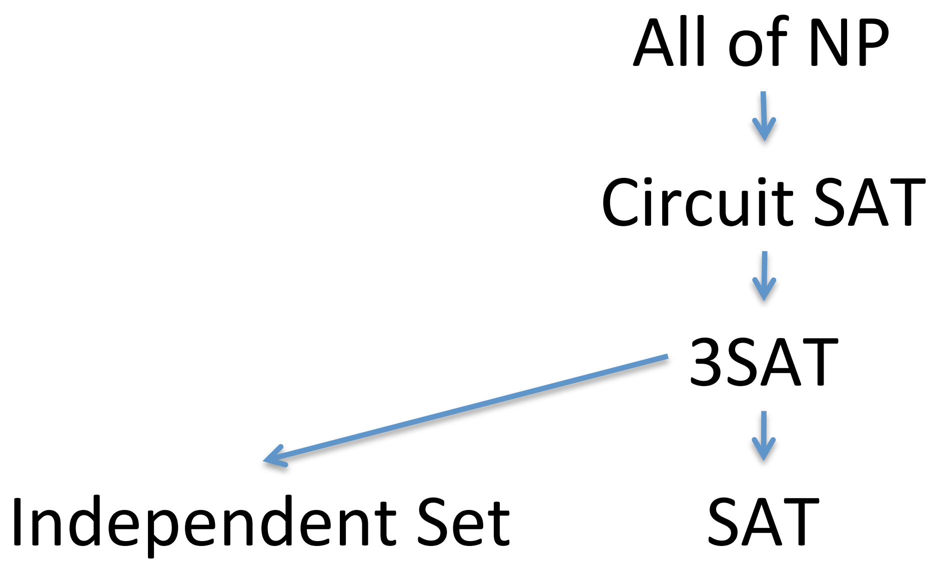
CS 161: Design and Analysis of Algorithms

NP-Complete II: More NP-Complete Problems

- The problems
- Some reductions

So Far



Recipe For Proving NP-Completeness

- Pick a well-known NP-Complete problem
- Prove that your problem can be used to solve the well-known NP-Complete problem
  - Given an instance of the well-known problem, construct an instance of your problem
  - Show that the instance of the well-known problem has a solution if and only if the instance of your problem does

Hamiltonian Cycle

- Given a graph $G = (V,E)$, is there a simple cycle that visits every node exactly once, and returns to the starting point?

Hamiltonian Path

- Given a graph G and two nodes s and t, is there a simple path from s to t that visits all nodes in G?

Longest Path

- Given a graph G and a goal b, determine if there is a simple path in G with length at least b

3D Matching

- n boys, n girls, n pets
- Set of triples (b,g,p) that means b, g, and p go well together
- Find a way to match up each boy, girl, and pet

Vertex Cover

- Find a collection of at most b nodes that touch every edge in the graph

Set Cover

- Given a set E and a collection of subsets $\{S_i\}$, pick at most b subsets such that their union is E

Clique

- A **clique** of a graph G = (V,E) is a set of n nodes such that every two nodes in the set have an edge between them
- The Clique problem is to, given a goal g, find a clique on at least g nodes

Subset Sum

- Given a set of integers v_i, find a subset of the integers whose sum is exactly V

Knapsack

- Given a set of of items {1,…,n}, weights for each item w_i, value of each item v_i, weight capacity W, and a target value V, find a subset of the items whose total weight is at most W and whose value is at least V

Integer Linear Programming

$$\max \sum_i c_i x_i$$

$$\sum_i A_{j,i} x_i \leq b_j \forall j$$

$$x_i \geq 0 \forall i$$

$$x_i \in \mathbb{Z}$$

Zero-One Equations

$$\text{find } x_i$$

$$\sum_i A_{j,i} x_i = 1 \forall j$$

$$A_{j,i} \in \{0,1\}$$

$$x_i \in \{0,1\}$$

Scheduling

- Given a set of n jobs, can only work on 1 at a time
- Job i is available to start working on at time r_i, due by time d_i, and has duration t_i
- Can we complete all the jobs before their deadlines?

