CS698F Advanced Data Management

Instructor: Medha Atre

Reachability indexing methods

- 2-hop cover
- Compressed bit-vectors
- Interval labeling.
  - Linear
  - Multi-dimensional

2-hop cover [SODA 2002]

- 2-hop reachability labeling graph G(V,E):
  - Each vertex $v \in V$ $L(v) = (L_{in}(v), L_{out}(v))$
  - $L_{in}(v)$ => all the nodes that can reach v.
  - $L_{out}(v)$ => all the nodes that can be reached from v.
  - What does this remind you of?
- u ---> v *iff* $L_{out}(u) \cap L_{in}(v) \neq \phi$
- But this is very expensive!

2-hop cover [SODA 2002]

- 2-hop cover for (u,v) pair s.t. u ----> v:
  - P_{uv} => set of all the paths between u and v.
  - Hop *($\mathscr{H}$, u)* => $\mathscr{H}$ is a path with an end point as *u*, *u* is the handle of $\mathscr{H}$.
  - 2-hop cover => for every *u,v* ∈ V, s.t. *v* reachable from *u*, there exist 2 hops, ($\mathscr{H}_1$, u), ($\mathscr{H}_2$, v), where $\mathscr{H}_1.\mathscr{H}_2$ is *some* path between *u, v*. (dot is concatenation operator).
  - Objective: Find an optimal (minimum) cover of $\mathscr{H}$, s.t. it covers all the paths in G – NP-hard, by reducing to set-cover problem.
  - Greedy suboptimal solutions – similar to greedy solutions for set-cover problem.

2-hop cover [SODA 2002]

- Set-cover instance of 2-hop problem:
  - Ground set of elements to be covered $T = \{(u,v) \mid P_{uv} \neq \phi\}$.
  - For each vertex w $\in$V and subsets construct $S(C_{in}, w, C_{out})$ as follows
  - $S(C_{in}, w, C_{out}) = \{(u,v) \in T \mid u \in C_{in}, v \in C_{out}, w \in B_{uv}\}$, B_{uv} => set of vertices on paths from P_{uv}.
  - Weight of this set = $|C_{in}| + |C_{out}|$
  - Objective: Find an optimal (minimum) cover of all such set S.

