VII. Approximation Algorithms: Randomisation and Rounding

Thomas Sauerwald
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover
Approximation Ratio

A randomised algorithm for a problem has approximation ratio $\rho(n)$, if for any input of size $n$, the expected cost $C$ of the returned solution and optimal cost $C^*$ satisfy:

$$\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n).$$
A randomised algorithm for a problem has approximation ratio \( \rho(n) \), if for any input of size \( n \), the expected cost \( C \) of the returned solution and optimal cost \( C^* \) satisfy:

\[
\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n).
\]

Call such an algorithm randomised \( \rho(n) \)-approximation algorithm.
A randomised algorithm for a problem has approximation ratio $\rho(n)$, if for any input of size $n$, the expected cost $C$ of the returned solution and optimal cost $C^*$ satisfy:

$$\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n).$$

Call such an algorithm randomised $\rho(n)$-approximation algorithm.

An approximation scheme is an approximation algorithm, which given any input and $\epsilon > 0$, is a $(1 + \epsilon)$-approximation algorithm.

- It is a polynomial-time approximation scheme (PTAS) if for any fixed $\epsilon > 0$, the runtime is polynomial in $n$. For example, $O(n^2/\epsilon)$.
- It is a fully polynomial-time approximation scheme (FPTAS) if the runtime is polynomial in both $1/\epsilon$ and $n$. For example, $O((1/\epsilon)^2 \cdot n^3)$. 

VII. Randomisation and Rounding
Performance Ratios for Randomised Approximation Algorithms

Approximation Ratio

A randomised algorithm for a problem has approximation ratio \( \rho(n) \), if for any input of size \( n \), the expected cost \( C \) of the returned solution and optimal cost \( C^* \) satisfy:

\[
\max \left( \frac{C}{C^*}, \frac{C^*}{C} \right) \leq \rho(n).
\]

Call such an algorithm randomised \( \rho(n) \)-approximation algorithm.

Approximation Schemes

An approximation scheme is an approximation algorithm, which given any input and \( \epsilon > 0 \), is a \( (1 + \epsilon) \)-approximation algorithm.

- It is a polynomial-time approximation scheme (PTAS) if for any fixed \( \epsilon > 0 \), the runtime is polynomial in \( n \). For example, \( O(n^2/\epsilon) \).
- It is a fully polynomial-time approximation scheme (FPTAS) if the runtime is polynomial in both \( 1/\epsilon \) and \( n \). For example, \( O((1/\epsilon)^2 \cdot n^3) \).
Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover
MAX-3-CNF Satisfiability

- Given: 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.
MAX-3-CNF Satisfiability

- **Given**: 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **Goal**: Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the *satisfiability* problem. Want to compute how “close” the formula to being satisfiable is.
MAX-3-CNF Satisfiability

Given: 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)

Goal: Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the satisfiability problem. Want to compute how “close” the formula to being satisfiable is.

Assume that no literal (including its negation) appears more than once in the same clause.
MAX-3-CNF Satisfiability

- **Given**: 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **Goal**: Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the satisfiability problem. Want to compute how “close” the formula to being satisfiable is.

Example:

\[(x_1 \lor x_3 \lor \overline{x_4}) \land (x_1 \lor \overline{x_3} \lor \overline{x_5}) \land (x_2 \lor \overline{x_4} \lor x_5) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})\]

Assume that no literal (including its negation) appears more than once in the same clause.
MAX-3-CNF Satisfiability

- **Given**: 3-CNf formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **Goal**: Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the satisfiability problem. Want to compute how “close” the formula to being satisfiable is.

Example:

\[
(x_1 \lor x_3 \lor \overline{x_4}) \land (x_1 \lor \overline{x_3} \lor \overline{x_5}) \land (x_2 \lor \overline{x_4} \lor x_5) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})
\]

\(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0\) and \(x_5 = 1\) satisfies 3 (out of 4 clauses)
MAX-3-CNF Satisfiability

- **Given**: 3-CNF formula, e.g.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5}) \land \cdots\)
- **Goal**: Find an assignment of the variables that satisfies as many clauses as possible.

**Example:**

\[(x_1 \lor x_3 \lor \overline{x_4}) \land (x_1 \lor \overline{x_3} \lor \overline{x_5}) \land (x_2 \lor \overline{x_4} \lor x_5) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})\]

- \(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0\) and \(x_5 = 1\) satisfies 3 (out of 4 clauses)

**Idea**: What about assigning each variable independently at random?
Analysis

**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

Proof:
Analysis

**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

**Proof:**
- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1\{\text{clause } i \text{ is satisfied}\}
  \]
Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[ Y_i = 1 \{ \text{clause } i \text{ is satisfied} \} \]
- Since each literal (including its negation) appears at most once in clause \( i \),
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1 \{ \text{clause } i \text{ is satisfied} \}
  \]
- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr [ \text{clause } i \text{ is not satisfied} ] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised 8/7-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[ Y_i = 1 \{ \text{clause } i \text{ is satisfied} \} \]
- Since each literal (including its negation) appears at most once in clause \( i \),
  \[ \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \]
  \[ \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{4} = \frac{3}{4} \]

\[
E[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{3}{4}
\]

Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
\[
E[Y] = E[\sum_{i=1}^{m} Y_i] = \sum_{i=1}^{m} E[Y_i] = \sum_{i=1}^{m} \frac{3}{4} = \frac{3}{4} \cdot m = \frac{3}{4} \cdot m
\]
**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised 8/7-approximation algorithm.

**Proof:**

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]
Analysis

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised 8/7-approximation algorithm.

**Theorem 35.6**

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
Analysis

**Theorem 35.6**
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised 8/7-approximation algorithm.

**Proof:**
- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1\{	ext{clause } i \text{ is satisfied}}
  \]
- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}
  \]
- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbb{E}[Y]
  \]
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1\{\text{clause } i \text{ is satisfied}\}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbb{E}[Y] = \mathbb{E}\left[ \sum_{i=1}^{m} Y_i \right]
  \]
Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbf{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right] = \frac{7}{8} \cdot m.
  \]
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( \frac{8}{7} \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1_{\text{clause } i \text{ is satisfied}}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbb{E}[Y_i]
  \]

  \[\text{Linearity of Expectations}\]
Analysis

Theorem 35.6
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised 8/7-approximation algorithm.

Proof:
- For every clause \( i = 1, 2, \ldots, m \), define a random variable:

\[
Y_i = 1 \{ \text{clause } i \text{ is satisfied} \}
\]

- Since each literal (including its negation) appears at most once in clause \( i \),

\[
\Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
\]

\[
\Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
\]

\[
\Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
\]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,

\[
\mathbb{E}[Y] = \mathbb{E}\left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} \mathbb{E}[Y_i] = \sum_{i=1}^{m} \frac{7}{8}
\]

Linearity of Expectations
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1\{\text{clause } i \text{ is satisfied}\}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad E[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  E[Y] = E\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} E[Y_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} \cdot m.
  \]

(Linearity of Expectations)
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.

Proof:

- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = 1 \{ \text{clause } i \text{ is satisfied} \}
  \]

- Since each literal (including its negation) appears at most once in clause \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- Let \( Y := \sum_{i=1}^{m} Y_i \) be the number of satisfied clauses. Then,
  \[
  \mathbb{E}[Y] = \mathbb{E}\left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} \mathbb{E}[Y_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} \cdot m.
  \]
Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

Proof:

- For every clause $i = 1, 2, \ldots, m$, define a random variable:

$$Y_i = 1\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause $i$,

$$\Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \quad \mathbb{E}[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then,

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbb{E}[Y_i] = \sum_{i=1}^{m} \frac{7}{8} = \frac{7}{8} \cdot m.$$

Linearity of Expectations

maximum number of satisfiable clauses is $m$
Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( \frac{8}{7} \)-approximation algorithm.
Interesting Implications

Theorem 35.6
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$-approximation algorithm.

Corollary
For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.
Interesting Implications

**Theorem 35.6**

Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$-approximation algorithm.

**Corollary**

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $7/8$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$. 
Interesting Implications

Theorem 35.6
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( \frac{8}{7} \)-approximation algorithm.

Corollary
For any instance of MAX-3-CNF, there exists an assignment which satisfies at least \( \frac{7}{8} \) of all clauses.

There is \( \omega \in \Omega \) such that \( Y(\omega) \geq \mathbb{E}[Y] \)

Probabilistic Method: powerful tool to show existence of a non-obvious property.
Interesting Implications

Theorem 35.6
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( 8/7 \)-approximation algorithm.

Corollary
For any instance of MAX-3-CNF, there exists an assignment which satisfies at least \( \frac{7}{8} \) of all clauses.

There is \( \omega \in \Omega \) such that \( Y(\omega) \geq E[Y] \)

Probabilistic Method: powerful tool to show existence of a non-obvious property.

