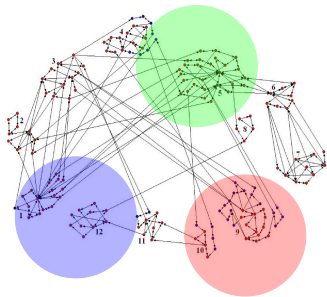


From Weak to Strong LP Gaps for all CSPs



Mrinalkanti Ghosh
Madhur Tulsiani

TTI-Chicago

Max-k-CSP

Max-k-CSP

- n Boolean variables.

Max-k-CSP

- n Boolean variables.
- m constraints (each on k variables)

Max-k-CSP

- n Boolean variables.
- m constraints (each on k variables)
- Satisfy as many as possible.

Max-3-SAT

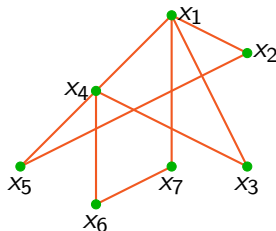
$$x_1 \vee x_{22} \vee \bar{x}_{19}$$

$$x_3 \vee \bar{x}_9 \vee x_{23}$$

$$x_5 \vee \bar{x}_7 \vee \bar{x}_9$$

⋮

Max-Cut



$$x_1 \neq x_2$$

$$x_2 \neq x_5$$

$$x_3 \neq x_4$$

⋮

Max-k-CSP

- n Boolean variables.
- m constraints (each on k variables)
- Satisfy as many as possible.

Max-3-SAT

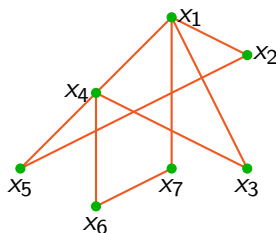
$$x_1 \vee x_{22} \vee \bar{x}_{19}$$

$$x_3 \vee \bar{x}_9 \vee x_{23}$$

$$x_5 \vee \bar{x}_7 \vee \bar{x}_9$$

⋮

Max-Cut



$$x_1 \neq x_2$$

$$x_2 \neq x_5$$

$$x_3 \neq x_4$$

⋮

Fundamental class of optimization problems.

Max-k-CSP_q

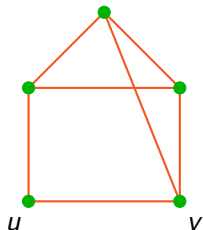
Max-k-CSP_q

- n variables taking values in $[q] = \{0, \dots, q - 1\}$.
- m constraints (each on k variables)
- Satisfy as many as possible.

Max-k-CSP_q

- n variables taking values in $[q] = \{0, \dots, q - 1\}$.
- m constraints (each on k variables)
- Satisfy as many as possible.

Unique Games



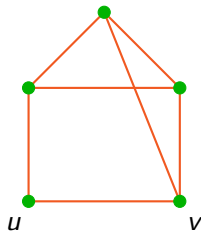
- For a graph, given:
 - **Set of colors:** $[q]$
 - **Constraints:** one for each edge $(u, v) \in E$

$$(u, v) = \left\{ \begin{array}{c} \text{green } u \\ \text{blue } v \end{array} \text{ or } \begin{array}{c} \text{blue } u \\ \text{red } v \end{array} \text{ or } \begin{array}{c} \text{red } u \\ \text{green } v \end{array} \right\}$$

Max-k-CSP_q

- n variables taking values in $[q] = \{0, \dots, q - 1\}$.
- m constraints (each on k variables)
- Satisfy as many as possible.

Unique Games



- For a graph, given:
 - **Set of colors:** $[q]$
 - **Constraints:** one for each edge $(u, v) \in E$

$$(u, v) = \left\{ \begin{array}{c} \text{green } u \\ | \\ \text{blue } v \end{array} \text{ or } \begin{array}{c} \text{blue } u \\ | \\ \text{red } v \end{array} \text{ or } \begin{array}{c} \text{red } u \\ | \\ \text{green } v \end{array} \right\}$$

- Each constraint is a bijection from $[q]$ to $[q]$.
Can in fact consider difference equations

$$x_u - x_v = c_{uv} \pmod{q}$$

Max-k-CSP_q(f)

- Characterized by $f : [q]^k \rightarrow \{0, 1\}$.

Max-k-CSP_q(f)

- Characterized by $f : [q]^k \rightarrow \{0, 1\}$.
- Each constraint is of the form

$$C_i \equiv f(x_{i_1} + b_{i,1}, \dots, x_{i_k} + b_{i,k})$$

for $i_1, \dots, i_k \in [n]$ and $b_{i,1}, \dots, b_{i,k} \in [q]$. (addition is mod q)

Max-k-CSP_q(f)

- Characterized by $f : [q]^k \rightarrow \{0, 1\}$.
- Each constraint is of the form

$$C_i \equiv f(x_{i_1} + b_{i,1}, \dots, x_{i_k} + b_{i,k})$$

for $i_1, \dots, i_k \in [n]$ and $b_{i,1}, \dots, b_{i,k} \in [q]$. (addition is mod q)

- **Max-3-SAT**: $f \equiv \text{OR}$. Each C_i is a clause. $b_{i,1} = 1$ if x_{i_1} is negated in clause C_i .

