Robust satisfiability of CSPs

Libor Barto (Charles University in Prague)

joint work with Marcin Kozik (Jagiellonian University)

Dagstuhl, November 4, 2012
Definition (Instance of the CSP)

Instance of the CSP consists of:

- $V$ . . . a set of **variables**
- $A$ . . . a **domain**
- list of **constraints** of the form $R(x_1, \ldots, x_k)$, where
  - $x_1, \ldots, x_k \in V$
  - $R$ is a $k$-ary relation on $A$ (i.e. $R \subseteq A^k$) **constraint relation**
Constraint Satisfaction Problem (CSP)

Definition (Instance of the CSP)

Instance of the CSP consists of:

- $V$ . . . a set of **variables**
- $A$ . . . a **domain**
- list of **constraints** of the form $R(x_1, \ldots, x_k)$, where
  - $x_1, \ldots, x_k \in V$
  - $R$ is a $k$-ary relation on $A$ (i.e. $R \subseteq A^k$) **constraint relation**

An assignment $f : V \rightarrow A$ **satisfies** $R(x_1, \ldots, x_k)$, if $(f(x_1), \ldots, f(x_k)) \in R$

$f : V \rightarrow A$ is a **solution** if it satisfies all the constraints
Some questions we can ask

- **Decision CSP:** Does a solution exist?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** Find a map satisfying at least \(0.7 \times \text{Optimum}\) constraints.
Some questions we can ask

- **Decision CSP:** Does a solution exist?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** Find a map satisfying at least $0.7 \times \text{Optimum}$ constraints

**Definition**

An algorithm $(\alpha, \beta)$-approximates CSP ($0 \leq \alpha \leq \beta \leq 1$) if it returns an assignment satisfying $\alpha$-fraction of the constraints given a $\beta$-satisfiable instance.
Some questions we can ask

- **Decision CSP:** Does a solution exist?
- **Max-CSP:** Find a map satisfying maximum number of constraints
- **Approx. Max-CSP:** Find a map satisfying at least \(0.7 \times \text{Optimum}\) constraints

**Definition**

An algorithm \((\alpha, \beta)\)-approximates CSP \((0 \leq \alpha \leq \beta \leq 1)\) if it returns an assignment satisfying \(\alpha\)-fraction of the constraints given a \(\beta\)-satisfiable instance.

**Example**

\((0.7\beta, \beta)\)-approximating algorithm returns a map satisfying at least \(0.7 \times \text{Optimum}\) constraints.
A constraint language $\Gamma$ is a finite set of relations on a finite set $A$. An instance of $\text{CSP}(\Gamma)$ is a CSP instance such that every constraint relation is from $\Gamma$. 

**Definition**

A constraint language $\Gamma$ is a finite set of relations on a finite set $A$. An instance of $\text{CSP}(\Gamma)$ is a CSP instance such that every constraint relation is from $\Gamma$. 

- Is decision CSP($\Gamma$) in P? = Is (1,1)-approximation in P?
- For which $\alpha, \beta$ is ($\alpha, \beta$)-approximation of CSP($\Gamma$) in P?
- In between: Is robust approximation of CSP($\Gamma$) in P?
A constraint language $\Gamma$ is a finite set of relations on a finite set $A$. An instance of $\text{CSP}(\Gamma)$ is a CSP instance such that every constraint relation is from $\Gamma$.

Given $\Gamma$:

- Is decision $\text{CSP}(\Gamma)$ in $P$? $= \text{Is (1, 1)-approximation in } P$?
- For which $\alpha, \beta$ is $(\alpha, \beta)$-approximation of $\text{CSP}(\Gamma)$ in $P$?
- In between: Is robust approximation of $\text{CSP}(\Gamma)$ in $P$?
**Definition (Zwick'98)**

CSP(Γ) admits a robust algorithm, if there is a polynomial time algorithm which

\[(1 - g(\varepsilon), 1 - \varepsilon)\]-approximates CSP(Γ) (for every \(\varepsilon\)),

where \(g(\varepsilon) \to 0\) when \(\varepsilon \to 0\), and \(g(0) = 0\).
### Definition (Zwick'98)

CSP(Γ) admits a robust algorithm, if there is a polynomial time algorithm which

\[(1 - g(\varepsilon), 1 - \varepsilon)\]-approximates CSP(Γ) (for every \(\varepsilon\)),

where \(g(\varepsilon) \to 0\) when \(\varepsilon \to 0\), and \(g(0) = 0\).

**Motivation:** Instances close to satisfiable (e.g. corrupted by noise), we want to find an “almost solution”.
Definition (Zwick'98)

CSP(Γ) admits a robust algorithm, if there is a polynomial time algorithm which 

\((1 - g(\varepsilon), 1 - \varepsilon)\)-approximates CSP(Γ) (for every \(\varepsilon\)), 

where \(g(\varepsilon) \to 0\) when \(\varepsilon \to 0\), and \(g(0) = 0\).

Motivation: Instances close to satisfiable (e.g. corrupted by noise), we want to find an “almost solution”.

Techniques: Linear programming (LP), Semidefinite programming (SDP)
Between decision and approximation

**Definition (Zwick'98)**

CSP(\(\Gamma\)) admits a robust algorithm, if there is a polynomial time algorithm which

\((1 - g(\varepsilon), 1 - \varepsilon)\)-approximates CSP(\(\Gamma\)) (for every \(\varepsilon\)),

where \(g(\varepsilon) \to 0\) when \(\varepsilon \to 0\), and \(g(0) = 0\).

