Robust algorithms for CSPs

Libor Barto

joint work with Marcin Kozik

McMaster University
and
Charles University in Prague

AAA 83 Novi Sad, March 15, 2012
(Part 1) Outline

- (Part 2) Introduction
- (Part 3) Problem
- (Part 4) Problem solved
- (Part 5) Proof of a different result
- (Part 6) Proof of one more different result
(Part 2)
Introduction
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**
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 $0.7 \times Optimum$ 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 $0.7 \times Optimum$ 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 $0.7 \times Optimum$ constraints.
Mentioned problems are computationally hard

One possible restriction (widely studied) — fix a set of possible constraint relations:

**Definition**

A *constraint language* $\Gamma$ is a finite set of relations on a finite set $A$.

An *instance of CSP($\Gamma$)* is a CSP instance such that every constraint relation is from $\Gamma$. 
Example: 2-coloring

\[ A = \{0, 1\}, \quad \Gamma = \{R\}, \quad R = \{(0, 1), (1, 0)\} \] (inequality)

Instance: \( R(x_1, x_2), R(x_1, x_3), R(x_2, x_4), \ldots \)
(can be drawn as a graph)

Solution = 2-coloring (bipartition)

- **Decision** \( CSP(\Gamma) \): Is a given graph bipartite? (easy)
- **Max-\( CSP(\Gamma) \)**: also called Max-Cut (hard)
- **Approx. Max-\( CSP(\Gamma) \)**
  - \((0.5 \beta, \beta)\)-approx easy
  - \((0.878 \beta, \beta)\)-approx easy Goemans and Williamson’95
  - \((16/17 \beta, \beta)\)-approx hard Trevisan, Sorkin, Sudan, Williamson’00, Hastad’01
  - \(((0.878 + \varepsilon) \beta, \beta)\) - approx UGC-hard Khot, Kindler, Mossel, O’Donnel’07
Example: 3-SAT

\[ A = \{0, 1\}, \Gamma = \{R_{000}, R_{001}, R_{011}, R_{111}\}, \quad R_{ijk} = \{0, 1\}^3 \{(i, j, k)\} \]

Instance: \( R_{000}(x_1, x_2, x_3), R_{001}(x_1, x_3, x_5), R_{011}(x_3, x_2, x_6) \)

or: \((x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_3 \lor \neg x_5) \land (x_3 \lor \neg x_2 \lor \neg x_6)\)

- **Decision** \(CSP(\Gamma)\): 3-SAT (hard)
- **Max-\(CSP(\Gamma)\)**: Max-3-SAT (hard)
- **Approx. Max-\(CSP(\Gamma)\)**:
  - \((7/8\beta, \beta)\)-approx easy Karloff, Zwick'96
  - \((\delta, 1)\)-approx hard for some \(\delta < 1\)
    (=PCP theorem, Arora, Lund, Motwani, Sudan, Szegedy'98)
  - \((7/8 + \varepsilon, 1)\)-approx hard Hastad'01
Example: 3-Lin-2

\[ A = \{0, 1\}, \quad \Gamma = \{ \text{affine subspaces of } \mathbb{Z}_2^3 \} \]

Instance: system of linear equation over \( \mathbb{Z}_2 \)
(each equation contains at most 3 variables)

- **Decision** \( CSP(\Gamma) \): easy (Gaussian elimination)
- **Max-\( CSP(\Gamma) \): hard
- **Approx. Max-\( CSP(\Gamma) \):**
  - \((1/2\beta, \beta)\)-approx easy
  - \((1/2 + \varepsilon, 1 - \varepsilon)\)-approx hard Hastad’01
(Part 3) Problem
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 \(g(\varepsilon) \to 0 \text{ when } \varepsilon \to 0\), and \(g(0) = 0\).

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

Between decision and approximation

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 $g(0) = 0$.

