Lecture 23: Approximation Algorithms

Michael Dinitz

November 16, 2021
601.433/633 Introduction to Algorithms
Introduction

What should we do if a problem is NP-hard?

- Give up on efficiency?
- Give up on correctness?
- Give up on worst-case analysis?
Introduction

What should we do if a problem is NP-hard?

- Give up on efficiency?
- Give up on correctness?
- Give up on worst-case analysis?

No right or wrong answer (other than giving up on analysis altogether).
Introduction

What should we do if a problem is NP-hard?

- Give up on efficiency?
- Give up on correctness?
- Give up on worst-case analysis?

No right or wrong answer (other than giving up on analysis altogether).

Popular answer: *approximation algorithms* (one of my main research areas!)

- Give up on correctness, but in a provable, bounded way.
- Applies to optimization problems only (not pure decision problems)
- Has to run in polynomial time, but can return answer that is *approximately* correct.
Main Definition

Definition

Let $\mathcal{A}$ be some (minimization) problem, and let $I$ be an instance of that problem. Let $\text{OPT}(I)$ be the cost of the optimal solution on that instance. Let $\text{ALG}$ be a polynomial-time algorithm for $\mathcal{A}$, and let $\text{ALG}(I)$ denote the cost of the solution returned by $\text{ALG}$ on instance $I$. Then we say that $\text{ALG}$ is an $\alpha$-approximation if

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \leq \alpha$$

for all instances $I$ of $\mathcal{A}$.

- Approximation always at least 1
- For maximization, can either require $\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \alpha$ (where $\alpha < 1$) or $\frac{\text{OPT}(I)}{\text{ALG}(I)} \leq \alpha$ (where $\alpha > 1$)
Main Definition

**Definition**

Let $\mathcal{A}$ be some (minimization) problem, and let $I$ be an instance of that problem. Let $\text{OPT}(I)$ be the cost of the optimal solution on that instance. Let $\text{ALG}$ be a polynomial-time algorithm for $\mathcal{A}$, and let $\text{ALG}(I)$ denote the cost of the solution returned by $\text{ALG}$ on instance $I$. Then we say that $\text{ALG}$ is an $\alpha$-approximation if

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \leq \alpha$$

for all instances $I$ of $\mathcal{A}$.

- Approximation always at least 1
- For maximization, can either require $\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \alpha$ (where $\alpha < 1$) or $\frac{\text{OPT}(I)}{\text{ALG}(I)} \leq \alpha$ (where $\alpha > 1$)
- Also gives “fine-grained” complexity: not all $\text{NP}$-hard problems are equally hard!
Definition: $S \subseteq V$ is a vertex cover of $G = (V, E)$ if $S \cap e \neq \emptyset$ for all $e \in E$.

Definition (Vertex Cover)
Instance is graph $G = (V, E)$. Find vertex cover $S$, minimize $|S|$.

Last time: Vertex Cover NP-hard (reduction from Independent Set)
Vertex Cover

**Definition:** $S \subseteq V$ is a *vertex cover* of $G = (V, E)$ if $S \cap e \neq \emptyset$ for all $e \in E$.

**Definition (Vertex Cover)**

Instance is graph $G = (V, E)$. Find vertex cover $S$, minimize $|S|$.

Last time: **Vertex Cover NP**-hard (reduction from **Independent Set**)

So cannot expect to compute a minimum vertex cover efficiently. What about an *approximately* minimum vertex cover?

- Not an approximate vertex cover: still needs to be an actual vertex cover!
Obvious Algorithm 1

\[ S = \emptyset \]

while there is at least one uncovered edge {
    Pick arbitrary vertex \( v \) incident on at least one uncovered edge
    Add \( v \) to \( S \)
\}

Not a good approximation: star graph.

\[ \text{OPT} = 1 \]
\[ \text{ALG} = n - 1 \]
Obvious Algorithm 1

\[ S = \emptyset \]
while there is at least one uncovered edge {  
  Pick arbitrary vertex \( v \) incident on at least one uncovered edge  
  Add \( v \) to \( S \)
}

Not a good approximation: star graph.

