Constraints, Symmetry, and Complexity
Part 2

Andrei Krokhin

Mantra

- Symmetries of solution spaces are relevant for complexity
- Lack of symmetries $\Rightarrow$ hardness
- Symmetries $\Rightarrow$ efficient algorithms
- Symmetries of higher dimension/arity are important

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?
- Old (in)famous problem: for any fixed const $3 \leq k \leq c$, is it **NP**-hard to find a c-colouring of a given k-colourable graph?

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?
- Old (in)famous problem: for any fixed const $3 \leq k \leq c$, is it **NP**-hard to find a c-colouring of a given k-colourable graph?
- 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?
- Old (in)famous problem: for any fixed const $3 \leq k \leq c$, is it **NP**-hard to find a c-colouring of a given k-colourable graph?
- 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.
- k vs. $c = 2k - 2$ is **NP**-hard [Brakensiek, Guruswami'16]

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?
- Old (in)famous problem: for any fixed const $3 \leq k \leq c$, is it **NP**-hard to find a c-colouring of a given k-colourable graph?
- 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.
- k vs. $c = 2k - 2$ is **NP**-hard [Brakensiek, Guruswami'16]
- k vs. $2^{\Omega(k^{1/3})}$ is **NP**-hard for large enough k [Huang'13]

Motivational example: Approximate graph colouring

Classical fact:

- For any fixed $k \geq 3$, it's **NP**-hard to find a k-colouring of a given k-colourable graph.

Natural question:

- How many colours are needed to efficiently find a colouring?
- Old (in)famous problem: for any fixed const $3 \leq k \leq c$, is it **NP**-hard to find a c-colouring of a given k-colourable graph?
- 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.
- k vs. $c = 2k - 2$ is **NP**-hard [Brakensiek, Guruswami'16]
- k vs. $2^{\Omega(k^{1/3})}$ is **NP**-hard for large enough k [Huang'13]
- **NP**-hard for all $3 \leq k \leq c$, assuming non-standard (perfect completeness) variants of the UGC [Dinur, Mossel, Regev'09]

Approximating CSP

Goal: Given a CSP instance, find an approximate solution.

Options:

Approximating CSP

Goal: Given a CSP instance, find an approximate solution.

Options:

1. Can violate some constraints, try to satisfy as many as poss
   - Full understanding modulo UGC [Raghavendra'08]

Approximating CSP

Goal: Given a CSP instance, find an approximate solution.

Options:

1. Can violate some constraints, try to satisfy as many as poss
   - Full understanding modulo UGC [Raghavendra'08]
2. Relax the constraints themselves (**this talk**)

Approximating CSP

Goal: Given a CSP instance, find an approximate solution.

Options:

1. Can violate some constraints, try to satisfy as many as poss
   — Full understanding modulo UGC [Raghavendra'08]
2. Relax the constraints themselves (**this talk**)
3. Perhaps do both simultaneously — some time in the future

Promise CSP (PCSP)

Fix structures $\mathbb{A} = (A; R_1, \ldots, R_n)$ and $\mathbb{B} = (B; S_1, \ldots, S_n)$ s.t.

1. $arity(R_i) = arity(S_i)$ for all i

Promise CSP (PCSP)

Fix structures $\mathbb{A} = (A; R_1, \ldots, R_n)$ and $\mathbb{B} = (B; S_1, \ldots, S_n)$ s.t.

1. $arity(R_i) = arity(S_i)$ for all i
2. $\exists\ h : \mathbb{A} \to \mathbb{B}$ (homomorphism, i.e. $h(R_i) \subseteq S_i$)
   - — special case: $A \subseteq B$, $h(x) = x$, and then $R_i \subseteq S_i$.

Promise CSP (PCSP)

Fix structures $\mathbb{A} = (A; R_1, \ldots, R_n)$ and $\mathbb{B} = (B; S_1, \ldots, S_n)$ s.t.

1. $arity(R_i) = arity(S_i)$ for all i
2. $\exists\ h : \mathbb{A} \to \mathbb{B}$ (homomorphism, i.e. $h(R_i) \subseteq S_i$)
   - — special case: $A \subseteq B$, $h(x) = x$, and then $R_i \subseteq S_i$.

Consider pairs of CSP instances:

$$\begin{aligned} \Phi_{strict} &= R_1(x,y,z), R_1(z,y,w), R_2(z), R_3(x,w), R_3(y,y) \\ \Phi_{relaxed} &= S_1(x,y,z), S_1(z,y,w), S_2(z), S_3(x,w), S_3(y,y) \end{aligned}$$

Promise CSP (PCSP)

Fix structures $\mathbb{A} = (A; R_1, \ldots, R_n)$ and $\mathbb{B} = (B; S_1, \ldots, S_n)$ s.t.

1. $arity(R_i) = arity(S_i)$ for all i
2. $\exists\ h : \mathbb{A} \to \mathbb{B}$ (homomorphism, i.e. $h(R_i) \subseteq S_i$)
   - special case: $A \subseteq B$, $h(x) = x$, and then $R_i \subseteq S_i$.

Consider pairs of CSP instances:

$$\begin{aligned} \Phi_{strict} &= R_1(x,y,z), R_1(z,y,w), R_2(z), R_3(x,w), R_3(y,y) \\ \Phi_{relaxed} &= S_1(x,y,z), S_1(z,y,w), S_2(z), S_3(x,w), S_3(y,y) \end{aligned}$$

Because of h, have Φ_{strict} is sat $\Rightarrow$ $\Phi_{relaxed}$ is sat

Promise CSP (PCSP)

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Decision version)

Accept if Φ_{strict} is sat, reject if $\Phi_{relaxed}$ is unsat.

