BILL AND NATHAN, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Approx Classes and Reductions

Approx Classes

Convention

There are two kinds of problems:

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

We will define terms only for MAX problems.

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

We will define terms only for MAX problems.
Analogous notions can be defined for MIN problems.

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

We will define terms only for MAX problems.
Analogous notions can be defined for MIN problems.

Alg means **Poly Time Algorithm**.

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

We will define terms only for MAX problems.
Analogous notions can be defined for MIN problems.

Alg means **Poly Time Algorithm**.

Alg will find actual solution (e.g., an assignment that satisfies many clauses).

Convention

There are two kinds of problems:

1. MAX probs: e.g., MAX3SAT, Cliq (CLIQ), Ind Set (IS).
2. MIN probs: e.g., Vertex Cover (VC), Dominating Set (DM).

We will define terms only for MAX problems.
Analogous notions can be defined for MIN problems.

Alg means **Poly Time Algorithm**.

Alg will find actual solution (e.g., an assignment that satisfies many clauses).

We Assume $\mathrm{P} \neq \mathrm{NP}$.

Max Alg: Benefit Notation

Let A be a max problem (e.g., MAX3SAT).

Max Alg: Benefit Notation

Let A be a max problem (e.g., MAX3SAT).

Let ALG be an alg that finds solutions for A (e.g., assignments).

Max Alg: Benefit Notation

Let A be a max problem (e.g., MAX3SAT).

Let ALG be an alg that finds solutions for A (e.g., assignments).

$\mathbf{benefit(ALG}(x))$ is how good $\mathrm{ALG}(x)$ is (e.g., numb clauses satisfied).

Max Alg: Benefit Notation

Let A be a max problem (e.g., MAX3SAT).

Let ALG be an alg that finds solutions for A (e.g., assignments).

$\mathbf{benefit(ALG(}\boldsymbol{x}\mathbf{))}$ is how good $\mathrm{ALG}(x)$ is (e.g., numb clauses satisfied).

If we dealt with min problems we would use **cost**.

Def of Approx

Def ALG an alg and $c \geq 1$ is a constant A is a max-problem.
ALG is ***c*-app-alg for *A*** if,

$$\mathrm{benefit}(\mathrm{ALG}(x)) \geq \frac{1}{c} \times \mathrm{benefit}(\mathrm{OPT}(x)).$$

Poly Time Approx Scheme (PTAS)

Def Let A be a MAX problem.
A **Poly time Approx Scheme (PTAS)** for A is an alg that,

Poly Time Approx Scheme (PTAS)

Def Let A be a MAX problem.
A **Poly time Approx Scheme (PTAS)** for A is an alg that, on input (x, ϵ),

Poly Time Approx Scheme (PTAS)

Def Let A be a MAX problem.
A **Poly time Approx Scheme (PTAS)** for A is an alg that,
on input (x, ϵ),
returns a y such that $\mathbf{benefit}(y) \geq (1 - \epsilon)\mathbf{OPT}(x).$

Poly Time Approx Scheme (PTAS)

Def Let A be a MAX problem.
A **Poly time Approx Scheme (PTAS)** for A is an alg that,
on input (x, ϵ),
returns a y such that $\mathbf{benefit}(y) \geq (1-\epsilon)\mathbf{OPT}(x)$.

Note Run time depends on ϵ.

Poly Time Approx Scheme (PTAS)

Def Let A be a MAX problem.
A **Poly time Approx Scheme (PTAS)** for A is an alg that,
on input (x, ϵ),
returns a y such that $\mathbf{benefit}(y) \geq (1-\epsilon)\mathbf{OPT}(x)$.

Note Run time depends on ϵ.

Can be bad, e.g., $n^{2^{1/\epsilon^2}}$.

Complexity Classes of Approx Problems

Let A be a MAX problem

Complexity Classes of Approx Problems

Let A be a MAX problem

(1) $\boldsymbol{A} \in \mathbf{PTAS}$ is $\exists$ a PTAS for A.

Complexity Classes of Approx Problems

Let A be a MAX problem

(1) $\boldsymbol{A} \in \mathbf{PTAS}$ is $\exists$ a PTAS for A.

(2) $\boldsymbol{A} \in \mathbf{APX}$ if $\exists$ $c \geq 1$ and alg M: $M(x) \geq \frac{1}{c}\mathrm{OPT}(x)$.

Complexity Classes of Approx Problems

Let A be a MAX problem

(1) $\boldsymbol{A} \in \mathbf{PTAS}$ is $\exists$ a PTAS for A.

(2) $\boldsymbol{A} \in \mathbf{APX}$ if $\exists$ $c \geq 1$ and alg M: $M(x) \geq \frac{1}{c}\mathrm{OPT}(x)$.

(3) $\boldsymbol{A} \in \mathbf{LAPX}$ if $\exists$ c and alg M: $M(x) \geq \frac{1}{c \log x}\mathrm{OPT}(x)$.

Complexity Classes of Approx Problems

Let A be a MAX problem

(1) $\boldsymbol{A} \in \mathbf{PTAS}$ is $\exists$ a PTAS for A.

(2) $\boldsymbol{A} \in \mathbf{APX}$ if $\exists$ $c \geq 1$ and alg M: $M(x) \geq \frac{1}{c}\mathrm{OPT}(x)$.

(3) $\boldsymbol{A} \in \mathbf{LAPX}$ if $\exists$ c and alg M: $M(x) \geq \frac{1}{c \log x}\mathrm{OPT}(x)$.

(4) $\boldsymbol{A} \in \mathbf{PAPX}$ if $\exists$ poly p and alg M: $M(x) \geq \frac{1}{p(x)}\mathrm{OPT}(x)$.

Complexity Classes of Approx Problems

Let A be a MAX problem

(1) $\boldsymbol{A} \in \mathbf{PTAS}$ is $\exists$ a PTAS for A.

(2) $\boldsymbol{A} \in \mathbf{APX}$ if $\exists$ $c \geq 1$ and alg M: $M(x) \geq \frac{1}{c}\mathrm{OPT}(x)$.

(3) $\boldsymbol{A} \in \mathbf{LAPX}$ if $\exists$ c and alg M: $M(x) \geq \frac{1}{c \log x}\mathrm{OPT}(x)$.

(4) $\boldsymbol{A} \in \mathbf{PAPX}$ if $\exists$ poly p and alg M: $M(x) \geq \frac{1}{p(x)}\mathrm{OPT}(x)$.

(5) Can define more classes.

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

(2) $\mathrm{P} \neq \mathrm{NP} \rightarrow$ the inclusions are proper:

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

(2) $\mathrm{P} \neq \mathrm{NP} \rightarrow$ the inclusions are proper:

a) $\mathrm{MAX3SAT} \in \mathrm{APX} - \mathrm{PTAS}$

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

(2) $\mathrm{P} \neq \mathrm{NP} \rightarrow$ the inclusions are proper:

a) $\mathrm{MAX3SAT} \in \mathrm{APX} - \mathrm{PTAS}$

b) $\mathrm{SETCOVER} \in \mathrm{LAPX} - \mathrm{APX}$

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

(2) $\mathrm{P} \neq \mathrm{NP} \rightarrow$ the inclusions are proper:

a) $\mathrm{MAX3SAT} \in \mathrm{APX} - \mathrm{PTAS}$

b) $\mathrm{SETCOVER} \in \mathrm{LAPX} - \mathrm{APX}$

c) $\mathrm{CLIQ} \in \mathrm{PAPX} - \mathrm{LAPX}$.

What do we Know?

The following are known:
(1) $\mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{LAPX} \subseteq \mathrm{PAPX}$ (this is obvious).

