Lecture 10: Gadget Reductions and SOS

Lecture Outline

- Part I: Hardness Reductions for SOS
- Part II: Inapproximability of Independent Set
- Part III: $\frac{16}{17}$-Inapproximability of MAX CUT

Part I: Hardness Reductions for SOS

Promise Problems Over $\{0,1\}^n$

- Definition: Let A be a class of maximization problems over $\{0,1\}^n$.
- We can view A as a class of multilinear polynomials. With this view, given an instance $a \in A$, define $v(a) = \max\{\mathrm{a(x)}: \mathrm{x} \in \{0,1\}^n\}$
- Let $A(c_1, c_2)$ to be the problem of distinguishing whether $\mathrm{v(a)} \geq c_2$ or $\mathrm{v(a)} \leq c_1$ for some $a \in A$.
- This is a promise problem as we are given a promise that either $\mathrm{v(a)} \geq c_2$ or $\mathrm{v(a)} \leq c_1$. If $c_1 < v(a) < c_2$ then we can say anything.

Standard Reductions

- Standard reduction: To show that $A(c_1, c_2)$ can be reduced to $B(c_1', c_2')$, we must give a reduction $R: A \rightarrow B$ which satisfies:
  1. Soundness: If $v(a) \leq c_1$ then $v\big(R(a)\big) \leq c_1'$
  2. Completeness: If $v(a) \geq c_2$ then $v\big(R(a)\big) \geq c_2'$

SOS Reductions

- How can we show that if problem A is hard for SOS than so is problem B?
- One way (done by Tulsiani [Tul09]): Show that valid pseudo-expectation values for A can be transformed into valid pseudo-expectation values for B.
- Simpler way: Look at the dual, Positivstellensatz/SOS proofs!

Subtle Point About SOS Proofs

- Setup: We have constraints $s_1(x_1, \ldots, x_n) = 0$, $s_2(x_1, \ldots, x_n) = 0$, etc. and want to prove that $h < c$.
- There are two subtly different variants of SOS proofs that $h < c$
- An SOS proof that $h < c$ is an equality $h = c' + \sum_i f_i s_i - \sum_j g_j^2$ where $c' < c$.
- An SOS proof that $h \geq c$ is infeasible is an equality $-1 = \sum_i f_i s_i + f(h - c - z^2) + \sum_j g_j^2$ (we effectively add the constraint $h = c + z^2$)

Subtle Point About SOS Proofs

- An SOS proof that $h < c$ is an equality $h = c' + \sum_i f_i s_i - \sum_j g_j^2$ where $c' < c$.
- An SOS proof that $h \geq c$ is infeasible is an equality $-1 = \sum_i f_i s_i + f(h - c - z^2) + \sum_j g_j^2$
- It's harder to find an SOS proof that $h < c$ than an SOS proof that $h \geq c$ is infeasible.
- Thus, we show a stronger lower bound if we show that there isn't even an SOS proof that $h \geq c$ is infeasible.

SOS Reductions

- SOS variant of $A(c_1, c_2)$: Given an instance $a \in A$ with value $v(a) \leq c_1$, give an SOS proof that $v(a) \geq c_2$ is infeasible.
- Call this variant $A_{SOS}(c_1, c_2)$.

SOS Reductions

- SOS reduction: To show that $A_{SOS}(c_1, c_2)$ can be reduced to $B_{SOS}(c_1', c_2')$, we must give a reduction $R: A \to B$ which satisfies:
  1. Soundness: If $v(a) \leq c_1$ then $v\big(R(a)\big) \leq c_1'$
  2. Completeness: If there is an SOS proof of degree d' that $v\big(R(a)\big) \geq c_2'$ is infeasible then there is an SOS proof of degree d that $v(a) \geq c_2$ is infeasible

