VI. Approx. Algorithms: Randomisation and Rounding

Thomas Sauerwald

Easter 2021
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
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, $O((1/\epsilon)^2 \cdot n^3)$. 

VI. Randomisation and Rounding Randomised Approximation
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 \textit{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 \textit{fully polynomial-time approximation scheme} (FPTAS) if the runtime is polynomial in both $1/\epsilon$ and $n$. For example, $O((1/\epsilon)^2 \cdot n^3)$.

VI. Randomisation and Rounding

Randomised Approximation
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
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 satisfaction 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:**

\[
(x_1 \lor x_3 \lor \overline{x}_4) \land (x_1 \lor \overline{x}_3 \lor \overline{x}_5) \land (x_2 \lor \overline{x}_4 \lor x_5) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_3)
\]
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.: \((x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor x_3 \lor \overline{x_5}) \land \cdots\)
- **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 uniformly and independently at random?
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

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:
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:
  \[
  Y_i = 1 \{ \text{clause } i \text{ is satisfied} \}
  \]
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 \( i \),
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 \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
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 $i$,
  \[ \Pr [ \text{clause } i \text{ is not satisfied} ] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \]
  \[ \Rightarrow \quad \Pr [ \text{clause } i \text{ is satisfied} ] = 1 - \frac{1}{8} = \frac{7}{8} \]

  \[ \Rightarrow \quad \mathbb{E} [ Y_i ] = \Pr [ Y_i = 1 ] \cdot 1 = \frac{7}{8}. \]
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,
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,
  \[
  \mathbb{E}[Y]
  \]
**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,
  \[
  \mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{m} Y_i\right]
  \]
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,
  \[ E[Y] = E \left[ \sum_{i=1}^{m} Y_i \right] \]
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,
  \[
  E[Y] = E\left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} E[Y_i]
  \]
  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:
  
  \[
  Y_i = \mathbf{1}_{\text{clause } i \text{ is satisfied}}
  \]

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

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

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

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

\[\text{Linearity of Expectations}\]
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:
  \[
  Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}
  \]

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

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

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

- 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*
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 \( i \),
  \[
  \Pr[ \text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[ \text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]

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

- Maximum number of satisfiable clauses is \( m \).
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 \( i \),
  \[
  \Pr[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
  \[
  \Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}
  \]
  \[
  \Rightarrow \quad E[Y_i] = \Pr[Y_i = 1] \cdot 1 = \frac{7}{8}.
  \]

- 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

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.

Theorem 35.6

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

Corollary

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

Corollary

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.

Follows from the previous Corollary.
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 $\frac{7}{8}$ of all clauses.

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

VI. Randomisation and Rounding

MAX-3-CNF
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 $Y(\omega) \geq \mathbb{E}[Y]$

**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 \( 7/8 \) 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.
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.
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 $1/(8m)$. 
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.
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 \(E[Y]\)!
**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 \( E[Y] \)!

**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}(\(\phi, n, m\))

\begin{itemize}
  \item This algorithm is deterministic.
  \item \textbf{Theorem}
  \begin{quote}
    \textbf{GREEDY-3-CNF}(\(\phi, n, m\)) is a polynomial-time 8/7-approximation.
  \end{quote}
\end{itemize}
Analysis of **GREEDY-3-CNF**($\phi, n, m$)

**Theorem**

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

**Proof:**

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

\begin{itemize}
  \item \textbf{Theorem}
  \end{itemize}

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

\begin{itemize}
  \item \textbf{Proof:}
  \begin{itemize}
    \item \textbf{Step 1:} polynomial-time algorithm
  \end{itemize}
\end{itemize}
Analysis of \textsc{Greedy-3-CNF}(\phi, n, m)

This algorithm is deterministic.

**Theorem**
\textsc{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
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
  - 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:

  \[ \mathbb{E}[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] \]
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:

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

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
  - 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:

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

    computable in $O(1)$
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:

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

computable in $O(1)$
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:
    $$E[Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^{m} E[Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1]$$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
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:}

- **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 \(j = 1, 2, \ldots, n\),

**This algorithm is deterministic.**
Analysis of \textit{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:

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

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

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

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,
    \[
    \mathbb{E}[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j] \geq \mathbb{E}[Y | x_1 = v_1, \ldots, x_{j-1} = v_{j-1}]
    \geq \mathbb{E}[Y | x_1 = v_1, \ldots, x_{j-2} = v_{j-2}]
    \]
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
  - 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:
    \[
    \mathbb{E}[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 ] = \sum_{i=1}^{m} \mathbb{E}[ Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 ]
    \]

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,
    \[
    \mathbb{E}[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j ] \geq \mathbb{E}[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} ] \\
    \geq \mathbb{E}[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} ] \\
    \geq \cdots \\
    \geq \mathbb{E}[ Y ]
    \]
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.

\begin{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

- Due to the greedy choice in each iteration \(j = 1, 2, \ldots, n\),

\[
E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v \right] \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right]
\]

\[
\geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \right]
\]

\[
\vdots
\]

\[
\geq E \left[ Y \right] = \frac{7}{8} \cdot m.
\]

\end{proof}
Analysis of \textsc{Greedy-3-CNF}(\(\phi, n, m\))

This algorithm is deterministic.