The Reductions

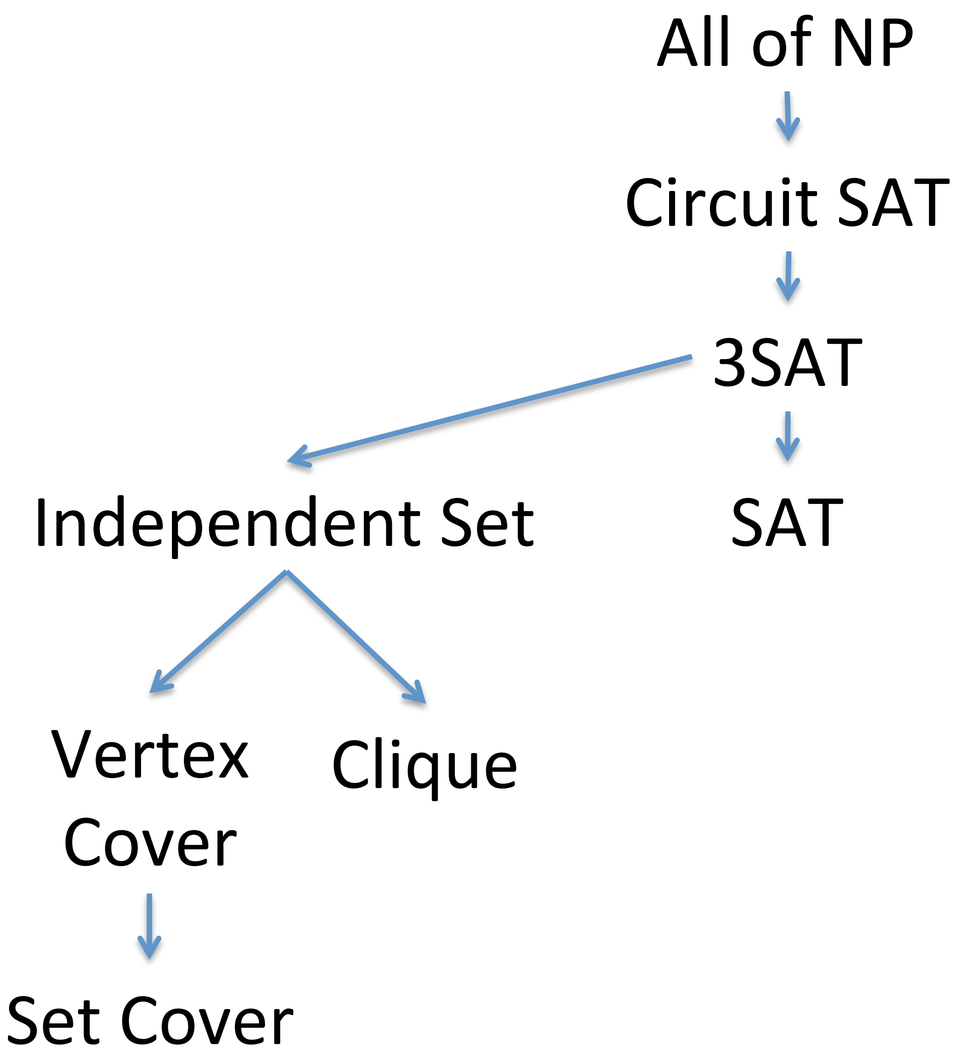
Independent Set $\leq_P$ Vertex Cover

- S is an independent set if and only if V-S is a vertex cover
- Given a graph G and a goal b, for the independent set problem, construct (G,|V|-b) as an instance of Vertex Cover

Independent Set $\leq_P$ Clique

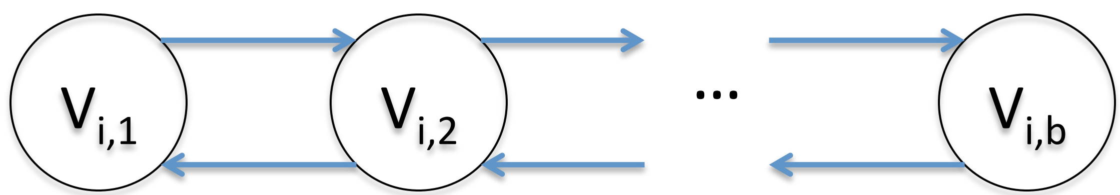
- Define the complement G* = (V,E*) of a graph G = (V,E) where E* consists of every edge not in E
- S is an independent set of G if and only if S is a clique in G*
- Given a graph G and a goal b for the Independent Set problem, simply compute (G*,b) as an instance of Clique