Corollary
Any instance of MAX-3-CNF with at most 7 clauses is satisfiable.
Interesting Implications

**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( 8/7 \)-approximation algorithm.

**Corollary**

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least \( \frac{7}{8} \) of all clauses.

There is \( \omega \in \Omega \) such that \( Y(\omega) \geq E[Y] \)

**Probabilistic Method:** powerful tool to show existence of a non-obvious property.

**Corollary**

Any instance of MAX-3-CNF with at most 7 clauses is satisfiable.

Follows from the previous Corollary.
Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( \frac{8}{7} \)-approximation algorithm.
Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$-approximation algorithm.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$. 

Algorithm:

1: for $j = 1, 2, \ldots, n$

2: Compute $E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1]$

3: Compute $E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 0]$

4: Let $x_j = v_j$ so that the conditional expectation is maximized

5: return the assignment $v_1, v_2, \ldots, v_n$
Expected Approximation Ratio

**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised 8/7-approximation algorithm.

One could prove that the probability to satisfy \((7/8) \cdot m\) clauses is at least \(1/(8m)\).

\[
E[Y] = \frac{1}{2} \cdot E[Y | x_1 = 1] + \frac{1}{2} \cdot E[Y | x_1 = 0].
\]

\(Y\) is defined as in the previous proof.
**Theorem 35.6**

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( 8/7 \)-approximation algorithm.

One could prove that the probability to satisfy \( (7/8) \cdot m \) clauses is at least \( 1/(8m) \).

\[
E[Y] = \frac{1}{2} \cdot E[Y | x_1 = 1] + \frac{1}{2} \cdot E[Y | x_1 = 0].
\]

\( Y \) is defined as in the previous proof.

One of the two conditional expectations is at least \( E[Y] \)!
Expected Approximation Ratio

**Theorem 35.6**

Given an instance of MAX-3-CNF with \(n\) variables \(x_1, x_2, \ldots, x_n\) and \(m\) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \(8/7\)-approximation algorithm.

One could prove that the probability to satisfy \((7/8) \cdot m\) clauses is at least \(1/(8m)\).

\[
E[Y] = \frac{1}{2} \cdot E[Y \mid x_1 = 1] + \frac{1}{2} \cdot E[Y \mid x_1 = 0].
\]

\(Y\) is defined as in the previous proof.

One of the two conditional expectations is at least \(E[Y]\)!

**Algorithm:** Assign \(x_1\) so that the conditional expectation is maximized and recurse.
Expected Approximation Ratio

**Theorem 35.6**
Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised \( 8/7 \)-approximation algorithm.

One could prove that the probability to satisfy \((7/8) \cdot m\) clauses is at least \( 1/(8m) \).

\[
E[Y] = \frac{1}{2} \cdot E[Y | x_1 = 1] + \frac{1}{2} \cdot E[Y | x_1 = 0].
\]

\( Y \) is defined as in the previous proof.

One of the two conditional expectations is at least \( E[Y]! \).

**GREEDY-3-CNF(\( \phi, n, m \))**
1: for \( j = 1, 2, \ldots, n \)
2: Compute \( E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] \)
3: Compute \( E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 0] \)
4: Let \( x_j = v_j \) so that the conditional expectation is maximized
5: return the assignment \( v_1, v_2, \ldots, v_n \)
Analysis of \textsc{Greedy-3-CNF}(\phi, n, m)

\textbf{Theorem}

\textsc{Greedy-3-Cnf}(\phi, n, m) is a polynomial-time 8/7-approximation.
Analysis of \textbf{GREEDY-3-CNF}(\phi, n, m)

This algorithm is deterministic.

\textbf{Theorem}

\textbf{GREEDY-3-CNF}(\phi, n, m) is a polynomial-time 8/7-approximation.
Analysis of \textsc{Greedy-3-CNF}(\(\phi, n, m\))

\textbf{Theorem}

\textsc{Greedy-3-CNF}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.

\textbf{Proof:}

This algorithm is deterministic.
Analysis of **GREEDY-3-CNF**\( (\phi, n, m) \)

**Theorem**

**GREEDY-3-CNF**\( (\phi, n, m) \) is a polynomial-time 8/7-approximation.

**Proof:**
- **Step 1:** polynomial-time algorithm

This algorithm is deterministic.
Analysis of \textsc{Greedy-3-CNF}(\(\phi, n, m\))

This algorithm is deterministic.

\textbf{Theorem}

\textsc{Greedy-3-Cnf}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.

\textbf{Proof:}

- **Step 1:** polynomial-time algorithm
  - In iteration \(j = 1, 2, \ldots, n\), \(Y = Y(\phi)\) averages over \(2^{n-j+1}\) assignments
Analysis of \( \text{GREEDY-3-CNF}(\phi, n, m) \)

**Theorem**

\( \text{GREEDY-3-CNF}(\phi, n, m) \) is a polynomial-time 8/7-approximation.

**Proof:**

- **Step 1:** polynomial-time algorithm
  - In iteration \( j = 1, 2, \ldots, n \), \( Y = Y(\phi) \) averages over \( 2^{n-j+1} \) assignments
  - A smarter way is to use linearity of (conditional) expectations:
    
    \[
    \mathbb{E} \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
    \]
Analysis of $\text{GREEDY-3-CNF}(\phi, n, m)$

This algorithm is deterministic.

**Theorem**

$\text{GREEDY-3-CNF}(\phi, n, m)$ is a polynomial-time $8/7$-approximation.

**Proof:**

- **Step 1:** polynomial-time algorithm
  - In iteration $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over $2^{n-j+1}$ assignments
  - A smarter way is to use linearity of (conditional) expectations:

$$
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_j, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_j, x_j = 1 \right]
$$
Theorem

GREEDY-3-CNF(φ, n, m) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1**: polynomial-time algorithm
  - In iteration $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over $2^{n-j+1}$ assignments
  - A smarter way is to use linearity of (conditional) expectations:

$$
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_j = v_j, x_{j-1} = v_{j-1} \right]
$$

This algorithm is deterministic.
Analysis of \textsc{Greedy-3-CNF}(\(\phi, n, m\))

This algorithm is deterministic.

\textsc{Greedy-3-CNF}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm \(\checkmark\)
  - In iteration \(j = 1, 2, \ldots, n\), \(Y = Y(\phi)\) averages over \(2^{n-j+1}\) assignments
  - A smarter way is to use linearity of (conditional) expectations:

  \[
  \mathbb{E} [ Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 ] = \sum_{i=1}^{m} \mathbb{E} [ Y_i | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 ]
  \]

  computable in \(O(1)\)
Analysis of \textbf{GREEDY-3-CNF}(\(\phi, n, m\))

\begin{center}
\textbf{Theorem}
\end{center}

\textbf{GREEDY-3-CNF}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.

\begin{proof}
\begin{itemize}
  \item \textbf{Step 1}: polynomial-time algorithm \(\checkmark\)
    \begin{itemize}
      \item In iteration \(j = 1, 2, \ldots, n\), \(Y = Y(\phi)\) averages over \(2^{n-j+1}\) assignments
      \item A smarter way is to use linearity of (conditional) expectations:
      \[
      E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
      \]
    \end{itemize}
  \item \textbf{Step 2}: satisfies at least \(7/8 \cdot m\) clauses
\end{itemize}
\end{proof}

\textbf{This algorithm is deterministic.}
Analysis of \textsc{Greedy-3-CNF}(\phi, n, m)

This algorithm is deterministic.

\textbf{Theorem}

\textsc{Greedy-3-Cnf}(\phi, n, m) is a polynomial-time 8/7-approximation.

\textbf{Proof:}

\begin{itemize}
  \item \textbf{Step 1}: polynomial-time algorithm \checkmark
    \begin{itemize}
      \item In iteration \( j = 1, 2, \ldots, n \), \( Y = Y(\phi) \) averages over \( 2^{n-j+1} \) assignments
      \item A smarter way is to use linearity of (conditional) expectations:
      \end{itemize}
      \begin{equation}
      \mathbb{E} \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} \mathbb{E} \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
      \end{equation}
  \item \textbf{Step 2}: satisfies at least \( \frac{7}{8} \cdot m \) clauses
    \begin{itemize}
      \item Due to the greedy choice in each iteration \( j = 1, 2, \ldots, n \),
    \end{itemize}
\end{itemize}
Analysis of \( \text{GREEDY-3-CNF}(\phi, n, m) \)

Theorem

\( \text{GREEDY-3-CNF}(\phi, n, m) \) is a polynomial-time 8/7-approximation.