\textbf{Theorem}

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

\textbf{Proof:}

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

**Theorem**

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

**Proof:**

- **Step 1:** polynomial-time algorithm √
  - In iteration \( j = 1, 2, \ldots, n \), \( Y = Y(ϕ) \) 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 \( j = 1, 2, \ldots, n \),
    \[
    E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \right] \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \right] \\
    \geq E \left[ Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \right] \\
    \vdots \\
    \geq E \left[ Y \right] = \frac{7}{8} \cdot m.
    \]
    □
Run of $\text{GREEDY-3-CNF}(\varphi, n, m)$

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

\[
(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (x_1 \lor x_2 \lor x_4) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor x_3 \lor x_4)
\]

VI. Randomisation and Rounding
Run of GREEDY-3-CNF(ϕ, n, m)

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

VI. Randomisation and Rounding

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

\[(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3) \land (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor \overline{x}_3 \lor \overline{x}_4) \land (x_1 \lor \overline{x}_3 \lor \overline{x}_4) \land (x_1 \lor x_3 \lor \overline{x}_4) \land (x_2 \lor \overline{x}_3 \lor \overline{x}_4)\]

**VI. Randomisation and Rounding**

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

$((x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_1 \lor x_3 \lor x_4) \land (x_4 \lor x_3 \lor x_4) \land (x_4 \lor x_1 \lor x_2) \land (x_4 \lor x_2 \lor x_3) \land (x_4 \lor x_3 \lor x_2) \land (x_4 \lor x_1 \lor x_3) \land (x_4 \lor x_2 \lor x_4) \land (x_4 \lor x_3 \lor x_4))$

VI. Randomisation and Rounding

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

\[1 \land 1 \land 1 \land (x_3 \lor x_4) \land 1 \land (\overline{x_2} \lor \overline{x_3}) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})\]
Run of \textbf{GREEDY-3-CNF}(\(\varphi, n, m\))

\[1 \land 1 \land 1 \land (\overline{x}_3 \lor x_4) \land 1 \land (\overline{x}_2 \lor \overline{x}_3) \land (x_2 \lor x_3) \land (\overline{x}_2 \lor x_3) \land 1 \land (x_2 \lor \overline{x}_3 \lor \overline{x}_4)\]
Run of **GREEDY-3-CNF**($\varphi, n, m$)

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

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

VI. Randomisation and Rounding
Run of **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($\varphi, n, m$)

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

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

\[
1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x_3} \lor x_4)
\]

### VI. 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

### VI. Randomisation and Rounding

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

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

\[ x_1 = 0 \quad \text{8.625} \]

\[ x_1 = 1 \quad \text{8.75} \]

\[ x_2 = 0 \]

\[ x_2 = 1 \]

\[ x_3 = 0 \]

\[ x_3 = 1 \]

\[ x_3 = 0 \]

\[ x_3 = 1 \]

\[ x_3 = 0 \]

\[ x_3 = 1 \]

\[ x_3 = 0 \]

\[ x_3 = 1 \]

\[ 0000 \quad 0001 \quad 0010 \quad 0011 \quad 0100 \quad 0101 \quad 0110 \quad 0111 \quad 1000 \quad 1001 \quad 1010 \quad 1011 \quad 1100 \quad 1101 \quad 1110 \quad 1111 \]

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

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

![Decision tree diagram]

- **VI. Randomisation and Rounding**
- **MAX-3-CNF**
Run of \textbf{GREEDY-3-CNF}(\(\varphi, n, m\))

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

\[ x_1 = 0 \]
\[ 8.625 \]

\[ x_2 = 0 \]
\[ x_3 = 0 \]

\[ x_2 = 1 \]
\[ x_3 = 1 \]

\[ x_1 = 1 \]
\[ 8.875 \]

\[ x_2 = 0 \]
\[ x_3 = 0 \]

\[ x_2 = 1 \]
\[ x_3 = 1 \]

\[ 9 \]

\[ 9 \]

\[ 8.75 \]

\[ 0000 \]
\[ 0001 \]
\[ 0010 \]
\[ 0011 \]
\[ 0100 \]
\[ 0101 \]
\[ 0110 \]
\[ 0111 \]
\[ 1000 \]
Run of \textsc{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(ϕ, n, m)

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

\[ x_1 = 0 \] \quad \text{8.75}

\[ x_1 = 1 \] \quad \text{8.875}

\[ x_2 = 0 \] \quad \text{8.625}

\[ x_2 = 1 \] \quad \text{9.25}

\[ x_3 = 0 \] \quad \text{9.5}

\[ x_3 = 1 \] \quad \text{9.5}

\[ x_4 = 0 \] \quad \text{8.75}

\[ x_4 = 1 \] \quad \text{8.5}

VI. Randomisation and Rounding

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

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

\[
\begin{align*}
&x_1 = 0 \\
&x_2 = 0 \\
&x_3 = 0 \\
&x_4 = 0 \\
&x_1 = 1 \\
&x_2 = 1 \\
&x_3 = 1 \\
&x_4 = 1 \\
\end{align*}
\]

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**

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

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

**Theorem**

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

\textsc{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-CNF unless P=NP.

Essentially there is nothing smarter than just guessing!
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
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**

**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**

**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.
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.
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
- Perform all tasks with the minimal amount of resources
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
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
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
- 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** $C$

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 $E'$, adds it to $C$, and removes from $E'$ every edge incident on either $u$ or $v$. Line 7 **return** $C$.
The Greedy Approach from (Unweighted) Vertex Cover

Algorithm 35.1: APPROX-VERTEX-COVER(G)

1. $C = \emptyset$
2. $E' = G \cdot E$
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 $C$

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 $G \cdot E$ of the graph. The loop of lines 3–6 repeatedly picks an edge $(u, v)$ from $E'$, adds it to $C$, and removes from $E'$ every edge incident on either $u$ or $v$. Line 7 returns the computed vertex cover $C$.

Computed solution has weight 101
Optimal solution has weight 4
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** \(C\)

**Figure 35.1** illustrates how \(\text{APPROX-VERTEX-COVER}\) 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 \(E' = G.E\). The loop of lines 3–6 repeatedly picks an edge \((u, v)\) from \(E'\), adds it to \(C\), and removes every edge incident on either \(u\) or \(v\). Lines 100 and 1 depict the input graph and the computed solution, respectively. The computed solution has weight 101, while the optimal solution has weight 4.
The Greedy Approach from (Unweighted) Vertex Cover

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

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

**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 \( E' \), adds it to \( C \), and removes from \( E' \) every edge incident on either \( u \) or \( v \). Line 7 returns the constructed vertex cover.