- 2-SAT, HORN-SAT have robust algorithms Zwick’98
Between decision and approximation

Definition (Zwick'98)

\[ \text{CSP}(\Gamma) \text{ admits a robust algorithm, if there is a polynomial time algorithm which} \]
\[ (1 - g(\varepsilon), 1 - \varepsilon) \text{-approximates } \text{CSP}(\Gamma) \text{ (for every } \varepsilon), \]
\[ \text{where } g(\varepsilon) \to 0 \text{ when } \varepsilon \to 0, \text{ and } g(0) = 0. \]

- 2-SAT, HORN-SAT have robust algorithms Zwick’98
  - \( (1 - O(\varepsilon^{1/3}), 1 - \varepsilon) \)-approx algorithm for 2-SAT
  - \( (1 - O(1/(\log(1/\varepsilon))), 1 - \varepsilon) \)-approx algorithm for HORN-SAT
Definition (Zwick'98)

\( \text{CSP}(\Gamma) \) admits a robust algorithm, if there is a polynomial time algorithm which
\((1 - g(\varepsilon), 1 - \varepsilon)\)-approximates \( \text{CSP}(\Gamma) \) (for every \( \varepsilon \)),
where \( g(\varepsilon) \rightarrow 0 \) when \( \varepsilon \rightarrow 0 \), and \( g(0) = 0 \).

- 2-SAT, HORN-SAT have robust algorithms Zwick’98
- If the decision problem for \( \text{CSP}(\Gamma) \) is NP-complete, then
\( \text{CSP}(\Gamma) \) has no robust algorithm (PCP,
for \( |A| = 2 \) Khanna, Sudan, Trevisan, Williamson’00
for larger Jonsson, Krokhin, Kuivinen’09)
Definition (Zwick'98)

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

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

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

- 2-SAT, HORN-SAT have robust algorithms Zwick’98
- If the decision problem for CSP(Γ) is NP-complete, then CSP(Γ) has no robust algorithm (PCP, for \(|A| = 2\) Khanna,Sudan,Trevisan, Williamson’00 for larger Jonsson, Krokhin, Kuivinen’09)
- LIN-\(p\) has no robust algorithm Hastad’01
Definition (Zwick'98)

\[ \text{CSP}(\Gamma) \text{ admits a robust algorithm, if there is a polynomial time algorithm which} \]
\[ (1 - g(\varepsilon), 1 - \varepsilon)-\text{approximates} \ \text{CSP}(\Gamma) \ (\text{for every} \ \varepsilon), \]
\[ \text{where} \ g(\varepsilon) \rightarrow 0 \ \text{when} \ \varepsilon \rightarrow 0, \ \text{and} \ g(0) = 0. \]

- 2-SAT, HORN-SAT have robust algorithms Zwick’98
- If the decision problem for CSP(\Gamma) is NP-complete, then CSP(\Gamma) has no robust algorithm (PCP, for |A| = 2 Khanna,Sudan,Trevisan, Williamson’00 for larger Jonsson, Krokhin, Kuivinen’09)
- LIN-\( p \) has no robust algorithm Hastad’01

What distinguishes between LIN-\( p \), 3-SAT and 2-SAT, HORN-SAT?
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

- 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", more precisely, \( \text{HSP}(\text{Pol} \, \Gamma) \) does not contain a reduct of a module (for core \( \Gamma \)) Barto, Kozik’09 Bulatov’09

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

- Conjecture (Guruswami-Zhou 11) CSP(\( \Gamma \)) admits a robust algorithm iff CSP(\( \Gamma \)) has bounded width.
Pol Γ = clone of polymorphisms (operations compatible with all relations in Γ)

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

CSP(Γ) has bounded width iff it can be solved by local consistency checking

CSP(Γ) has bounded width iff Γ "cannot encode linear equations", more precisely, HSP(Pol Γ) does not contain a reduct of a module (for core Γ) Barto, Kozik'09 Bulatov'09

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

Conjecture (Guruswami-Zhou 11) CSP(Γ) admits a robust algorithm iff CSP(Γ) 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 CSP($\Gamma$) controlled by $\text{HSP}(\text{Pol} \Gamma)$ 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”, more precisely, $\text{HSP}(\text{Pol} \Gamma)$ does not contain a reduct of a module (for core $\Gamma$) Barto, Kozik’09 Bulatov’09

