Improved NP-inapproximability for 2-variable linear equations

Johan Håstad  
KTH

Sangxia Huang  
KTH

Rajsekar Manokaran  
KTH

Ryan O’Donnell  
CMU

John Wright  
CMU
2Lin

\[ x_1 = x_5 \]
\[ x_{10} = -x_3 \]
\[ x_{61} = -x_{24} \]
\[ \ldots \]
\[ x_{48} = -x_5 \]

\((x_i = -1, 1)\)
\[2\text{Lin} \quad 2\text{Lin}(2) \in 2\text{Lin}(q) \approx \text{UniqueGames}\]

\[
x_1 = x_5 \\
x_{10} = -x_3 \\
x_{61} = -x_{24} \\
\ldots \\
x_{48} = -x_5 \\
(x_i = -1, 1)
\]
\begin{align*}
\text{2Lin} & \quad 2\text{Lin}(2) \in 2\text{Lin}(q) \approx \textbf{UniqueGames} \\
 x_1 &= x_5 \\
x_{10} &= -x_3 \\
x_{61} &= -x_{24} \\
 \cdots & \\
x_{48} &= -x_5 \\
(x_i &= -1,1)
\end{align*}

(Actually, simplest case of UG)
$2\text{Lin}$

$x_1 = x_5$

$x_{10} = -x_3$

$x_{61} = -x_{24}$

$...$

$x_{48} = -x_5$

$(x_i = -1,1)$

$2\text{Lin}(2) \in 2\text{Lin}(q) \approx \text{UniqueGames}$

(Actually, simplest case of $\text{UG}$)

Folklore wisdom: get $2\text{Lin}(2)$ right and $2\text{Lin}(q)$ will follow.
Known results

Suppose $\text{val}(I) = \alpha$. Can we guarantee a solution of value $C\alpha$?
Known results

Suppose \( \text{val}(I) = \alpha \). Can we guarantee a solution of value \( C^* \alpha \)?

[GW]: .878-approx algorithm
Known results

Suppose \text{val}(I) = \alpha. Can we guarantee a solution of value $C^*\alpha$?

\textbf{[GW]}: .878-approx algorithm

\textbf{[KKMO]}+[\text{MOO}]: (.878+\varepsilon)-approx \text{UG-hard}
Known results

Suppose $\text{val}(I) = \alpha$. Can we guarantee a solution of value $C^*\alpha$?

[GW]: .878-approx algorithm

[KKMO]+[MOO]: $(.878+\varepsilon)$-approx UG-hard

[Håstad]+[TSSW]: $\frac{16}{17} \approx .941$-approx NP-hard
Known results

Suppose $\text{val}(I) = \alpha$. Can we guarantee a solution of value $C^*\alpha$?

[GW]: .878-approx algorithm

[KKMO]+[MOO]: $(.878+\varepsilon)$-approx $\text{UG}$-hard

[Håstad]+[TSSW]: $\frac{16}{17} \approx .941$-approx $\text{NP}$-hard

seems we’re close, right?
A different perspective...

Suppose \( \text{val}(I) = (1 - \varepsilon) \).

Can we guarantee a solution of value \( (1 - C\varepsilon) \)?
A different perspective...

Suppose \( \text{val}(I) = (1 - \varepsilon) \).
Can we guarantee a solution of value \((1 - C^*\varepsilon)\)?

**Def:** Such an algo. gives an \((\varepsilon, C^*\varepsilon)\)-approx.
A different perspective...

Suppose \( \text{val}(I) = (1 - \varepsilon) \).
Can we guarantee a solution of value \((1 - f(\varepsilon))\)?

**Def:** Such an algo. gives an \((\varepsilon, f(\varepsilon))\)-approx.
A different perspective...

Suppose \( \text{val}(I) = (1 - \varepsilon) \).
Can we guarantee a solution of value \( (1 - f(\varepsilon)) \)?

