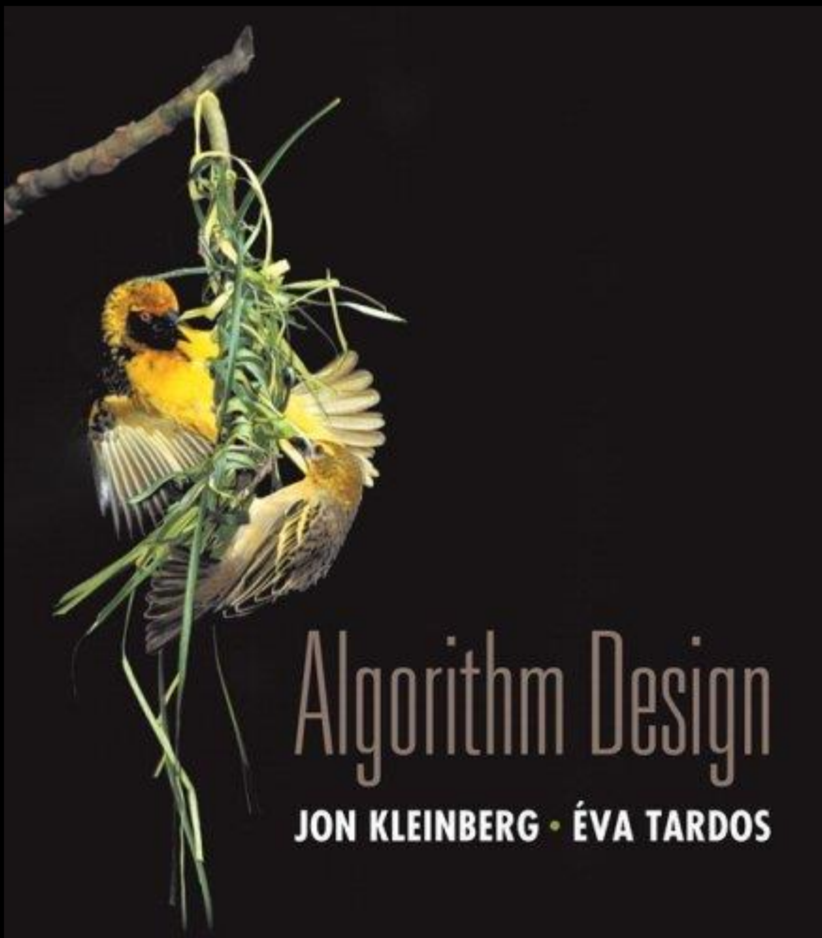


Chapter 10

Extending the Limits of Tractability

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.

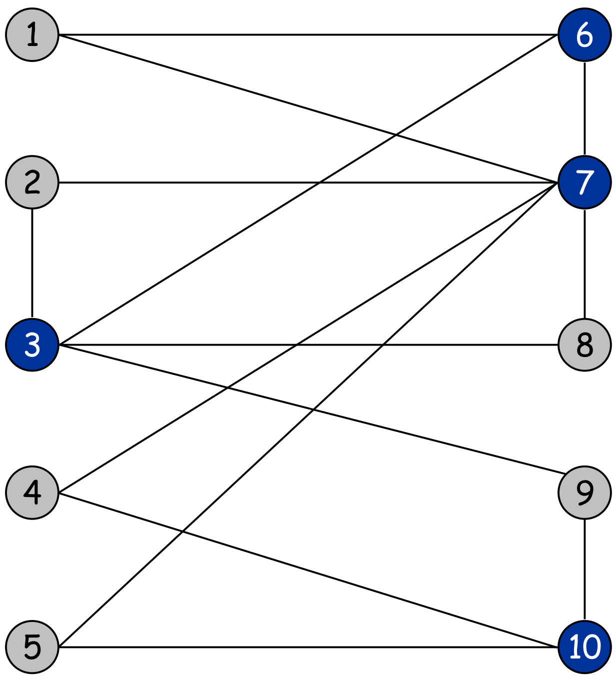
Must sacrifice one of three desired features.
- Solve problem to optimality.
- Solve problem in polynomial time.
- Solve arbitrary instances of the problem.

This lecture. Solve some special cases of NP-complete problems.

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 a small constant?

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.

Ex. $n = 1{,}000$, $k = 10$.

- $k\,n^{k+1} = 10^{34} \Rightarrow$ polynomial, but infeasible.

Finding Small Vertex Covers

Q. What if k is a small constant?

Goal. Move k out of the exponent on n, e.g., $O(2^k\, k\, n)$.

Ex. $n = 1{,}000$, $k = 10$.

- $2^k\, k\, n = 10^7 \;\Rightarrow$ polynomial and feasible.

