VI. Approx. Algorithms: Randomisation and Rounding

Thomas Sauerwald

Easter 2021

UNIVERSITY OF CAMBRIDGE

Outline

Performance Ratios for Randomised Approximation Algorithms

Approximation Ratio

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

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

Performance Ratios for Randomised Approximation Algorithms

Approximation Ratio

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

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

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

Performance Ratios for Randomised Approximation Algorithms

Approximation Ratio

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

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

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

Approximation Schemes

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

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

Performance Ratios for Randomised Approximation Algorithms

Approximation Ratio

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

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

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

extends in the natural way to randomised algorithms

Approximation Schemes

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

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

Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion

MAX-3-CNF Satisfiability

MAX-3-CNF Satisfiability

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

MAX-3-CNF Satisfiability

MAX-3-CNF Satisfiability

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

MAX-3-CNF Satisfiability

MAX-3-CNF Satisfiability

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

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

MAX-3-CNF Satisfiability

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.

MAX-3-CNF Satisfiability

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.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- 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:

$$(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

MAX-3-CNF Satisfiability

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.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- 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:

$$(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

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

MAX-3-CNF Satisfiability

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.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- 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:

$$(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

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

Idea: What about assigning each variable uniformly and independently at random?

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.

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:

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:

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

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,

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,

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

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:

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

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

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

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

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:

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

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

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

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

$$\Rightarrow \qquad \mathbf{E}\,[\,Y_i\,] = \mathbf{Pr}\,[\,Y_i = 1\,] \cdot 1 = \frac{7}{8}.$$

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:

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

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

$$\begin{aligned} \mathbf{Pr}[\,\text{clause } i \text{ is not satisfied}\,] &= \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \\ \Rightarrow \quad \mathbf{Pr}[\,\text{clause } i \text{ is satisfied}\,] &= 1 - \frac{1}{8} = \frac{7}{8} \\ \Rightarrow \quad \mathbf{E}[\,Y_i\,] = \mathbf{Pr}[\,Y_i = 1\,] \cdot 1 &= \frac{7}{8}. \end{aligned}$$

- 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:

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

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

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

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

$$\Rightarrow \qquad \mathbf{E}\left[\,Y_i\,\right] = \mathbf{Pr}\left[\,Y_i = 1\,\right] \cdot 1 = \frac{7}{8}.$$

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

$$\mathbf{E}\left[\,Y\,\right]$$

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:

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

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

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

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

$$\Rightarrow \qquad \mathbf{E}[\,Y_i\,] = \mathbf{Pr}[\,Y_i = 1\,] \cdot 1 = \frac{7}{8}.$$

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

$$\mathbf{E}[\,Y\,] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right]$$

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,
$$\mathbf{Pr}[\,\text{clause } i \text{ is not satisfied}\,] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$
$$\Rightarrow \qquad \mathbf{Pr}[\,\text{clause } i \text{ is satisfied}\,] = 1 - \frac{1}{8} = \frac{7}{8}$$
$$\Rightarrow \qquad \mathbf{E}[\,Y_i\,] = \mathbf{Pr}[\,Y_i = 1\,] \cdot 1 = \frac{7}{8}.$$
- Let $Y := \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then,
$$\mathbf{E}[\,Y\,] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right]$$

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,
$$\mathbf{Pr}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$
$$\Rightarrow \quad \mathbf{Pr}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$
$$\Rightarrow \quad \mathbf{E}[Y_i] = \mathbf{Pr}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$
- Let $Y := \sum_{i=1}^{m} Y_i$ be the number of satisfied clauses. Then,
$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbf{E}[Y_i]$$

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,

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

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

$$\Rightarrow \qquad \mathbf{E}[\,Y_i\,] = \mathbf{Pr}[\,Y_i = 1\,] \cdot 1 = \frac{7}{8}.$$

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

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

Linearity of Expectations

Analysis

Theorem 35.6

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

Proof:

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

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

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

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

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

$$\Rightarrow \qquad \mathbf{E}\left[\,Y_i\,\right] = \mathbf{Pr}\left[\,Y_i = 1\,\right] \cdot 1 = \frac{7}{8}.$$

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

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

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:

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

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

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

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

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

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

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

Linearity of Expectations

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:

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

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

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

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

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

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

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

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 $8/7$-approximation algorithm.

Interesting Implications

Theorem 35.6

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

Corollary

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

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

Interesting Implications

Theorem 35.6

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

Corollary

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

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

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 assigment which satisfies at least $\frac{7}{8}$ of all clauses.

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

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 assigment which satisfies at least $\frac{7}{8}$ of all clauses.

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

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.

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.

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 $1/(8m)$

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

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 $1/(8m)$

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

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[\,Y\,]$!

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 $1/(8m)$

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

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[Y]$!

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 $x_1, x_2, \dots, x_n$ 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 $1/(8m)$

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

Y is defined as in the previous proof.

One of the two conditional expectations is at least $\mathbf{E}[\,Y\,]$!

GREEDY-3-CNF(ϕ, n, m)

for $j = 1, 2, \dots, n$
    Compute $\mathbf{E}[\,Y \mid x_1 = v_1 \dots, x_{j-1} = v_{j-1}, x_j = 1\,]$
    Compute $\mathbf{E}[\,Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 0\,]$
    Let $x_j = v_j$ so that the conditional expectation is maximized
return the assignment $v_1, v_2, \dots, v_n$

Analysis of GREEDY-3-CNF(ϕ, n, m)

Theorem

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

- **Step 1:** polynomial-time algorithm
  - In iteration $j = 1, 2, \ldots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

computable in $O(1)$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

computable in $O(1)$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

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

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

$$\begin{aligned}\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \,\right] &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \,\right] \\ &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \,\right]\end{aligned}$$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

$$\begin{aligned}\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \,\right] &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \,\right] \\ &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \,\right] \\ &\;\;\vdots \\ &\geq \mathbf{E}\left[\, Y \,\right]\end{aligned}$$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

$$\begin{aligned}\mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j\,\right] &\geq \mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}\,\right] \\ &\geq \mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2}\,\right] \\ &\;\;\vdots \\ &\geq \mathbf{E}\left[\,Y\,\right] = \frac{7}{8} \cdot m.\end{aligned}$$

This algorithm is deterministic.

Theorem

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

Proof:

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

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

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

$$\begin{aligned}
\mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j\,\right] &\geq \mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}\,\right] \\
&\geq \mathbf{E}\left[\,Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2}\,\right] \\
&\;\;\vdots \\
&\geq \mathbf{E}\left[\,Y\,\right] = \frac{7}{8} \cdot m.
\end{aligned}$$

Analysis of GREEDY-3-CNF(ϕ, n, m)

This algorithm is deterministic.

Theorem

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

Proof:

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

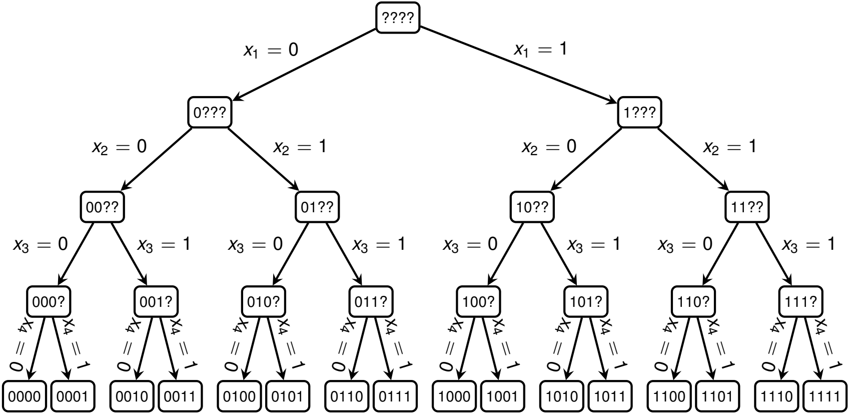
$$\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \,\right] = \sum_{i=1}^{m} \mathbf{E}\left[\, Y_i \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = 1 \,\right]$$

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

$$\begin{aligned}\mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1}, x_j = v_j \,\right] &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-1} = v_{j-1} \,\right] \\ &\geq \mathbf{E}\left[\, Y \mid x_1 = v_1, \ldots, x_{j-2} = v_{j-2} \,\right] \\ &\vdots \\ &\geq \mathbf{E}\left[\, Y \,\right] = \frac{7}{8} \cdot m. \qquad \square\end{aligned}$$

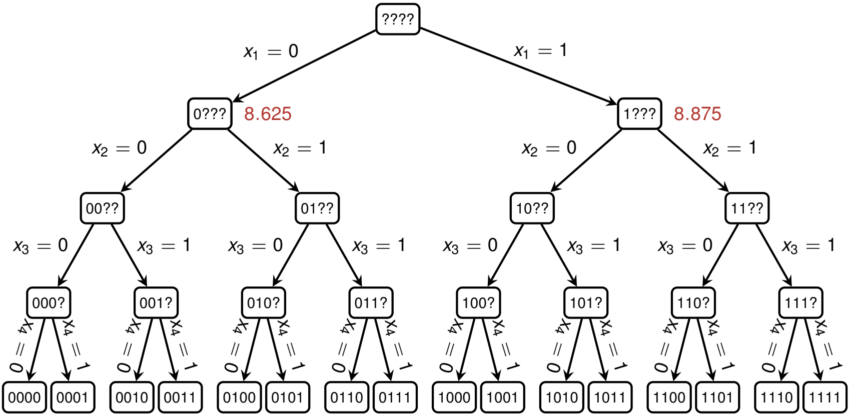
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