**Def:** Such an algo. gives an \((\varepsilon, f(\varepsilon))\)-approx.

Usually called “Min-2Lin(2)-Deletion”.
Let me just call this 2Lin.
Unratio state of affairs

[easy]: (ε, ε)-approx NP-hard
Unratio state of affairs

[easy]: \((\varepsilon, \varepsilon)\)-approx \text{NP-hard}

[Håstad]+[TSSW]: \((\varepsilon, \frac{5}{4}\varepsilon)\)-approx \text{NP-hard}
Unratio state of affairs

[easy]: $(\varepsilon, \varepsilon)$-approx $\text{NP}$-hard

[Håstad] + [TSSW]: $(\varepsilon, \frac{5}{4} \varepsilon)$-approx $\text{NP}$-hard

[KKMO] + [MOO]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx $\text{UG}$-hard
Unratio state of affairs

[easy]: $(\varepsilon, \varepsilon)$-approx NP-hard

[Håstad] + [TSSW]: $(\varepsilon, \frac{5}{4} \varepsilon)$-approx NP-hard

[KKMO] + [MOO]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx UG-hard

[GW]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx algorithm
Unratio state of affairs

[easy]: $(\varepsilon, \varepsilon)$-approx $\text{NP}$-hard

[Håstad]+[TSSW]: $(\varepsilon, \frac{5}{4} \varepsilon)$-approx $\text{NP}$-hard

[KKMO]+[MOO]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx $\text{UG}$-hard

[GW]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx algorithm

asymptotically off from the truth
Unratio state of affairs

[easy]: $(\varepsilon, \varepsilon)$-approx $\text{NP}$-hard

[Håstad]+[TSSW]: $(\varepsilon, \frac{5}{4}\varepsilon)$-approx $\text{NP}$-hard

[KKMO]+[MOO]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx $\text{UG}$-hard

[GW]: $(\varepsilon, O(\varepsilon^{1/2}))$-approx algorithm asymptotically off from the truth

[Rao]: If $(\varepsilon, O(f(q)\varepsilon^{1/2}))$-approx is $\text{NP}$-hard for $2\text{Lin}(q)$, for $f(q) = \Omega(1)$, then $\text{UG}$ is true.
This work

[Håstad]+[TSSW]: $(\varepsilon, \frac{5}{4}\varepsilon)$-approx NP-hard
This work

[Håstad]+[TSSW]: $(\epsilon, \frac{5}{4}\epsilon)$-approx NP-hard
[Us]: $(\epsilon, \frac{11}{8}\epsilon)$-approx NP-hard
This work

[Håstad]+[TSSW]: (\(\varepsilon\), 1.25*\(\varepsilon\))-approx NP-hard
[Us]: (\(\varepsilon\), 1.375*\(\varepsilon\))-approx NP-hard
This work

[Håstad]+[TSSW]: $(\varepsilon, \frac{5}{4}\varepsilon)$-approx $\text{NP}$-hard
[Us]: $(\varepsilon, \frac{11}{8}\varepsilon)$-approx $\text{NP}$-hard
This work

[Håstad]+[TSSW]: $(\varepsilon, \frac{5}{4}\varepsilon)$-approx $\text{NP}$-hard

[Us]: $(\varepsilon, \frac{11}{8}\varepsilon)$-approx $\text{NP}$-hard

Cons:
- Still haven’t proven $\text{UniqueGames}$. 😞
This work

[Håstad]+[TSSW]: \((\varepsilon, \frac{5}{4}\varepsilon)\)-approx NP-hard

[Us]: \((\varepsilon, \frac{11}{8}\varepsilon)\)-approx NP-hard

Cons:
- Still haven’t proven UniqueGames. 😞

Pros:
- First improvement since 1997.
This work

[Håstad]+[TSSW]: \((\varepsilon, \frac{5}{4}*\varepsilon)\)-approx \textbf{NP}-hard

[Us]: \((\varepsilon, \frac{11}{8}*\varepsilon)\)-approx \textbf{NP}-hard

Cons:
- Still haven’t proven \textit{UniqueGames}. 😞

Pros:
- First improvement since 1997.
- Study new type of “gadget reduction”
This work

[Åstad]+[TSSW]: $(\varepsilon, \frac{5}{4}\varepsilon)$-approx NP-hard

[Us]: $(\varepsilon, \frac{11}{8}\varepsilon)$-approx NP-hard (and more!)

