Introduction to
Approximation Algorithms

Mong-Jen Kao (高孟駿)

Friday 14:20 – 17:20

Outline

- Approximation Algorithm
  - What is it and Why?
- Our First Example
  - The Max-3SAT Problem
  - Supplementary Material for Max-3SAT
- General Goal of this Course

Approximation Algorithm –

What is it and Why?

The Big Theme

- Most important combinatorial optimization problems are known to be ***NP-hard*****.**
  - That says, it is ***unlikely*** that we can *compute* their *optimal solutions* ***efficiently in polynomial-time*****,**

    unless P = NP, which is conjectured & widely believed to be untrue.
  - In fact, only very few practical optimization problems can be solved efficiently in polynomial time.

What can we do ?

- To cope with this unsatisfying fact,
  when *quitting the hard problems is unfortunately not an option…*

1. Derive more clever algorithms, and
   live with the **super-polynomial running time,**

2. Identify special parameters known to be small (in practice) and
   derive **fixed-parameter tractable (FPT) algorithms**.

We will cover this topic in "Topics in Intractable Problems and Complexity" next semester (*hopefully*).

What can we do ?

- To cope with this unsatisfying fact,
  when *quitting the hard problems is unfortunately not an option…*

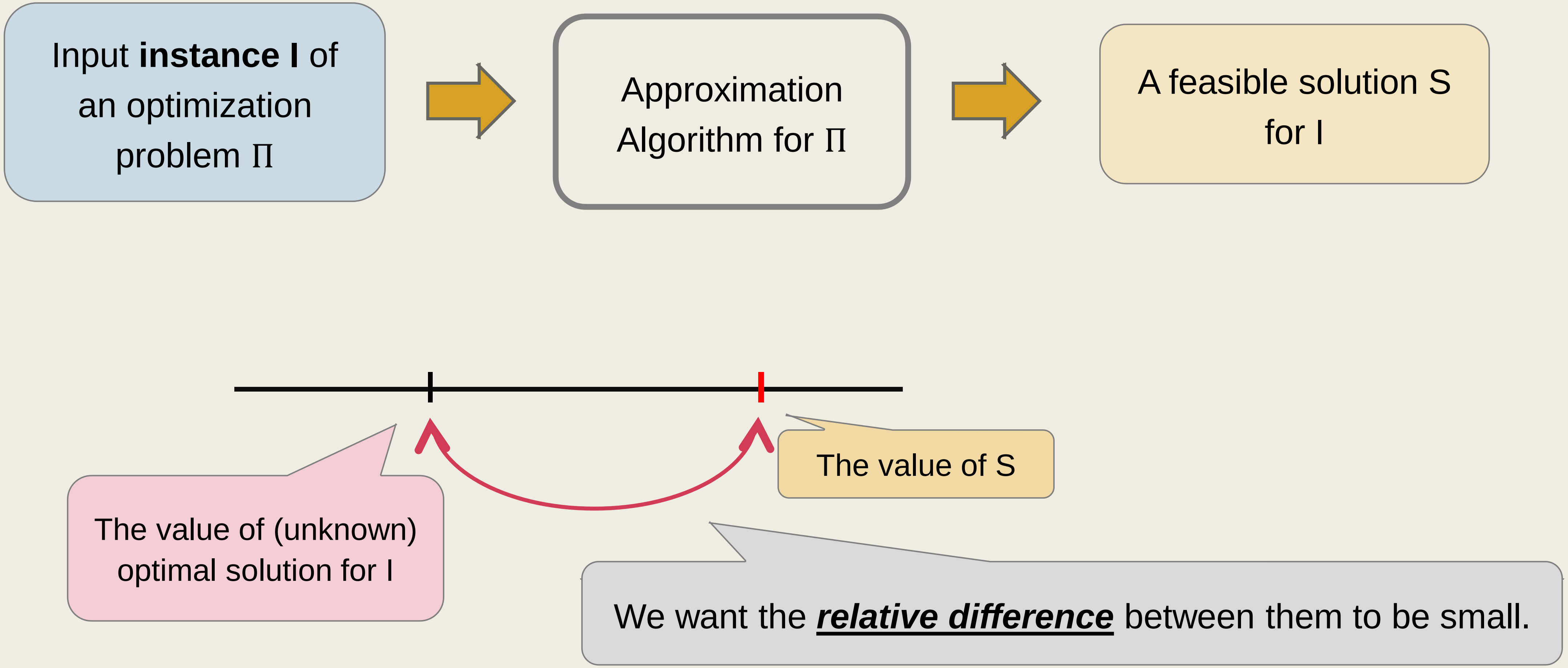
3. Derive ***efficient (polynomial-time) algorithms*** that computes ***near-optimal solutions***,

   i.e., the *approximation algorithms*.