Computed solution has weight 100
Optimal solution has weight 4
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.
Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

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

Idea: Round the solution of an associated linear program.

0-1 Integer Program

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

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

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

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

Linear Program

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

optimum is a lower bound on the optimal weight of a minimum weight-cover.
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. \(\text{if } \bar{x}(v) \geq 1/2\)
5. \(C = C \cup \{v\}\)
6. **return** \(C\)
The Algorithm

\[ \text{APPROX-MIN-WEIGHT-VC}(G, w) \]

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

**Theorem 35.7**

\text{APPROX-MIN-WEIGHT-VC} is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.
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 \(\nu \in V\)
4.  **if** \(\bar{x}(\nu) \geq 1/2\)
5.    \(C = C \cup \{\nu\}\)
6. **return** \(C\)

**Theorem 35.7**

**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**

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

fractional solution of LP
with weight $= 5.5$
Example of **APPROX-MIN-WEIGHT-VC**

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

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

fractional solution of LP with weight = 5.5

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

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

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

Rounding

Fractional solution of LP with weight = 5.5

Rounded solution of LP with weight = 10

Optimal solution with weight = 6
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

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 $w(C) \leq 2z^*$.

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

Proof (Approximation Ratio is 2 and Correctness):

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 $w(C) \leq 2 z^*$:

$x(u) + x(v) \geq 1 \Rightarrow w(C) = \sum v \in V w(v) x(v) \geq \sum v \in V : x(v) \geq 1/2 w(v) = 1/2 w(C)$. 

Rounding
Proof (Approximation Ratio is 2 and Correctness):

- Let $C^*$ be an optimal solution to the minimum-weight vertex cover problem

![Diagram showing the process of rounding and the approximated solution]
Proof (Approximation Ratio is 2 and Correctness):

- 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
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- 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

$$z^* \leq w(C^*)$$
Proof (Approximation Ratio is 2 and Correctness):

- 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:

  ![Diagram of vertex cover](image)

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

  \[ w(C^*) \geq z^* = \sum_{v \in V} w(v)x(v) \geq \sum_{v \in V} x(v) \geq \frac{1}{2}w(C) \cdot \frac{1}{2} = \frac{1}{4}w(C). \]
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):
- 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 and Correctness):
- 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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2$
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- 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 $x(u) + x(v) \geq 1$
  \[ \Rightarrow \text{ at least one of } \bar{x}(u) \text{ and } \bar{x}(v) \text{ is at least } 1/2 \Rightarrow C \text{ covers edge } (u, v) \]
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):
- 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 \( w(C) \leq 2z^* \):
Proof (Approximation Ratio is 2 and Correctness):
- 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 $w(C) \leq 2z^*$:
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- 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 and Correctness):
- 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 $w(C) \leq 2z^*$:
  \[
w(C^*) \geq z^* = \sum_{v \in V} w(v)\bar{x}(v)\]
Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- 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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2$  
  $\Rightarrow C$ covers edge $(u, v)$

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

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

VI. Randomisation and Rounding

Weighted Vertex Cover
Proof (Approximation Ratio is 2 and Correctness):

- 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 $w(C) \leq 2z^*$:

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

Proof (Approximation Ratio is 2 and Correctness):
- 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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2$ ⇒ $C$ covers edge $(u, v)$
- **Step 2:** The computed set $C$ satisfies $w(C) \leq 2z^*$:

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

Proof (Approximation Ratio is 2 and Correctness):

- 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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2$ \(\Rightarrow\) $C$ covers edge $(u, v)$

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

\[
w(C^*) \geq z^* = \sum_{v \in V} w(v)\overline{x}(v) \geq \sum_{v \in V: \overline{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C). \quad \square
\]
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
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 $C \subseteq \mathcal{F}$ such that $X = \bigcup_{S \in C} S$.
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 \( \mathcal{C} \subseteq \mathcal{F} \)

\[
X = \bigcup_{S \in \mathcal{C}} S.
\]

Sum over the costs of all sets in \( \mathcal{C} \).
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 $C \subseteq \mathcal{F}$

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

Sum over the costs of all sets in $C$
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 $C \subseteq \mathcal{F}$

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

Remarks:

- generalisation of the weighted vertex-cover problem
- models resource allocation problems
Exercise: Try to formulate the integer program and linear program of the weighted SET-COVER problem (solution on next slide!)
Setting up an Integer Program

**0-1 Integer Program**

\[
\begin{align*}
\text{minimize} & \quad \sum_{S \in \mathcal{F}} c(S) y(S) \\
\text{subject to} & \quad \sum_{S \in \mathcal{F} : \ x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\
& \quad y(S) \in \{0, 1\} \quad \text{for each } S \in \mathcal{F}
\end{align*}
\]
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 \( S \in \mathcal{F} \)
Back to the Example

<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>
</tbody>
</table>

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!
Back to the Example

\[ c : \begin{array}{ccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\ 2 & 3 & 3 & 5 & 1 & 2 \\ \end{array} \]

\[ y(\cdot) : \begin{array}{ccccccc} 1/2 & 1/2 & 1/2 & 1/2 & 1 & 1/2 \\ \end{array} \]

Cost equals 8.5. The strategy employed for Vertex-Cover would take all 6 sets! Even worse: If all \( y(\cdot) \)’s were below \( 1/2 \), we would not even return a valid cover!
Back to the Example

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
Back to the Example

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

Cost equals 8.5

<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>
Back to the Example

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>$S_1$</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>
</tr>
<tr>
<td>$y(.)$:</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
</tr>
</tbody>
</table>

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

The expected cost satisfies

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

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

$$\Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}.$$
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.