Max-k-CSP_q(f)

- Characterized by $f : [q]^k \rightarrow \{0, 1\}$.
- Each constraint is of the form

$$C_i \equiv f(x_{i_1} + b_{i,1}, \dots, x_{i_k} + b_{i,k})$$

for $i_1, \dots, i_k \in [n]$ and $b_{i,1}, \dots, b_{i,k} \in [q]$. (addition is mod q)

- **Max-3-SAT**: $f \equiv \text{OR}$. Each C_i is a clause. $b_{i,1} = 1$ if x_{i_1} is negated in clause C_i .
- **Unique Games**: $f \equiv \text{EQUAL}$. For i^{th} constraint (u, v) , let $i_1 = u$, $i_2 = v$ and let $b_{i,2} - b_{i,1} = c_{uv}$

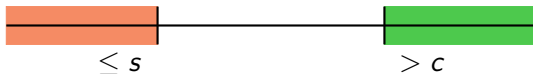
$$x_u - x_v = c_{uv} \quad \Leftrightarrow \quad x_{i_1} + b_{i,1} = x_{i_2} + b_{i,2}.$$

Approximating $\text{Max-k-CSP}_q(f)$

Relax the problem of finding **maximum fraction** of constraints satisfiable.

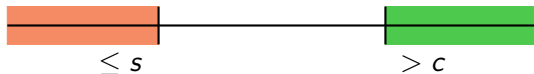
Approximating Max-k-CSP_q(f)

Relax the problem of finding **maximum fraction** of constraints satisfiable.



Approximating Max-k-CSP_q(f)

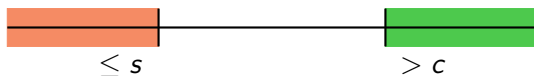
Relax the problem of finding **maximum fraction** of constraints satisfiable.



- **Goal:** Given f , characterize all pairs (s, c) for which the distinguishing problem can be solved.

Approximating Max-k-CSP_q(f)

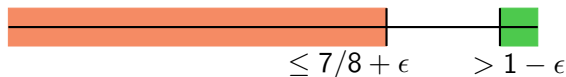
Relax the problem of finding **maximum fraction** of constraints satisfiable.



- **Goal:** Given f , characterize all pairs (s, c) for which the distinguishing problem can be solved.
- If for some $\gamma \leq 1$, all pairs $(\gamma \cdot c, c)$ can be solved, then can approximate within factor γ .

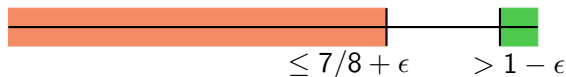
Characterizing approximability

- Max-3-SAT [Håstad 97]: For all $\epsilon > 0$, distinguishing $(7/8 + \epsilon, 1 - \epsilon)$ is NP-hard ($s < 7/8$ is trivial).

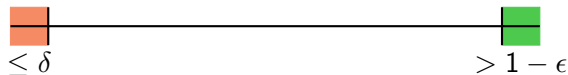


Characterizing approximability

- **Max-3-SAT [Håstad 97]**: For all $\epsilon > 0$, distinguishing $(7/8 + \epsilon, 1 - \epsilon)$ is NP-hard ($s < 7/8$ is trivial).



- **Unique Games Conjecture [Khot 02]**: For all $\delta, \epsilon > 0$, there exists q such that it is NP-hard to distinguish $(\delta, 1 - \epsilon)$ for UG with domain $[q]$.



An ultimate result assuming the UGC

- [Raghavendra 08]: For all q , for all f , if a **basic SDP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, it is NP-hard to distinguish $(s + \epsilon, c - \epsilon)$ **assuming the UGC**.

An ultimate result assuming the UGC

- [Raghavendra 08]: For all q , for all f , if a **basic SDP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, it is NP-hard to distinguish $(s + \epsilon, c - \epsilon)$ **assuming the UGC**.
- “All-or-nothing”: Either a simple algorithm (approximately solvable in almost linear time) can distinguish (s, c) or it is NP-hard to do so.

An ultimate result assuming the UGC

- [Raghavendra 08]: For all q , for all f , if a **basic SDP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, it is NP-hard to distinguish $(s + \epsilon, c - \epsilon)$ **assuming the UGC**.
- “All-or-nothing”: Either a simple algorithm (approximately solvable in almost linear time) can distinguish (s, c) or it is NP-hard to do so.
- Equivalent to UGC (because UG is a 2-CSP).

- For all q , for all f , if a **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP of any polynomial size in the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.

- For all q , for all f , if a **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP of any polynomial size in the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.
- [CLRS 13] If no polysize LP in Sherali-Adams hierarchy can distinguish $(s + \epsilon, c - \epsilon)$ then no polysize **extended formulation** can distinguish $(s + 2\epsilon, c - 2\epsilon)$.

- For all q , for all f , if a **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP of any polynomial size in the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.
- [CLRS 13] If no polysize LP in Sherali-Adams hierarchy can distinguish $(s + \epsilon, c - \epsilon)$ then no polysize **extended formulation** can distinguish $(s + 2\epsilon, c - 2\epsilon)$.
- “All-or-not-much” for LPs: If a simple (almost linear time) LP cannot do it, neither can any polysize LP extended formulation (captures all “natural” LPs).

Extended formulations

- Defined by a **feasible polytope** P , and a way of **encoding** instances Φ as a (linear) objective function w_Φ .

Extended formulations

- Defined by a **feasible polytope** P , and a way of **encoding** instances Φ as a (linear) objective function w_Φ .
- Introduce additional variables y . Optimize over polytope $Q = \{x \mid \exists y \ Ex + Fy = g, y \geq 0\}$.