(2) $\mathrm{P} \neq \mathrm{NP} \rightarrow$ the inclusions are proper:

a) $\mathrm{MAX3SAT} \in \mathrm{APX} - \mathrm{PTAS}$

b) $\mathrm{SETCOVER} \in \mathrm{LAPX} - \mathrm{APX}$

c) $\mathrm{CLIQ} \in \mathrm{PAPX} - \mathrm{LAPX}$.

d) $\mathrm{TSP} \notin \mathrm{PAPX}$.

Approx Reductions

Example of an Approx Reduction

Recall $\text{MAX3SAT} \notin \text{PTAS}$.

Example of an Approx Reduction

Recall $\mathrm{MAX3SAT} \notin \mathrm{PTAS}$.

Def IS returns the largest ind set.

Example of an Approx Reduction

Recall $\mathrm{MAX3SAT} \notin \mathrm{PTAS}$.

Def IS returns the largest ind set.

Thm If $\mathrm{IS} \in \mathrm{PTAS}$ then $\mathrm{MAX3SAT} \in \mathrm{PTAS}$.
Assume $\mathrm{IS} \in \mathrm{PTAS}$. We give PTAS for $\mathrm{MAX3SAT}$.

Example of an Approx Reduction

Recall $\mathrm{MAX3SAT} \notin \mathrm{PTAS}$.

Def IS returns the largest ind set.

Thm If $\mathrm{IS} \in \mathrm{PTAS}$ then $\mathrm{MAX3SAT} \in \mathrm{PTAS}$.
Assume $\mathrm{IS} \in \mathrm{PTAS}$. We give PTAS for $\mathrm{MAX3SAT}$.
1) Input (ϕ, ϵ).

Example of an Approx Reduction

Recall $\text{MAX3SAT} \notin \text{PTAS}$.

Def IS returns the largest ind set.

Thm If $\text{IS} \in \text{PTAS}$ then $\text{MAX3SAT} \in \text{PTAS}$.
Assume $\text{IS} \in \text{PTAS}$. We give PTAS for MAX3SAT.
1) Input (ϕ, ϵ).
2) Form graph G SEE NEXT SLIDE.

Example of an Approx Reduction

Recall $\text{MAX3SAT} \notin \text{PTAS}$.

Def IS returns the largest ind set.

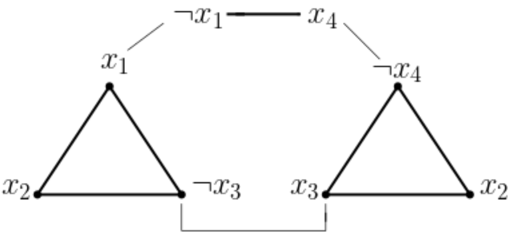
Thm If $\text{IS} \in \text{PTAS}$ then $\text{MAX3SAT} \in \text{PTAS}$.
Assume $\text{IS} \in \text{PTAS}$. We give PTAS for MAX3SAT.
1) Input (ϕ, ϵ).
2) Form graph G SEE NEXT SLIDE.
3) Use PTAS on (G, ϵ) to get Ind set of size $\geq (1 - \epsilon)\text{OPT}(G)$ clauses.

Example of an Approx Reduction

Recall $\mathrm{MAX3SAT} \notin \mathrm{PTAS}$.

Def IS returns the largest ind set.

Thm If $\mathrm{IS} \in \mathrm{PTAS}$ then $\mathrm{MAX3SAT} \in \mathrm{PTAS}$.
Assume $\mathrm{IS} \in \mathrm{PTAS}$. We give PTAS for $\mathrm{MAX3SAT}$.
1) Input (ϕ, ϵ).
2) Form graph G SEE NEXT SLIDE.
3) Use PTAS on (G, ϵ) to get Ind set of size $\geq (1-\epsilon)\mathrm{OPT}(G)$ clauses.
4) Easily map that Ind Set to a partial assignment that satisfies $\geq (1-\epsilon)\mathrm{OPT}(\phi)$.



$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4)$$

Figure: MAX3SAT $\leq$ IS

Formal Def of Approx Preserving Reduction

Def A, B be 2 problems. An **approximation preserving reduction (APR)** from A to B is a **pair of poly time functions** $x \rightarrow x'$ and $y' \rightarrow y$
1) If x is an instance of A then x' is an instance of B.
2) If y' is a solution for x' then y is a solution for x.
3) If y' is a **good** solution for x' then y is a **good** solution for x.
(The notion of **good** will vary.)

Formal Def of Approx Preserving Reduction

Def A, B be 2 problems. An **approximation preserving reduction (APR)** from A to B is a **pair of poly time functions** $x \rightarrow x'$ and $y' \rightarrow y$
1) If x is an instance of A then x' is an instance of B.
2) If y' is a solution for x' then y is a solution for x.
3) If y' is a **good** solution for x' then y is a **good** solution for x.
(The notion of **good** will vary.)

We are only interested in **good** solutions. Hence we may restrict y' to solutions that do not have an obvious improvement.
Example We assume a solutions for $\mathrm{MAX3SAT}$ will assign a var that only appears positively to T.

L-Reductions

Def An ***L*-reduction** $A \leq_L B$ is an APR where:

L-Reductions

Def An ***L*-reduction** $A \leq_L B$ is an APR where:
(1) $\mathrm{OPT}_B(x') = O(\mathrm{OPT}_A(x))$

L-Reductions

Def An ***L*-reduction** $A \leq_L B$ is an APR where:
(1) $\mathrm{OPT}_B(x') = O(\mathrm{OPT}_A(x))$
(2) $|\mathrm{benefit}_A(x') - \mathrm{OPT}_A(x)| = O(|\mathrm{benefit}_B(y') - \mathrm{OPT}_B(y)|)$

L-Reductions

Def An ***L*-reduction** $A \leq_L B$ is an APR where:
(1) $\mathrm{OPT}_B(x') = O(\mathrm{OPT}_A(x))$
(2) $|\mathrm{benefit}_A(x') - \mathrm{OPT}_A(x)| = O(|\mathrm{benefit}_B(y') - \mathrm{OPT}_B(y)|)$

Thm If $B \in \mathrm{PTAS}$ and $A \leq_L B$ then $A \in \mathrm{PTAS}$.

APX-Complete and APX-Hard

Def Let A be a max-problem.

APX-Complete and APX-Hard

Def Let A be a max-problem.

(1) A is APX-hard if $\mathrm{MAX3SAT} \leq_L A$.

APX-Complete and APX-Hard

Def Let A be a max-problem.

(1) A is APX-hard if $\mathrm{MAX3SAT} \le_L A$.

(2) A is APX-complete if A is APX-hard and $A \in \mathrm{APX}$.

APX-Complete and APX-Hard

Def Let A be a max-problem.

(1) A is APX-hard if $\mathrm{MAX3SAT} \le_L A$.

(2) A is APX-complete if A is APX-hard and $A \in \mathrm{APX}$.

We showed that IS is APX-hard.

APX-Complete and APX-Hard

Def Let A be a max-problem.

(1) A is APX-hard if $\mathrm{MAX3SAT} \le_L A$.

(2) A is APX-complete if A is APX-hard and $A \in \mathrm{APX}$.

We showed that IS is APX-hard.

Its PAPX-complete since CLIQ is PAPX-complete.

MAX3SAT $\leq_L$ MAX3SATE-3

MAXthSAT and Variants

Def
(1) **MAX3SAT** Input a 3CNF fml ϕ.
Output: Max number of clauses that can be satisfied.

MAXthSAT and Variants

Def
(1) **MAX3SAT** Input a 3CNF fml ϕ.
Output: Max number of clauses that can be satisfied.

(2) **MAX3SATE-a** Input 3CNF fml ϕ where every var occurs $\le a$.
Output: Max number of clauses that can be satisfied.