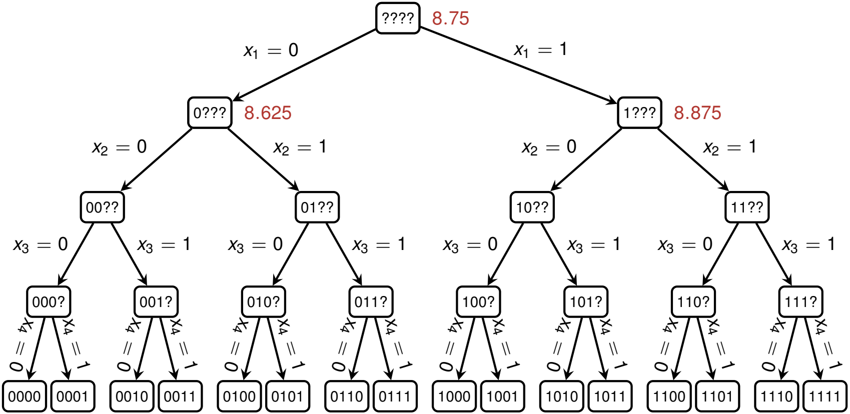
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



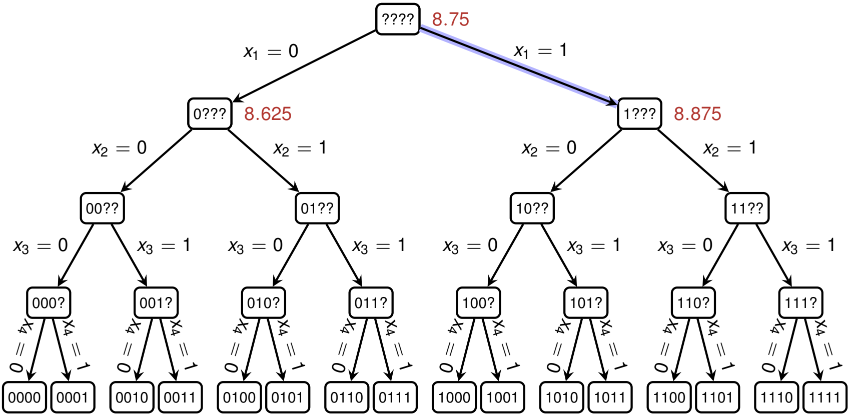
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



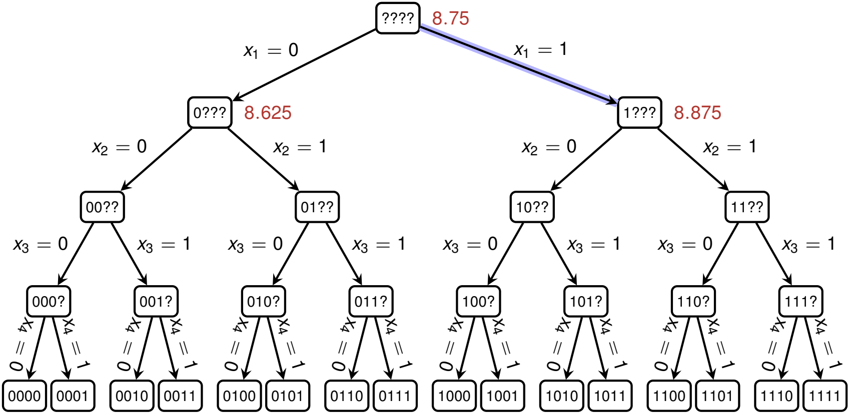
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



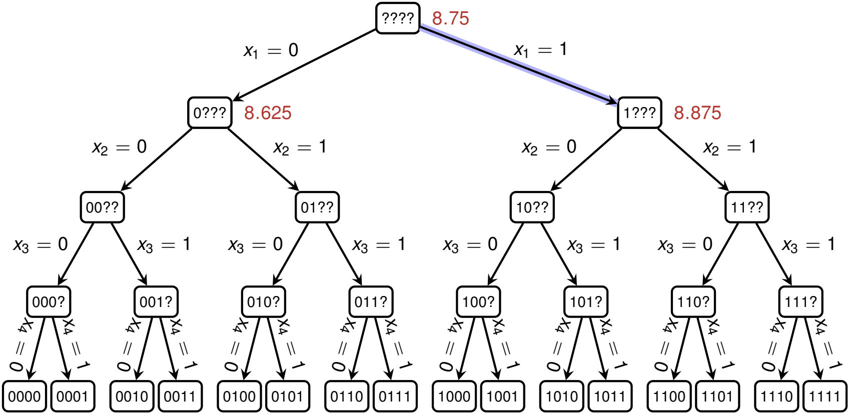
Run of GREEDY-3-CNF(φ, n, m)

$\cancel{(x_1 \vee x_2 \vee x_3)} \wedge \cancel{(x_1 \vee \overline{x_2} \vee \overline{x_4})} \wedge \cancel{(x_1 \vee x_2 \vee \overline{x_4})} \wedge (\cancel{\overline{x_1}} \vee \overline{x_3} \vee x_4) \wedge \cancel{(x_1 \vee x_2 \vee \overline{x_3})} \wedge (\cancel{\overline{x_1}} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\cancel{\overline{x_1}} \vee x_2 \vee x_3) \wedge (\cancel{\overline{x_1}} \vee \overline{x_2} \vee x_3) \wedge \cancel{(x_1 \vee x_3 \vee \overline{x_4})} \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



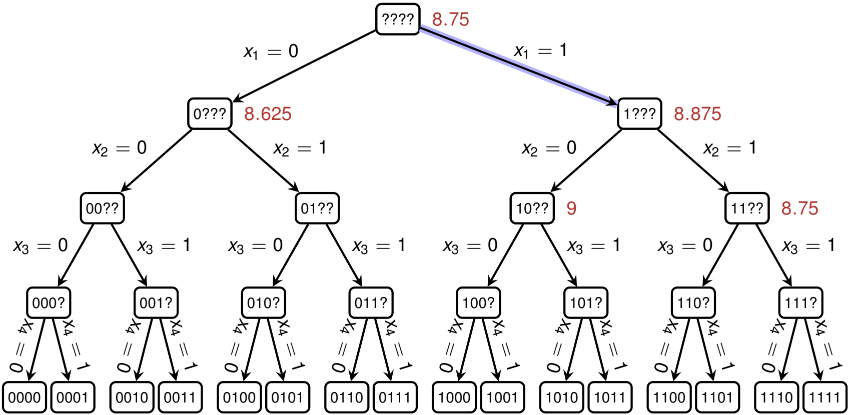
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



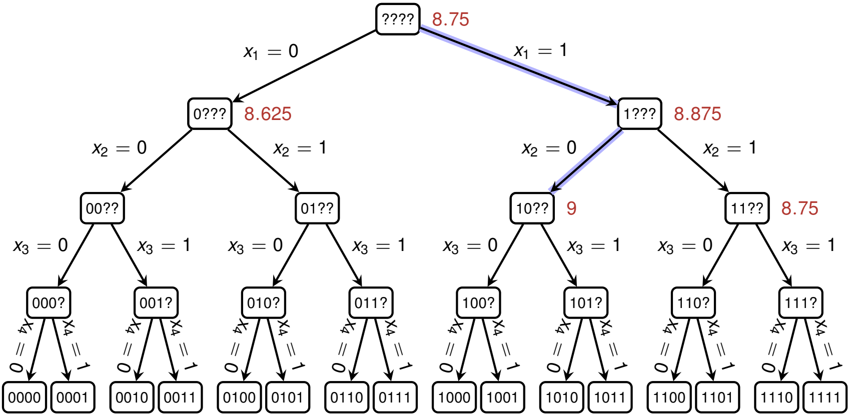
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



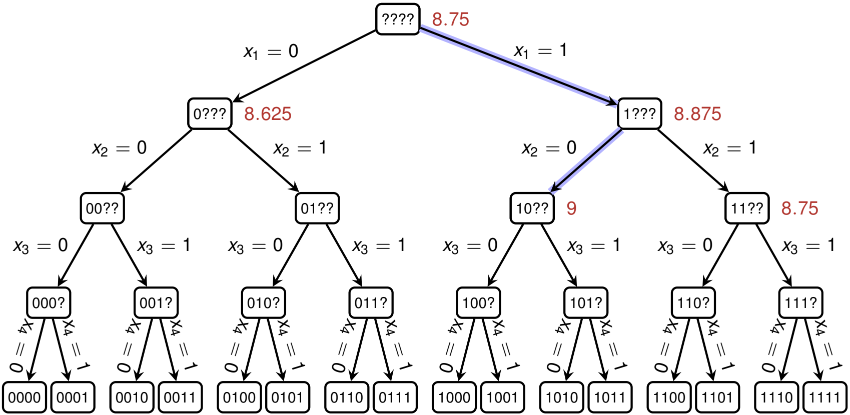
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



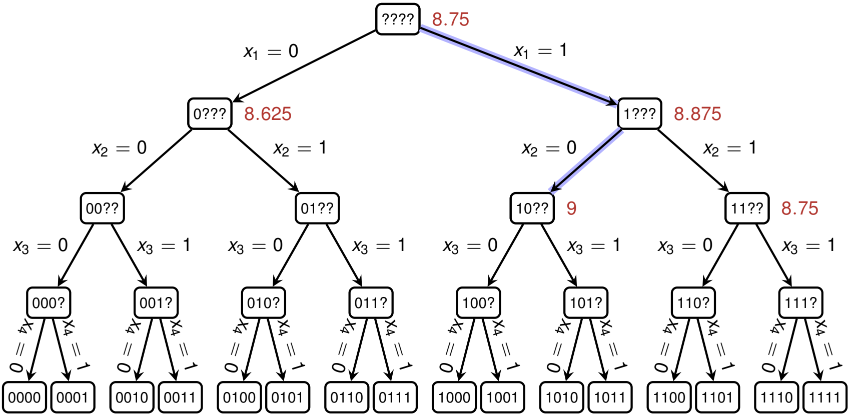
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