**Motivation:** Instances close to satisfiable (e.g. corrupted by noise), we want to find an “almost solution”.

**Techniques:** Linear programming (LP), Semidefinite programming (SDP)

**Questions:**
- For which \(\Gamma\) does CSP(\(\Gamma\)) admit a robust algorithm?
- What is (asymptotically) the best dependence of \(g\) on \(\varepsilon\)?
Positive results

- **HORN-\(k\)-SAT**
  - \((1 - O(1/(\log(1/\varepsilon))), 1 - \varepsilon)\) \textbf{LP} Zwick'98

- **HORN-2-SAT**
  - \((1 - 3\varepsilon, 1 - \varepsilon)\) Khanna, Sudan, Trevisan, Williamson’00
  - \((1 - 2\varepsilon, 1 - \varepsilon)\) Guruswami, Zhou’11

- **2-SAT**
  - \((1 - O(\varepsilon^{1/3}), 1 - \varepsilon)\) \textbf{SDP} Zwick’98
  - \((1 - O(\varepsilon^{1/2}), 1 - \varepsilon)\) Charikar, 2 × Makarychev’09
  - the same bound for CUT Goemans, Williamson’95

- **Unique-Games\((q)\) - generalization of CUT**
  - \((1 - O(\varepsilon^{1/5} \log^{1/2}(1/\varepsilon)), 1 - \varepsilon)\) Khot’02
  - \((1 - O(\varepsilon^{1/2}), 1 - \varepsilon)\) Charikar, 2 × Makarychev'06

Essentially optimal assuming UGC Khot’02, Khot, Kindler, Mossel, O'Donnell'07, Guruswami, Zhou’11
Positive results

- **HORN-\(k\)-SAT**
  - \((1 - O(1/(\log(1/\varepsilon))), 1 - \varepsilon)\) \text{ LP Zwick'98}

- **HORN-2-SAT**
  - \((1 - 3\varepsilon, 1 - \varepsilon)\) Khanna, Sudan, Trevisan, Williamson’00
  - \((1 - 2\varepsilon, 1 - \varepsilon)\) Guruswami, Zhou’11

- **2-SAT**
  - \((1 - O(\varepsilon^{1/3}), 1 - \varepsilon)\) \text{ SDP Zwick’98}
  - \((1 - O(\varepsilon^{1/2}), 1 - \varepsilon)\) Charikar, 2 × Makarychev’09
  - the same bound for CUT Goemans, Williamson’95

- **Unique-Games\((q)\) - generalization of CUT**
  - \((1 - O(\varepsilon^{1/5} \log^{1/2}(1/\varepsilon)), 1 - \varepsilon)\) Khot’02
  - \((1 - O(\varepsilon^{1/2}), 1 - \varepsilon)\) Charikar, 2 × Makarychev’06

Essentially optimal assuming UGC Khot’02, Khot, Kindler, Mossel, O’Donnell’07, Guruswami, Zhou’11
Negative results

- If the decision \( \text{CSP}(\Gamma) \) is \( \text{NP} \)-complete, then \( \text{CSP}(\Gamma) \) has no robust algorithm
  - PCP theorem for \( |A| = 2 \) Khanna, Sudan, Trevisan, Williamson’00
  - for larger \( A \) Jonsson, Krokhin, Kuivinen’09
Negative results

- If the decision $\text{CSP}(\Gamma)$ is NP-complete, then $\text{CSP}(\Gamma)$ has no robust algorithm
  - PCP theorem for $|A| = 2$ Khanna, Sudan, Trevisan, Williamson’00
  - for larger $A$ Jonsson, Krokhin, Kuivinen’09
- LIN-$p$ has no robust algorithm Hastad’01
If the decision $\text{CSP}(\Gamma)$ is NP-complete, then $\text{CSP}(\Gamma)$ has no robust algorithm

- PCP theorem for $|A| = 2$ Khanna, Sudan, Trevisan, Williamson’00
- for larger $A$ Jonsson, Krokhin, Kuivinen’09

- LIN-$p$ has no robust algorithm Hastad’01

What distinguishes between LIN-$p$, 3-SAT and 2-SAT, HORN-SAT?
Pol $\Gamma$ = clone of polymorphisms (operations compatible with all relations in $\Gamma$)

Complexity of the decision problem for $\text{CSP}(\Gamma)$ controlled by $\text{HSP}(\text{Pol } \Gamma)$ Bulatov, Jeavons, Krokhin 00

- Lin-$p$, 3-SAT do not have bounded width,
  2-SAT, HORN-SAT have bounded width

Conjecture (Guruswami-Zhou 11) $\text{CSP}(\Gamma)$ admits a robust algorithm iff $\text{CSP}(\Gamma)$ has bounded width.
Decision CSPs and bounded width

- \( \text{Pol}\, \Gamma = \text{clone of polymorphisms (operations compatible with all relations in } \Gamma) \)

- Complexity of the decision problem for \( \text{CSP}(\Gamma) \) controlled by \( \text{HSP}(\text{Pol}\, \Gamma) \) Bulatov, Jeavons, Krokhin 00