Promise CSP (PCSP)

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Decision version)

Accept if Φ_{strict} is sat, reject if $\Phi_{relaxed}$ is unsat.

Obvious facts:

- For any $\mathbb{A}$, $\mathrm{PCSP}(\mathbb{A}, \mathbb{A}) = \mathrm{CSP}(\mathbb{A})$
- $\mathrm{CSP}(\mathbb{A})$ or $\mathrm{CSP}(\mathbb{B})$ is in **P** $\Rightarrow$ $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in **P**.

Promise CSP (PCSP)

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Decision version)

Accept if Φ_{strict} is sat, reject if $\Phi_{relaxed}$ is unsat.

Obvious facts:

- For any $\mathbb{A}$, $\mathrm{PCSP}(\mathbb{A}, \mathbb{A}) = \mathrm{CSP}(\mathbb{A})$
- $\mathrm{CSP}(\mathbb{A})$ or $\mathrm{CSP}(\mathbb{B})$ is in **P** $\Rightarrow$ $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in **P**.

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Search version)

Given a satisfiable Φ_{strict}, find a solution for $\Phi_{relaxed}$.

Promise CSP (PCSP)

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Decision version)

Accept if Φ_{strict} is sat, reject if $\Phi_{relaxed}$ is unsat.

Obvious facts:

- For any $\mathbb{A}$, $\mathrm{PCSP}(\mathbb{A}, \mathbb{A}) = \mathrm{CSP}(\mathbb{A})$
- $\mathrm{CSP}(\mathbb{A})$ or $\mathrm{CSP}(\mathbb{B})$ is in **P** $\Rightarrow$ $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in **P**.

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Search version)

Given a satisfiable Φ_{strict}, find a solution for $\Phi_{relaxed}$.

Remark: there is an obvious reduction from decision to search.

Promise CSP (PCSP)

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Decision version)

Accept if Φ_{strict} is sat, reject if $\Phi_{relaxed}$ is unsat.

Obvious facts:

- For any $\mathbb{A}$, $\mathrm{PCSP}(\mathbb{A}, \mathbb{A}) = \mathrm{CSP}(\mathbb{A})$
- $\mathrm{CSP}(\mathbb{A})$ or $\mathrm{CSP}(\mathbb{B})$ is in **P** $\Rightarrow$ $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$ is in **P**.

Definition ($\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$, Search version)

Given a satisfiable Φ_{strict}, find a solution for $\Phi_{relaxed}$.

Remark: there is an obvious reduction from decision to search.

Conjecture

The search and the decision problems are always equivalent.

Example: Approximate hypergraph colouring

- Recall k-NAE:
  $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
- Essentially, this is k-colouring for 3-uniform hypergraphs.

Example: Approximate hypergraph colouring

- Recall k-NAE:
  $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
- Essentially, this is k-colouring for 3-uniform hypergraphs.
- Natural to consider (k-NAE, c-NAE).

Example: Approximate hypergraph colouring

- Recall k-NAE:
  $\mathbb{A} = ([k]; \{(a, b, c) \in [k]^3 \mid a \neq b \vee a \neq c \vee b \neq c\})$
- Essentially, this is k-colouring for 3-uniform hypergraphs.
- Natural to consider (k-NAE, c-NAE).

Theorem (Dinur, Regev, Smyth'05)

For any $2 \leq k \leq c$, PCSP(k-NAE, c-NAE) is **NP***-hard.*

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Fact (Brakensiek, Guruswami'16)

(1-in-3, NAE)-SAT is in **P**.

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Fact (Brakensiek, Guruswami'16)

(1-in-3, NAE)-SAT is in **P**.

Proof.

1. Replace each $R_{1/3}(x,y,z)$ constraint by $x + y + z = 1$

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Fact (Brakensiek, Guruswami'16)

(1-in-3, NAE)-SAT is in **P**.

Proof.

1. Replace each $R_{1/3}(x,y,z)$ constraint by $x + y + z = 1$
2. Solve the linear system over $\mathbb{Z}$ (can be done in poly-time)

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Fact (Brakensiek, Guruswami'16)

(1-in-3, NAE)-SAT is in **P**.

Proof.

1. Replace each $R_{1/3}(x,y,z)$ constraint by $x + y + z = 1$
2. Solve the linear system over $\mathbb{Z}$ (can be done in poly-time)
3. Round the solution: positive $\mapsto 1$, non-positive $\mapsto 0$

Example: (1-in-3, NAE)-SAT

- 1-in-3 is $R_{1/3} = \{(0,0,1),(0,1,0),(1,0,0)\}$
- NAE is 2-NAE as in hypergraph 2-colouring
- Both 1-in-3-SAT and NAE-SAT are standard **NP**-hard CSPs

Fact (Brakensiek, Guruswami'16)

(1-in-3, NAE)-SAT is in **P**.

Proof.

1. Replace each $R_{1/3}(x,y,z)$ constraint by $x + y + z = 1$
2. Solve the linear system over $\mathbb{Z}$ (can be done in poly-time)
3. Round the solution: positive $\mapsto 1$, non-positive $\mapsto 0$
4. 1-in-3 instance sat $\Rightarrow$ lin system sat $\Rightarrow$ NAE instance sat

How to prove tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

All known proofs work (more or less) like this:

- By def, have a homomorphism $h : \mathbb{A} \to \mathbb{B}$

How to prove tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

All known proofs work (more or less) like this:

- By def, have a homomorphism $h : \mathbb{A} \to \mathbb{B}$
- Find a (possibly infinite) structure $\mathbb{S}$ such that
  - (a) $\mathbb{A} \to \mathbb{S} \to \mathbb{B}$ and (b) $\mathrm{CSP}(\mathbb{S})$ is in **P**.