Proof:

- **Step 1:** polynomial-time algorithm \( \checkmark \)
  - In iteration \( j = 1, 2, \ldots, n \), \( Y = Y(\phi) \) averages over \( 2^{n-j+1} \) assignments
  - A smarter way is to use linearity of (conditional) expectations:

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
\]

- **Step 2:** satisfies at least \( 7/8 \cdot m \) clauses
  - Due to the greedy choice in each iteration \( j = 1, 2, \ldots, n \),
  
\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \right] \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right]
\]
Analysis of \textsc{Greedy-3-CNF}(\(\phi, n, m\))

\begin{theorem}
\textsc{Greedy-3-Cnf}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.
\end{theorem}

\textbf{Proof:}

- \textbf{Step 1:} polynomial-time algorithm \(\checkmark\)
  - In iteration \(j = 1, 2, \ldots, n\), \(Y = Y(\phi)\) averages over \(2^{n-j+1}\) assignments
  - A smarter way is to use linearity of (conditional) expectations:
    \[
    \mathbb{E}[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^{m} \mathbb{E}[Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1]
    \]

- \textbf{Step 2:} satisfies at least \(7/8 \cdot m\) clauses
  - Due to the greedy choice in each iteration \(j = 1, 2, \ldots, n\),
    \[
    \mathbb{E}[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j] \geq \mathbb{E}[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}]
    \geq \mathbb{E}[Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2}]
    \]
Analysis of \( \text{GREEDY-3-CNF}(\phi, n, m) \)

This algorithm is deterministic.

**Theorem**

\( \text{GREEDY-3-CNF}(\phi, n, m) \) is a polynomial-time 8/7-approximation.

**Proof:**

- **Step 1:** polynomial-time algorithm ✓
  - In iteration \( j = 1, 2, \ldots, n \), \( Y = Y(\phi) \) averages over \( 2^{n-j+1} \) assignments
  - A smarter way is to use linearity of (conditional) expectations:

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
\]

- **Step 2:** satisfies at least \( 7/8 \cdot m \) clauses
  - Due to the greedy choice in each iteration \( j = 1, 2, \ldots, n \),

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \right] \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right] \\
\geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \right] \\
\vdots \\
\geq E \left[ Y \right]
\]
Analysis of GREEDY-3-CNF($\phi, n, m$)

This algorithm is deterministic.

**Theorem**

GREEDY-3-CNF($\phi, n, m$) is a polynomial-time $8/7$-approximation.

**Proof:**

- **Step 1:** polynomial-time algorithm ✓
  - In iteration $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over $2^{n-j+1}$ assignments
  - A smarter way is to use linearity of (conditional) expectations:
    \[ E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^{m} E[Y_i | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] \]

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,
    \[ E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j] \geq E[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}] \]
    \[ \geq E[Y | x_1 = v_1, \ldots, x_{j-2} = v_{j-2}] \]
    \[ \vdots \]
    \[ \geq E[Y] = \frac{7}{8} \cdot m. \]
Analysis of \textbf{GREEDY-3-CNF}(\(\phi, n, m\))

This algorithm is deterministic.

\textbf{Theorem}

\textbf{GREEDY-3-CNF}(\(\phi, n, m\)) is a polynomial-time \(8/7\)-approximation.

\textbf{Proof:}

- **Step 1:** polynomial-time algorithm \(\checkmark\)
  - In iteration \(j = 1, 2, \ldots, n\), \(Y = Y(\phi)\) averages over \(2^{n-j+1}\) assignments
  - A smarter way is to use linearity of (conditional) expectations:
    \[
    E[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^{m} E[Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1]
    \]

- **Step 2:** satisfies at least \(7/8 \cdot m\) clauses \(\checkmark\)
  - Due to the greedy choice in each iteration \(j = 1, 2, \ldots, n\),
    \[
    E[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j] \geq E[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}] \geq E[Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2}] \geq \ldots \geq E[Y] = \frac{7}{8} \cdot m.
    \]
Analysis of \textsc{Greedy-3-CNF}(\phi, n, m)

\textbf{Theorem}

\textsc{Greedy-3-CNF}(\phi, n, m) is a polynomial-time $8/7$-approximation.

\textbf{Proof:}

- \textbf{Step 1:} polynomial-time algorithm \checkmark
  - In iteration $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over $2^{n-j+1}$ assignments
  - A smarter way is to use linearity of (conditional) expectations:

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right] = \sum_{i=1}^{m} E \left[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \right]
\]

- \textbf{Step 2:} satisfies at least $7/8 \cdot m$ clauses \checkmark
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \right] \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right] \\
\geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \right] \\
\vdots \\
\geq E \left[ Y \right] = \frac{7}{8} \cdot m.
\]
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$(x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (x_1 \lor x_2 \lor x_4) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_4}) \land (x_1 \lor x_3 \lor x_4) \land (\overline{x_1} \lor \overline{x_3} \lor \overline{x_4}) \land (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$$
Run of **GREEDY-3-CNF**$(\varphi, n, m)$

$$(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (x_1 \lor \overline{x_2} \lor \overline{x_3}) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_3} \lor \overline{x_4}) \land (x_1 \lor \overline{x_3} \lor \overline{x_4})$$

VII. Randomisation and Rounding

MAX-3-CNF
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$(x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \overline{x}_3 \lor \overline{x}_4)$$
Run of **GREEDY-3-CNФ(ϕ, n, m)**

\[(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_3}) \land (x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_4) \land (x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_3 \lor x_4) \land (x_2 \lor \overline{x_3} \lor \overline{x_4})\]

\[\text{VII. Randomisation and Rounding MAX-3-CNФ} \]
Run of GREEDY-3-CNF($\varphi, n, m$)

$$(x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_4 \lor \overline{x_5} \lor x_6) \land (x_7 \lor x_8 \lor x_9) \land (x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6) \land (x_7 \lor x_8 \lor x_9) \land (x_1 \lor x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6)$$

VII. Randomisation and Rounding

MAX-3-CNF
Run of $\text{GREEDY-3-CNF}(\varphi, n, m)$

$$1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor x_3) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$$
Run of Greedy-3-CNF($\varphi, n, m$)

1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor \overline{x_3}) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})

VII. Randomisation and Rounding MAX-3-CNF
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor \overline{x_3}) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$$

VII. Randomisation and Rounding

MAX-3-CNF
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor x_3) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$$
Run of \textsc{Greedy-3-CNF}(\varphi, n, m)

\[1 \land 1 \land 1 \land (\overline{x}_3 \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x}_3 \lor \overline{x}_4)\]
Run of \textbf{GREEDY-3-CNF}(\(\varphi, n, m\))

\[1 \land 1 \land 1 \land (\overline{x}_3 \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x}_3 \lor \overline{x}_4)\]
Run of \textbf{GREEDY-3-CNF}(\(\varphi, n, m\))

\[1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x_3} \lor \overline{x_4})\]
Run of \textbf{GREEDY-3-CNF}(\(\varphi, n, m\))

\[
1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x_3} \lor \overline{x_4})
\]
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$$
Run of **GREEDY-3-CNF** ($\varphi, n, m$)

$$1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$$
Run of GREEDY-3-CNF(\(\varphi, n, m\))

\[ 1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1 \]

\[
\begin{array}{c}
\text{\(x_1 = 0\)} \\
8.625 \\
\text{\(x_2 = 0\)} \\
\text{\(x_3 = 0\)} \\
000? \\
0001 \\
0010 \\
0011 \\
0100 \\
0101 \\
0110 \\
0111 \\
1000 \\
1001 \\
1010 \\
1011 \\
1100 \\
1101 \\
1110 \\
1111 \\
\end{array}
\]

\[
\begin{array}{c}
\text{\(x_1 = 1\)} \\
8.75 \\
\text{\(x_2 = 1\)} \\
\text{\(x_3 = 1\)} \\
100? \\
101? \\
110? \\
111? \\
\end{array}
\]

\[\text{Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.}\]
Run of **GREEDY-3-CNF**($\varphi, n, m$)

$$1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$$

VII. Randomisation and Rounding

**MAX-3-CNF**
Run of **Greedy-3-CNF** ($\varphi, n, m$)

$$1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$$
Run of GREEDY-3-CNF(\(\varphi, n, m\))

\[
1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1
\]
Run of **GREEDY-3-CNF**($\varphi, n, m$)

\[ 1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1 \]

### VII. Randomisation and Rounding

**MAX-3-CNF**

 Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.
Theorem 35.6

Given an instance of MAX-3-CNF with \( n \) variables \( x_1, x_2, \ldots, x_n \) and \( m \) clauses, the randomised algorithm that sets each variable independently at random is a randomised \( 8/7 \)-approximation algorithm.
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

**Theorem 35.6**

$\text{GREEDY-3-CNF}(\phi, n, m)$ is a polynomial-time $8/7$-approximation.
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

**Theorem 35.6**

GREEDY-3-CNF($\phi$, $n$, $m$) is a polynomial-time $8/7$-approximation.

**Theorem (Hastad’97)**

For any $\epsilon > 0$, there is no polynomial time $8/7 - \epsilon$ approximation algorithm of MAX3-SAT unless P=NP.
MAX-3-CNF: Concluding Remarks

Theorem 35.6
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

Theorem
$\text{GREEDY-3-CNF}(\phi, n, m)$ is a polynomial-time $8/7$-approximation.