- \( \text{CSP}(\Gamma) \) has bounded width iff it can be solved by local consistency checking

- \( \text{CSP}(\Gamma) \) has bounded width iff \( \Gamma \) “cannot encode linear equations”, equivalently, \( \text{HSP}(\text{Pol}\, \Gamma) \) does not contain a reduct of a module (for core \( \Gamma \)) Barto, Kozik'09 Bulatov'09

- \( \text{Lin-SAT}, \, 3\text{-SAT} \) do not have bounded width, \( 2\text{-SAT}, \, \text{HORN-SAT} \) have bounded width

- Conjecture (Guruswami-Zhou 11) \( \text{CSP}(\Gamma) \) admits a robust algorithm iff \( \text{CSP}(\Gamma) \) has bounded width.
Decision CSPs and bounded width

- $\mathrm{Pol}\,\Gamma =$ clone of polymorphisms (operations compatible with all relations in $\Gamma$)
- Complexity of the decision problem for $\mathrm{CSP}(\Gamma)$ controlled by $\mathrm{HSP}(\mathrm{Pol}\,\Gamma)$ by Bulatov, Jeavons, Krokhin 00
- $\mathrm{CSP}(\Gamma)$ has bounded width iff it can be solved by local consistency checking
- $\mathrm{CSP}(\Gamma)$ has bounded width iff $\Gamma$ “cannot encode linear equations”, equivalently, $\mathrm{HSP}(\mathrm{Pol}\,\Gamma)$ does not contain a reduct of a module (for core $\Gamma$) by Barto, Kozik’09 Bulatov’09

- $\text{Lin-}, 3\text{-SAT do not have bounded width}, 2\text{-SAT, HORN-SAT have bounded width}

Conjecture (Guruswami-Zhou 11) $\mathrm{CSP}(\Gamma)$ admits a robust algorithm iff $\mathrm{CSP}(\Gamma)$ has bounded width.
Decision CSPs and bounded width

- $\text{Pol} \Gamma = \text{clone of polymorphisms (operations compatible with all relations in } \Gamma)$

- Complexity of the decision problem for $\text{CSP}(\Gamma)$ controlled by $\text{HSP}(\text{Pol} \Gamma)$ \cite{Bulatov, Jeavons, Krokhin 00}

- $\text{CSP}(\Gamma)$ has bounded width iff it can be solved by local consistency checking

- $\text{CSP}(\Gamma)$ has bounded width iff $\Gamma$ “cannot encode linear equations”, equivalently, $\text{HSP}(\text{Pol} \Gamma)$ does not contain a reduct of a module (for core $\Gamma$) \cite{Barto, Kozik’09 Bulatov’09}

- Lin-$p$, 3-SAT do not have bounded width, 2-SAT, HORN-SAT have bounded width
Pol $\Gamma = \text{clone of polymorphisms (operations compatible with all relations in } \Gamma)$

Complexity of the decision problem for CSP($\Gamma$) controlled by HSP(Pol $\Gamma$) Bulatov, Jeavons, Krokhin 00

CSP($\Gamma$) has bounded width iff it can be solved by local consistency checking

CSP($\Gamma$) has bounded width iff $\Gamma$ “cannot encode linear equations”, equivalently, HSP(Pol $\Gamma$) does not contain a reduct of a module (for core $\Gamma$) Barto, Kozik’09 Bulatov’09

Lin-$p$, 3-SAT do not have bounded width, 2-SAT, HORN-SAT have bounded width !!!
Decision CSPs and bounded width

- $\text{Pol} \, \Gamma = \text{clone of polymorphisms (operations compatible with all relations in } \Gamma)$
- Complexity of the decision problem for $\text{CSP}(\Gamma)$ controlled by $\text{HSP}(\text{Pol} \, \Gamma)$ \[\text{Bulatov, Jeavons, Krokhin 00}\]
- $\text{CSP}(\Gamma)$ has **bounded width** iff it can be solved by local consistency checking
- $\text{CSP}(\Gamma)$ has bounded width iff $\Gamma$ “cannot encode linear equations”, equivalently, $\text{HSP}(\text{Pol} \, \Gamma)$ does not contain a reduct of a module (for core $\Gamma$) \[\text{Barto, Kozik’09 Bulatov’09}\]
- **Lin-$p$, 3-SAT do not have bounded width,** 2-SAT, HORN-SAT have bounded width

**Conjecture (Guruswami-Zhou 11)**

$\text{CSP}(\Gamma)$ admits a robust algorithm iff $\text{CSP}(\Gamma)$ has bounded width.
robust approximation also (+-) controlled by polymorphisms
Dalmau, Krokhin’11

⇒ one direction of the Guruswami-Zhou conjecture is true
Universal algebra attacks robust approximation

- robust approximation also (+-) controlled by polymorphisms Dalmau, Krokhin’11
- $\Rightarrow$ one direction of the Guruswami-Zhou conjecture is true
- Conjecture confirmed for width 1 CSPs Kun, O’Donell, Tamaki, Yoshida, Zhou’11, Dalmau, Krokhin’11.
width 1 iff linear programming relaxation can be used.
robust approximation also (+-) controlled by polymorphisms
Dalmau, Krokhin’11

⇒ one direction of the Guruswami-Zhou conjecture is true

Conjecture confirmed for width 1 CSPs
Kun, O'Donell, Tamaki, Yoshida, Zhou’11, Dalmau, Krokhin’11.

width 1 iff linear programming relaxation can be used.