Cons:
- Still haven’t proven UniqueGames. 🙁

Pros:
- First improvement since 1997.
- Study new type of “gadget reduction”
Proving \((\varepsilon, \frac{5}{4}\varepsilon)\)-hardness

Standard two-step plan.
Proving \((\varepsilon, \frac{5}{4} \varepsilon)\)-hardness

Standard two-step plan.

[Håstad]: Given 3Lin instance \(I\), NP-hard to distinguish

- **Yes**: \(\text{val}(I) \geq (1 - \varepsilon)\)
- **No**: \(\text{val}(I) \leq (\frac{1}{2} + \varepsilon)\)
Proving \((\varepsilon, \frac{5}{4} \varepsilon)\)-hardness

Standard two-step plan.

[Åhlander]: Given 3Lin instance \(I\), \(\text{NP}\)-hard to distinguish

- **Yes**: \(\text{val}(I) \geq (1 - \varepsilon)\)
- **No**: \(\text{val}(I) \leq (\frac{1}{2} + \varepsilon)\)

(In our language, \((\varepsilon, \frac{1}{2} - \varepsilon)\)-approxing 3Lin is \(\text{NP}\)-hard.)
Proving \((\varepsilon, \frac{5}{4} \varepsilon)\)-hardness

Standard two-step plan.

[Håstad]: Given 3Lin instance \(I\), \(\text{NP}\)-hard to distinguish

- **Yes**: \(\text{val}(I) \geq (1 - \varepsilon)\)
- **No**: \(\text{val}(I) \leq (\frac{1}{2} + \varepsilon)\)

(In our language, \((\varepsilon, \frac{1}{2} - \varepsilon)\)-approxing 3Lin is \(\text{NP}\)-hard.)

Step 2: gadget reduce 3Lin to 2Lin [TSSW]
Proving \((\varepsilon, \frac{5}{4} \varepsilon)\)-hardness

Standard two-step plan.

[Håstad]: Given 3Lin instance \(I\), \textbf{NP}-hard to distinguish

- \textbf{Yes}: \(\text{val}(I) \geq (1 - \varepsilon)\)
- \textbf{No}: \(\text{val}(I) \leq (\frac{1}{2} + \varepsilon)\)

(In our language, \((\varepsilon, \frac{1}{2} - \varepsilon)\)-approxing 3Lin is \textbf{NP}-hard.)

Step 2: gadget reduce 3Lin to 2Lin [TSSW]

(see also [OW12])
3Lin

\[ x_1 x_3 x_5 = 1 \]
\[ x_{10} x_{16} x_3 = -1 \]

... 

\[ x_{47} x_{11} x_{98} = -1 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]

\[ x_{10} x_{16} x_3 = -1 \]

... 

\[ x_{47} x_{11} x_{98} = -1 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]

\[ x_{10} x_{16} x_3 = -1 \]

... 

\[ x_{47} x_{11} x_{98} = -1 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]

\[ x_{10} x_{16} x_3 = -1 \]

\[ \ldots \]

\[ x_{47} x_{11} x_{98} = -1 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]

\[ x_{10} x_{16} x_3 = -1 \]

\[ x_{47} x_{11} x_{98} = -1 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]

\[ x_{10} x_{16} x_3 = -1 \]

\[ x_{47} x_{11} x_{98} = -1 \]

2Lin gadget

\[ x_{10} = -x_3 \]

\[ y_{61} = -y_{24} \]

\[ x_{16} = -y_5 \]
3Lin
\[ x_1x_3x_5 = 1 \]
\[ x_{10}x_{16}x_3 = -1 \]

... \[ x_{47}x_{11}x_{98} = -1 \]