Part II: Inapproximability of Independent Set

Inapproximability of Independent Set

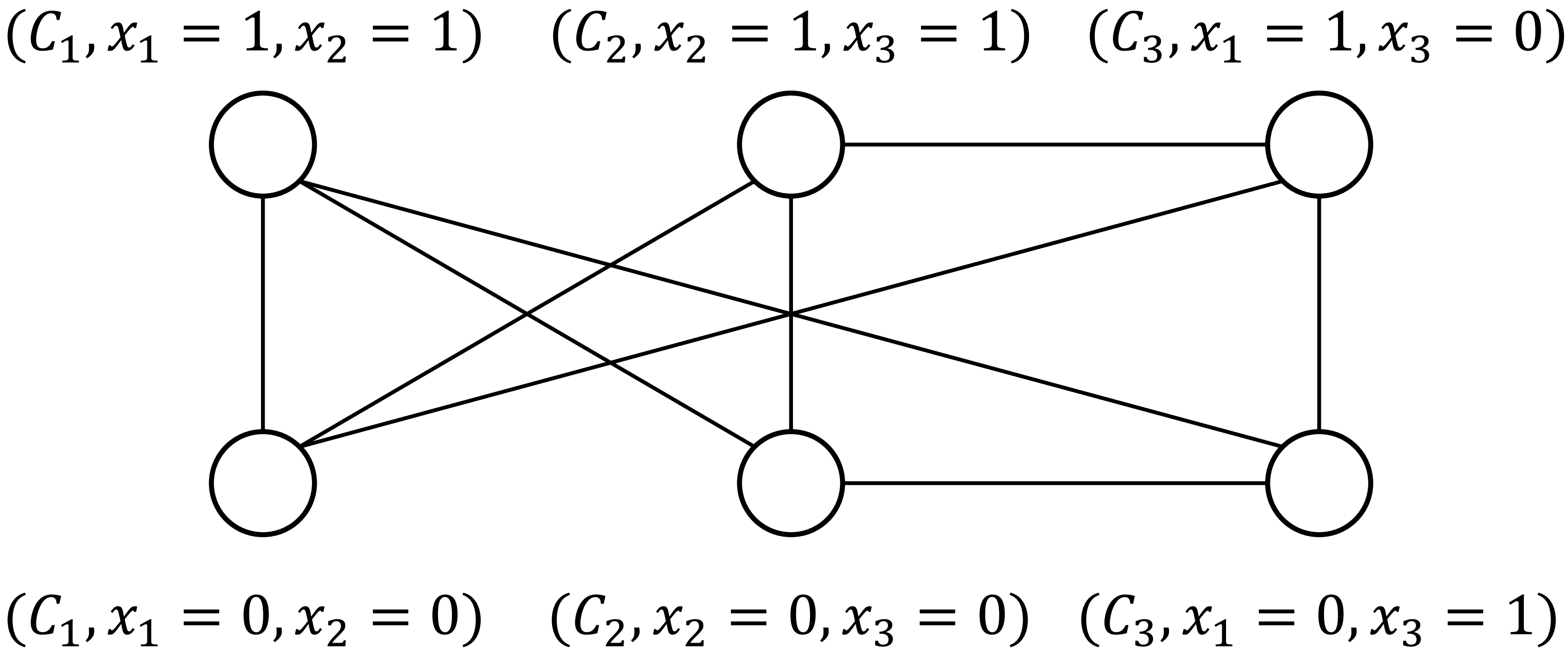
- Theorem [Hås99]: It is NP-hard to approximate independent set within a factor of $N^{-(1-o(1))}$
- Here we follow the presentation in Advanced Approximation Algorithms Lecture 25 taught by Ryan O'Donnell $[\text{AAALecture25}]$.

FGLSS Graph

- Given a CSP, the FGLSS graph [FGLSS96] $G_{\boldsymbol{\Phi}}$ is as follows:
- Have a vertex for each pair (C, x) where C is a constraint and x is an assignment of values to the variables in C which satisfies C
- Have an edge between two vertices (C_1, x_1) and (C_2, x_2) if x_1, x_2 disagree on the value of some variable.

FGLSS Graph Example

- Constraints $C_1: x_1 = x_2, C_2: x_2 = x_3, C_3: x_1 \neq x_3$



Independent Set on FGLSS Graph

- Proposition: The size of the largest independent set in the FGLSS graph $G_{\boldsymbol{\Phi}}$ is equal to the maximum number of clauses which can be satisfied at the same time.
- Proof: Given an x, we can take all vertices (C, x) in $G_{\boldsymbol{\Phi}}$ which match x. This is an independent set with one vertex for each satisfied clause.
- Conversely, given an independent set I in $G_{\boldsymbol{\Phi}}$, we can find a corresponding x by gluing the partial assignments together. No two vertices in I can have the same C, so $|I| \leq$ # of satisfied clauses

Capturing Argument with SOS

- How can we capture this argument with SOS?
- Equations we are trying to refute for independent set on $G_{\boldsymbol{\Phi}}$:
  1. $\forall (C, x)$: $v_{(C,x)}^2 = v_{(C,x)}$
  2. $v_{(C_1,x_1)} v_{(C_2,x_2)} = 0$ whenever $(C_1, x_1), (C_2, x_2)$ disagree on the value of some x_i.
  3. $\sum_{(C,x)} v_{(C,x)} \geq k$
- Given an SOS proof of infeasibility for these equations, want an SOS proof that it is impossible to satisfy k or more clauses.

Capturing Argument with SOS

- Key idea: The value of each variable $v_{(C,x)}$ is determined by the reduction, simply make this substitution!
- Definitions: Define $C(x)$ to be the multilinear polynomial which is 1 if C is satisfied and 0 otherwise. Take

$$v_{(C,x)} = \prod_{i:\{x\ sets\ x_i=1\}} x_i \prod_{i:\{x\ sets\ x_i=0\}} (1 - x_i)$$

- Proposition: $C(x) = \sum_{x:(C,x) \in V(G_{\boldsymbol{\Phi}})} v(C,x)$
- Corollary: $\sum_C C(x) = \sum_{(C,x)} v_{(C,x)}$

Boosting the Gap

- By itself, this argument only gives a constant gap.
- How can we boost the gap?
- If we don’t care too much about the number of clauses or how many variables each clause contains, we can use serial repetition.

Serial Repetition

- Serial repetition: Given m clauses, each with a variables, take the new clauses to be t-tuples of clauses (which are satisfied if and only if all the individual clauses are satisfied)
- This gives m^t clauses which have $\leq at$ variables.
- If at least $\mathrm{k} = sm$ of the original clauses could be satisfied at the same time, at least $k^t = s^t m^t$ of the new clauses can be satisfied.
- Note: Called serial repetition to distinguish it from parallel repetition.

