Approximation Algorithms

Lecture 1

Administrative Information

- **Instructors:** Pranabendu Misra & Nithin Varma
- **Lectures:** Thu & Fri; 3:30 PM to 4:45 PM
- **Prerequisites**: Algorithms, Discrete Maths
- **References:**
  - Approximation Algorithms – Vijay Vazirani
  - Design of Approximation Algorithms – David Williamson and David Shmoys
  - Lecture Notes by Chandra Chekuri
- **Evaluation Components**: Assignments (3-4) , Midsem, Endsem, Attendance and class participation
- **Collaboration Policy:** You can discuss assignments with other students who take the course. But you must write your own solutions and list your collaborators for each problem.

- In **Algorithms** course
  - Poly-time algorithms for several interesting problems
  - Different design paradigms – divide and conquer, greedy, dynamic programming
  - NP-completeness and NP-hardness

- Several interesting problems are NP-complete or NP-hard, e.g., Min Vertex Cover, SAT, Travelling Salesman Problem,..

  - Exp. time algos. are known for many of them

- No polynomial time algorithms that solve above exactly unless P = NP

- Heuristics that work well in practice, e.g, SAT solvers
- Heuristics do not give provable worst-case guarantees

We want algos. that
- run in poly. time
- output a "good" solution

on <u>every input</u>

- An α-approximation algorithm for an optimization problem runs in poly-time and outputs a solution whose value is within α factor of the value of the optimal solution
  - For minimization problems, $\alpha > 1$
  - For maximization problems, $\alpha < 1$

eg:- for MAX CLIQUE $\frac{2}{n}$ - approx algo.

- eg: min Vertex Cover
  $\exists$. 2-approx algo
  the algo outputs a set $\hat{V}$ s.t.
  $OPT \le |\hat{V}| \le 2 \cdot OPT$
  card. of min VC

- Reasons to learn approximation algorithms

- understand the rel. difficulty among NP-hard optimization problems
- techniques useful to design heuristics

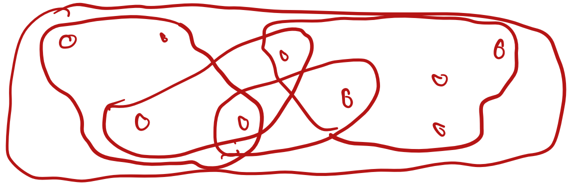
- In this course
  - Design poly-time algorithms that solve problems approximately
  - Strategies for the design of such approx. algorithms – greedy, primal-dual scheme, semi-definite programming, local search,..
  - A bit about hardness of approximation
  - Some advanced/ recent topics

- **Today**: Approximation algorithm for SET COVER & MAXIMUM COVERAGE

$U = \{e_1, e_2, \ldots, e_n\}$ $\quad F_i \subseteq U \ \forall i \in [m]$

- SET COVER
  - Given universe U of n elements and a family $F = \{F_1, \ldots F_m\}$ of m subsets of U
  - **Goal**: Find the smallest set $I \subseteq [m]$ such that $\bigcup_{i \in I} F_i = U$

Minimization problem



$\boxed{\text{min VC problem} \quad \text{vertex cover}}$

exercise

- MAXIMUM COVERAGE
  - Same setup as before, also given integer $k \leq m$
  - **Goal**: Select k sets from F such that their union has maximum cardinality

Greedy Cover (F, U)

- Initially, all elements in U are uncovered
- Repeat
  - Select the set in $\boldsymbol{F}$ that covers the maximum number of uncovered elements in $\boldsymbol{U}$
  - Mark elements in the selected set as covered
- Until ………

→ k sets are selected (MAX COVERAGE)

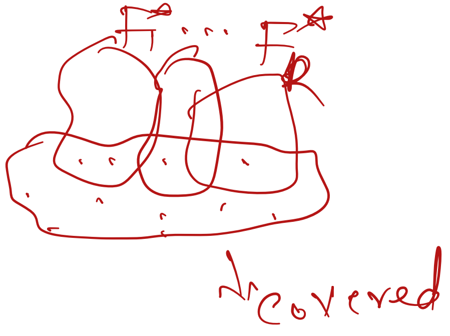
→ every element in U is covered (SET COVER)

- **Theorem:** Greedy algorithm gives a $(1 - \frac{1}{e})$ approximation to MAX COVERAGE problem

- OPT – optimal value
- x_i - Number of new elements covered in $\boldsymbol{i}$-th iteration
- $y_i = \sum_{j \in [i]} x_j$; number of elements covered after $\boldsymbol{i}$ iterations
- $z_i = OPT - y_i$

- Claim: $x_{i+1} \ge z_i/k$ for all $i \in [k] \cup \{0\}$

Exercise:
Can you improve to $\frac{z_i}{k-i}$?