- \( \text{OPT} = 1 \)
- \( \text{ALG} = n - 1 \)
Obvious Algorithm 2

\[
S = \emptyset
\]
while there is at least one uncovered edge {
    Let \( v \) be vertex incident on most uncovered edges
    Add \( v \) to \( S \)
}
Obvious Algorithm 2

\[ S = \emptyset \]

while there is at least one uncovered edge {
    Let \( v \) be vertex incident on most uncovered edges
    Add \( v \) to \( S \)
}

Better, but still not great.
Obvious Algorithm 2

\[ S = \emptyset \]

while there is at least one uncovered edge {
    Let \( v \) be vertex incident on most uncovered edges
    Add \( v \) to \( S \)
} 

Better, but still not great.

- \( |U| = t \)
- For all \( i \in \{2, 3, \ldots, t\} \), divide \( U \) into
  \( \lceil t/i \rceil \) disjoint sets of size \( i \):
  \( G^i_1, G^i_2, \ldots, G^i_{\lceil t/i \rceil} \)
- Add vertex for each set, edge to all elements.
Obvious Algorithm 2

\[ S = \emptyset \]
while there is at least one uncovered edge {
    Let \( v \) be vertex incident on most uncovered edges
    Add \( v \) to \( S \)
}

Better, but still not great.

- \(|U| = t\)
- For all \( i \in \{2, 3, \ldots, t\} \), divide \( U \) into \( \lfloor t/i \rfloor \) disjoint sets of size \( i \):
  \( G^i_1, G^i_2, \ldots, G^i_{\lfloor t/i \rfloor} \)
- Add vertex for each set, edge to all elements.

\[ \text{OPT} = t \]
Obvious Algorithm 2

\[ S = \emptyset \]
while there is at least one uncovered edge {
    Let \( v \) be vertex incident on most uncovered edges
    Add \( v \) to \( S \)
}

Better, but still not great.

- \(|U| = t\)
- For all \( i \in \{2, 3, \ldots, t\} \), divide \( U \) into \([t/i]\) disjoint sets of size \( i \):
  \( G_i^1, G_i^2, \ldots, G_i^{[t/i]} \)
- Add vertex for each set, edge to all elements.

\[ \text{OPT} = t \]
\[ \text{ALG} = \sum_{i=2}^{t} \frac{t}{i} \geq \sum_{i=2}^{t} \left( \frac{1}{2} \cdot \frac{t}{i} \right) = \frac{t}{2} \sum_{i=2}^{t} \frac{1}{i} = \Omega(t \log t) \]
Better Algorithm

\[ S = \emptyset \]
while there is at least one uncovered edge {
    Pick arbitrary uncovered edge \( \{u, v\} \)
    Add \( u \) and \( v \) to \( S \)
}

Theorem
This algorithm is a 2-approximation.

Suppose algorithm take \( k \) iterations. Let \( L \) be edges chosen by the algorithm, so \( |L| = k \).
\[ \implies |S| = 2k \]
\( L \) has structure: it is a matching!
\[ \implies \text{OPT} \geq k \]
\[ \implies \text{ALG} \leq 2 \cdot \text{OPT} \]
**Better Algorithm**

\[ S = \emptyset \]
while there is at least one uncovered edge {
    Pick arbitrary uncovered edge \( \{u, v\} \)
    Add \( u \) and \( v \) to \( S \)
}

**Theorem**

*This algorithm is a 2-approximation.*
Better Algorithm

\[ S = \emptyset \]

while there is at least one uncovered edge {
    Pick arbitrary uncovered edge \( \{u, v\} \)
    Add \( u \) and \( v \) to \( S \)
}

**Theorem**

*This algorithm is a 2-approximation.*

Suppose algorithm take \( k \) iterations. Let \( L \) be *edges* chosen by the algorithm, so \( |L| = k \).
Better Algorithm

\[ S = \emptyset \]

while there is at least one uncovered edge {
   Pick arbitrary uncovered edge \( \{u, v\} \)
   Add \( u \) and \( v \) to \( S \)
}

Theorem

This algorithm is a 2-approximation.