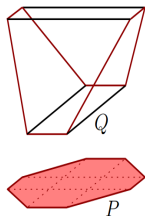


Image from [Fiorini-Rothvoss-Tiwari 2011]

Extended formulations

- Defined by a **feasible polytope** P , and a way of **encoding** instances Φ as a (linear) objective function w_Φ .
- Introduce additional variables y . Optimize over polytope
$$P = \{x \mid \exists y \quad Ex + Fy = g, y \geq 0\} .$$

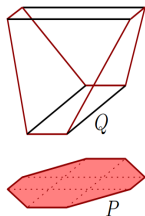


Image from [Fiorini-Rothvoss-Tiwari 2011]

Extended formulations

- Defined by a **feasible polytope** P , and a way of **encoding** instances Φ as a (linear) objective function w_Φ .
- Introduce additional variables y . Optimize over polytope $Q = \{x \mid \exists y \ E x + F y = g, y \geq 0\}$.

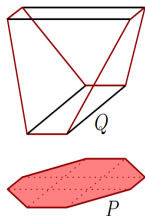


Image from [Fiorini-Rothvoss-Tiwari 2011]

Size equals $\#variables + \#constraints$.

Extended formulations

- Defined by a **feasible polytope** P , and a way of **encoding** instances Φ as a (linear) objective function w_Φ .
- Introduce additional variables y . Optimize over polytope $Q = \{x \mid \exists y \quad Ex + Fy = g, y \geq 0\}$.

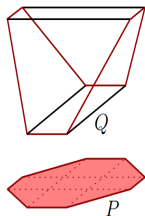


Image from [Fiorini-Rothvoss-Tiwari 2011]

Size equals **#variables** + **#constraints**.

- Optimize objective $\langle w_\Phi, x \rangle$ (depending on Φ) over P .

The Sherali-Adams hierarchy (t levels)

The Sherali-Adams hierarchy (t levels)

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α

The Sherali-Adams hierarchy (t levels)

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α (you wish!)

The Sherali-Adams hierarchy (t levels)

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α (you wish!)

Consistency: For all $j \notin S$, $\sum_{b \in [q]} X_{(S \cup \{j\}, \alpha \circ b)} = X_{(S,\alpha)}$

The Sherali-Adams hierarchy (t levels)

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α (you wish!)

Consistency: For all $j \notin S$, $\sum_{b \in [q]} X_{(S \cup \{j\}, \alpha \circ b)} = X_{(S,\alpha)}$

Linear Program: For variables $X_{(S,\alpha)} \in [0, 1]$ satisfying consistency

$$\text{Maximize } \frac{1}{m} \cdot \sum_{C_i} \sum_{\alpha \in [q]^{S_{C_i}}} X_{(S_{C_i}, \alpha)} \cdot f(\alpha_{i_1} + b_{i,1}, \dots, \alpha_{i_k} + b_{i,k})$$

(S_{C_i} denotes set of variables in constraint C_i .)

But what does it all mean??

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

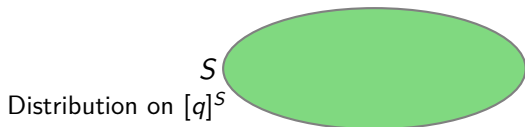
\approx Probability that vars in S assigned according to α

But what does it all mean??

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α

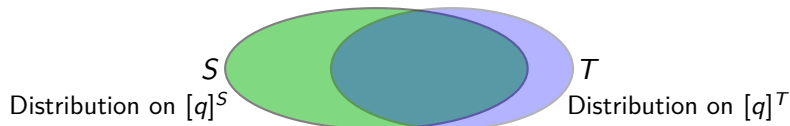


But what does it all mean??

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α

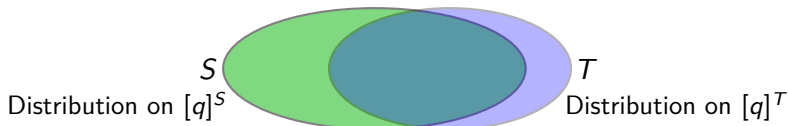


But what does it all mean??

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α



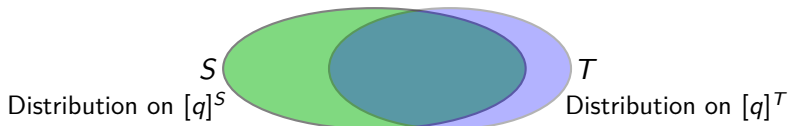
- Solution to LP defines local distributions consistent on intersections.

But what does it all mean??

Variables: For $|S| \leq t$ and $\alpha \in [q]^S$ define $X_{(S,\alpha)}$. Supposed to be

$$X_{(S,\alpha)} = \begin{cases} 1 & \text{if all variables in } S \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S assigned according to α



- Solution to LP defines local distributions consistent on intersections.
- $n^{O(t)} \cdot q^t$ variables.

The basic LP

The basic LP

- **Variables:** For S_{C_i} and $\alpha \in [q]^{S_{C_i}}$ define $X_{(S_{C_i}, \alpha)}$. Supposed to be

$$X_{(S_{C_i}, \alpha)} = \begin{cases} 1 & \text{if all variables in } S_{C_i} \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S_{C_i} assigned according to α

Also define $X_{(j,b)}$ for each $j \in [n], b \in [q]$.