Proof:-

At the end of i iterations, let us consider an optimal solution for MAX-COVERAGE $\underbrace{F_1^*, F_2^*, \dots, F_k^*}$

$\ge z_i$ elements from $\underline{\qquad}$ are not yet covered

$\exists\, j \in [k]$, F_j^* has $\ge \frac{z_i}{k}$ uncovered elements

$\Rightarrow x_{i+1} \ge \frac{z_i}{k}$ $\square$

- Claim: $y_i \geq \mathrm{OPT} \cdot \left(1 - \left(1 - \frac{1}{k}\right)^i\right)$

Proof: by induction; $i = 1$

$$y_1 \geq \frac{\mathrm{OPT}}{k}$$

complete the inductive argument.

- Proof of Theorem:

$$\left.\begin{matrix}\text{Value of solution}\\ \text{output}\end{matrix}\right\} = y_k \geq \mathrm{OPT}\left(1 - \left(1 - \frac{1}{k}\right)^k\right)$$

$$\geq \mathrm{OPT} \cdot \left(1 - \frac{1}{e}\right)$$

- **Theorem**: Greedy algorithm is a $\ln n + 1$ approximation to SET COVER.

- Let k^* denote the optimal value of the SET COVER instance

- Consider a MAX COVERAGE instance with $k = k^*$

- An optimal solution for given SET COVER instance is also an optimal solution for MAX COVERAGE with $k = k^*$

- The value of the optimal solution of MAX COVERAGE instance is n

- $z_i \leq n \cdot \left(1 - \frac{1}{k^*}\right)^i$ for all i

OPT-MAX-COVERAGE

k

$$z_t \leq n \cdot \left(1 - \frac{1}{k^*}\right)^{k^* \ln\left(\frac{n}{k^*}\right)} \leq n \cdot e^{-\ln\left(\frac{n}{k^*}\right)} = k^*$$

- After $t = k^* \ln \frac{n}{k^*}$ iterations, at most k^* elements remain uncovered

$\Rightarrow$ Greedy can go for $\leq k^*$ more iterations

OPT-SET-COVER

- Proof of Theorem:

$$\left.\begin{array}{l}\text{\# iterations until all} \\ \text{elements covered}\end{array}\right\} \leq k^* \ln\left(\frac{n}{k^*}\right) + k^* \leq k^*(\ln n + 1)$$

- **Corollary:** If each set has at most d elements, then Greedy Cover gives a $\ln d + 1$ approximation

$$k^* \geq \frac{n}{d} \Rightarrow d \geq \frac{n}{k^*} \Rightarrow k^*(\ln d + 1)$$

Weighted SET COVER problem

- Weighted SET COVER
  - Given universe U of n elements and a family $F = \{F_1, \dots F_m\}$ of m subsets of U, where set F_i has weight w_i for $i \in [m]$
  - **Goal**: Find the set $I \subseteq [m]$ that minimizes $\sum_{i \in I} w_i$ such that $\bigcup_{i \in I} F_i = U$ ~~and~~

Unweighted set cover

$W_1 = W_2 = \dots = W_m = 1$

Integer Program for Weighted SET COVER

- Variable x_i to indicate the presence or absence of set F_i for $i \in [m]$ in a solution to SET COVER

$$\min \sum_{i \in [m]} w_i x_i$$

$$\sum_{i : e \in F_i} x_i \geq 1 \quad \forall e \in U$$

$$\underbrace{x_i \in \{0,1\}} \quad \forall i \in [m]$$

- Solving the integer program is NP-hard

  - reduction from wtd. SET COVER

- Relaxing the integer program

Solve it in poly-time with LP solvers

$$\begin{cases} \min \sum_{i \in [m]} w_i x_i & \\ \sum_{i : e \in F_i} x_i \geq 1 & \forall e \in U \\ x_i \geq 0 & \forall i \in [m] \end{cases} \quad \text{(LP)}$$

- How to get an integer solution from the fractional solution? Rounding!

- Let f be the maximum number of sets that an element is a part of

  - No element belongs to more than f sets

- **Deterministic Rounding**

Suppose

- $\{x_i^*\}_{i \in [m]}$ be the fractional optimal soln. to LP

- Include F_i in set cover iff $x_i^* \geq \frac{1}{f}$

- **Claim**: The solution obtained after rounding is a valid SET COVER

for element e

$$\underbrace{\boxed{x_1 + x_{100} + x_{23} + \cdots + x_{500}}}_{\leq f} \geq 1$$

- **Claim:** The solution is an f-approximation to SET COVER

$\rightarrow$ each variable is rounded up by a factor of $\leq f$

after rounding, the solution will have cost $\leq f$ times cost opt. frac. solution.

$\boxtimes$

- **Next Time**: Primal Dual Method for SET COVER