Conjecture confirmed Barto, Kozik’11. Using a
semidefinite programming relaxation and Prague strategies.

- Randomized \((1 - O(\log \log(1/\varepsilon)/\log(1/\varepsilon)), 1 - \varepsilon)\)-approx
  algorithm
- Deterministic \((1 - O(\log \log(1/\varepsilon)/\sqrt{\log(1/\varepsilon)}), 1 - \varepsilon)\)-approx
  algorithm
robust approximation also (+-) controlled by polymorphisms
Dalmau, Krokhin’11
⇒ one direction of the Guruswami-Zhou conjecture is true
Conjecture confirmed for width 1 CSPs
Kun, O’Donell, Tamaki, Yoshida, Zhou’11,
Dalmau, Krokhin’11.
width 1 iff linear programming relaxation can be used.

Conjecture confirmed Barto, Kozik’11. Using a
semidefinite programming relaxation and Prague strategies.
  ▶ Randomized \( (1 - O(\log \log(1/\varepsilon)/\log(1/\varepsilon)), 1 - \varepsilon)\)-approx
  ▶ Deterministic \( (1 - O(\log \log(1/\varepsilon)/\sqrt{\log(1/\varepsilon)}), 1 - \varepsilon)\)-approx

Krokhin’11: even the quantitative dependence on \( \varepsilon \) is +-
controlled by polymorphisms.
Notation and simplifying assumptions:

- $A$: domain
- $\Gamma$: contains only binary relations, $\text{CSP}(\Gamma)$ has bounded width
- $V$: variables, $\mathcal{I}$: instance, $\mathcal{C}$: constraints

Canonical SDP relaxation is strong enough to get optimal approximation constants (assuming UGC) Raghavendra'08
Notation and simplifying assumptions:

- \(A\): domain
- \(\Gamma\) contains only binary relations, \(\text{CSP}(\Gamma)\) has bounded width
- \(V\): variables, \(I\): instance, \(C\): constraints
- \(\forall \{x, y\} \subseteq V, x \neq y\) there is at most one constraint \(R_{xy}(x, y) \in C\)
SDP relaxation for general CSP

Notation and simplifying assumptions:

- \( A \): domain
- \( \Gamma \) contains only binary relations, CSP(\( \Gamma \)) has bounded width
- \( V \): variables, \( I \): instance, \( C \): constraints
- \( \forall \{x, y\} \subseteq V, x \neq y \) there is at most one constraint \( R_{xy}(x, y) \in C \)
- \( \text{Opt}(I) = 1 - \varepsilon \): optimal fraction of satisfied constraints
- We want to find an assignment satisfying almost all constraints
SDP relaxation for general CSP

Notation and simplifying assumptions:

- $A$: domain
- $\Gamma$: contains only binary relations, $\text{CSP}(\Gamma)$ has bounded width
- $V$: variables, $\mathcal{I}$: instance, $\mathcal{C}$: constraints
- $\forall \{x, y\} \subseteq V, x \neq y$ there is at most one constraint $R_{xy}(x, y) \in \mathcal{C}$
- $\text{Opt}(\mathcal{I}) = 1 - \varepsilon$: optimal fraction of satisfied constraints
- We want to find an assignment satisfying almost all constraints

Canonical SDP relaxation is strong enough to get optimal approximation constants (assuming UGC) Raghavendra’08
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$)
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$
Canonical SDP relaxation

Find vectors \( g(x, a) =: x_a, x \in V, a \in A \) (notation: \( x_B = \sum_{a \in B} x_a \)) such that for all \( x, y \in V, a, b \in A \)

- (SDP1) \( x_a y_b \geq 0 \)
- (SDP2) \( x_a x_b = 0 \) if \( a \neq b \)
- (SDP3) \( x_A = y_A, \|x_A\|^2 = 1 \)

maximizing

\[
\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R_{xy}(x,y) \in C} \sum_{(a,b) \in R_{xy}} x_a y_b.
\]
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$

maximizing

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R_{xy}(x,y) \in C} \sum_{(a,b) \in R_{xy}} x_a y_b.$$ 

Intuition:

- $x_a y_b$ is the weight (nonnegative) of the pair $(a, b)$ between variables $x, y$
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$

maximizing

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|\mathcal{C}|} \sum_{R_{xy}(x,y) \in \mathcal{C}} \sum_{(a,b) \in R_{xy}} x_a y_b.$$ 

Intuition:

- $x_a y_b$ is the weight (nonnegative) of the pair $(a, b)$ between variables $x, y$
- Sum of all weights (between $x, y$) is 1 from (SDP3)
Canonical SDP relaxation

Find vectors \( g(x, a) =: x_a, x \in V, a \in A \) (notation: \( x_B = \sum_{a \in B} x_a \)) such that for all \( x, y \in V, a, b \in A \)

- (SDP1) \( x_a y_b \geq 0 \)
- (SDP2) \( x_a x_b = 0 \) if \( a \neq b \)
- (SDP3) \( x_A = y_A, \|x_A\|^2 = 1 \)

maximizing

\[
\operatorname{SDPOpt}(\mathcal{I}) = \frac{1}{|\mathcal{C}|} \sum_{R_{xy}(x,y) \in \mathcal{C}} \sum_{(a,b) \in R_{xy}} x_a y_b.
\]