Randomised Rounding

- 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 $S \in \mathcal{F}$. 

Therefore, $E[\bar{y}(S)] = y(S)$.
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.

- 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:

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

- Therefore, $\mathbb{E} [\bar{y}(S)] = y(S)$. 

Randomised Rounding
Randomised Rounding

$$\begin{array}{ccccccc}
S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\
c: & 2 & 3 & 3 & 5 & 1 & 2 \\
y(.): & 1/2 & 1/2 & 1/2 & 1/2 & 1 & 1/2
\end{array}$$

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

Lemma
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 $y$-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

\[
\begin{array}{ccccccc}
S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\
2 & 3 & 3 & 5 & 1 & 2 \\
c : & & & & & & \\
y(\cdot) : & 1/2 & 1/2 & 1/2 & 1/2 & 1 & 1/2 \\
\end{array}
\]

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

Lemma

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

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 $\Pr[x \in \cup_{S \in C} S] \geq 1 - \frac{1}{e}$.
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 $C$
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 $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- 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 $C$

  $$E[c(C)]$$
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 \( \Pr[x \in \cup_{S \in C} S] \geq 1 - \frac{1}{e} \).

Proof:

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

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

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 \( C \)

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

---

VI. Randomisation and Rounding

Weighted Set Cover 24
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 $C$

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

  $$\Pr[x \not\in \bigcup_{S \in C} S]$$
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 $\mathbb{E}[c(C)] = \sum_{S \in F} c(S) \cdot y(S)$.
- 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

  $\Pr[x \notin \bigcup_{S \in C} S] = \prod_{S \in F : \ x \in S} \Pr[S \notin C]$.
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 $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- 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 $C$

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

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

  $$\Pr[x \not\in \bigcup_{S \in C} S] = \prod_{S \in \mathcal{F} : x \in S} \Pr[S \not\in \mathcal{C}] = \prod_{S \in \mathcal{F} : x \in S} (1 - y(S)).$$
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 \( \mathbb{E}[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \).
- 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

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

\( 1 + x \leq e^x \) 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 $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- 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

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

  $1 + x \leq e^x$ 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 $E[c(C)] = \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
- 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

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

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

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

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

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

    $$= e^{-\sum_{S \in \mathcal{F}: x \in S} y(S)}$$

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

    $y$ solves the LP!
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

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

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

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

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

$y$ solves the LP!
Proof of Lemma

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 $C$

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

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

- **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!
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 $\mathbb{Pr}[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$. 

VI. Randomisation and Rounding

Weighted Set Cover
The Final Step

**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 $\Pr[x \in \cup_{S \in C} S] \geq 1 - \frac{1}{e}$.

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

**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 $\Pr[x \in \bigcup_{S \in \mathcal{C}} S] \geq 1 - \frac{1}{e}$.

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

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

---

VI. Randomisation and Rounding

Weighted Set Cover
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 $C$.

**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 $C$

VI. Randomisation and Rounding

Weighted Set Cover
The Final Step

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 $Pr[x \in \bigcup_{S \in C} S] \geq 1 - \frac{1}{e}$.

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

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

**Weighted Set Cover-LP** ($X, F, c$)

1: compute $y$, an optimal solution to the linear program
2: $C = \emptyset$
3: repeat 2 $\ln n$ times
4: for each $S \in F$
5: \hspace{1em} 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 $2 \ln(n)$.
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:
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
  \[
  \Pr[x \notin \bigcup_{S \in C} S] \leq \left(\frac{1}{e}\right)^{2 \ln n}
  \]
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

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

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

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

$$
\Pr[X = \bigcup_{S \in C} S] = 
$$
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:
    \[
    \Pr [X = \bigcup_{S \in C} S] = 1 - \Pr \left[ \bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\} \right].
    \]
**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
    
    $$\Pr \left[ x \notin \bigcup_{S \in C} S \right] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.$$
  - This implies for the event that all elements are covered:
    
    $$\Pr \left[ X = \bigcup_{S \in C} S \right] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right]$$

**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:

$$\Pr [X = \bigcup_{S \in C} S] = 1 - \Pr \left[ \bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\} \right] \geq 1 - \sum_{x \in X} \Pr [x \notin \bigcup_{S \in C} S].$$

**Step 2:** The expected approximation ratio

- By previous lemma, the expected cost of one iteration is $\sum_{S \in F} c(S) \cdot y(S)$.
- Linearity $\Rightarrow \mathbb{E} [c(C)] \leq 2 \ln(n) \cdot \sum_{S \in F} c(S) \cdot y(S)$.
**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:
    $$\Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right].$$

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

- 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

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

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

\[
\Pr \left[ X = \bigcup_{S \in C} S \right] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right] \geq 1 - \sum_{x \in X} \Pr \left[ x \notin \bigcup_{S \in C} S \right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
\]
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 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:
    \[
    \Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right] \geq 1 - \sum_{x \in X} \Pr[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]
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:
    \[
    \Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[ \bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\} \right]
    \]
- **Step 2:** The expected approximation ratio
  \[
  \Pr[A \cup B] \leq \Pr[A] + \Pr[B] \geq 1 - \sum_{x \in X} \Pr[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
  \]
Analysis of \textbf{W\textsc{EIGHTED SET COVER-LP}}

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

**Proof:**

\begin{itemize}
  \item **Step 1:** The probability that $C$ is a cover $\checkmark$
    \begin{itemize}
      \item 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
      \[
      \Pr [x \notin \bigcup_{S \in C} S] \leq \left( \frac{1}{e} \right)^{2 \ln n} = \frac{1}{n^2}.
      \]
      \item This implies for the event that all elements are covered:
      \[
      \Pr [X = \bigcup_{S \in C} S] = 1 - \Pr \left[ \bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\} \right] \geq 1 - \sum_{x \in X} \Pr [x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
      \]
    \end{itemize}
  \item **Step 2:** The expected approximation ratio
    \begin{itemize}
      \item By previous lemma, the expected cost of one iteration is $\sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
    \end{itemize}
\end{itemize}
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:
    \[
    \Pr[X = \bigcup_{S \in C} S] = 1 - \Pr\left[\bigcup_{x \in X} \{x \notin \bigcup_{S \in C} S\}\right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