Lin- 
3-SAT do not have bounded width, 
2-SAT, HORN-SAT have bounded width
Conjecture (Guruswami-Zhou 11)
CSP($\Gamma$) admits a robust algorithm iff 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 \( \text{HSP}(\text{Pol} \Gamma) \) \textbf{Bulatov, Jeavons, Krokhin 00}
- \( \text{CSP}(\Gamma) \) has \textit{bounded width} iff it can be solved by local consistency checking
- \( \text{CSP}(\Gamma) \) has bounded width iff \( \Gamma \) “cannot encode linear equations”, more precisely, \( \text{HSP}(\text{Pol} \Gamma) \) does not contain a reduct of a module \textbf{(for core} \( \Gamma \)) \textbf{Barto, Kozik’09 Bulatov’09}
- \( \text{Lin-}p \), \( 3\text{-SAT} \) do not have bounded width, \( 2\text{-SAT} \), \( \text{HORN-SAT} \) 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 CSP($\Gamma$) controlled by HSP($\text{Pol}\Gamma$) 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”, more precisely, HSP($\text{Pol}\Gamma$) 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 $\text{HSP}(\text{Pol } \Gamma)$ 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”, more precisely, $\text{HSP}(\text{Pol } \Gamma)$ does not contain a reduct of a module (for core $\Gamma$) Barto, Kozik’09 Bulatov’09
- $\text{Lin}-p$, $3\text{-SAT}$ 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.
robust approximation also (+-) controlled by polymorphisms
Dalmau, Krokhin'11

⇒ one direction of the Guruswami-Zhou conjecture is true
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.
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.

- Conjecture confirmed
  Barto, Kozik’11.
Universal algebra attacks robust approximation

- 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
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.
- 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
- Bonus Krokhin’11: even the quantitative dependence on \(\varepsilon\) is
  \(+\)- controlled by polymorphisms.
This was (Part 4)
Problem solved
Now (Part 5)
Proof of a different result
\[ A = \{-1, 1\}, \quad \Gamma = \{R\}, \quad R = \{(-1, 1), (1, -1)\} \quad (\text{inequality}) \]

Instance \( \mathcal{I} \): \( V = \{x_1, x_2, \ldots\} \), \( C = R(x_2, x_1), R(x_1, x_4), \ldots \)
MAX-CUT  Goemans and Williamson’95

\[ A = \{-1, 1\} \], \[ \Gamma = \{R\} \], \[ R = \{(-1, 1), (1, -1)\} \] (inequality)

Instance \( \mathcal{I} \): \( V = \{x_1, x_2, \ldots\} \), \( C = R(x_2, x_1), R(x_1, x_4), \ldots \)

Max-CSP – hard:
Find **numbers** \( f(x), x \in V \), \( f(x) \in \{-1, 1\} \) which maximize

\[
\text{Opt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - f(x)f(y)}{2}
\]
MAX-CUT Goemans and Williamson’95

\[ A = \{-1, 1\}, \Gamma = \{R\}, \ R = \{(-1, 1), (1, -1)\} \] (inequality)

Instance \( I \): \( V = \{x_1, x_2, \ldots, \} \), \( C = R(x_2, x_1), R(x_1, x_4), \ldots \)

Max-CSP – hard:
Find numbers \( f(x), x \in V, f(x) \in \{-1, 1\} \) which maximize

\[ \text{Opt}(I) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - f(x)f(y)}{2} \]

SDP (semidefinite programming) relaxation – easy:
Find vectors \( g(x), x \in V, \|g(x)\|^2 = 1 \) which maximize

\[ \text{SDPOpt}(I) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - g(x)g(y)}{2} \]
Find vectors $g(x), x \in V, \|g(x)\|^2 = 1$ which maximize

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - g(x)g(y)}{2}$$
Find vectors $g(x), x \in V$, $\|g(x)\|^2 = 1$ which maximize

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|\mathcal{C}|} \sum_{R(x,y) \in \mathcal{C}} \frac{1 - g(x)g(y)}{2}$$

$\text{SDPOpt}(\mathcal{I}) \geq \text{Opt}(\mathcal{I})$, if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\text{Opt}(\mathcal{I}) = 1$. 