Serial Repetition and SOS

- This argument is easily captured by SOS, as it boils down to the following:
- If $\sum_C C(x) \geq k \geq 0$ then $(\sum_C C(x))^t \geq k^t$

Sparsification

- How can we reduce the number of clauses?
- Pick a small subset of clauses at random!
- If at most s' fraction of the clauses were satisfiable, then for each $x \in \{0,1\}^n$, w.h.p. roughly s' fraction of the subset will be satisfied.
- Only have to take a union bound over 2^n possibilities

Lower Bound High Level Picture

1. Start with a CSP (actually, the CSP must also have a low number of satisfying assignments)
2. Apply serial repetition to amplify the gap
3. Use sparsification to reduce the number of clauses
4. Apply the FGLSS graph reduction

In-class Challenge

- In-class challenge: How does SOS capture the sparsification argument?
- For this, let's consider a simplified example. Let's say that the original statement we want to refute is $\sum_{i=1}^{m} C_i = m$. Sparsification corresponds to statements of the form $\sum_{i \in S} C_i = |S|$
- You may consider the case where we have SOS proofs that $\sum_{i \in S} C_i < |S|$ w.h.p. (which is stronger than having proofs that $\sum_{i \in S} C_i = |S|$ is infeasible)

In-class Challenge Answer #1

- In-class exercise: How does SOS capture the sparsification argument?
- One answer: If we have SOS proofs that $\sum_{i \in S} C_i < |S|$ for almost all subsets S, we can take a linear combination of these proofs to obtain an SOS proof that $\sum_{i=1}^{m} C_i < m$

In-class Challenge Answer #2

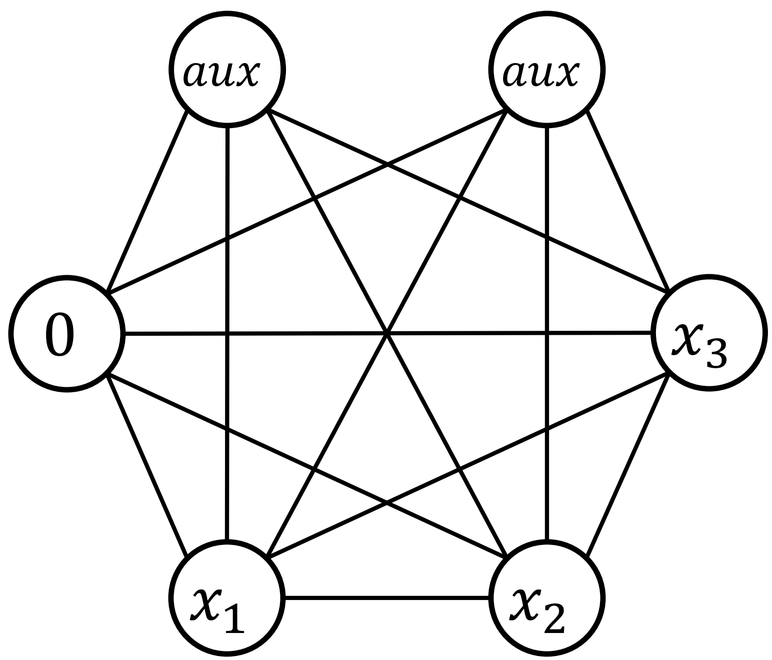
- In-class exercise: How does SOS capture the sparsification argument?
- Second answer (which gives the optimal lower bound): Our pseudo-expectation values for CSPs not only satisfy the constraint that $\sum_{i=1}^{m} C_i(x) = m$ (where m is the number of clauses), they in fact satisfy the constraint that $\sum_{i \in S} C_i\,(x) = |S|$ for every subset S of clauses. Thus, there cannot be an SOS proof that $\sum_{i \in S} C_i\,(x) = |S|$ is infeasible for any S.

Part II: $\frac{16}{17}$-Inapproximability of MAX CUT

Parity Checking Gadgets for MAX CUT

- Idea: find graphs PC_0 and PC_1 such that the following is true:
  1. PC_0 and PC_1 have special vertices labelled $x_1, x_2, x_3, 0$
  2. More edges can be cut in PC_0 if $x_1 + x_2 + x_3 = 0$ mod 2 than if $x_1 + x_2 + x_3 = 1$ mod 2
  3. More edges can be cut in PC_1 if $x_1 + x_2 + x_3 = 1$ mod 2 than if $x_1 + x_2 + x_3 = 0$ mod 2
- With these gadgets, we can transform our gap for 3-XOR into a gap for MAX CUT.