   Natural questions to raise:

   - How do we *measure the quality of the solution computed*?
   - What is *the best guarantee* we can make?

- To compute a <u>**near-optimal solution** *efficiently*</u>…



First Example –

The Max-3SAT Problem

The Max-3SAT Problem

- Given a Boolean formula in <u>*3-CNF*</u> (conjunctive normal form), what is the ***maximum number of clauses*** that can be ***satisfied*** simultaneously?

$$\Phi = (x_2 \lor \overline{x_3} \lor \overline{x_4}) \land (x_2 \lor x_3 \lor \overline{x_4}) \land (\overline{x_1} \lor x_2 \lor x_4) \land \ \ldots$$

Clause C_1

C_2

C_3

$x_i \in \{0, 1\}$

false / true

The Max-3SAT Problem

- For example,

$$C_1 = (x_2 \vee \overline{x_3} \vee \overline{x_4})$$

$$C_2 = (x_2 \vee x_3 \vee \overline{x_4})$$

$$C_3 = (\overline{x_1} \vee x_2 \vee x_4)$$

$$C_4 = (\overline{x_1} \vee \overline{x_2} \vee x_3)$$

  - Setting $(x_1, x_2, x_3, x_4) = (0,0,1,1)$ satisfies C_2, C_3, C_4.

The Max-3SAT Problem

- Input:
  - Boolean variables $x_1, x_2, \ldots, x_n$.
  - Boolean formula $\Phi = \{C_1, C_2, \ldots, C_m\}$, where

$$C_i = \{y_{i,1}, y_{i,2}, y_{i,3}\}, \qquad y_{i,j} \in \{x_{\sigma_i(j)}, \bar{x}_{\sigma_i(j)}\}.$$

- Goal:
  - Compute a truth assignment of $x_1, x_2, \ldots, x_n$ that satisfies the maximum umber of clauses in Φ.

Status of MAX-3SAT

- Unsurprisingly, the MAX-3SAT problem is NP-hard to solve.

- In fact, Max-3SAT is the *optimization version* of the 3-SAT problem, a classic NP-hard *decision* problem.

  – In 3-SAT, we ask “***Is Φ satisfiable?***“

    That is,

    “Is there a truth assignment that satisfies all the clauses in Φ? ”

Decision Problem
(Yes / No)

Status of MAX-3SAT

- The Max-3SAT is the *optimization version* of the 3-SAT problem.
  - In Max-3SAT, the philosophy is

    " *Provided that Φ is not satisfiable*,

    what is the maximum number of clauses we can satisfy? "

- Since *Max-3SAT can be used to answer 3-SAT*,

  it must be NP-hard to solve as well.

We say that, 3-SAT $\propto$ (is reducible to) Max-3SAT.

Some Remarks

- The following conjecture was made in 1999:

Exponential Time Hypothesis (ETH). [Impagliazzo, Paturi, 1999].

The 3-SAT problem cannot be solved in subexponential time in the worst case.

  - This is a stronger statement than $\mathrm{P} \neq \mathrm{NP}$.
  - This hypothesis is unproven but widely believed to be true.

A Simple Approximation Algorithm

for Max-3SAT

Flip a fair coin and let it decide!

A randomized algorithm

- Let $I = \left(\{x_i\}_{1 \le i \le n},\ \ \Phi = \{C_j\}_{1 \le j \le m} \right)$ be an instance of Max 3-SAT.

- Consider the following algorithm:

1. For each $1 \le i \le n$,

   set x_i to be true with probability $\frac{1}{2}$.

2. Output $(x_1, x_2, \ldots, x_n)$.

How well does this algorithm perform?

The Analysis

This algorithm can be derandomized to run *deterministically*. We will see this later.

