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

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

approximation ratio

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

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

Main Definition

Definition

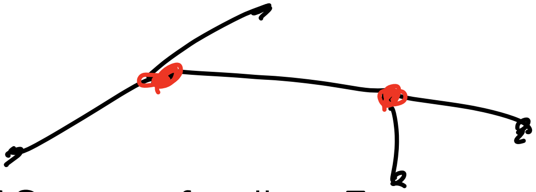
Let $\mathcal{A}$ be some (minimization) problem, and let **I** be an instance of that problem. Let **OPT(I)** be the cost of the optimal solution on that instance. Let **ALG** be a polynomial-time algorithm for $\mathcal{A}$, and let **ALG(I)** denote the cost of the solution returned by **ALG** on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

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

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

- Approximation always at least 1
- For maximization, can either require $\frac{\mathbf{ALG(I)}}{\mathbf{OPT(I)}} \geq \alpha$ (where $\alpha < 1$) or $\frac{\mathbf{OPT(I)}}{\mathbf{ALG(I)}} \leq \alpha$ (where $\alpha > 1$)
- Also gives "fine-grained" complexity: not all **NP**-hard problems are equally hard!

Vertex Cover



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

Definition (VERTEX COVER)

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

Last time: VERTEX COVER **NP**-hard (reduction from INDEPENDENT SET)

Vertex Cover

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

Definition (VERTEX COVER)

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

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

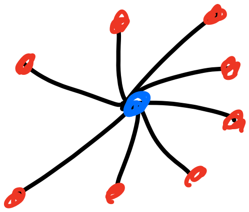
$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary vertex $\mathbf{v}$ incident on at least one uncovered edge
    Add $\mathbf{v}$ to $\mathbf{S}$
$\}$

Obvious Algorithm 1

S = ∅
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.

- **OPT = 1**
- **ALG = n − 1**



Obvious Algorithm 2

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge {
    Let **v** be vertex incident on most uncovered edges
    Add **v** to **S**
}

Obvious Algorithm 2

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge {
    Let **v** be vertex incident on most uncovered edges
    Add **v** to **S**
}

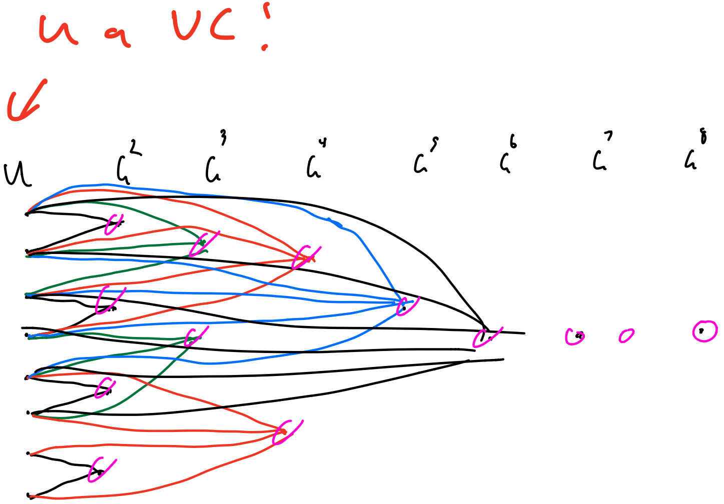
Better, but still not great.

Obvious Algorithm 2

S = ∅
while there is at least one uncovered edge {
    Let **v** be vertex incident on most uncovered edges
    Add **v** to **S**
}

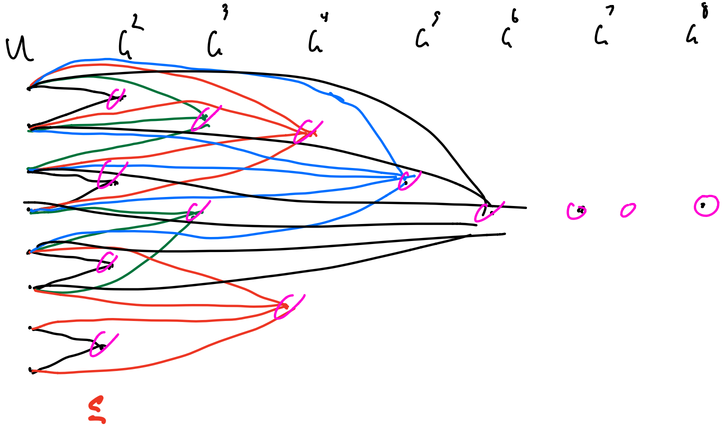
Better, but still not great.

