with the same $\sigma \in \mathrm{Aut}(Q)$, find all triples carrying $[z]$ to $[z']$.

$$[q \mapsto u\, z(\sigma^{-1}(q))\, v^{-1}] = [z'].$$

Exactly these triples lift to compatible maps $M \to M'$.

The glue part: cohomology transporter problem

- Choose a linear section $s : V \to M$. The glue block gives

$$z : Q \to X = \mathrm{Hom}_{\mathbb{F}_p}(V, U), \qquad z(q) = T_{U,V}(q) M_V(q)^{-1}.$$

- Multiplying the block matrices gives the 1-cocycle identity

$$z(qr) = z(q) + q \cdot z(r), \qquad q \cdot b = M_U(q) b M_V(q)^{-1} \quad (b \in X).$$

- Replacing s by $s + b$ adds the coboundary $q \mapsto q \cdot b - b$ to z. Thus the glue determines a class $[z] \in H^1(Q, X)$. Define $[z']$ similarly for M'.
- Among the layer isomorphisms $u : U \to U'$ and $v : V \to V'$ compatible with the same $\sigma \in \mathrm{Aut}(Q)$, find all triples carrying $[z]$ to $[z']$.

$$[\, q \mapsto u\, z(\sigma^{-1}(q))\, v^{-1} \,] = [z'].$$

Exactly these triples lift to compatible maps $M \to M'$.

The cohomology transporter problem

1. Start with a finite group Q, an $\mathbb{F}_p Q$-module B, and a degree $j \in \{1, 2\}$.
   - The module glue gives $[z] \in H^1(Q, \mathrm{Hom}_{\mathbb{F}_p}(V, U))$.
   - A group extension gives $[f_s] \in H^2(Q, A)$ for its action on A.
2. The input specifies a group Γ acting compatibly on Q and B.
3. On an invariant subquotient $W = N/P$ of $H^j(Q, B)$, each $g \in \Gamma$ induces a linear map L_g. A given shift $c_g \in W$ gives the affine action

$$\psi_g(w) = L_g w + c_g, \qquad c_{gh} = c_g + L_g c_h.$$

The cohomology transporter problem

1. Start with a finite group Q, an $\mathbb{F}_p Q$-module B, and a degree $j \in \{1, 2\}$.
   - The module glue gives $[z] \in H^1(Q, \mathrm{Hom}_{\mathbb{F}_p}(V, U))$.
   - A group extension gives $[f_s] \in H^2(Q, A)$ for its action on A.
2. The input specifies a group Γ acting compatibly on Q and B.
   - For the module glue, after identifying the radical U with U' and the quotient V with V', Γ consists of the triples (u, v, σ) preserving the actions on U and V. They act on Q by σ and on B by $b \mapsto ubv^{-1}$.
3. On an invariant subquotient $W = N/P$ of $H^j(Q, B)$, each $g \in \Gamma$ induces a linear map L_g. A given shift $c_g \in W$ gives the affine action

$$\psi_g(w) = L_g w + c_g, \qquad c_{gh} = c_g + L_g c_h.$$

The cohomology transporter problem

1. Start with a finite group Q, an $\mathbb{F}_p Q$-module B, and a degree $j \in \{1, 2\}$.
   - The module glue gives $[z] \in H^1(Q, \mathrm{Hom}_{\mathbb{F}_p}(V, U))$.
   - A group extension gives $[f_s] \in H^2(Q, A)$ for its action on A.
2. The input specifies a group Γ acting compatibly on Q and B.
   - For the module glue, after identifying the radical U with U' and the quotient V with V', Γ consists of the triples (u, v, σ) preserving the actions on U and V. They act on Q by σ and on B by $b \mapsto ubv^{-1}$.
3. On an invariant subquotient $W = N/P$ of $H^j(Q, B)$, each $g \in \Gamma$ induces a linear map L_g. A given shift $c_g \in W$ gives the affine action

$$\psi_g(w) = L_g w + c_g, \qquad c_{gh} = c_g + L_g c_h.$$

   - For the module glue, take $W = H^1(Q, B)$ and $c_g = 0$.

The cohomology transporter problem

1. Start with a finite group Q, an $\mathbb{F}_p Q$-module B, and a degree $j \in \{1, 2\}$.
   - The module glue gives $[z] \in H^1(Q, \mathrm{Hom}_{\mathbb{F}_p}(V, U))$.
   - A group extension gives $[f_s] \in H^2(Q, A)$ for its action on A.
2. The input specifies a group Γ acting compatibly on Q and B.
   - For the module glue, after identifying the radical U with U' and the quotient V with V', Γ consists of the triples (u, v, σ) preserving the actions on U and V. They act on Q by σ and on B by $b \mapsto ubv^{-1}$.
3. On an invariant subquotient $W = N/P$ of $H^j(Q, B)$, each $g \in \Gamma$ induces a linear map L_g. A given shift $c_g \in W$ gives the affine action

$$\psi_g(w) = L_g w + c_g, \qquad c_{gh} = c_g + L_g c_h.$$

   - For the module glue, take $W = H^1(Q, B)$ and $c_g = 0$.

Problem (Cohomology Transporter)

Given $x, y \in W$, *compute* $\mathrm{Trans}_{\Gamma,\psi}(x, y) = \{g \in \Gamma : \psi_g(x) = y\}$.
Return no solution, or one solution and generators of the subgroup fixing x.