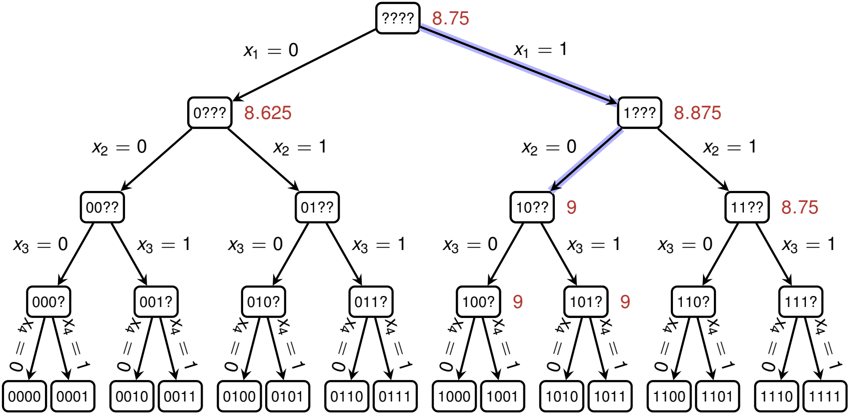
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



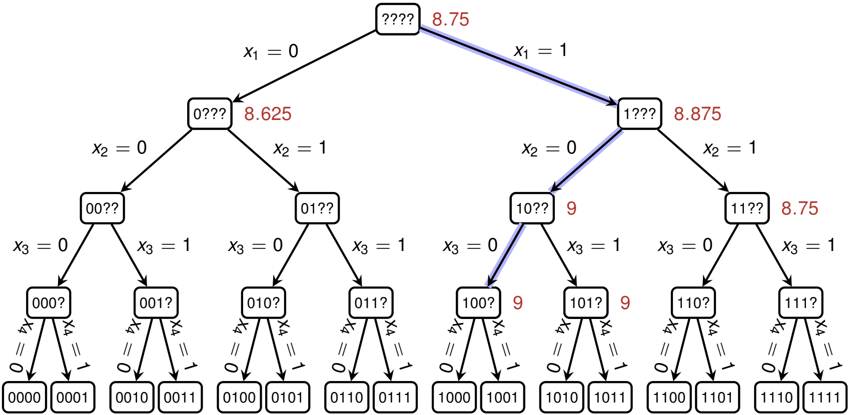
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



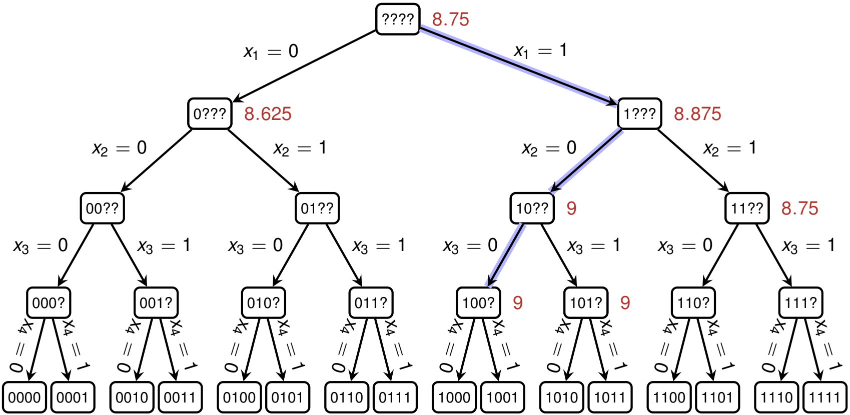
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



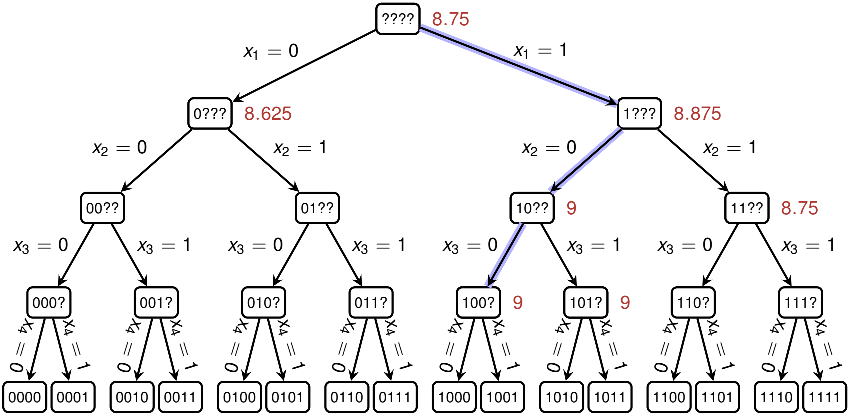
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



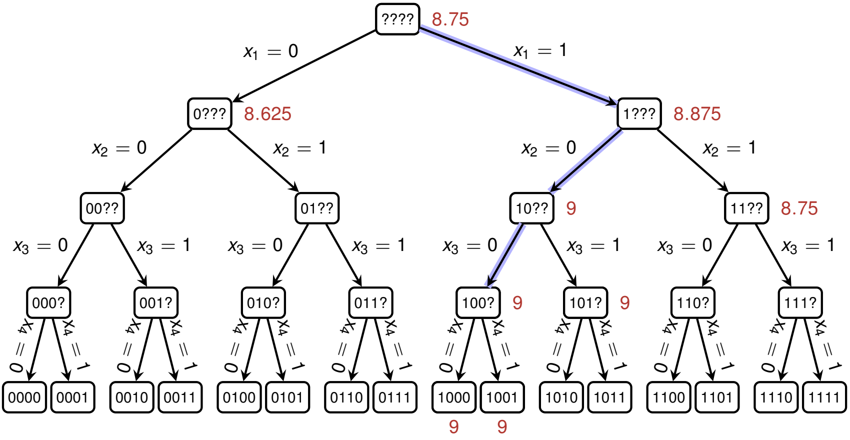
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



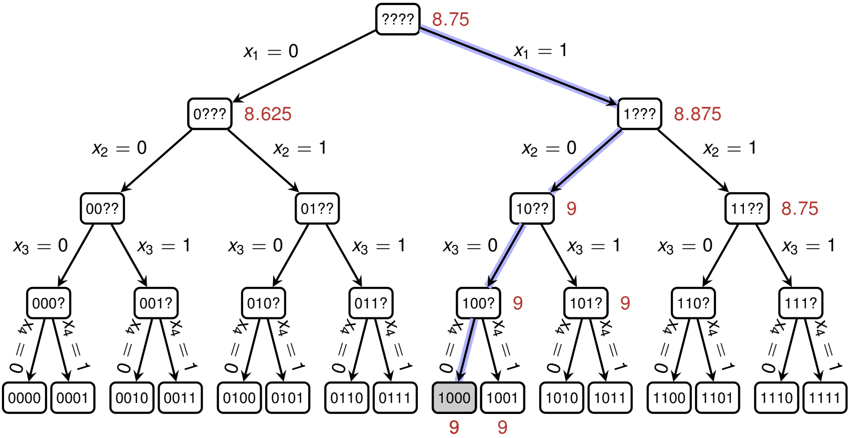
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



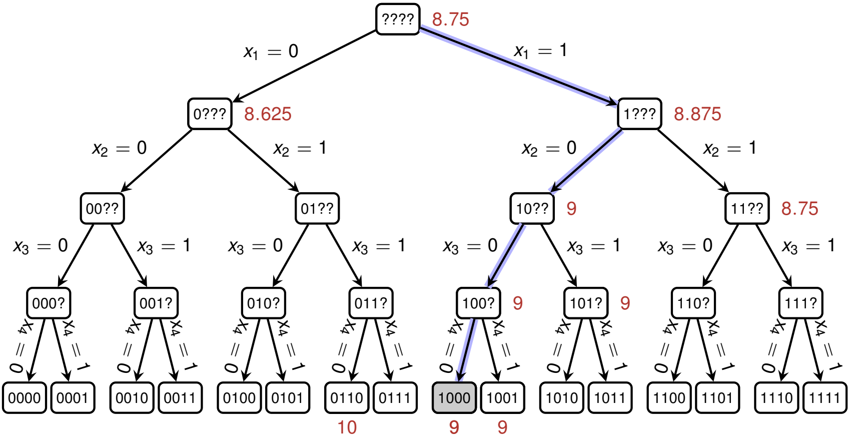
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



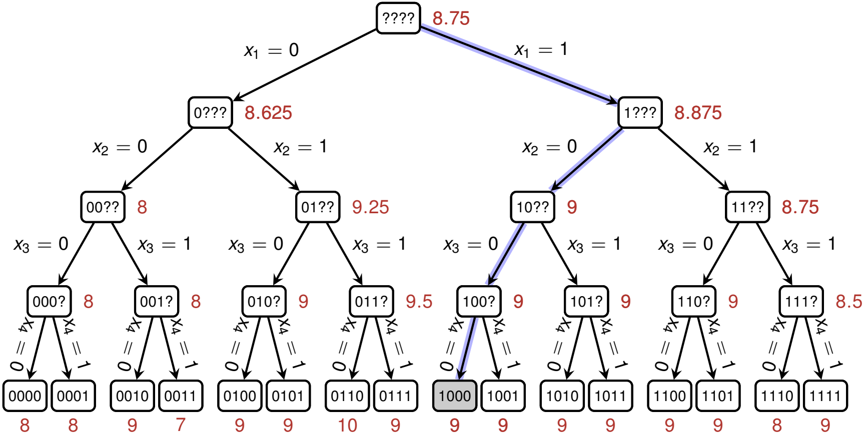
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



