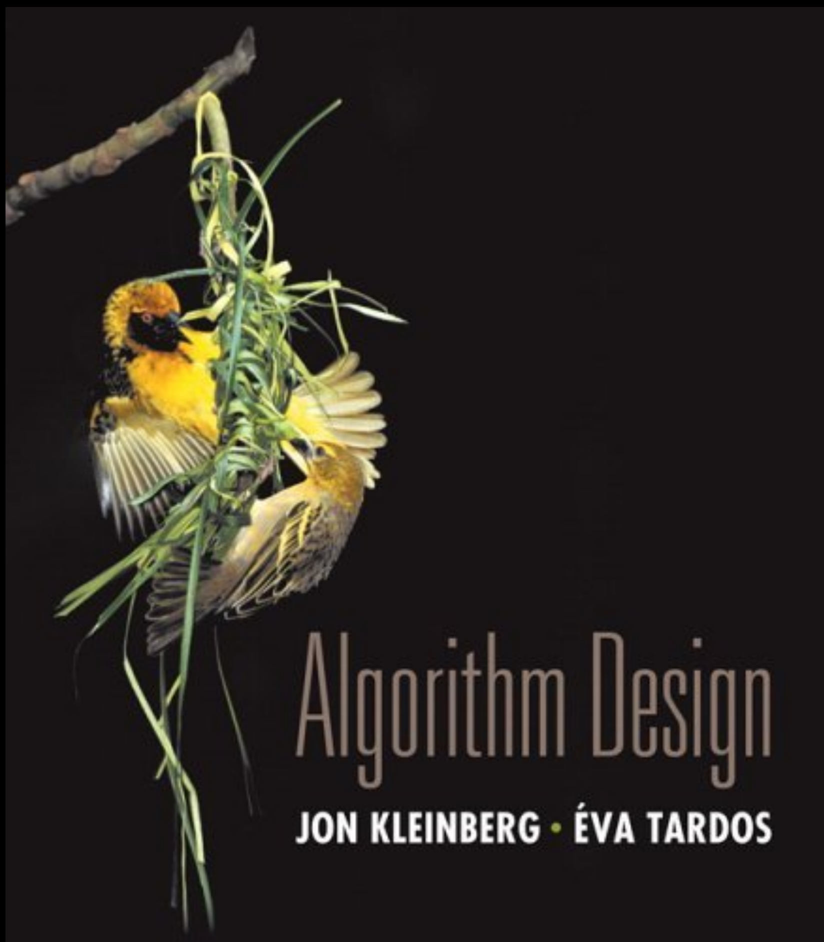


Chapter 10

Extending the Limits of Tractability

Slides by Kevin Wayne.

Coping With NP-Completeness

Q. Suppose I need to solve an NP-complete problem. What should I do?
A. Theory says you're unlikely to find poly-time algorithm.

Must sacrifice one of three desired features.

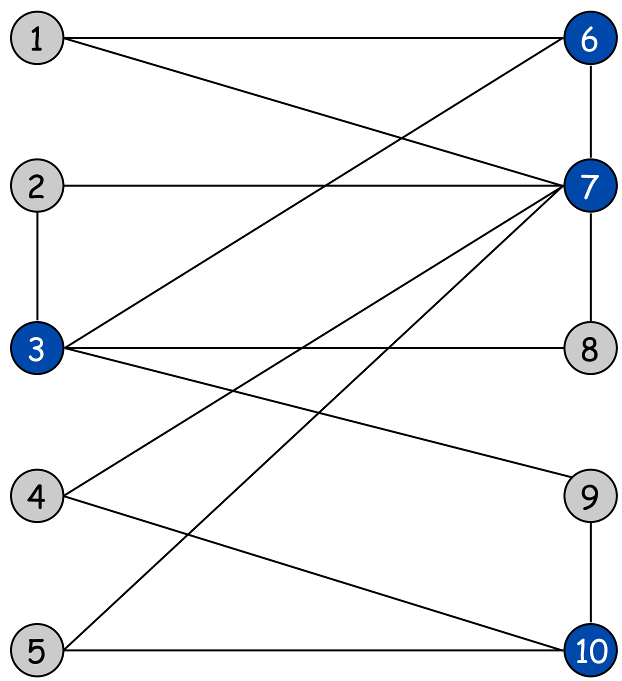
- Solve problem to optimality.
- Solve problem in polynomial time.
- Solve arbitrary instances of the problem.

This lecture. Solve some special cases of NP-complete problems that arise in practice.