- Consider the following algorithm:

1. For each $1 \leq i \leq n$, set x_i to be true with probability $\frac{1}{2}$.
2. Output $(x_1, x_2, \ldots, x_n)$.

- Let $X_j = \begin{cases} 1, & \text{if clause } C_j \text{ is satisfied,} \\ 0, & \text{otherwise.} \end{cases}$

- Then, $\Pr[X_j = 1] = 1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$ and $E[\sum X_j] = \frac{7}{8}m$.

The Analysis

- Let OPT_I be the optimal value of the instance I.
- Then,

$$E\left[\sum X_j\right] = \frac{7}{8}m \geq \frac{7}{8}OPT_I\,.$$

Since $OPT_I \leq m$

- The simple algorithm always guarantees an assignment that performs at least $7/8$ fraction of what an optimal solution does.
- It is called a ***$7/8$-approximation algorithm*** for MAX-3SAT.

Notes

Larger α means better approximation guarantee.

- An α-approximation algorithm $\mathcal{A}$ for Max-3SAT guarantees that

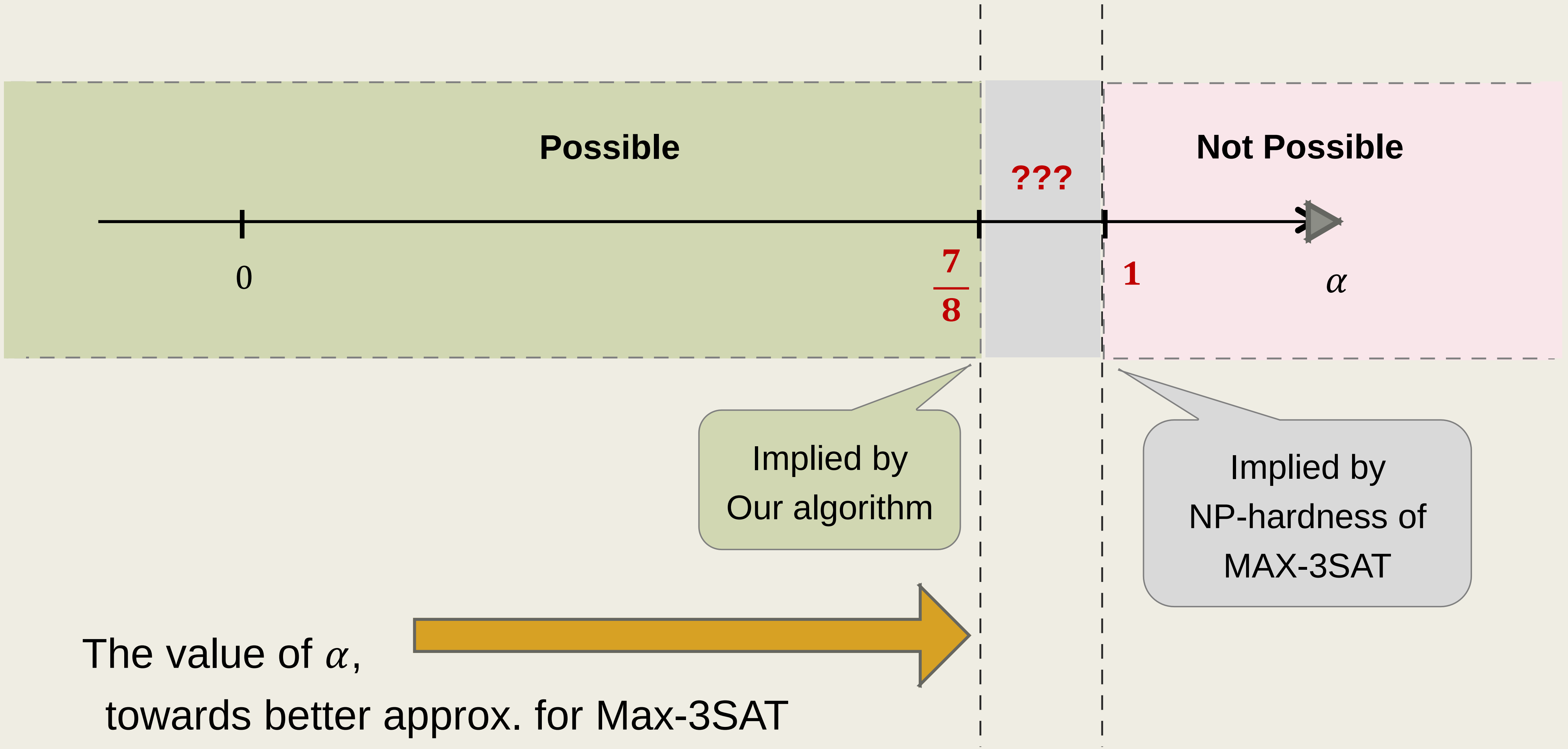
$$\mathrm{Val}(\,\mathcal{A}(\mathrm{I})\,) \;\geq\; \alpha \cdot \mathrm{Val}(\,OPT_I\,)$$

holds for all input instance I of Max-3SAT.