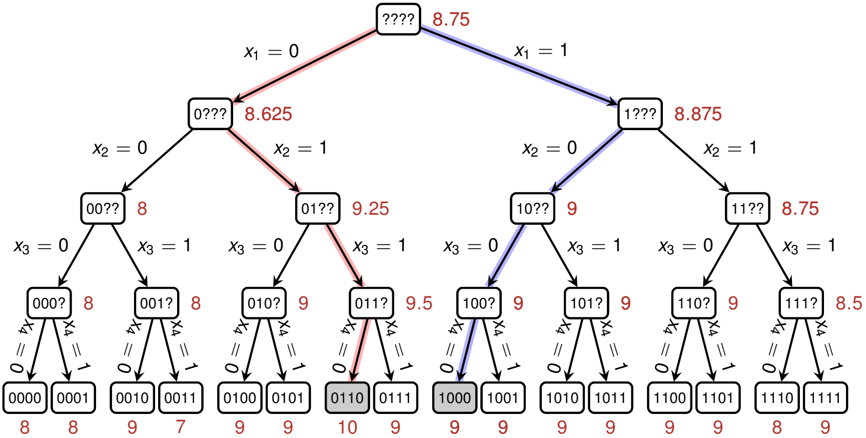
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



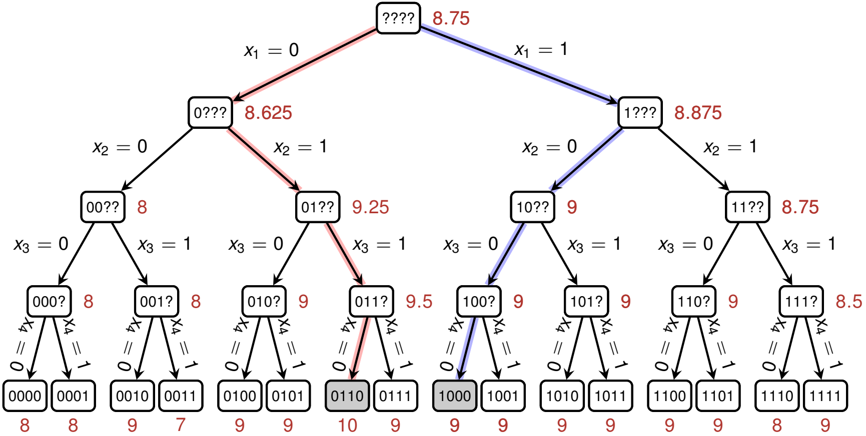
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



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

MAX-3-CNF: Concluding Remarks

Theorem 35.6

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

MAX-3-CNF: Concluding Remarks

Theorem 35.6

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

Theorem

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

MAX-3-CNF: Concluding Remarks

Theorem 35.6

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

Theorem

GREEDY-3-CNF(ϕ, 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.

MAX-3-CNF: Concluding Remarks

Theorem 35.6

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

Theorem

GREEDY-3-CNF(ϕ, 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.

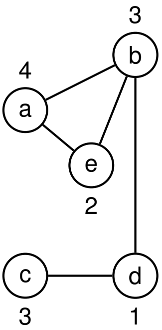
Essentially there is nothing smarter than just guessing!

Outline

The Weighted Vertex-Cover Problem

Vertex Cover Problem

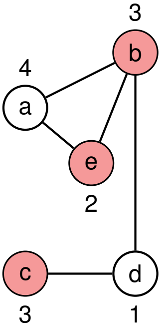
- Given: Undirected, vertex-weighted graph $G = (V, E)$
- Goal: Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.



The Weighted Vertex-Cover Problem

Vertex Cover Problem

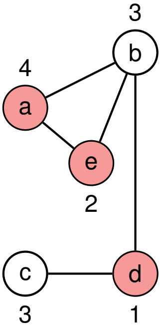
- Given: Undirected, vertex-weighted graph $G = (V, E)$
- Goal: Find a minimum-weight subset $V' \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V'$ or $v \in V'$.



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 $v \in V'$.

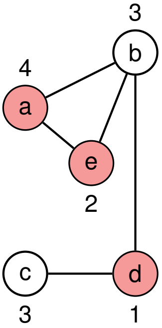


The Weighted Vertex-Cover Problem

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 $v \in V'$.

This is (still) an NP-hard problem.

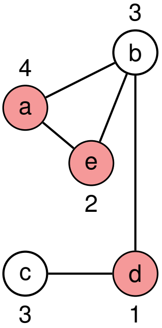


The Weighted Vertex-Cover Problem

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.



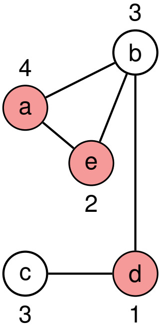
Applications:

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:

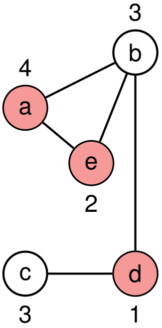
- Every edge forms a task, and every vertex represents a person/machine which can execute that task

The Weighted Vertex-Cover Problem

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:

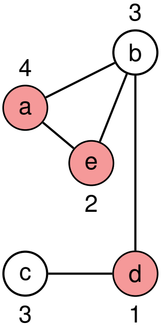
- Every edge forms a task, and every vertex represents a person/machine which can execute that task
- Weight of a vertex could be salary of a person

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

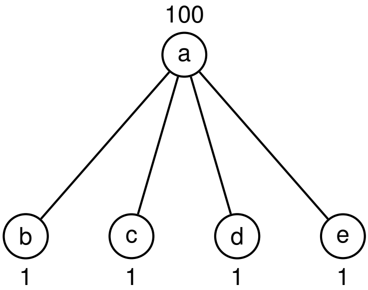
APPROX-VERTEX-COVER(G)

```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

The Greedy Approach from (Unweighted) Vertex Cover

APPROX-VERTEX-COVER(G)

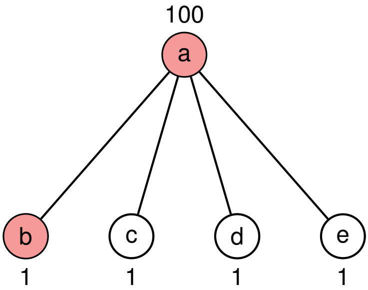
```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```



The Greedy Approach from (Unweighted) Vertex Cover

APPROX-VERTEX-COVER(G)

```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

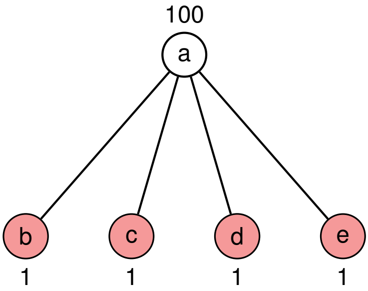


Computed solution has weight 101

The Greedy Approach from (Unweighted) Vertex Cover

APPROX-VERTEX-COVER(G)

$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let (u, v) be an arbitrary edge of E'
    $C = C \cup \{u, v\}$
    remove from E' every edge incident on either u or v
return C



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.

0-1 Integer Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u) + x(v) & \geq & 1 & \text{for each } (u,v) \in E \\ & x(v) & \in & \{0,1\} & \text{for each } v \in V \end{array}$$

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$
\begin{array}{llcll}
\text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\
\text{subject to} & x(u) + x(v) & \geq & 1 & \text{for each } (u,v) \in E \\
 & x(v) & \in & \{0,1\} & \text{for each } v \in V
\end{array}
$$

Linear Program

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

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{llll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & \\ \text{subject to} & x(u) + x(v) \geq 1 & & \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} & & \text{for each } v \in V \end{array}$$

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

Linear Program

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

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u) + x(v) & \geq & 1 & \text{for each } (u,v) \in E \\ & x(v) & \in & \{0,1\} & \text{for each } v \in V \end{array}$$

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

Linear Program

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

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

The Algorithm

```
APPROX-MIN-WEIGHT-VC(G, w)
C = ∅
compute x̄, an optimal solution to the linear program
for each v ∈ V
    if x̄(v) ≥ 1/2
        C = C ∪ {v}
return C
```

The Algorithm

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

```
C = ∅
compute x̄, an optimal solution to the linear program
for each v ∈ V
    if x̄(v) ≥ 1/2
        C = C ∪ {v}
return C
```

Theorem 35.7

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

The Algorithm