Suppose algorithm take \( k \) iterations. Let \( L \) be edges chosen by the algorithm, so \( |L| = k \).

\[ \implies |S| = 2k \]
Better Algorithm

\[ S = \emptyset \]

while there is at least one uncovered edge {
   Pick arbitrary uncovered edge \( \{u, v\} \)
   Add \( u \) and \( v \) to \( S \)
}

\[ \text{Theorem} \]

This algorithm is a 2-approximation.

Suppose algorithm take \( k \) iterations. Let \( L \) be edges chosen by the algorithm, so \( |L| = k \).

\[ \implies |S| = 2k \]

\( L \) has structure: it is a matching!
Better Algorithm

\[ S = \emptyset \]
while there is at least one uncovered edge {
  \[ \text{Pick arbitrary uncovered edge \{u, v\}} \]
  \[ \text{Add u and v to S} \]
}

Theorem

This algorithm is a 2-approximation.

Suppose algorithm take \( k \) iterations. Let \( L \) be edges chosen by the algorithm, so \( |L| = k \).

\[ \implies |S| = 2k \]

\( L \) has structure: it is a matching!

\[ \implies \text{OPT} \geq k \]
Better Algorithm

\[ S = \emptyset \]

while there is at least one uncovered edge {
    Pick arbitrary uncovered edge \( \{u, v\} \)
    Add \( u \) and \( v \) to \( S \)
}

Theorem

This algorithm is a 2-approximation.

Suppose algorithm take \( k \) iterations. Let \( L \) be edges chosen by the algorithm, so \( |L| = k \).

\[ \implies |S| = 2k \]

\( L \) has structure: it is a matching!

\[ \implies \text{OPT} \geq k \]

\[ \implies \text{ALG/OPT} \leq 2. \]
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Question: Is this enough?

Let \( \text{OPT}(LP) \) denote value of optimal LP solution: does \( \text{OPT}(LP) = \text{OPT} \)?

\( \text{OPT} = 2 \)

\( \text{OPT}(LP) = 3 \)

\( 2 \)
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

\[
\begin{align*}
\min & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Question: Is this enough?

Let \(\text{OPT}(\text{LP})\) denote value of optimal LP solution: does \(\text{OPT}(\text{LP}) = \text{OPT}\)?

\[
\text{OPT} = 2 \\
\text{OPT}(\text{LP}) = 3
\]

Michael Dinitz  
Lecture 23: Approximation Algorithms  
November 16, 2021  
8 / 14
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

\[
\begin{align*}
\min & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Question: Is this enough? ❌

- Let \(\text{OPT}(\text{LP})\) denote value of optimal LP solution: does \(\text{OPT}(\text{LP}) = \text{OPT}\)?
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

$$\min \sum_{v \in V} x_v$$

subject to

$$x_u + x_v \geq 1 \quad \forall \{u, v\} \in E$$

$$0 \leq x_u \leq 1 \quad \forall u \in V$$

Question: Is this enough?

- Let $\text{OPT}(\text{LP})$ denote value of optimal LP solution: does $\text{OPT}(\text{LP}) = \text{OPT}$?

  - $\text{OPT} = 2$
  - $\text{OPT}(\text{LP}) = 3/2$
LP Structure

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Lemma

\[\text{OPT}(LP) \leq \text{OPT}\]
LP Structure

\[
\begin{align*}
\min & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Lemma

\[\text{OPT}(\text{LP}) \leq \text{OPT}\]

Proof.

Let \(S\) be optimal vertex cover (so \(|S| = \text{OPT}\)).

Let \(x_v = \begin{cases} 
1 & \text{if } v \in S \\
0 & \text{otherwise}
\end{cases}\)
LP Structure

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Lemma

\[\text{OPT} (\text{LP}) \leq \text{OPT}\]

Proof.

Let \( S \) be optimal vertex cover (so \(|S| = \text{OPT}|\).