Theorem (Hastad’97)
For any $\epsilon > 0$, there is no polynomial time $8/7 - \epsilon$ approximation algorithm of MAX3-SAT unless $P=NP$.

Essentially there is nothing smarter than just guessing!
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover
The Weighted Vertex-Cover Problem

- **Given**: Undirected, vertex-weighted graph $G = (V, E)$
- **Goal**: Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.

Vertex Cover Problem

Applications:
- Every edge forms a task, and every vertex represents a person/machine which can execute that task
- Weight of a vertex could be salary of a person
- Perform all tasks with the minimal amount of resources
The Weighted Vertex-Cover Problem

Given: Undirected, vertex-weighted graph $G = (V, E)$

Goal: Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.
The Weighted Vertex-Cover Problem

Given: Undirected, vertex-weighted graph $G = (V, E)$

Goal: Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.

Applications:
Every edge forms a task, and every vertex represents a person/machine which can execute that task.
Weight of a vertex could be salary of a person.
Perform all tasks with the minimal amount of resources.

VII. Randomisation and Rounding
Weighted Vertex Cover
The Weighted Vertex-Cover Problem

Given: Undirected, vertex-weighted graph \( G = (V, E) \)

Goal: Find a minimum-weight subset \( V' \subseteq V \) such that if \( (u, v) \in E(G) \), then \( u \in V' \) or \( v \in V' \).

This is (still) an NP-hard problem.
The **Weighted Vertex-Cover Problem**

- **Given:** Undirected, vertex-weighted graph $G = (V, E)$
- **Goal:** Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.

This is (still) an NP-hard problem.

**Applications:**

Every edge forms a task, and every vertex represents a person/machine which can execute that task. The weight of a vertex could be the salary of a person. Perform all tasks with the minimal amount of resources.
The Weighted Vertex-Cover Problem

Given: Undirected, vertex-weighted graph \( G = (V, E) \)

Goal: Find a minimum-weight subset \( V' \subseteq V \) such that if \( (u, v) \in E(G) \), then \( u \in V' \) or \( v \in V' \).

Applications:
- Every edge forms a task, and every vertex represents a person/machine which can execute that task

This is (still) an NP-hard problem.
The **Weighted Vertex-Cover Problem**

**Vertex Cover Problem**

- **Given:** Undirected, vertex-weighted graph \( G = (V, E) \)
- **Goal:** Find a minimum-weight subset \( V' \subseteq V \) such that if \( (u, v) \in E(G) \), then \( u \in V' \) or \( v \in V' \).

This is (still) an NP-hard problem.

**Applications:**

- Every edge forms a task, and every vertex represents a person/machine which can execute that task
- Weight of a vertex could be salary of a person
The **Weighted Vertex-Cover Problem**

Given: Undirected, vertex-weighted graph \( G = (V, E) \)

Goal: Find a minimum-weight subset \( V' \subseteq V \) such that if \((u, v) \in E(G)\), then \( u \in V' \) or \( v \in V' \).

This is (still) an NP-hard problem.

**Applications:**
- Every edge forms a task, and every vertex represents a person/machine which can execute that task
- Weight of a vertex could be salary of a person
- Perform all tasks with the minimal amount of resources
The Greedy Approach from (Unweighted) Vertex Cover

APPROX-VERTEX-COVER \((G)\)

1. \(C = \emptyset\)
2. \(E' = G.E\)
3. while \(E' \neq \emptyset\)
   4. let \((u, v)\) be an arbitrary edge of \(E'\)
   5. \(C = C \cup \{u, v\}\)
   6. remove from \(E'\) every edge incident on either \(u\) or \(v\)
4. return \(C\)
The Greedy Approach from (Unweighted) Vertex Cover

**APPROX-VERTEX-COVER**(*G*)

1. \( C = \emptyset \)
2. \( E' = G.E \)
3. \( \text{while } E' \neq \emptyset \)
   4. let \((u, v)\) be an arbitrary edge of \( E' \)
   5. \( C = C \cup \{u, v\} \)
   6. remove from \( E' \) every edge incident on either \( u \) or \( v \)
4. return \( C \)
The Greedy Approach from (Unweighted) Vertex Cover

APPROX-VERTEX-COVER($G$)
1  $C = \emptyset$
2  $E' = G.E$
3  while $E' \neq \emptyset$
4    let $(u, v)$ be an arbitrary edge of $E'$
5    $C = C \cup \{u, v\}$
6    remove from $E'$ every edge incident on either $u$ or $v$
7  return $C$

Computed solution has weight 101

Optimal solution has weight 4
The Greedy Approach from (Unweighted) Vertex Cover

**APPROX-VERTEX-COVER**(*G*)

1. *C* = ∅
2. *E*' = *G*. *E*
3. while *E*' ≠ ∅
   4. let (*u*, *v*) be an arbitrary edge of *E'*
   5. *C* = *C* ∪ {*u*, *v*}
   6. remove from *E'* every edge incident on either *u* or *v*
4. return *C*

**Figure 35.1** illustrates how **APPROX-VERTEX-COVER** operates on an example graph. The variable *C* contains the vertex cover being constructed. Line 1 initializes *C* to the empty set. Line 2 sets *E*' to be a copy of the edge set *G*: *E* of the graph. The loop of lines 3–6 repeatedly picks an edge (*u*, *v*) from *E*' and adds it to *C*. Line 6 removes from *E*' every edge incident on either *u* or *v*. The computed solution has weight 100, while the optimal solution has weight 4.

Optimal solution has weight 4
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

minimize \( \sum_{v \in V} w(v) x(v) \)

subject to \( x(u) + x(v) \geq 1 \) for each \((u, v) \in E\)

\( x(v) \in \{0, 1\} \) for each \( v \in V \)

0-1 Integer Program

minimize \( \sum_{v \in V} w(v) x(v) \)

subject to \( x(u) + x(v) \geq 1 \) for each \((u, v) \in E\)

\( x(v) \in [0, 1] \) for each \( v \in V \)

Linear Program

Optimum is a lower bound on the optimal weight of a minimum weight-cover.

Rounding Rule: if \( x(v) \geq 1/2 \) then round up, otherwise round down.
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

\[
\text{minimize } \sum_{v \in V} w(v)x(v) \\
\text{subject to } x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\
x(v) \in \{0, 1\} \quad \text{for each } v \in V
\]
Invoking an (Integer) Linear Program

**Idea:** Round the solution of an associated linear program.

---

**0-1 Integer Program**

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w(v)x(v) \\
\text{subject to} & \quad x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\
& \quad x(v) \in \{0, 1\} \quad \text{for each } v \in V
\end{align*}
\]

**Linear Program**

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w(v)x(v) \\
\text{subject to} & \quad x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\
& \quad x(v) \in [0, 1] \quad \text{for each } v \in V
\end{align*}
\]
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w(v)x(v) \\
\text{subject to} & \quad x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\
& \quad x(v) \in \{0, 1\} \quad \text{for each } v \in V
\end{align*}
\]

optimum is a lower bound on the optimal weight of a minimum weight-cover.

Linear Program

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \in V} w(v)x(v) \\
\text{subject to} & \quad x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\
& \quad x(v) \in [0, 1] \quad \text{for each } v \in V
\end{align*}
\]
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

minimize \[ \sum_{v \in V} w(v)x(v) \]
subject to \[ x(u) + x(v) \geq 1 \] for each \((u, v) \in E\)
\[ x(v) \in \{0, 1\} \] for each \(v \in V\)

Linear Program

minimize \[ \sum_{v \in V} w(v)x(v) \]
subject to \[ x(u) + x(v) \geq 1 \] for each \((u, v) \in E\)
\[ x(v) \in [0, 1] \] for each \(v \in V\)

Rounding Rule: if \(x(v) \geq 1/2\) then round up, otherwise round down.
The Algorithm

\textsc{Approx-Min-Weight-VC}(G, w)

1. \(C = \emptyset\)
2. compute \(\bar{x}\), an optimal solution to the linear program
3. for each \(v \in V\)
4. \hspace{1em} if \(\bar{x}(v) \geq 1/2\)
5. \hspace{2em} \(C = C \cup \{v\}\)
6. return \(C\)
The Algorithm

**APPROX-MIN-WEIGHT-VC** \((G, w)\)

1. \(C = \emptyset\)
2. compute \(\bar{x}\), an optimal solution to the linear program
3. **for** each \(v \in V\)
4. \hspace{1em} **if** \(\bar{x}(v) \geq 1/2\)
5. \hspace{2em} \(C = C \cup \{v\}\)
6. return \(C\)

**Theorem 35.7**

**APPROX-MIN-WEIGHT-VC** is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.
The Algorithm

\textbf{APPROX-MIN-WEIGHT-VC} \((G, w)\)