How to prove tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

All known proofs work (more or less) like this:

- By def, have a homomorphism $h : \mathbb{A} \to \mathbb{B}$
- Find a (possibly infinite) structure $\mathbb{S}$ such that
  - (a) $\mathbb{A} \to \mathbb{S} \to \mathbb{B}$ and (b) $\mathrm{CSP}(\mathbb{S})$ is in **P**.
- Then algorithm for $\mathrm{CSP}(\mathbb{S})$ solves $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

How to prove tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

All known proofs work (more or less) like this:

- By def, have a homomorphism $h : \mathbb{A} \to \mathbb{B}$
- Find a (possibly infinite) structure $\mathbb{S}$ such that
  - (a) $\mathbb{A} \to \mathbb{S} \to \mathbb{B}$ and (b) $\mathrm{CSP}(\mathbb{S})$ is in **P**.
- Then algorithm for $\mathrm{CSP}(\mathbb{S})$ solves $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

For the example from previous slide:

- $\mathbb{A} = (\{0,1\}, R_{1/3})$, $\mathbb{B} = (\{0,1\}, R_{NAE})$, $\mathbb{S} = (\mathbb{Z}; x + y + z = 1)$

How to prove tractability of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

All known proofs work (more or less) like this:

- By def, have a homomorphism $h : \mathbb{A} \to \mathbb{B}$
- Find a (possibly infinite) structure $\mathbb{S}$ such that
  - (a) $\mathbb{A} \to \mathbb{S} \to \mathbb{B}$ and (b) $\mathrm{CSP}(\mathbb{S})$ is in **P**.
- Then algorithm for $\mathrm{CSP}(\mathbb{S})$ solves $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

For the example from previous slide:

- $\mathbb{A} = (\{0,1\}, R_{1/3})$, $\mathbb{B} = (\{0,1\}, R_{NAE})$, $\mathbb{S} = (\mathbb{Z}; x + y + z = 1)$
- For any finite $\mathbb{S}$ with $\mathbb{A} \to \mathbb{S} \to \mathbb{B}$, $\mathrm{CSP}(\mathbb{S})$ is **NP**-c [Barto'18]

Polymorphisms

An operation $f : A^m \to B$ is called a polymorphism from a $R \subseteq A^k$ to $S \subseteq B^k$ if for any k-tuples

$$\begin{array}{ccccccc}
 & f & & f & & f & \\
 & \downarrow & & \downarrow & & \downarrow & \\
(& a_{11} & , \ \ldots \ , & & a_{1k} &) & \in R \\
 & \vdots & & \vdots & & \vdots & \vdots \\
(& a_{m1} & , \ \ldots \ , & & a_{mk} &) & \in R \\
\hline
 & & & & & \Downarrow & \\
(& f(a_{11}, \ldots, a_{m1}) & , \ \ldots \ , & & f(a_{1k}, \ldots, a_{mk}) &) & \in S
\end{array}$$

Polymorphisms

An operation $f : A^m \to B$ is called a polymorphism from a $R \subseteq A^k$ to $S \subseteq B^k$ if for any k-tuples

$$\frac{\begin{array}{ccccccc} & f & & f & & f & \\ & \downarrow & & \downarrow & & \downarrow & \\ (& a_{11} & , \ \ldots \ , & & a_{1k} &) & \in R \\ & \vdots & & \vdots & & \vdots & \vdots \\ (& a_{m1} & , \ \ldots \ , & & a_{mk} &) & \in R \end{array}}{\begin{array}{ccccccc} & & & & & \Downarrow & \\ (& f(a_{11}, \ldots, a_{m1}) & , \ \ldots \ , & & f(a_{1k}, \ldots, a_{mk}) &) & \in S \end{array}}$$

Can do polymorphisms from $\mathbb{A}$ to $\mathbb{B}$. Notation — $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$.

Polymorphisms

An operation $f : A^m \to B$ is called a polymorphism from a $R \subseteq A^k$ to $S \subseteq B^k$ if for any k-tuples

$$\begin{array}{ccccccc} & f & & f & & f & \\ & \downarrow & & \downarrow & & \downarrow & \\ (& a_{11} & , \ \ldots \ , & & a_{1k} &) & \in R \\ & \vdots & & \vdots & & \vdots & \vdots \\ (& a_{m1} & , \ \ldots \ , & & a_{mk} &) & \in R \\ \hline & & & & & \Downarrow & \\ (& f(a_{11}, \ldots, a_{m1}) & , \ \ldots \ , & & f(a_{1k}, \ldots, a_{mk}) &) & \in S \end{array}$$

Can do polymorphisms from $\mathbb{A}$ to $\mathbb{B}$. Notation — $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$.
Example: n-ary functions from $\mathrm{Pol}(\mathbb{K}_k, \mathbb{K}_c) = c$-colourings of $\mathbb{K}_k^n$

Difference from CSP

Can't compose polymorphisms!

If $f(x,y), g(x,y) \in \mathrm{Pol}(\mathbb{A}, \mathbb{B})$, what is $f(g(x,y),z)$?

Difference from CSP

Can't compose polymorphisms!

If $f(x,y), g(x,y) \in \mathrm{Pol}(\mathbb{A}, \mathbb{B})$, what is $f(g(x,y),z)$?

Implications:

1. don't have clones, old structural theory not applicable
2. old way of getting strong symmetries from weak symmetries doesn't work

Difference from CSP

Can't compose polymorphisms!

If $f(x,y), g(x,y) \in \mathrm{Pol}(\mathbb{A}, \mathbb{B})$, what is $f(g(x,y),z)$?