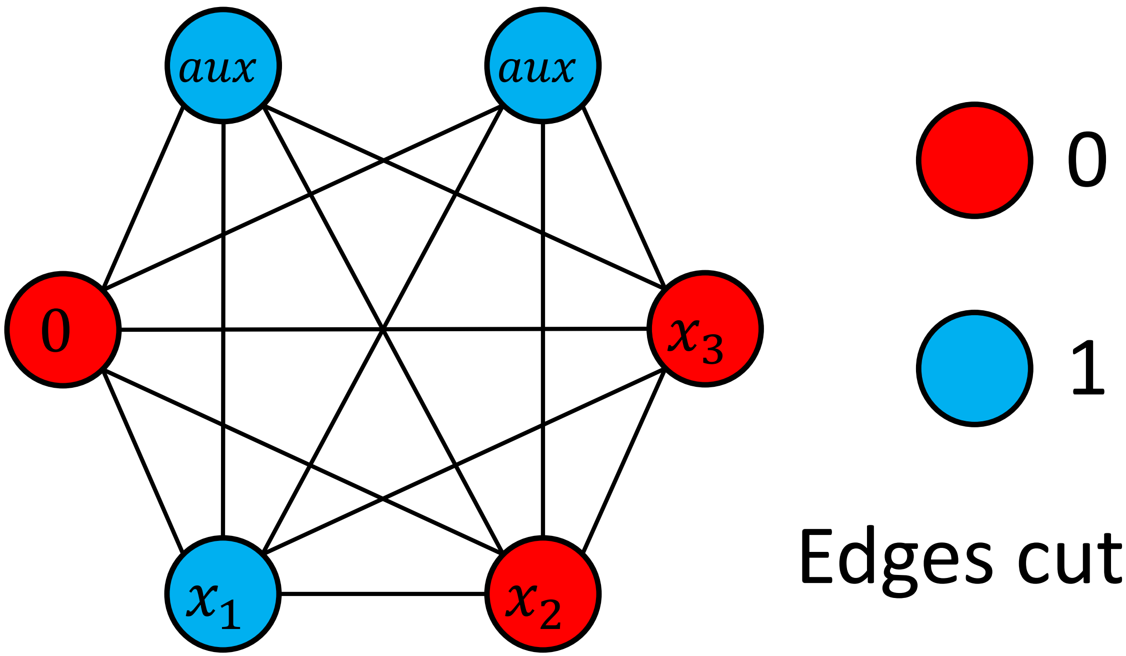
PC_1 Gadget



PC_1 Gadget

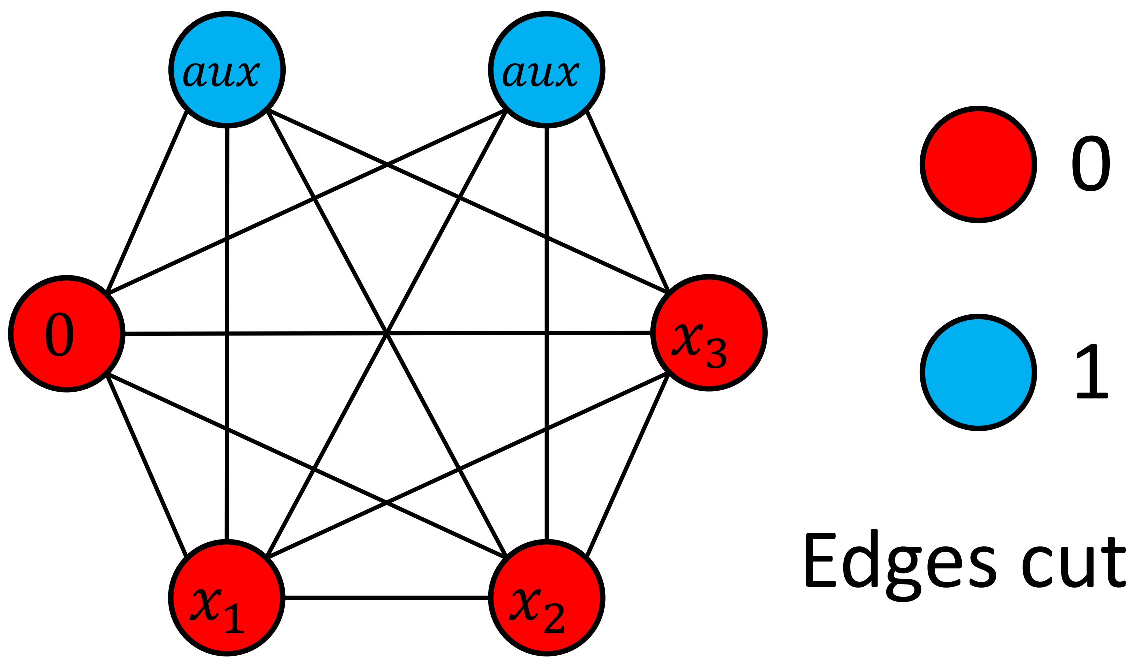
- Claim: If $x_1 + x_2 + x_3 = 1$ mod 2 then 9 of the 14 edges can be cut. Otherwise, exactly 8 of the 14 edges can be cut.
- Proof: To cut 9 edges, we need a cut $(S, \bar{S})$ where $|S| = |\bar{S}| = 3$ and both aux vertices are on the same side. This is possible if and only if $x_1 + x_2 + x_3 = 1$ mod 2.
- If $x_1 + x_2 + x_3 = 0$ mod 2, we can always find a cut $(S, \bar{S})$ where one side has 2 vertices and both aux vertices are on the same side