1 \hspace{1em} \(C = \emptyset\)
2 \hspace{1em} compute \(\bar{x}\), an optimal solution to the linear program
3 \hspace{1em} \textbf{for} each \(v \in V\)
4 \hspace{2em} \textbf{if} \(\bar{x}(v) \geq 1/2\)
5 \hspace{2em} \(C = C \cup \{v\}\)
6 \hspace{1em} \textbf{return} \(C\)

\textbf{Theorem 35.7}

\textbf{APPROX-MIN-WEIGHT-VC} is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

is polynomial-time because we can solve the linear program in polynomial time
**Example of APPROX-MIN-WEIGHT-VC**

\[ \bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \quad \bar{x}(d) = 1, \quad \bar{x}(c) = 0 \]

Fractional solution of LP with weight \( = 5.5 \)

Randomisation and Rounding

Weighted Vertex Cover
Example of \textsc{Approx-Min-Weight-VC}

\[ \bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0 \]

\[ x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0 \]

fractional solution of LP with weight = 5.5

rounded solution of LP with weight = 10
Example of APPROX-MIN-WEIGHT-VC

\[ \bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0 \]

\[ x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0 \]

fractional solution of LP with weight = 5.5

density solution of LP with weight = 10

optimal solution with weight = 6
Approximation Ratio

Proof (Approximation Ratio is 2):
Approximation Ratio

Proof (Approximation Ratio is 2):

\[ C^* \] be an optimal solution to the minimum-weight vertex cover problem
\[ z^* \] be the value of an optimal solution to the linear program, so
\[ z^* \leq w(C^*) \]

Step 1: The computed set \( C \) covers all vertices:
Consider any edge \((u, v) \in E\) which imposes the constraint
\[ x(u) + x(v) \geq 1 \]
\[ \implies \text{at least one of } x(u) \text{ and } x(v) \text{ is at least } 1/2 \]
\[ C \] covers edge \((u, v)\).

Step 2: The computed set \( C \) satisfies
\[ w(C) \leq 2z^* \]

\[ w(C^*) \geq z^* = \sum_{v \in V} w(v)x(v) \geq \sum_{v \in V} x(v) \geq 1/2 w(v) \cdot 1/2 = 1/2 w(C). \]
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem.

```
Let $z^*$ be the value of an optimal solution to the linear program, so $z^* \leq w(C^*)$

Step 1: The computed set $C$ covers all vertices:
Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$ ⇒ at least one of $x(u)$ and $x(v)$ is at least $1/2$ ⇒ $C$ covers edge $(u, v)$.

Step 2: The computed set $C$ satisfies $w(C) \leq 2z^*$:
$w(C^*) \geq z^* = \sum_{v \in V} w(v) x(v) \geq \sum_{v \in V} x(v) \geq 1/2 w(v) \cdot 1/2 = 1/4 w(C)$.
```
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem.
- Let $z^*$ be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

**Step 1:** The computed set $C$ covers all vertices:

Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1 \Rightarrow$ at least one of $x(u)$ and $x(v)$ is at least $1/2$ $\Rightarrow C$ covers edge $(u, v)$.

**Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v)x(v) \geq \sum_{v \in V} x(v) \cdot 1/2w(v) = \frac{1}{2}w(C).$$
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let \( C^* \) be an optimal solution to the minimum-weight vertex cover problem.
- Let \( z^* \) be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1:** The computed set \( C \) covers all vertices:

\[
\begin{array}{cccccc}
\text{a} & 4 & \text{b} & 3 & \text{c} & 3 \\
\text{e} & 2 & \text{d} & 1 & \text{e} & 2 \\
\end{array}
\]

Rounding

\[
\begin{array}{cccccc}
\text{a} & 4 & \text{b} & 3 & \text{c} & 3 \\
\text{e} & 2 & \text{d} & 1 & \text{e} & 2 \\
\end{array}
\]

\[
\begin{array}{cccccc}
\text{a} & 4 & \text{b} & 3 & \text{c} & 3 \\
\text{e} & 2 & \text{d} & 1 & \text{e} & 2 \\
\end{array}
\]
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

**Step 1:** The computed set $C$ covers all vertices:
- Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let \( C^* \) be an optimal solution to the minimum-weight vertex cover problem
- Let \( z^* \) be the value of an optimal solution to the linear program, so
  \[
  z^* \leq w(C^*)
  \]

- **Step 1:** The computed set \( C \) covers all vertices:
  - Consider any edge \((u, v) \in E\) which imposes the constraint \( x(u) + x(v) \geq 1\)
  - \( \Rightarrow \) at least one of \( \bar{x}(u) \) and \( \bar{x}(v) \) is at least \( 1/2 \)
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1**: The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
  - $\Rightarrow$ at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2$ $\Rightarrow$ $C$ covers edge $(u, v)$

---

VII. Randomisation and Rounding

Weighted Vertex Cover
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1**: The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
  \[ \Rightarrow \text{ at least one of } x(u) \text{ and } x(v) \text{ is at least } 1/2 \Rightarrow C \text{ covers edge } (u, v) 

- **Step 2**: The computed set $C$ satisfies $w(C) \leq 2z^*$: 

![Diagram showing the vertex cover process](image)
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so

\[ z^* \leq w(C^*) \]

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
  \[ \Rightarrow \] at least one of $x(u)$ and $x(v)$ is at least $1/2 \Rightarrow C$ covers edge $(u, v)$
- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:

\[ Z^* \]
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1**: The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
    \[ \Rightarrow \] at least one of $x(u)$ and $x(v)$ is at least $1/2$ \( \Rightarrow \) $C$ covers edge $(u, v)$

- **Step 2**: The computed set $C$ satisfies $w(C) \leq 2z^*$:
  \[ w(C^*) \geq z^* \]
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$  
    \[ \Rightarrow \text{ at least one of } \overline{x}(u) \text{ and } \overline{x}(v) \text{ is at least } \frac{1}{2} \Rightarrow C \text{ covers edge } (u, v) \]

- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:
  \[ w(C^*) \geq z^* = \sum_{v \in V} w(v)\overline{x}(v) \]
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the **minimum-weight vertex cover problem**
- Let $z^*$ be the value of an optimal solution to the linear program, so

  $$z^* \leq w(C^*)$$

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$  
    $\Rightarrow$ at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge $(u, v)$

- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:

  $$w(C^*) \geq z^* = \sum_{v \in V} w(v)\bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2}$$
Approximation Ratio

Proof (Approximation Ratio is 2):
- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem.
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$ \implies at least one of $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2$ \implies $C$ covers edge $(u, v)$.

- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:
  \[
  w(C^*) \geq z^* = \sum_{v \in V} w(v)\overline{x}(v) \geq \sum_{v \in V: \overline{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2}w(C).
  \]
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem.
- Let $z^*$ be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$.
  - $\Rightarrow$ at least one of $x(u)$ and $x(v)$ is at least $1/2$ $\Rightarrow$ $C$ covers edge $(u, v)$.

- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v)\bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C).$$

VII. Randomisation and Rounding
Weighted Vertex Cover
Approximation Ratio

Proof (Approximation Ratio is 2):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem
- Let $z^*$ be the value of an optimal solution to the linear program, so
  \[ z^* \leq w(C^*) \]

- **Step 1:** The computed set $C$ covers all vertices:
  - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$ \(\Rightarrow\) at least one of $x(u)$ and $x(v)$ is at least $1/2$ \(\Rightarrow\) $C$ covers edge $(u, v)$

- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:
  \[ w(C^*) \geq z^* = \sum_{v \in V} w(v)\bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C). \]
Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover
The **Weighted Set-Covering Problem**

---

**Set Cover Problem**

- **Given:** set $X$ and a family of subsets $\mathcal{F}$, and a cost function $c : \mathcal{F} \to \mathbb{R}^+$
- **Goal:** Find a **minimum-cost** subset $C \subseteq \mathcal{F}$

\[
\text{s.t. } X = \bigcup_{S \in C} S.
\]
The **Weighted Set-Covering Problem**

**Set Cover Problem**
- **Given:** set $X$ and a family of subsets $\mathcal{F}$, and a cost function $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a minimum-cost subset $C \subseteq \mathcal{F}$

\[
\text{s.t. } X = \bigcup_{S \in C} S.
\]

Sum over the costs of all sets in $C$
The **Weighted Set-Covering Problem**

- **Given:** set $X$ and a family of subsets $\mathcal{F}$, and a cost function $c : \mathcal{F} \rightarrow \mathbb{R}^+$
- **Goal:** Find a minimum-cost subset $C \subseteq \mathcal{F}$

$$X = \bigcup_{S \in C} S.$$
The Weighted Set-Covering Problem

Given: set $X$ and a family of subsets $\mathcal{F}$, and a cost function $c : \mathcal{F} \rightarrow \mathbb{R}^+$

Goal: Find a minimum-cost subset $C \subseteq \mathcal{F}$

such that $X = \bigcup_{S \in C} S$.

