CSE515 Advanced Algorithms
Lecture 21
Randomized Approximation Algorithm for MAX 3-SAT

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

May 11, 2021

Introduction

- Assignment 3 scores are posted.
- Solutions are also posted
- Assignment 4 will be posted by Thursday.
- This lecture is an introduction to randomized algorithms through a simple example.
- Reference: Section 13.4 of *Algorithm Design* by Kleinberg and Tardos.

Problem Statement

- We are given n *boolean* variables $x_1, \ldots, x_n$.
- So the value of any x_i is either 0 (false) or 1 (true).
- The *negation* of x_i is $\bar{x_i} = 1 - x_i$.
- A *clause* is a disjunction of terms in $x_1, \ldots, x_n, \bar{x_1}, \ldots, \bar{x_n}$.
  - Example: $x_1 \vee \bar{x_3} \vee \bar{x_4}$
    - Means: x_1 or not x_3 or not x_4.
- In this lecture, we will only consider clauses involving exactly *three different* variables.
  - Example: $x_2 \vee \bar{x_3} \vee x_4$
  - But not $x_2 \vee \bar{x_2} \vee x_4$, and not $x_2 \vee x_4$
- A *truth assignment* is an assignment of value 0 or 1 to each x_i.

Problem Statement

Problem (MAX 3-SAT)

Given a collection $C_1, \dots, C_k$ of clauses with (exactly) 3 variables each, find a truth assignment that satisfies the largest number of clauses.

Example:

- $C_1 = x_1 \vee x_2 \vee x_3$
- $C_2 = x_1 \vee \bar{x}_3 \vee x_4$
- $C_3 = x_1 \vee \bar{x}_2 \vee \bar{x}_3$
- $C_4 = \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$
- $C_5 = x_2 \vee \bar{x}_3 \vee \bar{x}_4$

- With the truth assignment $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, clauses C_1, C_2, and C_5 are satisfied.
- An optimal truth assignment: $x_1 = x_2 = 1$, $x_3 = x_4 = 0$. All clauses are satisfied.

MAX 3-SAT is **NP**-hard. Why?

A First Algorithm

Algorithm 1

Assign 0 or 1 to each variable $x_1, \ldots, x_n$, with probability $\frac{1}{2}$ each, independently.

- In each clause, each term is satisfied with probability $\frac{1}{2}$.
- So each clause is satisfied with probability $1 - (\frac{1}{2})^3 = \frac{7}{8}$.
- Let Z_i be the random variable such that
  - $Z_i = 0$ if clause C_i is not satisfied.
  - $Z_i = 1$ if clause C_i is satisfied.
- We say that Z_i is the *indicator random variable* associated with the event: "C_i is satisfied."
- Then the expected value of Z_i is

$$E[Z_i] = 0 \times \frac{1}{8} + 1 \times \frac{7}{8} = \frac{7}{8}$$

A First Algorithm

- The number of satisfied clauses is the random variable

$$Z = Z_1 + \cdots + Z_k.$$

- By *linearity of expectation*

$$E[Z] = E[Z_1] + \cdots + E[Z_k] = \frac{7}{8}k.$$

Theorem

The expected number of clauses satisfied by Algorithm 1 is $\frac{7}{8}k$. In particular, as there are k clauses, it is within a factor $\frac{7}{8}$ from optimal.

The Probabilistic Method

- We have just proved that the expected number of clauses satisfied by a random truth assignment is $\frac{7}{8}k$.
- So there should be at least one assignment that satisfies $\geqslant \frac{7}{8}k$ clauses:

Theorem

For any instance of 3-SAT with k clauses, there is a truth assignment that satisfies at least $\frac{7}{8}k$ clauses.

Corollary

Any instance of 3-SAT with less than 8 clauses is satisfiable.

Application to the example above

- As $k = 5$, there is a truth assignment satisfying all 5 clauses.

The Probabilistic Method

We just used the *probabilistic method*:

- Although the proof is based on probabilities, the conclusion is *certain*. We know for sure that there is one truth assignment satisfying $\geqslant \frac{7}{8}k$ clauses.
- Underlying idea: if an object belongs to a certain class with nonzero probability, then there should be (at least) one object in this class.
- This is an important idea in combinatorics.

Second Algorithm

- Algorithm 1 gives, in $O(k)$ time, a solution that is expected to be good.
- But if we are unlucky, it may return a bad solution.
- We can fix it as follows:

Algorithm 2

Repeat Algorithm 1 until it gives a solution that satisfies at least $\frac{7}{8}k$ clauses.

- This algorithm always returns a $\frac{7}{8}$-approximation.
- But its running time is random.
- In the following, we will show that its expected running time is $O(k^2)$.

Second Algorithm

- Our analysis is based on the following rule, which is often useful for analyzing randomized algorithms:

Theorem (Waiting-time bound)

If we repeatedly perform independent trials of an experiment, each of which succeeds with probability $p > 0$, then the expected number of trials we need to perform until the first success is $\frac{1}{p}$.

Proof:

- Let X denote the number of trials until the first success.
- For any $j > 0$, $\Pr[X = j] = (1-p)^{j-1}p$.
- So

$$E[X] = \sum_{j=1}^{\infty} j \cdot \Pr[X = j] = \sum_{j=1}^{\infty} j(1-p)^{j-1}p = p\sum_{j=1}^{\infty} j(1-p)^{j-1}.$$

Second Algorithm

- To complete the proof of the waiting-time bound, we need to prove that
$$\sum_{j=1}^{\infty} j(1-p)^{j-1} = \frac{1}{p^2}.$$

- It is true because for any $x \in (0,1)$:
$$\sum_{j=1}^{\infty} jx^{j-1} = \left(\sum_{j=0}^{\infty} x^j\right)' = \left(\frac{1}{1-x}\right)' = \frac{1}{(1-x)^2}.$$

Second Algorithm

- In order to analyze Algorithm 2, we need to bound the probability p that we succeed at each step.
- In other words, we want to bound the probability p that Algorithm 1 returns a solution that satisfies $\geqslant \frac{7}{8}k$ clauses.
- We find $p \geqslant \frac{1}{8k}$. (Proof done in class. See Kleinberg & Tardos.)
- So by the waiting-time bound, the expected number of times we run Algorithm 1 is at most $8k$.
- As Algorithm 1 runs in $O(k)$ time, we conclude that:

Theorem

Algorithm 2 is a randomized $\frac{7}{8}$-approximation agorithm with expected running time $O(k^2)$.