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 > k(n-1) 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 (can be made even less). ▪

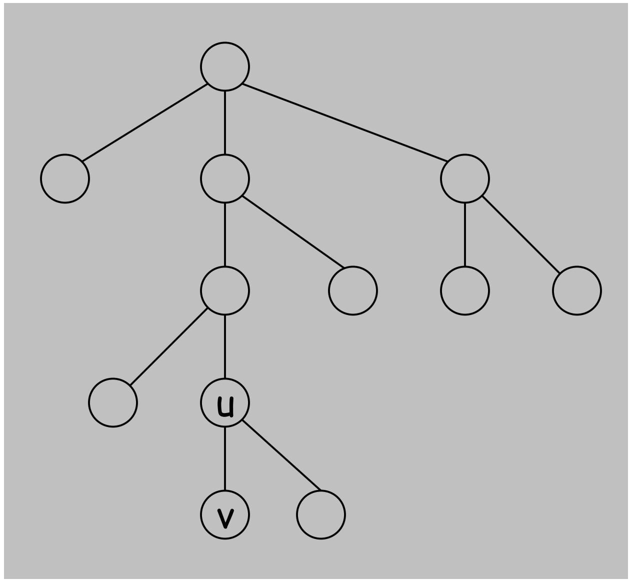
10.2 Solving NP-Hard Problems on Trees

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 of 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.
   }
   Add the remaining nodes to S
   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

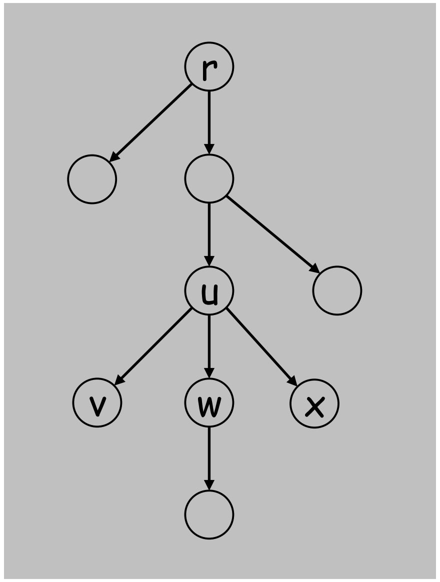
Weighted independent set on trees. Given a tree and node weights $w_v > 0$, find an independent set S that maximizes $\Sigma_{v \in S} w_v$.

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.

$$OPT_{in}(u) = w_u + \sum_{v \in \text{children}(u)} OPT_{out}(v)$$

$$OPT_{out}(u) = \sum_{v \in \text{children}(u)} \max\{OPT_{in}(v),\ OPT_{out}(v)\}$$