Remarks:
- generalisation of the weighted vertex-cover problem
- models resource allocation problems
Setting up an Integer Program

minimize \( \sum_{S \in F} c(S) y(S) \)

subject to \( \sum_{S \in F}: x \in S y(S) \geq 1 \) for each \( x \in X \)

\( y(S) \in \{0, 1\} \) for each \( S \in F \)

0-1 Integer Program

minimize \( \sum_{S \in F} c(S) y(S) \)

subject to \( \sum_{S \in F}: x \in S y(S) \geq 1 \) for each \( x \in X \)

\( y(S) \in [0, 1] \) for each \( S \in F \)

Linear Program
Setting up an Integer Program

0-1 Integer Program

minimize \( \sum_{S \in \mathcal{F}} c(S)y(S) \)

subject to \( \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \) for each \( x \in X \)

\( y(S) \in \{0, 1\} \) for each \( S \in \mathcal{F} \)
Setting up an Integer Program

0-1 Integer Program

minimize \( \sum_{S \in \mathcal{F}} c(S)y(S) \)

subject to \( \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \) for each \( x \in X \)

\( y(S) \in \{0, 1\} \) for each \( S \in \mathcal{F} \)

Linear Program

minimize \( \sum_{S \in \mathcal{F}} c(S)y(S) \)

subject to \( \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \) for each \( x \in X \)

\( y(S) \in [0, 1] \) for each \( S \in \mathcal{F} \)
The strategy employed for Vertex-Cover would take all 6 sets! Even worse: If all $y$'s were below $\frac{1}{2}$, we would not even return a valid cover!
Cost equals 8.5. The strategy employed for Vertex-Cover would take all 6 sets! Even worse: If all $y$’s were below $1/2$, we would not even return a valid cover!
The strategy employed for Vertex-Cover would take all 6 sets!

Even worse: If all $y$'s were below $1/2$, we would not even return a valid cover!

Cost equals 8.5
The strategy employed for Vertex-Cover would take all 6 sets!

Cost equals 8.5
The strategy employed for Vertex-Cover would take all 6 sets!

Even worse: If all \( y \)'s were below \( 1/2 \), we would not even return a valid cover!

Cost equals 8.5
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(\cdot)$</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y(\cdot)$-values as probabilities for picking the respective set.

The expected cost satisfies

$$E\left[c(C) \right] = \sum_{S \in F} c(S) \cdot y(S)$$

The probability that an element $x \in X$ is covered satisfies

$$\Pr\left[ x \in \bigcup_{S \in C} S \right] \geq 1 - \frac{1}{e}.$$
### Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(.)$:</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

**Idea:** Interpret the $y$-values as probabilities for picking the respective set.
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$:</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(.)$:</td>
<td>$1/2$</td>
<td>$1/2$</td>
<td>$1/2$</td>
<td>$1/2$</td>
<td>1</td>
<td>$1/2$</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y$-values as probabilities for picking the respective set.

- Let $C \subseteq \mathcal{F}$ be a random set with each set $S$ being included independently with probability $y(S)$.
- More precisely, if $y$ denotes the optimal solution of the LP, then we compute an integral solution $\bar{y}$ by:

$$\bar{y}(S) = \begin{cases} 
1 & \text{with probability } y(S) \\
0 & \text{otherwise.}
\end{cases}$$

for all $S \in \mathcal{F}$. 

---

VII. Randomisation and Rounding
Weighted Set Cover

23
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(.)$</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y$-values as probabilities for picking the respective set.

- Let $C \subseteq \mathcal{F}$ be a random set with each set $S$ being included independently with probability $y(S)$.
- More precisely, if $y$ denotes the optimal solution of the LP, then we compute an integral solution $\tilde{y}$ by:

$$\tilde{y}(S) = \begin{cases} 
1 & \text{with probability } y(S) \\
0 & \text{otherwise.} 
\end{cases}$$

- Therefore, $E[\tilde{y}(S)] = y(S)$. 
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(.)$:</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y$-values as probabilities for picking the respective set.

Lemma
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
<th>S₅</th>
<th>S₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>y(·)</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

**Idea:** Interpret the $y$-values as probabilities for picking the respective set.

**Lemma**
- The expected cost satisfies

$$
E \left[ c(C) \right] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)
$$
Randomised Rounding

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$y(.)$</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $y$-values as probabilities for picking the respective set.

Lemma

- The expected cost satisfies
  \[
  E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)
  \]

- The probability that an element $x \in X$ is covered satisfies
  \[
  \Pr \left[ x \in \bigcup_{S \in C} S \right] \geq 1 - \frac{1}{e}.
  \]
Proof of Lemma

Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\mathbb{P}[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.
Proof of Lemma

Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  $$\mathbb{E}[c(C)]$$
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\mathbb{P}[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$
  
  \[
  \mathbb{E}[c(C)] = \mathbb{E} \left[ \sum_{S \in C} c(S) \right]
  \]
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

$$E[c(C)] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right]$$
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[ c(C) ] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $Pr[ x \in \bigcup_{S \in C} S ] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  $E[ c(C) ] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right]$

  $= \sum_{S \in \mathcal{F}} Pr[ S \in C ] \cdot c(S)$
Proof of Lemma

Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E \left[ c(C) \right] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr \left[ x \in \bigcup_{S \in C} S \right] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  $$E \left[ c(C) \right] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr \left[ S \in C \right] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S).$$
Proof of Lemma

Lemma

Let $C \subseteq F$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in F} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  $$E[c(C)] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in F} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in F} \Pr[S \in C] \cdot c(S) = \sum_{S \in F} y(S) \cdot c(S).$$
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$ \checkmark
  \[
  E[c(C)] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S).
  \]

- **Step 2**: The probability for an element to be (not) covered
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$
  \[ E[c(C)] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S). \]

- **Step 2**: The probability for an element to be (not) covered
  \[ \Pr[x \notin \bigcup_{S \in C} S] \]
Proof of Lemma

Let $C \subseteq F$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in F} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  $E[c(C)] = E\left[ \sum_{S \in C} c(S) \right] = E\left[ \sum_{S \in F} 1_{S \in C} \cdot c(S) \right]$

  $= \sum_{S \in F} \Pr[S \in C] \cdot c(S) = \sum_{S \in F} y(S) \cdot c(S)$.

- **Step 2**: The probability for an element to be (not) covered

  $\Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in F : x \in S} \Pr[S \notin C]$
Proof of Lemma

**Lemma**

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

**Proof:**

- **Step 1:** The expected cost of the random set $\mathcal{C}$

  $$E[c(\mathcal{C})] = E \left[ \sum_{S \in \mathcal{C}} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in \mathcal{C}} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S).$$

- **Step 2:** The probability for an element to be (not) covered

  $$\Pr[x \notin \bigcup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F} : x \in S} \Pr[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S))$$
Proof of Lemma

Let \( C \subseteq F \) be a random subset with each set \( S \) being included independently with probability \( y(S) \).

- The expected cost satisfies \( \mathbb{E}[c(C)] = \sum_{S \in F} c(S) \cdot y(S) \).
- The probability that \( x \) is covered satisfies \( \Pr[x \in \cup_{S \in C} S] \geq 1 - \frac{1}{e} \).

Proof:

- **Step 1:** The expected cost of the random set \( C \)

  \[
  \mathbb{E}[c(C)] = \mathbb{E}\left[ \sum_{S \in C} c(S) \right] = \mathbb{E}\left[ \sum_{S \in F} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in F} \Pr[S \in C] \cdot c(S) = \sum_{S \in F} y(S) \cdot c(S).
  \]

- **Step 2:** The probability for an element to be (not) covered

  \[
  \Pr[x \notin \cup_{S \in C} S] = \prod_{S \in F : x \in S} \Pr[S \notin C] = \prod_{S \in F : x \in S} (1 - y(S)).
  \]

\[1 + x \leq e^x \text{ for any } x \in \mathbb{R}\]
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set $C$:
  
  $E[c(C)] = E \left[ \sum_{S \in C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} Pr[S \in C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S)$.

- **Step 2:** The probability for an element to be (not) covered:
  
  $Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in \mathcal{F} : x \in S} Pr[S \notin C] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S)) \leq \prod_{S \in \mathcal{F} : x \in S} e^{-y(S)}$

  $1 + x \leq e^x$ for any $x \in \mathbb{R}$
Proof of Lemma

Let \( C \subseteq \mathcal{F} \) be a random subset with each set \( S \) being included independently with probability \( y(S) \).

- The expected cost satisfies \( \mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \).
- The probability that \( x \) is covered satisfies \( \Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e} \).