MAX3SAT $\leq_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

MAX3SAT $\le_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.

MAX3SAT $\leq_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.
2. For each variable x that occurs ≥ 4 times do the following:

MAX3SAT $\le_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.
2. For each variable x that occurs ≥ 4 times do the following:
    2.1 Let k be the number of times x occurs. Introduce new variables $z_1, \ldots, z_k$.

MAX3SAT $\leq_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.
2. For each variable x that occurs ≥ 4 times do the following:
   2.1 Let k be the number of times x occurs. Introduce new variables $z_1, \ldots, z_k$.
   2.2 Replace the k occurrences of x with $z_1, \ldots, z_k$.

MAX3SAT $\leq_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.
2. For each variable x that occurs ≥ 4 times do the following:
   2.1 Let k be the number of times x occurs. Introduce new variables $z_1, \ldots, z_k$.
   2.2 Replace the k occurrences of x with $z_1, \ldots, z_k$.
   2.3 Add $(z_1 \rightarrow z_2)$, $(z_2 \rightarrow z_3)$, ..., $(z_{L-1} \rightarrow z_k)$, $(z_k \rightarrow z_1)$. These clauses are an attempt to force all of the z_i to have the same truth value. (If this was a decision-problem reduction then the attempt would succeed.)

MAX3SAT $\le_L$ MAX3SATE-3

We show **bad reduction** to motivate a **good reduction.**

1. Input $\phi(x_1, \ldots, x_n)$. Assume ϕ has m clauses.
2. For each variable x that occurs ≥ 4 times do the following:
   - 2.1 Let k be the number of times x occurs. Introduce new variables $z_1, \ldots, z_k$.
   - 2.2 Replace the k occurrences of x with $z_1, \ldots, z_k$.
   - 2.3 Add $(z_1 \rightarrow z_2)$, $(z_2 \rightarrow z_3)$, …, $(z_{L-1} \rightarrow z_k)$, $(z_k \rightarrow z_1)$. These clauses are an attempt to force all of the z_i to have the same truth value. (If this was a decision-problem reduction then the attempt would succeed.)

Output ϕ'.

What is Wrong with that Reduction?

Good News $\phi \in \mathrm{SAT}$ iff $\phi' \in \mathrm{SAT}$.

What is Wrong with that Reduction?

Good News $\phi \in \mathrm{SAT}$ iff $\phi' \in \mathrm{SAT}$.

Caveat We need to be able to take an assignment that satisfies **many** clauses of ϕ' and map it to an assignment that satisfies **many** clauses of ϕ.

What is Wrong with that Reduction?

Good News $\phi \in \mathrm{SAT}$ iff $\phi' \in \mathrm{SAT}$.

Caveat We need to be able to take an assignment that satisfies **many** clauses of ϕ' and map it to an assignment that satisfies **many** clauses of ϕ.

Bad News Example of why reduction does not work.

$$\phi(x) = (x \vee x \vee x) \wedge \cdots \wedge (x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x) \wedge \cdots \wedge (\neg x \vee \neg x \vee \neg x)$$

What is Wrong with that Reduction?

Good News $\phi \in \mathrm{SAT}$ iff $\phi' \in \mathrm{SAT}$.

Caveat We need to be able to take an assignment that satisfies **many** clauses of ϕ' and map it to an assignment that satisfies **many** clauses of ϕ.

Bad News Example of why reduction does not work.

$$\phi(x) = (x \vee x \vee x) \wedge \cdots \wedge (x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x) \wedge \cdots \wedge (\neg x \vee \neg x \vee \neg x)$$

There are m $(x \vee x \vee x)$ clauses and m $(\neg x \vee \neg x \vee \neg x)$ clauses. Note that $\mathrm{MAX3SAT}(\phi) = m$.

What is Wrong with that Reduction?

Good News $\phi \in \mathrm{SAT}$ iff $\phi' \in \mathrm{SAT}$.

Caveat We need to be able to take an assignment that satisfies **many** clauses of ϕ' and map it to an assignment that satisfies **many** clauses of ϕ.

Bad News Example of why reduction does not work.

$$\phi(x) = (x \vee x \vee x) \wedge \cdots \wedge (x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x) \wedge \cdots \wedge (\neg x \vee \neg x \vee \neg x)$$

There are m $(x \vee x \vee x)$ clauses and m $(\neg x \vee \neg x \vee \neg x)$ clauses. Note that $\mathrm{MAX3SAT}(\phi) = m$.

Next Slide has ϕ'

ϕ'

$$(z_1 \vee z_2 \vee z_3) \wedge \cdots \wedge (z_{3m-2} \vee z_{3m-1} \vee z_{3m}) \wedge$$

$$(\neg z_{3m+1} \vee \neg z_{3m+2} \vee \neg z_{3m+3}) \wedge \cdots \wedge (\neg z_{6m-2} \vee \neg z_{6m-1} \vee \neg z_{6m}) \wedge$$

$$(z_1 \rightarrow z_2) \wedge \cdots \wedge (z_{6m-1} \rightarrow z_{6m}) \wedge (z_{6m} \rightarrow z_1)$$

ϕ'

$$(z_1 \vee z_2 \vee z_3) \wedge \cdots \wedge (z_{3m-2} \vee z_{3m-1} \vee z_{3m}) \wedge$$

$$(\neg z_{3m+1} \vee \neg z_{3m+2} \vee \neg z_{3m+3}) \wedge \cdots \wedge (\neg z_{6m-2} \vee \neg z_{6m-1} \vee \neg z_{6m}) \wedge$$

$$(z_1 \rightarrow z_2) \wedge \cdots \wedge (z_{6m-1} \rightarrow z_{6m}) \wedge (z_{6m} \rightarrow z_1)$$

Set $z_1, \ldots, z_{3m}$ to T and $z_{3m+1}, \ldots, z_{6m}$ to F. We satisfy every single clause except $z_{3m} \rightarrow z_{3m+1}$. Thats $m + m + 6m - 1 = 8m - 1$ clauses.

ϕ'

$$(z_1 \vee z_2 \vee z_3) \wedge \cdots \wedge (z_{3m-2} \vee z_{3m-1} \vee z_{3m}) \wedge$$

$$(\neg z_{3m+1} \vee \neg z_{3m+2} \vee \neg z_{3m+3}) \wedge \cdots \wedge (\neg z_{6m-2} \vee \neg z_{6m-1} \vee \neg z_{6m}) \wedge$$

$$(z_1 \rightarrow z_2) \wedge \cdots \wedge (z_{6m-1} \rightarrow z_{6m}) \wedge (z_{6m} \rightarrow z_1)$$

Set $z_1, \ldots, z_{3m}$ to T and $z_{3m+1}, \ldots, z_{6m}$ to F. We satisfy every single clause except $z_{3m} \rightarrow z_{3m+1}$. Thats $m + m + 6m - 1 = 8m - 1$ clauses.

Upshot $\mathrm{MAX3SAT}(\phi) = m$ and $\mathrm{MAX3SAT}(\phi') = 8m - 1$. That doesn't seem to bad. But wait....

No Map from y' to y

It Gets Worse There is no useful way to take that assignment and map it to an assignment for ϕ that satisfies many clauses.

No Map from y' to y

It Gets Worse There is no useful way to take that assignment and map it to an assignment for ϕ that satisfies many clauses.

What We Did Wrong We replaced x with z_1, z_2, z_3 and we **intended** z_1, z_2, z_3 them to all get the same truth value. But we did nothing to enforce that.

No Map from y' to y

It Gets Worse There is no useful way to take that assignment and map it to an assignment for ϕ that satisfies many clauses.

What We Did Wrong We replaced x with z_1, z_2, z_3 and we **intended** z_1, z_2, z_3 them to all get the same truth value. But we did nothing to enforce that.