10.1 Finding Small Vertex Covers

Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge (u, v) either $u \in S$, or $v \in S$, or both.



$k = 4$
$S = \{ 3, 6, 7, 10 \}$

Finding Small Vertex Covers

Q. What if k is small?

Brute force. $O(k\, n^{k+1})$.

- Try all $C(n, k) = O(n^k)$ subsets of size k.
- Takes $O(k\, n)$ time to check whether a subset is a vertex cover.

Goal. Limit exponential dependency on k, e.g., to $O(2^k\, k\, n)$.

Ex. $n = 1{,}000$, $k = 10$.
Brute. $k\, n^{k+1} = 10^{34} \Rightarrow$ infeasible.
Better. $2^k\, k\, n = 10^7 \Rightarrow$ feasible.

Remark. If k is a constant, algorithm is poly-time; if k is a small constant, then it's also practical.

Finding Small Vertex Covers

Claim. Let u-v be an edge of G. G has a vertex cover of size $\leq k$ iff at least one of $G - \{u\}$ and $G - \{v\}$ has a vertex cover of size $\leq k-1$.

delete v and all incident edges

Pf. $\Rightarrow$

- Suppose G has a vertex cover S of size $\leq k$.
- S contains either u or v (or both). Assume it contains u.
- $S - \{u\}$ is a vertex cover of $G - \{u\}$.

Pf. $\Leftarrow$

- Suppose S is a vertex cover of $G - \{u\}$ of size $\leq k-1$.
- Then $S \cup \{u\}$ is a vertex cover of G. ▪

Claim. If G has a vertex cover of size k, it has $\leq k(n-1)$ edges.

Pf. Each vertex covers at most n-1 edges. ▪

Finding Small Vertex Covers: Algorithm

Claim. The following algorithm determines if G has a vertex cover of size $\leq k$ in $O(2^k\, kn)$ time.

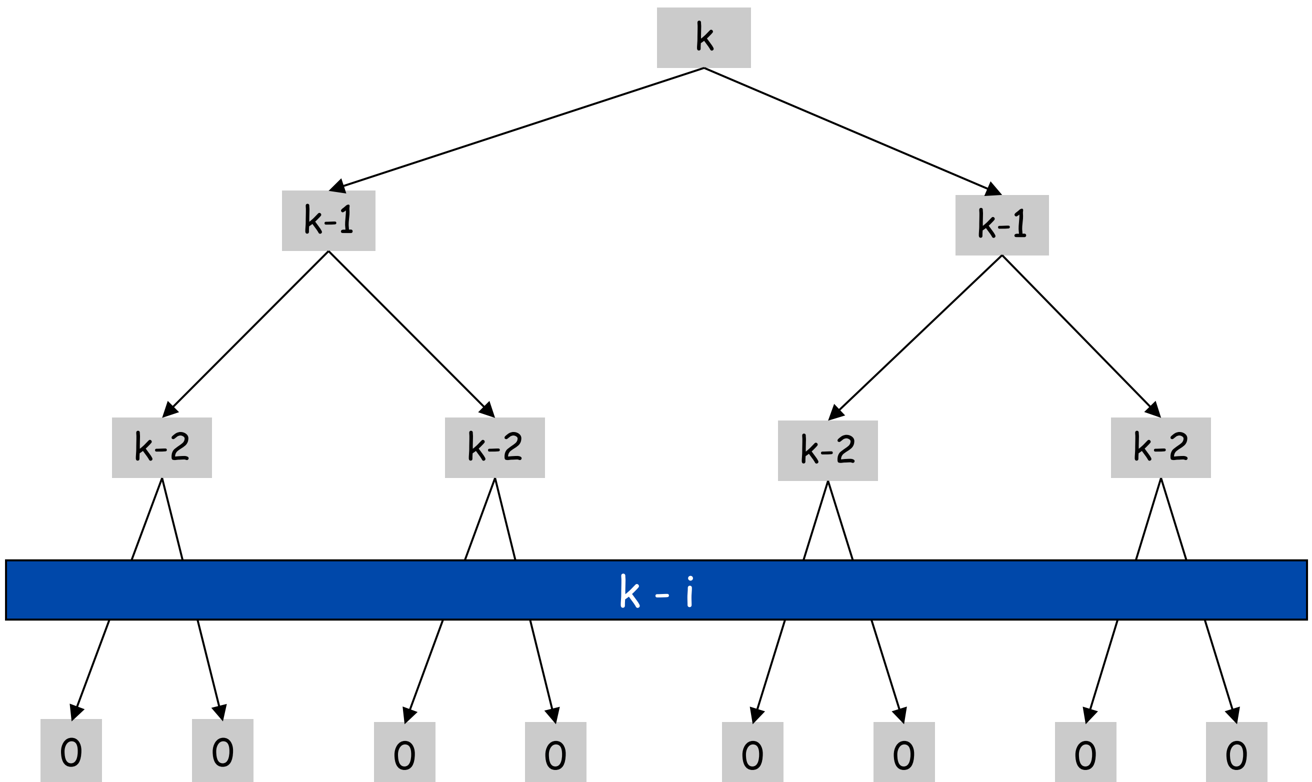
```
boolean Vertex-Cover(G, k) {
   if (G contains no edges)    return true
   if (G contains ≥ kn edges) return false

   let (u, v) be any edge of G
   a = Vertex-Cover(G - {u}, k-1)
   b = Vertex-Cover(G - {v}, k-1)
   return a or b
}
```

