Approximation

There are a lot of hard problems out there, our unit on NP-completeness detailed many of them. Faced with this, one idea is to simply give up. Another idea is within some polynomial time bound to try to get as good of a solution as possible: Apply some trivial greedy heuristic, but also guarantee the answer gets to what we want.

The approximation ratio of an approximation algorithm is defined to be

$$\rho(n) = \max\left(\frac{sol}{opt}, \frac{opt}{sol}\right)$$

whether or not you overestimate or underestimate it by how much is $\rho(n)$.

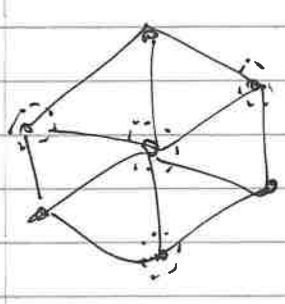
An approximation scheme is one in which the run time is a function of both n, ϵ and more time can give a more exact solution

PTAS $O(n^{2/\epsilon})$. If you only want 10% error $\epsilon = 0.1 \Rightarrow O(n^{20})$

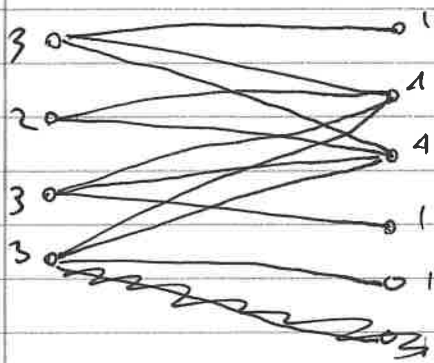
FPTAS $O((1/\epsilon)^2 n^3)$ small ϵ incur constant factor increases.

in n and $(1/\epsilon)$

Recall we proved the decision variant of VC was NP-complete.
$SAT \leq_p 3SAT \leq_p INDSET \leq_p VC$. $VC = \{\langle G, g\rangle \mid G \text{ has a VC of } \leq g\}$
a vertex cover is a selection $S \subseteq V$ such that all of E shares at least one end in S.



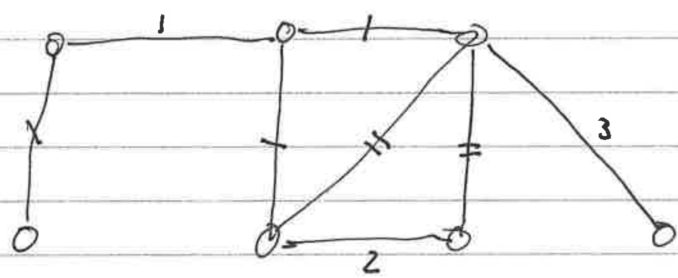
suppose we want to search for a vertex cover, a minimal one instead. First we show when an obvious greedy scheme does not work well, then we give one which approximates VC by two.



greedily choose and remove highest degree vertices. This approximation scheme is really bad, need $S \geq 6$. but since bipartite, we know solution is of size 4. This example can generalize to create instances this greedy algorithm is bad at approximation.

we give a 2-approximation algorithm for VC. Its output is guaranteed to be no more than <u>twice</u> the optimal.

Approx VC (G):
sol = ∅
E' = E
while E' ≠ ∅
choose (randomly) e=(u,v) of E'
sol.append u, v
remove from E' all edges connected to u or v.

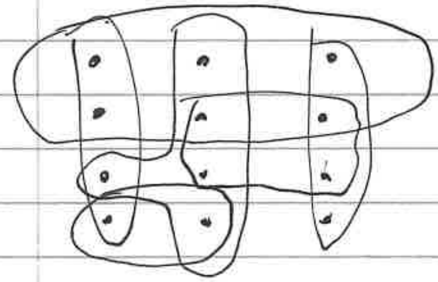


O(V+E) runtime, given E' represented as an adjacency list.
we prove $|sol| \leq 2 \cdot |opt|$
note an edge e is picked with <u>no</u> endpoints currently in sol.
let A = set of edges picked. <u>no two edges share same same vertex</u> of opt, so $|opt| \geq |A|$. also, e is chosen where both ends also not in sol. so $|sol| \leq 2|A|$.

$|sol| \leq 2|A| \leq 2|opt|$

prove we can 2-approx even if it is hard to find solutions. Matching in graphs set cover given $\{X, F = (S_1, \dots S_n)\}$ where $S_i \subseteq X$ want to find a selection of subsets which covers X. minimum set cover.

NP-complete.



greedy approach

```
Greedy Approx SC (X, F):
    U = X
    C = ∅
    while U ≠ ∅
        choose S_i to maximize |S ∩ U|
        U = U \ S_i
        C.append(S_i)
    return C.
```

This is a $(\ln(|X|+1))$ approximation algorithm. Proof really uses harmonic numbers.
if $|opt| = k$, we prove $|sol| = k\ln(n)$.

Let n_t be # of X still not covered after t rounds selections.
since $|opt| = k$, some set has n_t/k elements. so

$n_{t+1} \le n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right)$. we upperbound n_t by

$n_t \le n_0\left(1 - 1/k\right)^t$ $\quad$ $(1 - x \le e^{-x}$ with $= \Leftrightarrow x = 0)$

$n_t \le n_0(1 - 1/k)^t < n_0(e^{-1/k})^t = ne^{-t/k}$

$t = k\ln(n) \Rightarrow n_t \le 1$.

non-constant approximation factor

MAX 3SAT   recall 3SAT is NP-complete. in MAX3SAT we want to satisfy as many clauses as possible. MAX 3SAT is also NP-hard! in fact, some deep theorems, if you can approximate SAT too well you can solve SAT fast.

Approx Max3SAT(ϕ)
    choose $x_1 \ldots x_n$ randomly.

fix some clause of size 3. what is the probability a random assignment has this clause ~~sa~~ satisfied?   7/8.

$Z = \sum_{i=1}^{k} Z_i$ $Z_i = 1$ if C_i is satisfied. what is expectation of Z_i?

$$E[Z] = E\left[\sum_{i=1}^{k} Z_i\right] = \sum_{i=1}^{k} E[Z_i] = 7/8\, k$$

a random assignment will solve 7/8ths of the clauses of SAT!

Johnson 7/8th Approx Max 3SAT(ϕ)
    ~~choose~~ while True
        generate $x_1 \ldots x_n$ randomly.
        if 7/8th of clauses of ϕ satisfied
            break and return $x_1 \ldots x_n$

Alg only stops if such an assignment exists. a random variable is atleast its expectation some of the time. expected number of rounds is $O(k)$.