Independent Set on Trees: Dynamic Programming Algorithm

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

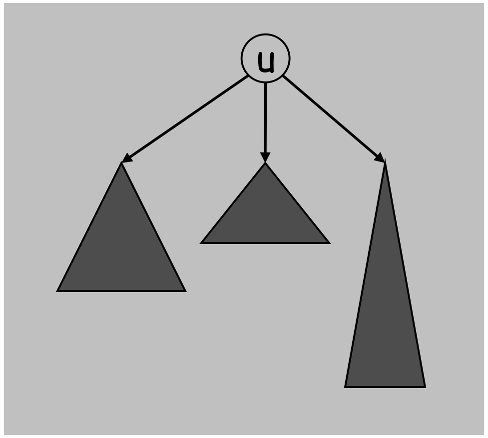
```
Weighted-Independent-Set-In-A-Tree(T) {
   Root the tree at a node r
   foreach (node u of T in postorder) {
      if (u is a leaf) {
         M_in [u] = w_u
         M_out[u] = 0
      }
      else {
         M_in [u] = Σ_{v∈children(u)} M_out[v]  +  w_v
         M_out[u] = Σ_{v∈children(u)} max(M_out[v],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

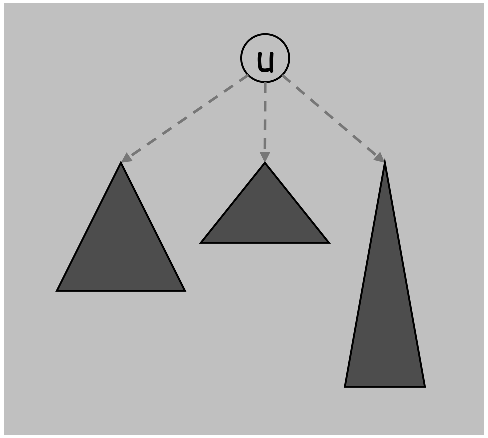
ensures a node is visited after all its children

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.





What if the graph is not a tree?

- Tree decomposition of graphs (Ch 10.4-5)

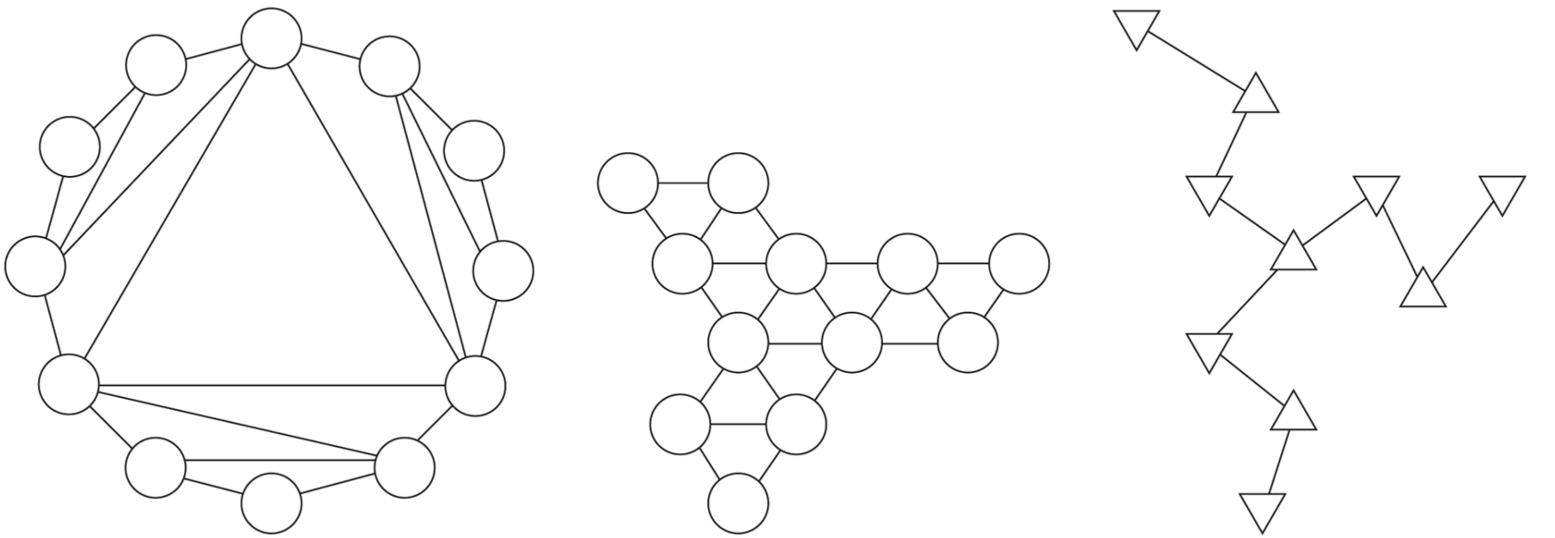
*10.4 Tree Decompositions of Graphs

(in a nutshell)

Graphs of Bounded Treewidth

Intuition

- Graphs that can be ...
  - represented by a "tree-like" structure
  - decomposed into disconnected pieces by removing a small number of nodes (allowing the use of div&con or dynamic programming)



(a) (b) (c)

Tree Decompositions of Graphs

Definition. A tree decomposition of a graph $G = (V, E)$ is a tree T s.t. each node t in T is associated with a piece $V_t \subseteq V$ which satisfies three properties:

- Node Coverage: Every node of G belongs to at least one piece V_t
- Edge Coverage: For every edge e of G, there is some piece V_t containing both ends of e
- Coherence: Let $t1$, $t2$ and $t3$ be three nodes of T such that $t2$ lies on the path from $t1$ to $t3$. Then, if node v of G belongs to both V_{t1} and V_{t3}, it also belongs to V_{t2}

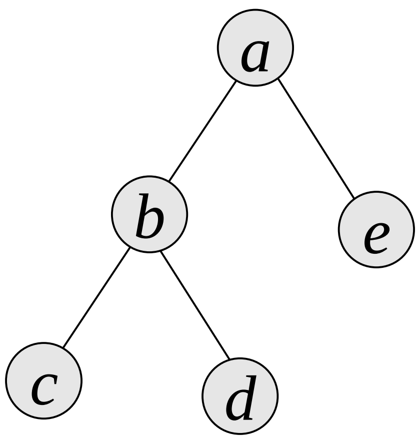
Also known as:

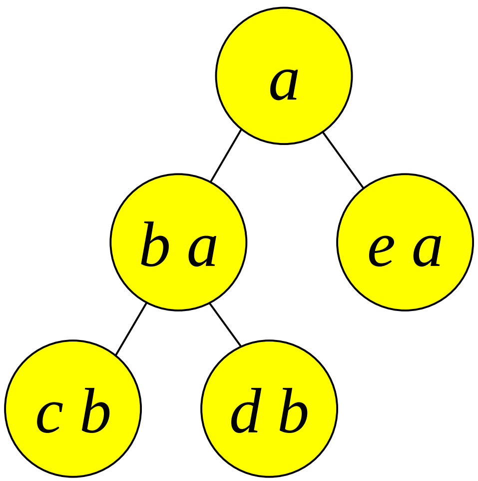
- Junction tree
- Clique tree
- Join tree.

Tree Decompositions of Graphs

Special case: G is a tree

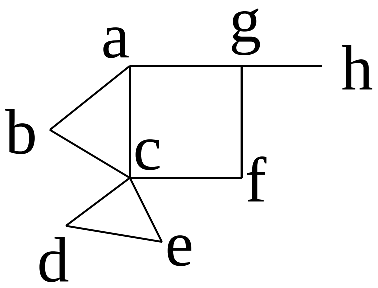
- Choose a root a
- Take X_a = {a}, and for each other node i: X_i = {i, parent(i)}

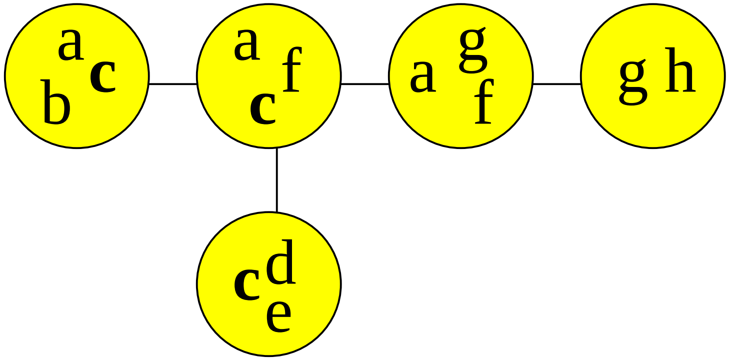




Tree Decompositions of Graphs

General case





Properties of Tree Decomposition

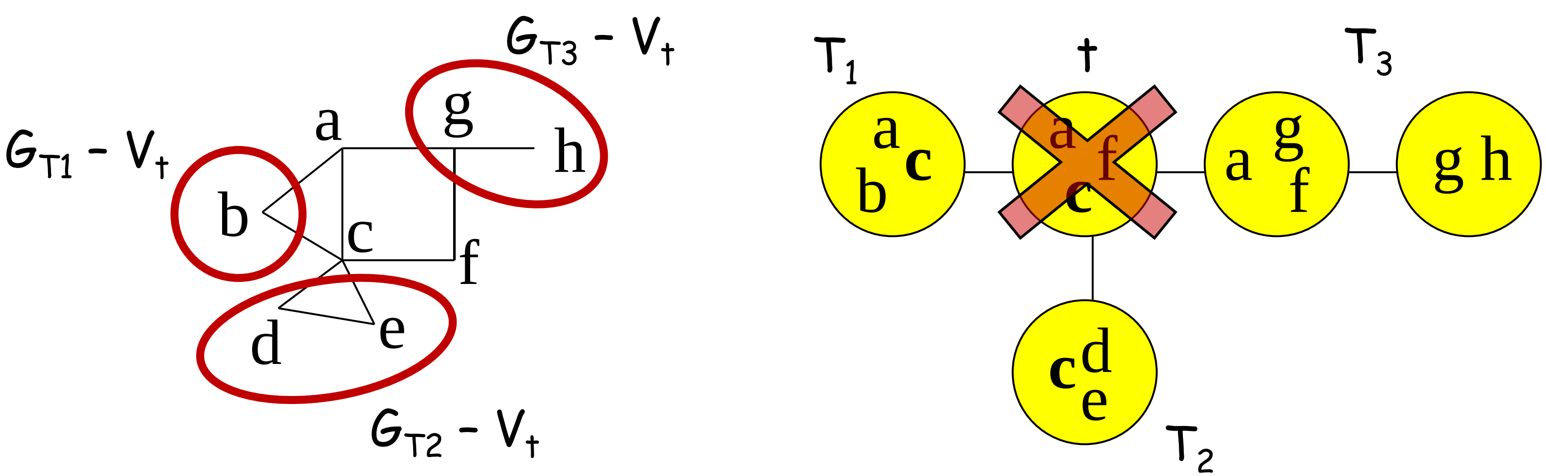
Properties of trees

- Deleting a node results in a number of connected components
- Deleting an edge results in two connected components

Properties of Tree Decomposition

Claim. Suppose deleting a node t from a tree decomposition T produces components $T_1, \ldots, T_d$. Then the subgraphs $G_{T1} - V_t$, $G_{T2} - V_t$, … , $G_{Td} - V_t$ have no node in common and no edge between them.