So Far


All of NP
Circuit SAT
3SAT
SAT
Independent Set
Vertex Cover
Clique
Set Cover

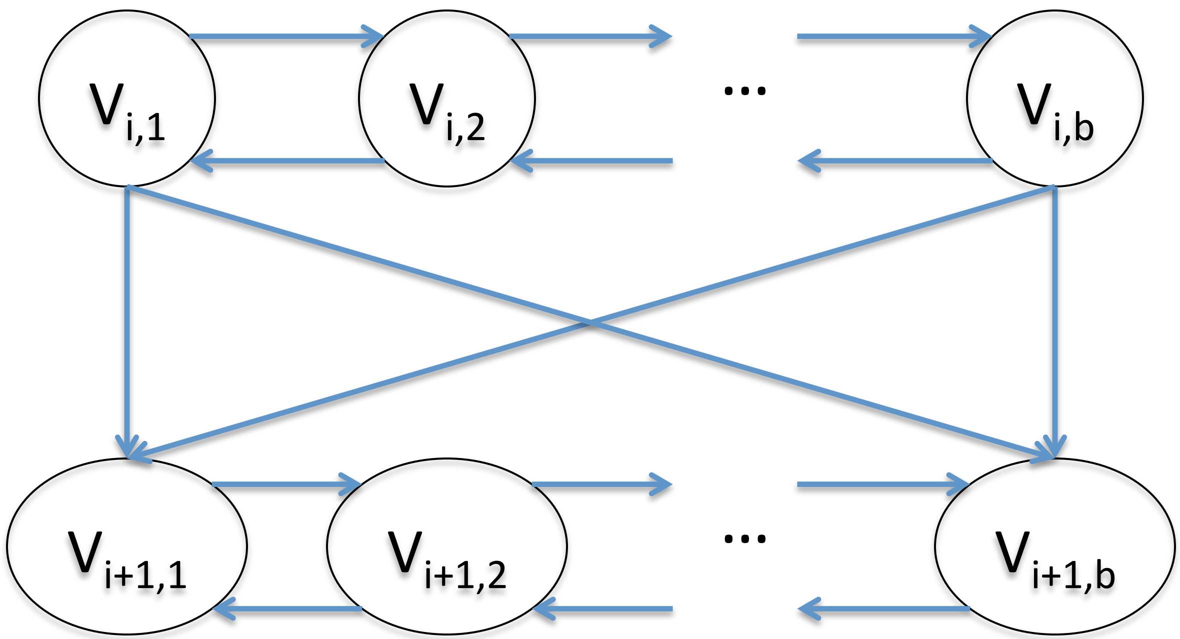
3SAT $\leq_P$ Directed Hamiltonian Cycle

- Given a 3SAT instance with n variables, k clauses
- Pick some b >> k (to be determined later)
- Construct n paths P_i