- **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 $\Rightarrow \mathbb{E}[c(C)] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$.
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:
    \[
    \Pr[X = \bigcup_{S \in C} S] \geq 1 - \sum_{x \in X} \Pr[x \notin \bigcup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

- **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 \( \Rightarrow \) \( \mathbb{E}[c(C)] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*) \)
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.
  - 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:
    \[ \Pr [ X = \bigcup_{S \in C} S ] = 1 - \Pr \left[ \bigcup_{x \in X} \{ x \notin \bigcup_{S \in C} S \} \right] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}. \]
- **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 $✓$
  - 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 $\Rightarrow E [ c(C) ] \leq 2 \ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2 \ln(n) \cdot c(C^*)$ $\square$
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, $\Pr [c(C) \leq 4 \ln(n) \cdot c(C^*)] \geq 1/2.$
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, $\Pr \left[ c(C) \leq 4 \ln(n) \cdot c(C^*) \right] \geq \frac{1}{2}$.

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

Typical Approach for Designing Approximation Algorithms based on LPs
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
Recall:

**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-CNF Satisfiability (MAX-SAT)**
MAX-CNF

Recall:

**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-CNF Satisfiability (MAX-SAT)**

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

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-CNF Satisfiability (MAX-SAT)

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

Why study this generalised problem?
Recall:

**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-CNFSatisfiability (MAX-SAT)**

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

Why study this generalised problem?

- Allowing arbitrary clause lengths makes the problem more interesting (we will see that simply guessing is not the best!)
- a nice concluding example where we can practice previously learned approaches
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

For any clause $i$ which has length $\ell$, $\Pr[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} = \alpha \ell$.

In particular, the guessing algorithm is a randomised 2-approximation.

Analysis

Proof: First statement as in the proof of Theorem 35.6. For clause $i$ not to be satisfied, all $\ell$ occurring variables must be set to a specific value. As before, let $Y = \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then, $E[Y] = E[m \sum_{i=1}^{m} Y_i] = m \sum_{i=1}^{m} E[Y_i] \geq m \sum_{i=1}^{m} \frac{1}{2} = \frac{1}{2} \cdot m$. 

VI. Randomisation and Rounding MAX-CNF
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause $i$ which has length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_\ell.$$

In particular, the guessing algorithm is a randomised 2-approximation.
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause $i$ which has length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_\ell.$$  

In particular, the guessing algorithm is a randomised 2-approximation.

Proof:
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause \(i\) which has length \(\ell\),

\[
\Pr[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_\ell.
\]

In particular, the guessing algorithm is a randomised 2-approximation.

Proof:

- First statement as in the proof of Theorem 35.6. For clause \(i\) not to be satisfied, all \(\ell\) occurring variables must be set to a specific value.
Approach 1: Guessing the Assignment

Assign each variable true or false uniformly and independently at random.

Recall: This was the successful approach to solve MAX-3-CNF!

Analysis

For any clause $i$ which has length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] = 1 - 2^{-\ell} := \alpha_\ell.$$  

In particular, the guessing algorithm is a randomised 2-approximation.

Proof:

- First statement as in the proof of Theorem 35.6. For clause $i$ not to be satisfied, all $\ell$ occurring variables must be set to a specific value.
- As before, let $Y := \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then,

$$\mathbb{E}[Y] = \mathbb{E} \left[ \sum_{i=1}^{m} Y_i \right] = \sum_{i=1}^{m} \mathbb{E}[Y_i] \geq \sum_{i=1}^{m} \frac{1}{2} = \frac{1}{2} \cdot m.$$
First solve a linear program and use fractional values for a **biased** coin flip.

\[
\max \sum_{i=1}^{m} z_i \\
\text{subject to } \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \text{ for each } i = 1, 2, \ldots, m \\
z_i \in \{0, 1\} \text{ for each } i = 1, 2, \ldots, m \\
y_j \in \{0, 1\} \text{ for each } j = 1, 2, \ldots, n
\]
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a biased coin flip. The same as randomised rounding!

0-1 Integer Program

maximize $\sum_{i=1}^{m} z_i$

subject to $\sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i$ for each $i = 1, 2, \ldots, m$

$z_i \in \{0, 1\}$ for each $i = 1, 2, \ldots, m$

$y_j \in \{0, 1\}$ for each $j = 1, 2, \ldots, n$
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a **biased** coin flip.

The same as **randomised rounding**!

0-1 Integer Program

maximize \( \sum_{i=1}^{m} z_i \)

subject to \( \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \) for each \( i = 1, 2, \ldots, m \)

\( z_i \in \{0, 1\} \) for each \( i = 1, 2, \ldots, m \)

\( y_j \in \{0, 1\} \) for each \( j = 1, 2, \ldots, n \)

\( C_i^+ \) is the index set of the un-negated variables of clause \( i \).
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a biased coin flip.

The same as randomised rounding!

0-1 Integer Program

maximize \[ \sum_{i=1}^{m} z_i \]
subject to \[ \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \text{ for each } i = 1, 2, \ldots, m \]

These auxiliary variables are used to reflect whether a clause is satisfied or not.

\[ C_i^+ \text{ is the index set of the unnegated variables of clause } i. \]

\[ z_i \in \{0, 1\} \text{ for each } i = 1, 2, \ldots, m \]
\[ y_j \in \{0, 1\} \text{ for each } j = 1, 2, \ldots, n \]
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

First solve a linear program and use fractional values for a \textbf{biased} coin flip.