Implications:

1. don't have clones, old structural theory not applicable
2. old way of getting strong symmetries from weak symmetries doesn't work
3. any algorithm that uses repeated applications of polymorphisms doesn't work

Difference from CSP

Can't compose polymorphisms!

If $f(x,y), g(x,y) \in \mathrm{Pol}(\mathbb{A}, \mathbb{B})$, what is $f(g(x,y),z)$?

Implications:

1. don't have clones, old structural theory not applicable
2. old way of getting strong symmetries from weak symmetries doesn't work
3. any algorithm that uses repeated applications of polymorphisms doesn't work

Possible conclusions:

- Things are hopeless for algebra here

Difference from CSP

Can't compose polymorphisms!

If $f(x,y), g(x,y) \in \mathrm{Pol}(\mathbb{A}, \mathbb{B})$, what is $f(g(x,y),z)$?

Implications:

1. don't have clones, old structural theory not applicable
2. old way of getting strong symmetries from weak symmetries doesn't work
3. any algorithm that uses repeated applications of polymorphisms doesn't work

Possible conclusions:

- Things are hopeless for algebra here
- There's a composition-free algebraic theory out there

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A}, \mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A}, \mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Proof is the same as Jeavons' proof of the analogous result for CSPs, goes via simulation (pp-definitions).

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A}, \mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Proof is the same as Jeavons' proof of the analogous result for CSPs, goes via simulation (pp-definitions).

Some results based on this theorem:

- A Boolean PCSP known as $(2+\epsilon)$-$\textsc{Sat}$ is **NP**-hard [AHG'14]

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A}, \mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Proof is the same as Jeavons' proof of the analogous result for CSPs, goes via simulation (pp-definitions).

Some results based on this theorem:

- A Boolean PCSP known as $(2+\epsilon)$-$\textsc{Sat}$ is **NP**-hard [AHG'14]
- **NP**-hardness for some approx graph and (variants of) hypergraph colouring [BG'16-18]

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A}, \mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Proof is the same as Jeavons' proof of the analogous result for CSPs, goes via simulation (pp-definitions).

Some results based on this theorem:

- A Boolean PCSP known as $(2+\epsilon)$-$\textsc{Sat}$ is **NP**-hard [AHG'14]
- **NP**-hardness for some approx graph and (variants of) hypergraph colouring [BG'16-18]
- Classification for Boolean symmetric predicates [BG'18]

Polymorphisms in control

Theorem (Pippenger'02, Brakensiek, Guruswami'18)

If $\mathrm{Pol}(\mathbb{A},\mathbb{B}) \subseteq \mathrm{Pol}(\mathbb{C},\mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C},\mathbb{D})$ *poly-time reduces to* $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$.

Proof is the same as Jeavons' proof of the analogous result for CSPs, goes via simulation (pp-definitions).

Some results based on this theorem:

- A Boolean PCSP known as $(2+\epsilon)$-$\textsc{Sat}$ is **NP**-hard [AHG'14]
- **NP**-hardness for some approx graph and (variants of) hypergraph colouring [BG'16-18]
- Classification for Boolean symmetric predicates [BG'18]

So this can take us some way, but we need a more abstract theory!

Minors and minions

Definition

Let $f : A^n \to B$ and let $\pi : [n] \to [m]$. We say that $g : A^m \to B$ is a minor of f (wrt. π) if $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$.

Meaning: g is obtained from f by identifying and permuting vars (and poss adding dummy vars)

Minors and minions

Definition

Let $f : A^n \to B$ and let $\pi : [n] \to [m]$. We say that $g : A^m \to B$ is a minor of f (wrt. π) if $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$.

Meaning: g is obtained from f by identifying and permuting vars (and poss adding dummy vars)

Fact

Every $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ *is closed under taking minors.*

Minors and minions

Definition

Let $f : A^n \to B$ and let $\pi : [n] \to [m]$. We say that $g : A^m \to B$ is a minor of f (wrt. π) if $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$.

Meaning: g is obtained from f by identifying and permuting vars (and poss adding dummy vars)

Fact

Every $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ *is closed under taking minors.*

Such sets are knowns as minions or clonoids.

Minor conditions

An identity of the form $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(m)})$ is called a minor identity.
It's a special case of a height-1 identity.

Minor conditions

An identity of the form $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(m)})$ is called a minor identity.
It's a special case of a height-1 identity.

Definition

A minor condition is a set of minor identities.

A bipartite minor condition (bmc) is one where the sets of function symbols on LHS and RHS are disjoint.

Minor conditions

An identity of the form $g(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(m)})$ is called a minor identity.
It's a special case of a height-1 identity.

Definition

A minor condition is a set of minor identities.

A bipartite minor condition (bmc) is one where the sets of function symbols on LHS and RHS are disjoint.

Example:

$$\begin{aligned} s(x,y) &= f(x,x,y,y,y,x) \\ s(x,y) &= f(x,y,x,y,x,y) \\ s(x,y) &= f(y,x,x,x,y,y) \end{aligned}$$

Polymorphisms and minor conditions

What determines the complexity of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

- $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? — yes!

Polymorphisms and minor conditions

What determines the complexity of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

- $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? — yes!
- more abstractly, identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$?
  - — Only height-1 (composition-free) identities makes sense!

Polymorphisms and minor conditions

What determines the complexity of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

- $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? — yes!
- more abstractly, identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$?
  - — Only height-1 (composition-free) identities makes sense!
- h1 identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? - yes!

Polymorphisms and minor conditions

What determines the complexity of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

- $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? — yes!
- more abstractly, identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$?
  - — Only height-1 (composition-free) identities makes sense!
- h1 identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? - yes!
- more generally, bmc's sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? - yes!