```
APPROX-MIN-WEIGHT-VC(G, w)
C = ∅
compute x̄, an optimal solution to the linear program
for each v ∈ V
    if x̄(v) ≥ 1/2
        C = C ∪ {v}
return C
```

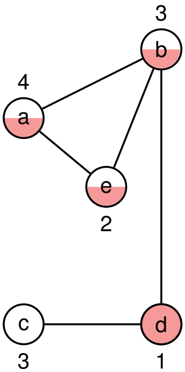
Theorem 35.7

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

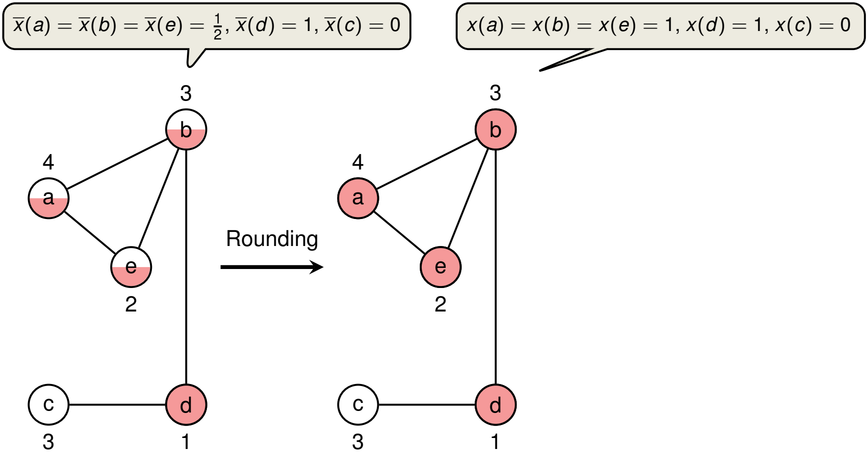
Example of APPROX-MIN-WEIGHT-VC

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



fractional solution of LP
with weight $= 5.5$

Example of APPROX-MIN-WEIGHT-VC



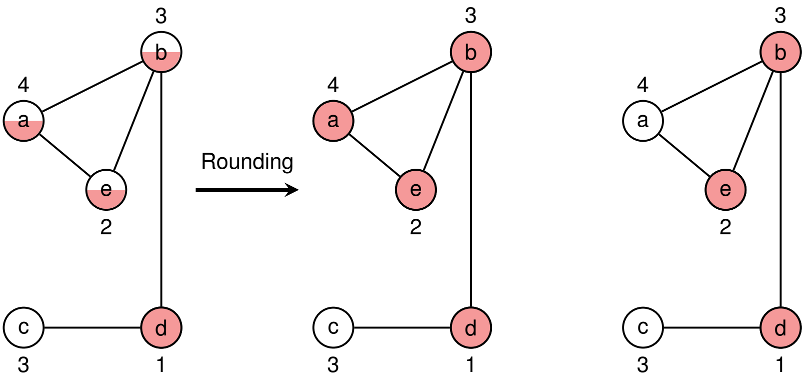
fractional solution of LP with weight $= 5.5$

rounded solution of LP with weight $= 10$

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

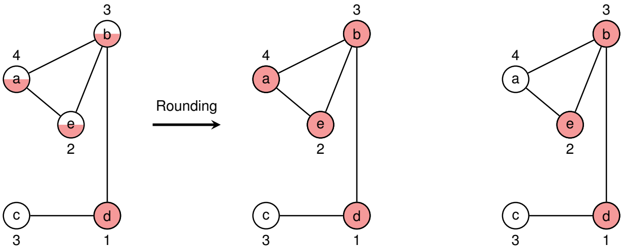
optimal solution
with weight = 6

Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

Approximation Ratio

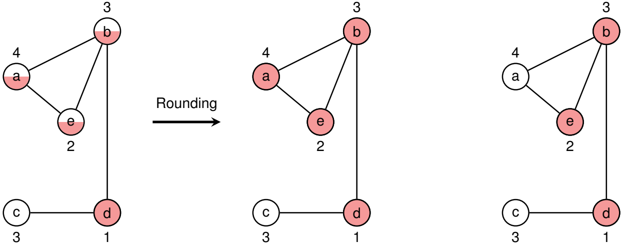
Proof (Approximation Ratio is 2 and Correctness):



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

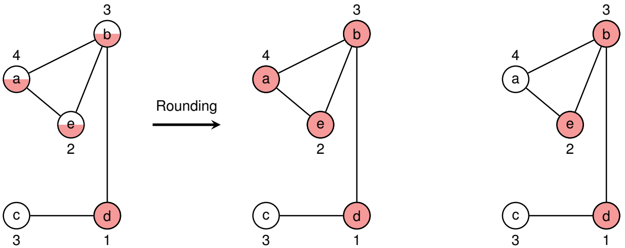
- Let C^* be an optimal solution to the minimum-weight vertex cover problem



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

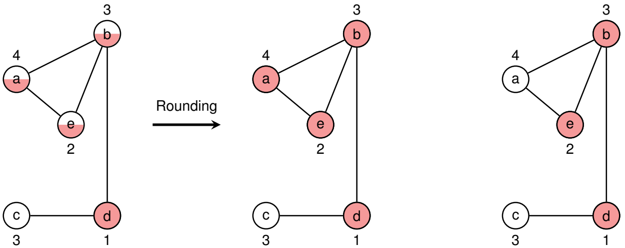


Approximation Ratio

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

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



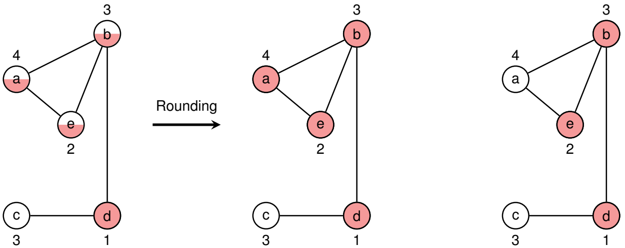
Approximation Ratio

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:



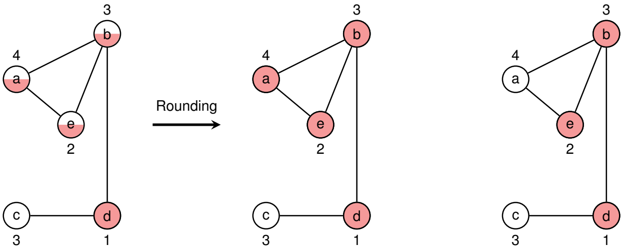
Approximation Ratio

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

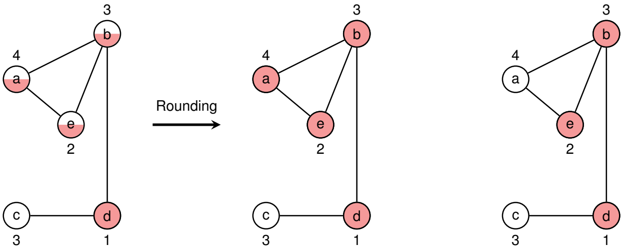
Proof (Approximation Ratio is 2 and Correctness):

- 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

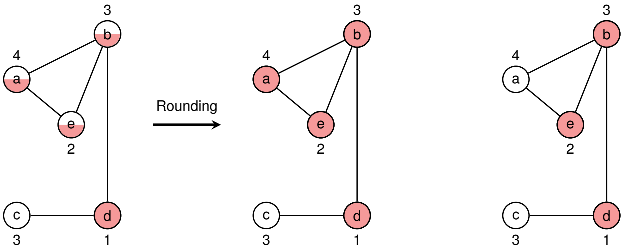
Proof (Approximation Ratio is 2 and Correctness):

- 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 $x(u) + x(v) \geq 1$

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



Approximation Ratio

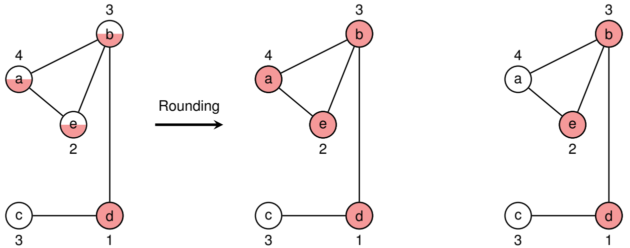
Proof (Approximation Ratio is 2 and Correctness):

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



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

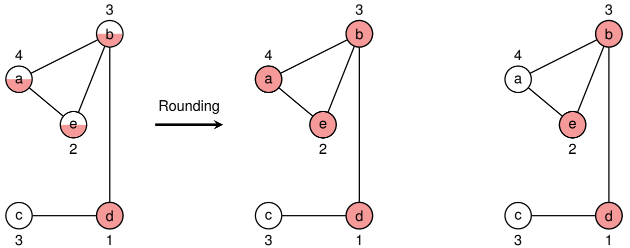
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

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

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

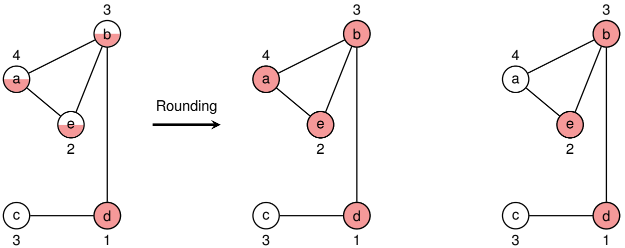
$$z^* \leq w(C^*)$$

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

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

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



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

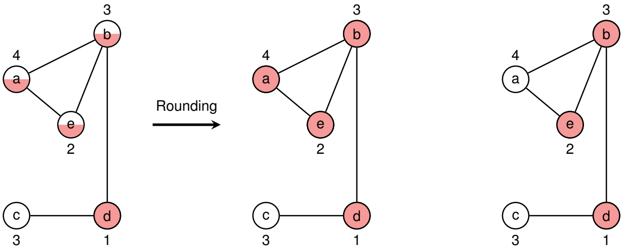
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

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

$$w(C^*) \geq z^* = \sum_{v \in V} w(v)\overline{x}(v)$$



Proof (Approximation Ratio is 2 and Correctness):

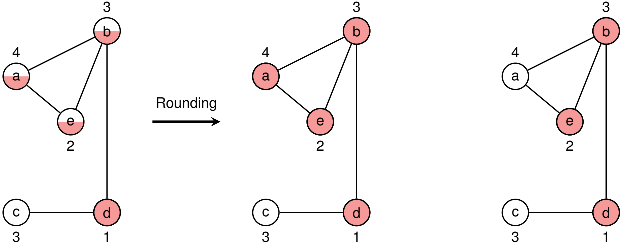
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

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

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



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

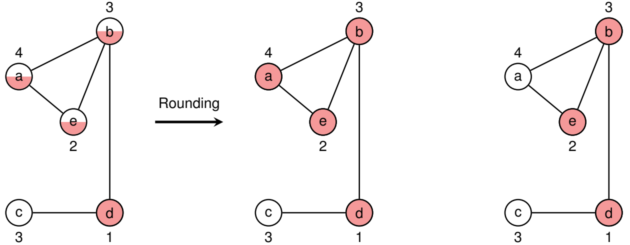
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

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

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



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

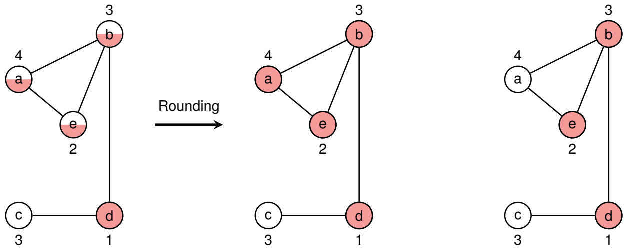
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

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

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



Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

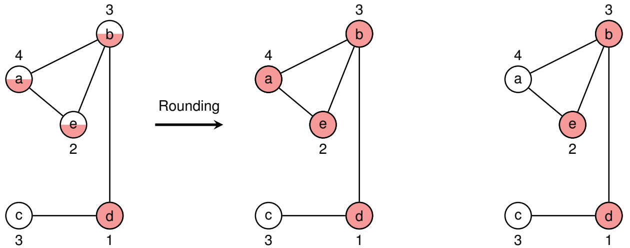
- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

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

- **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

The Weighted Set-Covering Problem

Set Cover Problem

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

$$\text{s.t.} \qquad X = \bigcup_{S \in \mathcal{C}} S.$$

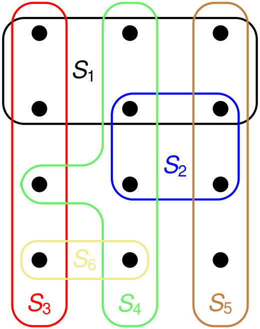
The Weighted Set-Covering Problem

Set Cover Problem

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

$$\text{s.t.} \qquad X = \bigcup_{S \in \mathcal{C}} S.$$

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



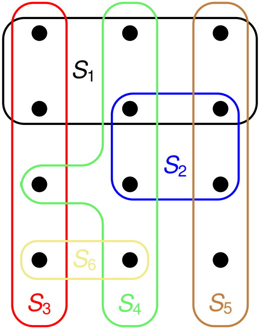
The Weighted Set-Covering Problem

Set Cover Problem

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

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

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



	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2

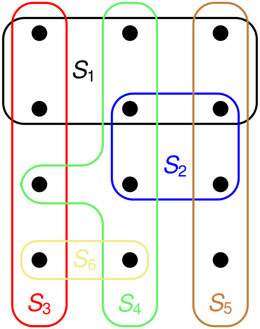
The Weighted Set-Covering Problem

Set Cover Problem

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

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

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



	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2

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

$$\begin{array}{llll} \text{minimize} & \sum_{S \in \mathcal{F}} c(S) y(S) & & \\ \text{subject to} & \sum_{S \in \mathcal{F} \colon x \in S} y(S) \geq 1 & & \text{for each } x \in X \\ & y(S) \in \{0,1\} & & \text{for each } S \in \mathcal{F} \end{array}$$

Setting up an Integer Program

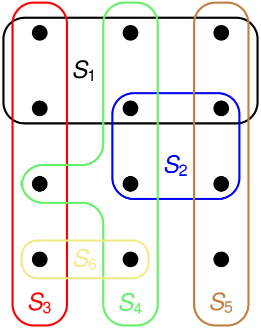
0-1 Integer Program

$$\begin{array}{llll} \text{minimize} & \displaystyle\sum_{S \in \mathcal{F}} c(S) y(S) & & \\ \text{subject to} & \displaystyle\sum_{S \in \mathcal{F}\,:\, x \in S} y(S) \geq 1 & & \text{for each } x \in X \\ & y(S) \in \{0,1\} & & \text{for each } S \in \mathcal{F} \end{array}$$

Linear Program

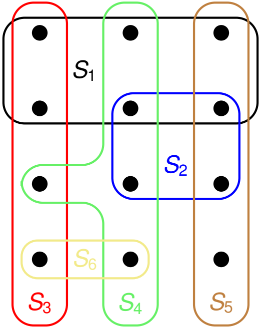