- $|\mathbf{U}| = \mathbf{t}$
- For all $\mathbf{i} \in \{\mathbf{2}, \mathbf{3}, \ldots, \mathbf{t}\}$, divide **U** into $\lfloor \mathbf{t}/\mathbf{i} \rfloor$ disjoint sets of size **i**: $\mathbf{G}^{\mathbf{i}}_{\mathbf{1}}, \mathbf{G}^{\mathbf{i}}_{\mathbf{2}}, \ldots, \mathbf{G}^{\mathbf{i}}_{\lfloor \mathbf{t}/\mathbf{i} \rfloor}$
- Add vertex for each set, edge to all elements.



Obvious Algorithm 2

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



OPT = t

Better, but still not great.

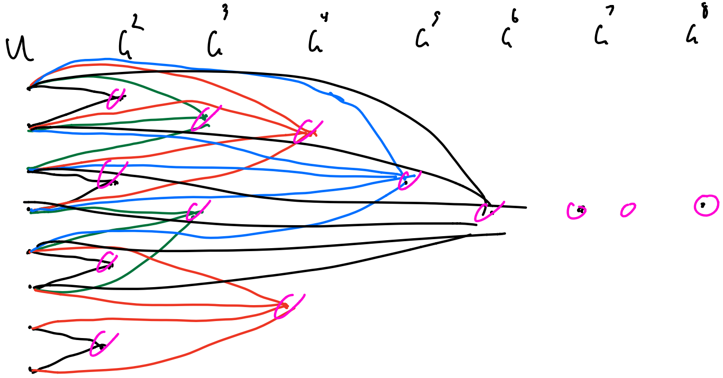
- $|\mathbf{U}| = \mathbf{t}$
- For all $\mathbf{i} \in \{\mathbf{2}, \mathbf{3}, \ldots, \mathbf{t}\}$, divide $\mathbf{U}$ into $\lfloor \mathbf{t}/\mathbf{i} \rfloor$ disjoint sets of size $\mathbf{i}$: $\mathbf{G}^{\mathbf{i}}_{\mathbf{1}}, \mathbf{G}^{\mathbf{i}}_{\mathbf{2}}, \ldots, \mathbf{G}^{\mathbf{i}}_{\lfloor \mathbf{t}/\mathbf{i} \rfloor}$
- Add vertex for each set, edge to all elements.

Obvious Algorithm 2

> $\mathbf{S} = \emptyset$
> while there is at least one uncovered edge {
>   Let $\mathbf{v}$ be vertex incident on most uncovered edges
>   Add $\mathbf{v}$ to $\mathbf{S}$
> }

Better, but still not great.

- $|\mathbf{U}| = \mathbf{t}$
- For all $\mathbf{i} \in \{2, 3, \ldots, \mathbf{t}\}$, divide $\mathbf{U}$ into $\lfloor \mathbf{t}/\mathbf{i} \rfloor$ disjoint sets of size $\mathbf{i}$: $\mathbf{G}^{\mathbf{i}}_{1}, \mathbf{G}^{\mathbf{i}}_{2}, \ldots, \mathbf{G}^{\mathbf{i}}_{\lfloor \mathbf{t}/\mathbf{i} \rfloor}$
- Add vertex for each set, edge to all elements.



OPT $= \mathbf{t}$

ALG $= \sum_{i=2}^{t} \left\lfloor \frac{t}{i} \right\rfloor \geq \sum_{i=2}^{t} \left(\frac{1}{2} \cdot \frac{t}{i} \right) = \frac{t}{2} \sum_{i=2}^{t} \frac{1}{i} = \Omega(t \log t)$

Better Algorithm

$\mathbf{S} = \emptyset$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add **u** and **v** to **S**
$\}$

Better Algorithm

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{S}$
$\}$

Theorem

This algorithm is a $\mathbf{2}$-approximation.