The same as \textit{randomised rounding}!

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{m} z_i \\
\text{subject to} & \quad \sum_{j \in C_i^+} y_j + \sum_{j \in C_i^-} (1 - y_j) \geq z_i \quad \text{for each } i = 1, 2, \ldots, m \\
& \quad z_i \in \{0, 1\} \quad \text{for each } i = 1, 2, \ldots, m \\
& \quad y_j \in \{0, 1\} \quad \text{for each } j = 1, 2, \ldots, n
\end{align*}
\]

- In the \textit{corresponding LP} each $\in \{0, 1\}$ is replaced by $\in [0, 1]$
- Let $(y^*, z^*)$ be the optimal solution of the LP
- Obtain an integer solution $y$ through randomised rounding of $y^*$

$C_i^+$ is the index set of the un-negated variables of clause $i$. These \textit{auxiliary} variables are used to reflect whether a clause is satisfied or not.

VI. Randomisation and Rounding
Lemma

For any clause $i$ of length $\ell$, \[
\Pr[\text{clause } i \text{ is satisfied}] \geq \left( 1 - \left( 1 - \frac{1}{\ell} \right)^{\ell} \right) \cdot z_i^*.
\]
Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (1/2):
Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x}_j$ in the whole formula)
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left( 1 - \left( 1 - \frac{1}{\ell} \right)^\ell \right) \cdot z_i^*.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x}_j$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x}_j$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \Pr[\text{clause } i \text{ is satisfied}] =$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\text{Pr}[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \text{Pr}[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^\ell \text{Pr}[\text{ } y_j \text{ is false }].$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[\ y_j \text{ is false } ] = 1 - \prod_{j=1}^{\ell} \left(1 - y_j^*\right)$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[ y_j \text{ is false } ] = 1 - \prod_{j=1}^{\ell} (1 - y_j^*)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}.$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z_i^*.$$ 

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[ y_j \text{ is false }] = 1 - \prod_{j=1}^{\ell} \left(1 - y_j^*\right)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}.$$
For any clause $i$ of length $\ell$, 

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$ 

**Proof of Lemma (1/2):**

- Assume w.l.o.g. all literals in clause $i$ appear non-negated (otherwise replace every occurrence of $x_j$ by $\overline{x_j}$ in the whole formula)
- Further, by relabelling assume $C_i = (x_1 \lor \cdots \lor x_\ell)$

$$\Rightarrow \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[\text{ } y_j \text{ is false }] = 1 - \prod_{j=1}^{\ell} (1 - y_j^*)$$

Arithmetic vs. geometric mean:

$$\frac{a_1 + \cdots + a_k}{k} \geq \sqrt[k]{a_1 \times \cdots \times a_k}.$$

$$\geq 1 - \left(\frac{\sum_{j=1}^{\ell} (1 - y_j^*)}{\ell}\right)^\ell$$

$$= 1 - \left(1 - \frac{\sum_{j=1}^{\ell} y_j^*}{\ell}\right)^\ell.$$
Analysis of Randomised Rounding

Lemma

For any clause \( i \) of length \( \ell \),

\[
\Pr[\text{clause } i \text{ is satisfied}] \geq \left( 1 - \left( 1 - \frac{1}{\ell} \right)^\ell \right) \cdot z_i^*.
\]

Proof of Lemma (1/2):

- Assume w.l.o.g. all literals in clause \( i \) appear non-negated (otherwise replace every occurrence of \( x_j \) by \( \overline{x}_j \) in the whole formula)
- Further, by relabelling assume \( C_i = (x_1 \lor \cdots \lor x_\ell) \)

\[
\Rightarrow \quad \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[\text{ } y_j \text{ is false }] = 1 - \prod_{j=1}^{\ell} \left( 1 - y_j^* \right)
\]

Arithmetic vs. geometric mean:

\[
\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}.
\]

\[
\geq 1 - \left( \frac{\sum_{j=1}^{\ell} (1 - y_j^*)}{\ell} \right) ^\ell
\]

\[
= 1 - \left( 1 - \frac{\sum_{j=1}^{\ell} y_j^*}{\ell} \right) ^\ell \geq 1 - \left( 1 - \frac{z_i^*}{\ell} \right) ^\ell.
\]
Lemma

For any clause $i$ of length $\ell$, 

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z_i^*.$$ 

Proof of Lemma (2/2):

- So far we have shown:

$$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^{\ell}.$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z_i^*.$$

Proof of Lemma (2/2):

- So far we have shown:

  $$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^{\ell}$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^{\ell}$.
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$, 

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$ 

Proof of Lemma (2/2):

- So far we have shown:

$$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a concave function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$. 

Analysis of Randomised Rounding

Lemma
For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$  

Proof of Lemma (2/2):

- So far we have shown:

$$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a concave function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.  

\[ g(z) \begin{cases} 1 - (1 - \frac{1}{3})^3 & z = 1 \\ \text{linearly increasing} & 0 < z < 1 \end{cases} \]
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$ 

Proof of Lemma (2/2):

- So far we have shown:

$$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a concave function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$.

$$g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

$g(z)$

0 1
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$ 

Proof of Lemma (2/2):

- So far we have shown:

$$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^\ell$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell$. This is a concave function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_\ell$. 

$$\Rightarrow g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

- Therefore, $\Pr[\text{clause } i \text{ is satisfied}] \geq \beta_\ell \cdot z_i^*$. 

\[ \begin{array}{c}
\text{g(z)} \\
\downarrow \\
0 \quad 1 \quad \text{z} \\
\end{array} \]

VI. Randomisation and Rounding
Analysis of Randomised Rounding

**Lemma**

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z_i^*.$$

**Proof of Lemma (2/2):**

- So far we have shown:

  $$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i^*}{\ell}\right)^{\ell}$$