$$\begin{array}{llll} \text{minimize} & \displaystyle\sum_{S \in \mathcal{F}} c(S) y(S) & & \\ \text{subject to} & \displaystyle\sum_{S \in \mathcal{F}\,:\, x \in S} y(S) \geq 1 & & \text{for each } x \in X \\ & y(S) \in [0,1] & & \text{for each } S \in \mathcal{F} \end{array}$$

Back to the Example



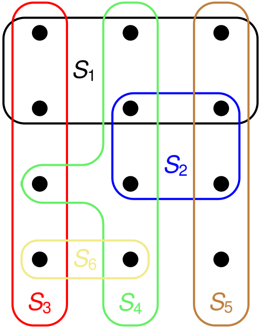
	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2

Back to the Example



	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	$1/2$

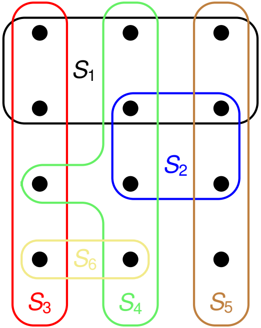
Back to the Example



	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	$1/2$

Cost equals 8.5

Back to the Example

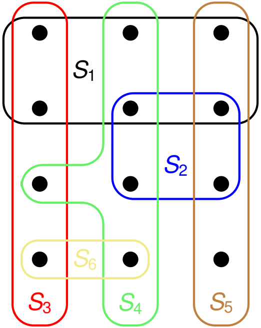


	S_1	S_2	S_3	S_4	S_5	S_6
c :	2	3	3	5	1	2
$y(.)$:	$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$

Cost equals 8.5

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

Back to the Example



	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	$1/2$

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!

Randomised Rounding

	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	$1/2$

Randomised Rounding

	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	$1/2$

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

Randomised Rounding

	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	$1/2$

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

Randomised Rounding

- Let $\mathcal{C} \subseteq \mathcal{F}$ 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:

$$\bar{y}(S) = \begin{cases} 1 & \text{with probability } y(S) \\ 0 & \text{otherwise.} \end{cases} \qquad \text{for all } S \in \mathcal{F}.$$

Randomised Rounding

	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	$1/2$

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

Randomised Rounding

- Let $\mathcal{C} \subseteq \mathcal{F}$ 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:

$$\bar{y}(S) = \begin{cases} 1 & \text{with probability } y(S) \\ 0 & \text{otherwise.} \end{cases} \qquad \text{for all } S \in \mathcal{F}.$$

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

Randomised Rounding

	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	$1/2$

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

Lemma

Randomised Rounding

	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	$1/2$

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

Lemma

- The expected cost satisfies

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

Randomised Rounding

	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	$1/2$

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

Lemma

- The expected cost satisfies

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

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

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

Proof of Lemma

Lemma

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

Proof of Lemma

Lemma

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

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

Proof:

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

$$\mathbf{E}\left[\, c(\mathcal{C}) \,\right]$$

Proof of Lemma

Lemma

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

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

Proof:

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

$$\mathbf{E}[\,c(\mathcal{C})\,] = \mathbf{E}\left[\sum_{S \in \mathcal{C}} c(S)\right]$$

Proof of Lemma

Lemma

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

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

Proof:

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

Proof of Lemma

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

Proof:

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

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

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

$$\mathbf{Pr}[x \notin \cup_{S \in \mathcal{C}} S]$$

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

$$\mathbf{Pr}[\, x \notin \cup_{S \in \mathcal{C}} S \,] = \prod_{S \in \mathcal{F}\colon\, x \in S} \mathbf{Pr}[\, S \notin \mathcal{C} \,]$$

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

$$\mathbf{Pr}[\, x \notin \cup_{S \in \mathcal{C}} S \,] = \prod_{S \in \mathcal{F}\colon x \in S} \mathbf{Pr}[\, S \notin \mathcal{C} \,] = \prod_{S \in \mathcal{F}\colon x \in S} (1 - y(S))$$

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

$$\mathbf{Pr}[\, x \notin \cup_{S \in \mathcal{C}} S \,] = \prod_{S \in \mathcal{F}\colon x \in S} \mathbf{Pr}[\, S \notin \mathcal{C} \,] = \prod_{S \in \mathcal{F}\colon x \in S} (1 - y(S))$$

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

y solves the LP!

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

y solves the LP!

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

y solves the LP!

Proof of Lemma

Lemma

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

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

Proof:

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

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

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

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

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

y solves the LP!

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

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

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

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

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

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

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

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

WEIGHTED SET COVER-LP$(X, \mathcal{F}, c)$