Better Algorithm

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{S}$
$\}$

Theorem

This algorithm is a $\mathbf{2}$*-approximation.*

Suppose algorithm take $\mathbf{k}$ iterations. Let $\mathbf{L}$ be *edges* chosen by the algorithm, so $|\mathbf{L}| = \mathbf{k}$.

Better Algorithm

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{S}$
$\}$

Theorem

This algorithm is a $\mathbf{2}$*-approximation.*

Suppose algorithm take $\mathbf{k}$ iterations. Let $\mathbf{L}$ be *edges* chosen by the algorithm, so $|\mathbf{L}| = \mathbf{k}$.
$\implies |\mathbf{S}| = \mathbf{2k}$

Better Algorithm

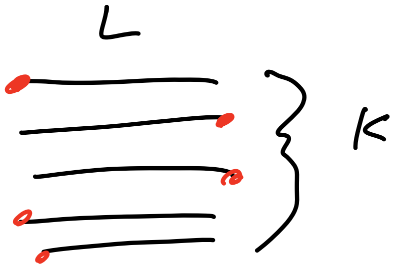
$\mathbf{S} = \varnothing$
while there is at least one uncovered edge {
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add **u** and **v** to **S**
}

Theorem

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

Suppose algorithm take **k** iterations. Let **L** be *edges* chosen by the algorithm, so $|\mathbf{L}| = \mathbf{k}$.
$\implies |\mathbf{S}| = \mathbf{2k}$

L has structure: it is a matching!



Better Algorithm

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{S}$
$\}$

Theorem

This algorithm is a $\mathbf{2}$*-approximation.*

Suppose algorithm take $\mathbf{k}$ iterations. Let $\mathbf{L}$ be *edges* chosen by the algorithm, so $|\mathbf{L}| = \mathbf{k}$.
$\implies |\mathbf{S}| = \mathbf{2k}$

$\mathbf{L}$ has structure: it is a matching!
$\implies \mathbf{OPT} \geq \mathbf{k}$

Better Algorithm

$\mathbf{S} = \varnothing$
while there is at least one uncovered edge $\{$
    Pick arbitrary uncovered edge $\{\mathbf{u}, \mathbf{v}\}$
    Add $\mathbf{u}$ and $\mathbf{v}$ to $\mathbf{S}$
$\}$

Theorem

This algorithm is a $\mathbf{2}$*-approximation.*

Suppose algorithm take $\mathbf{k}$ iterations. Let $\mathbf{L}$ be *edges* chosen by the algorithm, so $|\mathbf{L}| = \mathbf{k}$.
$\implies |\mathbf{S}| = \mathbf{2k}$

$\mathbf{L}$ has structure: it is a matching!
$\implies \mathbf{OPT} \geq \mathbf{k}$

$\implies \mathbf{ALG}/\mathbf{OPT} \leq 2$.

More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

$$\begin{array}{lll} \textbf{min} & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

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

Question: Is this enough? cost

- Let **OPT(LP)** denote ~~value~~ of optimal LP solution: does **OPT(LP)** = **OPT**?

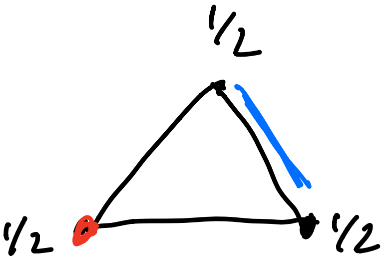
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

$$\begin{array}{lll} \min & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

Question: Is this enough?

- Let **OPT(LP)** denote value of optimal LP solution: does **OPT(LP)** = **OPT**?



- **OPT** = 2
- **OPT(LP)** = 3/2

LP Structure

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

Lemma

$OPT(LP) \leq OPT$

LP Structure

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

Lemma

$OPT(LP) \leq OPT$

Proof.

Let S be optimal vertex cover (so $|S| = OPT$).

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

LP Structure

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

Lemma

$OPT(LP) \leq OPT$

Proof.

Let S be optimal vertex cover (so $|S| = OPT$).

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

LP Structure

$$\begin{array}{lll} \min & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

Lemma

$OPT(LP) \leq OPT$

Proof.

Let S be optimal vertex cover (so $|S| = OPT$).

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

$\implies x$ feasible

LP Structure

$$\begin{array}{lll} \min & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

Lemma

$OPT(LP) \leq OPT$

Proof.

Let S be optimal vertex cover (so $|S| = OPT$).