- For any $\ell \geq 1$, define $g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^{\ell}$. This is a concave function with $g(0) = 0$ and $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^{\ell} =: \beta_\ell$.

  $$\Rightarrow g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

- Therefore, $\Pr[\text{clause } i \text{ is satisfied}] \geq \beta_\ell \cdot z_i^*$. $\square$
Analysis of Randomised Rounding

**Lemma**
For any clause $i$ of length $\ell$, 

$$
\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.
$$

**Theorem**
Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.
Lemma

For any clause $i$ of length $\ell$,\[
\mathbb{P}\left[\text{clause } i \text{ is satisfied}\right] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z^*_i.
\]

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:
Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \ldots, m$, let $\ell_i$ be the corresponding length.
Analysis of Randomised Rounding

**Lemma**

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z^*_i.$$

**Theorem**

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

**Proof of Theorem:**

- For any clause $i = 1, 2, \ldots, m$, let $\ell_i$ be the corresponding length.
- Then the expected number of satisfied clauses is:

$$E[Y] = \sum_{i=1}^{m} E[Y_i] \geq$$
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \ldots, m$, let $\ell_i$ be the corresponding length.
- Then the expected number of satisfied clauses is:

$$E[\mathbf{Y}] = \sum_{i=1}^{m} E[\mathbf{Y}_i] \geq \sum_{i=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot z_i^*$$

By Lemma
Analysis of Randomised Rounding

Lemma

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^\ell\right) \cdot z_i^*.$$

Theorem

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

Proof of Theorem:

- For any clause $i = 1, 2, \ldots, m$, let $\ell_i$ be the corresponding length.
- Then the expected number of satisfied clauses is:

$$\mathbb{E}[Y] = \sum_{i=1}^{m} \mathbb{E}[Y_i] \geq \sum_{i=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_i}\right)^\ell_i\right) \cdot z_i^* \geq \sum_{i=1}^{m} \left(1 - \frac{1}{e}\right) \cdot z_i^*$$

  By Lemma  
  Since $(1 - 1/x)^x \leq 1/e$
Analysis of Randomised Rounding

**Lemma**

For any clause $i$ of length $\ell$,

$$\Pr[\text{clause } i \text{ is satisfied}] \geq \left( 1 - \left( 1 - \frac{1}{\ell} \right) ^ {\ell} \right) \cdot z_i^*.$$

**Theorem**

Randomised Rounding yields a $1/(1 - 1/e) \approx 1.5820$ randomised approximation algorithm for MAX-CNF.

**Proof of Theorem:**

- For any clause $i = 1, 2, \ldots, m$, let $\ell_i$ be the corresponding length.
- Then the expected number of satisfied clauses is:

$$E[Y] = \sum_{i=1}^{m} E[Y_i] \geq \sum_{i=1}^{m} \left( 1 - \left( 1 - \frac{1}{\ell_i} \right) ^ {\ell_i} \right) \cdot z_i^* \geq \sum_{i=1}^{m} \left( 1 - \frac{1}{e} \right) \cdot z_i^* \geq \left( 1 - \frac{1}{e} \right) \cdot \text{OPT}$$

By Lemma

Since $(1 - 1/x)^x \leq 1/e$

LP solution at least as good as optimum
Approach 3: Hybrid Algorithm

Summary

- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

```plaintext
HYBRID-MAX-CNF(ϕ, n, m)
1: Let b ∈ {0, 1} be the flip of a fair coin
2: If b = 0 then perform random guessing
3: If b = 1 then perform randomised rounding
4: return the computed solution
```

Algorithm sets each variable $x_i$ to TRUE with prob. $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot y^*_i$. Note, however, that variables are not independently assigned!
Approach 3: Hybrid Algorithm

Summary
- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches
Approach 3: Hybrid Algorithm

Summary
- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches

\textbf{HYBRID-MAX-CNF}(\varphi, n, m)
1. Let $b \in \{0, 1\}$ be the flip of a fair coin
2. \textbf{If} $b = 0$ \textbf{then} perform random guessing
3. \textbf{If} $b = 1$ \textbf{then} perform randomised rounding
4. \textbf{return} the computed solution
Approach 3: Hybrid Algorithm

Summary
- Approach 1 (Guessing) achieves better guarantee on longer clauses
- Approach 2 (Rounding) achieves better guarantee on shorter clauses

Idea: Consider a hybrid algorithm which interpolates between the two approaches

HYBRID-MAX-CNF(\(\varphi, n, m\))

1. Let \(b \in \{0, 1\}\) be the flip of a fair coin
2. If \(b = 0\) then perform random guessing
3. If \(b = 1\) then perform randomised rounding
4. return the computed solution

Algorithm sets each variable \(x_i\) to TRUE with prob. \(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot y_i^*\).
Note, however, that variables are not independently assigned!
Analysis of Hybrid Algorithm

**Theorem**

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.
Analysis of Hybrid Algorithm

Theorem

\textbf{HYBRID-MAX-CNF}(\varphi, n, m) is a randomised 4/3-approx. algorithm.