The basic LP

- **Variables:** For S_{C_i} and $\alpha \in [q]^{S_{C_i}}$ define $X_{(S_{C_i}, \alpha)}$. Supposed to be

$$X_{(S_{C_i}, \alpha)} = \begin{cases} 1 & \text{if all variables in } S_{C_i} \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S_{C_i} assigned according to α

Also define $X_{(j, b)}$ for each $j \in [n]$, $b \in [q]$.

- **Consistency:** $\forall j \in S_{C_i}, \forall b \in [q], \sum_{\substack{\alpha \in [q]^{S_{C_i}} \\ \alpha(j)=b}} X_{(S_{C_i}, \alpha)} = X_{(j, b)}$

The basic LP

- **Variables:** For S_{C_i} and $\alpha \in [q]^{S_{C_i}}$ define $X_{(S_{C_i}, \alpha)}$. Supposed to be

$$X_{(S_{C_i}, \alpha)} = \begin{cases} 1 & \text{if all variables in } S_{C_i} \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S_{C_i} assigned according to α

Also define $X_{(j,b)}$ for each $j \in [n]$, $b \in [q]$.

- **Consistency:** $\forall j \in S_{C_i}, \forall b \in [q], \sum_{\substack{\alpha \in [q]^{S_{C_i}} \\ \alpha(j)=b}} X_{(S_{C_i}, \alpha)} = X_{(j,b)}$



The basic LP

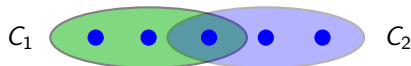
- **Variables:** For S_{C_i} and $\alpha \in [q]^{S_{C_i}}$ define $X_{(S_{C_i}, \alpha)}$. Supposed to be

$$X_{(S_{C_i}, \alpha)} = \begin{cases} 1 & \text{if all variables in } S_{C_i} \text{ are assigned according to } \alpha \\ 0 & \text{otherwise} \end{cases}$$

\approx Probability that vars in S_{C_i} assigned according to α

Also define $X_{(j,b)}$ for each $j \in [n], b \in [q]$.

- **Consistency:** $\forall j \in S_{C_i}, \forall b \in [q], \sum_{\substack{\alpha \in [q]^{S_{C_i}} \\ \alpha(j)=b}} X_{(S_{C_i}, \alpha)} = X_{(j,b)}$

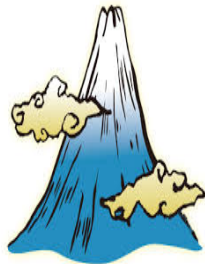


- $O(q^k \cdot m + q \cdot n)$ variables.

Inaccurate pictorial representations

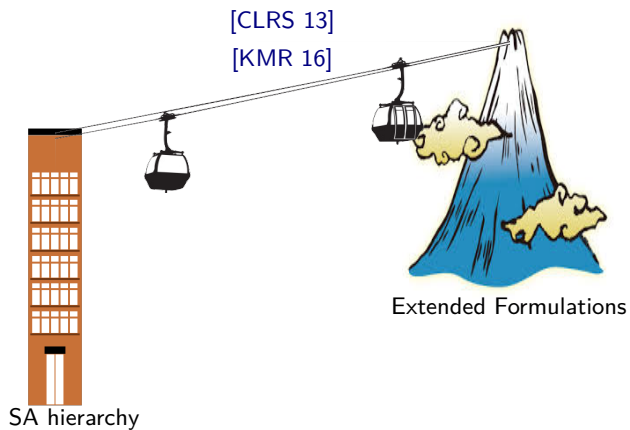


SA hierarchy

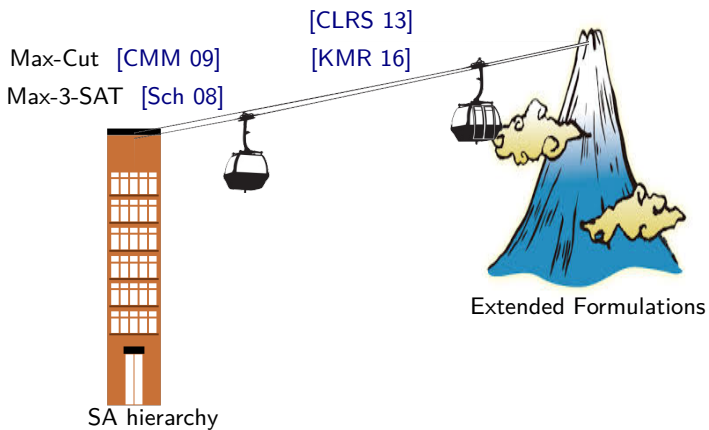


Extended Formulations

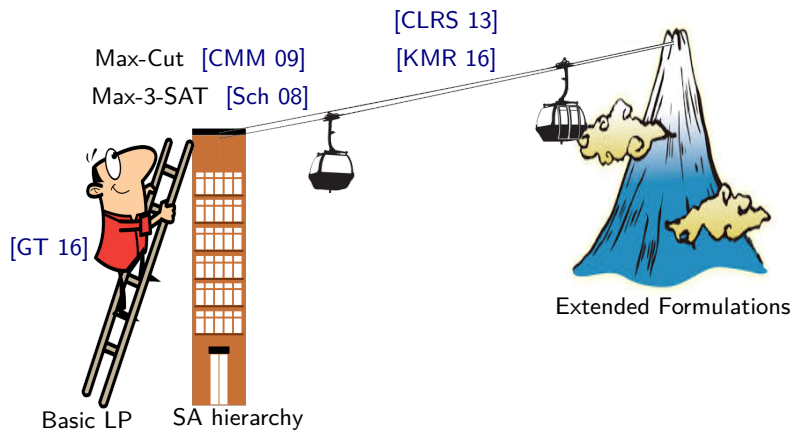
Inaccurate pictorial representations



Inaccurate pictorial representations



Inaccurate pictorial representations



An “All or not-much” phenomenon

- [Ghosh T 16]: For all q , for all f , if **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP given by $t = O_\epsilon\left(\frac{\log n}{\log \log n}\right)$ levels of the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.