Final 2Lin inst: union all the gadgets

2Lin gadget
\[ x_{10} = -x_3 \]
\[ y_{61} = -y_{24} \]

... \[ x_{16} = -y_5 \]
3Lin

\[ x_1 x_3 x_5 = 1 \]
\[ x_{10} x_{16} x_3 = -1 \]
\[ x_{47} x_{11} x_{98} = -1 \]

Final 2Lin inst: union all the gadgets

The hope: \( x_i \)’s satisfy 3Lin eq’n \( \Rightarrow \) good assgn to \( y_i \)’s
3Lin

\[ x_1 x_3 x_5 = 1 \]
\[ x_{10} x_{16} x_3 = -1 \]

\( \ldots \)

\[ x_{47} x_{11} x_{98} = -1 \]

Final 2Lin inst: union all the gadgets

The hope:  
- \( x_i \)'s satisfy 3Lin eq'n \( \Rightarrow \) good assgn to \( y_i \)'s  
- \( x_i \)'s don’t \( \Rightarrow \) no good assgn to \( y_i \)'s
Gadgets

Def: A \((c, s)\)-gadget
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)'s satisfy 3Lin eq'n \(\Rightarrow\) an assgn to \(y_i\)'s of value \((1 - c)\)
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)’s satisfy 3Lin eq’n \(\Rightarrow\) an assgn to \(y_i\)’s of value \((1 - c)\)
- \(x_i\)’s don’t \(\Rightarrow\) no assgn to \(y_i\)’s beats value \((1 - s)\)
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)’s satisfy 3Lin eq’n \(\Rightarrow\) an assgn to \(y_i\)’s of value \((1 - c)\)
- \(x_i\)’s don’t \(\Rightarrow\) no assgn to \(y_i\)’s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)'s satisfy 3Lin eq’n ⇒ an assgn to \(y_i\)'s of value \((1 - c)\)
- \(x_i\)'s don’t ⇒ no assgn to \(y_i\)'s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget

\((\varepsilon, \frac{1}{2} - \varepsilon)\)-hardess for 3Lin
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)'s satisfy 3Lin eq'n \(\Rightarrow\) an assgn to \(y_i\)'s of value \((1 - c)\)
- \(x_i\)'s don’t \(\Rightarrow\) no assgn to \(y_i\)'s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget

\((\varepsilon, \frac{1}{2} - \varepsilon)\)-hardness for 3Lin \(\Rightarrow\) - Yes case: \(\frac{1}{4}\)
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)'s satisfy 3Lin eq’n ⇒ an assign to \(y_i\)'s of value \((1 - c)\)
- \(x_i\)'s don’t ⇒ no assign to \(y_i\)'s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget

\((\varepsilon, \frac{1}{2} - \varepsilon)\)-hardess for 3Lin ⇒ - Yes case: \(\frac{1}{4}\)
- No case: \(\frac{1}{2} \times (\frac{1}{4} + \frac{3}{8})\)
Gadgets

Def: A \((c, s)\)-gadget

- \(x_i\)’s satisfy 3Lin eq’n \(\Rightarrow\) an assign to \(y_i\)’s of value \((1 - c)\)
- \(x_i\)’s don’t \(\Rightarrow\) no assign to \(y_i\)’s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget

\((\varepsilon, \frac{1}{2} - \varepsilon)\)-hardness for 3Lin \(\Rightarrow\)

- Yes case: \(\frac{1}{4}\)
- No case: \(\frac{1}{2} \* (\frac{1}{4} + \frac{3}{8}) = \frac{5}{16}\)
Gadgets

Def: A \((c, s)\)-gadget
- \(x_i\)'s satisfy 3Lin eq'n \(\Rightarrow\) an assgn to \(y_i\)'s of value \((1 - c)\)
- \(x_i\)'s don’t \(\Rightarrow\) no assgn to \(y_i\)'s beats value \((1 - s)\)

[TSSW]: there is a 3Lin-to-2Lin \((\frac{1}{4}, \frac{3}{8})\)-gadget

\((\varepsilon, \frac{1}{2} - \varepsilon)\)-hardess for 3Lin \(\Rightarrow\) - Yes case: \(\frac{1}{4}\)
- No case: \(\frac{1}{2} \times \left(\frac{1}{4} + \frac{3}{8}\right)\)
  \[= \frac{5}{16} = \frac{5}{4} \times \frac{1}{4}\]
How do you find gadgets?

