Localized Network Representations

David Peleg

The Weizmann Institute

Traditional graph representations

Store adjacency information in data structure

- Requires much storage

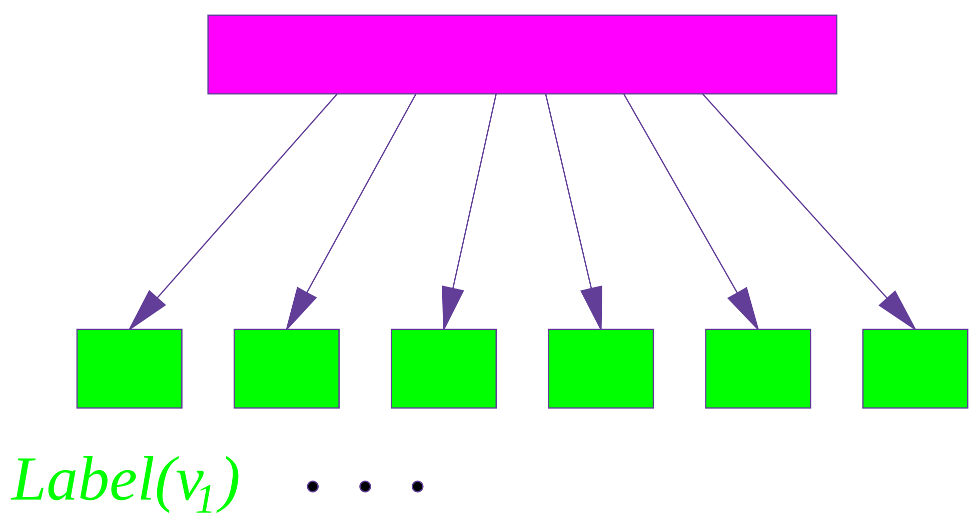
- Node labels contain no information,
  serve only as "pointers" to data structure

	1	2	3	4	5	
1	-					
2	-	-				
3	**1**	**1**	-			
4	-	-	**1**	-		
5	-	**1**	-	-	-	
						⋱

Local graph representations

Idea: Store *local* pieces of information, i.e.:

Associate label $Label(v)$ for each node v, s.t. labels allow inferring information *locally*, without using additional memory



Question:
Can this be done compactly & efficiently?

“Ancient” Example: Hamming adjacency labeling

Goal: Label each v with m-bit label $Label(v)$ s.t. $Label(u)$, $Label(v)$ allow deciding if u and v are adjacent:

u, v adjacent

$$\Updownarrow$$

Hamming_dist$(Label(u), Label(v)) \leq T$

Lemma:

$\forall$ n-node graph

$\exists$ Hamming distance adjacency labeling with

$m = 2n\Delta, \quad T = 4\Delta - 4$

where $\Delta =$ max node degree

[Breuer,Folkman,67]

Question: Can this be done with short labels?

Interest Revived:

Adjacency Labeling

[Kannan,Naor,Rudich,88]

What's in a label?

Adjacency-labeling [for graph family $\mathcal{F}$]:

1. Function $Label$ labeling nodes of any graph in $\mathcal{F}$ with distinct labels
2. Poly-time algorithm for deciding adjacency of two nodes given their labels

Note:

- Algorithm knows nothing beyond the labels (specifically, it does not know which graph they come from)

- Polynomiality is in the label length (logarithmic labels $\Rightarrow$ polylog time)

Example:

Adjacency Labeling on Trees

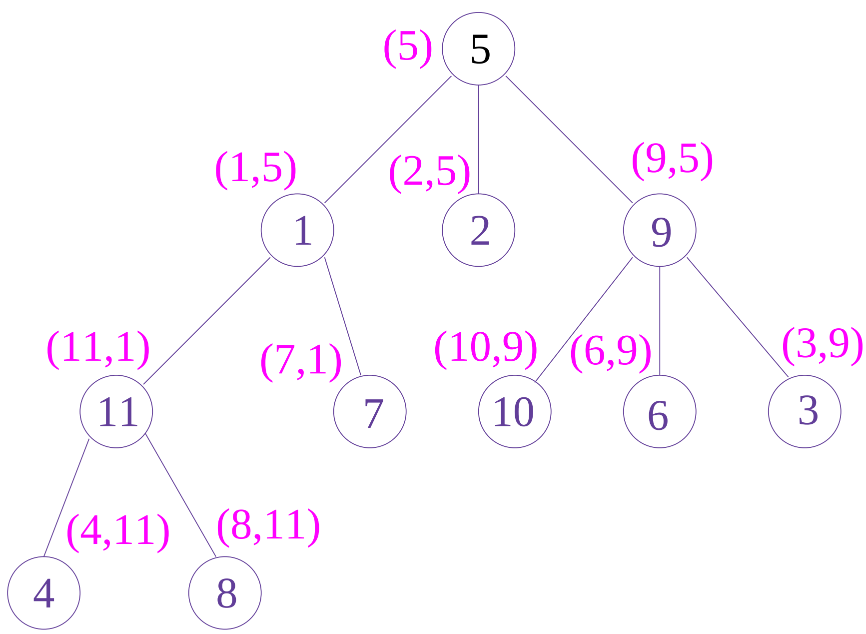
[Kannan,Naor,Rudich,88]

Marker Algorithm:

1. Arbitrarily root the tree and prelabel each node v with distinct integer $I(v)$ from $[1, n]$.

2. Label root r_0 by $Label(r_0) = (I(root))$.

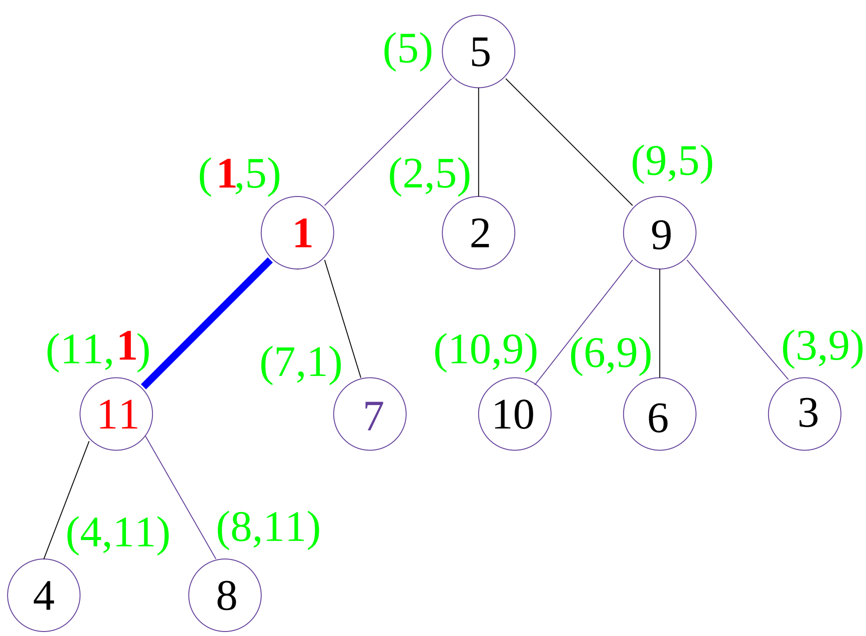
3. Label each nonroot node v with parent w by $Label(v) = (I(v), I(w))$.



New labels contain $\leq 2\lceil \log n \rceil$ bits.

Decoder Algorithm:

To decide adjacency of v, u: Check if
1st entry in $Label(v)$ = 2nd entry in $Label(u)$
or vice versa



Extensions

Note: Extendable to deciding neighborhood to distance k for fixed $k \geq 1$: simply label each node by prelabels of itself plus all ancestors up to height k.

Other examples:

- Bounded arboricity graphs (including bounded-degree / bounded-genus graphs, e.g. *planar graphs*),
- Intersection-based graphs (e.g. interval graphs),
- c-decomposable graphs

Question: Is this doable for *all* graphs with $O(\log n)$-bit labels?

Negative result

Note: Graph class $\mathcal{F}$ is adjacency-labelable using $O(\log n)$-bit labels

$\Rightarrow$ each G is fully defined by $O(n \log n)$ bits

$\Rightarrow$ # represented graphs $\leq 2^{O(n \log n)}$

Corollary: Graph family $\mathcal{F}$ containing $> 2^{O(n \log n)}$ n-node graphs is *not* adjacency-labelable using $O(\log n)$-bit labels

Negative examples: Bipartite / chordal graphs

Question: Can this be done
for *other* types of information
beyond adjacency?

Framework

Labeling: assignment $Label(v)$, $\forall v \in G$

Labeling scheme: $\langle \mathcal{M}, \mathcal{D} \rangle$

1. *Marker* algorithm $\mathcal{M}$:

   Given graph G,
   selects labeling $Label$ for G

2. *Decoder* algorithm $\mathcal{D}$:

   Given set of labels $\hat{L} = \{L_1, \ldots, L_k\}$,
   returns $\mathcal{D}(\hat{L})$

f labeling scheme for graph family $\mathcal{G}$

Marker-decoder pair $\langle \mathcal{M}_f, \mathcal{D}_f \rangle$ satisfying,
for function f on node subsets:

If marker $\mathcal{M}_f(G)$ assigns labeling $Label$ to G

then $\forall$ node subset W in G,
applying decoder $\mathcal{D}_f$
to set of labels $\hat{L}(W) = \{Label(v) \mid v \in W\}$
yields $f(W)$

Properties

- Decoder $\mathcal{D}_f$ is independent of G;
  knows nothing beyond the labels

- Decoding - polynomial in label length
  (logarithmic labels $\Rightarrow$ polylog time)

$\Downarrow$

$\langle \mathcal{M}_f, \mathcal{D}_f \rangle$ is a method for *storing* and *extracting* f-values in *"distributed"* fashion

Efficiency measure

Goal: f labeling schemes assigning *short* labels

$|Label(u)|$ = # bits in binary string $Label(u)$

Def: Given marker $\mathcal{M}$ assigning $Label$ to G

$$\mathcal{L}_{\mathcal{M}}(G) \;=\; \max_{u \in V} |Label(u)|$$

For finite graph family $\mathcal{G}$,

$$\mathcal{L}_{\mathcal{M}}(\mathcal{G}) = \max\{\mathcal{L}_{\mathcal{M}}(G) \mid G \in \mathcal{G}\}$$

Given function f and graph family $\mathcal{G}$,

$$\begin{aligned}\mathcal{L}(f,\mathcal{G}) \;=\; & \min\{\mathcal{L}_{\mathcal{M}}(\mathcal{G}) \mid \\ & \exists \mathcal{D},\ \langle \mathcal{M},\mathcal{D}\rangle = f \text{ labeling scheme for } \mathcal{G}\}\end{aligned}$$

Distance labeling schemes

Goal: Short labels that encode *distances* & algorithm for computing the distance between two nodes given their labels (in time polynomial in the label lengths)

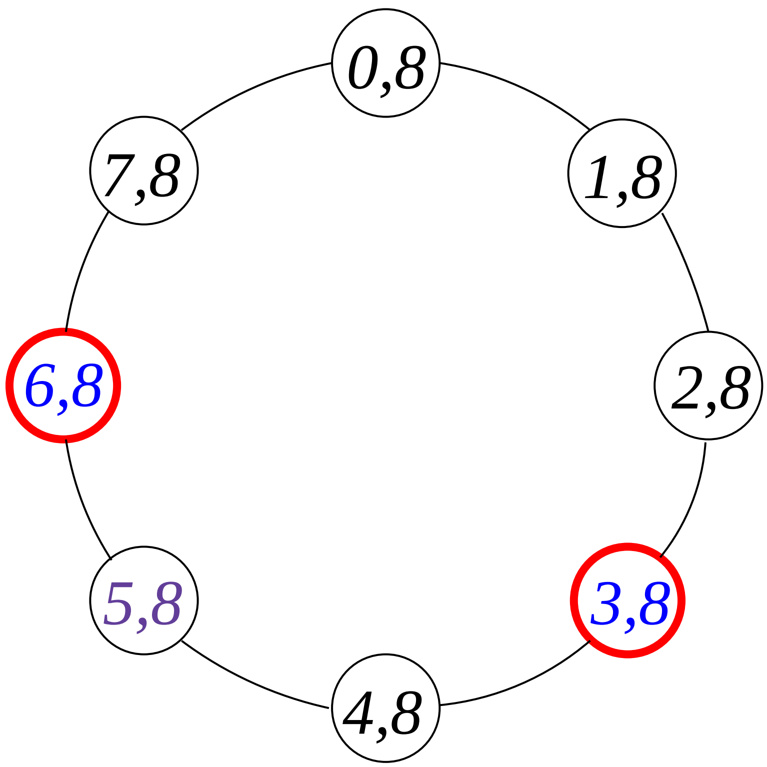
Simple examples:

Highly regular graph classes

- rings, meshes, tori, hypercubes -

enjoy $O(\log n)$-bit distance labeling schemes

Ring

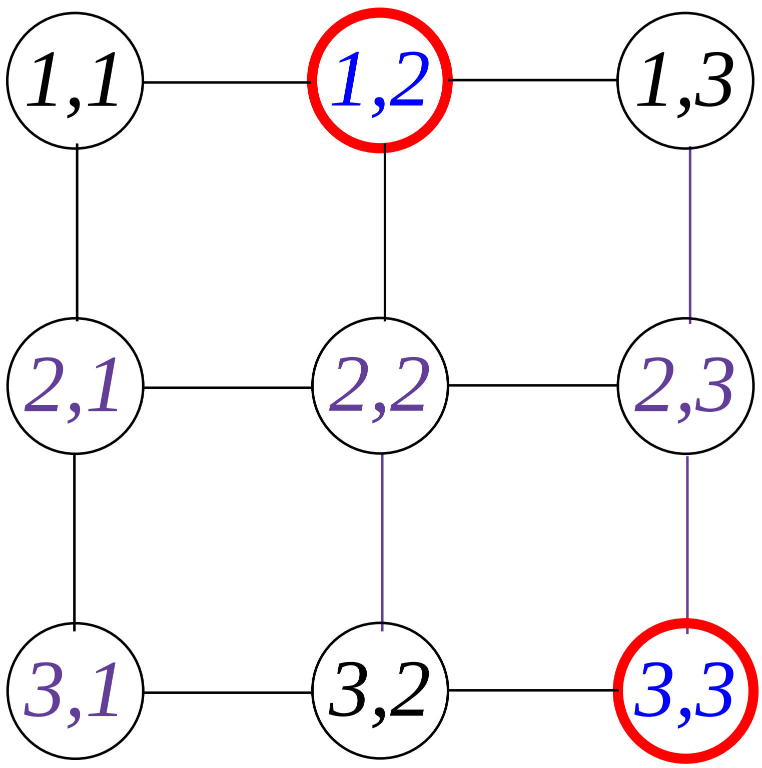


dist((3,8),(6,8)) =

min(6-3,8+3-6) =

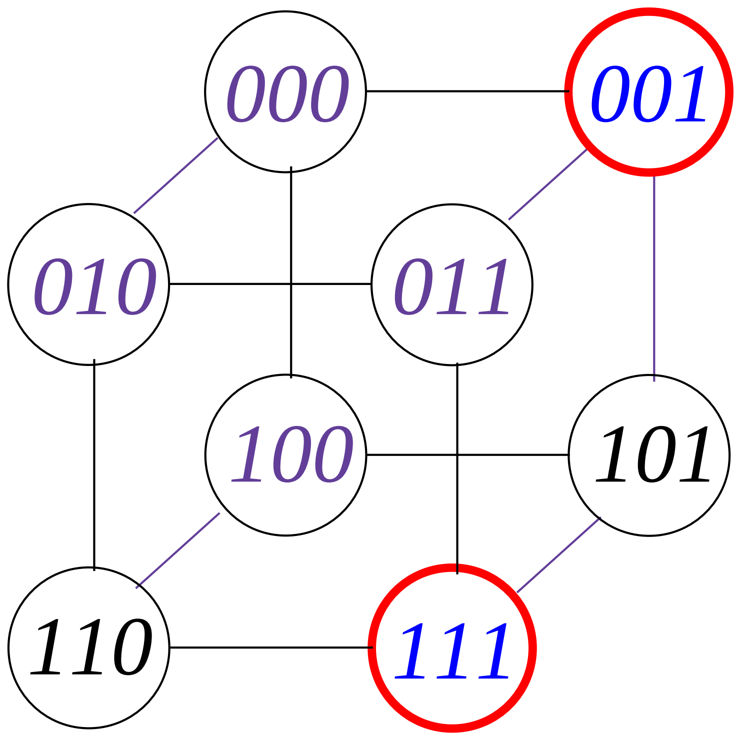
min(3.5) = 3

Grid



$$dist((1,2),(3,3)) =$$
$$|3-1| + |3-2| =$$
$$2 + 1 = 3$$

Hypercube



dist(001,111) = Hamming(001,111) = 2

Distance labeling on general graphs

Easy labeling:

Mark each node v_i in n-node graph G
on $V = \{v_1, \ldots, v_n\}$ with all its distances:

$$Label(v_i) = \langle v_i, dist_G(v_i, v_1), \ldots, dist_G(v_i, v_n) \rangle$$

$$\Downarrow$$

For the class $\mathcal{G}(n)$ of general n-node graphs,

$$\mathcal{L}(dist, \mathcal{G}(n)) = O(n \log n)$$

Thm: $\mathcal{L}(dist, \mathcal{G}(n)) = O(n)$
[Gavoille,P,Perennes,Raz,01]

Lower bound for general graphs

Def: For graph G on $V = \{v_1, \ldots, v_n\}$
and distance labeling scheme $\langle \mathcal{M}, \mathcal{D} \rangle$,

$$\mathcal{L}_{sum}(G) = \sum_{v_i \in V} |Label(v_i)|$$

Observ: the tuple $\langle Label(v_1), \ldots, Label(v_n) \rangle$
suffices to reconstruct the graph G
(by testing $\forall$ pair of nodes if distance $= 1$)
at cost $\mathcal{L}_{sum}(G) + O(n \log n)$ (for delimiters)

Cor: $\forall$ family $\mathcal{F}(n)$ of $2^{K(n)}$ n-node graphs
and distance labeling scheme $\langle \mathcal{M}, \mathcal{D} \rangle$ for $\mathcal{F}(n)$,
some graph $G \in \mathcal{F}(n)$ must satisfy
$\mathcal{L}_{sum}(G) + O(n \log n) \geq K(n)$

Thm: $\forall$ family $\mathcal{F}(n)$ of $2^{K(n)}$ n-node graphs

$$\mathcal{L}(dist, \mathcal{F}(n)) \ \geq \ \frac{K(n)}{n} - o(\log n)$$

For $\mathcal{G}(n)$ = family of n-node general graphs,

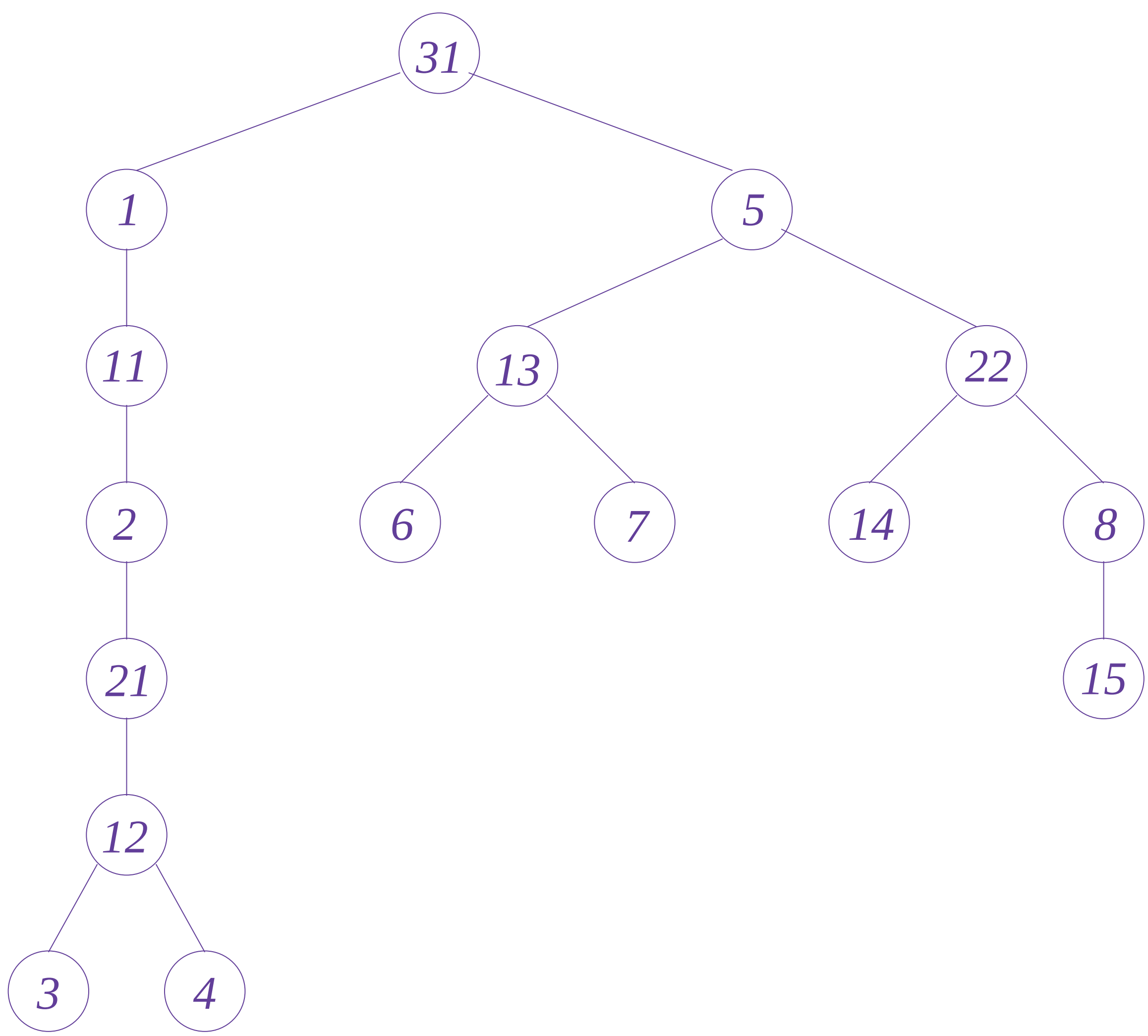
$$|\mathcal{G}(n)| = 2^{n(n-1)/2}$$

Cor: $\mathcal{L}(dist, \mathcal{G}(n)) \;\geq\; \frac{n-1}{2} - o(1)$

Similar bounds apply for other large
graph families (bipartite / chordal / ...)