```
compute y, an optimal solution to the linear program
C = ∅
repeat 2 ln n times
    for each S ∈ F
        let C = C ∪ {S} with probability y(S)
return C
```

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

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

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

WEIGHTED SET COVER-LP$(X, \mathcal{F}, c)$

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

clearly runs in polynomial-time!

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

- **Step 1**: The probability that $\mathcal{C}$ is a cover

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

Analysis of Weighted Set Cover-LP

Theorem

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

Proof:

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

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S\in\mathcal{C}} S \,\right] =$$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$$\mathbf{Pr}\left[\, A \cup B \,\right] \leq \mathbf{Pr}\left[\, A \,\right] + \mathbf{Pr}\left[\, B \,\right]$$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[X = \cup_{S \in \mathcal{C}} S \right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{ x \notin \cup_{S \in \mathcal{C}} S \} \right]$$

$$\geq 1 - \sum_{x \in X} \mathbf{Pr}\left[x \notin \cup_{S \in \mathcal{C}} S \right]$$

$\mathbf{Pr}[A \cup B] \leq \mathbf{Pr}[A] + \mathbf{Pr}[B]$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

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

$\mathbf{Pr}[A \cup B] \leq \mathbf{Pr}[A] + \mathbf{Pr}[B]$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}[\, X = \cup_{S \in \mathcal{C}} S \,] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

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

$\mathbf{Pr}\left[\, A \cup B \,\right] \leq \mathbf{Pr}\left[\, A \,\right] + \mathbf{Pr}\left[\, B \,\right]$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S\in\mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x\in X} \{x \notin \cup_{S\in\mathcal{C}} S\}\right]$$

$\mathbf{Pr}[\,A \cup B\,] \leq \mathbf{Pr}[\,A\,] + \mathbf{Pr}[\,B\,]$

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

- **Step 2**: The expected approximation ratio

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}[\, X = \cup_{S \in \mathcal{C}} S \,] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\} \right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

- **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 $\Rightarrow$ $\mathbf{E}\left[\, c(\mathcal{C}) \,\right] \leq 2\ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S)$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

- **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 $\Rightarrow \mathbf{E}\left[\, c(\mathcal{C}) \,\right] \leq 2\ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2\ln(n) \cdot c(\mathcal{C}^*)$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\, \bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\} \,\right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

- **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 $\Rightarrow \mathbf{E}\left[\, c(\mathcal{C}) \,\right] \leq 2\ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2\ln(n) \cdot c(\mathcal{C}^*)$

Analysis of WEIGHTED SET COVER-LP

Theorem

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

Proof:

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

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

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

$$\mathbf{Pr}\left[\, X = \cup_{S \in \mathcal{C}} S \,\right] = 1 - \mathbf{Pr}\left[\bigcup_{x \in X} \{x \notin \cup_{S \in \mathcal{C}} S\}\right]$$

$\mathbf{Pr}[\, A \cup B \,] \leq \mathbf{Pr}[\, A \,] + \mathbf{Pr}[\, B \,]$

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

- **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 $\Rightarrow \mathbf{E}\left[\, c(\mathcal{C}) \,\right] \leq 2\ln(n) \cdot \sum_{S \in \mathcal{F}} c(S) \cdot y(S) \leq 2\ln(n) \cdot c(\mathcal{C}^*)$ □

Analysis of WEIGHTED SET COVER-LP

Theorem

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

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

Analysis of WEIGHTED SET COVER-LP

Theorem

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

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

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 $\mathcal{C}$ is a valid cover of X.
- The expected approximation ratio is $2\ln(n)$.

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

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 $\mathcal{C}$ is a valid cover of X.
- The expected approximation ratio is $2\ln(n)$.

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

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

probability could be further increased by repeating

Typical Approach for Designing Approximation Algorithms based on LPs

Outline

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

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

MAX-CNF Satisfiability (MAX-SAT)

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

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

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

MAX-CNF Satisfiability (MAX-SAT)

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

Why study this generalised problem?

MAX-CNF

Recall:

MAX-3-CNF Satisfiability

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

MAX-CNF Satisfiability (MAX-SAT)

- Given: CNF formula, e.g.: $(x_1 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \cdots$
- 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.

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 ℓ,

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

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 ℓ,

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

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 ℓ,

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

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 ℓ 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 ℓ,

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

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 ℓ 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,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^{m} Y_i\right] = \sum_{i=1}^{m} \mathbf{E}[Y_i] \geq \sum_{i=1}^{m} \frac{1}{2} = \frac{1}{2} \cdot m. \qquad \square$$

Approach 2: Guessing with a "Hunch" (Randomised Rounding)

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

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

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

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

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

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

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

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

C_i^+ is the index set of the unnegated 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

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

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

C_i^+ is the index set of the un-negated variables of clause i.

- In the 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^*

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Proof of Lemma (1/2):

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

$$\Rightarrow \quad \mathbf{Pr}\left[\,\text{clause } i \text{ is satisfied}\,\right] =$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

Arithmetic vs. geometric mean:

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

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

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 ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