Let \( x_v = \begin{cases} 
1 & \text{if } v \in S \\
0 & \text{otherwise}
\end{cases} \)

\[x_u + x_v \geq 1 \text{ for all } \{u, v\} \in E \text{ by definition of } S\]

\[0 \leq x_v \leq 1 \text{ for all } v \in V \text{ by definition}\]
LP Structure

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Lemma

\[\text{OPT}(\text{LP}) \leq \text{OPT}\]

Proof.

Let \( S \) be optimal vertex cover (so \( |S| = \text{OPT} \)).

Let \( x_v = \begin{cases} 
1 & \text{if } v \in S \\
0 & \text{otherwise}
\end{cases} \)

\( x_u + x_v \geq 1 \) for all \( \{u, v\} \in E \) by definition of \( S \)

\( 0 \leq x_v \leq 1 \) for all \( v \in V \) by definition

\[\implies x \text{ feasible}\]
LP Structure

\[ \begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*} \]

Lemma

\[ \text{OPT}(LP) \leq \text{OPT} \]

Proof.

Let \( S \) be optimal vertex cover (so \(|S| = \text{OPT}|\)).

Let \( x_v = \begin{cases} 
1 & \text{if } v \in S \\
0 & \text{otherwise}
\end{cases} \)

\( x_u + x_v \geq 1 \) for all \( \{u, v\} \in E \) by definition of \( S \)

\( 0 \leq x_v \leq 1 \) for all \( v \in V \) by definition

\( \implies x \text{ feasible} \)

\( \implies \text{OPT}(LP) \leq \sum_{v \in V} x_v = |S| = \text{OPT} \)
**LP Structure**

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

**Lemma**

\[\text{OPT}(LP) \leq \text{OPT}\]

**Proof.**

Let \(S\) be optimal vertex cover (so \(|S| = \text{OPT}\)).

Let \(x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}\)

\(x_u + x_v \geq 1\) for all \(\{u, v\} \in E\) by definition of \(S\)

\(0 \leq x_v \leq 1\) for all \(v \in V\) by definition

\[\Rightarrow x\text{ feasible}
\]

\[\Rightarrow \text{OPT}(LP) \leq \sum_{v \in V} x_v = |S| = \text{OPT}\]
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x^*_v = \text{OPT}(\text{LP})$)
- Return $S = \{v \in V : x^*_v \geq 1/2\}$
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x_v^* = \text{OPT}(LP)$)
- Return $S = \{v \in V : x_v^* \geq 1/2\}$

Polytime: ✓
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x_v^* = \text{OPT}(LP)$)
- Return $S = \{v \in V : x_v^* \geq 1/2\}$

Polytime: ✓

Lemma

$S$ is a vertex cover.
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x_v^* = \text{OPT}(LP)$)
- Return $S = \{v \in V : x_v^* \geq 1/2\}$

Polytime: ✓

**Lemma**

$S$ is a vertex cover.

**Proof.**

Let $\{u, v\} \in E$.

By LP constraint, $x_u^* + x_v^* \geq 1$
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x_v^* = \text{OPT}(LP)$)
- Return $S = \{v \in V : x_v^* \geq 1/2\}$

**Lemma**

$S$ is a vertex cover.

**Proof.**

Let $\{u, v\} \in E$.

By LP constraint, $x_u^* + x_v^* \geq 1$

$\implies \max(x_u^*, x_v^*) \geq 1/2$
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x^*_v = \text{OPT}(LP)$)
- Return $S = \{v \in V : x^*_v \geq 1/2\}$

Polytime: ✓

Lemma

$S$ is a vertex cover.

Proof.

Let $\{u, v\} \in E$.

By LP constraint, $x^*_u + x^*_v \geq 1$

$\implies \max(x^*_u, x^*_v) \geq 1/2$

$\implies$ At least one of $u, v$ in $S$
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x^*_v = \text{OPT}(LP)$)
- Return $S = \{v \in V : x^*_v \geq 1/2\}$

Polytime: ✓

Lemma

$S$ is a vertex cover.

Proof.

Let $\{u, v\} \in E$.