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

$\implies x$ feasible

$\implies OPT(LP) \leq \sum_{v \in V} x_v = |S| = OPT$ □

LP Structure

$$\begin{array}{lll} \min & \sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

Lemma

$OPT(LP) \leq OPT$

Proof.

Let S be optimal vertex cover (so $|S| = OPT$).

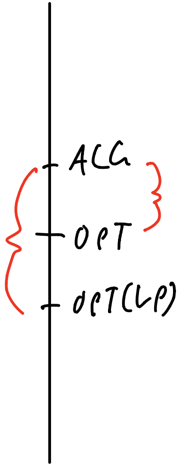
$$\text{Let } x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$$

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

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

$\implies x$ feasible

$\implies OPT(LP) \leq \sum_{v \in V} x_v = |S| = OPT$ □



LP Rounding Algorithm

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

LP Rounding Algorithm

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

Polytime: ✓

LP Rounding Algorithm

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

Polytime: ✓

Lemma

$\mathbf{S}$ *is a vertex cover.*

LP Rounding Algorithm

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

Polytime: ✓

Lemma

$\mathbf{S}$ *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 $\mathbf{x}^*$ (so $\sum_{\mathbf{v}\in\mathbf{V}} \mathbf{x}^*_{\mathbf{v}} = \mathbf{OPT(LP)}$)
- Return $\mathbf{S} = \{\mathbf{v} \in \mathbf{V} : \mathbf{x}^*_{\mathbf{v}} \geq \mathbf{1/2}\}$

Polytime: ✓

Lemma

$\mathbf{S}$ *is a vertex cover.*

Proof.

Let $\{\mathbf{u}, \mathbf{v}\} \in \mathbf{E}$.
By LP constraint, $\mathbf{x}^*_{\mathbf{u}} + \mathbf{x}^*_{\mathbf{v}} \geq \mathbf{1}$
$\implies \mathbf{max}(\mathbf{x}^*_{\mathbf{u}}, \mathbf{x}^*_{\mathbf{v}}) \geq \mathbf{1/2}$

LP Rounding Algorithm

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

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

Polytime: ✓

Lemma

$\mathbf{S}$ *is a vertex cover.*

Lemma

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

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 $\mathbf{S}$ □

LP Rounding Algorithm

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

Polytime: ✓

Lemma

$\mathbf{S}$ *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 $\mathbf{S}$ □

Lemma

$|S| \leq 2 \cdot OPT$.

Proof.

$$|S| = \sum_{v\in S} 1 \leq \sum_{v\in S} 2x_v^* \leq 2\sum_{v\in V} x_v^*$$
$$= 2 \cdot OPT(LP) \leq 2 \cdot OPT$$ □

Why Use LP Rounding?

Important reason: much more flexible!

Why Use LP Rounding?

Important reason: much more flexible!

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

Why Use LP Rounding?

Important reason: much more flexible!

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

$$\begin{array}{lll} \textbf{min} & \sum_{\mathbf{v \in V}} \mathbf{w(v) x_v} & \\ \text{subject to} & \mathbf{x_u + x_v \geq 1} & \forall \{\mathbf{u, v}\} \in \mathbf{E} \\ & \mathbf{0 \leq x_u \leq 1} & \forall \mathbf{u} \in \mathbf{V} \end{array}$$

Why Use LP Rounding?

Important reason: much more flexible!

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

$$
\begin{array}{lll}
\textbf{min} & \sum_{\mathbf{v \in V}} \mathbf{w(v) x_v} & \\
\text{subject to} & \mathbf{x_u + x_v \geq 1} & \forall \{\mathbf{u, v}\} \in \mathbf{E} \\
 & \mathbf{0 \leq x_u \leq 1} & \forall \mathbf{u} \in \mathbf{V}
\end{array}
$$

- Solve LP to get $\mathbf{x^*}$
- Return $\mathbf{S} = \{\mathbf{v \in V : x^*_v \geq 1/2}\}$

Why Use LP Rounding?

Important reason: much more flexible!