Intuition:

- \( x_a y_b \) is the weight (nonnegative) of the pair \((a, b)\) between variables \(x, y\)
- Sum of all weights (between \(x, y\)) is 1 from (SDP3)
- We are trying to give small weights to pairs outside \(R_{xy}\)
Canonical SDP relaxation

Find vectors $g(x, a) := x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$

maximizing

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|\mathcal{C}|} \sum_{R_{xy}(x,y) \in \mathcal{C}} \sum_{(a,b) \in R_{xy}} x_a y_b.$$  

Always $\text{SDPOpt}(\mathcal{I}) \geq \text{Opt}(\mathcal{I})$
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$

maximizing

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R_{xy}(x, y) \in C} \sum_{(a, b) \in R_{xy}} x_a y_b.$$ 

Further properties:

- $\|x_a\|^2$ is the weight of $a$
Find vectors \( g(x, a) =: x_a, x \in V, a \in A \) (notation: \( x_B = \sum_{a \in B} x_a \))
such that for all \( x, y \in V, a, b \in A \)

- **(SDP1)** \( x_a y_b \geq 0 \)
- **(SDP2)** \( x_a x_b = 0 \) if \( a \neq b \)
- **(SDP3)** \( x_A = y_A, \|x_A\|^2 = 1 \)

maximizing

\[
\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R_{xy}(x, y) \in C} \sum_{(a, b) \in R_{xy}} x_a y_b.
\]

Further properties:

- \( \|x_a\|^2 \) is the weight of \( a \)
- \( \|x_a\|^2 = (SDP2) \ x_a x_A = (SDP3) \ x_a y_A \)
Canonical SDP relaxation

Find vectors $g(x, a) =: x_a, x \in V, a \in A$ (notation: $x_B = \sum_{a \in B} x_a$) such that for all $x, y \in V, a, b \in A$

- (SDP1) $x_a y_b \geq 0$
- (SDP2) $x_a x_b = 0$ if $a \neq b$
- (SDP3) $x_A = y_A, \|x_A\|^2 = 1$

maximizing

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|\mathcal{C}|} \sum_{R_{xy}(x,y) \in \mathcal{C}} \sum_{(a,b) \in R_{xy}} x_a y_b.$$ 

Further properties:

- $\|x_a\|^2$ is the weight of $a$
- $\|x_a\|^2 = (SDP2)$ $x_a x_A = (SDP3)$ $x_a y_A$
- $\Rightarrow$ for every $y$, $\|x_a\|^2 = \text{sum of weights of edges between } x \text{ and } y \text{ via } a$
Strategy

We try to produce a good assignment from the SDP output vectors.
We try to produce a good assignment from the SDP output vectors.

In particular, is it true that if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\mathcal{I}$ has a solution? This was suggested by Guruswami as the first step to attack the conjecture.
Strategy

- We try to produce a good assignment from the SDP output vectors.
- In particular, is it true that if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\mathcal{I}$ has a solution? This was suggested by Guruswami as the first step to attack the conjecture.
- So, assume $\text{SDPOpt}(\mathcal{I}) = 1$. 
We try to produce a good assignment from the SDP output vectors.

In particular, is it true that if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\mathcal{I}$ has a solution? This was suggested by Guruswami as the first step to attack the conjecture.

So, assume $\text{SDPOpt}(\mathcal{I}) = 1$.

It follows that $x_a y_b = 0$ for every $(a, b) \not\in R_{xy}$. 
We try to produce a good assignment from the SDP output vectors.

In particular, is it true that if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\mathcal{I}$ has a solution? This was suggested by Guruswami as the first step to attack the conjecture.

So, assume $\text{SDPOpt}(\mathcal{I}) = 1$.

It follows that $x_a y_b = 0$ for every $(a, b) \notin R_{xy}$.

Define $P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}$. Replace $R_{xy}$ with $P_{xy}$.
If the new instance has a solution then the old one has a solution.
We try to produce a good assignment from the SDP output vectors.

In particular, is it true that if \( \text{SDPOpt}(\mathcal{I}) = 1 \) then \( \mathcal{I} \) has a solution?

This was suggested by Guruswami as the first step to attack the conjecture.

So, assume \( \text{SDPOpt}(\mathcal{I}) = 1 \).

It follows that \( x_a y_b = 0 \) for every \( (a, b) \notin R_{xy} \).

Define \( P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\} \).

Replace \( R_{xy} \) with \( P_{xy} \).

If the new instance has a solution then the old one has a solution.

