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

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2024

The **Weighted Set-Cover 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 \( C \subseteq \mathcal{F} \)

Sum over the costs of all sets in \( C \)

\[ \sum_{S \in C} c(S) \]

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

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

### 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

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

Linear Program

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

Randomised Rounding

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

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

Randomised Rounding

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

\[
y(S) = \begin{cases} 
1 & \text{with probability } \overline{y}(S) \\
0 & \text{otherwise.}
\end{cases} \quad \text{for all } S \in \mathcal{F}.
\]

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

Back to the Example

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

Cost equals 8.5

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

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

Randomised Rounding

- The expected cost satisfies

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

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

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

Proof:

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

- **Step 2:** The probability for an element to be (not) covered
  \[
  \mathbb{P}[x \not\in \cup_{S \in C} S] = \prod_{S \in F : x \in S} \mathbb{P}[x \not\in S] = \prod_{S \in F : x \in S} (1 - \gamma(S)) \leq \prod_{S \in F : x \in S} e^{-\gamma(S)} = e^{-\sum_{S \in F : x \in S} \gamma(S)} \leq e^{-1} \]

**The Final Step**

**Lemma**

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

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

**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 \( \gamma \), 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 \( \gamma(S) \)
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
    \[
    \mathbb{P}[x \not\in \cup_{S \in C} S] \leq \left(1 - \frac{1}{e}\right)^{2\ln n} = \frac{1}{n^2}.
    \]
  - This implies for the event that all elements are covered:
    \[
    \mathbb{P}[X = \cup_{S \in C} S] = 1 - \mathbb{P}\left[ \bigcup_{x \in X} \{x \not\in \cup_{S \in C} S\} \right] \geq 1 - \sum_{x \in X} \mathbb{P}[x \not\in \cup_{S \in C} S] \geq 1 - n \cdot \frac{1}{n^2} = 1 - \frac{1}{n}.
    \]

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

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

Weighted Set Cover

**MAX-CNF**

Recall: Given: 3-CNF formula, e.g.: $(x_1 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_5} \lor \overline{x_6}) \land \cdots$

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

**MAX-3-CNF Satisfiability**

Given: CNF formula, e.g.: $(x_1 \lor x_4)$ $\land (x_2 \lor x_3 \lor x_4 \lor x_5)$ $\land \cdots$

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

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

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,

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

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

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

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

Proof of Lemma (1/2):
- Assume w.l.o.g. all literals in clause \( i \) appear non-negated (otherwise replace every occurrence of \( x_j \) by \( \overline{x_j} \) in the whole formula)
- Further, by relabelling assume \( C_i = (x_1 \lor \cdots \lor x_k) \)
\[
\Rightarrow \Pr[\text{clause } i \text{ is satisfied}] = 1 - \prod_{j=1}^{\ell} \Pr[y_j \text{ is false }] = 1 - \prod_{j=1}^{\ell} (1 - \overline{y}_j)
\]

Arithmetic vs. geometric mean:
\[
\frac{a_1 + \cdots + a_k}{k} \geq \sqrt[k]{a_1 \cdots a_k}
\]
\[
1 - \left(\frac{\sum_{j=1}^{\ell} (1 - \overline{y}_j)}{\ell}\right)^\ell \geq 1 - \left(1 - \frac{z_i}{\ell}\right)^\ell.
\]

Analysis of Randomised Rounding

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

Proof of Lemma (2/2):
- So far we have shown:
\[
\Pr[\text{clause } i \text{ is satisfied}] \geq 1 - \left(1 - \frac{z_i}{\ell}\right)^\ell
\]
- For any \( \ell \geq 1 \), define \( g(z) := 1 - \left(1 - \frac{z}{\ell}\right)^\ell \). This is a concave function
  with \( g(0) = 0 \) and \( g(1) = 1 - \left(1 - \frac{1}{\ell}\right)^\ell =: \beta_{\ell} \).
\[
\Rightarrow g(z) \geq \beta_{\ell} \cdot z \quad \text{for any } z \in [0, 1] \quad 1 - (1 - \frac{1}{\ell})^\ell
\]
- Therefore, \( \Pr[\text{clause } i \text{ is satisfied}] \geq \beta_{\ell} \cdot z_i. \)

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

Algorithm sets each variable \( x \) to TRUE with prob. \( \frac{1}{2} + \frac{1}{2} \cdot \frac{z_i}{\ell} \).
Note, however, that variables are not independently assigned!

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:
\[
E[Y] = \sum_{i=1}^{m} \sum_{j=1}^{m} \left(1 - \left(1 - \frac{1}{\ell_i}\right)^{\ell_i}\right) \cdot z_i \geq \sum_{i=1}^{m} \left(1 - \frac{1}{e}\right) \cdot z_i \geq \left(1 - \frac{1}{e}\right) \cdot \text{OPT}
\]
By Lemma
Since \((1 - 1/x)^x \leq 1/e\)
LP solution at least as good as optimum
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 \( \frac{3}{4} \cdot Z_i \).
- 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(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 \( \frac{\alpha_\ell + \beta_\ell}{2} = \frac{3}{4} \) for \( \ell \in \{1, 2\} \), and for \( \ell \geq 3 \), \( \frac{\alpha_\ell + \beta_\ell}{2} \geq \frac{3}{4} \) (see figure).

\[ \implies \text{HYBRID-MAX-CNF(ϕ, n, m)} \text{ satisfies it with prob. at least } \frac{3}{4} \cdot Z_i \].

**MAX-CNF Conclusion**

**Summary**

- Since \( \alpha_2 = \beta_2 = \frac{3}{4} \), 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!