Lecture 21: NP completeness and Approximations

1. $P$ - problems with efficient algs
2. $NP$ - problems with efficiently verifiable witnesses

Fact: $P \leq NP$

Question: is $P = NP$ or $NP \neq P$?

Assumption: $P \neq NP$

If $Q \leq P$ then $NP = P$.
Q' is NP-complete (as a hardest problem in NP)
1. Q' ∈ NP
2. If Q' ∈ P then NP = P

Q < your problem

Find: 4-COLORABILITY is NP-complete

3-COLORABILITY is NP-complete

If 4-COLOR ∈ P
⇒ 3-COLOR ∈ P.

Q' is NP-complete
1. Q' ∉ NP
2. If Q' ∈ P then NP = P

⇒ Q ∈ P

NP = P
Cock Lovin Theorem

3SAT is NP-complete

Qn: does 3 satisfy or 1/False

\( x_1, x_2, \ldots, x_n \) Boolean vars

\[ \Psi = (x_1 \lor x_2 \lor \overline{x_7}) \land (x_3 \lor \overline{x_1} \lor x_5) \land (\ldots) \land (\overline{x_2} \lor x_9 \lor \overline{x_3} \lor \overline{x_7}) \]
3SAT

$\Sigma P$

3CVR

4CVR

$\Sigma P$

$\mathbf{Q}$
Makespan Minimization (MM)

- n jobs: P_1, P_2, ..., P_n ∈ Z ≥ 0
- m machines

Makespan = time at which all jobs done.

27 units of work
⇒ 29 in some machine

Assign jobs to machines to minimize the makespan

P_1, P_2, P_3, P_4, P_5

1 2 7 3 4 25

Makespan = 10 9
Q: Is MM NP complete?

No. MM is not a decision problem.

Redefine MM

given n, m, P_1, P_2, ..., P_n, \in \mathbb{Z}_{\geq 0}

and a target M

does \exists an assignment of jobs to machines s.t.

makespan (assignment) \leq M?

Q1 again: \quad MM \in NP

- Witness = assignment
- Verify: checks easily in linear time.

\begin{enumerate}
\item MM \in NP
\item \exists problem Q which is NP-complete
\end{enumerate}

\Rightarrow Q \in \Pi

\Rightarrow which is weird
Part 2: Find \( Q \) s.t. \( \text{NP} \not\subset \text{P} \) \( \Rightarrow \) \( \text{NP} \not\subset \text{P} \) (which is weird)

Thm: if \( M \in \text{P} \) \( \Rightarrow \) \( \text{PARTITION} \in \text{P} \)

Prf: Assume \( \exists \) algo \( A \) to solve \( M \) that runs in poly time

Take any instance of Partition

Construct an instance of \( M \) = \( \left\{ \begin{array}{l} n = 5 \# Q \\ m = 2 \\ P_i = c_i \\ M = \frac{1}{2} \left( \sum_{i=1}^{5} c_i \right) \end{array} \right\} \)

\( I \) is a YES instance of Partition

\( \Leftrightarrow f(I) \) in a YES instance of \( M \)
If \( Q \in \text{IP} \) then \( \text{PARTITION} \in \text{P} \)

\[ f(I) \quad \text{I instance} \]

\[ \text{such that } I \in \text{YES of PARTITION} \]

\[ f(I) \text{ is YES instance of } Q \]

Karp reduction

\[ \text{encourage you to use} \]

Q: Can you approximate the min makespan?
Algorithms for MM

Greedy

\[ \forall i = 1 \ldots n \]

Put \( P_i \) on some machine with least load.

\[ \text{Thm: } \text{Makespan (Greedy)} \leq 2 \cdot \text{OPTimal makespan} \]

LB(each case agree about)

LB(harder case difficult?)

LB(j) \leq \text{OPT(j)}

\[ P_1, P_2, \ldots, P_n \]

\[ P_{\text{avg}} = \frac{\sum_{i=1}^{n} P_i}{n} \leq \text{OPT} \]

\[ P_{\text{max}} \leq \text{OPT} \]
Claim: \( \text{Makeup} (\text{Greedy} (I)) \leq P_{\text{max}} (I) + P_{\text{avg}} (I) \)

\[
\text{Makeup} = P_j + \underbrace{\text{rest}} \text{ load of least loaded m/c } @ \text{ when } j \text{ be added}
\leq P_{\text{max}} + \left(\sum_{i=1}^{d-1} P_i\right)/m \leq P_{\text{max}} + P_{\text{avg}} \leq 2O P\text{.}
\]

2-approximation. ☺
Fact: 3 instances where \( \text{Makespan (Greedy)} \geq (2 - \frac{1}{3}) \text{OPT} \)

**Sort Greedy**

1. Sort so that \( P_1 \geq P_2 \geq \cdots \geq P_n \)
2. Run greedy

Then: \( \text{Makespan (Sorted Greedy)} \leq \frac{3}{2} \text{OPT} \).
Thm: Makespan \((S_{\text{sorted}} G) \leq \)

\[ Pf: \text{Makespan} \leq P_j + \text{rest} \]

\[ \text{SpS rest} = 0 \Rightarrow P_j + \text{rest} \leq P_{\text{max}} + 0 \leq \text{OPT} \]

else claim: \exists \geq m+1 \text{jobs of size } \geq P_j \Rightarrow \text{OPT } \geq 2P_j

\[ P_j + \text{rest} \leq \frac{\text{OPT}}{2} + P_{\text{avg}} \leq \frac{3}{2} \text{OPT} \]
Mak espan: 2

\[ \frac{3}{2} \rightarrow \frac{4}{3} \]

Greedy sorted greedy

\[ \leq 0.136 \text{ apx also, runtime } O(n) \]

Algorithm: 2-approx for Vertex Cover

\[ \leq 1.36 \text{ apx } \Rightarrow P = NP (\text{which is weird}) \]