Pf.

- Correctness follows previous two claims.
- There are $\leq 2^{k+1}$ nodes in the recursion tree; each invocation takes $O(kn)$ time. ▪

Finding Small Vertex Covers: Recursion Tree

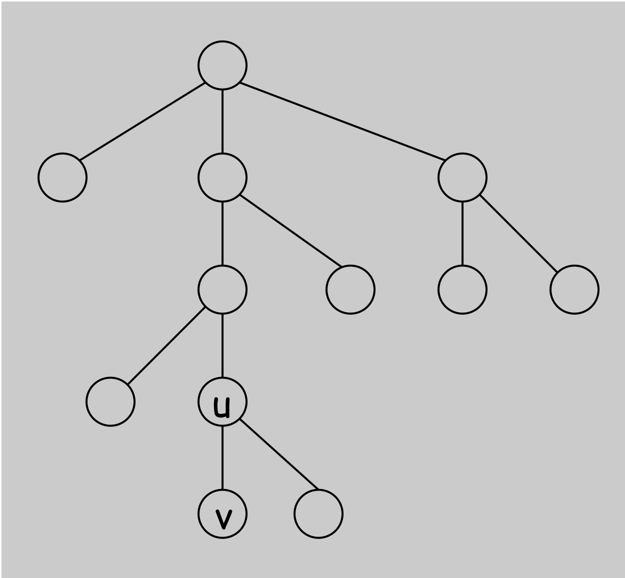
$$T(n, k) \le \begin{cases} cn & \text{if } k = 1 \\ 2T(n, k-1) + ckn & \text{if } k > 1 \end{cases} \quad \Rightarrow \quad T(n, k) \le 2^k c k n$$



10.2 Solving NP-Hard Problems on Trees

Independent Set on Trees

Independent set on trees. Given a tree, find a maximum cardinality subset of nodes such that no two share an edge.



Fact. A tree on at least two nodes has at least two leaf nodes.

degree = 1

Key observation. If v is a leaf, there exists a maximum size independent set containing v.

Pf. (exchange argument)

- Consider a max cardinality independent set S.
- If $v \in S$, we're done.
- If $u \notin S$ and $v \notin S$, then $S \cup \{v\}$ is independent $\Rightarrow$ S not maximum.
- IF $u \in S$ and $v \notin S$, then $S \cup \{v\} - \{u\}$ is independent. ▪

Independent Set on Trees: Greedy Algorithm

Theorem. The following greedy algorithm finds a maximum cardinality independent set in forests (and hence trees).

```
Independent-Set-In-A-Forest(F) {
   S ← φ
   while (F has at least one edge) {
      Let e = (u, v) be an edge such that v is a leaf
      Add v to S
      Delete from F nodes u and v, and all edges
         incident to them.
   }
   return S
}
```

Pf. Correctness follows from the previous key observation. ▪

Remark. Can implement in $O(n)$ time by considering nodes in postorder.