Theorem (Bulín, AK, Opršal)

If every bipartite minor condition sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ *is also sat in* $\mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *logspace reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

Polymorphisms and minor conditions

What determines the complexity of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$?

- $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? — yes!
- more abstractly, identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$?
  - — Only height-1 (composition-free) identities makes sense!
- h1 identities sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? - yes!
- more generally, bmc's sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$? - yes!

Theorem (Bulín, AK, Opršal)

If every bipartite minor condition sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ *is also sat in* $\mathrm{Pol}(\mathbb{C}, \mathbb{D})$ *then* $\mathrm{PCSP}(\mathbb{C}, \mathbb{D})$ *logspace reduces to* $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$.

bmc's sat in $\mathrm{Pol}(\mathbb{A}, \mathbb{B})$ is the right notion of symmetry for PCSPs!

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Theorem (Bulín, AK, Opršal)

Every bmc sat in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *is sat in* $\mathrm{Pol}(2 - NAE, 458 - NAE)$.

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Theorem (Bulín, AK, Opršal)

Every bmc sat in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *is sat in* $\mathrm{Pol}(2 - NAE, 458 - NAE)$.

By [DRS'05], $\mathrm{PCSP}(2 - NAE, 458 - NAE)$ is **NP**-hard, so ...

Corollary

$\mathrm{PCSP}(\mathbb{K}_3, \mathbb{K}_5)$ *is* **NP***-hard.*

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Theorem (Bulín, AK, Opršal)

Every bmc sat in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *is sat in* $\mathrm{Pol}(2 - NAE, 458 - NAE)$.

By [DRS'05], $\mathrm{PCSP}(2 - NAE, 458 - NAE)$ is **NP**-hard, so ...

Corollary

$\mathrm{PCSP}(\mathbb{K}_3, \mathbb{K}_5)$ *is* **NP***-hard.*

Remarks:

- 458 is the number of binary functions in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Theorem (Bulín, AK, Opršal)

Every bmc sat in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *is sat in* $\mathrm{Pol}(2 - NAE, 458 - NAE)$.

By [DRS'05], $\mathrm{PCSP}(2 - NAE, 458 - NAE)$ is **NP**-hard, so ...

Corollary

$\mathrm{PCSP}(\mathbb{K}_3, \mathbb{K}_5)$ *is* **NP***-hard.*

Remarks:

- 458 is the number of binary functions in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$
- The proof actually works for all $(k, 2k-1)$, not just $(3, 5)$

An application

Recall: 3 vs. 4 colouring is **NP**-hard [GJ76], but 3 vs. 5 is open.

Theorem (Bulín, AK, Opršal)

Every bmc sat in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *is sat in* $\mathrm{Pol}(2 - NAE, 458 - NAE)$.

By [DRS'05], $\mathrm{PCSP}(2 - NAE, 458 - NAE)$ is **NP**-hard, so ...

Corollary

$\mathrm{PCSP}(\mathbb{K}_3, \mathbb{K}_5)$ *is* **NP***-hard.*

Remarks:

- 458 is the number of binary functions in $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$
- The proof actually works for all $(k, 2k-1)$, not just $(3, 5)$
- Both $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ and $\mathrm{Pol}(2 - NAE, 458 - NAE)$ satisfy many non-trivial bmc's

Proof

Lemma (Bulín, AK, Opršal)

To prove the theorem, it is enough to show that $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *does* <u>not</u> *satisfy the following bmc:*

$$\begin{aligned} s(x,y) &= f(x,x,y,y,y,x) \\ s(x,y) &= f(x,y,x,y,x,y) \\ s(x,y) &= f(y,x,x,x,y,y) \end{aligned}$$

Proof

Lemma (Bulín, AK, Opršal)

To prove the theorem, it is enough to show that $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ *does* <u>not</u> *satisfy the following bmc:*

$$\begin{aligned} s(x,y) &= f(x,x,y,y,y,x) \\ s(x,y) &= f(x,y,x,y,x,y) \\ s(x,y) &= f(y,x,x,x,y,y) \end{aligned}$$

Let's show that $\mathrm{Pol}(\mathbb{K}_3, \mathbb{K}_5)$ contains no such f. What would such an f be? - a 5-colouring of $\mathbb{K}_3^6$ that that gives the same colour to some specified vertices.

End of proof

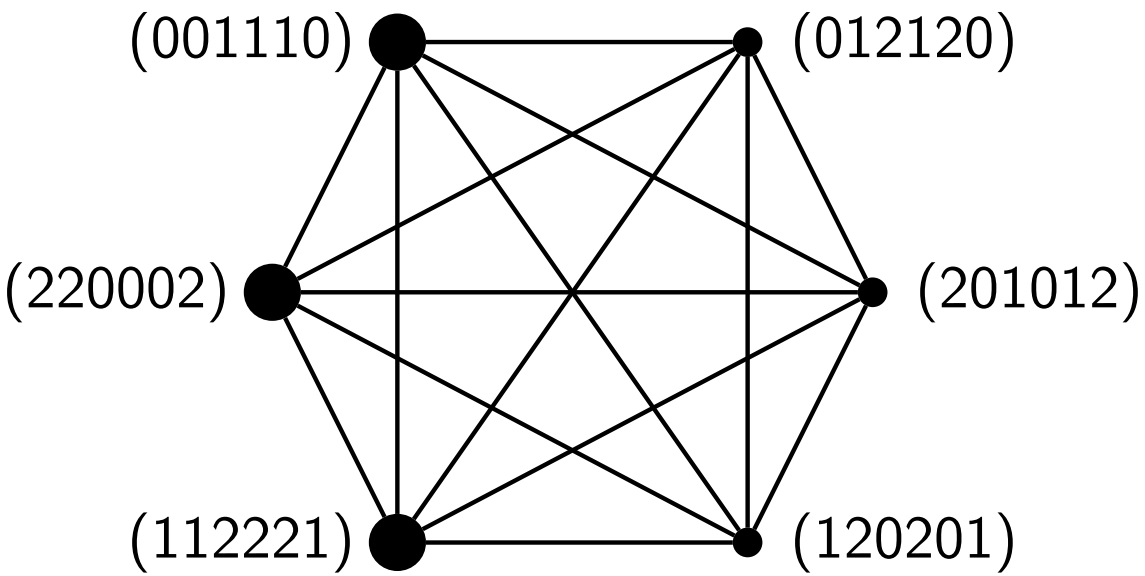
Take $\mathbb{K}_3^6$ and, for each pair $(x, y) \in [3]^2$, glue together 3 vertices: (x, x, y, y, y, x), (x, y, x, y, x, y), (y, x, x, x, y, y).