Gadgets are just 2Lin instances, so can just monkey around with small instances.
How do you find gadgets?

Gadgets are just 2Lin instances, so can just monkey around with small instances.

More principled: [TSSW] show that the optimal gadget can be found via linear program!
How do you find gadgets?

Gadgets are just 2Lin instances, so can just monkey around with small instances.

More principled: [TSSW] show that the optimal gadget can be found via linear program!

- **key insight**: one can bound # of auxiliary variables
- can certify optimality via dual LP.
How do you find gadgets?

Gadgets are just 2Lin instances, so can just monkey around with small instances.

More principled: 

- key insight: one can bound the number of auxiliary variables 
- can certify optimality via dual linear program
Our strategy

Old reduction from Håstad’s 3Lin hardness result.
Our strategy

Old reduction from Håstad’s 3Lin hardness result.

We now have **better** starting points.
Our strategy

Old reduction from Håstad’s 3Lin hardness result.
We now have better starting points.
[Chan]: Given 3Lin instance $I$, NP-hard to distinguish
Our strategy

Old reduction from Håstad’s 3Lin hardness result.

We now have better starting points.

[Chan]: Given 3Lin instance $I$, NP-hard to distinguish

- **Yes**: $\text{val}(I) = (1 - \varepsilon)$
Our strategy

Old reduction from Håstad’s 3Lin hardness result.

We now have better starting points.

[Chan]: Given 3Lin instance $I$, NP-hard to distinguish

- **Yes**: $\text{val}(I) = (1 - \varepsilon)$
- **No**: no matter what assignment, the variables “appear” to be uniformly random
Our strategy

Old reduction from Håstad’s 3Lin hardness result. We now have better starting points.

[Chan]: Given 3Lin instance $I$, \textbf{NP}-hard to distinguish

- **Yes**: $\text{val}(I) = (1 - \varepsilon)$
- **No**: no matter what assignment, the variables “appear” to be uniformly random

Stronger **No** condition. Might help out the gadget.
New gadgets

Def: A \((c, s)\)-Chan-gadget
New gadgets

Def: A \((c,s)\)-Chan-gadget
- \(x_i\)'s satisfy \(3\text{Lin eq'n} \Rightarrow \) an assgn to \(y_i\)'s of value \((1 - c)\)
New gadgets

Def: A \( (c,s) \)-Chan-gadget

- \( x_i \)'s satisfy 3Lin eq’n \( \Rightarrow \) an assgn to \( y_i \)'s of value \((1 - c)\)
- \( x_i \)'s random \( \Rightarrow \) on avg., expected best value \( \leq (1 - s) \)
New gadgets

Def: A \((c,s)\)-Chan-gadget
- \(x_i\)'s satisfy 3Lin eq'n \(\Rightarrow\) an assign to \(y_i\)'s of value \((1 - c)\)
- \(x_i\)'s random \(\Rightarrow\) on avg., expected best value \(\leq (1 - s)\)

Upside: can still solve using an LP
New gadgets

Def: A \((c, s)\)-Chan-gadget

- \(x_i\)’s satisfy 3Lin eq’n \(\Rightarrow\) an assgn to \(y_i\)’s of value \((1 - c)\)
- \(x_i\)’s random \(\Rightarrow\) on avg., expected best value \(\leq (1 - s)\)

Upside: can still solve using an LP
Downside: best 3Lin-to-2Lin gadget no better than in ’97!
Our strategy (revised)

Old reduction from Håstad’s 3Lin hardness result.

We now have better starting points.