What To Do We replaced x with z_1, z_2, z_3 in such a way that making them all the same will be **beneficial** towards getting more clauses satisfied.

No Map from y' to y

It Gets Worse There is no useful way to take that assignment and map it to an assignment for ϕ that satisfies many clauses.

What We Did Wrong We replaced x with z_1, z_2, z_3 and we **intended** z_1, z_2, z_3 them to all get the same truth value. But we did nothing to enforce that.

What To Do We replaced x with z_1, z_2, z_3 in such a way that making them all the same will be **beneficial** towards getting more clauses satisfied.

Small Caveat We will actually work with $z_1, \ldots, z_7$.

Instead of Cycles Use . . .

In failed reduction we used a **cycle** to connect the different variables that are supposed to all have the same truth value.

Instead of Cycles Use . . .

In failed reduction we used a **cycle** to connect the different variables that are supposed to all have the same truth value.

In the correct reduction we will use a more complicated graph.

Instead of Cycles Use ...

In failed reduction we used a **cycle** to connect the different variables that are supposed to all have the same truth value.

In the correct reduction we will use a more complicated graph.
Def Let $d \in \mathbb{N}$. A ***d*-expander graph** $G = (V, E)$ has:
(1) every vertex has degree d
(2) for every partition $V = V_1 \cup V_2$, $E(V_1, V_2) \geq \min(|V_1|, |V_2|)$.

Instead of Cycles Use ...

In failed reduction we used a **cycle** to connect the different variables that are supposed to all have the same truth value.

In the correct reduction we will use a more complicated graph.

Def Let $d \in \mathbb{N}$. A ***d*-expander graph** $G = (V, E)$ has:

(1) every vertex has degree d

(2) for every partition $V = V_1 \cup V_2$, $E(V_1, V_2) \geq \min(|V_1|, |V_2|)$.

An expander graph has properties of both sparse graphs (low degree) and dense graphs (lots of edges).

Instead of Cycles Use . . .

In failed reduction we used a **cycle** to connect the different variables that are supposed to all have the same truth value.

In the correct reduction we will use a more complicated graph.

Def Let $d \in \mathbb{N}$. A ***d*-expander graph** $G = (V, E)$ has:
(1) every vertex has degree d
(2) for every partition $V = V_1 \cup V_2$, $E(V_1, V_2) \geq \min(|V_1|, |V_2|)$.

An expander graph has properties of both sparse graphs (low degree) and dense graphs (lots of edges).

Known for all $k \equiv 0 \pmod 2$, there exists a 3-expander graph on k vertices. we will assume every var that appears ≥ 8 times appears an even number of times.

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.

MAX3SAT $\le_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \ge 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \to z_j)$ and $(z_j \to z_i)$.

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \rightarrow z_j)$ and $(z_j \rightarrow z_i)$.
   z_i will occur 7 times in ϕ':

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \rightarrow z_j)$ and $(z_j \rightarrow z_i)$.
   z_i will occur 7 times in ϕ':
   (a) once in the place it replaces x in the original formula,

MAX3SAT $\le_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \ge 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \to z_j)$ and $(z_j \to z_i)$.
   z_i will occur 7 times in ϕ':
   (a) once in the place it replaces x in the original formula,
   (b) three times in clauses of the form $z_i \to z_j$ (deg 3),

MAX3SAT $\leq_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \geq 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \rightarrow z_j)$ and $(z_j \rightarrow z_i)$.
   z_i will occur 7 times in ϕ':
   (a) once in the place it replaces x in the original formula,
   (b) three times in clauses of the form $z_i \rightarrow z_j$ (deg 3),
   (c) three times in clauses of the form $z_j \rightarrow z_i$ (deg 3).

MAX3SAT $\le_L$ MAX3SATE-7

Here is the reduction:

1. Input $\phi(x_1, \ldots, x_n)$.
2. If x occurs $k \ge 8$ times do the following:
   1) Introduce $z_1, \ldots, z_k$.
   2) Replace the k occurrences of x with $z_1, \ldots, z_k$.
   3) G is a 3-expander graph on k vertices $\{1, \ldots, k\}$.
   $\forall$ edges $\{i, j\}$ add clauses $(z_i \rightarrow z_j)$ and $(z_j \rightarrow z_i)$.
   z_i will occur 7 times in ϕ':
   (a) once in the place it replaces x in the original formula,
   (b) three times in clauses of the form $z_i \rightarrow z_j$ (deg 3),
   (c) three times in clauses of the form $z_j \rightarrow z_i$ (deg 3).
   The last two come from G having degree 3.

Clearly every var occurs ≤ 7 times.

Why Does This Reduction Work?

How to go from an assignment for ϕ' to an assignment for ϕ.

Why Does This Reduction Work?

How to go from an assignment for ϕ' to an assignment for ϕ.

Let $\vec{b}'$ be an assignment for ϕ'.

Why Does This Reduction Work?

How to go from an assignment for ϕ' to an assignment for ϕ.

Let $\vec{b}'$ be an assignment for ϕ'.

If x occurs ≤ 7 times in ϕ then x is in ϕ' and gets the same truth value that $\vec{b}'$ gave it.

Why Does This Reduction Work?

How to go from an assignment for ϕ' to an assignment for ϕ.

Let $\vec{b'}$ be an assignment for ϕ'.

If x occurs ≤ 7 times in ϕ then x is in ϕ' and gets the same truth value that $\vec{b'}$ gave it.

If x occurs $k \geq 8$ times in ϕ then there are $z_1, \ldots, z_k$ in ϕ'.

Why Does This Reduction Work?

How to go from an assignment for ϕ' to an assignment for ϕ.

Let $\vec{b'}$ be an assignment for ϕ'.

If x occurs ≤ 7 times in ϕ then x is in ϕ' and gets the same truth value that $\vec{b'}$ gave it.

If x occurs $k \geq 8$ times in ϕ then there are $z_1, \ldots, z_k$ in ϕ'.

This is the interesting case so goto the next slide.

Why Does This Reduction Work? Interesting Case

Recall We said that if can **easily** improve $\vec{b}'$ we assume that has already been done.

Why Does This Reduction Work? Interesting Case

Recall We said that if can **easily** improve $\vec{b}'$ we assume that has already been done.

We show that all the $z_1, \ldots, z_k$ are assigned same value.
We can then set x to that value in $\vec{b}$.

Why Does This Reduction Work? Interesting Case

Recall We said that if can **easily** improve $\vec{b}'$ we assume that has already been done.

We show that all the $z_1, \ldots, z_k$ are assigned same value.
We can then set x to that value in $\vec{b}$.
Intuition If we set all of the z_i to SAME truth value then all of the clauses from the expander, of the form $z_i \rightarrow z_j$, will be set to T.

Why Does This Reduction Work? Interesting Case

Recall We said that if can **easily** improve $\vec{b}'$ we assume that has already been done.

