VI. Approx. Algorithms: Randomisation and Rounding

Thomas Sauerwald
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

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

extends in the natural way to randomised algorithms
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
MAX-3-CNF Satisfiability

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

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

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

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

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

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

**Theorem 35.6**

**Proof:**
- For every clause \( i = 1, 2, \ldots, m \), define a random variable:
  \[
  Y_i = \begin{cases} 
  1 & \text{clause } i \text{ is satisfied} \\
  0 & \text{otherwise}
  \end{cases}
  \]

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

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

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

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

- **Linearity of Expectations**

- **maximum number of satisfiable clauses is** \( m \)
Interesting Implications

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

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

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

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

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

Follows from the previous Corollary.
Expected Approximation Ratio

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

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

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

$Y$ is defined as in the previous proof.

One of the two conditional expectations is at least $E[Y]$!

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

Theorem

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

Proof:

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

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

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

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

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

\[ 1 \land 1 \land 1 \land (\overline{x}_3 \lor x_4) \land 1 \land (\overline{x}_2 \lor \overline{x}_3) \land (x_2 \lor x_3) \land (\overline{x}_2 \lor x_3) \land 1 \land (x_2 \lor \overline{x}_3 \lor \overline{x}_4) \]

\[
\begin{array}{c}
\text{0???} \quad 8.625 \\
\text{1???} \quad 8.875 \\
\end{array}
\]

\[
\begin{array}{c}
\text{00??} \quad x_1 = 0 \\
\text{01??} \quad x_1 = 1 \\
\text{10??} \quad x_2 = 0 \\
\text{11??} \quad x_2 = 1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{000?} \quad x_2 = 0 \\
\text{001?} \quad x_2 = 1 \\
\text{010?} \quad x_3 = 0 \\
\text{011?} \quad x_3 = 1 \\
\text{100?} \quad x_3 = 0 \\
\text{101?} \quad x_3 = 1 \\
\text{110?} \quad x_3 = 0 \\
\text{111?} \quad x_3 = 1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{0000} \quad x_4 = 0 \\
\text{0001} \quad x_4 = 1 \\
\text{0010} \quad x_4 = 0 \\
\text{0011} \quad x_4 = 1 \\
\text{0100} \quad x_4 = 0 \\
\text{0101} \quad x_4 = 1 \\
\text{0110} \quad x_4 = 0 \\
\text{0111} \quad x_4 = 1 \\
\text{1000} \quad x_4 = 0 \\
\text{1001} \quad x_4 = 1 \\
\text{1010} \quad x_4 = 0 \\
\text{1011} \quad x_4 = 1 \\
\text{1100} \quad x_4 = 0 \\
\text{1101} \quad x_4 = 1 \\
\text{1110} \quad x_4 = 0 \\
\text{1111} \quad x_4 = 1 \\
\end{array}
\]

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

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

**VI. Randomisation and Rounding**

**MAX-3-CNF**

10
Run of GREEDY-3-CNF(\(\varphi, n, m\))

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

\[ x_1 = 0 \]
\[ x_1 = 1 \]

\[ x_2 = 0 \]
\[ x_2 = 1 \]

\[ x_3 = 0 \]
\[ x_3 = 1 \]

\[ x_4 = 0 \]
\[ x_4 = 1 \]

\[ x_5 = 0 \]
\[ x_5 = 1 \]

\[ x_6 = 0 \]
\[ x_6 = 1 \]

\[ x_7 = 0 \]
\[ x_7 = 1 \]

\[ x_8 = 0 \]
\[ x_8 = 1 \]

\[ x_9 = 0 \]
\[ x_9 = 1 \]

\[ x_{10} = 0 \]
\[ x_{10} = 1 \]

\[ x_{11} = 0 \]
\[ x_{11} = 1 \]

\[ x_{12} = 0 \]
\[ x_{12} = 1 \]

\[ x_{13} = 0 \]
\[ x_{13} = 1 \]

\[ x_{14} = 0 \]
\[ x_{14} = 1 \]

\[ x_{15} = 0 \]
\[ x_{15} = 1 \]

\[ x_{16} = 0 \]
\[ x_{16} = 1 \]

\[ x_{17} = 0 \]
\[ x_{17} = 1 \]

\[ x_{18} = 0 \]
\[ x_{18} = 1 \]

\[ x_{19} = 0 \]
\[ x_{19} = 1 \]

\[ x_{20} = 0 \]
\[ x_{20} = 1 \]

\[ x_{21} = 0 \]
\[ x_{21} = 1 \]

\[ x_{22} = 0 \]
\[ x_{22} = 1 \]

\[ x_{23} = 0 \]
\[ x_{23} = 1 \]

\[ x_{24} = 0 \]
\[ x_{24} = 1 \]

\[ x_{25} = 0 \]
\[ x_{25} = 1 \]

\[ x_{26} = 0 \]
\[ x_{26} = 1 \]

\[ x_{27} = 0 \]
\[ x_{27} = 1 \]

\[ x_{28} = 0 \]
\[ x_{28} = 1 \]

\[ x_{29} = 0 \]
\[ x_{29} = 1 \]

\[ x_{30} = 0 \]
\[ x_{30} = 1 \]

\[ x_{31} = 0 \]
\[ x_{31} = 1 \]

\[ x_{32} = 0 \]
\[ x_{32} = 1 \]

\[ x_{33} = 0 \]
\[ x_{33} = 1 \]

\[ x_{34} = 0 \]
\[ x_{34} = 1 \]

\[ x_{35} = 0 \]
\[ x_{35} = 1 \]

\[ x_{36} = 0 \]
\[ x_{36} = 1 \]

\[ x_{37} = 0 \]
\[ x_{37} = 1 \]

\[ x_{38} = 0 \]
\[ x_{38} = 1 \]
Run of Greedy-3-CNF($\varphi, n, m$)