Distance labeling on trees
[P,99]

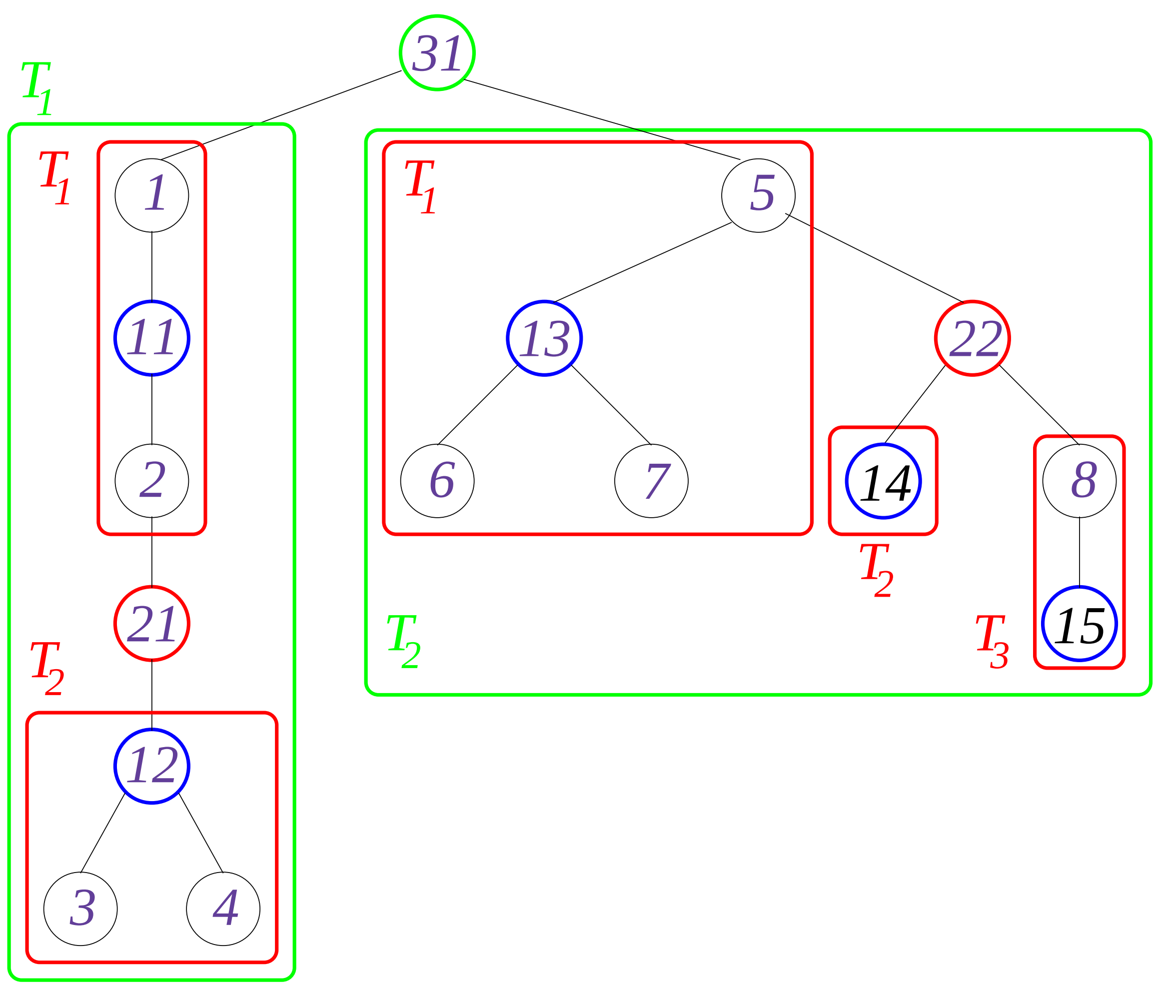
Preprocessing: Arbitrarily prelabel each node v of T with distinct integer $I(v)$ from $[1, n]$



Recursive partitioning

Tree separator: Node v_0 in n-node tree T, whose removal breaks T into subtrees of size $\leq n/2$

Fact: Every tree T has a separator (can be found in linear time)



Label assignment

Recursive marker procedure `Sub_Tree_Label`(T')
(Applied to subtree T')

1. **If** T' contains single node v_0
   then set $Label(v_0) \leftarrow (I(v_0)\ ,\ 0\ ,\ 0)$
   and return

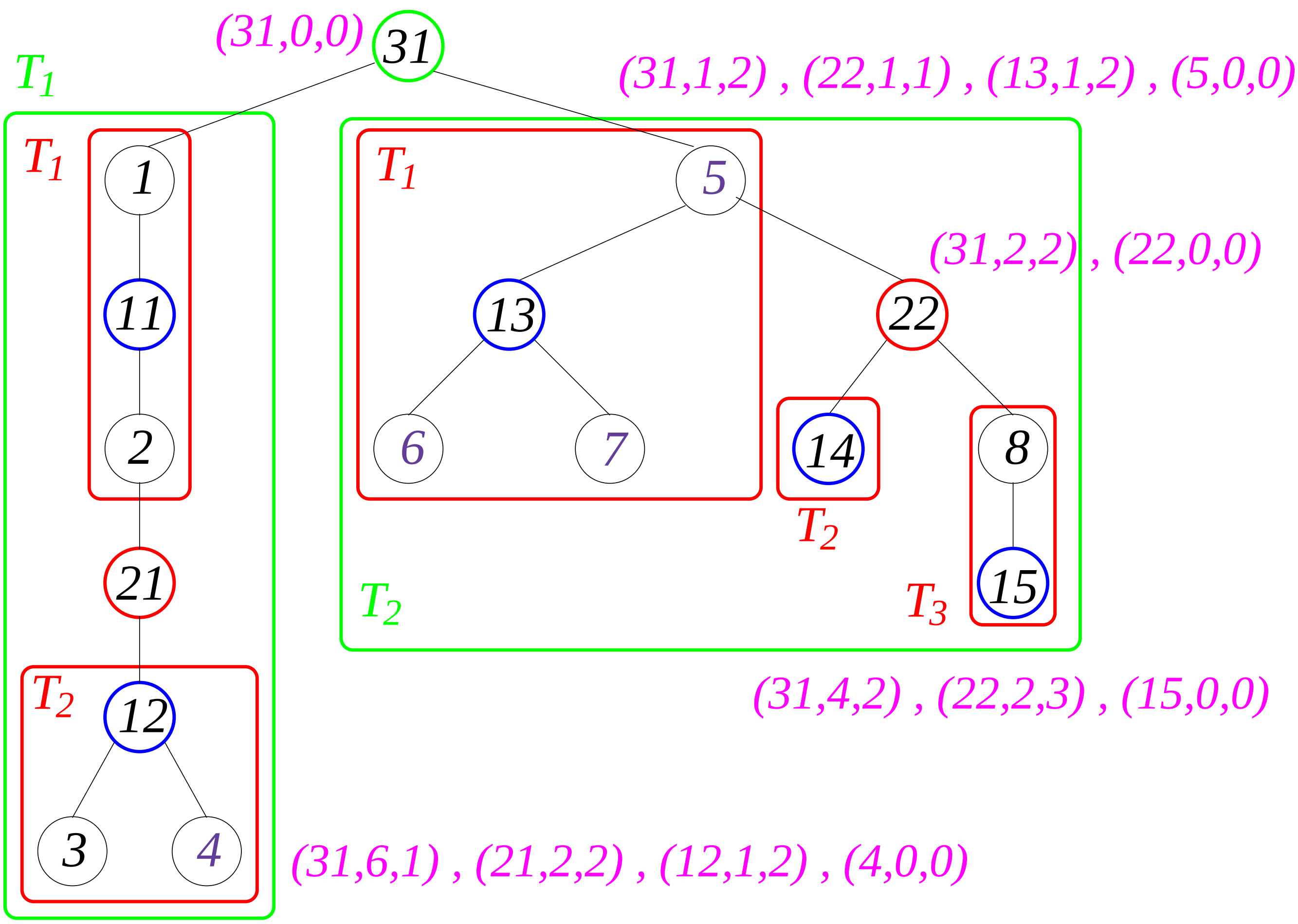
2. */* T' contains > 1 node */*
   Find separator v_0 for T' and remove it.
   / Breaks T' into $T_1, \ldots$ of size $\leq n/2$ */*

3. Recursively apply Procedure `Sub_Tree_Label`(T_i),
   label each v in each subtree T_i by $Label_i(v)$

4. **For** each node $v \in T_i$ **do:**
   - Let $\mathcal{J}(v) \leftarrow (I(v_0)\ ,\ dist(v, v_0, T')\ ,\ i)$
   - Label v by $Label(v) \leftarrow \mathcal{J}(v) \circ Label_i(v)$

5. Label v_0 by $Label(v_0) \leftarrow (I(v_0)\ ,\ 0\ ,\ 0)$

Label assignment - example



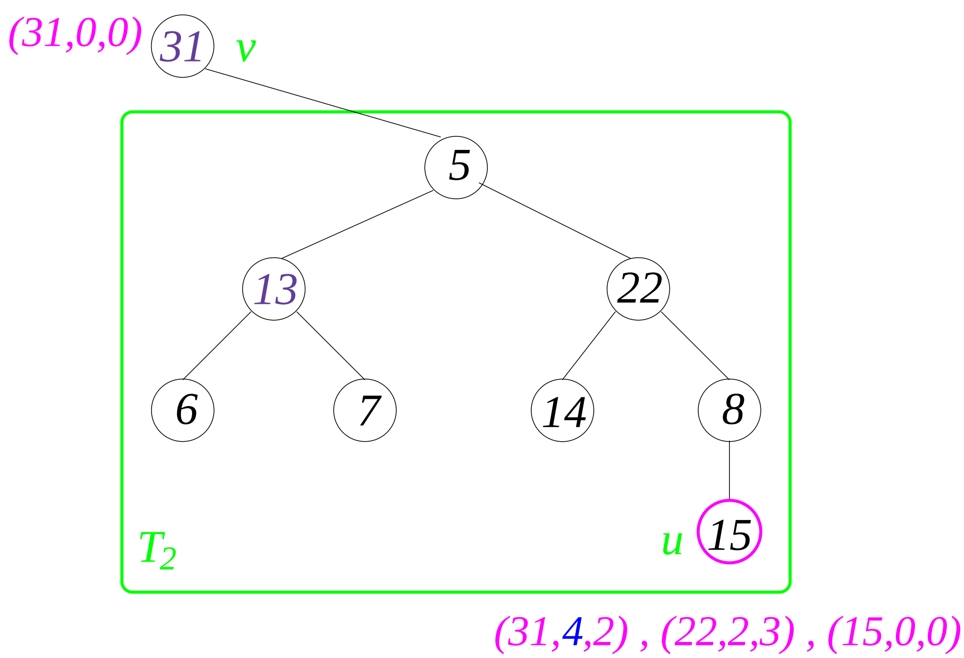
Fields: *(separator , distance , tree #)*

Distance computation

Consider two nodes u, v in T, with labels
$Label(u) = \mathcal{J}_1(u) \circ \ldots \circ \mathcal{J}_q(u)$
$Label(v) = \mathcal{J}_1(v) \circ \ldots \circ \mathcal{J}_p(v)$

Decoder Procedure `Dist_Calc`

$p = 1$: /* v is the separator of T */
Return 2nd field in $\mathcal{J}_1(u)$



Fields: *(separator , distance , tree #)*

$q = 1$: Return 2nd field in $\mathcal{J}_1(v)$

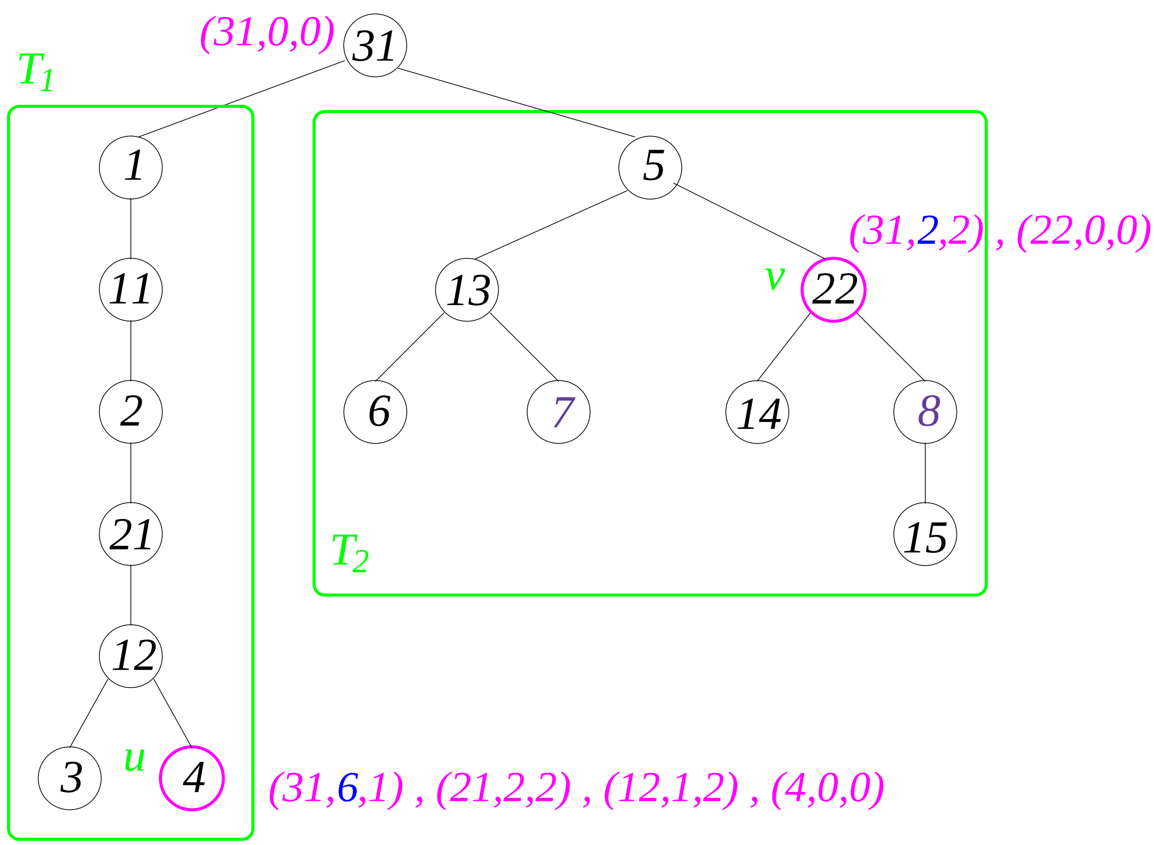
$p, q > 1$: Let

$\mathcal{J}_1(u) = (I(w)\ ,\ dist(u, w, T)\ ,\ i)$

$\mathcal{J}_1(v) = (I(w)\ ,\ dist(v, w, T)\ ,\ j)$

$i \neq j$: /* u, v in different subtrees */

Return sum of 2nd fields in $\mathcal{J}_1(u)$ and $\mathcal{J}_1(v)$

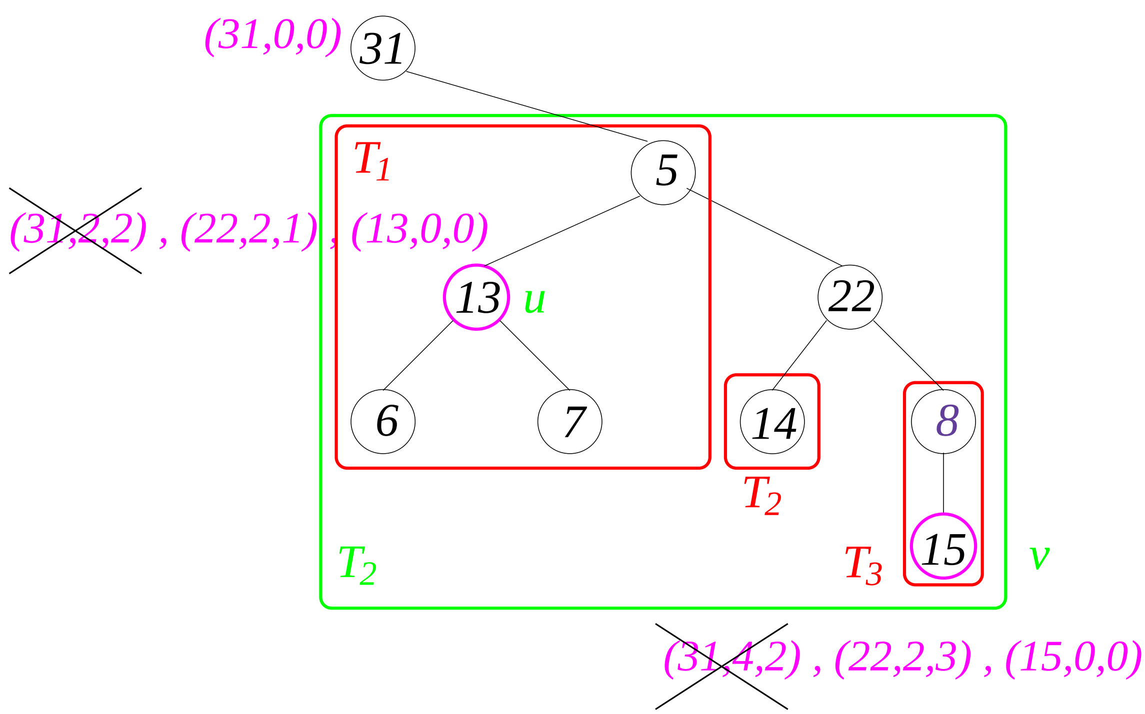


Fields: *(separator , distance , tree #)*

$i = j$: /* u, v in same subtree */

- Discard 1st triple from $Label(u)$, $Label(v)$, remaining with
  $Label_i(u) = \mathcal{J}_2(u) \circ \ldots \circ \mathcal{J}_q(u)$
  $Label_i(v) = \mathcal{J}_2(v) \circ \ldots \circ \mathcal{J}_p(v)$

- Invoke procedure `Dist_Calc` recursively on $Label_i(u)$ and $Label_i(v)$;
  Return $dist(u, v, T_i)$



Fields: *(center , distance , tree #)*

Analysis

Lemma: Label length is $O(\log^2 n)$

Proof: Each sublabel $\mathcal{J}(v)$ has $\leq \lceil 3 \log n \rceil$ bits.
Recursion has $\leq \log n$ levels.