By LP constraint, $x^*_u + x^*_v \geq 1$

$\implies \max(x^*_u, x^*_v) \geq 1/2$

$\implies$ At least one of $u, v$ in $S$
LP Rounding Algorithm

- Solve LP to get $x^*$ (so $\sum_{v \in V} x_v^* = \text{OPT}(\text{LP})$)
- Return $S = \{v \in V : x_v^* \geq 1/2\}$

Polytime: $\checkmark$

**Lemma**

$S$ is a vertex cover.

**Proof.**

Let $\{u, v\} \in E$.

By LP constraint, $x_u^* + x_v^* \geq 1$

$\implies \max(x_u^*, x_v^*) \geq 1/2$

$\implies$ At least one of $u, v$ in $S$

**Lemma**

$|S| \leq 2 \cdot \text{OPT}$.

**Proof.**

$|S| = \sum_{v \in S} 1 \leq \sum_{v \in S} 2x_v^* \leq 2 \sum_{v \in V} x_v^*$

$= 2 \cdot \text{OPT}(\text{LP}) \leq 2 \cdot \text{OPT}$
Why Use LP Rounding?

Important reason: much more flexible!

Weighted Vertex Cover: Also given $w: V \rightarrow \mathbb{R}^+$. Find vertex cover $S$ minimizing $\sum_{v \in S} w(v)$ subject to $x_u + x_v \geq 1 \quad \forall \{u, v\} \in E$.

Solve LP to get $x^\ast$. Return $S = \{v \in V : x^\ast_v \geq 1\}$.

Still: Polytime $S$ a vertex cover $\leq \text{OPT}(LP) \leq 2\cdot \text{OPT}(LP) \leq 2 \cdot \text{OPT}$. Higher level: LP provides lower bound on $\text{OPT}$. Often handy!
Why Use LP Rounding?

Important reason: much more flexible!

*Weighted Vertex Cover:* Also given \( w : V \rightarrow \mathbb{R}^+ \). Find vertex cover \( S \) minimizing \( \sum_{v \in S} w(v) \)
Why Use LP Rounding?

Important reason: much more flexible!

*Weighted Vertex Cover:* Also given $w : V \to \mathbb{R}^+$. Find vertex cover $S$ minimizing $\sum_{v \in S} w(v)$

$$\min \sum_{v \in V} w(v)x_v$$

subject to

$$x_u + x_v \geq 1 \quad \forall \{u, v\} \in E$$

$$0 \leq x_u \leq 1 \quad \forall u \in V$$
Why Use LP Rounding?

Important reason: much more flexible!

**Weighted Vertex Cover:** Also given \( w : V \to \mathbb{R}^+ \). Find vertex cover \( S \) minimizing \( \sum_{v \in S} w(v) \)

\[
\begin{align*}
\min \quad & \sum_{v \in V} w(v)x_v \\
\text{subject to} \quad & x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
\quad & 0 \leq x_u \leq 1 \quad \forall u \in V \\
\end{align*}
\]

- Solve LP to get \( x^* \)
- Return \( S = \{v \in V : x_v^* \geq 1/2\} \)
Why Use LP Rounding?

Important reason: much more flexible!

*Weighted Vertex Cover*: Also given \( w : V \rightarrow \mathbb{R}^+ \). Find vertex cover \( S \) minimizing \( \sum_{v \in S} w(v) \)

\[
\begin{align*}
\min & \quad \sum_{v \in V} w(v)x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V \\
\end{align*}
\]

Still:
- Polytime
- \( S \) a vertex cover
- \( \text{OPT}(LP) \leq \text{OPT} \)

- Solve LP to get \( x^* \)
- Return \( S = \{v \in V : x_v^* \geq 1/2\} \)
Why Use LP Rounding?

Important reason: much more flexible!

**Weighted Vertex Cover:** Also given $w: V \to \mathbb{R}^+$. Find vertex cover $S$ minimizing $\sum_{v \in S} w(v)$

\[
\begin{align*}
\text{min} & \quad \sum_{v \in V} w(v)x_v \\
\text{subject to} & \quad x_u + x_v \geq 1 \quad \forall \{u, v\} \in E \\
& \quad 0 \leq x_u \leq 1 \quad \forall u \in V
\end{align*}
\]

Still:

- Polytime
- $S$ a vertex cover
- $\text{OPT}(LP) \leq \text{OPT}$

Solve LP to get $x^*$

Return $S = \{v \in V : x^*_v \geq 1/2\}$

\[
\sum_{v \in S} w(v) \leq \sum_{v \in S} 2x^*_v w(v) \leq 2 \sum_{v \in V} w(v)x^*_v = 2 \cdot \text{OPT}(LP) \leq 2 \cdot \text{OPT}
\]
Why Use LP Rounding?