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

VI. Randomisation and Rounding

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

$$(x_1 \lor x_2 \lor x_3) \land (\overline{x}_1 \lor x_2 \lor x_4) \land (x_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor \overline{x}_4) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor x_3 \lor \overline{x}_4)$$

Return solution satisfies 9 out of 10 clauses, but the formula is satisfiable.
Given an instance of MAX-3-CNF with $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

**Theorem 35.6**

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

**Theorem (Hastad’97)**

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

Essentially there is nothing smarter than just guessing!
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
The Weighted Vertex-Cover Problem

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

This is (still) an NP-hard problem.

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

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

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

---

*Figure 35.1* illustrates how **APPROX-VERTEX-COVER** operates on an example graph. The variable \(C\) contains the vertex cover being constructed. Line 1 initializes \(C\) to the empty set. Line 2 sets \(E'\) to be a copy of the edge set \(G.E\) of the graph. The loop of lines 3–6 repeatedly picks an edge \((u, v)\) from \(E'\), adds it to \(C\), and removes from \(E'\) every edge incident on either \(u\) or \(v\). Line 7 returns \(C\).

---

*Computed solution has weight 101*

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

\textsc{Approx-Vertex-Cover}(G)

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

**Computed solution has weight 101**

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

Idea: Round the solution of an associated linear program.

0-1 Integer Program

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

Linear Program

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

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

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

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

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

**Theorem 35.7**

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

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

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

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

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):

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

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

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

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

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
The **Weighted Set-Covering Problem**

**Set Cover Problem**

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

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

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

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

0-1 Integer Program

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

Linear Program

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

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

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

Cost equals 8.5
Randomised Rounding

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

More precisely, if \( y \) denotes the optimal solution of the LP, then we compute an integral solution \( \bar{y} \) by:

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

for all \( S \in F \).

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

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

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

Lemma

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

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

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

Proof:

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

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

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

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

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

Lemma

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

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

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

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

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

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

Clearly runs in polynomial-time!
Analysis of Weighted Set Cover-LP

Theorem

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

Proof:

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

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

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

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

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

Typical Approach for Designing Approximation Algorithms based on LPs
Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
Recall:

### MAX-3-CNF Satisfiability

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

### MAX-CNFSatisfiability (MAX-SAT)

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

**Why study this generalised problem?**

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

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

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

Analysis

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

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

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

Proof:

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

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

The same as randomised rounding!

0-1 Integer Program

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

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

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

- 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 \( \ell \),

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

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 \( C_i = (x_1 \lor \cdots \lor x_\ell) \)

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

Arithmetic vs. geometric mean:

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

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

\[
= 1 - \left( 1 - \frac{\sum_{j=1}^{\ell} y_j^*}{\ell} \right)^{\ell} \geq 1 - \left( 1 - \frac{z_i^*}{\ell} \right)^{\ell}.
\]
For any clause $i$ of length $\ell$, 

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

Proof of Lemma (2/2):

- So far we have shown:

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

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

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

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

**Lemma**

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

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

**Theorem**

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

**Proof of Theorem:**

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

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

  - By Lemma
  - Since $(1 - 1/x)^x \leq 1/e$
  - LP solution at least as good as optimum
Approach 3: Hybrid Algorithm

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

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**

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

**Proof:**

- It suffices to prove that clause \( i \) is satisfied with probability at least \( 3/4 \cdot z_i^* \).
- For any clause \( i \) of length \( \ell \):
  - Algorithm 1 satisfies it with probability \( 1 - 2^{-\ell} = \alpha_\ell \geq \alpha_\ell \cdot z_i^* \).
  - Algorithm 2 satisfies it with probability \( \beta_\ell \cdot z_i^* \).
  - HYBRID-MAX-CNF\( (\varphi, n, m) \) satisfies it with probability \( \frac{1}{2} \cdot \alpha_\ell \cdot z_i^* + \frac{1}{2} \cdot \beta_\ell \cdot z_i^* \).
- Note \( \frac{\alpha_\ell + \beta_\ell}{2} = 3/4 \) for \( \ell \in \{1, 2\} \), and for \( \ell \geq 3 \), \( \frac{\alpha_\ell + \beta_\ell}{2} \geq 3/4 \) (see figure)
- \( \Rightarrow \) HYBRID-MAX-CNF\( (\varphi, n, m) \) satisfies it with prob. at least \( 3/4 \cdot z_i^* \) \( \square \)
Since $\alpha_2 = \beta_2 = 3/4$, we cannot achieve a better approximation ratio than $4/3$ by combining Algorithm 1 & 2 in a different way.

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

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

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

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

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

VI. Randomisation and Rounding MAX-CNF
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 $X := \sum_{i=1}^{n} X_i$, $\mathbb{E}[X] \geq 2$, then 

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

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Weighted Set Cover

MAX-CNF

Conclusion
Spectrum of Approximations

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

FPTAS, PTAS, APX, $\log$-APX, poly-APX
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!