How cohomology transporter enters GroupIso

- For two extensions with kernel A and quotient Q, we use

$$H^1 \text{ transport} \xrightarrow{\text{glue}} \text{action compatibility} \xrightarrow{(\alpha,\sigma)} H^2 \text{ transport.}$$

- By the Criterion Lemma, a compatible pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ lifts to an isomorphism $G \to H$ exactly when it takes $[f_G]$ to $[f_H]$ in $H^2(Q, A)$.
- To compare two classes in $H^j(R, B)$, recurse on the group and the module.

Back to GroupIso: Recursively quotient by the selected AC subgroups to reach the [Babai–Codenotti–Qiao] base case, then match actions and extension classes to lift isomorphisms back to G and H.

How cohomology transporter enters GroupIso

- For two extensions with kernel A and quotient Q, we use

$$H^1 \text{ transport} \xrightarrow{\text{glue}} \text{action compatibility} \xrightarrow{(\alpha,\sigma)} H^2 \text{ transport.}$$

- By the Criterion Lemma, a compatible pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ lifts to an isomorphism $G \to H$ exactly when it takes $[f_G]$ to $[f_H]$ in $H^2(Q, A)$.
- To compare two classes in $H^j(R, B)$, recurse on the group and the module.

Back to GroupIso: Recursively quotient by the selected AC subgroups to reach the [Babai–Codenotti–Qiao] base case, then match actions and extension classes to lift isomorphisms back to G and H.

How cohomology transporter enters GroupIso

- For two extensions with kernel A and quotient Q, we use

$$H^1 \text{ transport} \xrightarrow{\text{glue}} \text{action compatibility} \xrightarrow{(\alpha,\sigma)} H^2 \text{ transport.}$$

- By the Criterion Lemma, a compatible pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ lifts to an isomorphism $G \to H$ exactly when it takes $[f_G]$ to $[f_H]$ in $H^2(Q, A)$.
- To compare two classes in $H^j(R, B)$, recurse on the group and the module.
  - On the group side, match the restrictions to a characteristic subgroup $S \leq R$, then work with the smaller quotient R/S.
  - On the module side, keep R fixed and, when needed, break the comparison into smaller ones using a filtration of B.

Back to GroupIso: Recursively quotient by the selected AC subgroups to reach the [Babai–Codenotti–Qiao] base case, then match actions and extension classes to lift isomorphisms back to G and H.

How cohomology transporter enters GroupIso

- For two extensions with kernel A and quotient Q, we use

$$H^1 \text{ transport} \xrightarrow{\text{glue}} \text{action compatibility} \xrightarrow{(\alpha,\sigma)} H^2 \text{ transport.}$$

- By the Criterion Lemma, a compatible pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ lifts to an isomorphism $G \to H$ exactly when it takes $[f_G]$ to $[f_H]$ in $H^2(Q, A)$.
- To compare two classes in $H^j(R, B)$, recurse on the group and the module.
  - On the group side, match the restrictions to a characteristic subgroup $S \leq R$, then work with the smaller quotient R/S.
  - On the module side, keep R fixed and, when needed, break the comparison into smaller ones using a filtration of B.

Back to GroupIso: Recursively quotient by the selected AC subgroups to reach the [Babai–Codenotti–Qiao] base case, then match actions and extension classes to lift isomorphisms back to G and H.

How cohomology transporter enters GroupIso

- For two extensions with kernel A and quotient Q, we use

$$H^1 \text{ transport} \xrightarrow{\text{glue}} \text{action compatibility} \xrightarrow{(\alpha,\sigma)} H^2 \text{ transport}.$$

- By the Criterion Lemma, a compatible pair $(\alpha, \sigma) \in \mathrm{Aut}(A) \times \mathrm{Aut}(Q)$ lifts to an isomorphism $G \to H$ exactly when it takes $[f_G]$ to $[f_H]$ in $H^2(Q, A)$.
- To compare two classes in $H^j(R, B)$, recurse on the group and the module.
  - On the group side, match the restrictions to a characteristic subgroup $S \leq R$, then work with the smaller quotient R/S.
  - On the module side, keep R fixed and, when needed, break the comparison into smaller ones using a filtration of B.

Back to GroupIso: Recursively quotient by the selected AC subgroups to reach the [Babai–Codenotti–Qiao] base case, then match actions and extension classes to lift isomorphisms back to G and H.

Section 5

Summary and Open Problems

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Summary of this paper

The main results are as follows.

- GroupIso in time polynomial in the group order.
- $\ell \times m \times n$ 3-TensorIso over $\mathbb{F}_q$ in time $q^{O(\ell+m+n)}$.
- Matrix tuple congruence (MTC) for $\mathrm{M}(\ell \times \ell, 2)^n$ in time $2^{O(\ell)} \cdot \mathrm{poly}(n)$.

The main reflections on the techniques are as follows.

1. The results build on many works in TCS and CGT, including several conceptual and technical innovations in the past two decades.
2. The strategy for TensorIso has some similarities with Babai's breakthrough for GraphIso: a hybrid of group-theoretic and combinatorial-type techniques.
3. The strategy for GroupIso follows the group extension and cohomology formulation [Grochow–Q], utilising a characteristic subgroup chain [Cannon–Holt], and relying on the result on semisimple groups [Babai–Codenotti–Q].

Open problems

The results in this paper opens up more opportunities for furture research on isomorphism problems.

1. GraphIso in P.
2. GroupIso or TensorIso in NC.
3. Linear code equivalence via matrix group algorithms.
4. TensorIso for $n \times n \times n$ tensors over $\mathbb{F}_q$ in time $q^{o(n)}$?
5. TensorIso over $\mathbb{C}$ in AM $\cap$ coAM?

Question and Answer

Thank you so much!



https://group-iso.github.io/