- We have just seen a simple randomized $7/8$-approximation algorithm, which can also be derandomized.
  - It means that, $\alpha = 7/8$ is possible to achieve.
  - So, ***a very natural question is***…

Can We Do Better than 7/8 ?

The Largest α Achievable for MAX-3SAT



Inapproximability Result of MAX-3SAT

We will see the proof next semester (hopefully).

Theorem. [Håstad, Johnson, 2001].

It is *NP-hard* to approximate MAX-3SAT to a ratio better than $\left(\frac{7}{8}+\epsilon\right)$, for any $\epsilon > 0$.

- It means, for any $\alpha > 7/8$, α-approximation algorithm for MAX-3SAT is unlikely to exist.
- The simple randomized algorithm is the best possible.

What can we do ?

- To cope with this unsatisfying fact,

  when *quitting the hard problems is unfortunately not an option…*

3. Derive ***efficient (polynomial-time) algorithms*** that computes ***near-optimal solutions***,

   i.e., the *approximation algorithms*.

   Natural questions to raise:

   - How do we *measure the quality of the solution computed*?
   - What is *the best guarantee* we can make?

Let’s take a break.

Supplements

for the Max-3SAT Problem.

Deterministic

$7/8$-approximation for Max-3SAT

Derandomizing via *Conditional Expectation*

- Consider the simple randomized algorithm for Max-3SAT:

1. For each $1 \leq i \leq n$, set x_i to be true with probability $\frac{1}{2}$.
2. Output $(x_1, x_2, \ldots, x_n)$.

- We have shown that,

$$E\left[\sum_{1 \leq i \leq m} X_i\right] \geq \frac{7}{8} OPT_I \, .$$

Derandomizing via *Conditional Expectation*

- By the definition of conditional expectation,

  we have

$$E\left[\sum_{1\le i\le m} X_i\right] = \sum_{k\in\{0,1\}}\left(\Pr[x_1 = k]\cdot E\left[\sum_{1\le i\le m} X_i \middle| x_1 = k\right]\right).$$

- Hence,

$$\max_{k\in\{0,1\}}\left(E\left[\sum_{1\le i\le m} X_i \middle| x_1 = k\right]\right) \ge E\left[\sum_{1\le i\le m} X_i\right].$$

- For example, consider the following example:

$$C_1 = (x_2 \lor \overline{x_3} \lor \overline{x_4}) \qquad C_3 = (\overline{x_1} \lor x_2 \lor x_4)$$

$$C_2 = (x_2 \lor x_3 \lor \overline{x_4}) \qquad C_4 = (\overline{x_1} \lor \overline{x_2} \lor x_3)$$

  - We have $E[\, X_3 \mid x_1 = 0\,] = 1$ and $E[\, X_3 \mid x_1 = 1\,] = \frac{3}{4}$.
  - Similarly, $E[\, X_1 \mid x_1 = 0\,] = E[\, X_1 \mid x_1 = 1\,] = \frac{7}{8}$.
  - So, $E[\, \sum X_i \mid x_1 = 0\,] = \frac{15}{4}$, $E[\, \sum X_i \mid x_1 = 1\,] = \frac{13}{4}$, while $E[\sum X_i] = \frac{7}{2}$.

Derandomizing via *Conditional Expectation*

- Continuing with the same argument, we obtain

$$\max_{k\in\{0,1\}}\left(E\left[\sum_{1\le i\le m} X_i \,\middle|\, x_1, x_2, \dots, x_{j-1}, x_j = k \right] \right)$$

$$\ge \quad E\left[\sum_{1\le i\le m} X_i \,\middle|\, x_1, x_2, \dots, x_{j-1} \right]$$

for all $1 \le j < m$.

