Vertex Cover

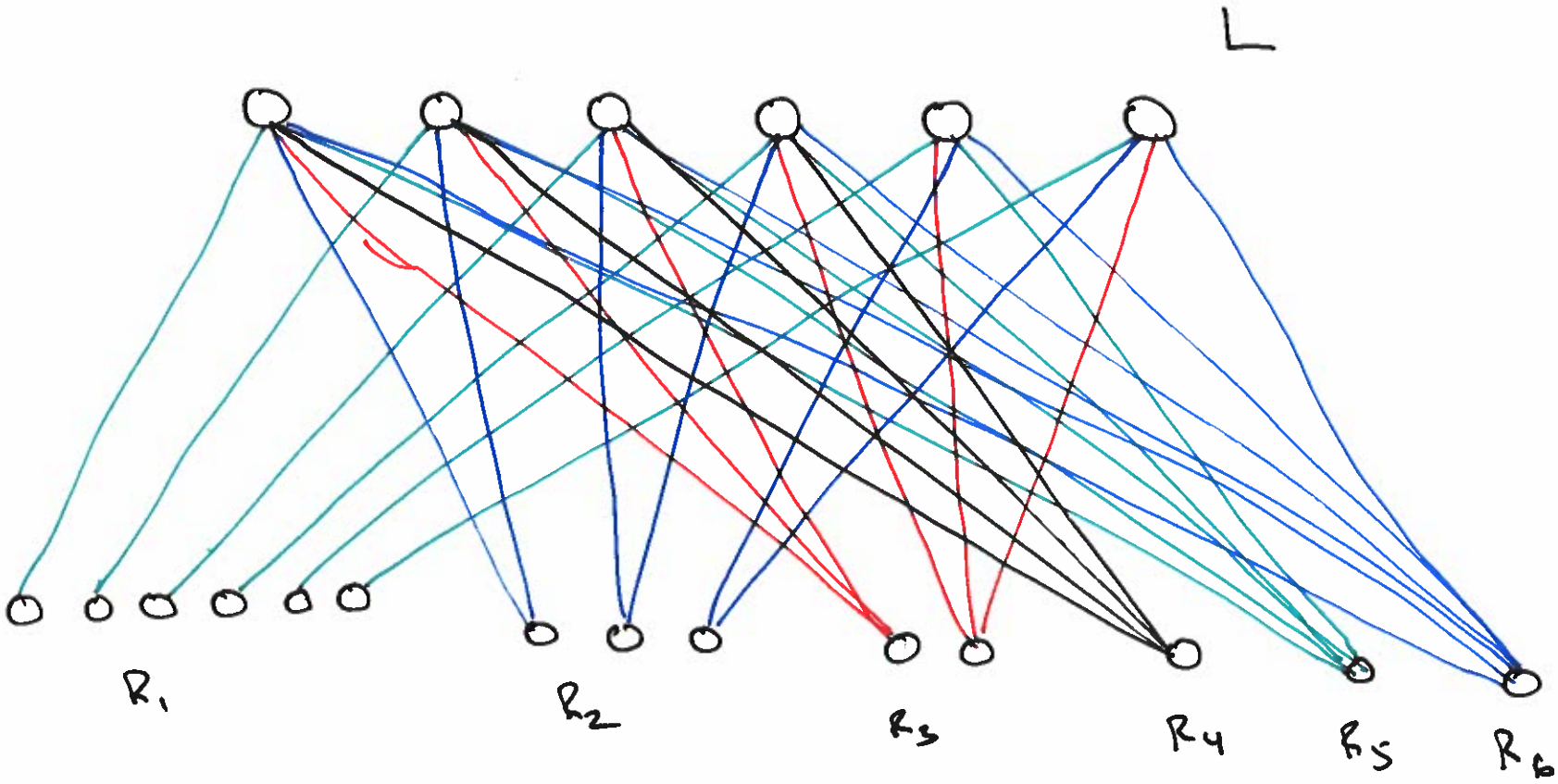
- Given an undirected graph $G = (V, E)$, find a subset $V' \subseteq V$ such that if $(u, v) \in E$ then either $u \in V'$ or $v \in V'$ or both. Find a V' such that $|V'|$ is minimized.

Greedy-VC

- select a vertex v of maximal degree
- add v to V'
- remove all edge incident to v from E

```
GreedyVC(G)
    V' ← ∅
    while E is not empty
        pick a vertex v ∈ V of maximal degree
        V' ← V' ∪ {v}
        E ← E \ { e ∈ E | v ∈ e }
    return V'
```

- consider a bipartite graph with a set L of t nodes on the left and then a collection of sets $R_1, R_2, R_3, \ldots$ of nodes on the right. where each R_i has $\lfloor \frac{t}{i} \rfloor$ nodes
- this graph has $n \in \Theta(t \log t)$ nodes
- Connect each set R_i to L so that each $v \in R_i$ has i neighbors in L and no two vertices in R_i share any neighbor in common


L
R1
R2
R3
R4
R5
R6

- The optimal vertex cover is L of size t but this greedy solution might first choose R_t then R_{t-1} and so on down to R_1.
- this results in a vertex cover of size $n - t$
- this achieves a $O(\log n)$ approximation ratio

Approximation

- An algorithm for a problem of size n has an approximation ratio $\rho(n)$ if for any input the algorithm produces a solution of cost/size c such that

$$\max\left(\frac{c}{c_{opt}}, \frac{c_{opt}}{c}\right) \leq \rho(n)$$

Dumb Vertex Cover

```
DVC(G)
    V' ← ∅
    while E is not empty
        (u,v) ← any edge in E
        V' ← V' ∪ {u,v}
        E' ← E \ { all edges incident to u or v }
    return V'
```

Theorem

Dumb vertex cover is a 2-approximation algorithm for VC.