Define \( P_x = \{a \in A : x_a \neq o\} \).
Random facts about $P_x, P_{xy}$

$$P_{xy} = \{(a, b) \in A^2 : x_ay_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\}$$

$\blacktriangleright$ $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)
Random facts about $P_x, P_{xy}$

$P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\}$

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)
  - It is a subset: If $x_a y_b > 0$ then $x_a, y_b \neq o$
  - It is subdirect: If $x_a \neq o$ then $0 \neq \|x_a\|^2 = x_a y A$, therefore $x_a y_b \neq 0$ for some $b$
Random facts about $P_x, P_{xy}$

$P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\}$

$P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let $B + (x, y) = \{c \in A : (\exists b \in B) (b, c) \in P_{xy}\}$
Random facts about $P_x, P_{xy}$

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\} \]

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let \( B + (x, y) = \{ c \in A : (\exists b \in B) (b, c) \in P_{xy} \} \)

- For $B \subseteq P_x$, we have $y_{B+(x,y)} = x_B + w$, where $wx_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$. 
Random facts about $P_x, P_{xy}$

$P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\}$

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let $B + (x, y) = \{c \in A : (\exists b \in B) \ (b, c) \in P_{xy}\}$

- For $B \subseteq P_x$, we have $y_{B+(x,y)} = x_B + w$, where $wx_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$.
  - $wx_B = (y_{B+(x,y)} - x_B)x_B = y_{B+(x,y)}x_B - x_B x_B = y_{B+(x,y)}x_B - y_A x_B = -y_A + y_{B+(x,y)}x_B = -y_A - (B+(x,y))x_B = 0$
  - $ww = \cdots = x_{A-B} y_{B+(x,y)}$
Random facts about $P_x, P_{xy}$

$$P_{xy} = \{(a, b) \in A^2 : x_a y_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\}$$

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let $B + (x, y) = \{c \in A : (\exists b \in B) \ (b, c) \in P_{xy}\}$

- For $B \subseteq P_x$, we have $y_{B + (x, y)} = x_B + w$, where $wx_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$.

A (correct) sequence of variables is called a pattern $B + p, B - p$ defined in a natural way for a pattern $p$
Random facts about $P_x, P_{xy}$

\[ P_{xy} = \{(a, b) \in A^2 : x_ay_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\} \]

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let $B + (x, y) = \{c \in A : (\exists b \in B) \ (b, c) \in P_{xy}\}$

- For $B \subseteq P_x$, we have $y_{B+(x,y)} = x_B + w$, where $wx_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$.

A (correct) sequence of variables is called a pattern $B + p, B - p$ defined in a natural way for a pattern $p$

For any $B \subseteq P_x$ and patterns $p, q$ from $x$ to $x$ we have

- If $B + p = B$ then $B - p = B$
Random facts about $P_x, P_{xy}$

\[ P_{xy} = \{(a, b) \in A^2 : x_ay_b > 0\}, \quad P_x = \{a \in A : x_a \neq o\} \]

- $P_{xy}$ is a subdirect subset of $P_x \times P_y$ (1-minimality)

For $B \subseteq P_x$ let $B + (x, y) = \{c \in A : (\exists b \in B) (b, c) \in P_{xy}\}$

- For $B \subseteq P_x$, we have $y_{B+(x,y)} = x_B + w$, where $wx_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$.

A (correct) sequence of variables is called a pattern $B + p, B - p$ defined in a natural way for a pattern $p$

For any $B \subseteq P_x$ and patterns $p, q$ from $x$ to $x$ we have
- If $B + p = B$ then $B - p = B$
- If $B + p + q = B$ then $B + p = B$
Random facts about $P_x, P_{xy}$ - summary

The new instance with constraints $P_{xy}(x, y)$ and subsets $P_x \subseteq A, x \in V$ satisfies
(for every $x, y \in V, B \subseteq P_x$ and patterns $p, q$ from $x$ to $x$)

(P1) It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)
(P2) If $B + p = B$ then $B - p = B$
(P3) If $B + p + q = B$ then $B + p = B$
An instance with constraints $P_{xy}(x, y)$ and subsets $P_x \subseteq A, x \in V$ is a **weak Prague instance** if

(for every $x, y \in V$, $B \subseteq P_x$ and patterns $p, q$ from $x$ to $x$)

(P1) It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)

(P2) If $B + p = B$ then $B - p = B$

(P3) If $B + p + q = B$ then $B + p = B$
Weak Prague instance

Definition

An instance with constraints $P_{xy}(x, y)$ and subsets $P_x \subseteq A, x \in V$ is a **weak Prague instance** if

(1) It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)

(2) If $B + p = B$ then $B - p = B$

(3) If $B + p + q = B$ then $B + p = B$

- Slightly weaker notion than Prague strategy
- Every Prague strategy has a solution (if $P_{xy}$’s are invariant under Pol $\Gamma$...) BK
An instance with constraints $P_{xy}(x, y)$ and subsets $P_x \subseteq A, x \in V$ is a weak Prague instance if (for every $x, y \in V$, $B \subseteq P_x$ and patterns $p, q$ from $x$ to $x$)

(P1) It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)

(P2) If $B + p = B$ then $B - p = B$

(P3) If $B + p + q = B$ then $B + p = B$

- Slightly weaker notion than Prague strategy
- Every Prague strategy has a solution (if $P_{xy}$’s are invariant under Pol $\Gamma$...) BK
- Every weak Prague strategy has a solution BK
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- Choose $\delta$ (randomly with some distribution)
- Put $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- Put $P_x = \{a : \|x_a\|^2 > \delta\}$
General case

- SDPOpt(Γ) = 1 − ε, ε small
- Choose δ (randomly with some distribution)
- Put $P_{xy} = \{(a, b) : \mathbf{x}_a \mathbf{x}_b > \delta\}$
- Put $P_x = \{a : \|\mathbf{x}_a\|^2 > \delta\}$
- If δ not too tiny
  then for almost all x, y we have $P_{xy} \subseteq R_{xy}$. 
General case