We show that all the $z_1, \ldots, z_k$ are assigned same value.
We can then set x to that value in $\vec{b}$.
Intuition If we set all of the z_i to SAME truth value then all of the clauses from the expander, of the form $z_i \rightarrow z_j$, will be set to T. More of these clauses will be set T then clauses in the original formula which might get flipped to F.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

For every edge (i, j) where $i \in Z_T$ and $j \in Z_F$, there are **two** clauses, $z_i \to z_j$ and $z_j \to z_i$. Hence there are $\geq 2|Z_T|$ clauses that connect a variable from Z_T to a variable from Z_F.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

For every edge (i, j) where $i \in Z_T$ and $j \in Z_F$, there are **two** clauses, $z_i \rightarrow z_j$ and $z_j \rightarrow z_i$. Hence there are $\geq 2|Z_T|$ clauses that connect a variable from Z_T to a variable from Z_F.
IF set all of the $z \in Z_F$ to T then
(1) $2|Z_F|$ clauses from the expander graph switch from F to T.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

For every edge (i, j) where $i \in Z_T$ and $j \in Z_F$, there are **two** clauses, $z_i \rightarrow z_j$ and $z_j \rightarrow z_i$. Hence there are $\geq 2|Z_T|$ clauses that connect a variable from Z_T to a variable from Z_F.
IF set all of the $z \in Z_F$ to T then
(1) $2|Z_F|$ clauses from the expander graph switch from F to T.
(2) $\leq |Z_F|$ clauses from the main formula switch from T to F.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

For every edge (i, j) where $i \in Z_T$ and $j \in Z_F$, there are **two** clauses, $z_i \rightarrow z_j$ and $z_j \rightarrow z_i$. Hence there are $\geq 2|Z_T|$ clauses that connect a variable from Z_T to a variable from Z_F.
IF set all of the $z \in Z_F$ to T then
(1) $2|Z_F|$ clauses from the expander graph switch from F to T.
(2) $\leq |Z_F|$ clauses from the main formula switch from T to F.
Net Gain of $\geq |Z_F|$ clauses are T.

Why Does This Reduction Work? Interesting Case

Let Z_T be the subset of $\{z_1, \ldots, z_k\}$ that are assigned T.
Let Z_F be the subset of $\{z_1, \ldots, z_k\}$ that are assigned F.

Assume $|Z_T| > |Z_F|$ (other case similar).
The expander graph has $\geq |Z_F|$ edges from Z_T to Z_F.

For every edge (i, j) where $i \in Z_T$ and $j \in Z_F$, there are **two** clauses, $z_i \rightarrow z_j$ and $z_j \rightarrow z_i$. Hence there are $\geq 2|Z_T|$ clauses that connect a variable from Z_T to a variable from Z_F.
IF set all of the $z \in Z_F$ to T then
(1) $2|Z_F|$ clauses from the expander graph switch from F to T.
(2) $\leq |Z_F|$ clauses from the main formula switch from T to F.
Net Gain of $\geq |Z_F|$ clauses are T.
Hence can assume all vars in $z_1, \ldots, z_k$ are set **the same**.

Recap and Nitpicks

Recap

Recap and Nitpicks

Recap

1) Via expander graphs, force all $z_1, \ldots, z_k$ to have same truth value.

Recap and Nitpicks

Recap

1) Via expander graphs, force all $z_1, \ldots, z_k$ to have same truth value.

2) Have a reduction ϕ to ϕ' where ϕ is 3CNF and ϕ' is 3CNF with each var occurring ≤ 7 times.

Recap and Nitpicks

Recap

1) Via expander graphs, force all $z_1, \ldots, z_k$ to have same truth value.

2) Have a reduction ϕ to ϕ' where ϕ is 3CNF and ϕ' is 3CNF with each var occurring ≤ 7 times.

3) Routine to show that this is an L-reduction.

Recap and Nitpicks

Recap

1) Via expander graphs, force all $z_1, \ldots, z_k$ to have same truth value.

2) Have a reduction ϕ to ϕ' where ϕ is 3CNF and ϕ' is 3CNF with each var occurring ≤ 7 times.

3) Routine to show that this is an L-reduction.

4) $\mathrm{MAX3SAT\text{-}7}$ is APX-complete.

What About MAX3SATE-3

Thm MAX3SAT-7 $\leq_L$ MAX3SAT-3.
This is an easy reduction using the cycles from the bad reduction.

What About MAX3SATE-3

Thm $\text{MAX3SAT-7} \leq_L \text{MAX3SAT-3}$.
This is an easy reduction using the cycles from the bad reduction.

We leave the details to the reader.

Two Reasons Lower Bounds On MAX3SAT-3 Important

Two Reasons Lower Bounds On MAX3SAT-3 Important

1. We will use $\mathrm{MAX3SAT\text{-}3}$ to prove many graph problems on bounded-degree graphs are hard to approximate. These proofs will be clever but elementary.

Two Reasons Lower Bounds On MAX3SAT-3 Important

1. We will use $\mathrm{MAX3SAT\text{-}3}$ to prove many graph problems on bounded-degree graphs are hard to approximate. These proofs will be clever but elementary.
2. We will use $\mathrm{MAX3SAT\text{-}3}$ to prove SET COVER is hard to approximate. This proof will use PCP-like machinery. (We won't get that far.)

Bounded Degree Graph Problems

Graph Problems

Notation If G is a graph then $\Delta(G)$ is the max degree.
Def

Graph Problems

Notation If G is a graph then $\Delta(G)$ is the max degree.
Def
ISB-a: Given graph G, $\Delta(G) \le a$, ret. size of max ind set.

Graph Problems

Notation If G is a graph then $\Delta(G)$ is the max degree.

Def

ISB-a: Given graph G, $\Delta(G) \le a$, ret. size of max ind set.

VCB-a: Given graph G, $\Delta(G) \le a$, ret. size of min vert. cov.

Graph Problems

Notation If G is a graph then $\Delta(G)$ is the max degree.

Def

ISB-a: Given graph G, $\Delta(G) \le a$, ret. size of max ind set.

VCB-a: Given graph G, $\Delta(G) \le a$, ret. size of min vert. cov.

DOMB-a: Given graph G, $\Delta(G) \le a$, ret. size of min dom set.

Graph Problems

Notation If G is a graph then $\Delta(G)$ is the max degree.
Def
ISB-a: Given graph G, $\Delta(G) \le a$, ret. size of max ind set.
VCB-a: Given graph G, $\Delta(G) \le a$, ret. size of min vert. cov.
DOMB-a: Given graph G, $\Delta(G) \le a$, ret. size of min dom set.

We show that, for some constant a, all of these are APX-complete.

ISB-4 Is APX-Complete

ISB-4 Is APX-Complete

AP-Hard

ISB-4 Is APX-Complete

AP-Hard

MAX3SAT-3 $\leq_L$ ISB-4 by the MAX3SAT $\leq_L$ IS reduction.

ISB-4 Is APX-Complete

AP-Hard

MAX3SAT-3 $\leq_L$ ISB-4 by the MAX3SAT $\leq_L$ IS reduction.
So ISB-4 is APX-hard.

ISB-4 Is APX-Complete

AP-Hard

$\mathrm{MAX3SAT\text{-}3} \le_L \mathrm{ISB\text{-}4}$ by the $\mathrm{MAX3SAT} \le_L \mathrm{IS}$ reduction.
So $\mathrm{ISB\text{-}4}$ is APX-hard.

$\mathrm{ISB\text{-}4} \in \mathrm{APX}$:
Halldórsson-Radhakrishnan showed that
$\mathrm{ISB\text{-}}\Delta$ has a $\frac{\Delta+2}{3}$-approx.

ISB-4 Is APX-Complete

AP-Hard

$\mathrm{MAX3SAT\text{-}3} \le_L \mathrm{ISB\text{-}4}$ by the $\mathrm{MAX3SAT} \le_L \mathrm{IS}$ reduction.
So $\mathrm{ISB\text{-}4}$ is APX-hard.

$\mathrm{ISB\text{-}4} \in \mathrm{APX}$:
Halldórsson-Radhakrishnan showed that
$\mathrm{ISB\text{-}}\Delta$ has a $\frac{\Delta+2}{3}$-approx.

If $\Delta = 4$ we get $\mathrm{ISB\text{-}3}$ has a 2-approx.

ISB-4 Is APX-Complete

AP-Hard

$\text{MAX3SAT-3} \le_L \text{ISB-4}$ by the $\text{MAX3SAT} \le_L \text{IS}$ reduction.
So ISB-4 is APX-hard.