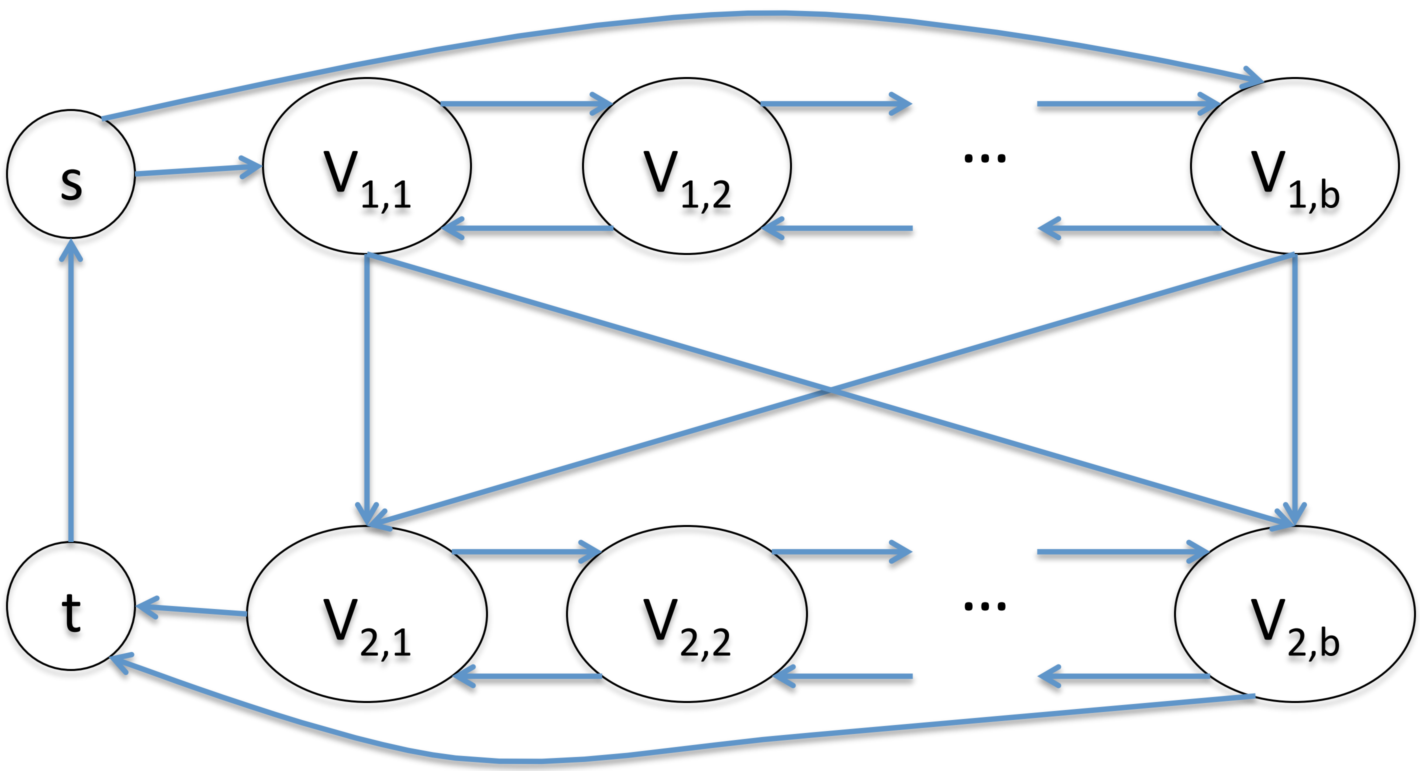
3SAT $\leq_P$ Directed Hamiltonian Cycle

- Add edges $(v_{i,1}, v_{i+1,1})$, $(v_{i,1}, v_{i+1,b})$, $(v_{i,b}, v_{i+1,1})$, $(v_{i,b}, v_{i+1,b})$



3SAT $\leq_P$ Directed Hamiltonian Cycle

- Add a source node s, target node t, and 5 edges: $(s,v_{1,1})$, $(s,v_{1,b})$, $(v_{n,1},t)$, $(v_{n,b},t)$, (t,s)