Proof

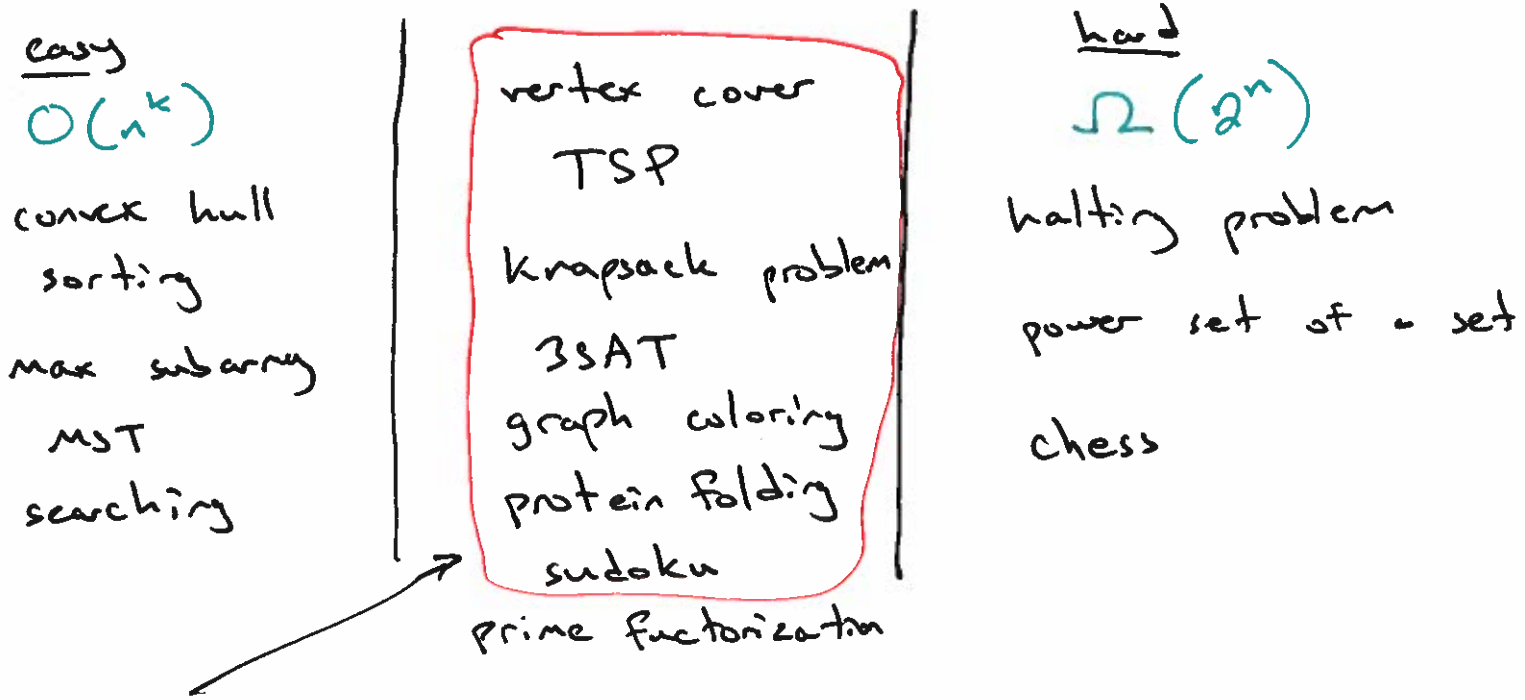
- let A be the set of edges picked by DVC
- the optimal vertex cover V'_{opt} must include at least one endpoint for each edge in A.
- No two edges in A share an endpoint.
- $|A|$ is a lower bound for $|V'_{opt}|$
- the number of vertices in V' is $2|A|$

$$|V'| = 2|A|$$

- therefore

$$|V'| \leq 2|V'_{opt}|$$

Complexity Theory



Decision Problem : problem with a yes/no answer

NP-Complete

Complexity Classes

P

- informally: the set of all decision problems we can efficiently solve.
- the set of all decision problems that have a polynomial time solution

$$P = \bigcup_{k \geq 0} O(n^k)$$

NP

- the set of all decision problems that can be verified in polynomial time
- the set of all decision problems that be solved in poly-time by a ~~not~~ non-deterministic TM.

P vs NP

is $P \subsetneq NP$ or is $P = NP$

is there a problem that can be verified in poly-time but not solved in poly-time?

NP-Complete

- the hardest problems in NP
- a solution to any NP-Complete problem can be used to solve any NP problem.

Polynomial Reduction

- to reduce problem A to problem B means that any instance of problem A can be transformed into an instance of problem B

- $$A \leq_p B$$

- on input to A, ω, if $\omega \in A$ then $F(\omega) \in B$ and if $\omega \notin A$ the $F(\omega) \notin B$.

An algorithm for B can be used to solve A.

NPC Def

A language A is NP-Complete iff

1. $\forall L \in NP \quad L \leq_p A$
2. $A \in NP$

Cook-Levin Theorem

3SAT is NP-Complete

- the language of boolean formulas can be used to represent any verifiable language

Polynomial time reductions are transitive

if $A \leq_p B$ and $B \leq_p C$ then $A \leq_p C$