Need to show: this graph is not 5-colourable.

End of proof

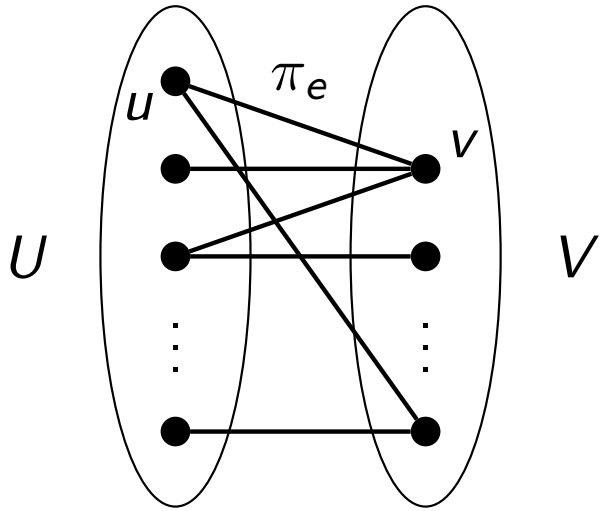
Take $\mathbb{K}_3^6$ and, for each pair $(x, y) \in [3]^2$, glue together 3 vertices: (x, x, y, y, y, x), (x, y, x, y, x, y), (y, x, x, x, y, y).

Need to show: this graph is not 5-colourable. But it contains the following

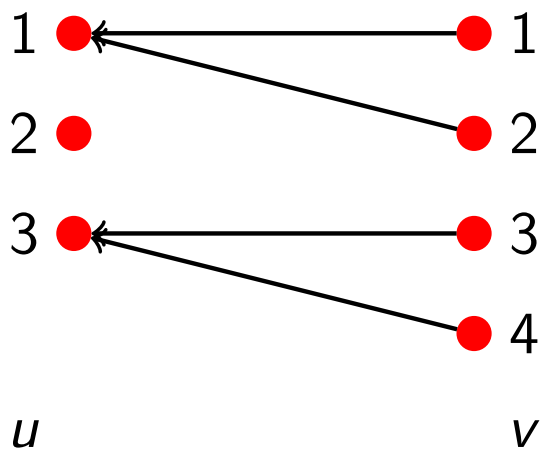


The Label Cover problem

The Label Cover, $\mathrm{LC}(N)$ problem is a CSP that looks like this:



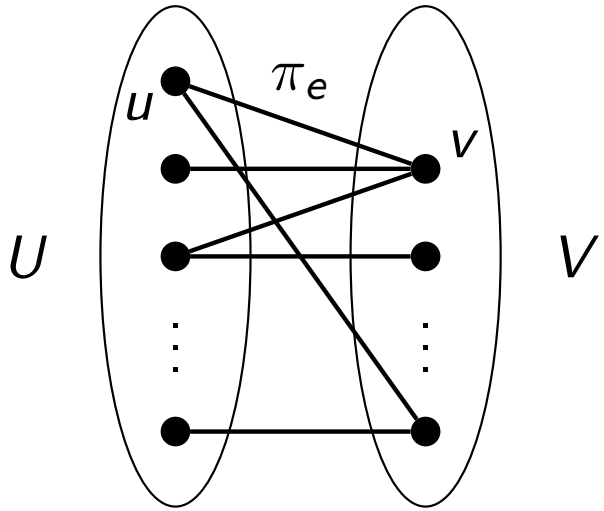
typical instance of $\mathrm{LC}(4)$



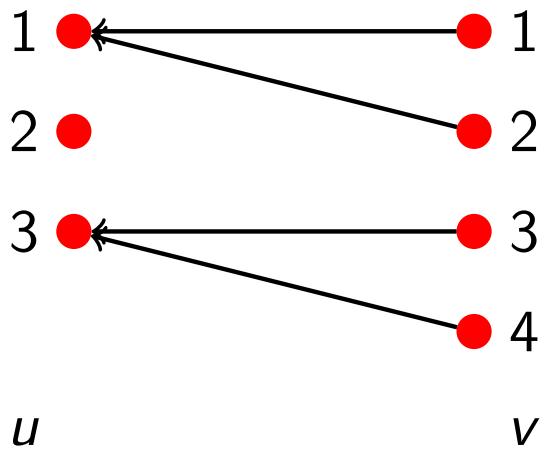
typical constraint π_e

The Label Cover problem

The Label Cover, $\mathrm{LC}(N)$ problem is a CSP that looks like this:



typical instance of $\mathrm{LC}(4)$



typical constraint π_e

Fact

For any $N \geq 3$, $\mathrm{LC}(N)$ is **NP***-complete.*

Label Cover is an algebraic problem

Definition (Problem $\mathrm{MC}(N)$)

Given a bmc Σ involving functions of arity $\leq N$, decide whether Σ is trivial (i.e. satisfiable in projections).

