We have discussed pre-processing as a way of tackling NP complete problems.

Now, lets look at Approximations.

The idea is to come up with an approximate soln for the benefit of polynomial running time.

eg: instead of the shortest tour, lets find 'quite' a short tour

In what scenarios, would approximations be appropriate?

→ Stakes are low

→ Approximations are "good" (eg: ±1%)

→ Exact solns. cannot be obtained

Analysing such algorithms is important to know the qualities of the same

Approximations aren't usually the first thing you should resort to.

Approximation quality:

→ Even if not exact, we want our soln. to to have some quality.

→ Constant factor approximation

Minimization ⇒ soln ≤ C. optimum soln.

maximization ⇒ soln ≥ $\frac{1}{C}$. optimization

eg: factor 2 approx. for V.C. with soln = 100. (size)

max. #vertices in optimum soln.

$100 \leq 2 \cdot (0)$

$0 \geq 50$

$100 \geq \frac{1}{2} \cdot q_{max}$

$q_{max} \leq 200$

$O_{max} = 100$ (we already found a soln of size 100)

$O_{min} = 50$ $(\theta \geq 50)$

eg: factor -2 with clique returns soln of size 100

↳ maximization problem

$$100 \geq \frac{1}{c} \cdot \theta$$

$$200 \geq \theta \qquad \theta_{max} = 200$$

$\theta_{min} = 100$ (given)

θ_{min} ~~= 200~~

θ_{max}

optimum soln.

→ Polynomial time approximation scheme

• quality of the approximation depends on running time.

Vertex cover

Algo 1 (~~factor-2 approximate~~)

While some edges uncovered:

e ← an uncovered edge

put both ~~edges~~ endpoints into the vertex cover

Algo 2 (greedy)

while some edges uncovered

v ← vertex that can cover most edges

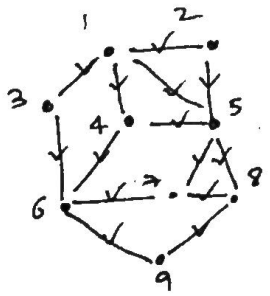
put v into the vertex cover

If "feels" Algo 2 should have better approximations.

X WRONG!

In terms of approximation quality Algo 1 is better. Let's discuss.

Consider



~~Using Algo 1~~, we get the optimal soln as

$\{5, 6, 1, 8\}$

Size $\Rightarrow$ 4

In Algo 1, in each loop picks at least 1 vertex from the min vertex cover

Since no matter what edge we take it is connected to ≥ 1 of the vertices in the min. vertex cover

In Algo 1., ~~takes at most~~ the loop runs at most k times, where k is the size of the minimum vertex cover.

So we know two qualities of this algo.

- → In each iteration we put atleast one vertex and at most two vertices
- → The loop runs at most K times.

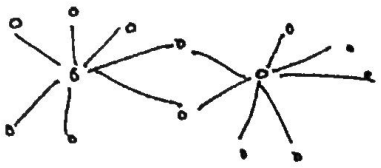
Size of the soln $\Rightarrow \leq 2k$

$\geq k$

↳ factor-2 approximation.

Let's now analyze greedy algorithm.

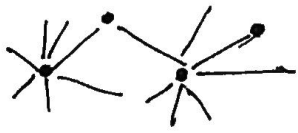
Consider



Size of optimum soln: 2

Size of greedy soln: 2

Size of take-2 soln: 4



But we haven't proven approx. quality for greedy

Greedy algo has a worse approximation factor (>2)

Although there's no guarantee, greedy algo runs well on most instances.

In practice, we can just run both algorithms.

Sounds like a good idea ≠ good idea