Lemma: `Dist_Calc` computes distances correctly

Theorem: $\exists$ $O(\log^2 n)$ distance labeling scheme for class of n-node trees

Approach extends to c-decomposable graphs
(fixed c, affects constant in length bound)

Theorem: $\exists$ $O(\log^2 n)$ distance labeling scheme for class of n-node c-decomposable graphs

Lower bound for trees
[Gavoille,P,Perennes,Raz,01]

$\mathcal{F}$= all labeled trees on $V_n = \{1, \ldots n\}$

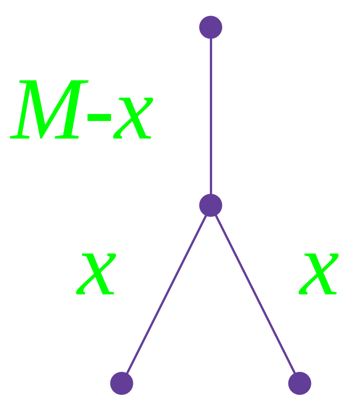
Cayley's formula: $|\mathcal{F}| = n^{n-2}$

$\Rightarrow$ avg/max label length $=\Omega(\log n)$

Goal: For family $\mathcal{T}(n, M)$ = n-node trees with weights from $[0, M-1]$, any distance labeling scheme requires $\mathcal{L}(dist, \mathcal{T}(n, M)) = \Omega((\log M + \log n) \log n)$

The class of trees

$(1, M)$-tree T: integral $x \in [0, M-1]$



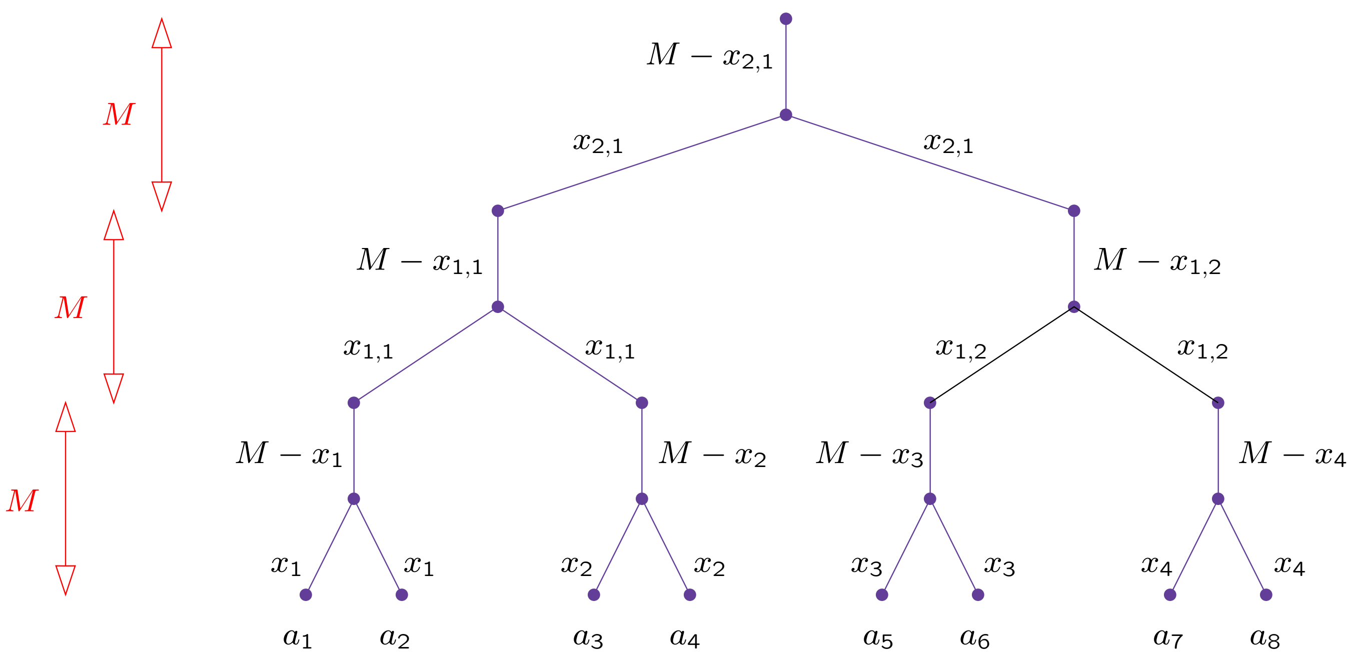
(h,M)-**tree:** $(1,M)$-tree with

$(h-1,M)$-tree attached to each leaf

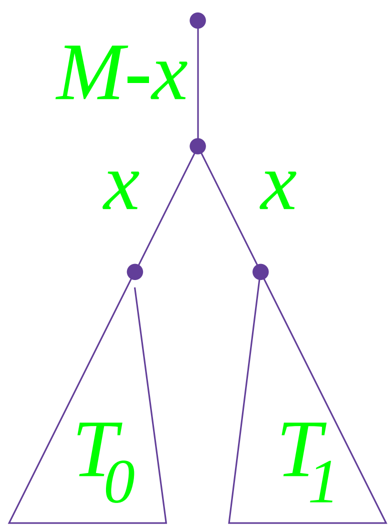
$\Rightarrow$ 2^h leaves, $a_1,\ldots,a_{2^h}$.

$\mathcal{C}(h,M)$= class of all (h,M)-trees

(same structure, different weight assignments)



Note: (h, M)-tree fixed by triple (T_0, T_1, x) :
$x =$ weight of two edges of top $(1, M)$-tree,
$T_0, T_1 =$ two attached $(h-1, M)$-trees



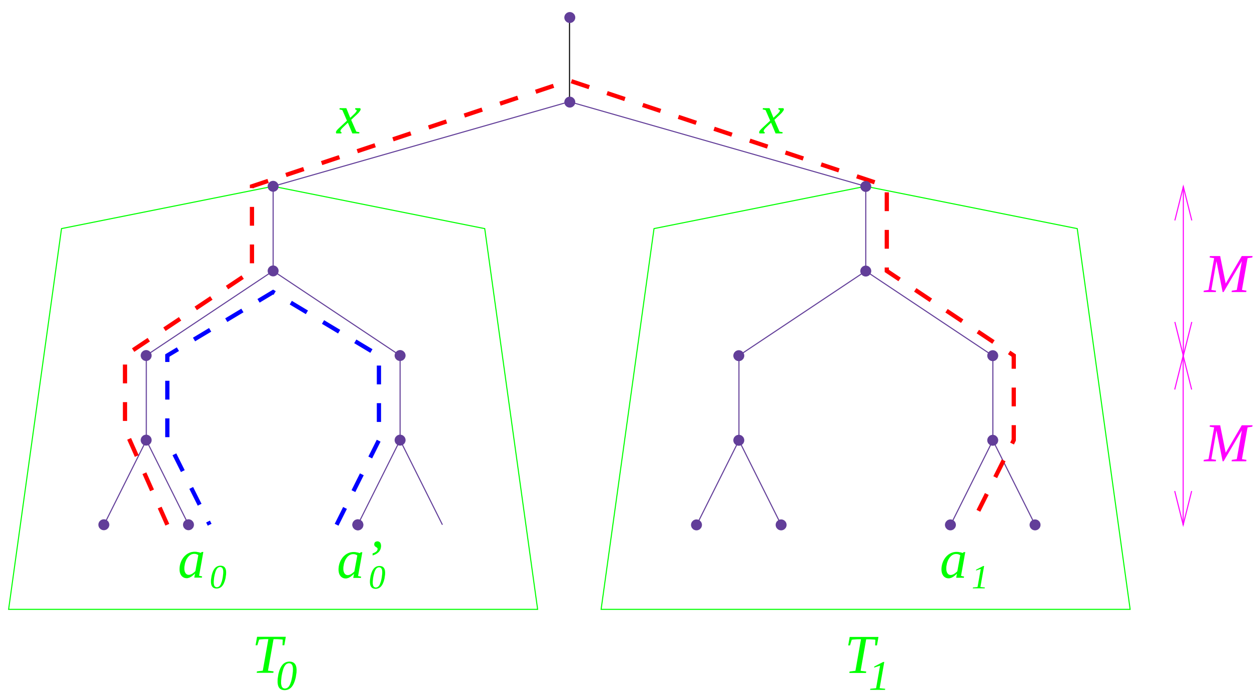
$\mathcal{C}(h, M, x) =$ subclass of $\mathcal{C}(h, M)$ consisting of (h, M)-trees with topmost weight x

$$\mathcal{C}(h, M) = \bigcup_{x=0}^{M-1} \mathcal{C}(h, M, x)$$

Lemma: $\forall$ two leaves of $T \in \mathcal{C}(h, M, x)$

$$a_0, a'_0 \in T_0 \;\Rightarrow\; d_T(a_0, a'_0) = d_{T_0}(a_0, a'_0)$$

$$a_0 \in T_0 \;\wedge\; a_1 \in T_1 \;\Rightarrow\; d_T(a_0, a_1) = 2(h-1)M + 2x$$

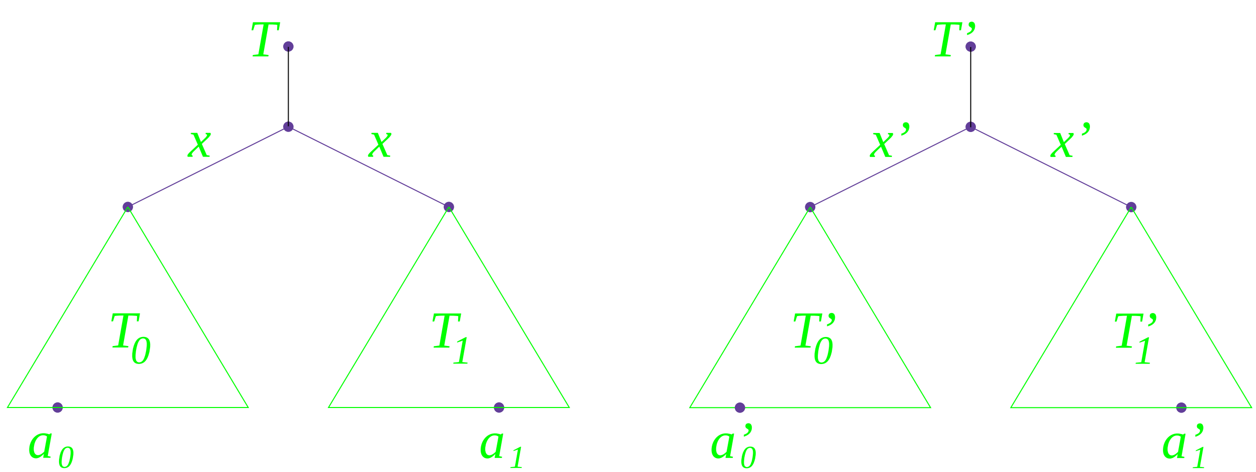


Cor: For (h, M)-trees $T = (T_0, T_1, x)$,

$T' = (T'_0, T'_1, x')$,

and leaves a_0, a_1, a'_0, a'_1 as drawn,

$$d_T(a_0, a_1) = d_{T'}(a'_0, a'_1) \iff x = x'$$



Label sets

For scheme $\langle \mathcal{M}, \mathcal{D} \rangle$ on $\mathcal{C}(h, M)$:
$W(\mathcal{M}, h, M) =$ set of all labels $Label(v)$ assigned by $\mathcal{M}$

$g(h, M) = \min |W(\mathcal{M}, h, M)|$
over all possible schemes for $\mathcal{C}(h, M)$

$\hat{L} \leftarrow$ scheme attaining $g(h, M)$
$(|W(\hat{L}, h, M)| = g(h, M))$

$W(h, M, x) = \{(\lambda_0, \lambda_1) \mid$ labels λ_0, λ_1 assigned by $\hat{L}$
to some leaves $a_j \in T_0,\ a_t \in T_1,$ for
some tree $T = (T_0, T_1, x) \in \mathcal{C}(h, M, x)\}$

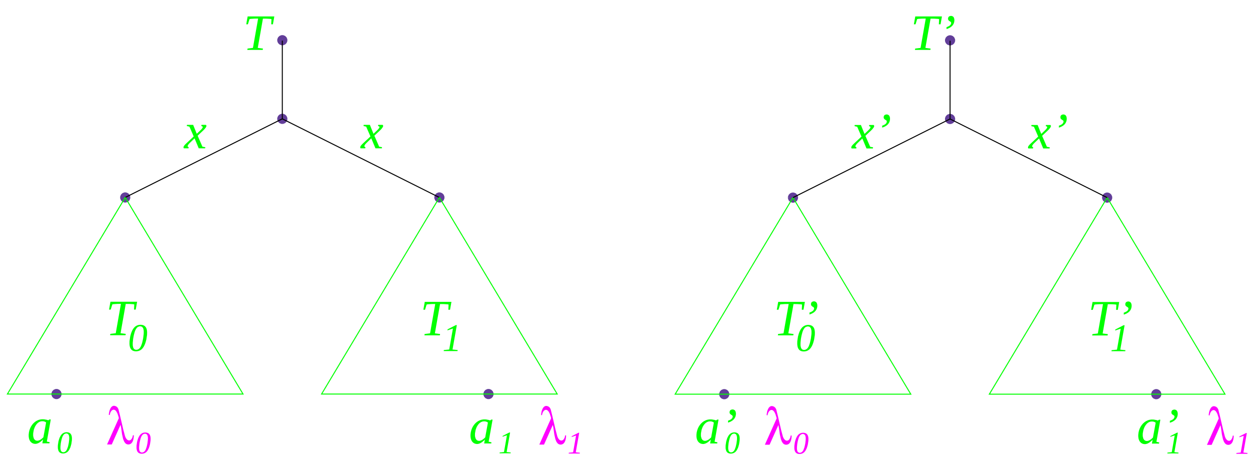
$\mathcal{W} = \bigcup_{x=0}^{M-1} W(h, M, x)$

Lemma: $|\mathcal{W}| \leq g(h, M)^2$

Proof: $\mathcal{W} \subseteq W(\hat{L}, h, M) \times W(\hat{L}, h, M)$

Lemma: $\forall x \neq x'$, $W(h,M,x) \cap W(h,M,x') = \emptyset$

Proof: Suppose $x \neq x'$ yet
$\exists\ (\lambda_0, \lambda_1) \in W(h,M,x) \cap W(h,M,x')$.
Then $\exists$ two $(h-1,M)$-trees T, T'



$$
\begin{aligned}
2(h-1)M + 2x &= d_T(a_0, a_1) \\
&= f(\lambda_0, \lambda_1) \\
&= d_{T'}(a'_0, a'_1) \ = \ 2(h-1)M + 2x'
\end{aligned}
$$

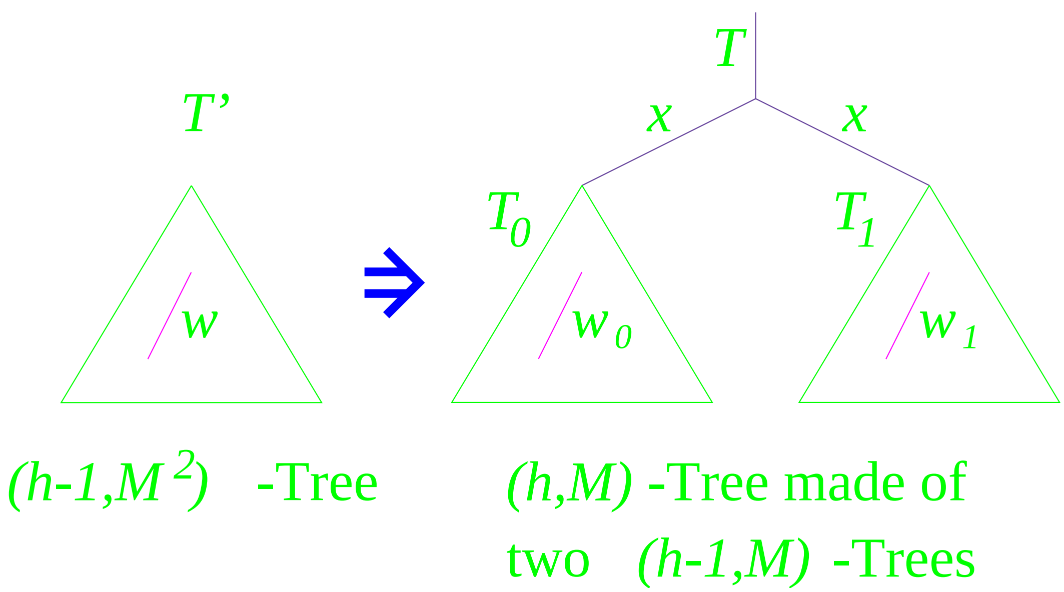
implying $x = x'$, contradiction.

Main Lemma: $\forall\ 0 \leq x < M$,

$$|W(h, M, x)| \geq g(h-1, M^2)$$

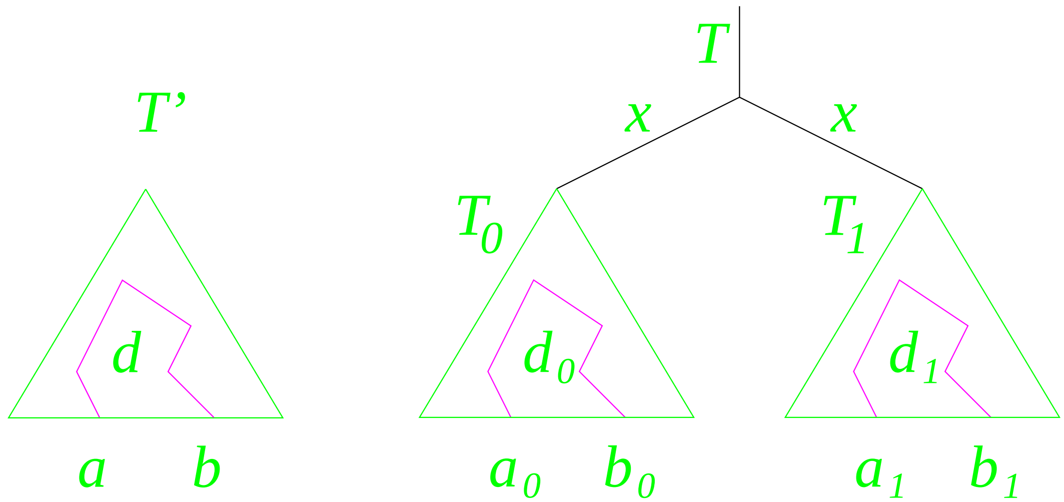
Proof idea: Derive a labeling scheme for all $(h-1, M^2)$-trees using $\leq |W(h, M, x)|$ labels

Tree representation: Match $(h-1, M^2)$-tree T' with (h, M)-tree $T = (T_0, T_1, x)$ in $\mathcal{C}(h, M, x)$



Weight representation: $w \in [0, M^2 - 1]$ in T' represented as $\boxed{w = w_0 + M \cdot w_1}$ s.t. $w_0, w_1 \in [0, M-1]$

Leaf mapping: Associate leaf a of T' with two homologous leaves a_0, a_1 of T



$$\boxed{d = d_0 + M \cdot d_1}$$

$$\begin{aligned} d_{T'}(a,b) &= d_{T_0}(a_0,b_0) + M \cdot d_{T_1}(a_1,b_1) \\ &= d_T(a_0,b_0) + M \cdot d_T(a_1,b_1) \end{aligned}$$

Labeling T' (given $\hat{L}$ for T):

1. Use $\hat{L}$ to label T

2. Label leaf $a \in T'$ by

$$Label'(a) = \left\langle \hat{L}(a_0, T), \hat{L}(a_1, T) \right\rangle$$

(Note: pair belongs to $W(h, M, x)$)

Distance decoder $\mathcal{D}'$ for T' (given f for T):

$$\mathcal{D}'(Label'(a), Label'(b)) = f(\hat{L}(a_0), \hat{L}(b_0)) + M \cdot f(\hat{L}(a_1), \hat{L}(b_1))$$

$\Rightarrow$ $\langle \mathcal{M}', \mathcal{D}' \rangle$ labels $(h-1, M^2)$-trees with labels from $W(h, M, x)$

$\Rightarrow$ $|W(h, M, x)| \geq g(h-1, M^2)$

Cor: $g(h, M) \geq \sqrt{M} \cdot \sqrt{g(h-1, M^2)}$

Proof: $g(h, M)^2 \geq |\mathcal{W}| \geq M \cdot g(h-1, M^2)$

Cor: $g(h, M) \geq M^{h/2}$

Proof: By induction on h.

Cor: $\mathcal{L}(dist, \mathcal{C}(h, M)) \geq \frac{h}{2} \cdot \log M$

Setting $h = \Theta(\log n)$:

Thm: For family $\mathcal{T}(n, M)$ of n-node binary trees with weights from $[0, M-1]$,

$$\mathcal{L}(dist, \mathcal{T}(n, M)) \;\geq\; \frac{1}{2}(\log n - 2) \log M$$

Proof:

$$\mathcal{L}(dist, \mathcal{C}(h, M)) \geq \frac{h}{2} \cdot \log M$$

$$n = 3 \cdot 2^h - 2 \quad \Rightarrow \quad h \geq \log n - 2 \quad \Rightarrow$$

$$\mathcal{L}(dist, \mathcal{T}(n, M)) \;\geq\; \frac{1}{2}(\log n - 2) \log M$$

Cor: For family $\mathcal{T}(n)$ of *unweighted* n-node binary trees,

$$\mathcal{L}(dist, \mathcal{T}(n)) \;\geq\; \frac{1}{8} \log^2 n - O(\log n)$$

Distance labeling for planar graphs
[Gavoille,P,Perennes,Raz,01]

Idea: Recursively use $O(\sqrt{n})$-*separator property*

Def: *Separator* for n-node graph $G =$
node subset S whose deletion splits G
into connected components of size $\leq \alpha n$,
for constant $\alpha < 1$

Def: Graph class $\mathcal{G}$ has $r(n)$-*separator* if
$\forall$ connected n-node graph $G \in \mathcal{G}$
$\exists$ separator S of size $\leq r(n)$
s.t. after removing S, every resulting connected
component belongs to $\mathcal{G}$

Separator-based distance labeling

Consider family $\mathcal{G}(n)$ of graphs with $r(n)$-separator

Marker algorithm $\mathcal{M}$: Given graph $G \in \mathcal{G}(n)$:

1. Choose separator S of size $\leq r(n)$ for G.

2. $\forall$ connected component A of $V(G) \setminus S$,
   G_A = graph induced by nodes of A
   $c = \#$ components

3. Mark each component A by unique identifier
   $ID(A) \in [0, 1, 2, \ldots, c-1]$

4. Asign separator S the identifier $ID(S) = c$

5. Fix *ordering* on nodes of S

6. $\forall$ component G_A, apply marker $\mathcal{M}$ *recursively*

Marker algorithm (cont.)