Label Cover is an algebraic problem

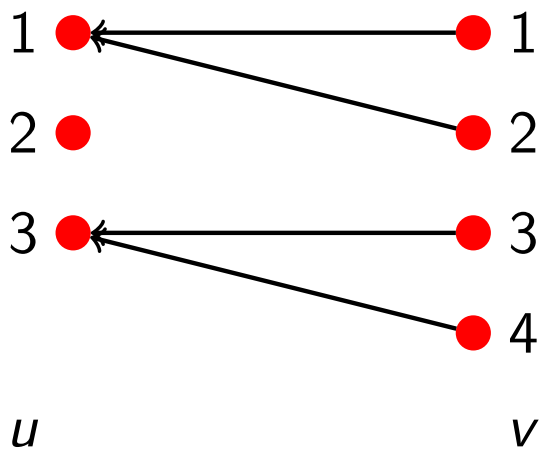
Definition (Problem $\mathrm{MC}(N)$)

Given a bmc Σ involving functions of arity $\leq N$, decide whether Σ is trivial (i.e. satisfiable in projections).

Fact

$\mathrm{LC}(N)$ *and* $\mathrm{MC}(N)$ *are the same problem.*

$$g_u(x_1, x_2, x_3) = f_v(x_1, x_1, x_3, x_3)$$



minor identity $\longleftrightarrow$ constraint π_e

Label Cover is an algebraic problem

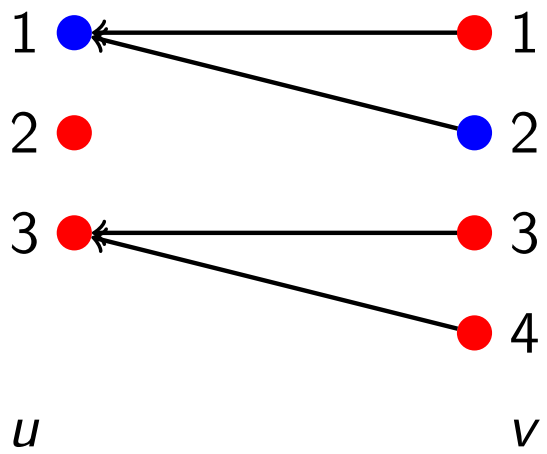
Definition (Problem $\mathrm{MC}(N)$)

Given a bmc Σ involving functions of arity $\leq N$, decide whether Σ is trivial (i.e. satisfiable in projections).

Fact

$\mathrm{LC}(N)$ *and* $\mathrm{MC}(N)$ *are the same problem.*

$$g_u(\mathbf{x}_1, x_2, x_3) = f_v(x_1, \mathbf{x}_1, x_3, x_3)$$



minor identity $\longleftrightarrow$ constraint π_e

Gap Label Cover (GLC)

Definition

For $\epsilon > 0$ and integer N, the problem $\mathrm{GLC}_{1,\epsilon}(N)$ is, given an instance of $\mathrm{LC}(N)$, to distinguish between two cases:

- (yes) all constraints can be satisfied, and
- (no) no assignment satisfies ϵ-fraction of constraints

Gap Label Cover (GLC)

Definition

For $\epsilon > 0$ and integer N, the problem $\mathrm{GLC}_{1,\epsilon}(N)$ is, given an instance of $\mathrm{LC}(N)$, to distinguish between two cases:

- (yes) all constraints can be satisfied, and
- (no) no assignment satisfies ϵ-fraction of constraints

Theorem (PCP + Raz's Parallel repetition)

For any $\epsilon > 0$, there is N such that $\mathrm{GLC}_{1,\epsilon}(N)$ is **NP***-hard.*

Gap Label Cover (GLC)

Definition

For $\epsilon > 0$ and integer N, the problem $\mathrm{GLC}_{1,\epsilon}(N)$ is, given an instance of $\mathrm{LC}(N)$, to distinguish between two cases:

- (yes) all constraints can be satisfied, and
- (no) no assignment satisfies ϵ-fraction of constraints

Theorem (PCP + Raz's Parallel repetition)

For any $\epsilon > 0$, there is N such that $\mathrm{GLC}_{1,\epsilon}(N)$ is **NP***-hard.*

Fact

$\mathrm{GLC}_{1,\epsilon}(N)$ is equivalent to the problem of distiguishing whether a given bmc is trivial or any ϵ-fraction of it is non-trivial.

The Promise MC problem (=PCSP)

Definition

Fix a set $\mathcal{M}$ of functions. For an integer N, the problem $\mathrm{PMC}_{\mathcal{M}}(N)$ is, given a bmc, to distinguish between two cases:

- (yes) the bmc is trivial (sat in projections)
- (no) the bmc is not sat in $\mathcal{M}$

The Promise MC problem (=PCSP)

Definition

Fix a set $\mathcal{M}$ of functions. For an integer N, the problem $\mathrm{PMC}_{\mathcal{M}}(N)$ is, given a bmc, to distinguish between two cases:

- (yes) the bmc is trivial (sat in projections)
- (no) the bmc is not sat in $\mathcal{M}$

Theorem (Bulín, AK, Opršal)

Fix $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ *and let* $\mathcal{M} = \mathrm{Pol}(\mathbb{A},\mathbb{B})$.

1. *For each* N, $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ *is logspace reducible to* $\mathrm{PMC}_{\mathcal{M}}(N)$.