PC_1 Gadget Examples



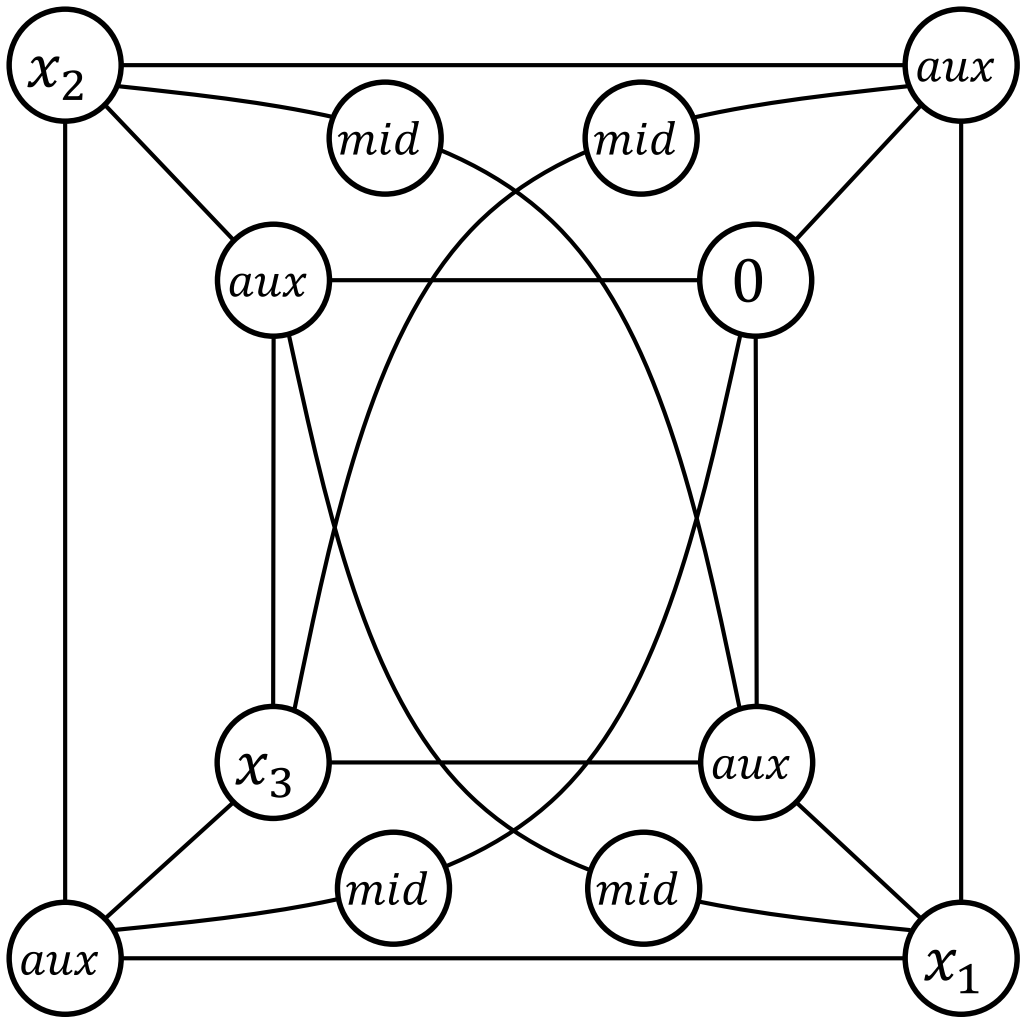
Edges cut: $\frac{9}{14}$

PC_1 Gadget Examples



Edges cut: $\frac{8}{14}$

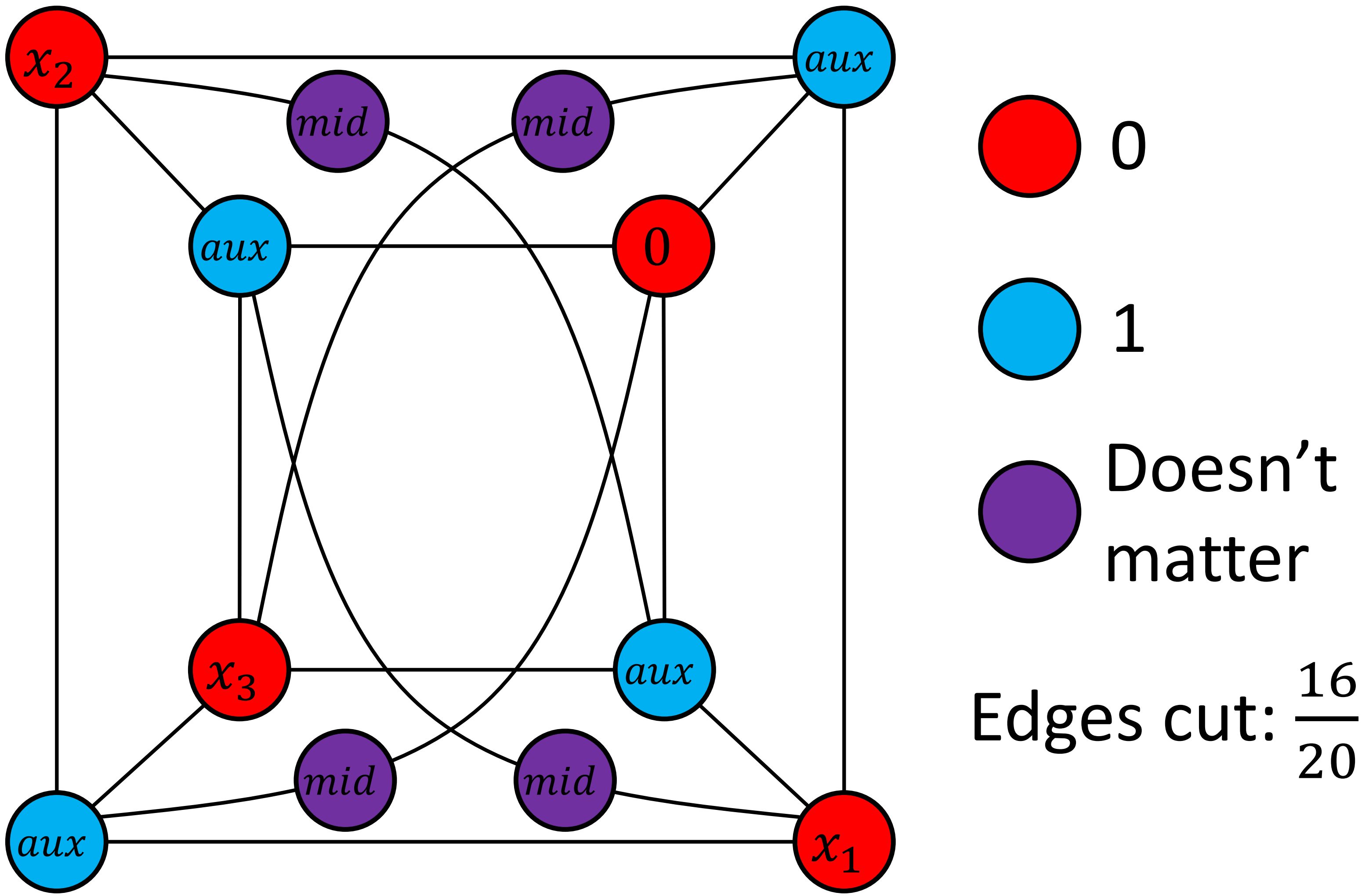
PC_0 Gadget



PC_0 Gadget

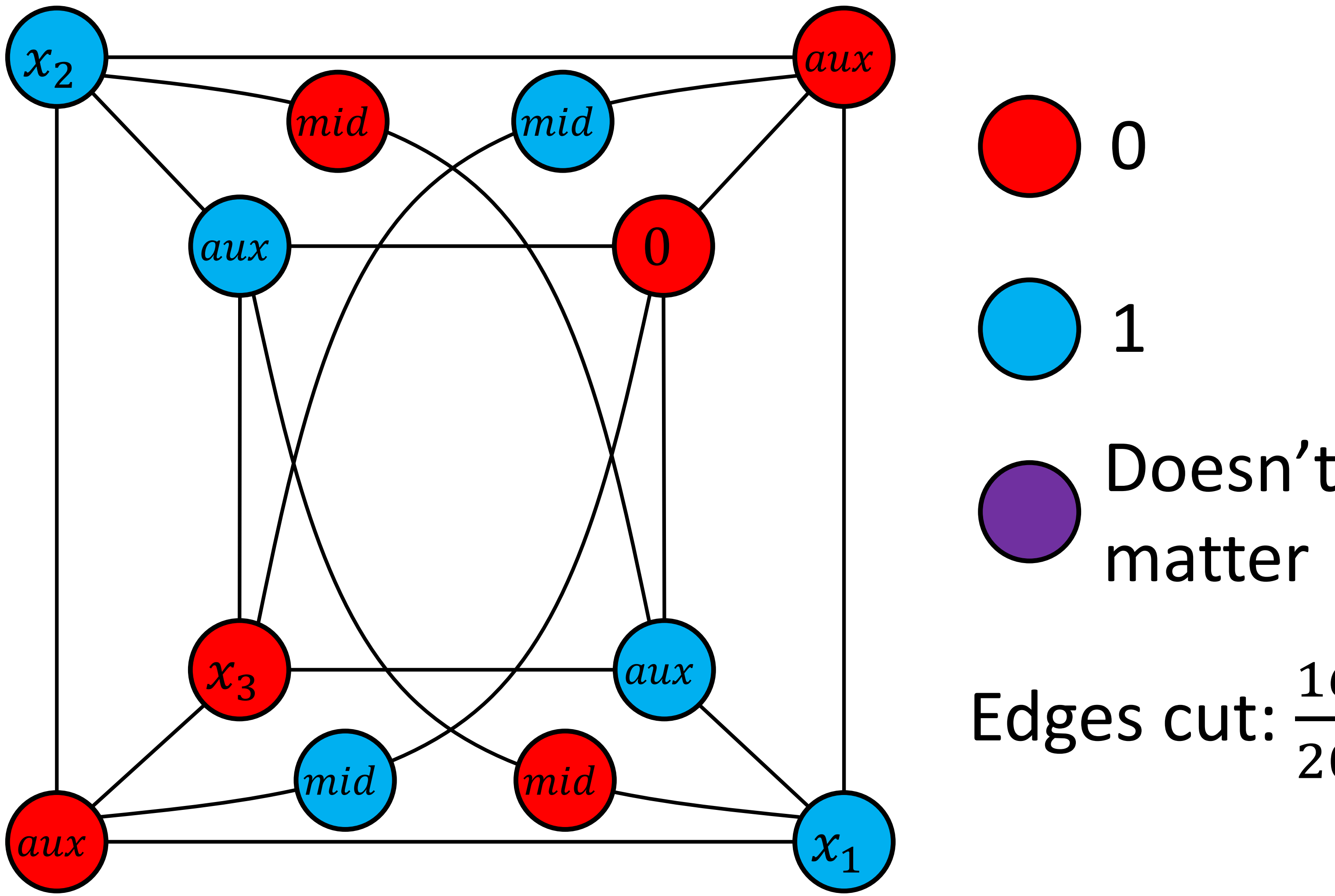
- Claim: If $x_1 + x_2 + x_3 = 0$ mod 2 then 16 of the 20 edges can be cut. Otherwise, exactly 14 of the 20 edges can be cut.
- Proof idea: Note that it is always optimal for the aux vertices to be on the opposite side from the majority of their non-mid neighbors. Now for each of the mid vertices, if their two neighbors are on the same side we place the mid vertex on the opposite side. Otherwise, we make an arbitrary choice.