- SDPOpt(Γ) = 1 − ε, ε small
- Choose δ (randomly with some distribution)
- Put \( P_{xy} = \{(a, b) : x_a x_b > \delta\} \)
- Put \( P_x = \{a : \|x_a\|^2 > \delta\} \)
- If δ not too tiny
  then for almost all \( x, y \) we have \( P_{xy} \subseteq R_{xy} \).
- Give up other constraints
General case

- SDPOpt(Γ) = 1 − ε, ε small
- Choose δ (randomly with some distribution)
  - Put $P_{xy} = \{(a, b): x_a x_b > \delta\}$
  - Put $P_x = \{a: \|x_a\|^2 > \delta\}$
- If δ not too tiny
  then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$.
- Give up other constraints
- Now we can work with $P_{xy}$ instead of $R_{xy}$
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

We want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \).
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

- Give up \( P_{xy} \) for which some \( x_a y_b \) is in \((\delta - \text{enough}, \delta)\)
  \( \delta \) is chosen so that we don’t delete too much
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : | |x_a| |^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

- Give up \( P_{xy} \) for which some \( x_a y_b \) is in \((\delta - \text{enough}, \delta)\)
  \( \delta \) is chosen so that we don't delete too much

- (P1) \( P_{xy} \subseteq P_x \times P_y \):
  If \( x_a y_b > \delta \) then \( | |x_a| |^2 = x_a y_B \geq x_a y_b > \delta \)
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

▶ Give up \( P_{xy} \) for which some \( x_a y_b \) is in \((\delta -\text{enough}, \delta)\)
\( \delta \) is chosen so that we don’t delete too much

▶ (P1) \( P_{xy} \subseteq P_x \times P_y \):
If \( x_a y_b > \delta \) then \( \|x_a\|^2 = x_a y_B \geq x_a y_b > \delta \)

▶ (P1) Subdirectness: If \( \|x_a\|^2 = x_a y_B > \delta \) then \( x_a y_b \geq \delta/|A| \).
Then \( x_a y_B \geq \delta \) (as \( \delta/|A| > \delta -\text{enough} \))
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : ||x_a||^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

- Give up \( P_{xy} \) for which some \( x_a y_b \) is in \((\delta - \text{enough}, \delta)\)
  \( \delta \) is chosen so that we don’t delete too much

- (P1) \( P_{xy} \subseteq P_x \times P_y \):
  If \( x_a y_b > \delta \) then \( ||x_a||^2 = x_a y_B \geq x_a y_b > \delta \)

- (P1) \textbf{Subdirectness:} If \( ||x_a||^2 = x_a y_B > \delta \) then \( x_a y_b \geq \delta / |A| \). Then \( x_a y_B \geq \delta \) (as \( \delta / |A| > \delta - \text{enough} \))

Important property: \( y_{B+(x,y)} \) is
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

- Give up \( P_{xy} \) for which some \( x_a y_b \) is in \((\delta - \text{enough}, \delta)\)
  \( \delta \) is chosen so that we don’t delete too much

- (P1) \( P_{xy} \subseteq P_x \times P_y \):
  If \( x_a y_b > \delta \) then \( \|x_a\|^2 = x_a y_B \geq x_a y_b > \delta \)

- (P1) Subdirectness: If \( \|x_a\|^2 = x_a y_B > \delta \) then \( x_a y_b \geq \delta / |A| \). Then \( x_a y_B \geq \delta \) (as \( \delta / |A| > \delta - \text{enough} \))

Important property: \( y_B + (x, y) \) is

- either almost (\( \ll \delta \)) the same as \( x_B \)
  (in case that \( B + (x, y, x) = B \)),
Enforcing (P1)

\[ P_{xy} = \{(a, b) \in A^2 : x_ay_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

we want (P1) \( P_{xy} \) is a subdirect subset of \( P_x \times P_y \)

- Give up \( P_{xy} \) for which some \( x_ay_b \) is in \((\delta - enough, \delta)\)
  \( \delta \) is chosen so that we don’t delete too much

- (P1) \( P_{xy} \subseteq P_x \times P_y \):
  If \( x_ay_b > \delta \) then \( \|x_a\|^2 = x_ay_B \geq x_ay_b > \delta \)

- (P1) **Subdirectness:** If \( \|x_a\|^2 = x_ay_B > \delta \) then \( x_ay_b \geq \delta/|A| \). Then \( x_ay_B \geq \delta \) (as \( \delta/|A| > \delta - enough \))

Important property: \( y_B + (x,y) \) is

- either almost (\( \ll \delta \)) the same as \( x_B \)
  (in case that \( B + (x, y, x) = B \)),
- or significantly (\( > \delta \)) longer than \( x_B \) (otherwise)
Enforcing (P2)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : ||x_a||^2 > \delta\} \]

We want (P2) If \( B + p = B \) then \( B - p = B \)
Enforcing (P2)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

We want (P2) If \( B + p = B \) then \( B - p = B \)

- Divide the unit ball into layers
  (thickness about \( \delta \), randomly shifted)
Enforcing (P2)

\[
P_{xy} = \{(a, b) \in A^2 : x_ay_b > \delta\}, \quad P_{x} = \{a \in A : ||x_a||^2 > \delta\}
\]

We want (P2) if \(B + p = B\) then \(B - p = B\)

- Divide the unit ball into layers
  (thickness about \(\delta\), randomly shifted)
- Give up \(P_{xy}\) if there are almost the same vectors \(x_B, y_{B+}(x,y)\)
  in different layers.