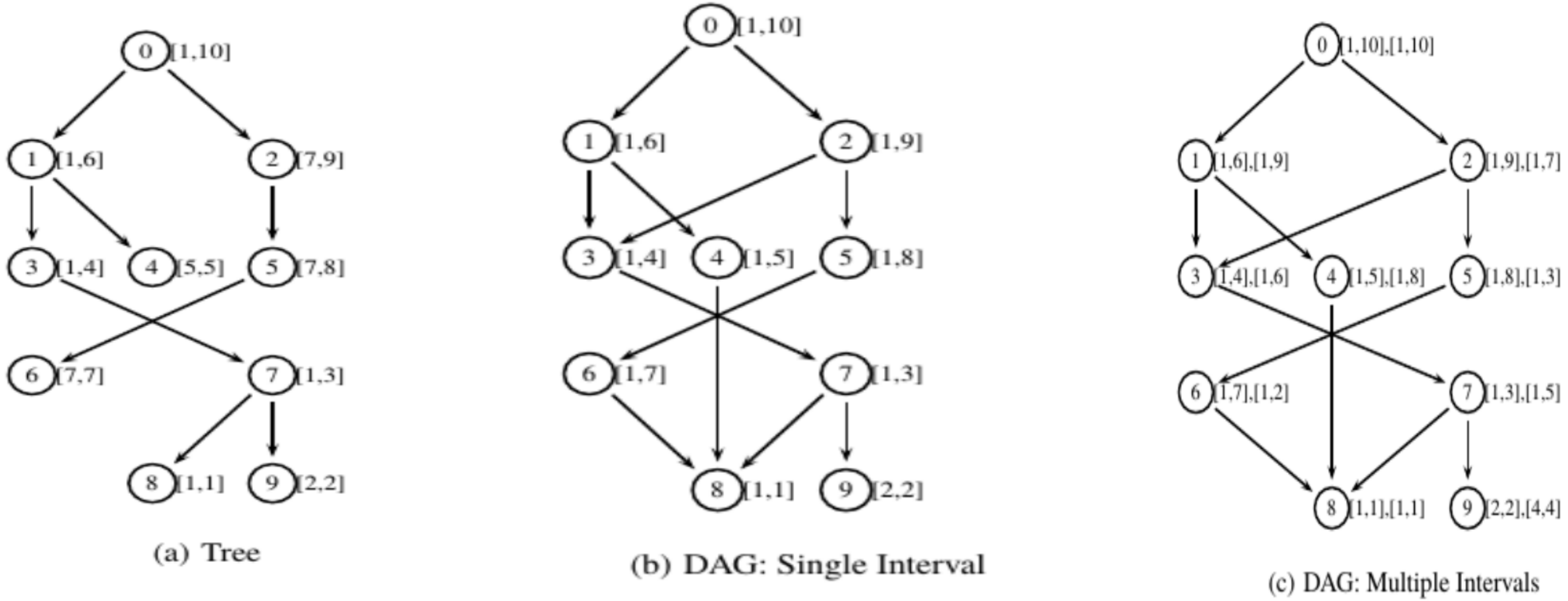
Compressed bit-vectors

- Merge SCCs.
- Sort all the vertices in DAG for *topological ordering*.
- Topological ordering assigns topological order labels to each vertex.
  - Label indicates the *maximum* length of any incoming path to that vertex.
- *Reverse* DFS walk on this DAG considering topological sort – all the neighbors traversed by topological order in a reverse direction.
- Vertices that are adjacent in the topological sort tend to cluster together in the adjacency list – can apply bit-vector compression techniques.
- "A memory efficient reachability data structure through bit-vector compr ession" [SIGMOD2011]
- Index is complete, space $O(V^2)$, but compression saves space.

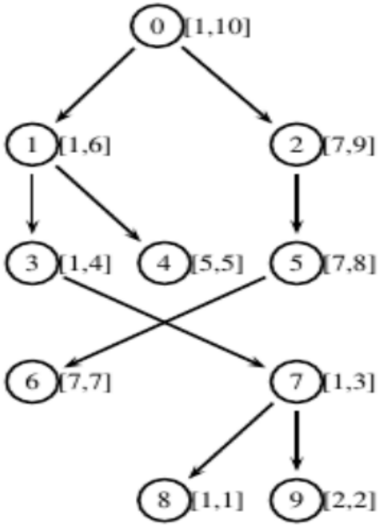
Interval Labeling

- Each graph node has a an interval [x, y] associated with it.
  – This interval is decided after traversing the graph first.
- To decide reachability
  – Node "t" is reachable from "s" *iff* $[x_t, y_t]$ is completely contained in $[x_s, y_s]$
    - e.g., let t's interval be [3, 5] and s's interval be [1, 10], then node 't' is reachable from 's'
    - Does NOT work for DAGs! Why?