An “All or not-much” phenomenon

- [Ghosh T 16]: For all q , for all f , if **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP given by $t = O_\epsilon \left(\frac{\log n}{\log \log n} \right)$ levels of the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.
- Using [CLRS 13, KMR 16]: For all $\epsilon > 0$, no **extended formulation** of size $\exp \left(O_\epsilon \left(\frac{(\log n)^2}{(\log \log n)^2} \right) \right)$ can distinguish $(s + \epsilon, c - \epsilon)$.

An “All or not-much” phenomenon

- [Ghosh T 16]: For all q , for all f , if **basic LP** cannot distinguish (s, c) for $\text{Max-k-CSP}_q(f)$, then for all $\epsilon > 0$, no LP given by $t = O_\epsilon \left(\frac{\log n}{\log \log n} \right)$ levels of the **Sherali-Adams** hierarchy can distinguish $(s + \epsilon, c - \epsilon)$.
- Using [CLRS 13, KMR 16]: For all $\epsilon > 0$, no **extended formulation** of size $\exp \left(O_\epsilon \left(\frac{(\log n)^2}{(\log \log n)^2} \right) \right)$ can distinguish $(s + \epsilon, c - \epsilon)$.
- “Amplify” a hard instance for basic LP to a hard instance for Sherali-Adams.

What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if

What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if
 - No assignment satisfies more than s fraction of constraints.

What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if
 - No assignment satisfies more than s fraction of constraints.
 - All local distributions on constraints are supported **only on satisfying assignments**.



What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if
 - No assignment satisfies more than s fraction of constraints.
 - All local distributions on constraints are supported **only on satisfying assignments**.



- Using Φ_0 , create a (level- t) hard instance Φ where
 - No assignment satisfies more than s fraction of constraints.

What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if
 - No assignment satisfies more than s fraction of constraints.
 - All local distributions on constraints are supported **only on satisfying assignments**.



- Using Φ_0 , create a (level- t) hard instance Φ where
 - No assignment satisfies more than s fraction of constraints.
 - There exist local distributions on all subsets S , $|S| \leq t$, consistent on all intersections.

What is a hard instance ($c = 1$)

- Φ_0 is a (c, s) hard instance of basic LP, for $c = 1$ if
 - No assignment satisfies more than s fraction of constraints.
 - All local distributions on constraints are supported **only on satisfying assignments**.



- Using Φ_0 , create a (level- t) hard instance Φ where
 - No assignment satisfies more than s fraction of constraints.
 - There exist local distributions on all subsets S , $|S| \leq t$, consistent on all intersections.
 - Distribution on S only supported on assignments satisfying (almost) **all constraints in S** .

Intuition for the proof

- Use hard instance (say Φ_0) for basic LP as a “template” to produce a hard instance Φ for Sherali-Adams.

Intuition for the proof

- Use hard instance (say Φ_0) for basic LP as a “template” to produce a hard instance Φ for Sherali-Adams.
- Instance Φ looks “easily satisfiable” locally.

Intuition for the proof

- Use hard instance (say Φ_0) for basic LP as a “template” to produce a hard instance Φ for Sherali-Adams.
- Instance Φ looks “easily satisfiable” locally.
- Think of instance as (hyper)graph. Each constraint adds a hyperedge. Locally like (hyper)trees.

Intuition for the proof

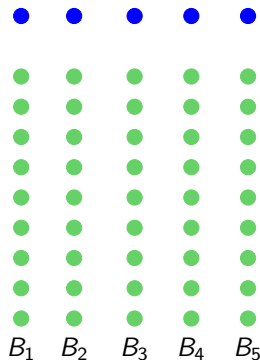
- Use hard instance (say Φ_0) for basic LP as a “template” to produce a hard instance Φ for Sherali-Adams.
- Instance Φ looks “easily satisfiable” locally.
- Think of instance as (hyper)graph. Each constraint adds a hyperedge. Locally like (hyper)trees.
- Trees are easy.

The gap construction

- Will use (s, c) hard instance Φ_0 for basic LP as template.

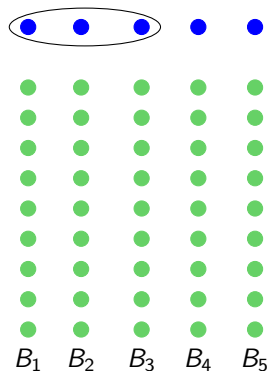


The gap construction



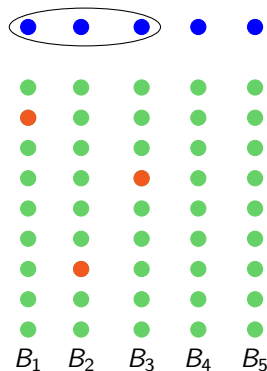
- Will use (s, c) hard instance Φ_0 for basic LP as template.
- Consider a bucket of variables B_r for every variable x_r in Φ_0 . $|B_r| = n$.