We need to round the vector solution $g$ to a reasonably good assignment $f$. Choose a random hyperplane through the origin and choose one side $S$. Put $f(v) = 1$ if $g(v) \in S$ and $f(v) = -1$ otherwise. This is $(0.878, \beta, \beta)$-approx and robust algorithm.
Find vectors $g(x), x \in V, \|g(x)\|^2 = 1$ which maximize

$$\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - g(x)g(y)}{2}$$

- $\text{SDPOpt}(\mathcal{I}) \geq \text{Opt}(\mathcal{I})$, if $\text{SDPOpt}(\mathcal{I}) = 1$ then $\text{Opt}(\mathcal{I}) = 1$.
- We need to round the vector solution $g$ to a reasonably good assignment $f$
Find vectors $g(x), x \in V, \|g(x)\|^2 = 1$ which maximize

$$SDPOpt(I) = \frac{1}{|\mathcal{C}|} \sum_{R(x,y) \in \mathcal{C}} \frac{1 - g(x)g(y)}{2}$$

- $SDPOpt(I) \geq Opt(I)$, if $SDPOpt(I) = 1$ then $Opt(I) = 1$.
- We need to round the vector solution $g$ to a reasonably good assignment $f$
  - Choose a random hyperplane through the origin and choose one side $S$
  - Put $f(v) = 1$ if $g(v) \in S$ and $f(v) = -1$ otherwise
MAX-CUT cont’d

Find vectors \( g(x), x \in V, \|g(x)\|^2 = 1 \) which maximize

\[
\text{SDPOpt}(\mathcal{I}) = \frac{1}{|C|} \sum_{R(x,y) \in C} \frac{1 - g(x)g(y)}{2}
\]

- \( \text{SDPOpt}(\mathcal{I}) \geq \text{Opt}(\mathcal{I}) \), if \( \text{SDPOpt}(\mathcal{I}) = 1 \) then \( \text{Opt}(\mathcal{I}) = 1 \).
- We need to round the vector solution \( g \) to a reasonably good assignment \( f \)
  - Choose a random hyperplane through the origin and choose one side \( S \)
  - Put \( f(v) = 1 \) if \( g(v) \in S \) and \( f(v) = -1 \) otherwise
- This is \( (0.878\beta, \beta) \)-approx and robust algorithm
(Part 6)
Proof of one more different result
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, \( C \) – constraints
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 $R_{xy}(x, y) \in \mathcal{C}$

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

Let's try to use it for our problem.
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 $R_{xy}(x, y) \in \mathcal{C}$
- $\text{Opt}(\mathcal{I})$ – optimal fraction of satisfied constraints
- ... and we want to find an assignment satisfying a big fraction of the constraints

[picture]
SDP relaxation for general CSP

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 $R_{xy}(x, y) \in C$
- $\text{Opt}(I)$ – optimal fraction of satisfied constraints
- ... and we want to find an assignment satisfying a big fraction of the constraints

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

Let’s try to use it for our problem.
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) $x_A = y_A, \|x_A\|^2 = 1$
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:
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 \( x, y \in V, a, b \in A \)

\begin{itemize}
  \item (SDP1) \( x_a y_b \geq 0 \)
  \item (SDP2) \( x_a x_b = 0 \) if \( a \neq b \)
  \item (SDP3) \( x_A = y_A, \|x_A\|^2 = 1 \)
\end{itemize}

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 a 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

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

Intuition:

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

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

Intuition:

- $x_a y_b$ is a 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 $R_{xy}$
Strategy

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

- We try to produce a good assignment from the SDP output vectors.
- In particular, is it true that if $SDPOpt(I) = 1$ then $I$ has a solution? This was suggested by Guruswami as the first step to attack the conjecture.
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 $x_ay_b = 0$ for every $(a, b) \notin R_{xy}$. 
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 \( (a, b) \not\in R_{xy} \).