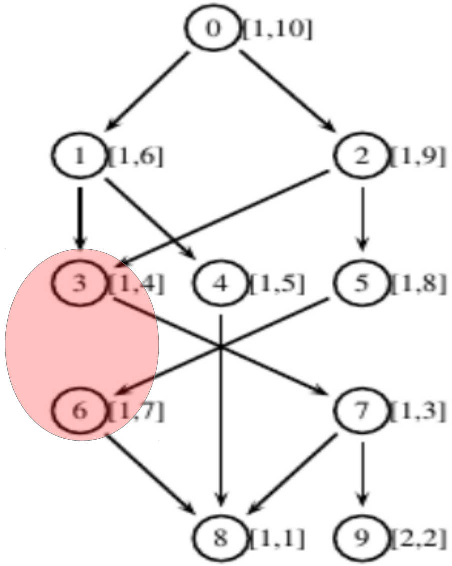
Interval Labeling



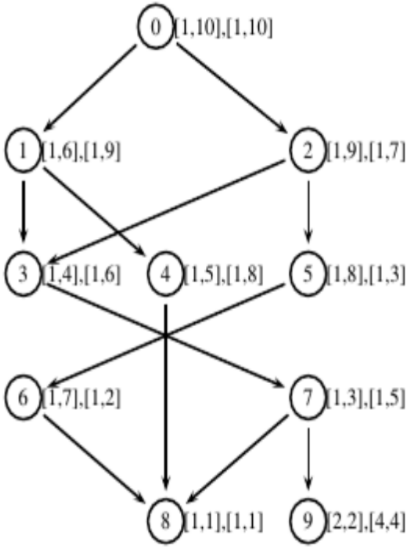
Interval labeling



(a) Tree

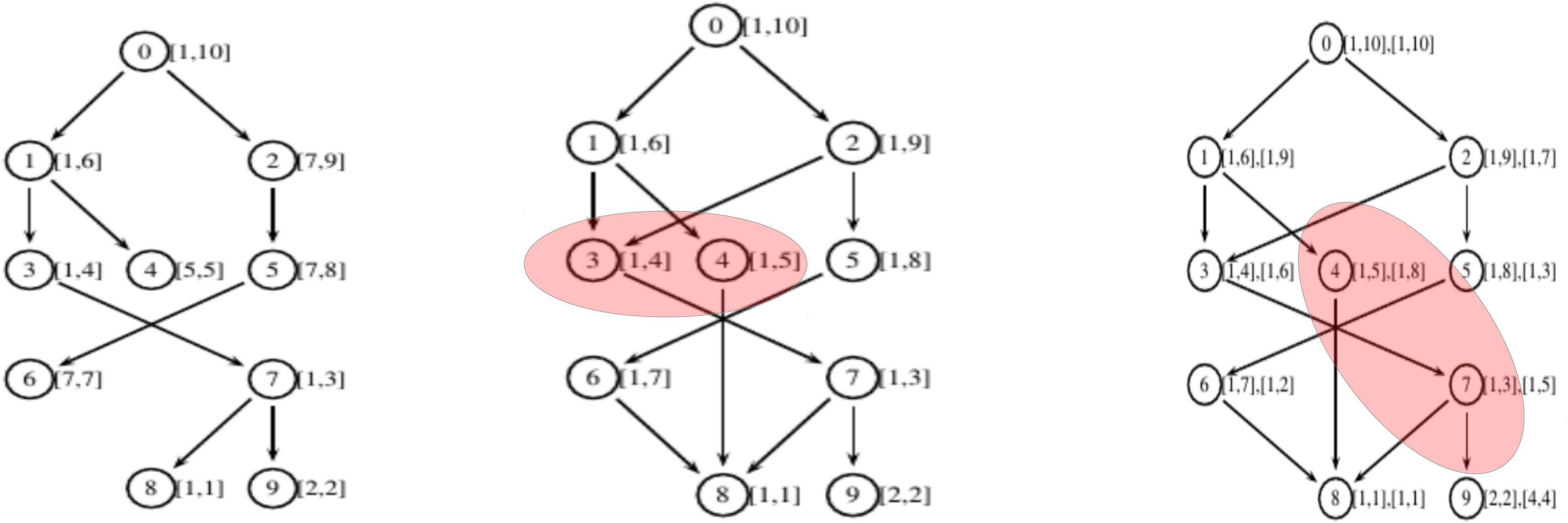


(b) DAG: Single Interval



(c) DAG: Multiple Intervals

Interval labeling



(a) Tree

(b) DAG: Single Interval

(c) DAG: Multiple Intervals

Grail [VLDB 2010]

- Consider *k* different spanning trees over a given DAG.
- For each spanning tree, general 1-dimensional interval labels.
- Combine all *k* labels => <u>*k-dimensional*</u> interval label.
- Can generate false positives => (u, v), v *not* reachable from u, but interv(v) $\in$ interv(u).
- Never generates false negatives => If interv(v) $\notin$ interv(u), then definitely v not reachable from u.
- Why would you use such a scheme, which does not give a correct answer always?