Derandomizing via *Conditional Expectation*

- This leads to the following simple algorithm:

> – Consider the variables in any order, say, $x_1, x_2, \ldots, x_n$.
>
> For each variable,
>
> use the value that gives the ***better conditional expectation***.

$$E\left[\sum_{1\le i\le m} X_i \,\middle|\, \{x_1, \ldots, x_{i-1}\}, \boldsymbol{x_i = 0}\right] \overset{?}{>} E\left[\sum_{1\le i\le m} X_i \,\middle|\, \{x_1, \ldots, x_{i-1}\}, \boldsymbol{x_i = 1}\right]$$

Derandomizing via *Conditional Expectation*

- This leads to the following algorithm:

1. For each $1 \leq i \leq n$, do
   - Let $C = \{x_1, x_2, \dots, x_{i-1}\}$.
   - If $E\left[\sum_{1 \leq i \leq m} X_i \,\middle|\, C, \boldsymbol{x_i = 0}\right] > E\left[\sum_{1 \leq i \leq m} X_i \,\middle|\, C, \boldsymbol{x_i = 1}\right]$, then

     $x_i \leftarrow 0$.

     else

     $x_i \leftarrow 1$.
2. Output $(x_1, x_2, \dots, x_n)$.

Derandomizating via *Conditional Expectation*

- Clearly, this algorithm is deterministic and outputs a solution with value at least $\frac{7}{8} \cdot OPT_I$.

Johnson's

7/8-approximation for Max-3SAT

Another Simple $7/8$-approximation Algorithm

- For the Max-3SAT problem, we know that, the *expected value* is already large by uniform random assignment.

- This suggests the following simple algorithm:

  – Repeatedly generate a random assignment until at least $\frac{7}{8}m$ clauses are satisfied.

Repeatedly flip the coins until we succeed!

Running Time of this Algorithm

- Clearly, we have a $7/8$-approximation when this algorithm terminates.
- In the following, we bound its running time.
  - Consider one round of the algorithm.
  - Let p_j be the probability that exactly j clauses are satisfied, and p be the probability that at least $\frac{7}{8}m$ clauses are satisfied.

Lemma 1. $p \geq 1/8m$.

p is the probability that we succeed in each round.

Lemma 1. $p \geq 1/8m$.

p is the probability that we succeed in each round.

- We have

$$\frac{7}{8}m = E[\sum X_i] = \sum_{1 \le j \le m} j \cdot p_j = \sum_{0 \le j < \frac{7}{8}m} j \cdot p_j + \sum_{\frac{7}{8}m \le j \le m} j \cdot p_j$$

$j \le \left(\frac{7}{8}m - \frac{1}{8}\right)$

$j \le m$

$$\le \left(\frac{7}{8}m - \frac{1}{8}\right) \cdot \sum_{0 \le j < \frac{7}{8}m} p_j + m \cdot \sum_{\frac{7}{8}m \le j \le m} p_j$$

$$\le \left(\frac{7}{8}m - \frac{1}{8}\right) \cdot 1 + m \cdot p\,.$$

- Solving for p gives $p \geq 1/8m$.

Q: Can you point out where the slack comes from? :)

Lemma 1. $p \geq 1/8m$.

p is the probability that we succeed in each round.

- Lemma 1 says that,
  each round of the algorithm has a fair chance to succeed.

- Let N be the number of rounds the algorithm takes.

  Then,

$$\Pr[\, N = j \,] = (1 - p)^{j-1} \cdot p\,.$$

This is the *geometric distribution*!

- Let N be the number of rounds the algorithm takes.

  Then,

$$\Pr[\, N = j \,] \; = \; (1-p)^{j-1} \cdot p \, .$$

This is the *geometric distribution*!

- We have

$$E[N] \; = \; \sum_{j \geq 1} j \cdot \Pr[N = j] \; = \; \sum_{j \geq 1} j \cdot p(1-p)^{j-1}$$

$$= \; -p \cdot \frac{\mathrm{d}}{\mathrm{d}p} \sum_{j \geq 1} (1-p)^j \; = \; -p \cdot \frac{\mathrm{d}}{\mathrm{d}p} \frac{1}{p} \; = \; \frac{1}{p} \; = 8m.$$

$0 \leq p \leq 1$,
so, the series converges to $1/p$.

Notes

- In the analysis, only the assumption $E[\sum X_i] = c' \cdot m$ for some $c' > 0$ is used to prove the $O(m)$ bound on the number of rounds.