7. Assign label to each node x in component A, with following fields:

[a] List of distances to nodes of S,
by fixed ordering
($\leq |S| \log n$ bits)

[b] Identifier $ID(A)$
($\leq \log n$ bits)

[c] Recursive Label $Label(x, G_A)$
($\leq \mathcal{L}_{\mathcal{M}}(\mathcal{G}(|A|))$ bits)

8. Assign label to each node x in S, with only first two fields

Decoding

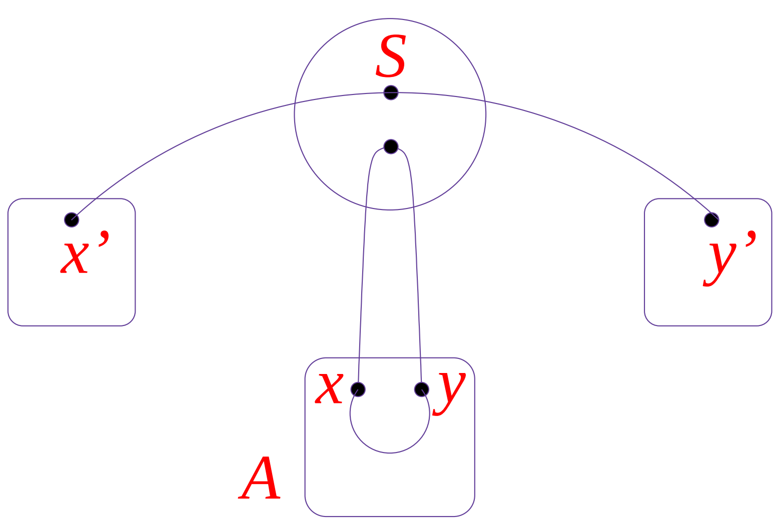
Def: For nodes $x, y \in V(G)$, *Distance via S* is

$$\hat{d}(x,y) \;=\; \min_{s\in S}\{dist_G(x,s) + dist_G(s,y)\}$$

Observ:

1. If x and y belong to same component A, then
   $dist_G(x,y) \;=\; \min\{dist_{G_A}(x,y), \hat{d}(x,y)\}$

2. Otherwise, $dist_G(x,y) = \hat{d}(x,y)$



Decoder algorithm $\mathcal{D}$

To compute $dist_G(x, y)$:

1. Compute $\widehat{d}(x, y)$ using field [a] of $Label(x, G)$ and $Label(y, G)$.

2. Compare component identifier $ID(A)$ of x and y from field [b] of $Label(x, G)$ and $Label(y, G)$

3. If they are equal and $\neq c$
   /* x and y belong to same component */
   then do:

   (a) From field [c] of $Label(x, G)$ and $Label(y, G)$, get $Label(x, G_A)$ and $Label(y, G_A)$

   (b) Compute $dist_{G_A}(x, y)$ recursively

   (c) $dist_G(x, y) \ \leftarrow \ \min\{dist_{G_A}(x, y), \widehat{d}(x, y)\}$

4. Else return $\widehat{d}(x, y)$

Analysis

$|S| \leq r(n)$
$|A| \leq \alpha \cdot n \quad \forall$ connected component A
$\Downarrow$

$$\mathcal{L}(dist, \mathcal{G}(n)) \;\leq\; \mathcal{L}\left(dist, \mathcal{G}(\alpha \cdot n)\right) + r(n) \log n + \log n$$

solving to

$$\mathcal{L}(dist, \mathcal{G}(n)) = O(R(n) \log n) + O(\log^2 n)$$

where

$$R(n) = \sum_{i \leq \log_{1/\alpha} n} r(\alpha^i \cdot n)$$

Thm: $\forall$ family $\mathcal{G}(n)$ of graphs with $r(n)$-separator,

$$\mathcal{L}(dist, \mathcal{G}(n)) \;=\; O(R(n) \log n + \log^2 n)$$

Distance labeling on planar graphs

Thm *[Lipton,Tarjan,79]***:**
n-node planar graphs have $O(\sqrt{n})$-separator

Cor: For the family $\mathcal{P}_n$ of n-node planar graphs,

$$\mathcal{L}(dist, \mathcal{P}_n) \ \leq \ O(\sqrt{n} \log n)$$

Lower bound for planar graphs
[Gavoille,P,Perennes,Raz,01]

Idea: Focus on labels of small node sets

Let $A \subseteq V_n = \{1, \ldots, n\}$
Consider family $\mathcal{F}$ of labeled graphs on V_n

Def: Two graphs $G, H \in \mathcal{F}$ *exhibit gap over A*
if $\exists\ x, y \in A$ s.t. $dist_G(x, y) \neq dist_H(x, y)$

$\mathcal{F}$ is *A-family* if
$\forall$ two distinct $G, H \in \mathcal{F}$ exhibit gap over A.

For $\mathcal{F} = A$-family for $A = \{a_1, \ldots, a_\alpha\}$
and $\langle \mathcal{M}, \mathcal{D} \rangle =$ distance labeling scheme on $\mathcal{F}$,
$\forall$ graph $G \in \mathcal{F}$:

$\vec{L}(G) = \langle Label(a_1, G), \ldots, Label(a_\alpha, G) \rangle$
$\mathcal{S} = \{\vec{L}(G) \mid G \in \mathcal{F}\}$

Lemma: For A-family $\mathcal{F}$,
$\vec{L}(G) \neq \vec{L}(H)$ $\forall$ distinct $G, H \in \mathcal{F}$
(i.e., $|\mathcal{S}| = |\mathcal{F}|$)

Proof: By contradiction.
Assume $\vec{L}(G) = \vec{L}(H)$ for some $G, H \in \mathcal{F}$, i.e.
$\forall\ a_i \in A, \quad Label(a_i, G) = Label(a_i, H)$

By definition of $\mathcal{F}$, $\exists\ a_i, a_j \in A$ s.t.
$dist_G(a_i, a_j) \neq dist_H(a_i, a_j)$

But since $\langle \mathcal{M}, \mathcal{D} \rangle$ is distance labeling scheme,

$$\begin{array}{lcl} \mathcal{D}(Label(a_i, G), Label(a_j, G)) & = & dist_G(a_i, a_j) \\ \quad \| & & \\ \mathcal{D}(Label(a_i, H), Label(a_j, H)) & = & dist_H(a_i, a_j) \end{array}$$

contradiction. ∎

Main Thm: For A-family $\mathcal{F}$,

$$\mathcal{L}(dist, \mathcal{F}) \geq \frac{1}{|A|} \cdot \log |\mathcal{F}|$$

Proof: Let $\mathcal{L}(dist, \mathcal{F}) = X$, $\quad \alpha = |A|$

Then $\exists$ distance labeling scheme $\langle \mathcal{M}, \mathcal{D} \rangle$ for $\mathcal{F}$ s.t. $\mathcal{L}_{\mathcal{M}}(G) \leq X \quad \forall\, G \in \mathcal{F}$

$\Rightarrow \ \mathcal{S} \subseteq [0, 2^X - 1]^\alpha$

By Lemma, $|\mathcal{F}| = |\mathcal{S}| \leq 2^{X\alpha}$

$\Rightarrow \ \log |\mathcal{F}| \leq \mathcal{L}_{\mathcal{M}}(\mathcal{F}) \cdot |A|$

$\Rightarrow \ \mathcal{L}_{\mathcal{M}}(\mathcal{F}) \geq \log |\mathcal{F}| / |A|$. $\blacksquare$

The lower bound

Thm: $\mathcal{L}(dist, \mathcal{P}_n) = \Omega(n^{1/3})$

Proof outline:

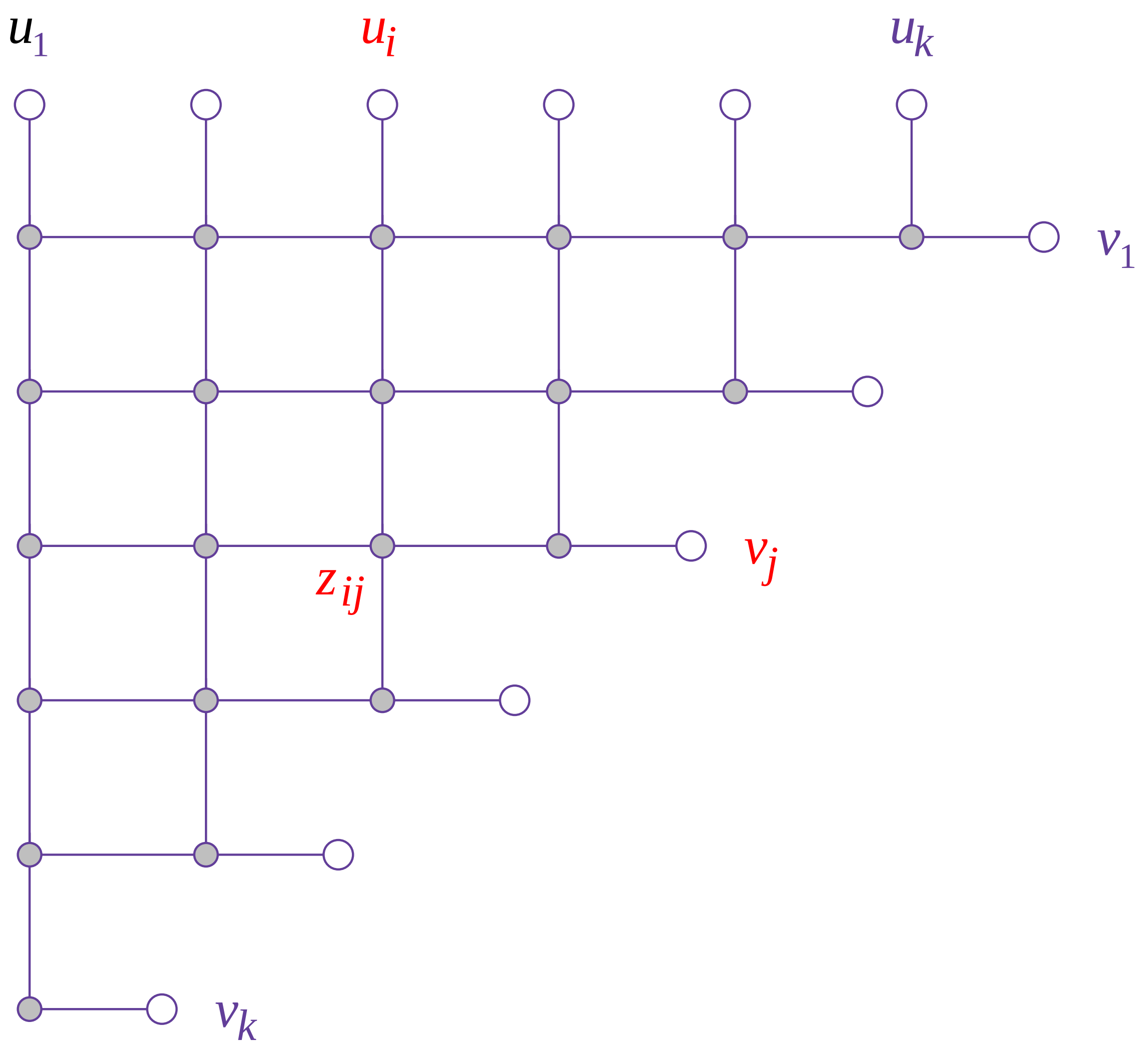
1. Construct subclass $\widehat{\mathcal{P}}_n$ of planar n-node graphs s.t.

   - $\widehat{\mathcal{P}}_n = A$-family for set of size $|A| = O(n^{1/3})$
   - $\log |\widehat{\mathcal{P}}_n| = \Omega(n^{2/3})$

2. Apply Main Thm to conclude

$$\mathcal{L}(dist, \widehat{\mathcal{P}}_n) = \Omega(n^{1/3})$$

Underlying structure G_k

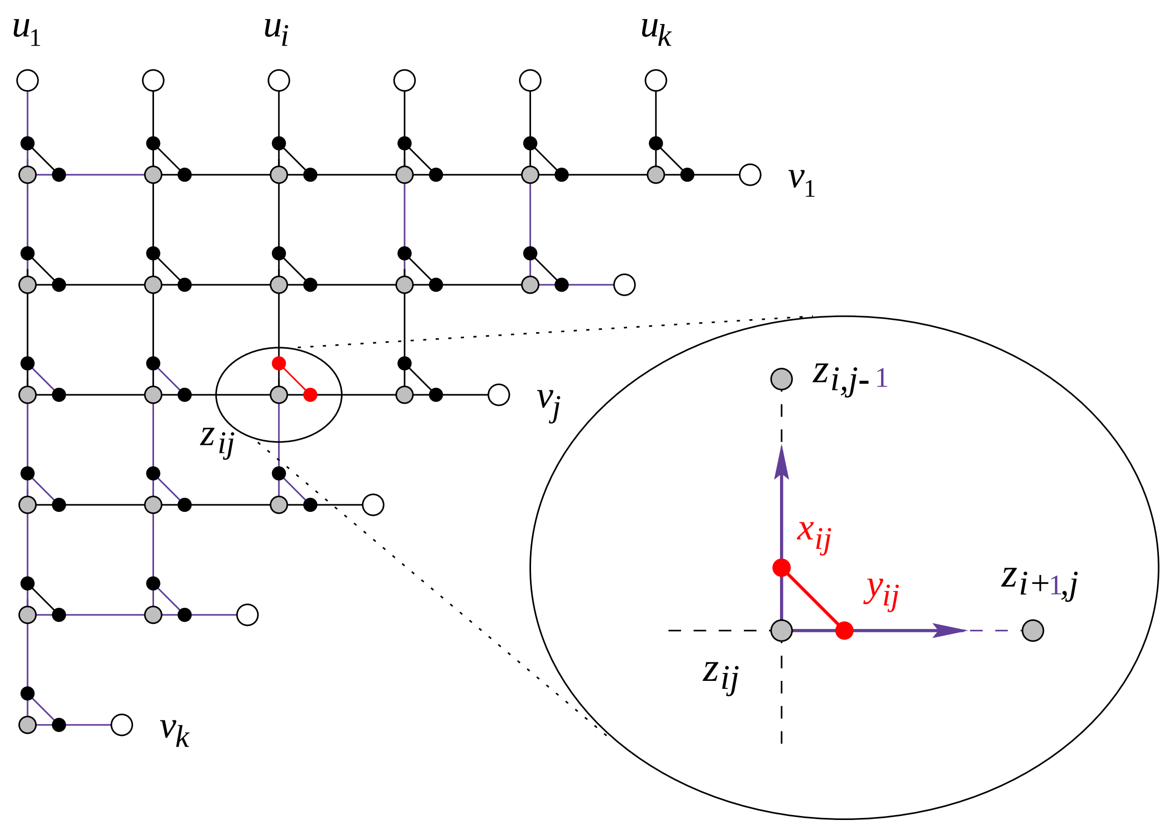


Upper-left half of $k \times k$ grid

$z_{i,j}$ = node on ith column, jth row

Attached leaves u_i, v_j at borders

Underlying structure G_k (cont.)

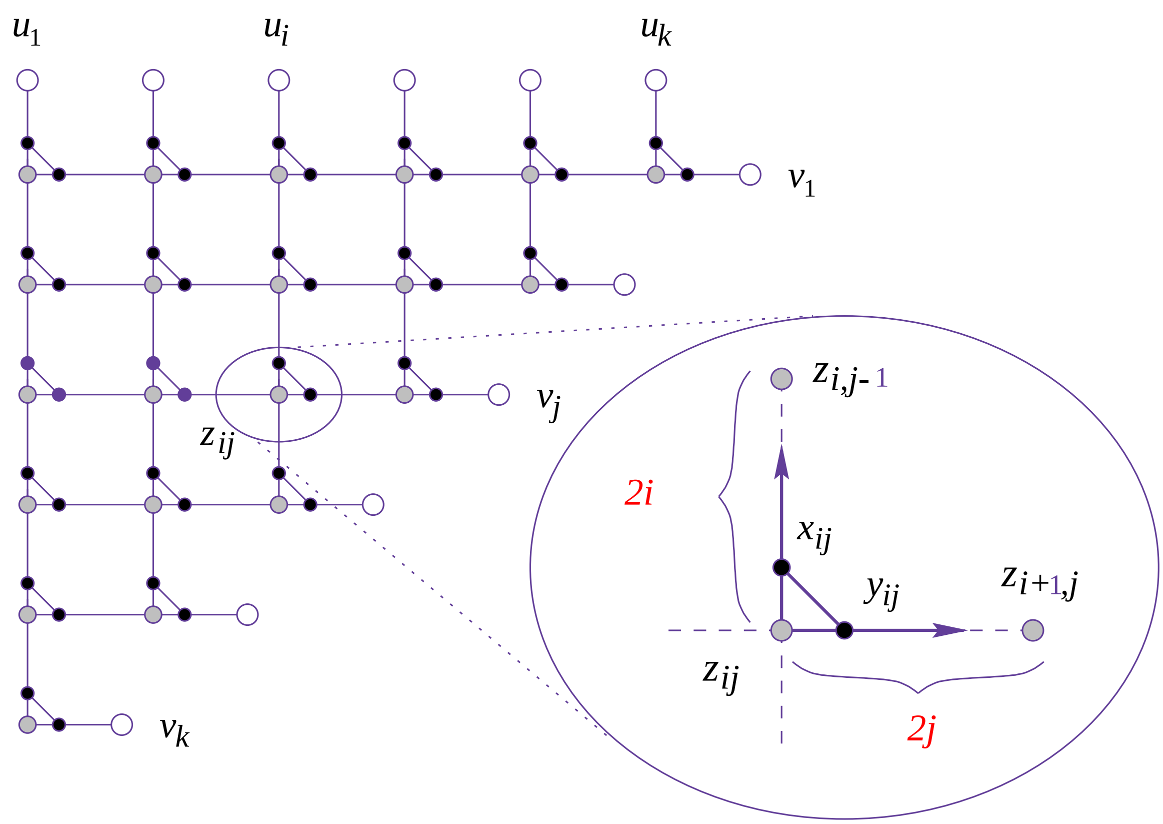


Each *vertical* edge $(z_{i,j}, z_{i,j-1})$ subdivided into $(z_{i,j}, x_{i,j})$ and $(x_{i,j}, z_{i,j-1})$ by adding node $x_{i,j}$

Each *horizontal* edge $(z_{i,j}, z_{i+1,j})$ subdivided into $(z_{i,j}, y_{i,j})$ and $(y_{i,j}, z_{i+1,j})$ by adding node $y_{i,j}$

Added *diagonal* edges $e_{i,j} = (x_{i,j}, y_{i,j})$, $\forall\ i, j$

Underlying structure G_k (cont.)



Edge weight assignment:

Edges $(x_{i,j}, z_{i,j-1})$ assigned weight $2i-1$,

Edges $(y_{i,j}, z_{i+1,j})$ assigned weight $2j-1$,

$\forall\ 2 \le i+j \le k+1$

All other edges assigned $w(e) = 1$

Note: G_k can be made *unweighted* by replacing each edge e of weight $w(e)$ with simple path of $w(e)$ edges

Resulting *unweighted* G_k has $n = \Theta(k^3)$

The A-family $\widehat{\mathcal{P}}_n$

$A = \{u_1, \ldots, u_k, v_1, \ldots, v_k\}$.

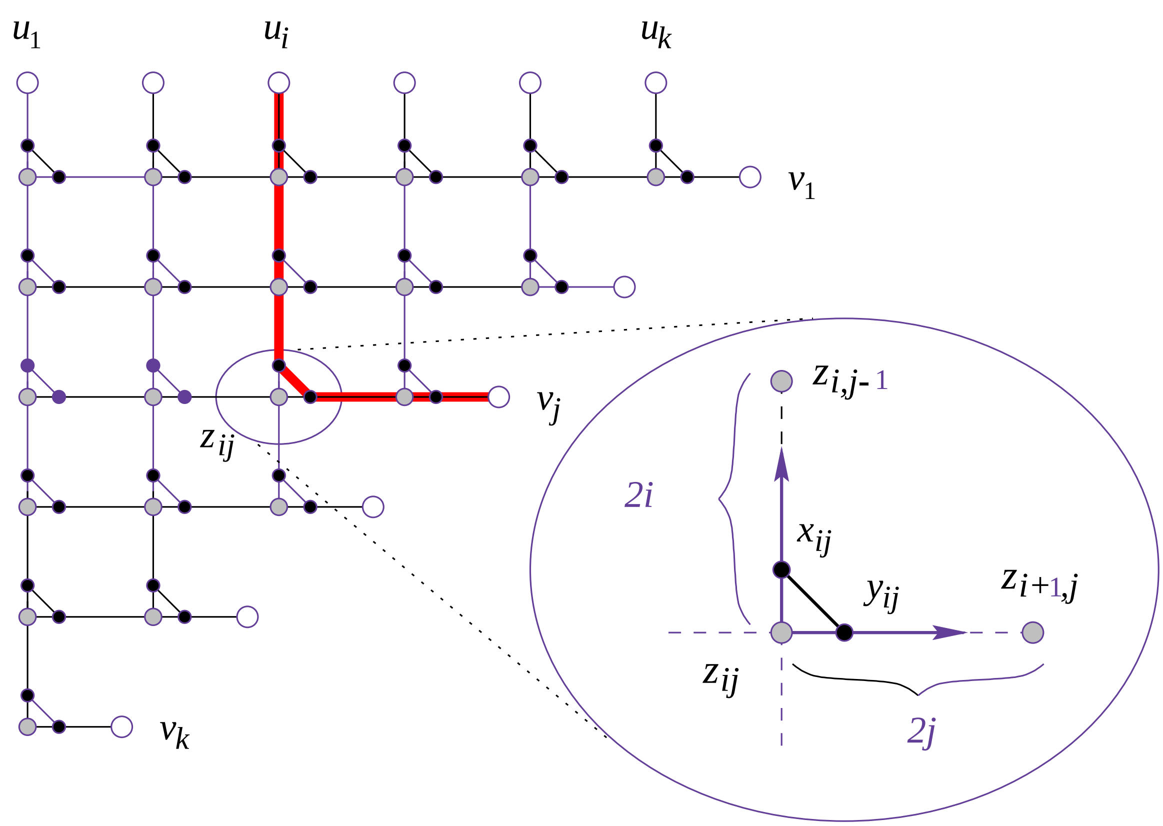
The family $\widehat{\mathcal{P}}_n$: Composed of all graphs resulting from G_k by removing some subset of diagonal edges $e_{i,j}$