PC_0 Gadget Examples



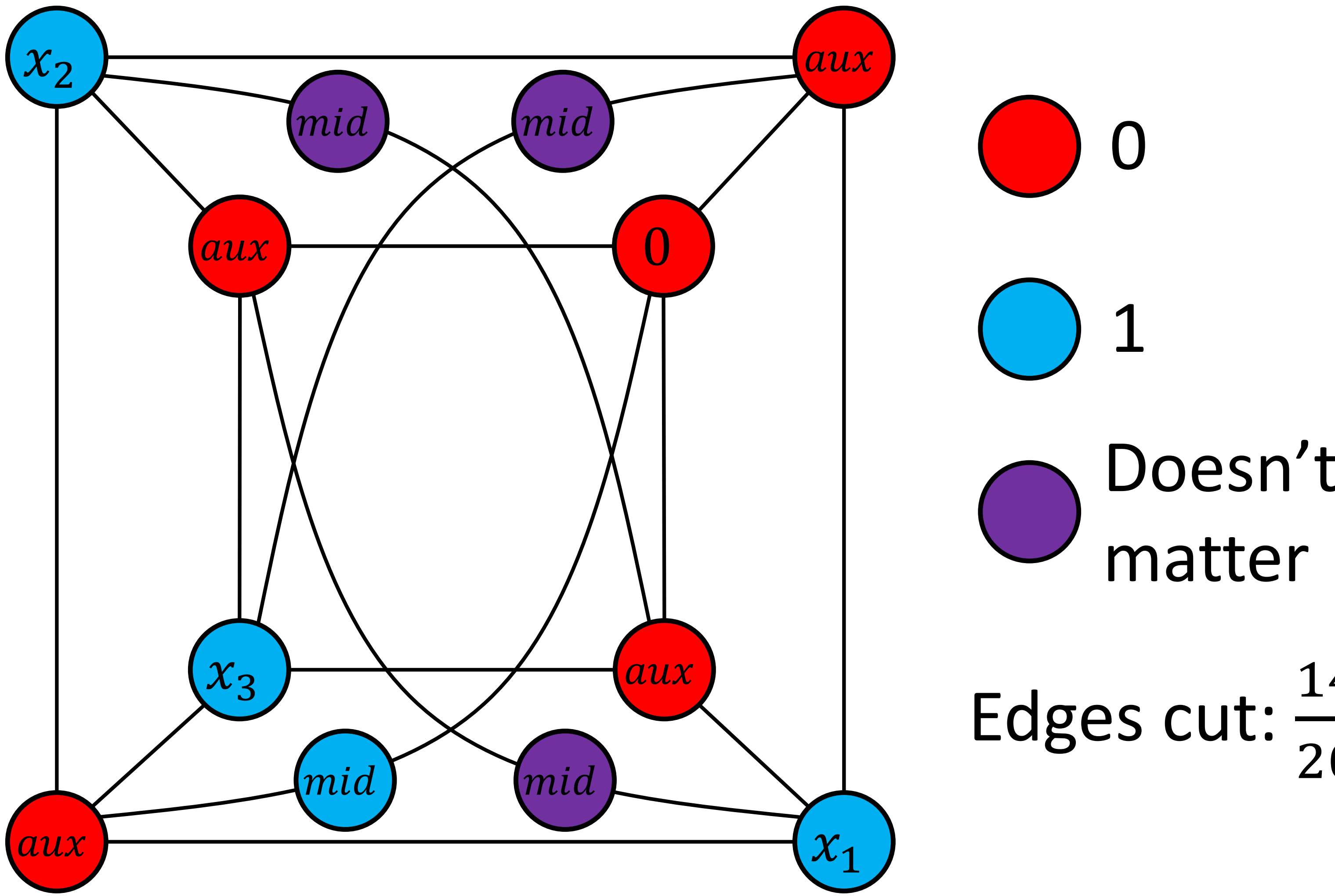
Edges cut: $\frac{16}{20}$

PC_0 Gadget Examples



Edges cut: $\frac{16}{20}$

PC_0 Gadget Examples



Edges cut: $\frac{14}{20}$

Gap for MAX CUT

- Let m_0 be the number of 0 constraints and let m_1 be the number of 1 constraints. Without loss of generality, $m_0 \geq m_1$.
- For each 0 constraint, take one PC_0 gadget. For each 1 constraint, take two PC_1 gadgets.
- Key idea: Every failed constraint gives a penalty of two edges.