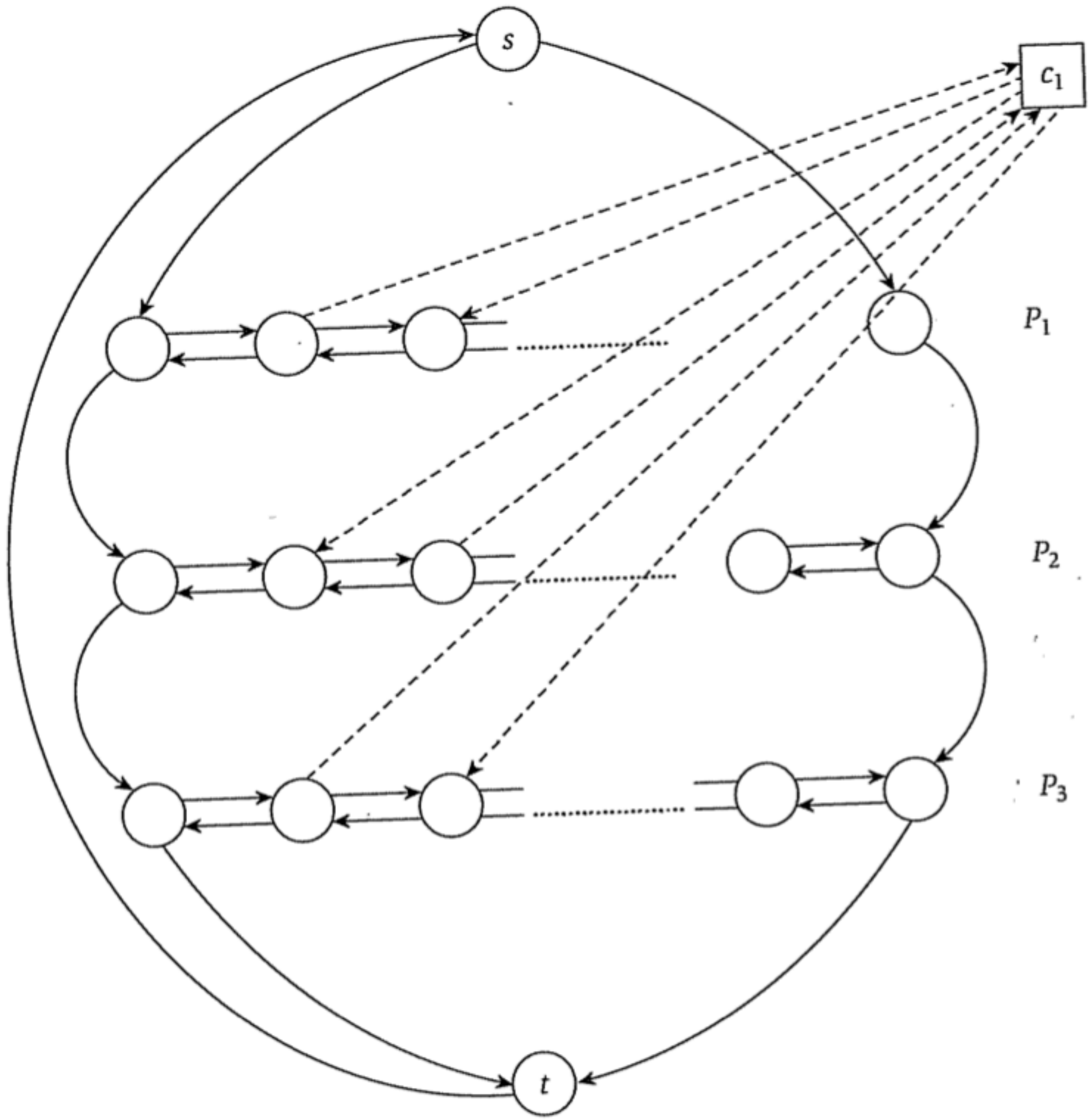


3SAT $\leq_P$ Directed Hamiltonian Cycle

- Any cycle through this graph must hit every path P_i and travel through it
- For each path, have a choice: travel from right to left or from left to right
- 2^n different cycles
- Identify with variable assignments: if P_i goes left-to-right, x_i true, false otherwise

3SAT $\leq_P$ Directed Hamiltonian Cycle

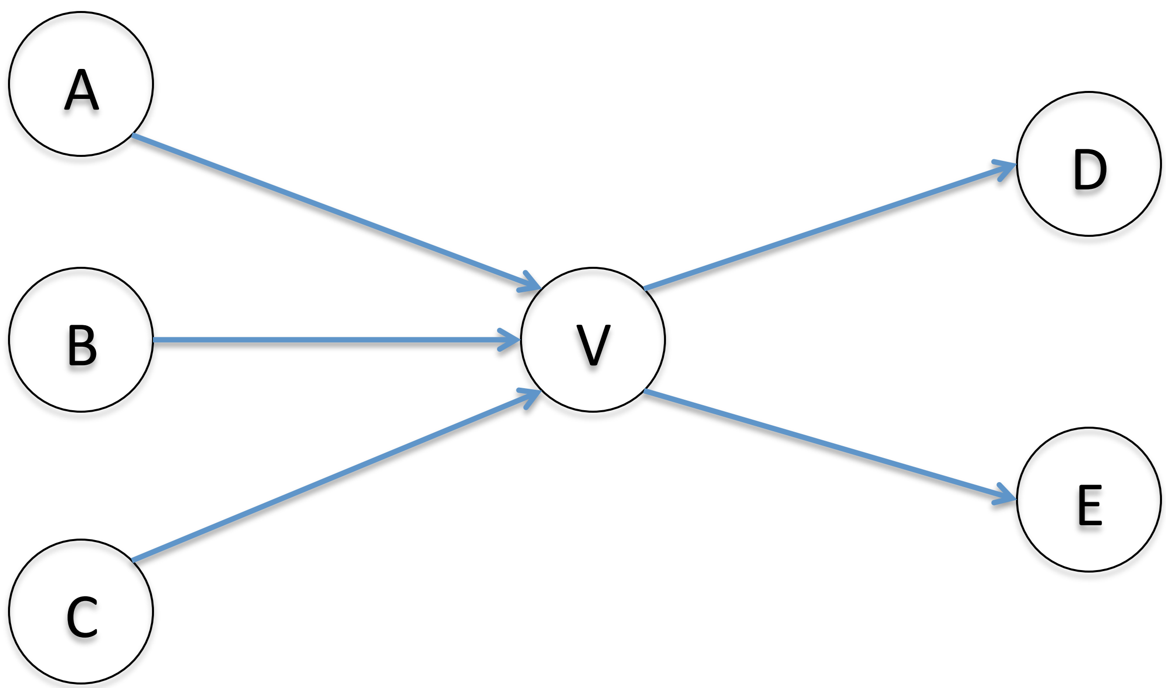
- How to enforce clauses?
- Example: $(x_1 \vee \overline{x_2} \vee x_3)$
  - Either path P_1 goes left to right, or P_2 goes right to left, or P_3 goes left to right
  - How do we enforce this?