[Chan]: Given 3Lin instance $I$, NP-hard to distinguish

- **Yes**: $\text{val}(I) = (1 - \varepsilon)$
- **No**: no matter what assignment, the variables “appear” to be uniformly random
Our strategy (revised)

Old reduction from Håstad’s 3Lin hardness result.

We now have better starting points.

[Chan]: Given “balanced pairwise independent subgroup predicate” instance $I$, NP-hard to distinguish

- Yes: $\text{val}(I) = (1 - \varepsilon)$
- No: no matter what assignment, the variables “appear” to be uniformly random
Our strategy (revised)

Old reduction from Håstad’s 3Lin hardness result.

We now have **better** starting points.

[Chan]: Given “balanced pairwise independent subgroup predicate” instance $I$, NP-hard to distinguish

- **Yes**: $\text{val}(I) = (1 - \varepsilon)$
- **No**: no matter what assignment, the variables “appear” to be uniformly random

We instantiate with $\text{BPISP} = \text{Had}_k$, specifically $k = 3$. 
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff } \]
- \( x_i \)'s allowed to be arbitrary
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]

- \( x_i \)'s allowed to be arbitrary
- \( x_{a,b} = x_a \cdot x_b \)
One of Chan’s problems

$$\text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1$$ iff

- $x_i$’s allowed to be arbitrary
- $x_{a,b} = x_a \cdot x_b$
- $x_{1,2,3} = x_1 \cdot x_2 \cdot x_3$
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]
- \( x_i \)'s allowed to be arbitrary
- \( x_{a,b} = x_a \cdot x_b \)
- \( x_{1,2,3} = x_1 \cdot x_2 \cdot x_3 = x_{1,2} \cdot x_3 \)
One of Chan’s problems

\[
\text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff }
\]
- \( x_i \)'s allowed to be arbitrary
- \( x_{a,b} = x_a \cdot x_b \)
- \( x_{1,2,3} = x_1 \cdot x_2 \cdot x_3 = x_{1,2} \cdot x_3 = x_{1,3} \cdot x_2 = x_1 \cdot x_{2,3} \)
One of Chan’s problems

$$\text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff}$$

- $x_i$’s allowed to be arbitrary
- $x_a \cdot x_b \cdot x_{a,b} = 1$
- $x_{1,2,3} = x_1 \cdot x_2 \cdot x_3 = x_{1,2} \cdot x_3 = x_{1,3} \cdot x_2 = x_1 \cdot x_{2,3}$
One of Chan’s problems

$\text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1$ iff

- $x_i$’s allowed to be arbitrary
- $x_a \cdot x_b \cdot x_{a,b} = 1$
- $x_a \cdot x_{\{1,2,3\}\setminus a} \cdot x_{1,2,3} = 1$ for all $a \in \{1,2,3\}$
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]
- \( x_i \)'s allowed to be arbitrary
- \( x_a \cdot x_b \cdot x_{a,b} = 1 \)
- \( x_a \cdot x_{\{1,2,3\}\setminus a} \cdot x_{1,2,3} = 1 \) for all \( a \in \{1,2,3\} \)

Contains many simultaneous 3Lin tests. Difficult to satisfy!
One of Chan’s problems

\( \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \iff \)

- \( x_i \)’s allowed to be arbitrary
- \( x_a \cdot x_b \cdot x_{a,b} = 1 \)
- \( x_a \cdot x_{\{1,2,3\}\setminus a} \cdot x_{1,2,3} = 1 \) for all \( a \in \{1,2,3\} \)

Contains many simultaneous 3Lin tests. Difficult to satisfy!

(not too hard to generalize to \( \text{Had}_k \))
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]
- \(x_i\)'s allowed to be arbitrary
- \(x_a \cdot x_b \cdot x_{a,b} = 1\)
- \(x_a \cdot x_{\{1,2,3\}\setminus a} \cdot x_{1,2,3} = 1\) for all \(a \in \{1,2,3\}\)

Contains many simultaneous 3Lin tests. Difficult to satisfy!