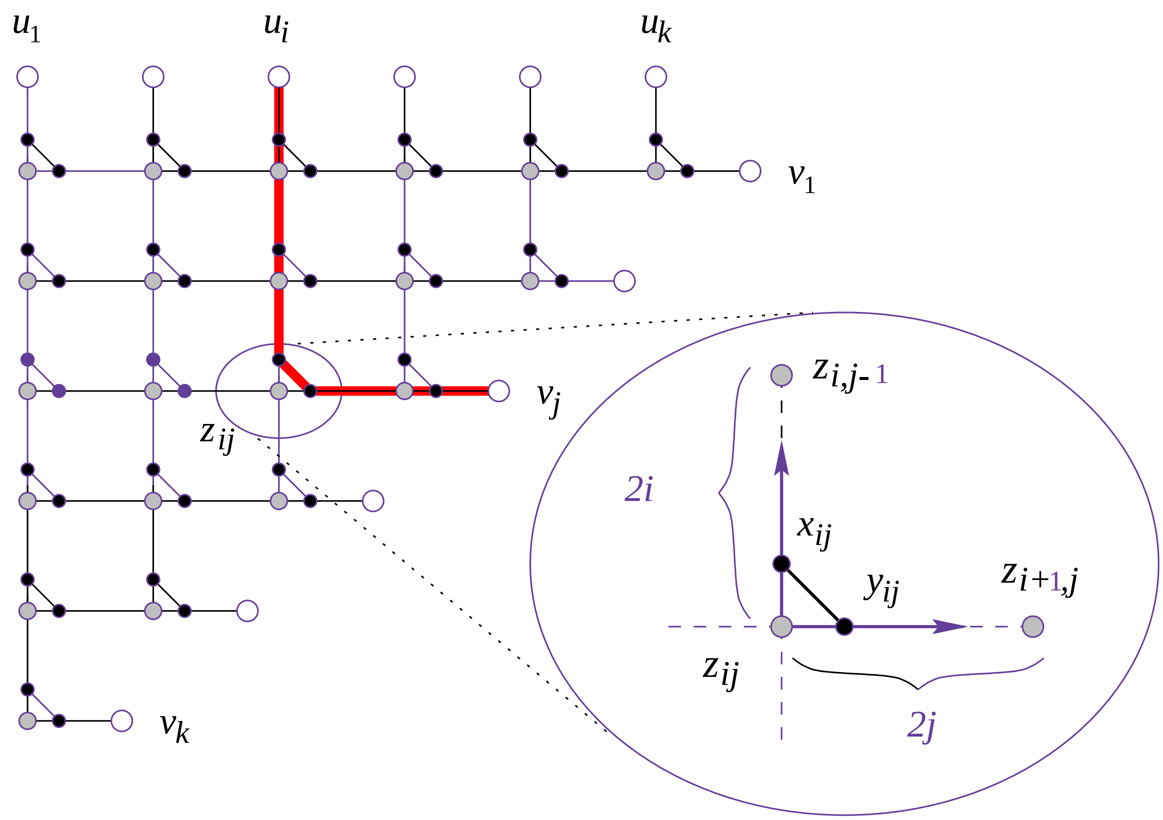
diagonal edges in $G_k = k(k+1)/2$

$\Rightarrow \quad |\widehat{\mathcal{P}}_n| = 2^{k(k+1)/2}$

Need to show that $\widehat{\mathcal{P}}_n$ is an A-family

Lemma: $\forall\ 2 \le i + j \le k + 1$,
the shortest path in G_k from u_i to v_j
is precisely the one highlighted in figure

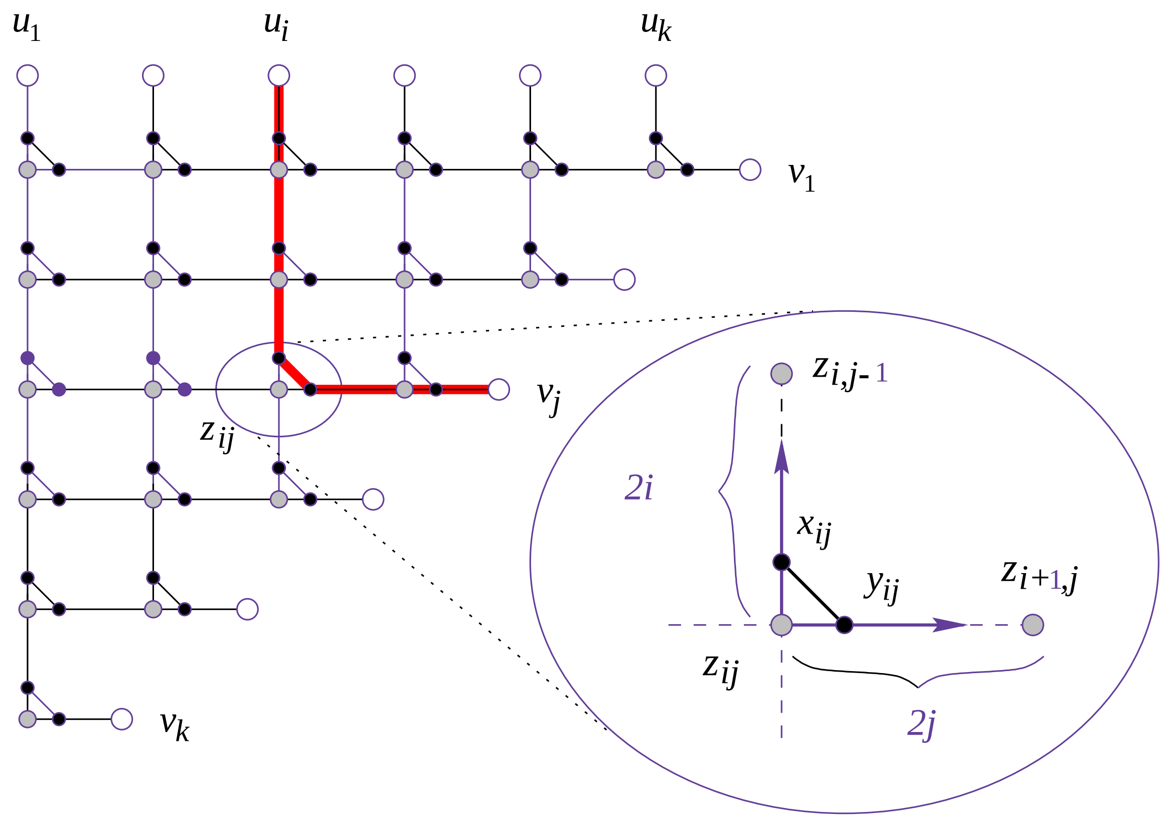




Implication 1:

Any shortest path $p(u_i, v_j)$ from u_i to v_j must use diagonal edge $e_{i,j}$

$\Rightarrow$ Removing this edge from the graph increases $dist_G(u_i, v_j)$ by at least 1



Implication 2:
Path $p(u_i, v_j)$ uses no *other* diagonal edge $e_{i',j'}$

$\Rightarrow$ $dist_G(u_i, v_j)$ depends only on whether $e_{i,j}$ exists or not

Implication 3:
For $G, H \in \widehat{\mathcal{P}}_n$ differing by diagonal edge $e_{i,j}$, $dist_G(u_i, v_j) \neq dist_H(u_i, v_j)$

Hence $\widehat{\mathcal{P}}_n$ is an A-family. ∎

Application of Main Thm

$\widehat{\mathcal{P}}_n = A$-family

$\Rightarrow$ By Main Thm,

$$\mathcal{L}(dist, \widehat{\mathcal{P}}_n) \;\geq\; \frac{\log|\widehat{\mathcal{P}}_n|}{|A|} \;=\; \frac{k(k-1)}{2|A|}$$

Here $\qquad |A| = 2k, \qquad n = \Theta(k^3)$

thus $\mathcal{L}(dist, \widehat{\mathcal{P}}_n) \;=\; \Omega(n^{1/3}) \quad$ ∎

Dodging the size lower bound

Question: Can *shorter* labels be used
to yield *approximate*
distance estimates?

Approximate distance labeling
[P,99]

R-approximate-distance labeling scheme:

[for family $\mathcal{F}$]

Decoder provides estimate $\tilde{D}(u,v)$ for distance between u,v given their labels, s.t.

$$\frac{1}{R} \cdot dist(u,v,G) \leq \tilde{D}(u,v) \leq R \cdot dist(u,v,G)$$

Results *[P,99]***:**

Diameter $= Diam(G) \ = \ \max_{u,v \in V}(dist(u,v,G))$

Denote $\Lambda = \log Diam(G)$

For n-node weighted graphs, $\kappa \geq 1$:

$O(\sqrt{\kappa})$-approx-distance labeling scheme of label size $O(\text{polylog } n \cdot \kappa \cdot n^{1/\kappa})$

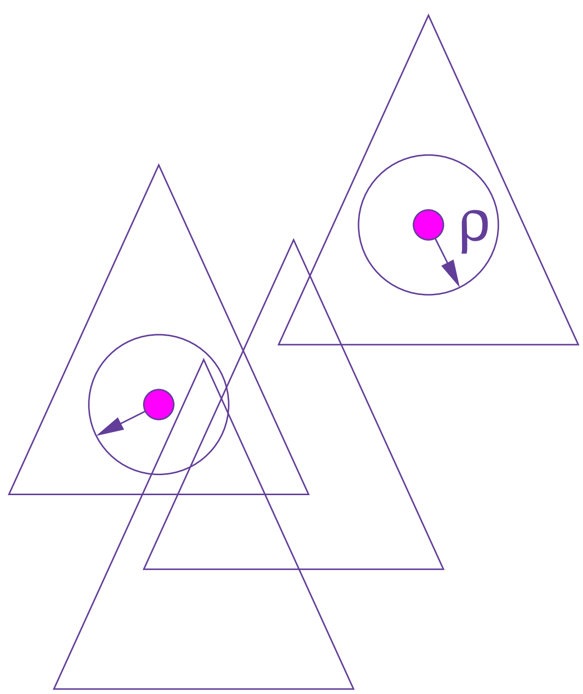
In particular: /* setting $\kappa = \log n/8$ */

$O(\sqrt{\log n})$-approx-distance labeling scheme of label size $O(\text{polylog } n)$

Distance-approx labelings

ρ-neighborhood of v:
Set of nodes at distance $\leq \rho$ from v
$\Gamma_\rho(v) = \{w \mid dist(w, v, G) \leq \rho\}$

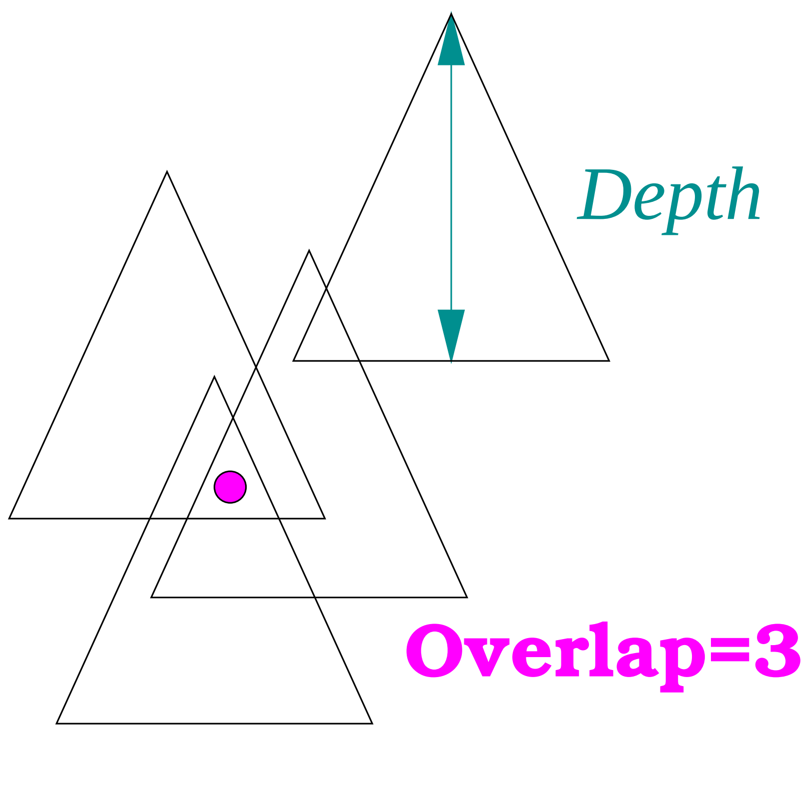
ρ-tree cover: Set $\mathcal{TC}$ of trees in G s.t.
$\forall$ node v $\exists$ tree $T \in \mathcal{TC}$
spanning v's entire ρ-neighborhood
$\Gamma_\rho(v) \subseteq V(T)$



Tree cover measures

$$Depth(\mathcal{TC}) = \max_{T \in \mathcal{TC}} \{Depth(T)\}$$

$$\mathtt{Overlap}(\mathcal{TC}) = \max_{v \in V} |\{T \in \mathcal{TC} \mid v \in V(T)\}|$$



Extremal tree cover constructions

Global tree cover:

Single BFS tree T

Depth: high, $Depth(\mathcal{TC}) = Diam(G)$

Overlap: low, $\mathtt{Overlap}(\mathcal{TC}) = 1$

Neighborhood tree cover:

n trees: $T(v)$ = spanning tree for $\Gamma_\rho(v)$

Depth: low, $Depth(\mathcal{TC}) = \rho$

Overlap: potentially high

Tree covers tradeoff

Theorem: *[Awerbuch,Kutten,P,91]*

$\forall$ $G = (V, E, \omega)$, $|V| = n$, and integers $\kappa, \rho \geq 1$,

$\exists$ ρ-tree cover $\mathcal{TC} = \mathcal{TC}_{\kappa,\rho}$ for G, with

$Depth(\mathcal{TC}) \leq (2\kappa - 1)\rho$ and

$\texttt{Overlap}(\mathcal{TC}) \leq \lceil 2\kappa \cdot n^{1/\kappa} \rceil$

Distance approximation using tree covers

The marker algorithm:

1. $\forall\ \ 1 \leq i \leq \Lambda$, construct 2^i-tree-cover

$$\mathcal{TC}_i = \mathcal{TC}_{\kappa,2^i}$$

   for G as in Theorem

2. $\forall$ tree cover $\mathcal{TC}_i$, assign distinct *tag* to each tree in it

3. $Tags_i(v) \leftarrow$ tags of all $\mathcal{TC}_i$ trees spanning v

4. Label each node v

$$Label(v) = (Tags_1(v), \ldots, Tags_\Lambda(v))$$

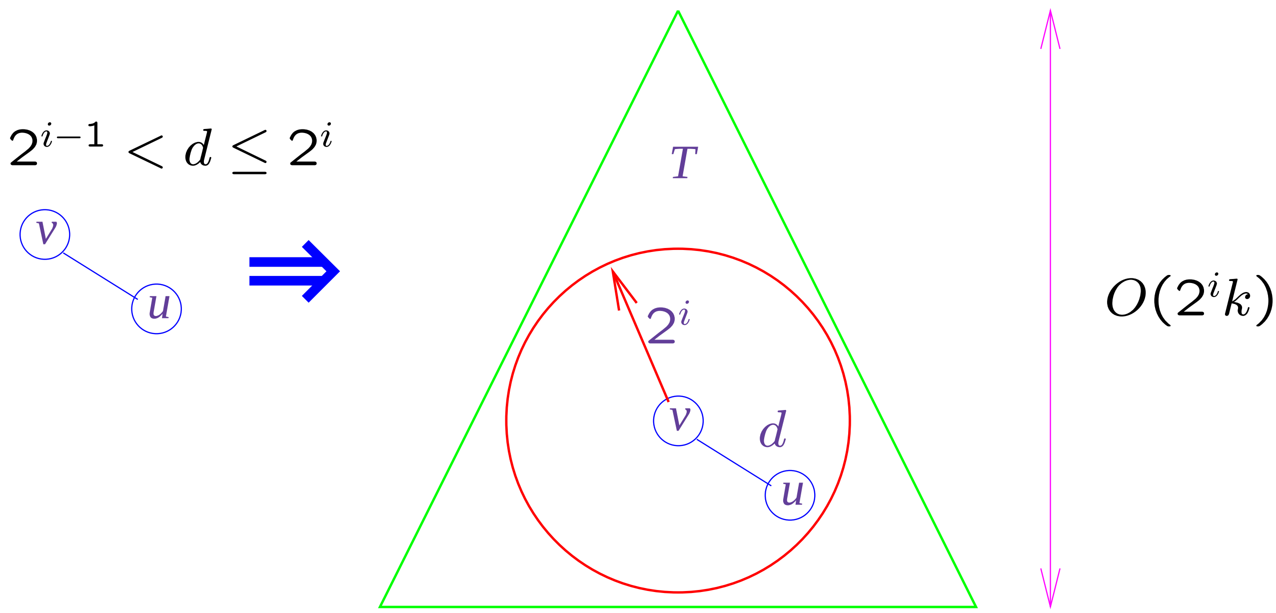
The distance estimation algorithm

Given $Label(v)$ **and** $Label(u)$**:**

1. Find lowest level j s.t. $Tags_j(v)$ and $Tags_j(u)$ contain a common tag

2. Return $\tilde{D}(u,v) = \sqrt{2\kappa} \cdot 2^j$
   as dist estimate for $dist(u,v,G)$

Analysis



T in 2^i Tree Cover

$\Downarrow$

T contains both u and v

$\Downarrow$

T's tag $\in Tags_i(v) \cap Tags_i(u)$

Lemma: $\forall \;\; G, u, v,$ estimate satisfies

$$\frac{1}{R} \cdot dist(u,v,G) \leq \tilde{D}(u,v) \leq R \cdot dist(u,v,G)$$

for $R = \sqrt{8\kappa}$

Proof: Consider u, v s.t. $2^{i-1} < dist(u,v,G) \leq 2^i$
Algorithm's estimate $= \tilde{D}(u,v) = \sqrt{2\kappa} \cdot 2^j$
$j =$ lowest level on which u and v share common tree T' in $\mathcal{TC}_j$

Lower bound: Tree cover $\mathcal{TC}_i = \mathcal{TC}_{\kappa,2^i}$ contains tree T s.t. $\Gamma_{2^i}(u) \subseteq V(T)$

$\Rightarrow \quad u, v \in V(T) \quad \Rightarrow \quad j \leq i$

$\Rightarrow \quad dist(u,v,G) \geq 2^{i-1} \geq 2^{j-1} =$

$$= \frac{1}{\sqrt{8\kappa}} \cdot (\sqrt{2\kappa} \cdot 2^j) = \frac{1}{R} \cdot \tilde{D}(u,v)$$

Upper bound: $Depth(T') \leq (2\kappa - 1) \cdot 2^j$

$\Rightarrow$ T has $u - v$ path of length $\leq 2 \cdot (2\kappa - 1) \cdot 2^j$

$$\Rightarrow dist(u,v,G) \;\leq\; 4\kappa \cdot 2^j \;=\; \sqrt{8\kappa} \cdot (\sqrt{2\kappa} \cdot 2^j) \;=\; R \cdot \tilde{D}(u,v)$$

Lemma: Label length is $O(\Lambda \log n \cdot \kappa \cdot n^{1/\kappa})$

Proof: Each tag requires $O(\log n)$ bits.
Each v occurs on $\leq \texttt{Overlap}(\mathcal{TC}_i) \leq \lceil 2\kappa \cdot n^{1/\kappa} \rceil$ different trees in $\mathcal{TC}_i$
$\Rightarrow$ ith tuple $Tags_i(v)$ contains $\leq \lceil 2\kappa \cdot n^{1/\kappa} \rceil$ tags

Theorem: $\forall\ \kappa \geq 1$:

$\exists\ (O(\Lambda \log n \cdot \kappa \cdot n^{1/\kappa}), \sqrt{8\kappa})$ approx-distance labeling scheme for class of all graphs,

$\exists\ (O(\log^2 n \cdot \kappa \cdot n^{1/\kappa}), \sqrt{8\kappa})$ approx-distance labeling scheme for class of all *unweighted* graphs.

Setting $\kappa = \log n / 8$

Cor:
$\exists\ (O(\Lambda \log^2 n), \sqrt{\log n})$ approx-distance labeling scheme for class of all graphs,

$\exists\ (O(\log^3 n), \sqrt{\log n})$ approx-distance labeling scheme for class of all *unweighted* graphs.

More on approx distance

Theorem *[Thorup,Zwick,01]*:
$\forall\, \kappa \geq 1$, $\exists\, \sqrt{2\kappa - 1}$-approx-distance labeling scheme
for the class of all graphs
using $O(\log(nD) \cdot \log^{1-1/\kappa} n \cdot n^{1/\kappa})$ bit labels

Theorem *[Gavoille,Katz,Katz,Paul,P,01]*:
$\exists\, (1 + 1/\log n)$-approx-distance labeling scheme
for the class of trees
using $O(\log n \cdot \log\log n)$ bit labels

Lower bounds for R-approx distance:

1. $\exists$ graphs requiring $\Omega(n)$ bit labels for $R < \sqrt{3}$ *[GPPR,99]*
2. $\exists$ graphs requiring $\Omega(n^{1/\kappa})$ bit labels for $R = \Omega(\kappa)$ *[TZ,01]*, *[GKKPP,01]*
3. $\forall\ \ k \leq n/2$, $\exists\ \ $ *k-separator* graphs requiring $\Omega(k)$ bit labels for $R < \sqrt{3}$ *[GPPR,99]*
4. $\exists$ trees requiring $\Omega(\log n \cdot \log\log n)$ bit labels for $R = 1 + 1/\log n$ *[GKKPP,01]*

Additive approx distance

(α,β)-approximate-distance labeling scheme: Decoder provides estimate $\tilde{D}(u,v)$ for distance between u,v given their labels, s.t.

$$dist(u,v,G) \le \tilde{D}(u,v) \le \alpha \cdot dist(u,v,G) + \beta$$

Multiplicative approximation scheme: $\beta = 0$

Additive approximation scheme: $\alpha = 1$

Exact scheme: $\alpha = 1$ *and* $\beta = 0$

Observation: Moving from $(1,0)$-approximate to $(1+o(1),0)$-approximate or $(1,O(1))$-approximate schemes impacts label size.