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 \( P_x = \{a \in A : x_a \neq o\} \). And let’s see what we get.
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)
Random facts about $P_x, P_{xy}$

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

- $P_{xy}$ 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 $0 \neq \|x_a\|^2 = x_a x_A = x_a y_A$, therefore $x_a y_b \neq 0$ for some $b$
Random facts about $P_x, P_{xy}$

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

- $P_{xy}$ 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 $w x_B = 0$, and $w = o$ iff $B = B + (x, y) - (x, y)$.
Random facts about $P_x, P_{xy}$

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

- $P_{xy}$ 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 $B = B + (x, y) - (x, y)$.
  - $wx_B = (y_{B+(x,y)} - x_B)x_B = y_{B+(x,y)}x_B - x_Bx_B = y_{B+(x,y)}x_B - y_Ax_B = -(y_A - y_{B+(x,y)})x_B = -y_A - (B+(x,y))x_B = 0$
  - $ww = \cdots = x_{A-B}y_{B+(x,y)}$

- $w$ is a pattern defined in a natural way for a pattern $p$, $q$ from $x$ to $x$:
  - If $B + p = B$ then $B - p = B$
  - If $B + p + q = B$ 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 $p$.
Random facts about $P_x, P_{xy}$

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

- $P_{xy}$ 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 $P_x, P_{xy}$

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

- $P_{xy}$ 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 $w x_B = 0$, 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 $x$)

- It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)
- If $B + p = B$ then $B - p = B$
- 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 $x$)

- It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)
- If $B + p = B$ then $B - p = B$
- If $B + p + q = B$ then $B + p = B$
Definition

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 $x$)

- It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)
- If $B + p = B$ then $B - p = B$
- 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 $\text{Pol} \Gamma$...) \text{BK}
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

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

- It is 1-minimal ($P_{xy}$ is a subdirect subset of $P_x \times P_y$)
- If $B + p = B$ then $B - p = B$
- 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 **K**
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

QED
General case

- SDPOpt(Γ) = 1 − ε, ε small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If δ is big enough then for almost all x, y we have $P_{xy} \subseteq R_{xy}$
- If δ is small enough then the calculations will almost work...

..........................................................
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

..............................................................................

..............................................................................
General case

- SDPOpt(Γ) = 1 − ε, ε small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If δ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If δ is small enough then the calculations will almost work...
- ..........................................................................................
- ..........................................................................................
- .....W........W.....OOO.....RRR........K.....K........
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

QED
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

\[\ldots W \ldots W \ldots OOO \ldots RRR \ldots K \ldots K \ldots\]
\[\ldots W \ldots W \ldots O \ldots O \ldots R \ldots R \ldots K \ldots K \ldots\]
\[\ldots W \ldots W \ldots W \ldots O \ldots O \ldots RRR \ldots K K \ldots\]
\[ \text{SDP}_{\text{Opt}}(\Gamma) = 1 - \varepsilon, \varepsilon \text{ small} \]

We define \( P_{xy} = \{(a, b) : x_a b > \delta\} \)

If \( \delta \) is big enough then for almost all \( x, y \) we have \( P_{xy} \subseteq R_{xy} \)

If \( \delta \) is small enough then the calculations will almost work...

\[ \text{QED} \]
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

QED
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................
General case

- $\text{SDPOpt}(\Gamma) = 1 - \varepsilon$, $\varepsilon$ small
- We define $P_{xy} = \{(a, b) : x_a x_b > \delta\}$
- If $\delta$ is big enough then for almost all $x, y$ we have $P_{xy} \subseteq R_{xy}$
- If $\delta$ is small enough then the calculations will almost work...

QED
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?

Wild guess: NU $\Rightarrow$ 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?

  Wild guess: NU $\Rightarrow$ polynomial loss

- SDP, LP outputs $\leftrightarrow$ consistency notions (within CSP).
  What is the precise connection?
  Is there any connection beyond CSPs?
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 $\epsilon$ in general?
  
  Wild guess: NU $\Rightarrow$ polynomial loss

- SDP, LP outputs $\leftrightarrow$ consistency notions (within CSP).
  
  What is the precise connection?
  Is there any connection beyond CSPs?

- Thank you!