Lecture 24: Approximation Algorithms

Michael Dinitz

November 21, 2024

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 $OPT(I)$ 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

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

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

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

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

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

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

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)

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

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

Obvious Algorithm 1

$S = \varnothing$
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

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

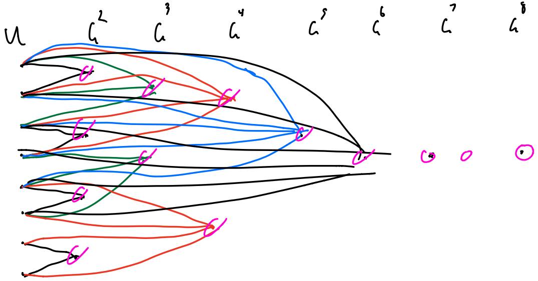
Obvious Algorithm 2

$S = \emptyset$
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

$S = \varnothing$
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.

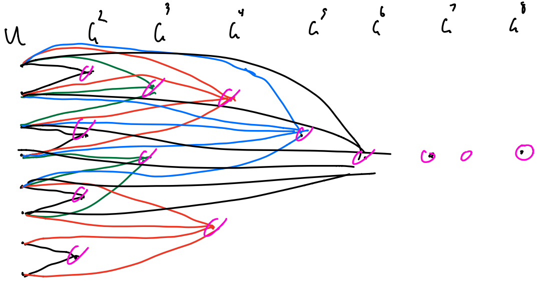
- $|U| = t$
- For all $i \in \{2, 3, \ldots, t\}$, divide U into $\lfloor t/i \rfloor$ disjoint sets of size i: $G^i_1, G^i_2, \ldots, G^i_{\lfloor t/i \rfloor}$
- Add vertex for each set, edge to all elements.

Obvious Algorithm 2

$S = \varnothing$
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.

- $|U| = t$
- 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.



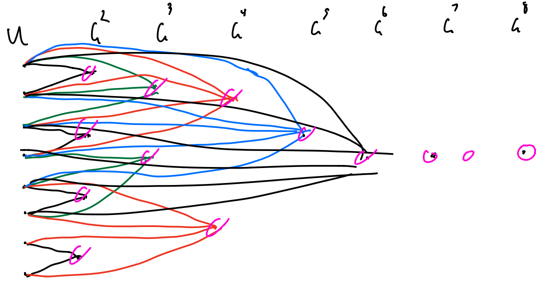
$OPT = t$

Obvious Algorithm 2

$S = \varnothing$
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.

- $|U| = t$
- 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.



$OPT = t$

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

Better Algorithm

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

Better Algorithm

$S = \varnothing$
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

$S = \varnothing$
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
$\}$

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 $\{$
    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!
$\implies OPT \geq k$

Better Algorithm

$S = \varnothing$
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 $\mathbf{2}$*-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!
$\implies OPT \geq k$

$\implies ALG/OPT \leq 2.$

More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

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

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 $OPT(LP) = OPT$?

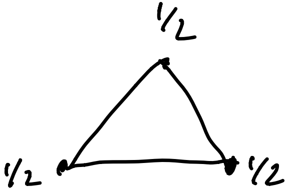
More Complicated Algorithm: LP Rounding

Write LP for vertex cover:

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

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

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

Lemma

$OPT(LP) \leq OPT$

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 $|S| = OPT$).

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

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

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

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

Lemma

$OPT(LP) \leq OPT$

Proof.

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

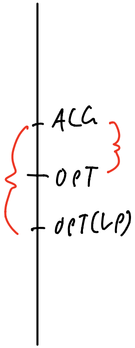
Let $x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$

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

LP Rounding Algorithm

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

Polytime: ✓

LP Rounding Algorithm

- Solve LP to get x^* (so $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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 $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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 $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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, $x_u^* + x_v^* \geq 1$
$\implies \max(x_u^*, x_v^*) \geq 1/2$

LP Rounding Algorithm

- Solve LP to get x^* (so $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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 $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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 □

Lemma

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

LP Rounding Algorithm

- Solve LP to get x^* (so $\sum_{v \in V} x_v^* = OPT(LP)$)
- 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 □

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 $w : V \to \mathbb{R}^+$. 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 $\sum_{v \in S} w(v)$

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

Why Use LP Rounding?

Important reason: much more flexible!

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

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

- Solve LP to get 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 $\sum_{v \in S} w(v)$

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

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

Still:

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

Why Use LP Rounding?

Important reason: much more flexible!

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

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

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

Still:

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

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

Why Use LP Rounding?

Important reason: much more flexible!

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

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

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

Still:

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

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

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 **2**-approximation for Vertex Cover $\implies$ **2**-approximation for Independent Set?

No!

Theorem

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

Reductions and Approximation

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

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

So does this mean that a $\mathbf{2}$-approximation for Vertex Cover $\Longrightarrow$ $\mathbf{2}$-approximation for Independent Set?

No!

Theorem

Assuming $\boldsymbol{P} \neq \boldsymbol{NP}$, for all constants $\epsilon > \mathbf{0}$ there is no polytime $\boldsymbol{n}^{\mathbf{1}-\epsilon}$-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 $\Longrightarrow$ polytime algorithm for Independent Set

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

No!

Theorem

Assuming $\boldsymbol{P} \neq \boldsymbol{NP}$*, for all constants* $\epsilon > \mathbf{0}$ *there is no polytime* $\boldsymbol{n}^{\mathbf{1}-\epsilon}$*-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 $\leq \mathbf{3}$ literals

- E3-SAT: Same, but every clause has *exactly* three literals (still $\boldsymbol{NP}$-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 $\boldsymbol{NP}$-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 $\leq \mathbf{3}$ literals

- E3-SAT: Same, but every clause has *exactly* three literals (still $\boldsymbol{NP}$-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 $\boldsymbol{NP}$-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

Max-E3SAT: Analysis

Algorithm: Choose random assignment

Clause i: probability satisfied =

Max-E3SAT: Analysis

Algorithm: Choose random assignment

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

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 = $7/8$

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 = $\sum_{i=1}^{m} X_i$

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 = $\sum_{i=1}^m X_i$

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

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 = $\sum_{i=1}^{m} X_i$

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

Can be derandomized (method of conditional expectations)

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 = $\sum_{i=1}^{m} X_i$

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

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.