Graphs with $r(n)$-separator support $(1+1/\log n, 0)$-approximate scheme with $O(R(n) \cdot \log\log n)$ bit labels *[GKKPP,01]*

Other graphs
[GKKPP,01]

Planar: $(3, 0)$-approx, $O(n^{1/3} \log n)$ bits

Interval: $(1, 1)$-approx, $O(\log n)$ bits

Permutation & AT-free: $(1, 2)$-approx, $O(\log n)$ bits

c-chordal (longest induced cycle $\leq c$): $(1, \lfloor c/2 \rfloor)$-approx, $O(\log^2 n)$ bits $\Rightarrow$

Chordal: 1. $(1, 1)$-approx, $O(\log^2 n)$ bits
2. Exact ($(1, 0)$-approx) scheme requires $\Omega(n)$ bits

Open: Exact label size complexity of distance labeling scheme of interval and permutation graphs (Known: $\Omega(\log n)$, $O(\log^2 n)$ *[Katz,Katz,P,00]*)

Discussion: Alternative approach to distance labeling

Proximity-preserving labeling schemes based on low-distortion embeddings of grneral metrics in low-dimensional Euclidean spaces *[Bourgain,85,Linial,London,Rabinovich,95]*

Can be used to derive labeling systems with properties similar to ours *[Hassin,99]*

Limitations:

- Approximation ratio for general graphs $\Omega(\log n)$
  (Here: any approximation ratio $\kappa \leq \log n$)

- Embedding algorithm is *randomized*;
  Can probably be derandomized but likely to yield random-looking labels, revealing little or nothing on network structure.
  (Here: More direct, resulting labels capture information on network topology)

Question: What about *other* types of information beyond adjacency and distance?

Example information types

In trees:

- Ancestry
- Separation level
- Center
- Least common ancestor

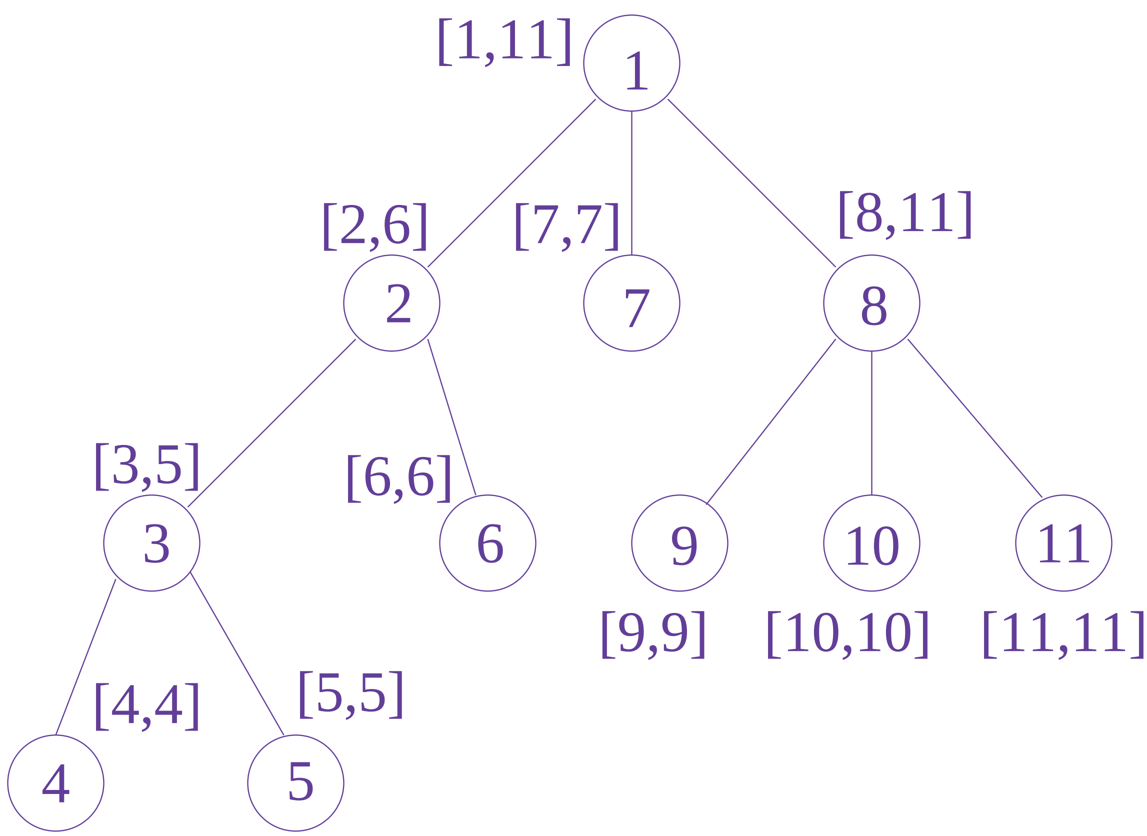
In general graphs:

- Steiner tree weight
- flow
- Edge/node connectivity

Ancestry labeling schemes
[Santoro,Khatib,85]

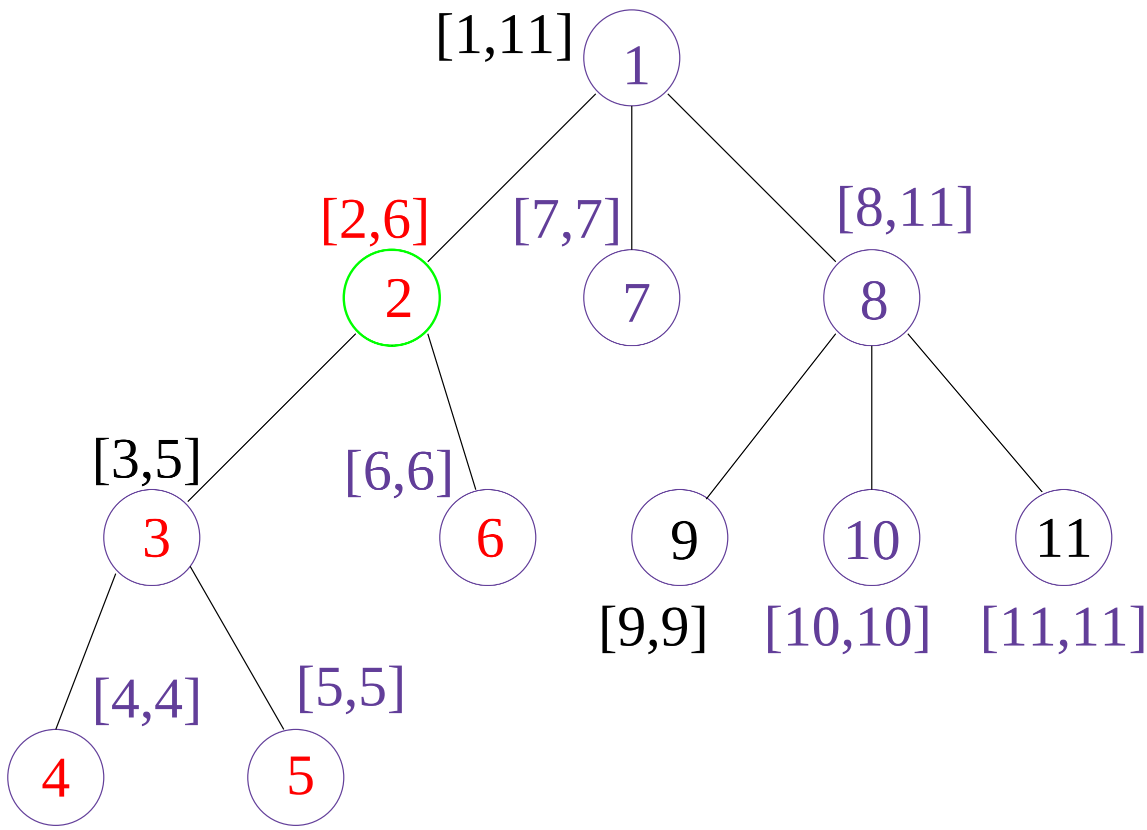
Marker algorithm: Assign each v *interval label* $Int(v)$ satisfying

Inclusion property: $\forall\ u, v$
$Int(v) \subseteq Int(u) \Leftrightarrow v$ is a descendent of u in T



Decoder algorithm: Use inclusion property.

Interval label selection



1. By depth-first tour of T, starting at root, assign each $u \in T$ a depth-first number $DFS(u)$.

2. Label u by interval $[DFS(u), DFS(w)]$, where w = last descendent of u visited by DFS (Labels contain $\leq \lceil 2 \log n \rceil$ bits)

Ancestry labeling - Improvements

1. Ancestor scheme with $\log n + O(\sqrt{\log n})$ bit labels *[TZ,01]*,*[Kaplan,Milo,01]*

2. Combined parent and ancestor scheme with $2\log n + O(\log\log n)$ bit labels *[KM,01]*

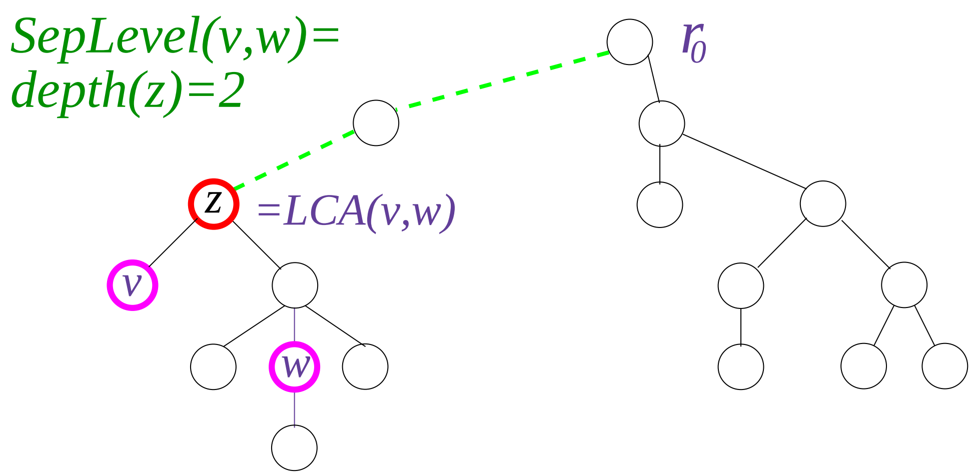
Application: Use ancestor labeling schemes on trees to optimize XML search engine queries on large database *[Abiteboul,Kaplan,Milo,01]*

SEP labeling schemes [P,00]

Rooted tree T, root r_0.

$depth(v)$ = distance from r_0

Separation level: $SEP(v,w) = \delta$ if their least common ancestor z has $depth(z) = \delta$.



$\mathcal{T}(n)$ = class of n-node trees

Lemma:

1. $\mathcal{L}(SEP, \mathcal{T}(n)) \leq \mathcal{L}(dist, \mathcal{T}(n)) + \log n$

2. $\mathcal{L}(dist, \mathcal{T}(n)) \leq \mathcal{L}(SEP, \mathcal{T}(n)) + \log n$

Proof of lemma

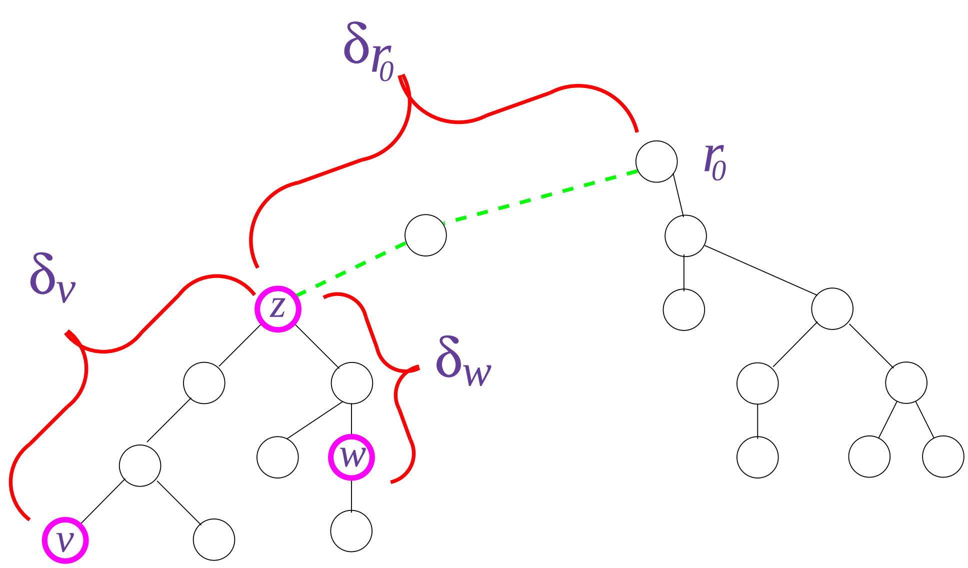
Given distance labeling scheme for trees:

$Label$ = labeling assigned for tree T.

SEP **marker:** Augment each label $Label(v)$ with $depth(v)$ (additional $\log n$ bits)

SEP **decoding:** Consider v, w with $z = LCA(v, w)$.

$\delta_v = dist(z, v)$,
$\delta_w = dist(z, w)$,
$\delta_{r_0} = dist(z, r_0) = depth(z)$

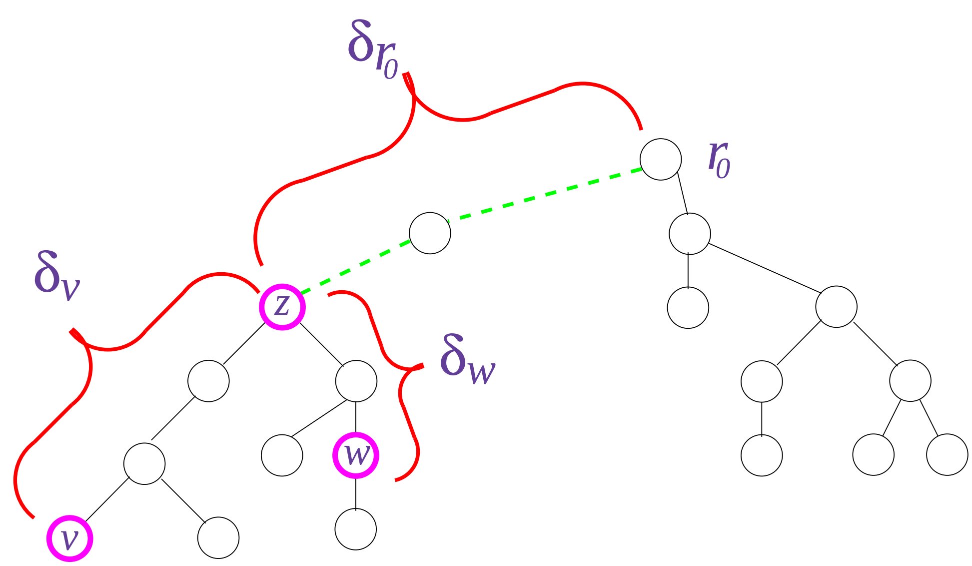


Given labels

$Label'(v) = \langle Label(v), depth(v) \rangle$,

$Label'(w) = \langle Label(w), depth(w) \rangle$,

1. Deduce $dist(v, w) = \delta_v + \delta_w$,
2. Retrieve $depth(v) = dist(v, r_0) = \delta_v + \delta_{r_0}$ and $depth(w) = dist(w, r_0) = \delta_w + \delta_{r_0}$.
3. Deduce $depth(z) = \delta_{r_0}$.



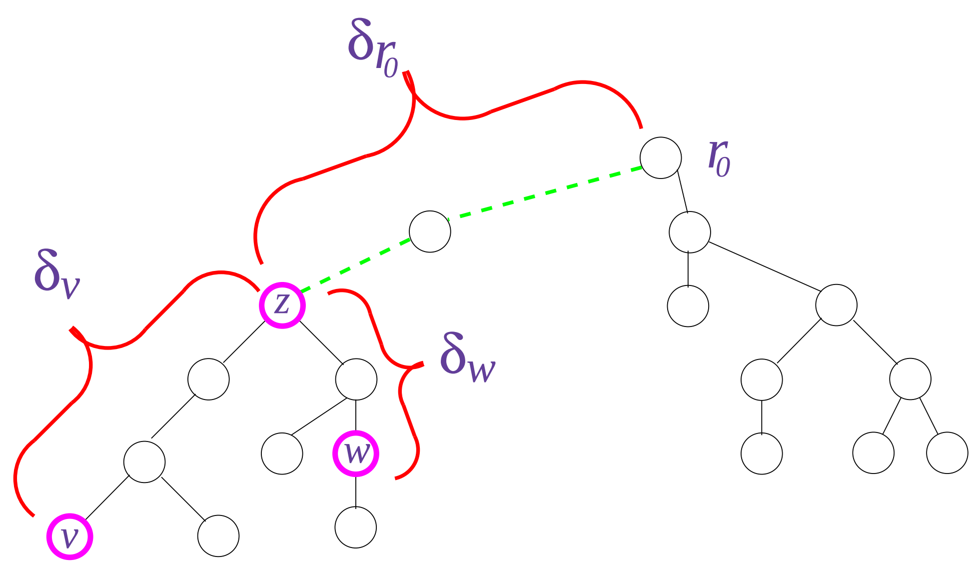
Proof - opposite direction

Given SEP labeling scheme for trees:

Distance marker: Augment each label $Label(v)$ with $depth(v)$ (additional $\log n$ bits)

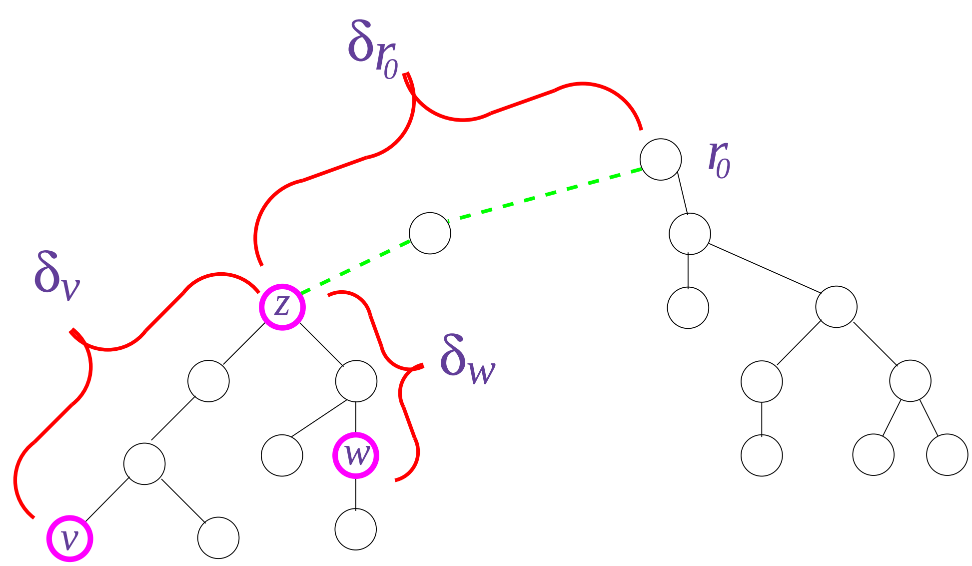
Distance decoding: Consider $v, w,\ z = LCA(v, w)$

$\delta_v = dist(z, v)$,
$\delta_w = dist(z, w)$,
$\delta_{r_0} = dist(z, r_0) = depth(z)$



Given $Label'(v) = \langle Label(v), depth(v) \rangle$,
$Label'(w) = \langle Label(w), depth(w) \rangle$,

1. Deduce δ_{r_0} from $Label(v)$ and $Label(w)$,
2. Deduce $depth(z)$ from $Label(v)$ and $Label(w)$,
3. Retrieve $depth(v) = dist(v, r_0) = \delta_v + \delta_{r_0}$ and $depth(w) = dist(w, r_0) = \delta_w + \delta_{r_0}$ from additional fields
4. Deduce δ_v and δ_w, hence also $\delta_v + \delta_w = dist(v, w)$.



Cor:

- $\exists$ SEP labeling scheme for $\mathcal{T}(n)$ using $O(\log^2 n)$ bit labels;

- any SEP labeling scheme for $\mathcal{T}(n)$ requires $\Omega(\log^2 n)$ bit labels.

