Randomised Algorithms
Lecture 10: Approximation Algorithms: Set-Cover and MAX-CNF

Thomas Sauerwald (tms41@cam.ac.uk)
Weighted Set Cover

MAX-CNF
The **Weighted Set-Cover Problem**

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

**Set Cover Problem**

- Sum over the costs of all sets in $C$

**Remarks**:
- generalisation of the weighted Vertex-Cover problem
- models resource allocation problems

### Example

- $\mathcal{F} = \{ S_1, S_2, S_3, S_4, S_5, S_6 \}$
- Costs: $c(S_1) = 2$, $c(S_2) = 3$, $c(S_3) = 3$, $c(S_4) = 5$, $c(S_5) = 1$, $c(S_6) = 2$
- Solution: $C = \{ S_1, S_2, S_3, S_5 \}$
- Cost: $2 + 3 + 3 + 1 = 9$
Setting up an Integer Program

**Question:** 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

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

Linear Program

\[
\begin{align*}
\text{minimize} \quad & \sum_{S \in \mathcal{F}} c(S)y(S) \\
\text{subject to} \quad & \sum_{S \in \mathcal{F} : x \in S} y(S) \geq 1 \quad \text{for each } x \in X \\
& y(S) \in [0, 1] \quad \text{for each } S \in \mathcal{F}
\end{align*}
\]
The strategy employed for Vertex-Cover would take all 6 sets!

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

Cost equals 8.5
Randomised Rounding

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

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

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

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

for all \( S \in F \).

Therefore, \( \mathbb{E} [ y(S) ] = \bar{y}(S) \).
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>$\overline{y}(\cdot)$:</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
</tbody>
</table>

Idea: Interpret the $\overline{y}$-values as probabilities for picking the respective set.

Lemma

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

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

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

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

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

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

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

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

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

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

  $\overline{y}$ solves the LP!
The Final Step

**Lemma**

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

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

**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**

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

clearly runs in polynomial-time!
Theorem

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

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

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

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

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

Hence with probability at least $1 - \frac{1}{n} - \frac{1}{2} > \frac{1}{3}$, solution is valid and 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

[Exercise Question (9/10).10] gives a different perspective on the amplification procedure through non-linear randomised rounding.
Outline

Weighted Set Cover

MAX-CNF
Recall:

MAX-3-CNF Satisfiability

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

MAX-CNF Satisfiability (MAX-SAT)

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

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 $\ell$,

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

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

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

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

\[\square\]
Approach 2: Guessing with a “Hunch” (Randomised Rounding)

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

The same as randomised rounding!

0-1 Integer Program

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

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

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

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

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

Lemma

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

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

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

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

Arithmetic vs. geometric mean:

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

\[
\geq 1 - \left( \frac{\sum_{j=1}^{\ell} \left( 1 - y_j \right)}{\ell} \right)^\ell = 1 - \left( 1 - \frac{\sum_{j=1}^{\ell} y_j}{\ell} \right)^\ell \geq 1 - \left( 1 - \frac{\bar{z}_i}{\ell} \right)^\ell.
\]
Analysis of Randomised Rounding

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

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

Proof of Lemma (2/2):

- So far we have shown:
  $$\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left( 1 - \frac{\bar{Z}_i}{\ell} \right)^\ell$$

- 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 $g(1) = 1 - \left( 1 - \frac{1}{\ell} \right)^\ell =: \beta_\ell$. Then:
  $$\Rightarrow g(z) \geq \beta_\ell \cdot z \quad \text{for any } z \in [0, 1]$$

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

Lemma

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

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

Theorem

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

Proof of Theorem:

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

\[
\mathbb{E}[Y] = \sum_{i=1}^{m} \mathbb{E}[Y_i] \geq \sum_{i=1}^{m} \left( 1 - \left( 1 - \frac{1}{\ell_i} \right)^{\ell_i} \right) \cdot \overline{z}_i \geq \sum_{i=1}^{m} \left( 1 - \frac{1}{e} \right) \cdot \overline{z}_i \geq \left( 1 - \frac{1}{e} \right) \cdot \text{OPT}
\]

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

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

**Theorem**

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

**Proof:**

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

\[\Rightarrow\] HYBRID-MAX-CNF(\(\varphi, n, m\)) satisfies it with prob. at least \(3/4 \cdot \overline{z}_i\). \(\square\)
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!