(not too hard to generalize to \(\text{Had}_k\))

[U]: there is a \((\frac{1}{8}, \frac{11}{64})\)-Chan-gadget from \(\text{Had}_3\) to \(2\text{Lin}\)
One of Chan’s problems

\[ \text{Had}_3(x_1, x_2, x_3, x_{1,2}, x_{1,3}, x_{2,3}, x_{1,2,3}) = 1 \text{ iff} \]

- \( x_i \)'s allowed to be arbitrary
- \( x_a \cdot x_b \cdot x_{a,b} = 1 \)
- \( x_a \cdot x_{\{1,2,3\}\setminus a} \cdot x_{1,2,3} = 1 \) for all \( a \in \{1,2,3\} \)

Contains many simultaneous 3Lin tests. Difficult to satisfy!

(not too hard to generalize to \( \text{Had}_k \))

[Us]: there is a \( (\frac{1}{8}, \frac{11}{8} \cdot \frac{1}{8}) \)-Chan-gadget from \( \text{Had}_3 \) to \( 2\text{Lin} \)
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables

(so no pictures like these)
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables

LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments to these variables: too large to be feasible!
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments
to these variables: too large to be feasible!
So…….
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments to these variables: too large to be feasible!
So……
- lots of work by hand
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments to these variables: too large to be feasible!

So……
- lots of work by hand
- lots of computer simulation (/brute force searching)
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables

LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments to these variables: too large to be feasible!

So……

- lots of work by hand
- lots of computer simulation (/brute force searching)
- lots more work by hand
Solving the LP

[TSSW]: optimal gadget only needs $2^7 = 128$ variables
LP has to consider all possible $2^{128} = 3 \times 10^{38}$ assignments to these variables: too large to be feasible!
So…….
- lots of work by hand
- lots of computer simulation (/brute force searching)
- lots more work by hand

but (spoiler alert) it all works out in the end.
Full statement of our results

- \((\varepsilon, \frac{11}{8} \varepsilon)\)-approx \(\text{NP}\)-hard (for 2Lin and Max-Cut)
Full statement of our results

- \((\varepsilon, \frac{11}{8} \varepsilon)\)-approx \textsf{NP}-hard (for 2Lin and Max-Cut)

- \((\frac{1}{8}, \frac{11}{8} \cdot \frac{1}{8})\)-Chan-gadget from \textsf{Had}_3 to 2Lin
Full statement of our results

- \((\varepsilon, \frac{11}{8} \varepsilon)\)-approx \text{NP}-hard (for 2Lin and Max-Cut)

- \((\frac{1}{8}, \frac{11}{8} \cdot \frac{1}{8})\)-Chan-gadget from \text{Had}_3 to 2Lin

- prove optimality of this gadget via dual solution
Full statement of our results

- \((\varepsilon, \frac{11}{8} \varepsilon)\)-approx \(\text{NP}\)-hard (for 2Lin and Max-Cut)

- \((\frac{1}{8}, \frac{11}{8} \cdot \frac{1}{8})\)-Chan-gadget from \(\text{Had}_3\) to 2Lin
  - prove optimality of this gadget via dual solution

- can’t beat \((\varepsilon, 2.54 \cdot \varepsilon)\)-hardness via a Chan gadget starting from *any* BPISP predicate
Open problems

We give an optimal gadget reduction from $\text{Had}_3$ to $2\text{Lin}$. 
Open problems

We give an optimal gadget reduction from $\text{Had}_3$ to 2Lin.

We give a gadget from $\text{Had}_k$ to 2Lin, along with a Game Show Conjecture which would imply $(\varepsilon, 1.5 \cdot \varepsilon)$-hardness.
Open problems

We give an optimal gadget reduction from Had$_3$ to 2Lin.

We give a gadget from Had$_k$ to 2Lin, along with a Game Show Conjecture which would imply ($\varepsilon$, $1.5 \cdot \varepsilon$)-hardness.

We couldn’t say anything about the normal hardness ratio. Maybe you can?
Thanks!