Thm:

$$\mathcal{L}(SEP, \mathcal{T}(n)) = \Theta(\log^2 n)$$

LCA labeling schemes
[P,00]

Assumption: each node u has unique $O(\log n)$-bit *identifier* $ID(u)$

$LCA(u, v)$ = Least common ancestor of u and v

LCA labeling scheme:
Given labels $Label(v), Label(w)$,
compute *least common ancestor* $z = LCA(v, w)$,
return $ID(z)$

Thm *[P,00]***:**
1. $\exists$ LCA labeling scheme with $O(\log^2 n)$-bit labels for trees

Also, by reduction from SEP:

2. $\forall$ LCA labeling scheme for trees requires some $\Omega(\log^2 n)$-bit labels

Thm: If only required to return $Label(LCA(v, w))$ then $\exists$ scheme with $O(\log n)$ bit labels *[Alstrup,Gavoille,Kaplan,Rauhe,01]*

Definitions

$T(v)$ = subtree of T rooted at v
$\gamma_i(v)$ = v's ancestor at level i of T
(E.g. $\gamma_0(v) = r_0$, $\quad \gamma_{depth(v)}(v) = v$)

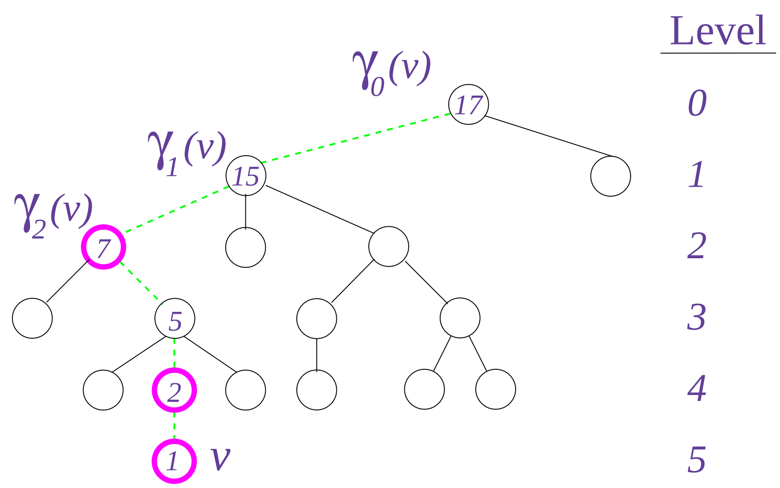
Def: v is *small* if $|T(v)| \leq |T(parent(v))|/2$

"Small ancestor" levels of v:

$$SAL(v) \ = \ \{i \mid 1 \leq i \leq depth(v),\ \gamma_i(v) \text{ is small}\}$$

Small ancestors of v:

$$SA(v) \ = \ \{\gamma_i(v) \mid i \in SAL(v)\}$$

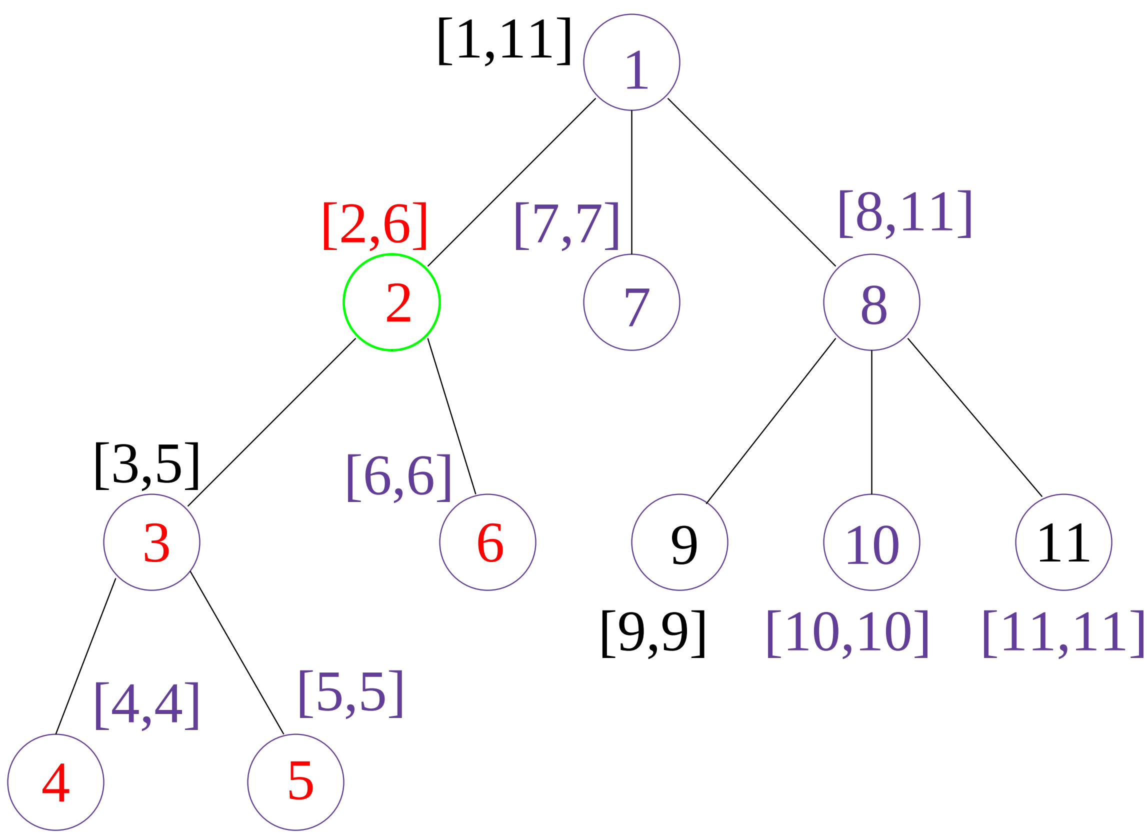


In example: $SAL(v) = \{2, 4, 5\}$

The LCA-marker

Preprocessing: Assign each v *identifier* $ID(v)$ and *interval* $Int(v)$ as in *interval labeling* of *[Santoro,Khatib,85]*

Inclusion property: $\forall\ u, v,$
$Int(v) \subseteq Int(u) \Leftrightarrow v$ is a descendent of u in T



i-triple of v: Identifiers of v's ancestors on levels $i-1,\ i,\ i+1$,

$$\begin{aligned} Q_i(v) \;=\; \langle\;\; & \langle i-1, ID(\gamma_{i-1}(v))\rangle\ , \\ & \langle i, ID(\gamma_i(v))\rangle\ , \\ & \langle i+1, ID(\gamma_{i+1}(v))\rangle\ \rangle \end{aligned}$$

LCA labels:

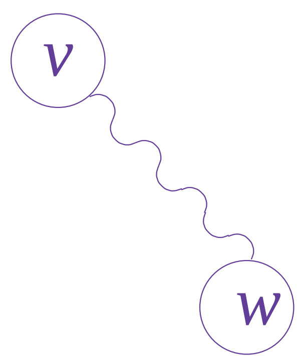
Identifier
+ interval
+ i-triples of all small ancestor levels

$$\begin{aligned} Label(v) \;=\; \langle\;\; & ID(v)\ , \\ & Int(v)\ , \\ & \{Q_i(v) \mid 1 \le i < depth(v), \\ & \qquad\quad i \in SAL(v)\}\ \rangle \end{aligned}$$

The LCA-decoder $\mathcal{D}_{LCA}$

Given labels $Label(v)$ and $Label(w)$,
infer identifier $ID(z)$ of $z = LCA(v, w)$

1. /* $v =$ ancestor of w */
   If $Int(w) \subseteq Int(v)$ then return $ID(v)$



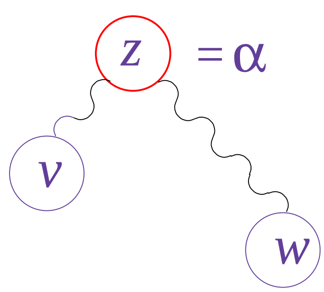
2. /* $w =$ ancestor of v */
   If $Int(v) \subseteq Int(w)$ then return $ID(w)$

3. /* w, v unrelated */
Extract from $Label(v)$ and $Label(w)$ the sets $SAL(v), SAL(w), SA(v), SA(w)$

α = highest level node in $SA(v) \cap SA(w)$
/* = least_common_*small*_ ancestor(v, w) */
K = its level /* i.e. $\alpha = \gamma_K(v) = \gamma_K(w)$ */

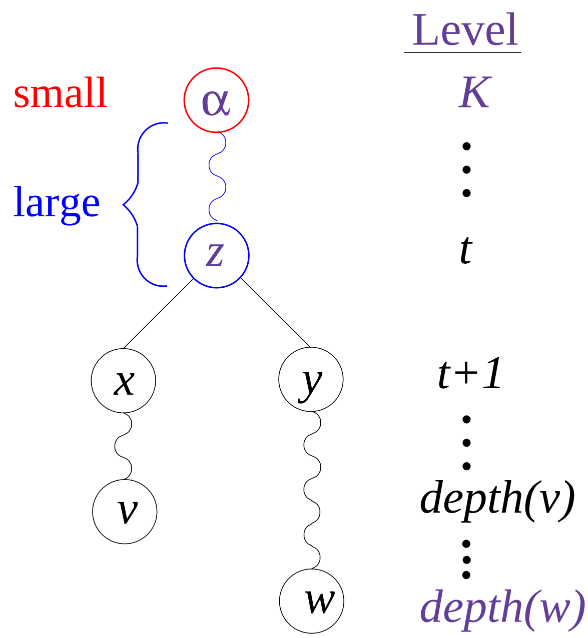
4. /* α is $LCA(v, w)$ */
If $\gamma_{K+1}(v) \neq \gamma_{K+1}(w)$ then return $ID(\alpha)$

Level



t=K

5. /* α above $z = LCA(v, w)$ (z large) */



/* Fact: Each node has ≤ 1 large child
$\Rightarrow$ either x or y is small */

$i_v \leftarrow \min\{i \in SAL(v) \mid i > K\}$
$i_w \leftarrow \min\{i \in SAL(w) \mid i > K\}$
$i_m \leftarrow \min\{i_v, i_w\}$

/* $\Rightarrow$ $i_m = t + 1$ */

6. If $i_v \leq i_w$: get $\gamma_{i_m - 1}(v)$ from i_m-triple $Q_{i_m}(v)$
Else: get $\gamma_{i_m - 1}(w)$ from i_m-triple $Q_{i_m}(w)$
Return the extracted identifier.

Label size analysis

Lemma: For $T \in \mathcal{T}(n)$, every node v has $\leq \log n$ small ancestors

$\Rightarrow$ $\forall v$, $Q_i(v)$ has $\leq \log n$ i-triples

Thm: $\langle \mathcal{D}_{LCA}, \mathcal{M}_{LCA} \rangle$ is an LCA labeling scheme with $O(g(n) \log n)$-bit labels for n-node trees with identifiers of size $g(n)$. ∎

Using $\log n$-bit identifiers,

Cor: $\mathcal{L}(LCA, \mathcal{T}(n)) = O(\log^2 n)$

Lower bound

Lemma: If $\mathcal{T}(n)$ has an LCA labeling scheme with $l(n) \cdot g(n)$-bit labels over $g(n)$-bit identifiers, then it has a SEP labeling scheme with $l(n) \cdot (g(n) + \log n)$-bit labels.

Since $g(n) = \Omega(\log n)$

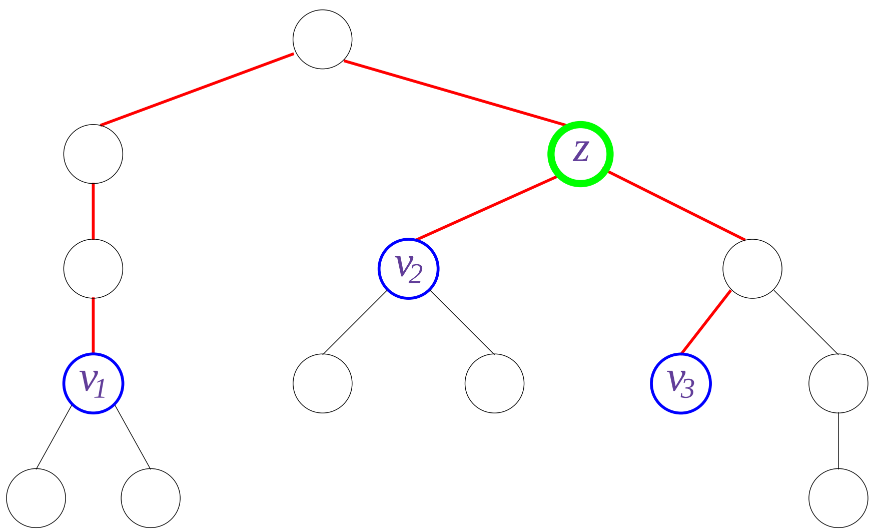
Cor: Any LCA labeling scheme for $\mathcal{T}(n)$ requires some $\Omega(\log^2 n)$-bit labels.

Thm: $\mathcal{L}(LCA, \mathcal{T}(n)) = \Theta(\log^2 n)$

$Center$ labeling schemes
[P,00]

For nodes v_1, v_2, v_3 in tree T,

$Center(v_1, v_2, v_3)$ = unique node z s.t. paths connecting z to v_1, v_2 and v_3 are edge-disjoint



Claim: LCA-marker serves also as $Center$-marker, provided the identifiers are also ancestry and depth labelings:

- $ID(v)$ contains $depth(v)$
- two identifiers $ID(v)$ and $ID(w)$ allow deducing ancestry

(both achievable using $O(\log n)$-size identifiers)

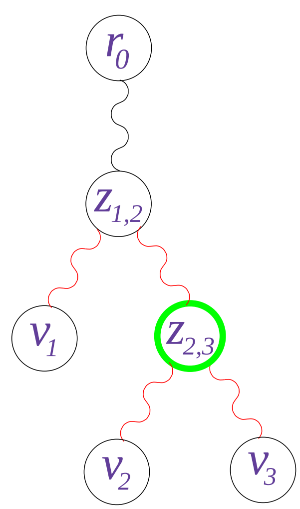
The $Center$-decoder $\mathcal{D}_{Center}$

Denote $z_{i,j} = LCA(v_i, v_j)$

1. Compute $ID(z_{1,2})$, $ID(z_{1,3})$, $ID(z_{2,3})$

2. If *three LCA's coincide* then return one (say, $ID(z_{1,2})$)

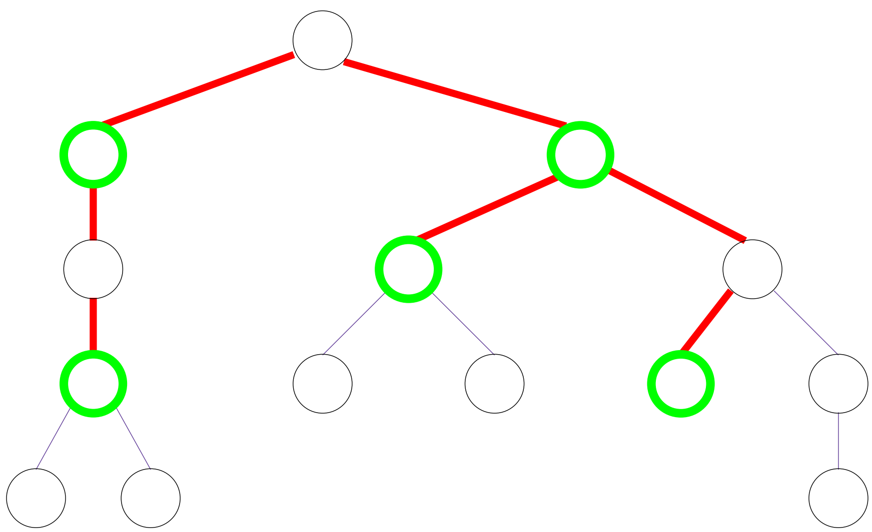
3. If *exactly two LCA's coincide*,
say, $z_{1,3} = z_{1,2}$,
then *return third*, $ID(z_{2,3})$



Thm: $\mathcal{L}(Center, \mathcal{T}(n)) = O(\log^2 n)$

Steiner labeling schemes
[P,00]

Steiner tree: For node set W in weighted G,
$T_S(W)$ = min-weight subtree spanning W
$Steiner(W)$ = its weight



Steiner **labeling scheme:** deduces $Steiner(W)$ given labels $\{Label(v) \mid v \in W\}$

$\mathcal{T}(n,\mu)$ = class of n-node trees with μ-bit edge weights

Thm: $\mathcal{L}(Steiner, \mathcal{T}(n,\mu)) = \Theta(\mu \log n + \log^2 n)$

$Steiner$ labeling schemes

Claim: $Center$-marker serves also as $Steiner$-marker

Facts used:

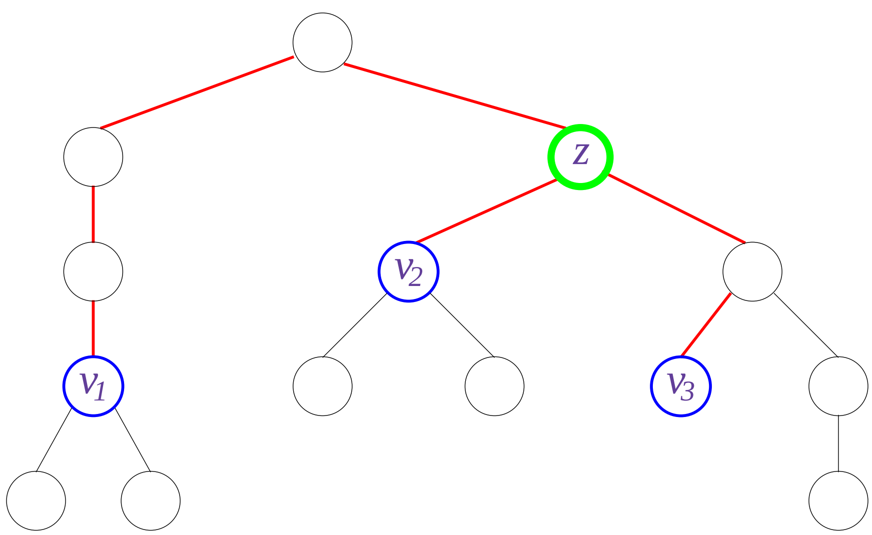
1. $Center$ scheme can deduce *distances* from $z = Center(v_1, v_2, v_3)$ to each v_i

2. In labelings of $Center$-marker, the identifiers $ID(v)$ provide $depth(v)$

$Steiner$ **decoder** $\mathcal{D}_{Steiner}$

$W = \{v_1, v_2, v_3\}$:

1. Deduce center $z = Center(v_1, v_2, v_3)$
2. Calculate distances $d_i = dist(v_i, z)$, $1 \leq i \leq 3$
3. Return $\omega(W) = d_1 + d_2 + d_3$

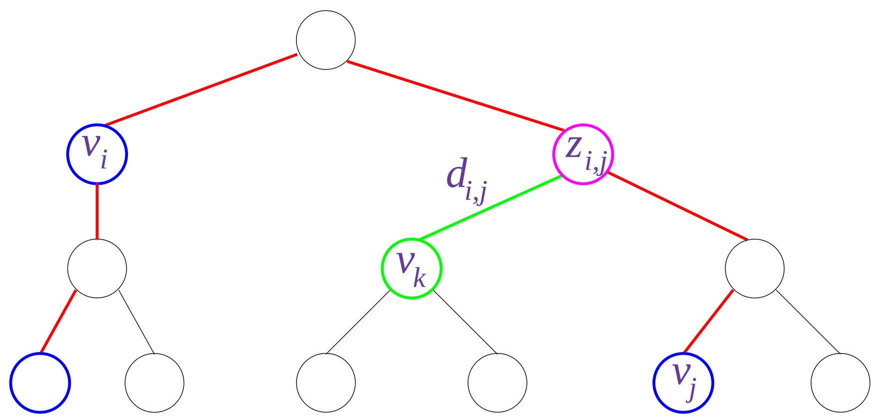


$W = \{v_1, \ldots, v_q\}$ **for** $q > 3$:

Denote $W_i = \{v_1, \ldots, v_i\}$

1. Calculate $\omega(W_3)$ as before

2. **For** $k = 4$ to q **do:** /* Add v_k */



   (a) For every $1 \leq i < j \leq k$ compute:
      - $z_{i,j} = Center(v_i, v_j, v_{k+1})$
      - $d_{i,j} = dist(z_{i,j}, v_{k+1})$

   (b) Let i', j' = pair minimizing $d_{i,j}$

   (c) Let $\omega(W_{k+1}) = \omega(W_k) + d_{i',j'}$

3. Return $\omega(W_q)$

Correctness proof

Def: For subtree T' and node v in T,
$\varphi(v, T')$ = unique shortest path from v to T'

Lemma: $\forall$ node set $W = \{v_1, \ldots, v_k\}$ and $v \notin W$,
$\exists\ v_i, v_j \in W$, connected by path $P_{i,j}$ in T,
s.t. $\varphi(v, T_S(W)) = \varphi(v, P_{i,j})$

Thm: $\mathcal{L}(Steiner, \mathcal{T}(n, \mu)) = \Theta(\mu \log n + \log^2 n)$

Approximate $Steiner$ labeling schemes for general graphs

$\mathcal{G}(n,\mu)$ = class of arbitrary n-node graphs with μ-bit edge weights.

Steiner and min-weight spanning trees:

For weighted $G(V,E,\omega)$ and node set W:

$G'(W,E',\omega')$ = complete weighted graph on W setting $\omega'(x,y) = dist(x,y,G)$ $\forall x,y \in W$.