s
c_1
P_1
P_2
P_3
t

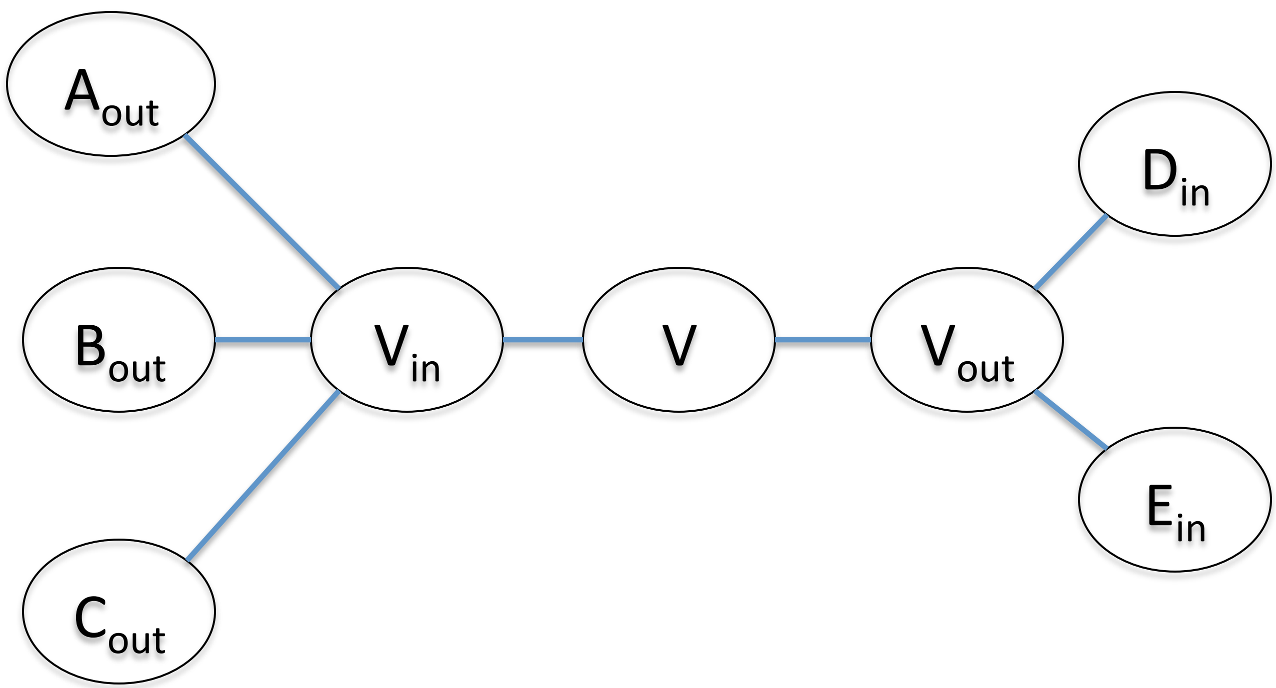
3SAT $\leq_P$ Directed Hamiltonian Cycle

- If our 3SAT instance has a satisfying assignment, there will be a Hamiltonian path in G where each path P_i is traveled in the direction indicated by x_i
- If G has a Hamiltonian path, then we can get a satisfying assignment by setting x_i to the direction P_i was traversed
- Note: delete edge (t,s), and we can show that Directed Hamiltonian Path is also NP-Complete

Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle



Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle



Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle

- If G has a directed Hamiltonian Cycle, G has an undirected Hamiltonian Cycle
  - If we follow edge (u,v) in G, we follow edges (u,u_{out}), (u_{out},v_{in}), (v_{in},v)

Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle

- If G' has an undirected Hamiltonian Cycle, pick some node v, and orient the cycle so that the path goes from v_{in} to v to v_{out}
- Next node after v_{out} has to be w_{in} for some other node w
- Must visit w next (otherwise, we will never be able to visit w in the future)
- Then must visit w_{out}

Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle

- Thus, we always visit nodes in order u_{in}, u, u_{out}.
- To construct Hamiltonian path in G: If we follow some edge (u_{out}, u'_{in}) in G', follow the edge (u,u') in G
- Therefore, G has a directed Hamiltonian cycle if and only if G' has an undirected Hamiltonian cycle
- Note: reduction also words for Directed Hamiltonian Path $\leq_P$ Hamiltonian Path

Hamiltonian Cycle $\leq_P$ TSP

- Given a graph G = (V,E), construct a new weighted complete graph G' on V as follows:
  - If e is in E, the w(e) = 1
  - If e is not in E, then w(e) = 1 + x
    - Determine x later

Hamiltonian Cycle $\leq_P$ TSP

- If G has a Hamiltonian cycle, then that cycle is a tour of length |V| in G'
- Similarly, if G' has a tour of length |V|, all edges must be of weight 1, they must be present in G. Therefore, we have a Hamiltonian cycle in G

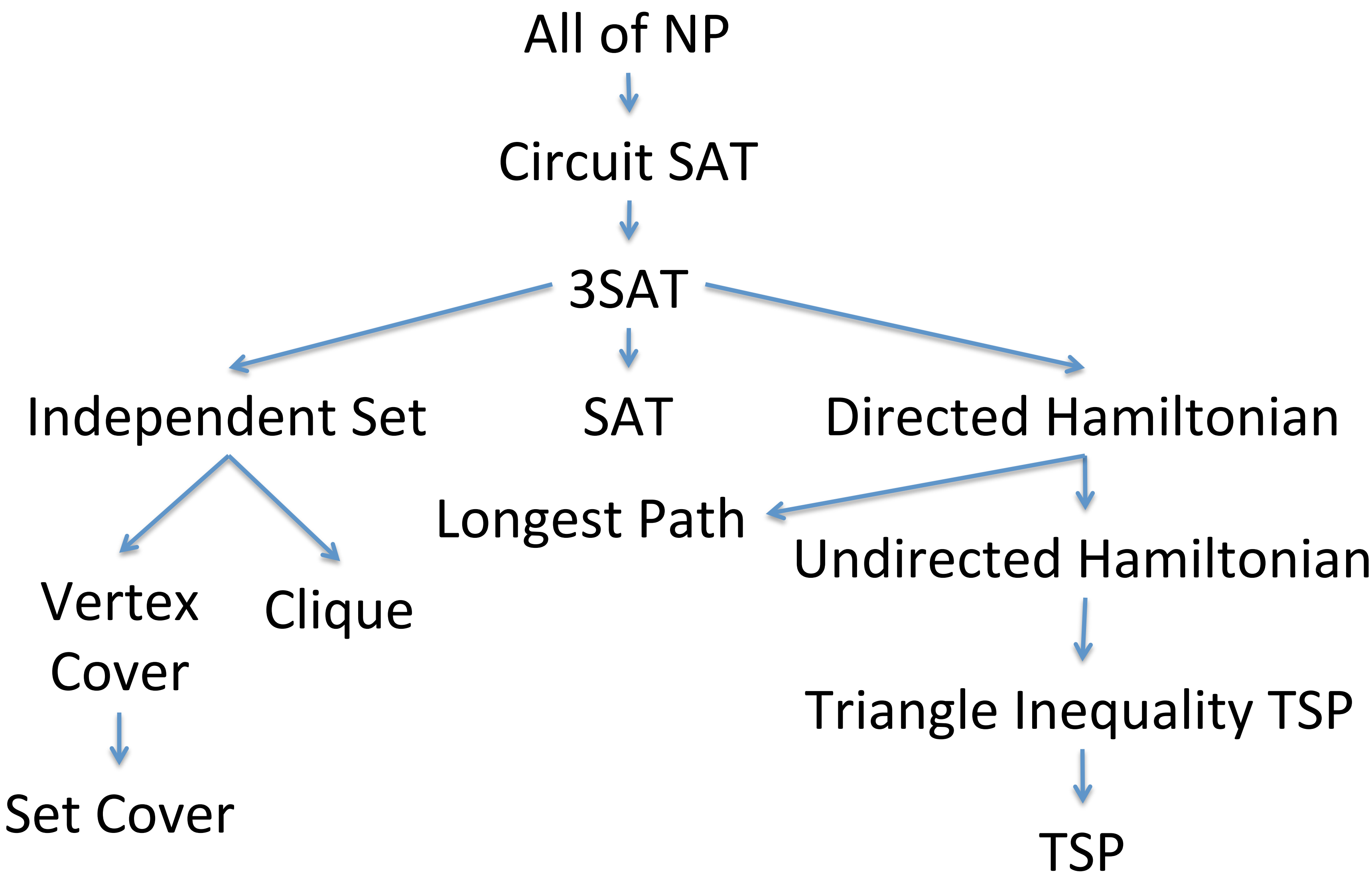
$x = 1$

- All edges have weight 1 or 2
- Triangle inequality satisfied:
  - $w(u,v) + w(v,w) \geq w(u,w)$
- Therefore, TSP where edge weights satisfy triangle inequality is still NP-Complete

Large x

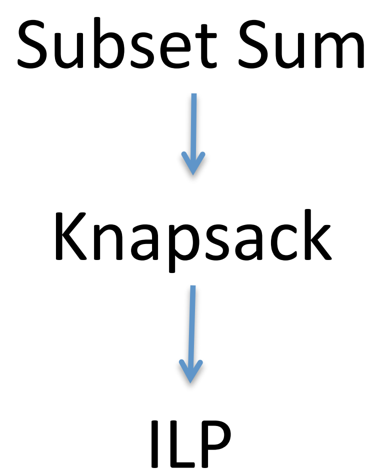
- Notice in general that either there is a tour of length n, or the lightest tour has length at least n + x
- What if we could approximate TSP to within a factor of p(n) for some polynomial n
- Set x = n p(n)
- If there is a Hamiltonian cycle, then there is a cycle with cost n → obtain cycle with cost ≤ n p(n) → must have optimal cycle

So Far


All of NP
Circuit SAT
3SAT
Independent Set
SAT
Directed Hamiltonian
Longest Path
Undirected Hamiltonian
Vertex Cover
Clique
Triangle Inequality TSP
Set Cover
TSP