The Promise MC problem (=PCSP)

Definition

Fix a set $\mathcal{M}$ of functions. For an integer N, the problem $\mathrm{PMC}_{\mathcal{M}}(N)$ is, given a bmc, to distinguish between two cases:

- (yes) the bmc is trivial (sat in projections)
- (no) the bmc is not sat in $\mathcal{M}$

Theorem (Bulín, AK, Opršal)

Fix $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ *and let* $\mathcal{M}=\mathrm{Pol}(\mathbb{A},\mathbb{B})$.

1. *For each* N, $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$ *is logspace reducible to* $\mathrm{PMC}_{\mathcal{M}}(N)$.
2. *If* N *is larger than* $|A|$ *and max size of a relation in* $\mathbb{A}$ *then* $\mathrm{PMC}_{\mathcal{M}}(N)$ *is logspace reducible to* $\mathrm{PCSP}(\mathbb{A},\mathbb{B})$.

A strategy for proving hardness of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

Reduction from $\mathrm{GLC}_L 1, \epsilon(N)$ to $\mathrm{PMC}_{\mathcal{M}}(N)$:

A strategy for proving hardness of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

Reduction from $\mathrm{GLC}_L 1, \epsilon(N)$ to $\mathrm{PMC}_{\mathcal{M}}(N)$:
Carefully choose ϵ.
Transform a given GLC instance to a bmc as before and prove:

A strategy for proving hardness of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

Reduction from $\mathrm{GLC}_L 1, \epsilon(N)$ to $\mathrm{PMC}_{\mathcal{M}}(N)$:
Carefully choose ϵ.
Transform a given GLC instance to a bmc as before and prove:

1. GLC instance is sat $\Rightarrow$ the bmc is trivial (we know this)

A strategy for proving hardness of $\mathrm{PCSP}(\mathbb{A}, \mathbb{B})$

Reduction from $\mathrm{GLC}_L 1, \epsilon(N)$ to $\mathrm{PMC}_{\mathcal{M}}(N)$:
Carefully choose ϵ.
Transform a given GLC instance to a bmc as before and prove:

1. GLC instance is sat $\Rightarrow$ the bmc is trivial (we know this)
2. The bmc is sat in $\mathcal{M} \Rightarrow$ ϵ-fraction of the bmc is non-trivial.

   Since bmc $=$ LC, ϵ-fraction of the LC constraints are now sat.

Key idea: (2) should work because symmetries of $\mathcal{M}$ are limited.

Is there a dichotomy for PCSP?

It's too early too predict!

Is there a dichotomy for PCSP?

It's too early too predict!

- We already know it's all about symmetries
  - bipartite minor conditions satisfiable in polymorphisms

Is there a dichotomy for PCSP?

It's too early too predict!

- We already know it's all about symmetries
  - bipartite minor conditions satisfiable in polymorphisms
- For CSPs, have either no symmetries or strong symmetries

Is there a dichotomy for PCSP?

It's too early too predict!

- We already know it's all about symmetries
  - bipartite minor conditions satisfiable in polymorphisms
- For CSPs, have either no symmetries or strong symmetries
- For PCSPs:
  - limited non-trivial symmetries $\Rightarrow$ hardness
  - strong enough symmetries $\Rightarrow$ efficient algorithms

Is there a dichotomy for PCSP?

It's too early too predict!

- We already know it's all about symmetries
  - bipartite minor conditions satisfiable in polymorphisms
- For CSPs, have either no symmetries or strong symmetries
- For PCSPs:
  - limited non-trivial symmetries $\Rightarrow$ hardness
  - strong enough symmetries $\Rightarrow$ efficient algorithms
- Gap between "limited non-trivial" and "strong enough"?

Is there a dichotomy for PCSP?

It's too early too predict!

- We already know it's all about symmetries
  - bipartite minor conditions satisfiable in polymorphisms
- For CSPs, have either no symmetries or strong symmetries
- For PCSPs:
  - limited non-trivial symmetries $\Rightarrow$ hardness
  - strong enough symmetries $\Rightarrow$ efficient algorithms
- Gap between "limited non-trivial" and "strong enough"?
- Need more evidence, e.g. classifications for
  - all Boolean PCSPs? (cf. [Schaefer'78] for CSP)
  - promise graph homomorphisms? (cf. [Hell-Nešetřil'90])

Summary

- Promise CSP - a vast generalisation of finite domain CSP

Summary

- Promise CSP - a vast generalisation of finite domain CSP
- Involves both (in)approximability and infinite domain CSP
  - Label Cover is an algebraic problem!

Summary

- Promise CSP - a vast generalisation of finite domain CSP
- Involves both (in)approximability and infinite domain CSP
  - Label Cover is an algebraic problem!
- Includes famous open problems as special cases

Summary

- Promise CSP - a vast generalisation of finite domain CSP
- Involves both (in)approximability and infinite domain CSP
  - Label Cover is an algebraic problem!
- Includes famous open problems as special cases
- Amenable to algebraic theory (symmetries!)

Summary

- Promise CSP - a vast generalisation of finite domain CSP
- Involves both (in)approximability and infinite domain CSP
  - Label Cover is an algebraic problem!
- Includes famous open problems as special cases
- Amenable to algebraic theory (symmetries!)
- New alternatives to traditional approaches in complexity

Summary

- Promise CSP - a vast generalisation of finite domain CSP
- Involves both (in)approximability and infinite domain CSP
  - Label Cover is an algebraic problem!
- Includes famous open problems as special cases
- Amenable to algebraic theory (symmetries!)
- New alternatives to traditional approaches in complexity
- Complexity quest for PCSP - big, important, wide open