The gap construction



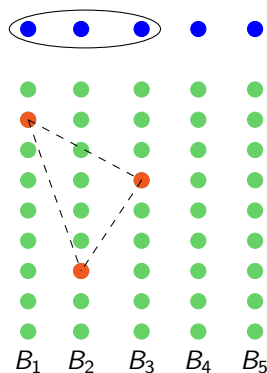
- Will use (s, c) hard instance Φ_0 for basic LP as template.
- Consider a bucket of variables B_r for every variable x_r in Φ_0 . $|B_r| = n$.
- Repeat m times:
 - Sample $C \sim \Phi_0$. Let $C \equiv f(x_{i_1} + b_{i_1,1}, \dots, x_{i_k} + b_{i_k,k})$.

The gap construction



- Will use (s, c) hard instance Φ_0 for basic LP as template.
- Consider a bucket of variables B_r for every variable x_r in Φ_0 . $|B_r| = n$.
- Repeat m times:
 - Sample $C \sim \Phi_0$. Let $C \equiv f(x_{i_1} + b_{i,1}, \dots, x_{i_k} + b_{i,k})$.
 - Pick j^{th} variable uniformly from bucket B_{i_j} . Let z_{i_j} be the sampled variable from this bucket.

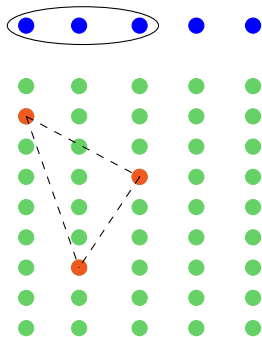
The gap construction



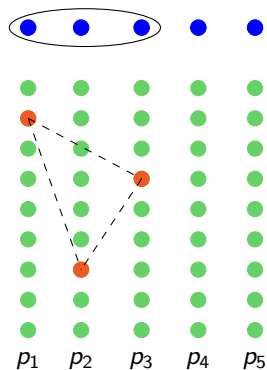
- Will use (s, c) hard instance Φ_0 for basic LP as template.
- Consider a bucket of variables B_r for every variable x_r in Φ_0 . $|B_r| = n$.
- Repeat m times:
 - Sample $C \sim \Phi_0$. Let $C \equiv f(x_{i_1} + b_{i_1,1}, \dots, x_{i_k} + b_{i_k,k})$.
 - Pick j^{th} variable uniformly from bucket B_{i_j} . Let z_{i_j} be the sampled variable from this bucket.
 - Include constraint $f(z_{i_1} + b_{i_1,1}, \dots, z_{i_k} + b_{i_k,k})$.

Proving soundness

- Fix an assignment σ to all vars in new instance Φ

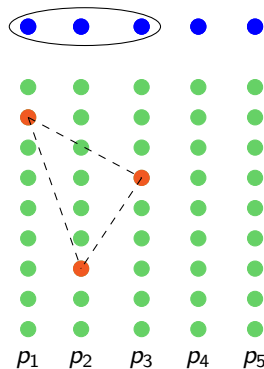


Proving soundness



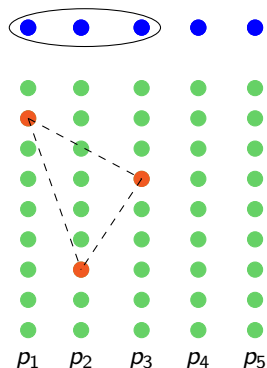
- Fix an assignment σ to all vars in new instance Φ
- Let p_r be the empirical distribution on $[q]$ for variables in B_r .

Proving soundness



- Fix an assignment σ to all vars in new instance Φ
- Let p_r be the empirical distribution on $[q]$ for variables in B_r .
- Let x_r be a var in constraint $C \in \Phi_0$. A random copy of C sees a value for this variable independently distributed with p_r .

Proving soundness

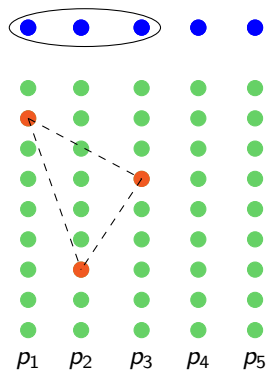


- Fix an assignment σ to all vars in new instance Φ
- Let p_r be the empirical distribution on $[q]$ for variables in B_r .
- Let x_r be a var in constraint $C \in \Phi_0$. A random copy of C sees a value for this variable independently distributed with p_r .
- For a fixed σ ,

$$\mathbb{E}_{\Phi} [\text{Fraction of sat. constraints in } \Phi]$$

equals fraction satisfied in Φ_0 by rounding each x_r independently from p_r ($\leq s$).

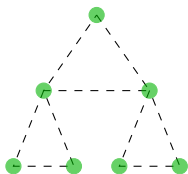
Proving soundness



- Fix an assignment σ to all vars in new instance Φ
- Let p_r be the empirical distribution on $[q]$ for variables in B_r .
- Let x_r be a var in constraint $C \in \Phi_0$. A random copy of C sees a value for this variable independently distributed with p_r .
- For a fixed σ ,
$$\mathbb{E}_\Phi [\text{Fraction of sat. constraints in } \Phi]$$
equals fraction satisfied in Φ_0 by rounding each x_r independently from p_r ($\leq s$).
- Concentration and union bound.

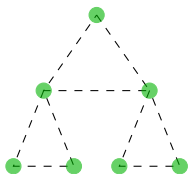
Propagation on trees

- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.

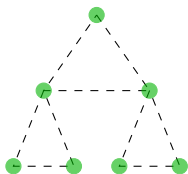


Propagation on trees

- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).

