1/20/26: Intro, Vertex Cover

Course Staff:

- Me: www.cs.jhu.edu/~mdinitz/
- TAs: None!

Course webpage: (Not Canvas)

- www.cs.jhu.edu/~mdinitz/classes/ApproxAlgorithms/Spring2026/
- Courselore
- Gradescope

Prereq: Intro Algorithms

Grading:
- 50% homework
- 35% final project
- 15% Participation

Final project: Flexible!
- Possibly presentations last week of class, reports by scheduled exam

Note: PhD-level class!
- Course schedule more flexible
- Not much handholding
- Hopefully not a ton of work, so you can focus on research

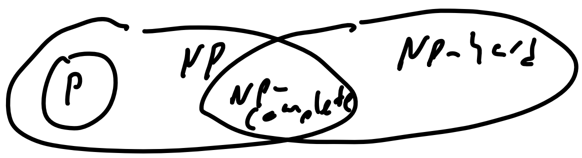
Approximation Algorithms: What and Why

P: Solvable in polytime

NP: Verifiable in polytime

NP-hard: Problems that all problems in NP reduce to in polytime

NP-complete: in NP and NP-hard



Given NP-hard <u>optimization</u> problem, what do we want that NP-hardness prevents?

1. Find the optimal solution
2. In polynomial time
3. For every instance

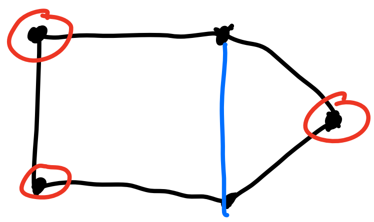
Could relax any of these!

Problem consists of:

1) Description of inputs/instances
2) Description of feasible solutions for input
3) Objective function

Example: Vertex Cover

- Input: Graph $G = (V, E)$
- Feasible solutions: $V' \subseteq V$ s.t. $e \cap V' \neq \emptyset \ \forall e \in E$
- Objective: minimize $|V'|$



Def: The optimal solution is the feasible solution with the best objective value

Def: A some problem, I instance of A

- ALG: some algorithm for A
- $OPT(I)$: objective value of optimal solution for I
- $ALG(I)$: objective value of solution returned by ALG on I

ALG is an α-approximation if:

- always returns a feasible solution
- runs in polytime
- $\dfrac{ALG(I)}{OPT(I)} \leq \alpha$ $\forall$ instances I of A (min problem)

  $\dfrac{ALG(I)}{OPT(I)} \geq \alpha$ $\forall$ instances I of A (max problem)

α is the approximation ratio or approximation factor

why?

- Really do need algorithms for NP-hard problems!
- "Fine-grained complexity": not all NP-hard problems are the same!

  A_1, A_2 problems.

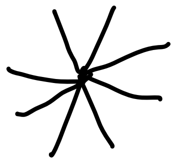
  A_1: design a 2-approximation

  A_2: Assuming $P \neq NP$, no α-approx for $\alpha < 10$

- Forcing ourselves to handle worst-case forces new techniques

Approximating Vertex Cover

Idea 1: Arbitrarily add vertices until have feasible vertex cover

Star: 

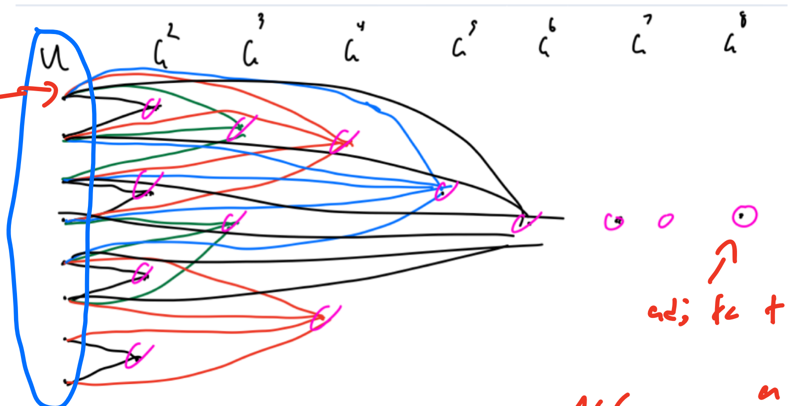
OPT: 1

ALG: $n-1$

Idea 2: Greedy algorithm

Add node covering most uncovered edges

adj to f edges



adj to f edges

$|U| = f$

$n = \Theta(f \log f)$

$OPT = |U| = f \leq \frac{n}{\log n}$

$ALG = n - |U|$

$= \Theta(f \log f)$ n

$\Rightarrow \frac{ALG}{OPT} = \frac{n}{\frac{n}{\log n}} = \log n$

ALG: n

OPT: $\frac{n}{\log n}$

Better Algorithm:

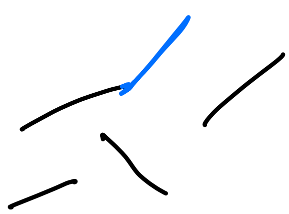
- $S = \emptyset$
- while $E \neq \emptyset$ {
    - Let $\{u,v\}$ be arbitrary edge
    - $S \leftarrow S \cup \{u,v\}$
    - Remove u,v, all incident edges from G

}

Poly time:

<u>Lemma</u>: S feasible (S a vertex cover)

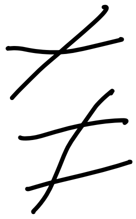
<u>Pf</u>: ✓



<u>Def</u>: $M \subseteq E$ is a **matching** if no two edges in M share an endpoint: $e \cap e' = \emptyset$ $\forall e, e' \in M$ with $e \neq e'$

Lemma: Let M be a matching, and S be a vertex cover. Then $|S| \geq |M|$

Pf:



Thm: Algorithm is a 2-approx.

Pf: polytime, feasible: ✓

Let S^* optimal vertex cover.

WTS: $|S| \leq 2|S^*|$

Structure of edges considered by algorithm: matching M

$$\Rightarrow \quad |S| \overset{\text{def of ALG}}{=} 2|M| \leq 2|S^*|$$

by lemma

Q: Are we happy?

Q1: Is analysis tight?

A: yes (star, $K_{n,n}$, ...)

Q2: Is algorithm tight?

- Best known approx: $2 - \frac{1}{\sqrt{\log n}}$
- Assuming $P \neq NP$, no alg better than $10\sqrt{5} - 21 \approx 1.3606$-approx
- Assuming UGC, $\forall$ constant $\varepsilon > 0$ there is no $(2-\varepsilon)$-approx