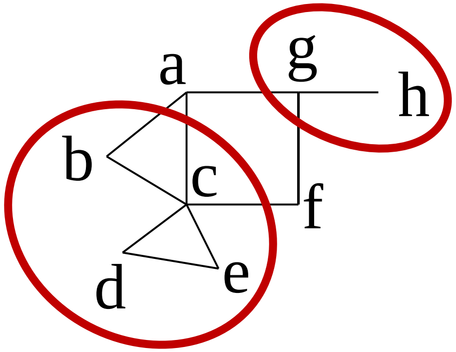
subgraph of G with nodes in $\cup_{t \in Ti} V_t$



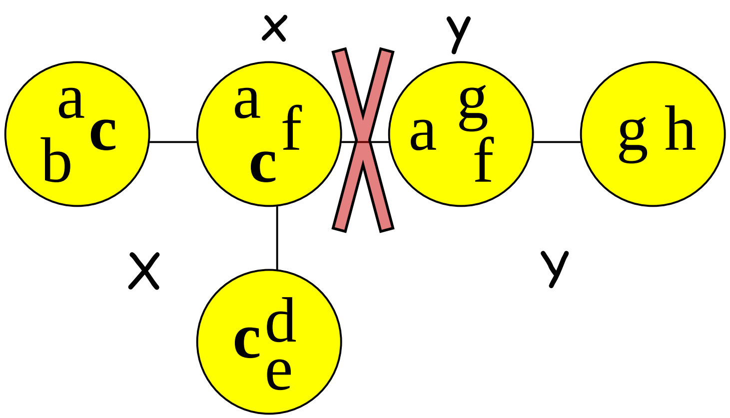
Properties of Tree Decomposition

Claim. Let X and Y be the two components of T after the deletion of the edge (x,y). Then deleting the set $V_x \cap V_y$ from V disconnects G into two subgraphs $G_X - (V_x \cap V_y)$ and $G_Y - (V_x \cap V_y)$ that have no node in common and no edge between them.

$G_Y - (V_x \cap V_y)$



$G_X - (V_x \cap V_y)$



Tree-width

Def. Width of a tree decomposition $(T, \{V_t\})$ is one less than the maximum size of any piece V_t

$$Width(T, \{Vt\}) = \max_{t} |V_t| - 1$$

Def. Tree-width of a graph G is the minimum width of any tree decomposition of G

Comment. "-1" in the definition: so that trees have tree-width 1

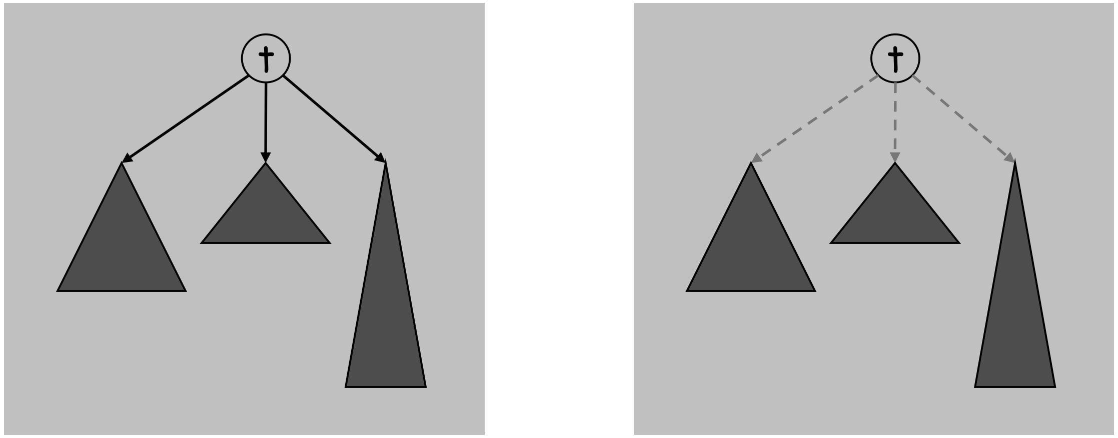
Dynamic Programming over a Tree Decomposition

Example problem:

- Weighted Independent Set on graphs with small tree-width.

Intuition:

- G has a tree decomposition $(T, \{V_t\})$ with tree-width w.
  - Each piece V_t has at most w+1 nodes.
- For each $t \in T$, the optimal independent set specifies a subset of V_t
  - We can enumerate all 2^{w+1} possible subsets of V_t.
  - Once the optimal subset is fixed, the subtrees below t become independent