$\text{ISB-4} \in \text{APX}$:
Halldórsson-Radhakrishnan showed that
$\text{ISB-}\Delta$ has a $\frac{\Delta+2}{3}$-approx.

If $\Delta = 4$ we get ISB-3 has a 2-approx.
So $\text{ISB-4} \in \text{APX}$.

ISB-4 Is APX-Complete

AP-Hard

$\mathrm{MAX3SAT\text{-}3} \le_L \mathrm{ISB\text{-}4}$ by the $\mathrm{MAX3SAT} \le_L \mathrm{IS}$ reduction.
So $\mathrm{ISB\text{-}4}$ is APX-hard.

$\mathrm{ISB\text{-}4} \in \mathrm{APX}$:
Halldórsson-Radhakrishnan showed that
$\mathrm{ISB\text{-}}\Delta$ has a $\frac{\Delta+2}{3}$-approx.

If $\Delta = 4$ we get $\mathrm{ISB\text{-}3}$ has a 2-approx.
So $\mathrm{ISB\text{-}4} \in \mathrm{APX}$.

We now know that $\mathrm{ISB\text{-}4}$ does not have a PTAS.

ISB-4 Is APX-Complete

AP-Hard

$\mathrm{MAX3SAT\text{-}3} \le_L \mathrm{ISB\text{-}4}$ by the $\mathrm{MAX3SAT} \le_L \mathrm{IS}$ reduction.
So $\mathrm{ISB\text{-}4}$ is APX-hard.

$\mathrm{ISB\text{-}4} \in \mathrm{APX}$:
Halldórsson-Radhakrishnan showed that
$\mathrm{ISB\text{-}\Delta}$ has a $\frac{\Delta+2}{3}$-approx.

If $\Delta = 4$ we get $\mathrm{ISB\text{-}3}$ has a 2-approx.
So $\mathrm{ISB\text{-}4} \in \mathrm{APX}$.

We now know that $\mathrm{ISB\text{-}4}$ does not have a PTAS.

Can the 2 be lowered?

ISB-4 Is APX-Complete

AP-Hard

$\mathrm{MAX3SAT\text{-}3} \le_L \mathrm{ISB\text{-}4}$ by the $\mathrm{MAX3SAT} \le_L \mathrm{IS}$ reduction.
So $\mathrm{ISB\text{-}4}$ is APX-hard.

$\mathrm{ISB\text{-}4} \in \mathrm{APX}$:
Halldórsson-Radhakrishnan showed that
$\mathrm{ISB\text{-}\Delta}$ has a $\frac{\Delta+2}{3}$-approx.

If $\Delta = 4$ we get $\mathrm{ISB\text{-}3}$ has a 2-approx.
So $\mathrm{ISB\text{-}4} \in \mathrm{APX}$.

We now know that $\mathrm{ISB\text{-}4}$ does not have a PTAS.

Can the 2 be lowered? Might be open.

VCB-4 is APX-Complete

VCB-4 is APX-Complete

APX-hard

VCB-4 is APX-Complete

APX-hard

ISB-4 $\leq_L$ VCB-4: The usual,

(1) Map G to G.

VCB-4 is APX-Complete

APX-hard

$\mathrm{ISB\text{-}4} \leq_L \mathrm{VCB\text{-}4}$: The usual,
(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

VCB-4 is APX-Complete

APX-hard

ISB-4 $\le_L$ VCB-4: The usual,

(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

VCB-4 $\in$ APX:

VCB-4 is APX-Complete

APX-hard

$\text{ISB-4} \le_L \text{VCB-4}$: The usual,

(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

$\text{VCB-4} \in \text{APX}$:

Know that VC has a 2-Approx.

VCB-4 is APX-Complete

APX-hard

$\mathrm{ISB\text{-}4} \le_L \mathrm{VCB\text{-}4}$: The usual,

(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

$\mathrm{VCB\text{-}4} \in \mathrm{APX}$:

Know that VC has a 2-Approx.

There are reasons to think VC does not have a $(2-\epsilon)$-approx.

VCB-4 is APX-Complete

APX-hard

$\text{ISB-4} \le_L \text{VCB-4}$: The usual,

(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

$\text{VCB-4} \in \text{APX}$:

Know that VC has a 2-Approx.

There are reasons to think VC does not have a $(2 - \epsilon)$-approx.
For VCB-4 is there a $(2 - \epsilon)$-approx?

VCB-4 is APX-Complete

APX-hard

$\mathrm{ISB\text{-}4} \le_L \mathrm{VCB\text{-}4}$: The usual,

(1) Map G to G.

(2) Map a vertex cover for G to its complement to get an independent set for G.

$\mathrm{VCB\text{-}4} \in \mathrm{APX}$:

Know that VC has a 2-Approx.

There are reasons to think VC does not have a $(2-\epsilon)$-approx.
For $\mathrm{VCB\text{-}4}$ is there a $(2-\epsilon)$-approx? Might be open.

DOMB-8 is APX-Complete

DOMB-8 is APX-Complete

APX-Hard

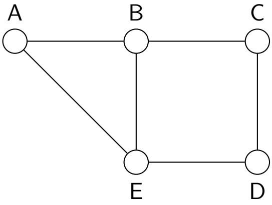
DOMB-8 is APX-Complete

APX-Hard

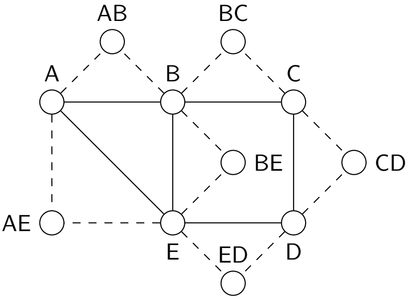
VCB-4 $\leq_L$ DOMB-8:

See next slide

VCB-4 $\leq_L$ DOMB-8



G =
Vertex Cover {E,B,C}



G' =
Dominating Set {BE,E,AB,C}
= {B,E,C}

DOMB-Δ in APX

Parekh showed that DOMB-Δ has a $O(\log \Delta)$-approx.

DOMB-Δ in APX

Parekh showed that DOMB-Δ has a $O(\log \Delta)$-approx.
Pinning down the exact constant may be open.

DOMB-Δ in APX

Parekh showed that DOMB-Δ has a $O(\log \Delta)$-approx.
Pinning down the exact constant may be open.

We know that DOMB-Δ is APX but not PTAS.

Recap

We have shown

$$\text{MAX3SAT} \leq_L \text{MAX3SAT-3} \leq_L \text{ISB-4} \leq_L \text{VCB-4} \leq_L \text{DOMB-8}$$

and that all of these problems are APX-complete.

Recap

We have shown

$$\mathrm{MAX3SAT} \leq_L \mathrm{MAX3SAT\text{-}3} \leq_L \mathrm{ISB\text{-}4} \leq_L \mathrm{VCB\text{-}4} \leq_L \mathrm{DOMB\text{-}8}$$

and that all of these problems are APX-complete.

Our next goal is to show that MAXCUT is APX-complete.

Recap

We have shown

$$\mathrm{MAX3SAT} \leq_L \mathrm{MAX3SAT\text{-}3} \leq_L \mathrm{ISB\text{-}4} \leq_L \mathrm{VCB\text{-}4} \leq_L \mathrm{DOMB\text{-}8}$$

and that all of these problems are APX-complete.

Our next goal is to show that MAXCUT is APX-complete.

We will need two more problems in logic to help us get there.

Logic Problems We Will Need

Def
MAX2SAT Given a 2CNF formula ϕ,

Logic Problems We Will Need

Def

MAX2SAT Given a 2CNF formula ϕ,
what is the max number of clauses that can be satisfied by an assignment?