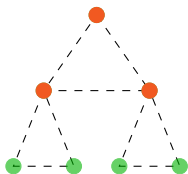


Propagation on trees



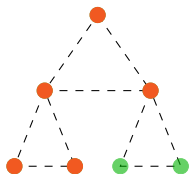
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).

Propagation on trees



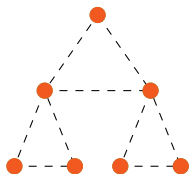
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).

Propagation on trees



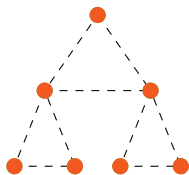
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).

Propagation on trees



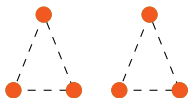
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).

Propagation on trees



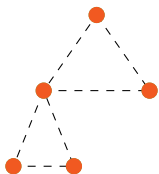
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).
- Does not depend on choice of root.

Propagation on trees



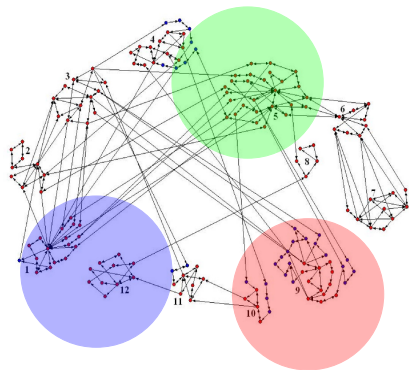
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).
- Does not depend on choice of root.
- May not be consistent between tree and disconnected sub-forest.

Propagation on trees



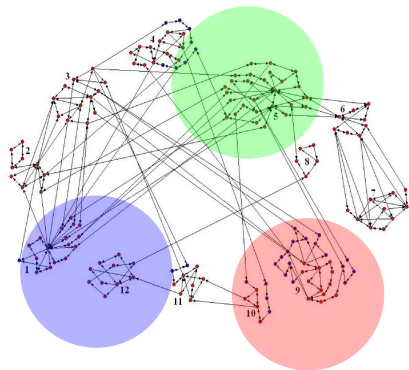
- Random hypergraphs have no cycles of size $O(\log n)$. Locally like trees.
- Each hyperedge e in a tree comes from a constraint in Φ_0 . Comes with a given distribution on e (from basic LP).
- Propagate to child conditioned on parent. Can be done by consistency on variables (vertices).
- Does not depend on choice of root.
- May not be consistent between tree and disconnected sub-forest.
- Is consistent on a **subtree**.

Breaking up the graph



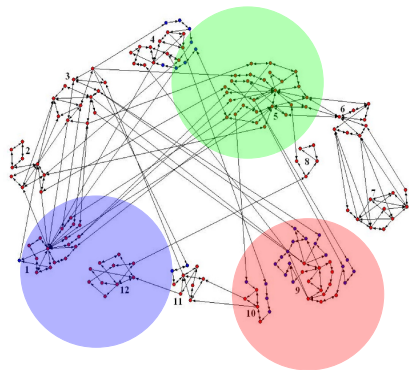
- **Idea:** Given set $S \subseteq V$, break S into low-diameter components. Connect all paths in each component - always a tree, never a forest.

Breaking up the graph



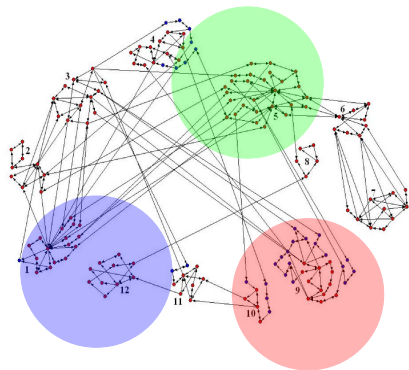
- **Idea:** Given set $S \subseteq V$, break S into low-diameter components. Connect all paths in each component - always a tree, never a forest.
- Propagate on each component tree.

Breaking up the graph



- **Idea:** Given set $S \subseteq V$, break S into low-diameter components. Connect all paths in each component - always a tree, never a forest.
- Propagate on each component tree.
- If $T \subset S$, components of T induced by S should be same as obtained by partitioning T .

Breaking up the graph



- **Idea:** Given set $S \subseteq V$, break S into low-diameter components. Connect all paths in each component - always a tree, never a forest.
- Propagate on each component tree.
- If $T \subset S$, components of T induced by S should be same as obtained by partitioning T .
- Cut only few edges.

Subset consistent partitioning schemes

Subset consistent partitioning schemes

- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

Subset consistent partitioning schemes

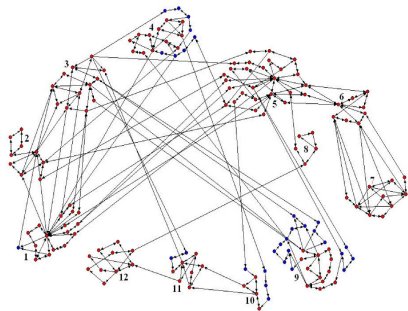
- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

- [CCGGP 98]: Low-diameter decomposition of ℓ_2 embedding.

Subset consistent partitioning schemes



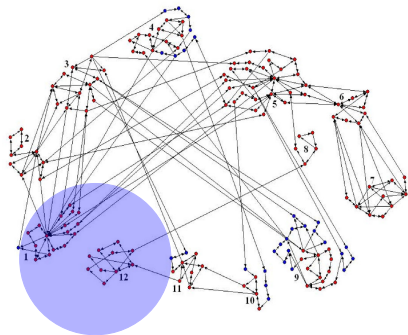
- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

- [CCGGP 98]: Low-diameter decomposition of ℓ_2 embedding.