Proof:
Analysis of Hybrid Algorithm

**Theorem**

\textsc{Hybrid-Max-Cnf}(\varphi, n, m) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause $i$ is satisfied with probability at least $3/4 \cdot z_i^*$
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
Theorem

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
Analysis of Hybrid Algorithm

**Theorem**

\[ \text{HYBRID-MAX-CNF}(\varphi, n, m) \] is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \( i \) is satisfied with probability at least \( 3/4 \cdot z_i^* \)
- For any clause \( i \) of length \( \ell \):
  - Algorithm 1 satisfies it with probability \( 1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^* \).
  - Algorithm 2 satisfies it with probability \( \beta_\ell \cdot z_i^* \).
Analysis of Hybrid Algorithm

**Theorem**

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
Analysis of Hybrid Algorithm

**Theorem**

\[ \text{HYBRID-MAX-CNF}(\varphi, n, m) \text{ is a randomised } 4/3\text{-approx. algorithm.} \]

**Proof:**

- It suffices to prove that clause \( i \) is satisfied with probability at least \( 3/4 \cdot z_i^* \).
- For any clause \( i \) of length \( \ell \):
  - **Algorithm 1** satisfies it with probability \( 1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^* \).
  - **Algorithm 2** satisfies it with probability \( \beta_\ell \cdot z_i^* \).
  - **HYBRID-MAX-CNF(\varphi, n, m)** satisfies it with probability \( \frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^* \).
- Note \( \frac{\alpha_\ell + \beta_\ell}{2} = 3/4 \) for \( \ell \in \{1, 2\} \),
Analysis of Hybrid Algorithm

**Theorem**

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure)
Analysis of Hybrid Algorithm

**Theorem**

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure)
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure)
Analysis of Hybrid Algorithm

**Theorem**

**HYBRID-MAX-CNF(\(\varphi, n, m\))** is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - **HYBRID-MAX-CNF(\(\varphi, n, m\))** satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure)
Analysis of Hybrid Algorithm

**Theorem**

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

**Proof:**

- It suffices to prove that clause \(i\) is satisfied with probability at least 3/4 \(z_i^*\).
- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).
- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure)
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(\(\varphi, n, m\)) is a randomised 4/3-approx. algorithm.

Proof:

- It suffices to prove that clause \(i\) is satisfied with probability at least \(3/4 \cdot z_i^*\).

- For any clause \(i\) of length \(\ell\):
  - Algorithm 1 satisfies it with probability \(1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*\).
  - Algorithm 2 satisfies it with probability \(\beta_\ell \cdot z_i^*\).
  - HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with probability \(\frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^*\).

- Note \(\frac{\alpha_\ell + \beta_\ell}{2} = 3/4\) for \(\ell \in \{1, 2\}\), and for \(\ell \geq 3\), \(\frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4\) (see figure).

\(\Rightarrow\) HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with prob. at least \(3/4 \cdot z_i^*\) \(\square\)
Since $\alpha_2 = \beta_2 = 3/4$, we cannot achieve a better approximation ratio than 4/3 by combining Algorithm 1 & 2 in a different way.

The 4/3-approximation algorithm can be easily derandomised.

- Idea: use the conditional expectation trick for both Algorithm 1 & 2 and output the better solution.

- The 4/3-approximation algorithm applies unchanged to a weighted version of MAX-CNF, where each clause has a non-negative weight.

- Even MAX-2-CNF (every clause has length 2) is NP-hard!
Exercise (easy): Consider any minimisation problem, where $x$ is the optimal cost of the LP relaxation, $y$ is the optimal cost of the IP and $z$ is the solution obtained by rounding up the LP solution. Which of the following statements are true?

1. $x \leq y \leq z$,
2. $y \leq x \leq z$,
3. $y \leq z \leq x$. 
Exercise (trickier): Consider a version of the SET-COVER problem, where each element \( x \in X \) has to be covered by \textbf{at least two} subsets. Design and analyse an efficient approximation algorithm.  

\textbf{Hint:} You may use the result that if \( X_1, X_2, \ldots, X_n \) are independent Bernoulli random variables with \( X := \sum_{i=1}^{n} X_i, \mathbb{E}[X] \geq 2 \), then

\[
\Pr[X \geq 2] \geq \frac{1}{4} \cdot (1 - e^{-1}).
\]
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
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
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
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

VI. Randomisation and Rounding

Conclusion
Spectrum of Approximations

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

- FPTAS
- PTAS
- APX
- log-APX
- poly-APX
Spectrum of Approximations

SCHEDULING, EUCLIDEAN-TSP
KNAPSACK SUBSET-SUM
FPTAS PTAS APX
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
Spectrum of Approximations

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

- FPTAS
- PTAS
- APX
- log-APX

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

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

I. Sorting and Counting Networks
   - 0/1-Sorting Principle, Bitonic Sorting, Batcher’s Sorting Network
     Bonus Material: A Glimpse at the AKS network
   - Balancing Networks, Counting Network Construction, Counting vs. Sorting

II. Linear Programming
   - Geometry of Linear Programs, Applications of Linear Programming
   - Simplex Algorithm, Finding a Feasible Initial Solution
   - Fundamental Theorem of Linear Programming

III. Approximation Algorithms: Covering Problems
   - Intro to Approximation Algorithms, Definition of PTAS and FPTAS
   - (Unweighted) Vertex-Cover: 2-approx. based on Greedy
   - (Unweighted) Set-Cover: \( O(\log n) \)-approx. based on Greedy

IV. Approximation Algorithms via Exact Algorithms
   - Subset-Sum: FPTAS based on Trimming and Dynamic Programming
   - Scheduling: 2-approx. based on Simple Greedy, 4/3-approx. using LPT
     Bonus Material: A PTAS for Machine Scheduling based on Rounding and Dynamic Programming

V. The Travelling Salesman Problem
   - Inapproximability of the General TSP problem
   - Metric TSP: 2-approx. based on MST, 3/2-approx. based on MST + matching

VI. Approximation Algorithms: Rounding and Randomisation
   - MAX3-CNF: 8/7-approx. based on Guessing, Derandomisation with Greedy
   - (Weighted) Vertex-Cover: 2-approx. based on Deterministic Rounding
   - (Weighted) Set-Cover: \( O(\log n) \)-approx. based on Randomised Rounding
   - MAX-CNF: 4/3-approx. based on Guessing + Randomised Rounding
Thank you and Best Wishes for the Exam!