Logic Problems We Will Need

Def
MAX2SAT Given a 2CNF formula ϕ,
what is the max number of clauses that can be satisfied by an assignment?

MAX3NAESAT Given a 3CNF formula ϕ,

Logic Problems We Will Need

Def
MAX2SAT Given a 2CNF formula ϕ,
what is the max number of clauses that can be satisfied by an assignment?

MAX3NAESAT Given a 3CNF formula ϕ,
what is the max number of clauses that can be satisfied by an assignment **with the extra condition** that no clause has all of its literals T. (NAE stands for Not-All-Equal.)

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show ISB-4 $\le_L$ MAX2SAT.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T
It may make more 2-clauses true.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\mathrm{ISB}\text{-}4 \le_L \mathrm{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T
It may make more 2-clauses true.
It will make ONE 1-clause, $\{u\}$ F.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T
It may make more 2-clauses true.
It will make ONE 1-clause, $\{u\}$ F.
Hence number of clauses satisfied will not decrease.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T
It may make more 2-clauses true.
It will make ONE 1-clause, $\{u\}$ F.
Hence number of clauses satisfied will not decrease.

The set of variables set T are an IS in G.

MAX2SAT is APX-Complete: APX-Hard

APX-Hard We show $\text{ISB-4} \le_L \text{MAX2SAT}$.

1. Input $G = (V, E)$, a graph of degree ≤ 4.
2. Form ϕ as follows:
   (1) For every $v \in V$ we have clause $\{v\}$.
   (2) For every $(u, v) \in E$ we have clause $\{\overline{u} \vee \overline{v}\}$.

We map an assignment for ϕ to an IS set of G.
Key If the assignment makes some 2-clause $\overline{u} \vee \overline{v}$ F, then change the assignment to make u F.
This will make $\overline{u} \vee \overline{v}$ T
It may make more 2-clauses true.
It will make ONE 1-clause, $\{u\}$ F.
Hence number of clauses satisfied will not decrease.

The set of variables set T are an IS in G.

Leave to the reader that this works.

MAX2SAT is APX-Complete: In APX

The randomized algorithm gives a $\frac{1}{2}$-approx.

MAX2SAT is APX-Complete: In APX

The randomized algorithm gives a $\frac{1}{2}$-approx.

Can be made deterministic by Cond. Prob. Method.

MAX2SAT is APX-Complete: In APX

The randomized algorithm gives a $\frac{1}{2}$-approx.

Can be made deterministic by Cond. Prob. Method.

We now know that $\mathrm{MAX2SAT}$ is APX but not PTAS.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\leq_L$ MAX3NAESAT.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\leq_L$ MAX3NAESAT.

1. Input ϕ, a formula where every clause has ≤ 2 literals.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\le_L$ MAX3NAESAT.

1. Input ϕ, a formula where every clause has ≤ 2 literals.
2. Let z be a new variable.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\le_L$ MAX3NAESAT.

1. Input ϕ, a formula where every clause has ≤ 2 literals.
2. Let z be a new variable.
3. We form a formula ϕ' by adding $\vee z$ to all 2-clauses, and $\vee z \vee z$ to all 1-clause.

MAX3NAESAT is APX-Complete: APX-Hard

$\text{MAX2SAT} \leq_L \text{MAX3NAESAT}$.

1. Input ϕ, a formula where every clause has ≤ 2 literals.
2. Let z be a new variable.
3. We form a formula ϕ' by adding $\vee z$ to all 2-clauses, and $\vee z \vee z$ to all 1-clause.

Let $\vec{b}'$ be an assignment for ϕ' where m of the clauses are satisfied but no clause has TTT. We map it to an assignment $\vec{b}$ for ϕ.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\le_L$ MAX3NAESAT.

1. Input ϕ, a formula where every clause has ≤ 2 literals.
2. Let z be a new variable.
3. We form a formula ϕ' by adding $\vee z$ to all 2-clauses, and $\vee z \vee z$ to all 1-clause.

Let $\vec{b}'$ be an assignment for ϕ' where m of the clauses are satisfied but no clause has TTT. We map it to an assignment $\vec{b}$ for ϕ.

Case $z = T$ The m satisfied clauses all have ≥ 1 literal set F. If we flip the truth value of all the vars (z is now F) we get an assignment where m clauses are T, and none of them are TTT. Hence we can assume $z = F$.

MAX3NAESAT is APX-Complete: APX-Hard

MAX2SAT $\le_L$ MAX3NAESAT.

1. Input ϕ, a formula where every clause has ≤ 2 literals.
2. Let z be a new variable.
3. We form a formula ϕ' by adding $\vee z$ to all 2-clauses, and $\vee z \vee z$ to all 1-clause.

Let $\vec{b}'$ be an assignment for ϕ' where m of the clauses are satisfied but no clause has TTT. We map it to an assignment $\vec{b}$ for ϕ.

Case $z = T$ The m satisfied clauses all have ≥ 1 literal set F. If we flip the truth value of all the vars (z is now F) we get an assignment where m clauses are T, and none of them are TTT. Hence we can assume $z = F$.

Case $z = F$ T The assignment, not including z, is an assignment for ϕ that makes m clauses T. Hence

$$\text{benefit}(\phi', \vec{b}') = \text{benefit}(\phi, \vec{b}).$$

MAX3NAESAT is APX-Complete: In APX

MAX3NAESAT has a $\frac{3}{4}$-approx.

MAX3NAESAT is APX-Complete: In APX

MAX3NAESAT has a $\frac{3}{4}$-approx.

The usual: Pick a random assignment.

MAX3NAESAT is APX-Complete: In APX

MAX3NAESAT has a $\frac{3}{4}$-approx.

The usual: Pick a random assignment.

The prob that a clause is satisfied with the condition is the (prob that the assignment is NOT TTT or FFF), so is $\frac{6}{8} = \frac{3}{4}$

MAX3NAESAT is APX-Complete: In APX

$\mathrm{MAX3NAESAT}$ has a $\frac{3}{4}$-approx.

The usual: Pick a random assignment.

The prob that a clause is satisfied with the condition is the (prob that the assignment is NOT TTT or FFF), so is $\frac{6}{8} = \frac{3}{4}$

Hence expected number of satisfied clauses is $\frac{3}{4}$ of the clauses.

MAX3NAESAT is APX-Complete: In APX

MAX3NAESAT has a $\frac{3}{4}$-approx.

The usual: Pick a random assignment.

The prob that a clause is satisfied with the condition is the (prob that the assignment is NOT TTT or FFF), so is $\frac{6}{8} = \frac{3}{4}$

Hence expected number of satisfied clauses is $\frac{3}{4}$ of the clauses.

Can be made deterministic by the Cond. Prob. Method.. Method.

MAX3NAESAT is APX-Complete: In APX

MAX3NAESAT has a $\frac{3}{4}$-approx.

The usual: Pick a random assignment.

The prob that a clause is satisfied with the condition is the (prob that the assignment is NOT TTT or FFF), so is $\frac{6}{8} = \frac{3}{4}$

Hence expected number of satisfied clauses is $\frac{3}{4}$ of the clauses.

Can be made deterministic by the Cond. Prob. Method.. Method.

Now we know that MAX3NAESAT is in APX but not PTAS.

MAXCUT IS APX-Complete

Def **MAXCUT** Given graph G, finds $U \subseteq V$ that maximizes $E(U, V-U)$.

MAXCUT IS APX-Complete

Def MAXCUT Given graph G, finds $U \subseteq V$ that maximizes $E(U, V - U)$.

Thm MAXCUT is APX-complete.
We omit proof that
$\mathrm{MAX3NAESAT} \le_L \mathrm{MAXCUT}$

More Logic Problems!

