Approximation Algorithms 1: Vertex Cover and MAX-3SAT

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
In computer science, there is a set of **NP-hard** problems such that

- nobody has found a polynomial-time algorithm for **any** of those problems;

- no polynomial-time algorithms can exist for **any** of those problems unless **P = NP**.

**P** = the set of problems that can be solved in polynomial time on a **deterministic** Turing machine

**NP** = the set of problems that can be solved in polynomial time on a **non-deterministic** Turing machine

Turing machines are formalized in CSCI3130 (Formal Languages and Automata Theory), and so is the notion of NP-hard.

Whether **P = NP** is still unsolved to this day.
What can we do if a problem is NP-hard?

The rest of the course will focus on a principled approach for tackling NP-hard problems: approximation.

In many problems, even though an optimal solution may be expensive to find, we can find near-optimal solutions efficiently.

Next, we will see two examples: vertex cover and MAX-3SAT.
The Vertex Cover Problem
$G = (V, E)$ is a simple undirected graph. A subset $S \subseteq V$ is a **vertex cover** of $G$ if every edge $\{u, v\} \in E$ is incident to at least one vertex in $S$.

**The V.C. Problem:** Find a vertex cover of the smallest size.

**Example:**

![Graph diagram]

An optimal solution is $\{a, f, c, e\}$.
The vertex cover problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in $|V|$.
- Such algorithms cannot exist if $P \neq NP$. 
Approximation Algorithms

\( \mathcal{A} \) = an algorithm that, given any legal input \( G = (V, E) \), returns a vertex cover of \( G \).

\( OPT_G \) = the smallest size of all the vertex covers of \( G \).

\( \mathcal{A} \) is a \( \rho \)-approximate algorithm for the vertex cover problem if, for any legal input \( G = (V, E) \), \( \mathcal{A} \) can return a vertex cover with size at most \( \rho \cdot OPT_G \).

The value \( \rho \) is the approximation ratio. We say that \( \mathcal{A} \) achieves an approximation ratio of \( \rho \).
Consider the following algorithm.

**Input:** \( G = (V, E) \)

\( S = \emptyset \)

**while** \( E \) is not empty **do**

- pick an arbitrary edge \( \{u, v\} \) in \( E \)
- add \( u, v \) to \( S \)
- remove from \( E \) all the edges of \( u \) and all the edges of \( v \)

**return** \( S \)

It is easy to show:

- \( S \) is a vertex cover of \( G \);

- The algorithm runs in time polynomial to \( |V| \) and \( |E| \).

We will prove later that the algorithm is 2-approximate.
Example:

Suppose we start by picking edge \{b, c\}. Then, \(S = \{b, c\}\) and \(E = \{\{a, e\}, \{a, d\}, \{d, e\}, \{d, f\}\}\).

Any edge in \(E\) can then be chosen. Suppose we pick \{a, e\}. Then, \(S = \{a, b, c, e\}\) and \(E = \{\{d, f\}\}\).

Finally, pick \{d, f\}. \(S = \{a, b, c, d, e, f\}\) and \(E = \emptyset\).
**Theorem 1**: The algorithm returns a set of at most $2 \cdot \text{OPT}_G$ vertices.

Let $M$ be the set of edges picked.

**Example**: In the previous example, $M = \{\{b, c\}, \{a, e\}, \{d, f\}\}$. 
Lemma 1: The edges in $M$ do not share any vertices.

Proof: Suppose that $M$ has edges $e_1$ and $e_2$ both incident to a vertex $v$. W.l.o.g., assume that $e_1$ was picked before $e_2$. After picking $e_1$, the algorithm deleted all the edges of $v$, because of which $e_2$ could not have been picked, giving a contradiction.

Lemma 2: $|M| \leq \text{OPT}_G$.

Proof: Any vertex cover must include at least one vertex of each edge in $M$. $|M| \leq \text{OPT}$ follows from Lemma 1.

Theorem 1 holds because the algorithm returns exactly $2|M|$ vertices.
The MAX-3SAT Problem
A **variable**: a boolean unknown $x$ whose value is 0 or 1.
A **literal**: a variable $x$ or its negation $\overline{x}$.
A **clause**: the OR of 3 literals with different variables.

$S = \text{a set of clauses}$
$\mathcal{X} = \text{the set of variables appearing in at least one clause of } S$
A **truth assignment** of $S$: a function from $\mathcal{X}$ to $\{0, 1\}$.

A truth assignment $f$ **satisfies** a clause in $S$ if the clause evaluates to 1 under $f$.

**The MAX-3SAT Problem:** Let $S$ be a set of $n$ clauses. Find a truth assignment of $S$ to maximize the number of clauses satisfied.
Example:

\[ S = \{ x_1 \lor x_2 \lor x_3, \]
\[ x_1 \lor x_2 \lor \bar{x}_3, \]
\[ x_1 \lor \bar{x}_2 \lor x_3, \]
\[ x_1 \lor \bar{x}_2 \lor \bar{x}_3, \]
\[ \bar{x}_1 \lor x_3 \lor x_4, \]
\[ \bar{x}_1 \lor x_3 \lor \bar{x}_4, \]
\[ \bar{x}_1 \lor \bar{x}_3 \lor x_4, \]
\[ \bar{x}_1 \lor \bar{x}_3 \lor \bar{x}_4 \}. \]

\[ n = 8 \text{ and } \mathcal{X} = \{ x_1, x_2, x_3, x_4 \}. \]

The truth assignment \( x_1 = x_2 = x_3 = x_4 = 1 \) satisfies 7 clauses. It is impossible to satisfy 8.
The MAX-3SAT problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in $n$.
- Such algorithms cannot exist if $P \neq NP$. 
Approximation Algorithms

$A$ = an algorithm that, given any legal input $S$, returns a truth assignment of $S$.

$OPT_S$ = the largest number of clauses that a truth assignment of $S$ can satisfy.

$Z_S$ = the number of clauses satisfied by the truth assignment $A$ returns.

- $Z_S$ is a random variable if $A$ is randomized.

$A$ is a randomized $\rho$-approximate algorithm for MAX-3SAT if $E[Z_S] \geq \rho \cdot OPT_S$ holds for any legal input $S$.

The value $\rho$ is the approximation ratio. We also say that $A$ achieves an approximation ratio of $\rho$ in expectation.
Consider the following algorithm.

**Input:** a set \( S \) of clauses with variable set \( \mathcal{X} \)

for each variable \( x \in \mathcal{X} \) do
    toss a fair coin
    if the coin comes up heads then \( x \leftarrow 1 \)
    else \( x \leftarrow 0 \)

It is clear that the algorithm runs in \( O(n) \) time.
Next, we show that the algorithm achieves an approximation ratio \( 7/8 \) in expectation.
**Theorem 2:** The algorithm produces a truth assignment that satisfies $\frac{7}{8} n$ clauses in expectation.

**Proof:** It suffices to show that each clause is satisfied with probability $7/8$. W.l.o.g., suppose that the clause is $x_1 \lor x_2 \lor x_3$. The clause is 0 if and only if $x_1$, $x_2$, and $x_3$ are all 0. The probability for $x_1 = x_2 = x_3 = 0$ is $1/8$.

**Think:** What about a clause like $x_1 \lor x_2 \lor \overline{x}_3$?