Recall that 
\[
y_{B+}(x,y)\]

- either almost the same as \(x_B\) (in case that \(B + (x,y) = B\)),
- or significantly longer (>\(\delta\)) than \(x_B\) \(\Rightarrow\) vector jumps to
  higher layer
This guarantees (P2)
(note: so far we only used lengths \(\Rightarrow\) can be done for LP relaxation)
Enforcing (P2)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

We want (P2) If \( B + p = B \) then \( B - p = B \)

- Divide the unit ball into layers
  (thickness about \( \delta \), randomly shifted)
- Give up \( P_{xy} \) if there are almost the same vectors \( x_B, y_{B+(x,y)} \)
in different layers.

Recall that \( y_{B+(x,y)} \) is

- either almost the same as \( x_B \) (in case that \( B + (x, y, x) = B \)),
- or significantly longer (\( > \delta \)) than \( x_B \) \( \Rightarrow \) vector jumps to higher layer

This guarantees (P2)
Enforcing (P2)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

We want (P2) If \( B + p = B \) then \( B - p = B \)

- Divide the unit ball into layers
  (thickness about \( \delta \), randomly shifted)
- Give up \( P_{xy} \) if there are almost the same vectors \( x_B, y_{B+}(x,y) \) in different layers.

Recall that \( y_{B+(x,y)} \) is

- either almost the same as \( x_B \) (in case that \( B + (x, y, x) = B \)),
- or significantly longer (\( > \delta \)) than \( x_B \) \(\Rightarrow\) vector jumps to higher layer

This guarantees (P2)

(note: so far we only used lengths \(\Rightarrow\) can be done for LP relaxation)
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

▶ Choose sufficiently many hyperplanes (randomly)
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : ||x_a||^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

▶ Choose sufficiently many hyperplanes (randomly)

(\(x\)) Give up variables \( x \) for which some pair \( x_B, x_C \) is not cut
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

- Choose sufficiently many hyperplanes (randomly)
- Give up variables \( x \) for which some pair \( x_B, x_C \) is not cut
- Give up constraints \( P_{xy} \) for which there are almost the same vectors \( x_B, y_{B+\langle x, y \rangle} \) which are cut
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

- Choose sufficiently many hyperplanes (randomly)
- (x) Give up variables \( x \) for which some pair \( x_B, x_C \) is not cut
- (y) Give up constraints \( P_{xy} \) for which there are almost the same vectors \( x_B, y_{B+(x,y)} \) which are cut

- This guarantees (P3)
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_ay_b > \delta\}, \quad P_x = \{a \in A : \|x_a\|^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

▶ Choose sufficiently many hyperplanes (randomly)

(x) Give up variables \( x \) for which some pair \( x_B, x_C \) is not cut

(y) Give up constraints \( P_{xy} \) for which there are almost the same vectors \( x_B, y_{B+(x,y)} \) which are cut

▶ This guarantees (P3)

▶ Remark: Different number of hyperplanes is used for different layers otherwise (x) or (y) would delete too much
Enforcing (P3)

\[ P_{xy} = \{(a, b) \in A^2 : x_a y_b > \delta\}, \quad P_x = \{a \in A : ||x_a||^2 > \delta\} \]

(P3) If \( B + p + q = B \) then \( B + p = B \)

- Choose sufficiently many hyperplanes (randomly)
- Give up variables \( x \) for which some pair \( x_B, x_C \) is not cut
- Give up constraints \( P_{xy} \) for which there are almost the same vectors \( x_B, y_{B+(x,y)} \) which are cut

- This guarantees (P3)
- Remark: Different number of hyperplanes is used for different layers otherwise (x) or (y) would delete too much

Now we have a Prague instance. Algebraic closure has a solution
Final remarks

- Is the quantitative dependence optimal?
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
- What can we say about the quantitative dependence on $\varepsilon$ in general?
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
- What can we say about the quantitative dependence on $\varepsilon$ in general?
  - Guess: $\text{NU} \Rightarrow$ polynomial loss, or $\text{SD}(\land) \Rightarrow$ polynomial loss
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
- What can we say about the quantitative dependence on $\varepsilon$ in general?
  - Guess: $\text{NU} \Rightarrow$ polynomial loss, or $\text{SD}(\wedge) \Rightarrow$ polynomial loss
  - Candidates for hardness: 2-semilattices?

Explore further SDP, LP $\leftrightarrow$ consistency notions

Thank you!
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
- What can we say about the quantitative dependence on $\varepsilon$ in general?
  - Guess: NU $\Rightarrow$ polynomial loss, or SD($\wedge$) $\Rightarrow$ polynomial loss
  - Candidates for hardness: 2-semilattices?
- Explore further SDP,LP $\leftrightarrow$ consistency notions
Final remarks

- Is the quantitative dependence optimal?
- How to improve derandomization to match the randomized version?
- What can we say about the quantitative dependence on $\varepsilon$ in general?
  - Guess: NU $\Rightarrow$ polynomial loss, or $SD(\wedge) \Rightarrow$ polynomial loss
  - Candidates for hardness: 2-semilattices?
- Explore further SDP, LP $\leftrightarrow$ consistency notions

Thank you!