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

We refer to **rels satisfied** rather than **clauses satisfied** since we may regard (say) satisfying exactly 1 out of the 7 literals as T as satisfying the relationship, but not 2 out of the 7.

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

We refer to **rels satisfied** rather than **clauses satisfied** since we may regard (say) satisfying exactly 1 out of the 7 literals as T as satisfying the relationship, but not 2 out of the 7.

(1) **MAXCSP** Given a C-type fml, find an assignment that maximizes the number of rels satisfied.

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

We refer to **rels satisfied** rather than **clauses satisfied** since we may regard (say) satisfying exactly 1 out of the 7 literals as T as satisfying the relationship, but not 2 out of the 7.

(1) **MAXCSP** Given a C-type fml, find an assignment that maximizes the number of rels satisfied.

(2) **MINCSP** Given a C-type fml, find an assignment that minimizes the number of rels unsatisfied. (Equiv to MAXCSP as functions, but not as approx.)

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

We refer to **rels satisfied** rather than **clauses satisfied** since we may regard (say) satisfying exactly 1 out of the 7 literals as T as satisfying the relationship, but not 2 out of the 7.

(1) **MAXCSP** Given a C-type fml, find an assignment that maximizes the number of rels satisfied.

(2) **MINCSP** Given a C-type fml, find an assignment that minimizes the number of rels unsatisfied. (Equiv to MAXCSP as functions, but not as approx.)

(3) **MAXONES** Given a C-type fml, find a assignment that satisfies all rels and maximizes the number of 1's in the assignment.

Max CSP, Min CSP, Max Ones, Min Ones

Def Let C be some type of SAT problem (e.g., 3CNF, 1-in-7-SAT, NAE-8-SAT).

We refer to **rels satisfied** rather than **clauses satisfied** since we may regard (say) satisfying exactly 1 out of the 7 literals as T as satisfying the relationship, but not 2 out of the 7.

(1) **MAXCSP** Given a C-type fml, find an assignment that maximizes the number of rels satisfied.

(2) **MINCSP** Given a C-type fml, find an assignment that minimizes the number of rels unsatisfied. (Equiv to MAXCSP as functions, but not as approx.)

(3) **MAXONES** Given a C-type fml, find a assignment that satisfies all rels and maximizes the number of 1's in the assignment.

(4)**MINONES** Given a C-type fml, find an assignment that satisfies all rels and minimizes the number of 1's in the assignment.

Connection to Other Problems

Some problems can be phrased as one of these four types.

Connection to Other Problems

Some problems can be phrased as one of these four types.
(1) **MAXCUT** is a MAXCSP problem where the relations are all 2-ary $\oplus$.
If $G = (V, E)$ is a graph then the corresponding **MAXCSP** problem is the set of relations:
$\{x_i \oplus x_j : (i, j) \in E\}$.

Connection to Other Problems

Some problems can be phrased as one of these four types.
(1) **MAXCUT** is a MAXCSP problem where the relations are all 2-ary $\oplus$.
If $G = (V, E)$ is a graph then the corresponding **MAXCSP** problem is the set of relations:
$\{x_i \oplus x_j : (i, j) \in E\}$.
(2) **VC** is a MINONES problem where the relations are 2-ary $\vee$.
If $G = (V, E)$ is a graph then the corresponding **MINCSP** problem is the set of relations:
$\{x_i \vee x_j : (i, j) \in E\}$.

There is a Classification Thm! Yeah! But. . .

Recall Schaefer's theorem classified all types of formulas as either P or NPC.

There is a Classification Thm! Yeah! But...

Recall Schaefer's theorem classified all types of formulas as either P or NPC.
Do we have the same here for MAXCSP, etc?

There is a Classification Thm! Yeah! But. . .

Recall Schaefer's theorem classified all types of formulas as either P or NPC.
Do we have the same here for MAXCSP, etc?
Good News There is a theorem that classifies all such functions in terms of how hard to approx!

There is a Classification Thm! Yeah! But. . .

Recall Schaefer's theorem classified all types of formulas as either P or NPC.
Do we have the same here for MAXCSP, etc?
Good News There is a theorem that classifies all such functions in terms of how hard to approx!

Bad News Its a mess involving some classes that are contrived just for the theorem.

There is a Classification Thm! Yeah! But...

Recall Schaefer's theorem classified all types of formulas as either P or NPC.
Do we have the same here for MAXCSP, etc?
Good News There is a theorem that classifies all such functions in terms of how hard to approx!

Bad News Its a mess involving some classes that are contrived just for the theorem.

Good News I won't be presenting it.

List of APX-Complete Problems

About this List

We list several APX-complete problems.

About this List

We list several APX-complete problems.

There are many more.

About this List

We list several APX-complete problems.

There are many more.

We will not prove any of them APX-complete.

About this List

We list several APX-complete problems.

There are many more.

We will not prove any of them APX-complete.

We note that they do not just use MAX3SAT. They use each other or the problems proven APX-complete earlier in this slide packet.

About this List

We list several APX-complete problems.

There are many more.

We will not prove any of them APX-complete.

We note that they do not just use MAX3SAT. They use each other or the problems proven APX-complete earlier in this slide packet.

This will in in contrast to LAPX which we will see has much fewer problems and only uses SET COVER for reductions.

EDGEMATCHPUZ (EMP)

Input A set of unit squares with the edges colored, and a target rectangle RECT.

EDGEMATCHPUZ (EMP)

Input A set of unit squares with the edges colored, and a target rectangle RECT.

Question Is there a packing of the squares into RECT such that all tiles sharing an edge have matching colors. (The colors are unary numbers, hence the frame will be shown strongly NP-complete. (These tiles are called *Wang Tiles* and were introduced by Hao Wang to study frames in logic.) he function version of EMP is to maximize the number of edges that match.

Max Ind Set on 3-regular, 3-edge colorable graphs

Input A 3-regular graph and a 3-coloring of the edges (no two incident edges are the same color).

Max Ind Set on 3-regular, 3-edge colorable graphs

Input A 3-regular graph and a 3-coloring of the edges (no two incident edges are the same color).

Question Find the largest independent set.

(This problem is used to show EMP is APX-complete.)

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

Question The intuition is that some alien species has three sexes and we are trying to arrange n 3-person-marriages.

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

Question The intuition is that some alien species has three sexes and we are trying to arrange n 3-person-marriages.

The output is an $M' \subseteq M$ such that

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

Question The intuition is that some alien species has three sexes and we are trying to arrange n 3-person-marriages.

The output is an $M' \subseteq M$ such that

1. All the triples in M' are disjoint (no polygamy).

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

Question The intuition is that some alien species has three sexes and we are trying to arrange n 3-person-marriages.

The output is an $M' \subseteq M$ such that

1. All the triples in M' are disjoint (no polygamy).
2. Every element of $A \cup B \cup C$ is in some triple of M' (no unmarried people).

3-Dim Matching with 2 occurrence (3DM-2)

Input Disjoint sets A, B, C with $|A| = |B| = |C| = n$, and $M \subseteq A \times B \times C$ such that every elements of $A \cup B \cup C$ appears twice.

Question The intuition is that some alien species has three sexes and we are trying to arrange n 3-person-marriages.

The output is an $M' \subseteq M$ such that

1. All the triples in M' are disjoint (no polygamy).
2. Every element of $A \cup B \cup C$ is in some triple of M' (no unmarried people).
3. In the function version of this we are trying to maximize the size of M' that satisfies the two above.

Metric TSP

Input A Weighted graph G such that for all vertices a, b, c
$w(a, c) \leq w(a, b) + w(b, c)$.

Metric TSP

Input A Weighted graph G such that for all vertices a, b, c
$w(a, c) \leq w(a, b) + w(b, c)$.

Question Find the lowest cost HAM cycle.