More applications of tree decomposition

- Graph minor theory (Robertson and Seymour)
- Optimization
- Probabilistic networks
- Expert Systems.
- Telecommunication Network Design.
- VLSI-design.
- Natural Language Processing.
- Compilers.
- Choleski Factorisation.
- maximum independent set.
- Hamiltonian circuit
- vertex coloring problem
- Edge coloring problem
- Graph Isomorphism
- etc.

Constructing a Tree Decomposition (*10.5)

Bad news: Given a graph, it is NP-hard to determine its tree-width.

Good news: Given a graph G of tree-width less than w, we can find a tree decomposition of G of width less than $4w$ in time $O(f(w) \cdot mn)$.

- If the tree-width of G is small, it is fast to find a good tree decomposition.

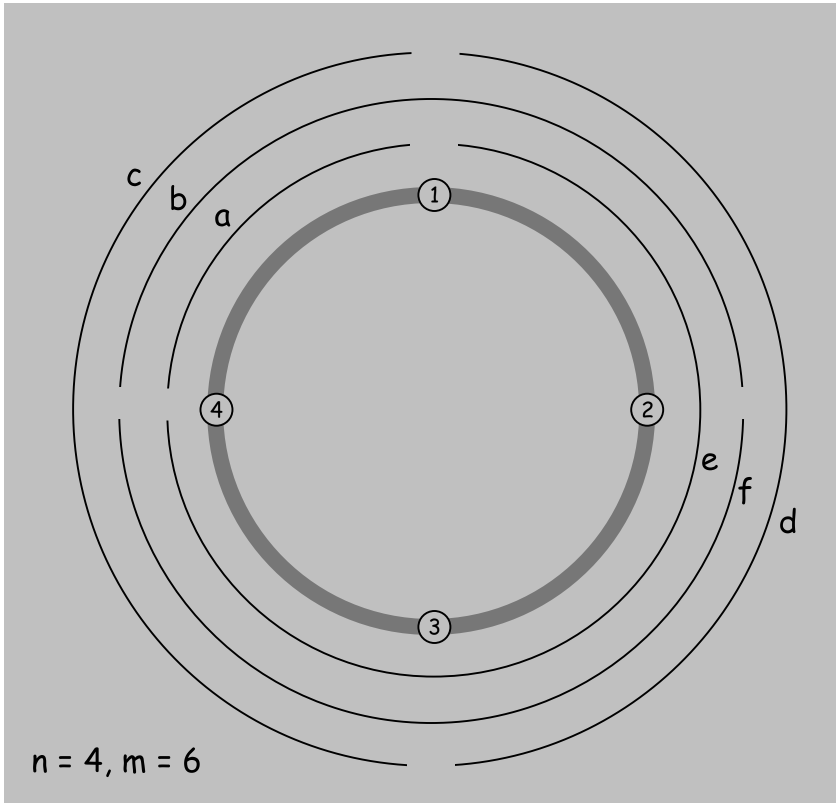
10.3 Circular Arc Coloring

Wavelength-Division Multiplexing

Wavelength-division multiplexing (WDM). Allows m communication streams (arcs) to share a portion of a fiber optic cable, provided they are transmitted using different wavelengths.

Ring topology. Special case is when network is a cycle on n nodes.

Q: Do k colors (wavelengths) suffice?



Wavelength-Division Multiplexing

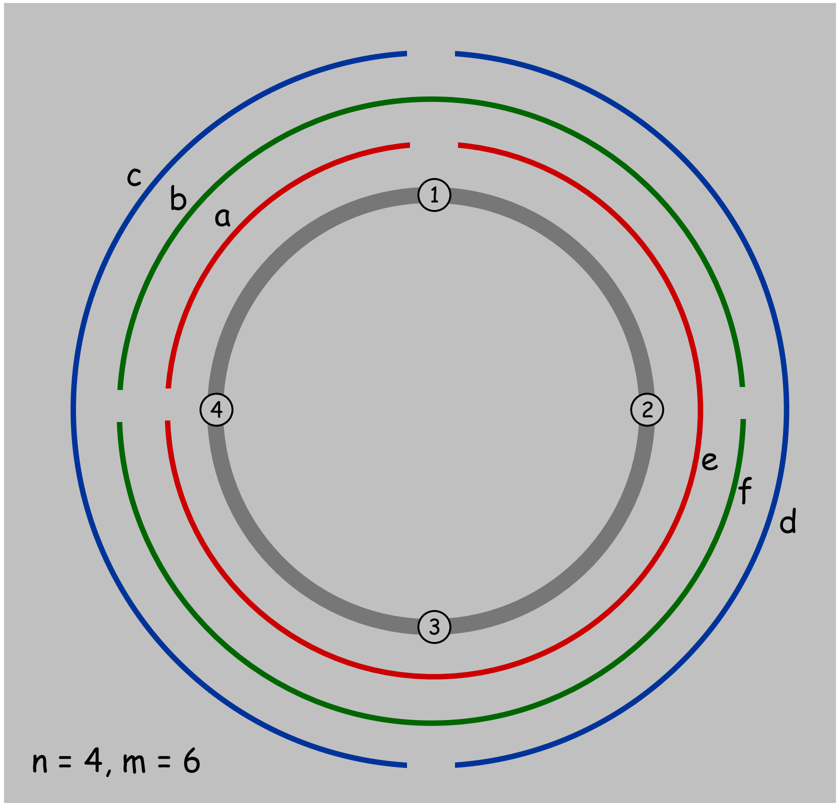