Proof:

- **Step 1**: The expected cost of the random set \( C \)
  \[
  \mathbb{E}[c(C)] = \mathbb{E}\left[ \sum_{S \in C} c(S) \right] = \mathbb{E}\left[ \sum_{S \in \mathcal{F}} \mathbf{1}_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S).
  \]

- **Step 2**: The probability for an element to be (not) covered
  \[
  \Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in \mathcal{F}: x \in S} \Pr[S \notin C] = \prod_{S \in \mathcal{F}: x \in S} (1 - y(S)) \leq \prod_{S \in \mathcal{F}: x \in S} e^{-y(S)} = e^{-\sum_{S \in \mathcal{F}: x \in S} y(S)}.
  \]

1 + \( x \) \leq e\(^x\) for any \( x \in \mathbb{R} \)
Proof of Lemma

Let \( C \subseteq F \) be a random subset with each set \( S \) being included independently with probability \( y(S) \).

- The expected cost satisfies \( \mathbb{E}[c(C)] = \sum_{S \in F} c(S) \cdot y(S) \).
- The probability that \( x \) is covered satisfies \( \Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e} \).

Proof:

- **Step 1**: The expected cost of the random set \( C \)

\[
\mathbb{E}[c(C)] = \mathbb{E}
\left[
\sum_{S \in C} c(S)
\right]
= \mathbb{E}
\left[
\sum_{S \in F} 1_{S \in C} \cdot c(S)
\right]
= \sum_{S \in F} \Pr[S \in C] \cdot c(S)
= \sum_{S \in F} y(S) \cdot c(S).
\]

- **Step 2**: The probability for an element to be (not) covered

\[
\Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in F: x \in S} \Pr[S \notin C] = \prod_{S \in F: x \in S} (1 - y(S))
\leq \prod_{S \in F: x \in S} e^{-y(S)}\quad\text{\(y\) solves the LP!}
\]

\[
1 + x \leq e^x \text{ for any } x \in \mathbb{R}
\]

\[
= e^{-\sum_{S \in F: x \in S} y(S)}
\]
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1**: The expected cost of the random set $C$

  \[ E[c(C)] = E \left[ \sum_{S \subseteq C} c(S) \right] = E \left[ \sum_{S \in \mathcal{F}} 1_{S \subseteq C} \cdot c(S) \right] 
  \]

  \[ = \sum_{S \in \mathcal{F}} \Pr[S \subseteq C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S). \]

- **Step 2**: The probability for an element to be (not) covered

  \[ \Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in \mathcal{F} : x \in S} \Pr[S \notin C] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S)) \]

  \[ \leq \prod_{S \in \mathcal{F} : x \in S} e^{-y(S)} \]

  \[ = e^{-\sum_{S \in \mathcal{F} : x \in S} y(S)} \leq e^{-1} \]

  $1 + x \leq e^x$ for any $x \in \mathbb{R}$

  $y$ solves the LP!
Proof of Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set $C$.

  \[ \mathbb{E}[c(C)] = \mathbb{E}\left[ \sum_{S \in C} c(S) \right] = \mathbb{E}\left[ \sum_{S \in \mathcal{F}} 1_{S \in C} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in C] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S). \]

- **Step 2:** The probability for an element to be (not) covered.

  \[ \Pr[x \not\in \bigcup_{S \in C} S] = \prod_{S \in \mathcal{F} : x \in S} \Pr[S \not\in C] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S)) \leq \prod_{S \in \mathcal{F} : x \in S} e^{-y(S)} = e^{-\sum_{S \in \mathcal{F} : x \in S} y(S)} \leq e^{-1} \]

\[1 + x \leq e^x \text{ for any } x \in \mathbb{R}\]
Proof of Lemma

Let $\mathcal{C} \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(\mathcal{C})] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

Proof:

- **Step 1:** The expected cost of the random set $\mathcal{C}$

  $\mathbb{E}[c(\mathcal{C})] = \mathbb{E} \left[ \sum_{S \in \mathcal{C}} c(S) \right] = \mathbb{E} \left[ \sum_{S \in \mathcal{F}} 1_{S \in \mathcal{C}} \cdot c(S) \right] = \sum_{S \in \mathcal{F}} \Pr[S \in \mathcal{C}] \cdot c(S) = \sum_{S \in \mathcal{F}} y(S) \cdot c(S)$.

- **Step 2:** The probability for an element to be (not) covered

  $\Pr[x \notin \bigcup_{S \in \mathcal{C}} S] = \prod_{S \in \mathcal{F} : x \in S} \Pr[S \notin \mathcal{C}] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S)) \leq \prod_{S \in \mathcal{F} : x \in S} e^{-y(S)} = e^{-\sum_{S \in \mathcal{F} : x \in S} y(S)} \leq e^{-1}$

  $1 + x \leq e^x$ for any $x \in \mathbb{R}$

  $y$ solves the LP!
The Final Step

Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E \left[ c(C) \right] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr \left[ x \in \bigcup_{S \in C} S \right] \geq 1 - \frac{1}{e}$.
The Final Step

**Lemma**

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $\mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

**Problem**: Need to make sure that every element is covered!
The Final Step

Lemma

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

Problem: Need to make sure that every element is covered!

Idea: Amplify this probability by taking the union of $\Omega(\log n)$ random sets $C$. 
The Final Step

**Lemma**

Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

**Problem:** Need to make sure that every element is covered!

**Idea:** Amplify this probability by taking the union of $\Omega(\log n)$ random sets $C$.

**Algorithm: Weighted Set Cover-LP**

1: compute $y$, an optimal solution to the linear program
2: $C = \emptyset$
3: repeat $2 \ln n$ times
4: for each $S \in \mathcal{F}$
5: let $C = C \cup \{S\}$ with probability $y(S)$
6: return $C$
Let $C \subseteq \mathcal{F}$ be a random subset with each set $S$ being included independently with probability $y(S)$.

- The expected cost satisfies $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- The probability that $x$ is covered satisfies $\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

**Problem:** Need to make sure that every element is covered!

**Idea:** Amplify this probability by taking the union of $\Omega(\log n)$ random sets $C$.

**Algorithm:**

**Weighted Set Cover-LP** ($X, \mathcal{F}, c$)

1: compute $y$, an optimal solution to the linear program
2: $C = \emptyset$
3: repeat 2 ln $n$ times
   4: for each $S \in \mathcal{F}$
      5: let $C = C \cup \{S\}$ with probability $y(S)$
6: return $C$

Clearly runs in polynomial-time!
Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$. 

By Markov's inequality, 

$$
\Pr\left[c(\mathcal{C}) \leq 4 \ln(n) \cdot c(C^*) \right] \geq \frac{1}{2}.
$$

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, the solution is within a factor of $4 \ln(n)$ of the optimum.

Probability could be further increased by repeating. 

**Typical Approach for Designing Approximation Algorithms based on LPs**

**Proof:**

**Step 1:** The probability that $C$ is a cover

By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln(n)$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$
\Pr\left[x \notin \bigcup_{S \in \mathcal{C}} S\right] \leq \left(\frac{1}{e}\right)^{2 \ln(n)} = \frac{1}{n^2}.
$$

This implies for the event that all elements are covered:

$$
\Pr\left[X = \bigcup_{S \in \mathcal{C}} S\right] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in \mathcal{C}} S\}\right] \geq 1 - \sum_{x \in X} \Pr\left[x \notin \bigcup_{S \in \mathcal{C}} S\right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n^2}.
$$

**Step 2:** The expected approximation ratio

By previous lemma, the expected cost of one iteration is

$$
\sum_{S \in \mathcal{F}} c(S) \cdot y(S).
$$

Linearity $\Rightarrow$

$$
E\left[c(\mathcal{C})\right] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*).
$$
### Analysis of Weighted Set Cover-LP

**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**
Analysis of **Weighted Set Cover-LP**

**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

- **Step 1:** The probability that $C$ is a cover
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1**: The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
Analysis of WEIGHTED SET COVER-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1**: The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    \[
    \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n}
    \]
Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
  \[
  \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.
  \]
Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

**Step 1:** The probability that $C$ is a cover

- By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$
\Pr [x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.
$$

- This implies for the event that all elements are covered:
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1**: The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
  \[
  \Pr[x \not\in \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.
  \]
  - This implies for the event that all elements are covered:
  \[
  \Pr[X = \bigcup_{S \in C} S] =
  \]
Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

**Step 1**: The probability that $C$ is a cover

- By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln(n)$ iterations with probability at least $1 - \frac{1}{e}$, so that

$$\Pr[x \not\in \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.$$  

- This implies for the event that all elements are covered:

$$\Pr[X = \bigcup_{S \in C} S] = 1 - \Pr \left[ \bigcup_{x \in X} \{x \not\in \bigcup_{S \in C} S\} \right]$$
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1**: The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    $$\Pr[ x \notin \bigcup_{S \in C} S ] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}. $$
  - This implies for the event that all elements are covered:
    $$\Pr[ X = \bigcup_{S \in C} S ] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right].$$

**Principle of Inclusion-Exclusion:**

$$\Pr[A \cup B] \leq \Pr[A] + \Pr[B].$$
Theorem

- With probability at least \( 1 - \frac{1}{n} \), the returned set \( C \) is a valid cover of \( X \).
- The expected approximation ratio is \( 2 \ln(n) \).