Important reason: much more flexible!

**Weighted Vertex Cover:** Also given $w : V \rightarrow \mathbb{R}^+$. Find vertex cover $S$ minimizing $\sum_{v \in S} w(v)$

$$\min \sum_{v \in V} w(v)x_v$$

subject to

- $x_u + x_v \geq 1 \quad \forall \{u, v\} \in E$
- $0 \leq x_u \leq 1 \quad \forall u \in V$

Still:

- Polytime
- $S$ a vertex cover
- $\text{OPT}(LP) \leq \text{OPT}$

Higher level: LP provides *lower bound* on $\text{OPT}$. Often main difficulty!
Reductions and Approximation

Proved Vertex Cover \( \text{NP} \)-hard by reduction from Independent Set:

- Polytime algorithm for Vertex Cover \( \implies \) polytime algorithm for Independent Set

So does this mean that a 2-approximation for Vertex Cover \( \implies \) 2-approximation for Independent Set?
Reductions and Approximation

Proved Vertex Cover NP-hard by reduction from Independent Set:

- Polytime algorithm for Vertex Cover $\implies$ polytime algorithm for Independent Set

So does this mean that a 2-approximation for Vertex Cover $\implies$ 2-approximation for Independent Set?

No!
Reductions and Approximation

Proved **Vertex Cover** NP-hard by reduction from **Independent Set**:

- Polytime algorithm for **Vertex Cover** $\implies$ polytime algorithm for **Independent Set**

So does this mean that a 2-approximation for **Vertex Cover** $\implies$ 2-approximation for **Independent Set**?

No!

**Theorem**

*Assuming P $\neq$ NP, for all constants $\epsilon > 0$ there is no polytime $n^{1-\epsilon}$-approximation for **Independent Set**.*
Reductions and Approximation

Proved **Vertex Cover** NP-hard by reduction from **Independent Set**:

- Polytime algorithm for **Vertex Cover** $\implies$ polytime algorithm for **Independent Set**

So does this mean that a 2-approximation for **Vertex Cover** $\implies$ 2-approximation for **Independent Set**?

**No!**

**Theorem**

*Assuming P $\neq$ NP, for all constants $\epsilon > 0$ there is no polytime $n^{1-\epsilon}$-approximation for **Independent Set**.*

So these two problems are actually very different!
Reductions and Approximation

Proved **Vertex Cover** **NP**-hard by reduction from **Independent Set**:

- Polytime algorithm for **Vertex Cover** $\implies$ polytime algorithm for **Independent Set**

So does this mean that a 2-approximation for **Vertex Cover** $\implies$ 2-approximation for **Independent Set**?

**No!**

**Theorem**

*Assuming P ≠ NP, for all constants $\epsilon > 0$ there is no polytime $n^{1-\epsilon}$-approximation for **Independent Set**.*

So these two problems are actually very different!

There is a notion of “approximation-preserving reduction”, but it is more involved than a normal reduction.
Recall 3-SAT: CNF formula (AND of ORs) where every clause has $\leq 3$ literals

- E3-SAT: Same, but every clause has exactly three literals (still NP-complete)
Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has $\leq 3$ literals

- E3-SAT: Same, but every clause has \textit{exactly} three literals (still \textbf{NP}-complete)

Optimization version: Max-E3SAT

- Find assignment to maximize \# satisfied clauses
Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has $\leq 3$ literals
- E3-SAT: Same, but every clause has \textit{exactly} three literals (still \textbf{NP}-complete)

Optimization version: Max-E3SAT
- Find assignment to maximize \# satisfied clauses

Easy \textit{randomized} algorithm:
Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has \( \leq 3 \) literals

- E3-SAT: Same, but every clause has \textit{exactly} three literals (still \textbf{NP}-complete)

Optimization version: Max-E3SAT

- Find assignment to maximize \# satisfied clauses

Easy \textit{randomized} algorithm: Choose random assignment!
Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has $\leq 3$ literals

- E3-SAT: Same, but every clause has *exactly* three literals (still NP-complete)

Optimization version: Max-E3SAT

- Find assignment to maximize # satisfied clauses

Easy *randomized* algorithm: Choose random assignment!