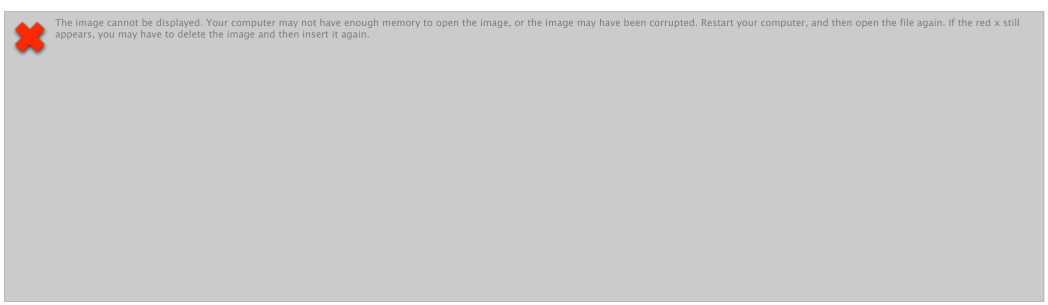
Weighted Independent Set on Trees

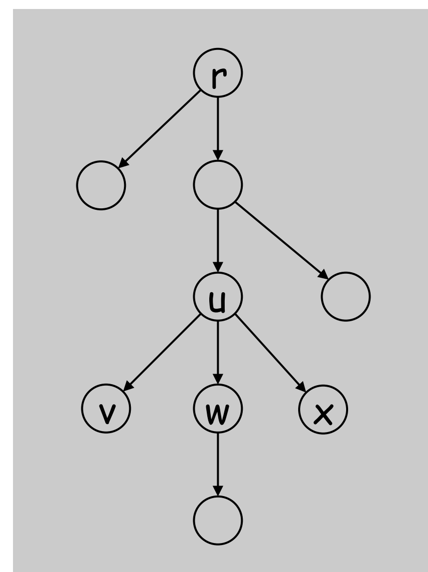
Weighted independent set on trees. Given a tree and node weights $w_v > 0$, find an independent set S that maximizes $\Sigma_{v \in S} w_v$.

Observation. If (u, v) is an edge such that v is a leaf node, then either OPT includes u, or it includes all leaf nodes incident to u.

Dynamic programming solution. Root tree at some node, say r.

- $OPT_{in}(u)$ = max weight independent set rooted at u, containing u.
- $OPT_{out}(u)$ = max weight independent set rooted at u, not containing u.





children(u) = { v, w, x }

Independent Set on Trees: Greedy Algorithm

Theorem. The dynamic programming algorithm find a maximum weighted independent set in trees in O(n) time.

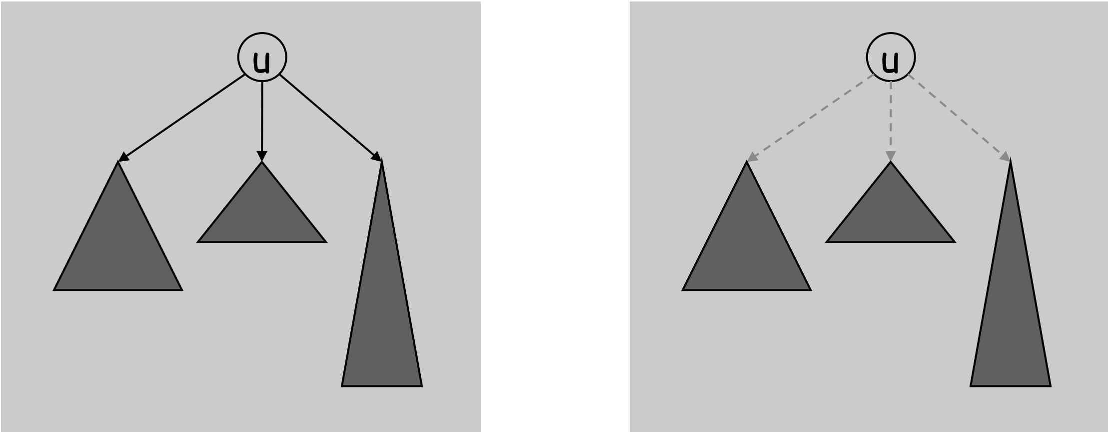
```
Weighted-Independent-Set-In-A-Tree(T) {
   Root the tree at a node r
   foreach (node u of T in postorder) {
      if (u is a leaf) {
         M_in [u] = w_u
         M_out[u] = 0
      }
      else {
         M_in [u] = Σ_{v∈children(u)} M_out[v]  +  w_v
         M_out[u] = Σ_{v∈children(u)} max(M_out[v], M_in[v])
      }
   }
   return max(M_in[r], M_out[r])
}
```

ensures a node is visited after all its children

Pf. Takes O(n) time since we visit nodes in postorder and examine each edge exactly once. ▪

Context

Independent set on trees. This structured special case is tractable because we can find a node that breaks the communication among the subproblems in different subtrees.



see Chapter 10.4, but proceed with caution

Graphs of bounded tree width. Elegant generalization of trees that:

- Captures a rich class of graphs that arise in practice.
- Enables decomposition into independent pieces.

10.3 Circular Arc Coloring