Proof:

- **Step 1**: The probability that \( C \) is a cover
  - By previous Lemma, an element \( x \in X \) is covered in one of the \( 2 \ln n \) iterations with probability at least \( 1 - \frac{1}{e} \), so that
    \[
    \Pr [ x \notin \bigcup_{S \in C} S ] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \Pr [ X = \bigcup_{S \in C} S ] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right] \geq 1 - \sum_{x \in X} \Pr [ x \notin \bigcup_{S \in C} S ]
    \]
Analysis of **Weighted Set Cover-LP**

**Theorem**
- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**
- **Step 1:** The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    \[
    \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right] \geq 1 - n \cdot \frac{1}{n^2}.
    \]
Analysis of Weighted Set Cover-LP

**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

- **Step 1:** The probability that $C$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    $$\Pr[x \notin \bigcup_{S \in C} S] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.$$  
  - This implies for the event that all elements are covered:
    $$\Pr[X = \bigcup_{S \in C} S] = 1 - \Pr \left[ \bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\} \right] \geq 1 - \sum_{x \in X} \Pr[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.$$
Analysis of **Weighted Set Cover-LP**

**Theorem**
- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

**Step 1:** The probability that $C$ is a cover.
- By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
  \[ \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}. \]
- This implies for the event that all elements are covered:
  \[ \Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}. \]
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1**: The probability $\mathbb{P}$ that $C$ is a cover.
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    \[
    \mathbb{P}[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \mathbb{P}[X = \bigcup_{S \in C} S] = 1 - \mathbb{P}\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right] 
    \geq 1 - \sum_{x \in X} \mathbb{P}[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

- **Step 2**: The expected approximation ratio
**Analysis of** WEIGHTED SET COVER-LP

**Theorem**
- With probability at least $1 - \frac{1}{n}$, the returned set $\mathcal{C}$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

- **Step 1:** The probability that $\mathcal{C}$ is a cover
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    $$\Pr[x \notin \bigcup_{S \in \mathcal{C}} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.$$  
  - This implies for the event that all elements are covered:
    $$\Pr[X = \bigcup_{S \in \mathcal{C}} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in \mathcal{C}} S\}\right].$$

- **Step 2:** The expected approximation ratio
  - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

- **Step 1:** The probability that $C$ is a cover.
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    
    $$
    \Pr [ x \notin \bigcup_{S \in C} S ] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
    $$
  - This implies for the event that all elements are covered:
    
    $$
    \Pr [ X = \bigcup_{S \in C} S ] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right]
    \geq 1 - \sum_{x \in X} \Pr [ x \notin \bigcup_{S \in C} S ] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    $$

- **Step 2:** The expected approximation ratio.
  - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
  - Linearity $\Rightarrow \mathbf{E} [ c(C) ] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
Analysis of Weighted Set Cover-LP

Theorem
- With probability at least \(1 - \frac{1}{n}\), the returned set \(C\) is a valid cover of \(X\).
- The expected approximation ratio is \(2 \ln(n)\).

Proof:
- **Step 1**: The probability that \(C\) is a cover
  - By previous Lemma, an element \(x \in X\) is covered in one of the \(2 \ln n\) iterations with probability at least \(1 - \frac{1}{e}\), so that
    \[
    \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \Pr[X = \bigcup_{S \in C} S] = 1 - \sum_{x \in X} \Pr[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]
- **Step 2**: The expected approximation ratio
  - By previous lemma, the expected cost of one iteration is \(\sum_{S \in \mathcal{F}} c(S) \cdot y(S)\).
  - Linearity \(\Rightarrow\) \(\mathbb{E}[c(C)] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*)\)
Analysis of Weighted Set Cover-LP

---

**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

**Proof:**

- **Step 1:** The probability that $C$ is a cover $\checkmark$
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    \[
    \Pr [ x \not\in \bigcup_{S \in C} S ] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \Pr [ X = \bigcup_{S \in C} S ] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \not\in \bigcup_{S \in C} S \} \right] \geq 1 - \sum_{x \in X} \Pr [ x \not\in \bigcup_{S \in C} S ] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

- **Step 2:** The expected approximation ratio $\checkmark$
  - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
  - Linearity $\Rightarrow$ $E [ c(C) ] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*)$
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

Proof:

- **Step 1:** The probability that $C$ is a cover $\checkmark$
  - By previous Lemma, an element $x \in X$ is covered in one of the $2 \ln n$ iterations with probability at least $1 - \frac{1}{e}$, so that
    \[
    \Pr[ x \notin \bigcup_{S \in C} S ] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \Pr[ X = \bigcup_{S \in C} S ] = 1 - \Pr\left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right] \geq 1 - \sum_{x \in X} \Pr[ x \notin \bigcup_{S \in C} S ] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

- **Step 2:** The expected approximation ratio $\checkmark$
  - By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
  - Linearity $\Rightarrow \mathbb{E}[c(C)] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*)$
Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

By Markov’s inequality, $\Pr [ c(C) \leq 4 \ln(n) \cdot c(C^*) ] \geq 1/2.$
Analysis of **Weighted Set Cover-LP**

**Theorem**
- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

By Markov’s inequality, $\Pr[ c(C) \leq 4 \ln(n) \cdot c(C^*) ] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is within a factor of $4 \ln(n)$ of the optimum.
Analysis of **Weighted Set Cover-LP**

**Theorem**

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

By Markov’s inequality, $\Pr [c(C) \leq 4 \ln(n) \cdot c(C^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is within a factor of $4 \ln(n)$ of the optimum.

Probability could be further increased by repeating...
Analysis of Weighted Set Cover-LP

Theorem

- With probability at least $1 - \frac{1}{n}$, the returned set $C$ is a valid cover of $X$.
- The expected approximation ratio is $2 \ln(n)$.

By Markov's inequality, $\Pr[c(C) \leq 4 \ln(n) \cdot c(C^*)] \geq 1/2$.

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is within a factor of $4 \ln(n)$ of the optimum.

Probability could be further increased by repeating

Typical Approach for Designing Approximation Algorithms based on LPs
Spectrum of Approximations

- KNAPSACK
- SUBSET-SUM
- SCHEDULING
- EUCLIDEAN-TSP
- VERTEX-COVER
- MAX-3-CNF
- MAX-CUT
- METRIC-TSP
- SET-COVER
- MAX-CLIQUE

FPTAS

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- KNAPSACK
- SUBSET-SUM
- SCHEDULING
- EUCLIDEAN-TSP
- VERTEX-COVER
- MAX-3-CNF
- MAX-CUT
- METRIC-TSP
- SET-COVER
- MAX-CLIQUE

FPTAS

PTAS

APX

log-APX

poly-APX

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- KNAPSACK
- SUBSET-SUM
- SCHEDULING
- EUCLIDEAN-TSP
- VERTEX-COVER
- MAX-3-CNF
- MAX-CUT
- METRIC-TSP
- SET-COVER
- MAX-CLIQUE

- FPTAS
- PTAS
- APX
- log-APX
- poly-APX

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- SCHEDULING, EUCLIDEAN-TSP
- KNAPSACK, SUBSET-SUM
- FPTAS, PTAS

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- KNAPSACK
- SUBSET-SUM
- SCHEDULING, EUCLIDEAN-TSP
- VERTEX-COVER, MAX-3-CNF, MAX-CUT
- METRIC-TSP
- SET-COVER
- MAX-CLIQUE

FPTAS  PTAS  APX

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

1. KNAPSACK
2. SET-COVER
3. MAX-CLIQUE
4. MAX-3-CNF
5. MAX-CUT
6. METRIC-TSP
7. EUCLIDEAN-TSP
8. SCHEDULING
9. VERTEX-COVER

Classification:
- FPTAS
- PTAS
- APX
- log-APX
- poly-APX

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- VERTEX-COVER, MAX-3-CNF, MAX-CUT, METRIC-TSP
- SCHEDULING, EUCLIDEAN-TSP
- KNAPSACK, SUBSET-SUM
- FPTAS, PTAS, APX, log-APX

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

- **FPTAS**
- **PTAS**
- **APX**
- **log-APX**
- **poly-APX**

**SET-COVER**

- **VERTEX-COVER**, **MAX-3-CNF**, **MAX-CUT**, **METRIC-TSP**

**SCHEDULING**, **EUCLIDEAN-TSP**

- **KNAPSACK**
- **SUBSET-SUM**

Thank you and Best Wishes for the Exam!
Spectrum of Approximations

MAX-CLIQUE

SET-COVER

VERTEX-COVER, MAX-3-CNF, MAX-CUT METRIC-TSP

SCHEDULING, EUCLIDEAN-TSP

KNAPSACK SUBSET-SUM

FPTAS PTAS APX log-APX poly-APX
Thank you and Best Wishes for the Exam!