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

$$
\begin{array}{lll}
\textbf{min} & \sum_{\mathbf{v \in V}} \mathbf{w(v) x_v} & \\
\text{subject to} & \mathbf{x_u + x_v \geq 1} & \forall \{\mathbf{u, v}\} \in \mathbf{E} \\
 & \mathbf{0 \leq x_u \leq 1} & \forall \mathbf{u} \in \mathbf{V}
\end{array}
$$

- Solve LP to get $\mathbf{x^*}$
- Return $\mathbf{S} = \{\mathbf{v \in V : x_v^* \geq 1/2}\}$

Still:

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

Why Use LP Rounding?

Important reason: much more flexible!

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

$$\begin{array}{lll} \min & \sum_{v \in V} w(v) x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_u \leq 1 & \forall u \in V \end{array}$$

- Solve LP to get $\mathbf{x}^*$
- Return $\mathbf{S} = \{v \in V : x_v^* \geq 1/2\}$

Still:

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

$$\sum_{v \in S} w(v) \leq \sum_{v \in S} 2x_v^* w(v) \leq 2 \sum_{v \in V} w(v) x_v^* = 2 \cdot OPT(LP) \leq 2 \cdot OPT$$

Why Use LP Rounding?

Important reason: much more flexible!

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

$$\begin{array}{lll} \textbf{min} & \sum_{\mathbf{v} \in \mathbf{V}} \mathbf{w}(\mathbf{v}) \mathbf{x}_\mathbf{v} & \\ \text{subject to} & \mathbf{x}_\mathbf{u} + \mathbf{x}_\mathbf{v} \geq 1 & \forall \{\mathbf{u}, \mathbf{v}\} \in \mathbf{E} \\ & 0 \leq \mathbf{x}_\mathbf{u} \leq 1 & \forall \mathbf{u} \in \mathbf{V} \end{array}$$

- Solve LP to get $\mathbf{x}^*$
- Return $\mathbf{S} = \{\mathbf{v} \in \mathbf{V} : \mathbf{x}^*_\mathbf{v} \geq 1/2\}$

Still:

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

$$\sum_{\mathbf{v} \in \mathbf{S}} \mathbf{w}(\mathbf{v}) \leq \sum_{\mathbf{v} \in \mathbf{S}} 2\mathbf{x}^*_\mathbf{v} \mathbf{w}(\mathbf{v}) \leq 2 \sum_{\mathbf{v} \in \mathbf{V}} \mathbf{w}(\mathbf{v}) \mathbf{x}^*_\mathbf{v} = 2 \cdot \mathbf{OPT(LP)} \leq 2 \cdot \mathbf{OPT}$$

Higher level: LP provides *lower bound* on **OPT**. Often main difficulty!

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?

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 $\mathbf{2}$-approximation for VERTEX COVER $\implies$ $\mathbf{2}$-approximation for INDEPENDENT SET?

No!

Theorem

Assuming $\mathbf{P} \neq \mathbf{NP}$*, for all constants* $\epsilon > \mathbf{0}$ *there is no polytime* $\mathbf{n}^{\mathbf{1}-\epsilon}$*-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 $\mathbf{2}$-approximation for Vertex Cover $\implies$ $\mathbf{2}$-approximation for Independent Set?

No!

Theorem

Assuming $\mathbf{P} \neq \mathbf{NP}$*, for all constants* $\epsilon > \mathbf{0}$ *there is no polytime* $\mathbf{n}^{\mathbf{1}-\epsilon}$*-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 $\mathbf{2}$-approximation for Vertex Cover $\implies$ $\mathbf{2}$-approximation for Independent Set?

No!

Theorem

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

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)

Max-E3SAT

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

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

Optimization version: Max-E3SAT

- Find assignment to maximize # satisfied clauses

Max-E3SAT

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

Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has $\leq \mathbf{3}$ 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!

Max-E3SAT

Recall 3-SAT: CNF formula (AND of ORs) where every clause has ≤ 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 $\mathbf{x_i}$, set $\mathbf{x_i} = \mathbf{T}$ with probability $\mathbf{1/2}$ and $\mathbf{F}$ with probability $\mathbf{1/2}$

Max-E3SAT: Analysis

Algorithm: Choose random assignment

Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause **i**: probability satisfied =

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Random variables:

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

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Random variables:

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

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Random variables:

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

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

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Random variables:

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

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

Can be derandomized (method of conditional expectations)

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

Random variables:

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

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

Can be derandomized (method of conditional expectations)

Theorem (Håstad '01)

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