Wavelength-division multiplexing (WDM). Allows m communication streams (arcs) to share a portion of a fiber optic cable, provided they are transmitted using different wavelengths.

Ring topology. Special case is when network is a cycle on n nodes.

Q: Do k colors (wavelengths) suffice?
Bad news. NP-complete, even on rings.

Brute force. Can determine if k colors suffice in $O(k^m)$ time by trying all k-colorings.

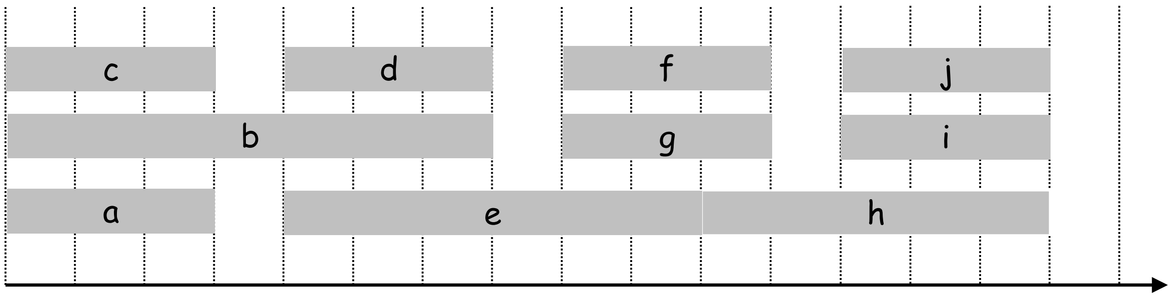
Goal. $O(f(k)) \cdot \text{poly}(m, n)$ on rings.



Review: Interval Coloring

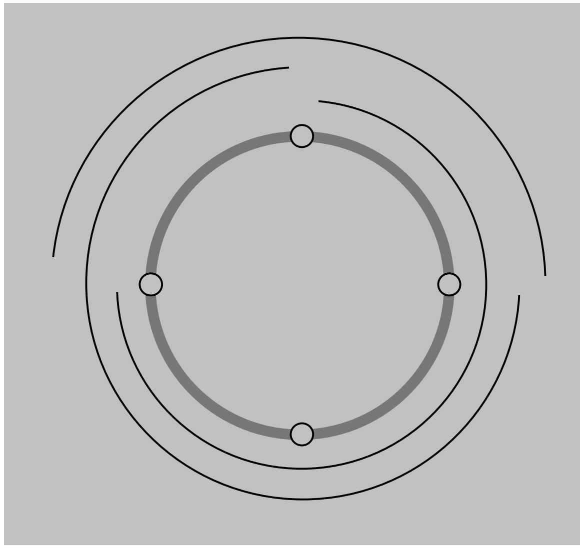
Interval coloring (interval partitioning). Greedy algorithm finds coloring such that number of colors equals depth of schedule.

maximum number of streams at one location



Circular arc coloring.

- Weak duality: number of colors ≥ depth.
- Strong duality does not hold.

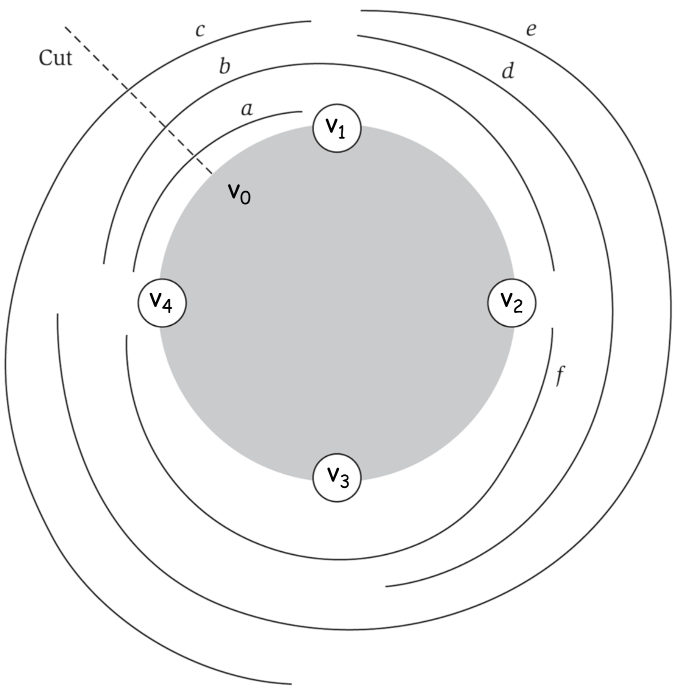


max depth = 2
min colors = 3

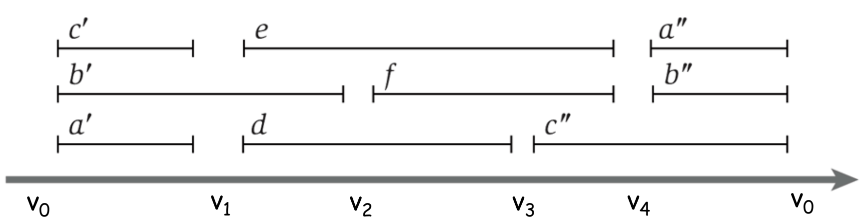
(Almost) Transforming Circular Arc Coloring to Interval Coloring

Circular arc coloring. Given a set of n arcs with depth $d \leq k$, can the arcs be colored with k colors?

Equivalent problem. Cut the network between nodes v_1 and v_n. The arcs can be colored with k colors iff the intervals can be colored with k colors in such a way that "sliced" arcs have the same color.