Subset consistent partitioning schemes



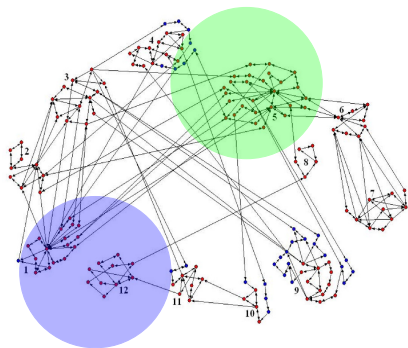
- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

- [CCGGP 98]: Low-diameter decomposition of ℓ_2 embedding.

Subset consistent partitioning schemes



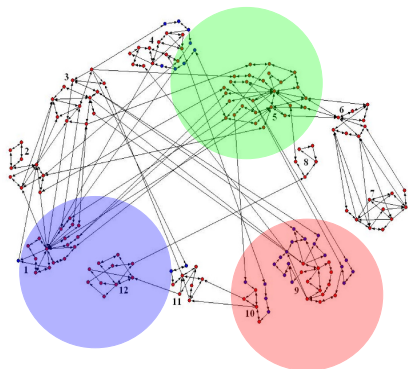
- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

- [CCGGP 98]: Low-diameter decomposition of ℓ_2 embedding.

Subset consistent partitioning schemes



- [CMM 07]: Define a metric ρ on random (hyper)graph H

$$\rho(u, v) \approx \sqrt{1 - \exp(-\mu \cdot d_H(u, v))}$$

ρ embeds in ℓ_2 on small sets S (for small enough μ).

- [CCGGP 98]: Low-diameter decomposition of ℓ_2 embedding.
- Easy to check partitioning is consistent on subsets.

The dimensionality problem

- Low-diameter decomposition in \mathbb{R}^d cuts each edge with probability $O(\sqrt{\mu \cdot d})$.

The dimensionality problem

- Low-diameter decomposition in \mathbb{R}^d cuts each edge with probability $O(\sqrt{\mu \cdot d})$.
- For $|S| = t$, ℓ_2 embedding is in \mathbb{R}^t . Probability of cutting an edge is $O(\sqrt{\mu \cdot t})$. Limits t to $O(\frac{\log n}{\log \log n})$.

The dimensionality problem

- Low-diameter decomposition in \mathbb{R}^d cuts each edge with probability $O(\sqrt{\mu \cdot d})$.
- For $|S| = t$, ℓ_2 embedding is in \mathbb{R}^t . Probability of cutting an edge is $O(\sqrt{\mu \cdot t})$. Limits t to $O(\frac{\log n}{\log \log n})$.
- [JL 84]: Random Gaussian projection in $O(\log t)$ dimensions approximately preserves all distances **with high probability**.

The dimensionality problem

- Low-diameter decomposition in \mathbb{R}^d cuts each edge with probability $O(\sqrt{\mu \cdot d})$.
- For $|S| = t$, ℓ_2 embedding is in \mathbb{R}^t . Probability of cutting an edge is $O(\sqrt{\mu \cdot t})$. Limits t to $O(\frac{\log n}{\log \log n})$.
- [JL 84]: Random Gaussian projection in $O(\log t)$ dimensions approximately preserves all distances **with high probability**.
- For sets S and T , can one **consistently** discard bad Gaussian projections?

Open Problems

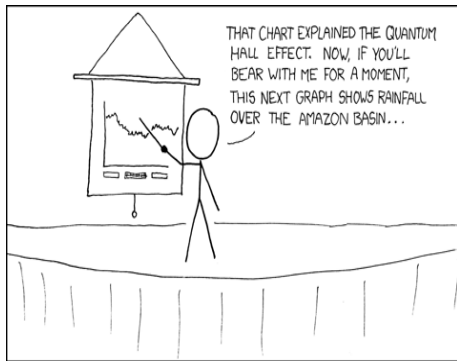
- Extend the result to $\Omega(n)$ levels of the SA hierarchy. Will give a size bound of $\Omega(\exp(n^{O(1)}))$ on extended formulation size using [KMR16].

Open Problems

- Extend the result to $\Omega(n)$ levels of the SA hierarchy. Will give a size bound of $\Omega(\exp(n^{O(1)}))$ on extended formulation size using [KMR16].
- “All-or-nothing” for SDP hierarchies. Would go a long way towards proving the UGC. Even results for specific CSPs would be interesting ($k \geq 3?$).

Open Problems

- Extend the result to $\Omega(n)$ levels of the SA hierarchy. Will give a size bound of $\Omega(\exp(n^{O(1)}))$ on extended formulation size using [KMR16].
- “All-or-nothing” for SDP hierarchies. Would go a long way towards proving the UGC. Even results for specific CSPs would be interesting ($k \geq 3?$).
- Perhaps in the worst case nothing does better than basic LP/SDP. Are there testable properties of the instance, under which it is better to use higher levels in the hierarchies.



IF YOU KEEP SAYING "BEAR WITH ME FOR A MOMENT",
PEOPLE TAKE A WHILE TO FIGURE OUT THAT
YOU'RE JUST SHOWING THEM RANDOM SLIDES.

Thank You

Questions?