- For each variable $x_i$, set $x_i = T$ with probability $1/2$ and $F$ with probability $1/2$
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied $= \frac{7}{8}$

Random variables:

For $i \in \{1, 2, \ldots, m\}$, let $X_i = \begin{cases} 1 & \text{if clause } i \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases}$

$E[X_i] = \frac{7}{8}$

Let $X = \# \text{clauses satisfied} = \sum_{i=1}^{m} X_i$

$E[X] = E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = \frac{7}{8} m \geq \frac{7}{8} \text{OPT}$

Can be derandomized (method of conditional expectations)

Theorem (Håstad '01)

Assuming $P \neq \text{NP}$, for all constant $\varepsilon > 0$, there is no polytime $\frac{7}{8} + \varepsilon$-approximation for Max-E3SAT.
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied = $\frac{7}{8}$
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied $= \frac{7}{8}$
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied = $7/8$

Random variables:

- For $i \in \{1, 2, \ldots, m\}$, let $X_i = \begin{cases} 1 & \text{if clause } i \text{ satisfied} \\ 0 & \text{otherwise} \end{cases}$

- $E[X_i] = 7/8$
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied $= \frac{7}{8}$

Random variables:

- For $i \in \{1, 2, \ldots, m\}$, let $X_i = \begin{cases} 1 & \text{if clause } i \text{ satisfied} \\ 0 & \text{otherwise} \end{cases}$
  - $E[X_i] = \frac{7}{8}$
- Let $X = \# \text{ clauses satisfied} = \sum_{i=1}^{m} X_i$
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied = $7/8$

Random variables:

- For $i \in \{1, 2, \ldots, m\}$, let $X_i = \begin{cases} 
1 & \text{if clause } i \text{ satisfied} \\
0 & \text{otherwise} 
\end{cases}$

  - $E[X_i] = 7/8$

- Let $X = \#$ clauses satisfied = $\sum_{i=1}^{m} X_i$

\[
E[X] = E\left[ \sum_{i=1}^{m} X_i \right] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8}m \geq \frac{7}{8}\text{OPT}
\]
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause $i$: probability satisfied = $\frac{7}{8}$

Random variables:

- For $i \in \{1, 2, \ldots, m\}$, let $X_i = \begin{cases} 1 & \text{if clause } i \text{ satisfied} \\ 0 & \text{otherwise} \end{cases}$
  
  - $E[X_i] = \frac{7}{8}$

- Let $X = \# \text{ clauses satisfied} = \sum_{i=1}^{m} X_i$

  $$E[X] = E\left[\sum_{i=1}^{m} X_i\right] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8}m \geq \frac{7}{8} \text{OPT}$$

Can be derandomized (method of conditional expectations)
Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause \( i \): probability satisfied = \( \frac{7}{8} \)

Random variables:

- For \( i \in \{1, 2, \ldots, m\} \), let \( X_i = \begin{cases} 1 & \text{if clause } i \text{ satisfied} \\ 0 & \text{otherwise} \end{cases} \)

- \( E[X_i] = \frac{7}{8} \)

- Let \( X = \# \text{ clauses satisfied} = \sum_{i=1}^{m} X_i \)

- \( E[X] = E \left[ \sum_{i=1}^{m} X_i \right] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} m \geq \frac{7}{8} \text{OPT} \)

Can be derandomized (method of conditional expectations)

**Theorem (Håstad ’01)**

Assuming \( P \neq NP \), for all constant \( \epsilon > 0 \) there is no polytime \( \left( \frac{7}{8} + \epsilon \right) \)-approximation for Max-E3SAT.