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

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

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 ℓ,

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

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 $C_i = (x_1 \vee \cdots \vee x_\ell)$

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

Arithmetic vs. geometric mean:

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

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

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Proof of Lemma (2/2):

- So far we have shown:

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Proof of Lemma (2/2):

- So far we have shown:

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

- 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 ℓ,

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

Proof of Lemma (2/2):

- So far we have shown:

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

- 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 ℓ,

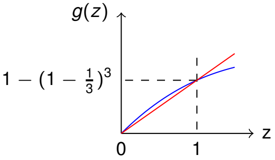
$$\mathbf{Pr}[\,\text{clause } i \text{ is satisfied}\,] \geq \left(1 - \left(1 - \frac{1}{\ell}\right)^{\ell}\right) \cdot z_i^*.$$

Proof of Lemma (2/2):

- So far we have shown:

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

- 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 ℓ,

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

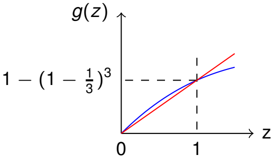
Proof of Lemma (2/2):

- So far we have shown:

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

- 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 $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^{\ell} =: \beta_\ell$.

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



Lemma

For any clause i of length ℓ,

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

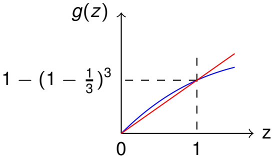
Proof of Lemma (2/2):

- So far we have shown:

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

- 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 $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^{\ell} =: \beta_\ell$.

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



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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

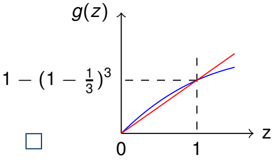
Proof of Lemma (2/2):

- So far we have shown:

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

- 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 $g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^{\ell} =: \beta_\ell$.

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



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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Proof of Theorem:

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Proof of Theorem:

- For any clause $i = 1, 2, \ldots, m$, let ℓ_i be the corresponding length.

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

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 ℓ_i be the corresponding length.
- Then the expected number of satisfied clauses is:

$$\mathbf{E}[Y] = \sum_{i=1}^{m} \mathbf{E}[Y_i] \geq$$

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Proof of Theorem:

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

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

By Lemma

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Proof of Theorem:

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

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

By Lemma

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

Analysis of Randomised Rounding

Lemma

For any clause i of length ℓ,

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

Theorem

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

Proof of Theorem:

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

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

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

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

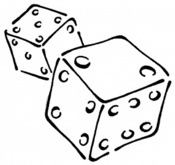
Summary

- 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

HYBRID-MAX-CNF(φ, 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



Approach 3: Hybrid Algorithm

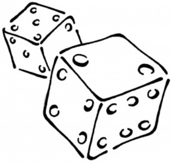
Summary

- 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(φ, 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(φ, n, m) is a randomised $4/3$-approx. algorithm.

Analysis of Hybrid Algorithm

Theorem

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

Proof:

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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(φ, 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 ℓ:

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - Algorithm 1 satisfies it with probability $1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^*$.

Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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 ℓ:
  - 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(φ, 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

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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 $\ell \in \{1, 2\}$,

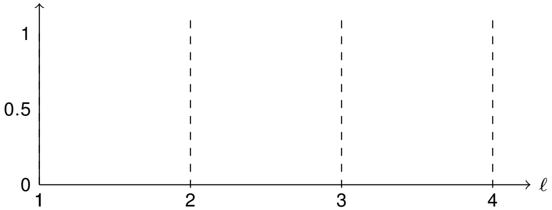
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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)



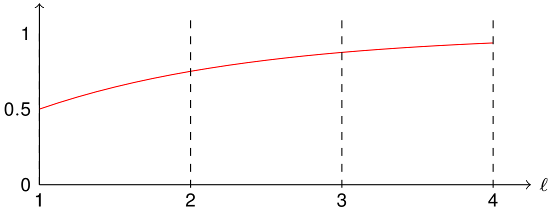
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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)



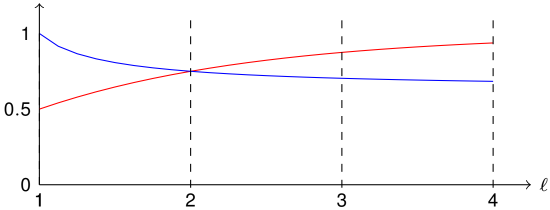
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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)



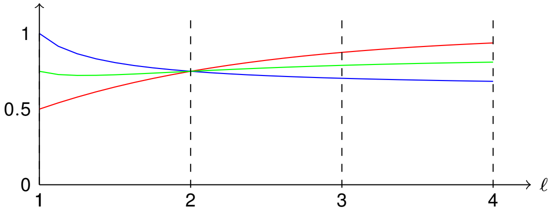
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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)



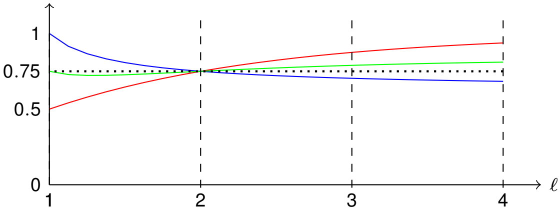
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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)



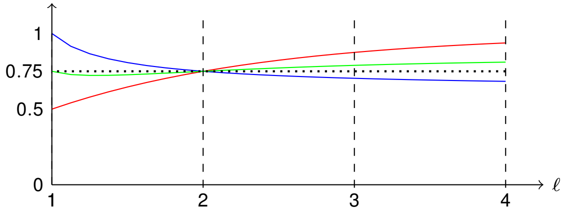
Analysis of Hybrid Algorithm

Theorem

HYBRID-MAX-CNF(φ, 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 ℓ:
  - 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(φ, 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(φ, n, m) satisfies it with prob. at least $3/4 \cdot z_i^*$ □



MAX-CNF Conclusion

Summary

- 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 minimsation 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 follwing statements are true?

1. $x \leq y \leq z$,
2. $y \leq x \leq z$,
3. $y \leq z \leq x$.

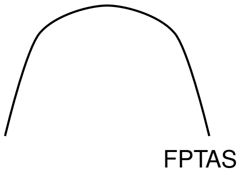
Exercise (trickier): Consider a version of the SET-COVER problem, where each element $x \in X$ has to be covered by **at least two** subsets. Design and analyse an efficient approximation algorithm.

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

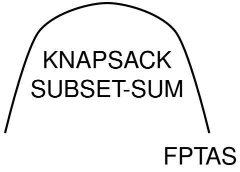
$$\mathbf{Pr}[X \geq 2] \geq 1/4 \cdot (1 - e^{-1}).$$

Outline

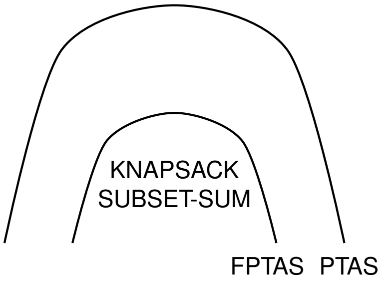
Spectrum of Approximations



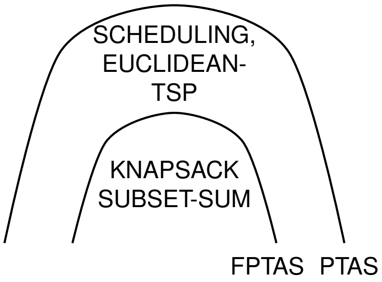
Spectrum of Approximations



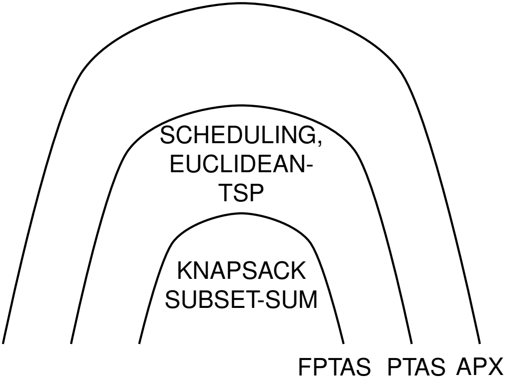
Spectrum of Approximations



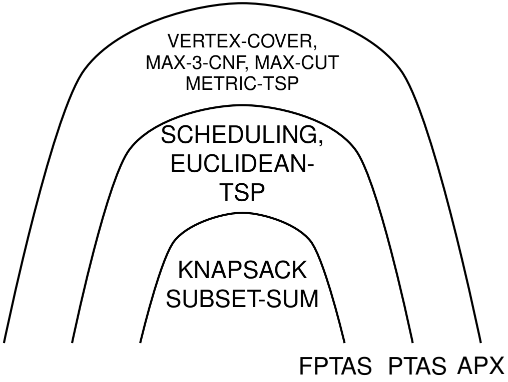
Spectrum of Approximations



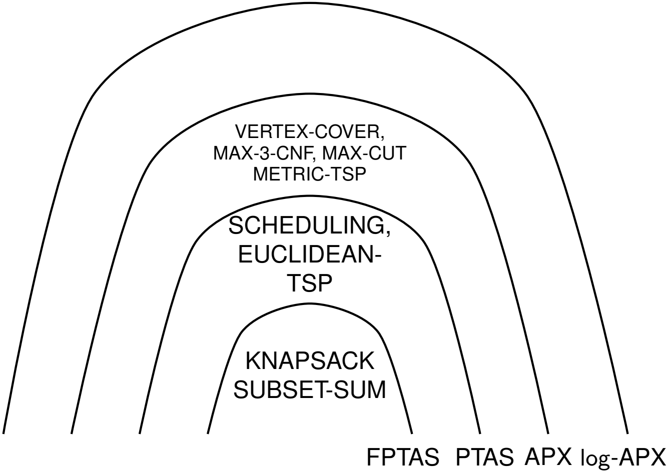
Spectrum of Approximations



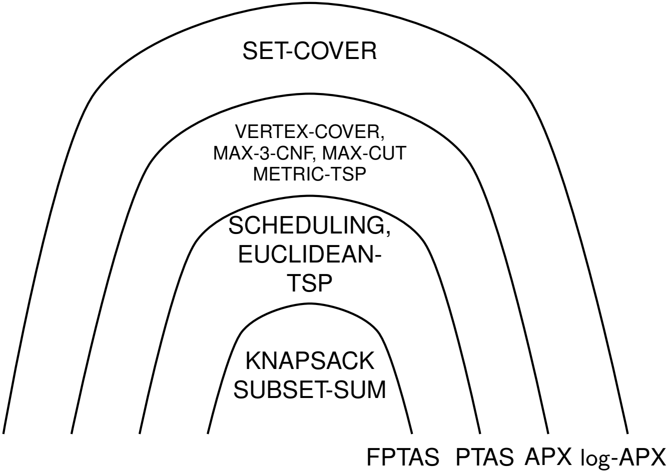
Spectrum of Approximations



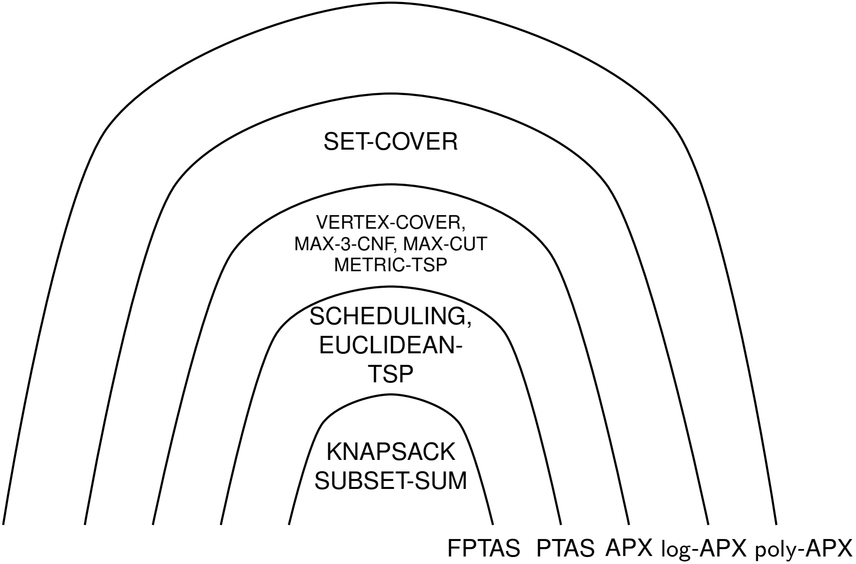
Spectrum of Approximations



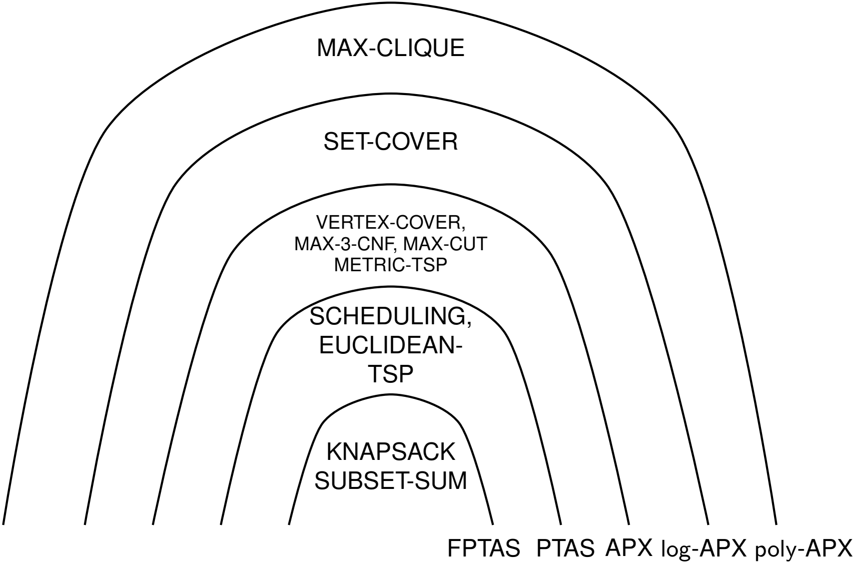
Spectrum of Approximations



Spectrum of Approximations



Spectrum of Approximations



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!