Problems We Haven't Proved

- 3D Matching
- ILP
- ZOE
- Subset Sum
- Knapsack
- Scheduling



Coping With NP-Completeness

Coping With NP-Completeness

- Suppose we need to solve some *optimaztion* problem
- We realize that the decision version is NP-Complete
- We probably cannot solve all instances exactly
- Approximate?

Approximation Algorithms

- Recall our greedy algorithm for set cover
  - Repeatedly pick the set that contains the most number of uncovered elements
  - If there is a set cover of size k, greedy returns a set cover of size at most k ln n
- Approximation ratio: ratio of obtained solution to optimal solution
  - Set cover: ln n

Vertex Cover

- Special case of set cover, so also a ln n ratio
- Can we do better?

Vertex Cover

- Vertex Cover: set of nodes touching every edge
- Matching: Set of edges that don’t share endpoints
- Any vertex cover is at least as large as any matching

Vertex Cover

- Maximal Matching: a matching such that we can't add any more edges
  - Easy to compute: keep adding an edge until we can't any more
- Algorithm: compute maximal matching M, and return the set S containing both endpoints of every edge of M
  - $|S| = 2|M|$

Vertex Cover

- Is S a vertex cover?
  - Say some edge e has both endpoints not in S
  - Then we can add e to the matching M, meaning M wasn't maximal
- Let optimal vertex cover have size Opt
- $|S| = 2|M| \leq 2$ Opt
- Approximation ratio = 2

Knapsack

- For any constant ε, possible to devise a polynomial time algorithm with an approximation ratio of 1+ε
- Called a polynomial time approximation scheme (PTAS)

Travelling Salesman

- We briefly saw that Triangle Inequality TSP can be approximated within a factor of 2
- General TSP: cannot be approximated to within any polynomial unless P = NP
  – Reduction from Hamiltonian

Not all NP-Complete Problems are Created Equal

- Polynomial time approximation scheme
  - Subset sum, Euclidean TSP
- Constant factor approximation
  - Vertex cover, Triangle Inequality TSP
- Some polynomial factor
  - Set cover
- No Approximation
  - TSP