Gap for MAX CUT

- If almost all clauses are satisfiable, the max cut has size $\approx 16m_0 + 18m_1 \leq 17(m_0 + m_1)$
- If not much more than half the clauses are satisfiable, the max cut has size $\approx 16m_0 + 18m_1 - (m_0 + m_1)$
- Gap is $\frac{16m_0+18m_1-(m_0+m_1)}{16m_0+18m_1} \leq \frac{16}{17}$

SOS MAX CUT Reduction

- Need to show that an SOS proof that the maximum cut cannot have value more than $16m_0 + 18m_1 - 2x$ can be transformed into an SOS proof that at least x constraints must be unsatisfied.
- Key idea: Similar to before, substitute polynomials for the MAX CUT variables based on the reduction

SOS Proof for Gadgets

- Lemma: If $x_1, x_2, x_3 \in \{0,1\}$ then $h(x_1, x_2, x_3) = 1 - x_1x_2 - x_1x_3 - x_2x_3 + 2x_1x_2x_3$ is equal to 1 if $x_1 + x_2 + x_3 \leq 1$ and is equal to 0 if $x_1 + x_2 + x_3 \geq 2$

SOS Proof for Gadgets

- If we give value $h(x_1, x_2, x_3)$ to the auxiliary vertices in CP_1, the number of edges cut will be 9 if $x_1 + x_2 + x_3 = 1$ mod 2 and 8 if $x_1 + x_2 + x_3 = 0$ mod 2.
- We can make similar substitutions for CP_0 so that the number of edges cut will be 16 if $x_1 + x_2 + x_3 = 0$ mod 2 and 14 if $x_1 + x_2 + x_3 = 0$ mod 2.
- These facts are captured by constant degree SOS.

Obtaining an SOS Proof for 3-XOR

- If we apply these substitutions to an SOS proof that the maximum cut cannot have value more than $16m_0 + 18m_1 - 2x$, we obtain an SOS proof that at least x constraints must be unsatisfied.

References

- [BGS98] M. Bellare, O. Goldreich, M. Sudan. Free Bits, PCPs, and Nonapproximability---Towards Tight Results. SIAM J. Comput. 27 (3), p. 804–915, 1998
- [FGLSS96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, M. Szegedy. Interactive proofs and the hardness of approximating cliques. JACM 34 (2), p. 268-292, 1996.
- [Hås99] J. Håstad. Clique is hard to approximate within n $1-\varepsilon$. Acta Mathematica, 182:105–142, 1999.
- [Hås01] Johan Håstad. Some optimal inapproximability results. JACM 48 (4), p. 798–859, 2001.
- [AAALecture25] R. O'Donnell. Advanced Approximation Algorithms Lecture 25. April 15, 2008.
- [TSSW00] L. Trevisan, G. Sorkin, M. Sudan, D. Williamson. Gadgets, approximation and linear programming. SIAM J. Comput. 29, p. 2074–2097, 2000
- [Tul09] M. Tulsiani. CSP gaps and reductions in the lasserre hierarchy. In STOC, pages 303–312, 2009.

Appendix: Parameter Calculations for Independent Set

Parameter Calculations

- Starting parameters:
  - m clauses
  - Each clause has z satisfying assignments
  - Trying to distinguish between the case when at most s fraction of the clauses can be satisfied and the case when almost all clauses can be satisfied (gap is s)

Parameter Calculations

- After applying parallel repetition t times
  - m^t clauses
  - Each clause has z^t satisfying assignments
  - Trying to distinguish between the case when at most s^t fraction of the clauses can be satisfied and the case when almost all clauses can be satisfied (gap is s^t)

Sparsification Parameters

- If at most s^t proportion of clauses can be satisfied, then if we pick s^{-t} clauses at random, for any $x \in \{0,1\}^n$, the number of clauses satisfied $\approx$ Poisson distribution with expected value ≤ 1
- Poisson distribution with expected value 1:
$P[k] = \frac{1}{e(k!)}$
- $P[n] \ll 2^{-n}$, so we can assume $\leq n$ clauses are satisfied.
- Note: O'Donnell has m, I'm not sure why…

After Sparsification

- After sparsification:
  - s^{-t} clauses
  - Each clause has z^t satisfying assignments
  - Trying to distinguish between the case when at most m clauses can be satisfied and the case when almost all clauses can be satisfied (gap is ms^t)

FGLSS Graph

- FGLSS Graph $G_{\boldsymbol{\Phi}}$:
  - $s^{-t}z^{t}$ vertices
  - Largest independent set has size $\leq n$ if at most sm of the original clauses were satisfiable. Largest independent set has size almost s^{-t} if almost all the original clauses were satisfiable.
- To get our gap, we need a predicate with $\log(z) \ll -\log(s)$ (then we can take $t = O(\log n)$)

Finding a Predicate

- To get our gap, we need a predicate with $\log(z) \ll -\log(s)$
- This can be done, as shown by the following theorem:
- Theorem [Samorodnitsky, Trevisan 00]: For any constant k, there exists a predicate on $q := O(k^2)$ bits with $w = 2^k$ satisfying assignments for which we have $1 - \epsilon$ versus $\frac{2^k}{2^q} + \epsilon$ hardness for all $\epsilon > 0$.