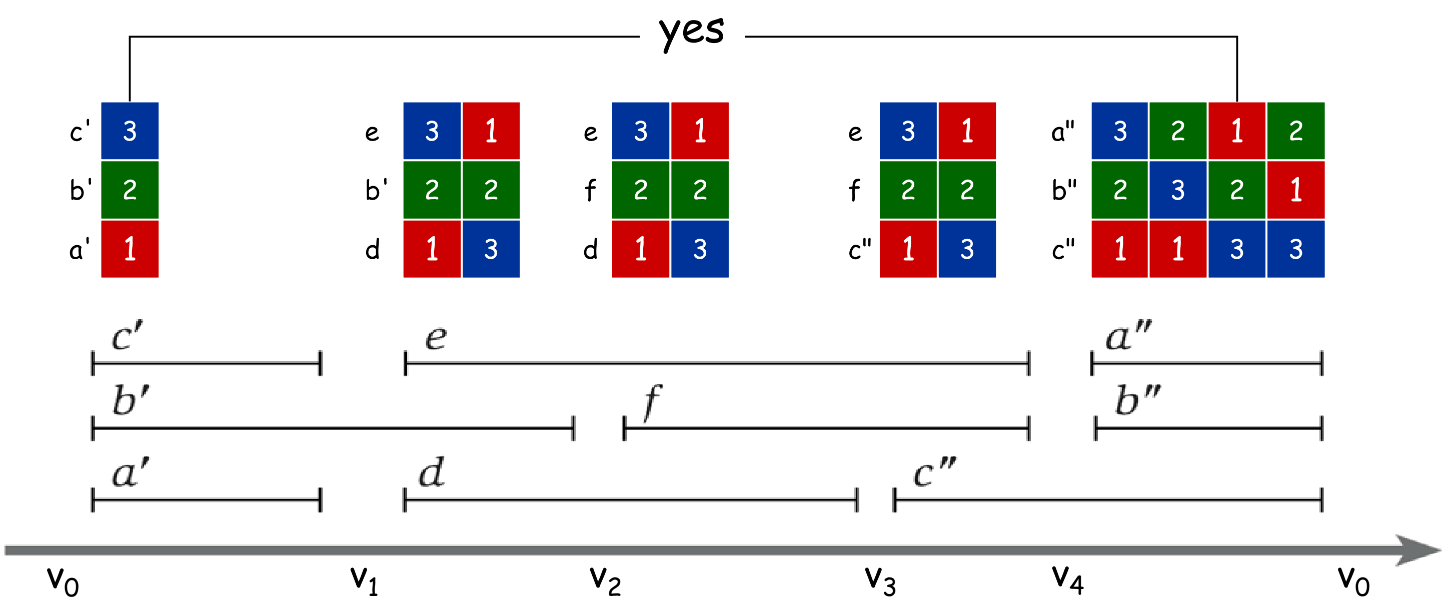


colors of a', b', and c' must correspond to colors of a", b", and c"



Circular Arc Coloring: Algorithm

- Assign distinct color to each interval which begins at cut node v_0.
- At each node v_i, some intervals may finish, and others may begin.
- Enumerate all k-colorings of the intervals through v_i that are consistent with the colorings of the intervals through v_{i-1}.
- The arcs are k-colorable iff some coloring of intervals ending at cut node v_0 is consistent with original coloring of the same intervals.



Circular Arc Coloring: Running Time

Running time. $O(k!^2 \cdot n)$.

- n phases of the algorithm.
- At each node v_i, we enumerate all consistent colorings. There are at most k! colorings before v_i, each leading to at most k! consistent colorings after v_i.

Remark. This algorithm is practical for small values of k even if the number of nodes n or number of arcs m is large.

Chapter Summary

Summary

To solve NP-complete problems, we sacrifice the desired feature of:

- Solve arbitrary instances of the problem.

Example problems

- Finding Small Vertex Covers
- (Weighted) Independent Set on Trees
- Circular Arc Coloring (small number of colors)