$MST(W)$ = min-weight of spanning tree for G'

Lemma: *[Kou,Markowsky,Berman,81]*
$Steiner(W) \ \leq \ MST(W) \ \leq \ 2 \cdot Steiner(W)$

Approximate Decoder $\mathcal{D}'_{Steiner}$

Claim: Given R-approximate *distance* labeling scheme $\langle \mathcal{D}'_{dist}, \mathcal{M}'_{dist} \rangle$, same marker algorithm yields $2R$-approximate $Steiner$ labeling scheme

Idea: Given labels $\{Label(v) \mid v \in W\}$:
1. Calculate distance estimates in $G' = (W, E', \tilde{\omega})$,
2. Construct MST for G'
3. Return its weight

Lemma: $\mathcal{D}'_{Steiner}$ yields $2R$-approximation

Cor: If $\mathcal{G}(n, \mu)$ enjoys R-approximate *distance* labeling scheme, then it also enjoys $2R$-approximate $Steiner$ labeling scheme with same size labels

Lemma: *[P,00]*
$\exists\, 2\sqrt{2\log n}$-approximate *distance* labeling scheme for $\mathcal{G}(n, \mu)$ with $O((\mu + \log n)\log^2 n)$-size labels

Cor: $\mathcal{G}(n, \mu)$ enjoys $4\sqrt{2\log n}$-approximate $Steiner$ scheme with $O((\mu + \log n)\log^2 n)$-size labels

Flow & connectivity labeling schemes

[Katz,Katz,Korman,P,02]

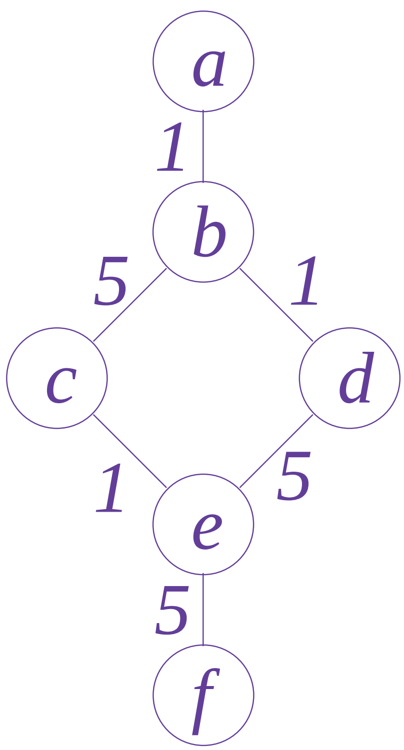
Weighted $G = \langle V, E, \omega \rangle$
$\omega(e)$ = (integral) *edge capacity*
(Assume infinite-capacity self-loops)

Represent each edge e of capacity $\omega(e)$
as $\omega(e)$ parallel edges of unit capacity

Maximum flow: In path $p = (e_1, \ldots, e_m)$:
$\texttt{flow}(p) \;=\; \min_{1 \le i \le m}\{\omega(e_i)\}$.

Max flow in set P of edge-disjoint paths:
$\texttt{flow}(P) \;=\; \Sigma_{p \in P}\texttt{flow}(p)$.

$\texttt{flow}(u,v) \;=\;$ max$\{\texttt{flow}(P) \mid P$ is a set of
edge-disjoint paths between $u, v\}$.



	a	b	c	d	e	f
a	∞	1	1	1	1	1
b		∞	6	2	2	2
c			∞	2	2	2
d				∞	6	5
e					∞	5
f						∞

Connectivity

Edge-connectivity: For nodes u, w:

$$\texttt{e-conn}(u,w) \;=\; \text{max flow between } u,w \text{ assuming } \forall e,\ \omega(e) = 1$$

node-connectivity: For nodes u, w:

$\texttt{v-conn}(u,w)$ = cardinality of largest set P of node-disjoint paths connecting u, w.

By Menger's theorem, for nonadjacent u, w,

$$\texttt{v-conn}(u,w) \;=\; \min \# \text{ nodes in } G \setminus \{u,w\} \text{ whose removal from } G \text{ (with incident edges) disconnects } u \text{ from } w.$$

Results

Flow:
$\exists$ flow labeling scheme with $O(\log n \cdot \log \omega)$ bit labels for general n-node graphs with maximum integral capacity $\hat{\omega}$, and this is tight

Edge-connectivity:
Tight bound of $\Theta(\log^2 n)$ bit labels

Node connectivity:
Label size depends on connectivity parameter:

1. Label sizes in k-node-connectivity labeling scheme for general n-node graphs:
   $\log n$ for $k = 1$
   $3 \log n$ for $k = 2$
   $5 \log n$ for $k = 3$
   $2^k \log n$ for $k > 3$

2. Lower bound of $\Omega(k \log n)$ on label size for $k = \text{polylog}\, n$.

Flow labeling schemes

Graph family: $\mathcal{G}(n, \widehat{\omega}) =$
capacitated n-node graphs
with maximum capacity $\widehat{\omega}$

Flow relations:
Given $G = \langle V, E, \omega \rangle$ and integer $1 \leq k \leq \widehat{\omega}$,

$$R_k \;=\; \{(x, y) \mid x, y \in V, \;\; \texttt{flow}(x, y) \geq k\}.$$

Lemma: R_k is an equivalence relation.

Proof:

∞-capacity self-loops $\Rightarrow R_k$ is reflexive

G undirected $\Rightarrow R_k$ is symmetric.

Transitivity follows by duality of flows and cuts:

Suppose (for contradiction) that $\exists$ nodes x, y, z s.t. $R_k(x, y)$, $R_k(y, z)$ hold, but $R_k(x, z)$ doesn't.

$R_k(x, z)$ does not hold
$\Downarrow$
$\texttt{flow}_G(x, z) \leq k - 1$
$\Downarrow$
$\exists$ cut $(S; \bar{S})$ of capacity $\leq k - 1$ in G
s.t. $x \in S$ and $z \in \bar{S}$.

W.l.o.g. $y \in S$.

Then same cut also implies $\texttt{flow}_G(y, z) \leq k - 1$, hence $R_k(y, z)$ is false too, contradiction. ∎

Equivalence class tree

$\forall k \geq 1$, R_k induces a collection

$$\mathcal{C}_k = \{C_k^1, \ldots, C_k^{m_k}\},$$

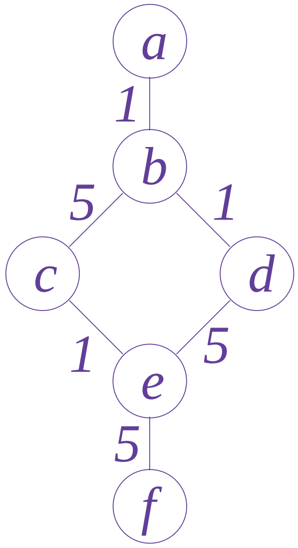
of equivalence classes on V, s.t.

$$C_k^i \cap C_k^j = \emptyset \quad and \quad \bigcup_i C_k^i = V$$

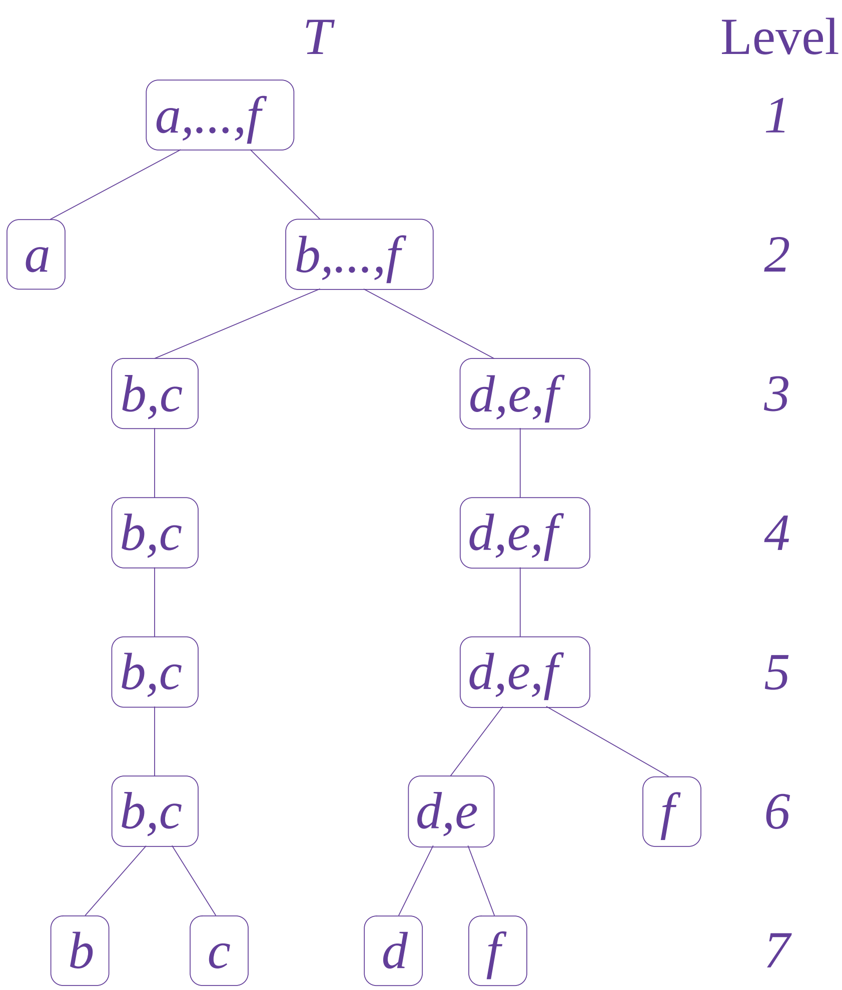
Note: For $k < k'$, $R_{k'}$ is a *refinement* of R_k, namely, $\forall$ class $C_{k'}^i$ $\exists$ class C_k^j s.t. $C_{k'}^i \subseteq C_k^j$.

Given G, construct a tree T_G corresponding to its equivalence relations:

- k'th level corresponds to relation R_k, i.e., has m_k nodes marked by classes $C_k^1, \ldots, C_k^{m_k}$.
- Root of T_G marked by $C_1^1 = V$ = unique equivalence class of R_1.
- T_G is truncated at a node once equivalence class associated with it is singleton.



	a	b	c	d	e	f
a	∞	1	1	1	1	1
b		∞	6	2	2	2
c			∞	2	2	2
d				∞	6	5
e					∞	5
f						∞



Separation level

For nodes x, y in tree T rooted at r,

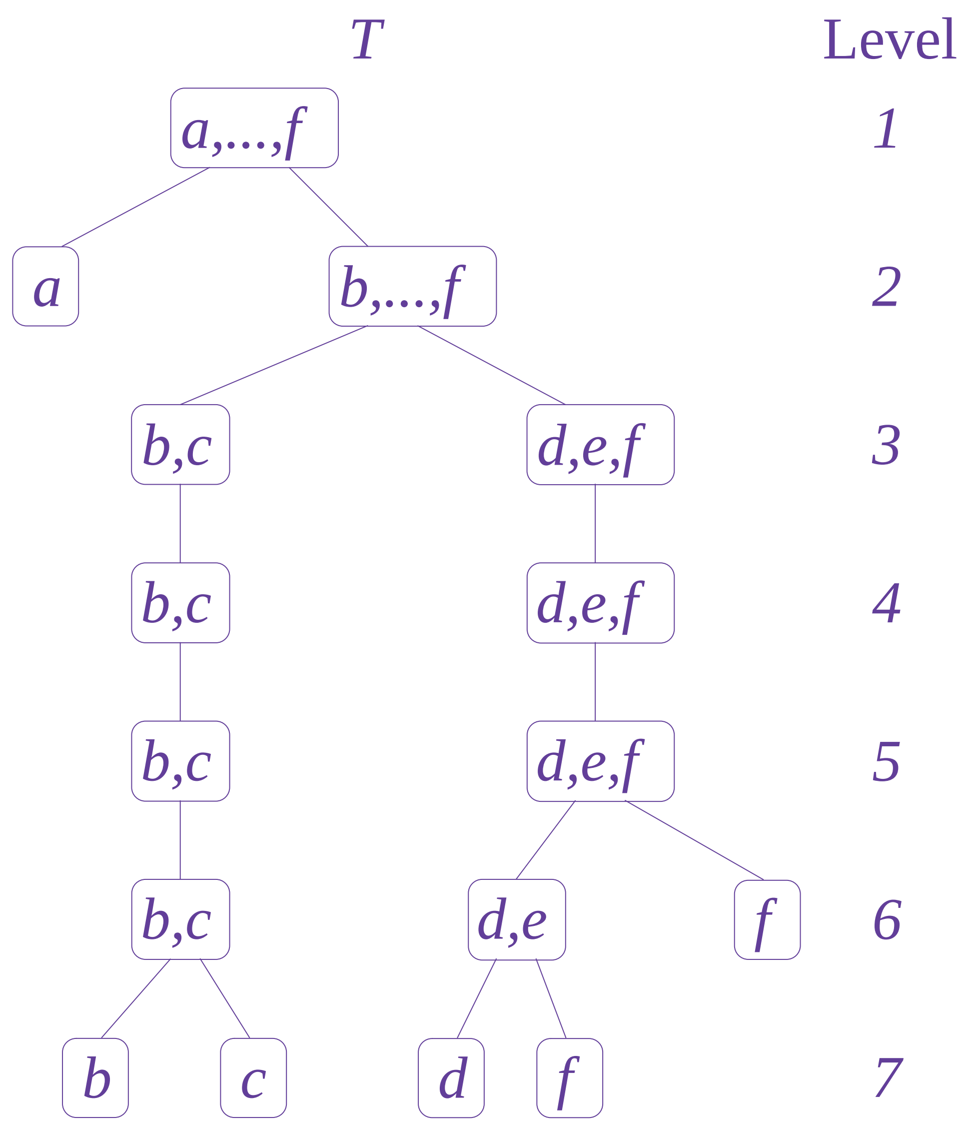
$SEP_T(x, y)$ = depth of least common ancestor of x and y, $z = LCA(x, y)$.

I.e., $SEP_T(x, y) = dist_T(z, r)$.

$\forall v \in G$, let $t(v)$ = leaf in equivalence class tree T_G associated with the singleton $\{v\}$.

Lemma: $\forall v, w \in V$,

$$\texttt{flow}_G(v, w) = SEP_T(t(v), t(w)) + 1.$$



Recall:
For class $\mathcal{T}(n)$ of n-node unweighted trees,
$\exists$ SEP labeling scheme with $O(\log^2 n)$-bit labels
(and this is tight)

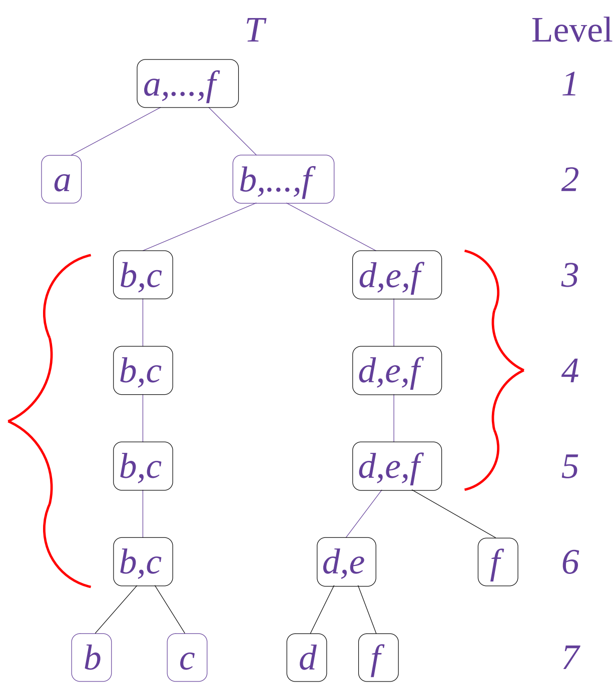
Observe:
If maximum capacity in n-node graph G is $\hat{\omega}$,
then depth of tree $T_G \leq n\hat{\omega}$ levels,
and it has $\leq n$ nodes per level,

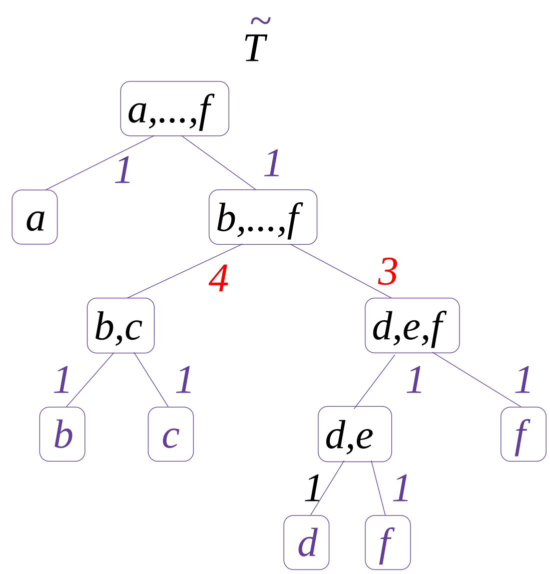
$\Rightarrow$ # nodes in $T_G = O(n^2\hat{\omega})$.

Cor: $\mathcal{L}(\texttt{flow}, \mathcal{G}(n, \hat{\omega})) = O(\log^2(n\hat{\omega}))$.

More careful design

- Compact *long chains*
- Get compacted *weighted* tree $\tilde{T}_G$





Analysis improvement

Define separation level for weighted rooted trees:
$SEP_T(x,y)$ = *weighted* depth of $LCA(x,y)$.

Observe: Bounds for SEP labeling schemes on unweighted trees - apply to weighted trees too:
For class $\mathcal{T}(\tilde{n},\tilde{\omega})$ of weighted $\tilde{n}$-node trees with maximum weight $\tilde{\omega}$, $\exists$ SEP schemes with $O(\log \tilde{n} \log \tilde{\omega} + \log^2 \tilde{n})$-bit labels.

Observe: Separation level of leaves $t(x), t(y)$ in $\tilde{T}_G$ - still related to $\mathtt{flow}_G(x,y)$ as in Lemma.

Observe: $\tilde{T}_G$ has n leaves,
every non-leaf node in $\tilde{T}_G$ has ≥ 2 children
$\Rightarrow$ # nodes in $\tilde{T}_G$ is $\tilde{n} \leq 2n-1$.
Max edge weight in $\tilde{T}_G$ is $\tilde{\omega} \leq \hat{\omega} \cdot n$.

Theorem:
$\mathcal{L}(\mathtt{flow}, \mathcal{G}(n,\hat{\omega})) = O(\log n \cdot \log \hat{\omega} + \log^2 n)$.

Let $\mathcal{G}(n)$ = class of n-node unweighted graphs

Cor: $\mathcal{L}(\mathtt{e\text{-}conn}, \mathcal{G}(n)) = O(\log^2 n)$.

Lower bound for flow

Tree class $\mathcal{T}(n, \widehat{\omega})$:
n-node balanced binary trees, max capacity $\widehat{\omega}$

Def: For nodes u, v in tree T,
$path(u, u)$ = unique path from u to v in T
$MaxE(u, v)$ = max edge weight on $path(u, v)$
$MinE(u, v)$ = min edge weight on $path(u, v)$.

Observe 1: On a tree, $\texttt{flow}(u, v) = MinE(u, v)$.

Observe 2: On trees, $MaxE$ and $MinE$ are *equivalent* w.r.t. labeling schemes.

Transformation: Given weighted tree T, let T' = weighted tree obtained by replacing weight $\omega(e)$, $\forall e$, with $\omega'(e) = \widehat{\omega} - \omega(e)$.

$\Rightarrow$ To get lower bound for *flow* labeling schemes on trees, suffices to prove it instead for $MaxE$.

Thm:

$$\mathcal{L}(MaxE, \mathcal{T}(n,\widehat{\omega})) \;\geq\; \frac{1}{2}(\log(n+1)\log(\widehat{\omega}+1) - \log(n+1)\log\log(n+1))$$

Proof idea: Modification of lower bound proof for distance labeling schemes.

Cor: For $\widehat{\omega} > \log(n+1) - 1$,

$$\mathcal{L}(\texttt{flow}, \mathcal{T}(n,\widehat{\omega})) = \Omega(\log n \log \widehat{\omega})$$

Cor: $\mathcal{L}(\texttt{e-conn}, \mathcal{G}(n)) = \Theta(\log^2 n)$.

Application 1: Connection setup
[P,99]

Model: circuit-switched communication network (represented as unweighted graph $G(V,E)$)

Virtual channel: $VC(x,y)$ = logical connection for transmitting all traffic between x and y

VC setup: By *connection setup* (CS) proc, in response to request by an endpoint

"Best-service" connection setup procedures

Satisfy two "Quality-of-service" properties:

(P1) Shortest channel: $VC(x, y)$ uses shortest (length $dist(x, y, G)$) route between endpoints x, y

Time measure: Time elapsing since request until $VC(x, y)$ is established

Assume: message crosses link in ≤ 1 time unit

(P2) Minimal setup time: $VC(x, y)$ is established in time $dist(x, y, G)$ once request is issued

Complexity measures for connection setup procedures

Memory:

M_{CS} = memory requirements per switch

Quiescence time:

$T_{CS}(x,y)$ = termination time of procedure CS

(Note: Procedure might continue operating *after* virtual channel $VC(x,y)$ is set up)

Two extreme approaches

Full tables (FT) method:
Store full routing tables in each node
Use to set-up virtual channel upon request

Quiescence time: $T_{FT}(x, y) = dist(x, y, G)$

Memory: $M_{FT} = \Theta(n \log n)$ bits per switch

Bellman-Ford (BF) method:
Search for shortest path for the virtual channel from scratch upon request, by applying shortest-path algorithm

Quiescence time: $T_{BF}(x, y) = Diam(G)$

Memory: M_{BF} is minimal

New label-based (LB) approach

1. Assign distance-approximating labels to nodes

2. Upon request for setting up $VC(x,y)$:
   Use BF-style flooding for connection setup
   but limit flooding to distance

$$\hat{d} = \tilde{D}(x,y) \cdot \sqrt{\log n}$$

   ($\tilde{D}(x,y)$ = estimate obtained from labels $Label(x)$ and $Label(y)$ to $dist(x,y,G)$)

Analysis

Quiescence time:
Shortest path is contained within range $\hat{d}$
(upper bounds actual distance)

$\hat{d}/\log n$ lower bounds actual distance
$\Rightarrow\ T_{LB}(x,y) = dist(x,y,G)\cdot\log n$

Memory requirements: $M_{LB} = O(\log^3 n)$,
as each node x stores only its own label $Label(x)$

Theorem: Label-based procedure LB achieves best-service connection setup, with quiescence time and memory requirements within polylog factor of optimum,

$T_{LB}(x,y) = dist(x,y,G)\cdot\log n$

$M_{LB} = O(\log^3 n)$

Application 2: Memory-free routing
[P,99]

Traditional routing schemes:

1. *Addresses:* Attached to messages
   (Must be *very* small)

2. *Local routing tables:* Stored at nodes
   (Hopefully not too large)

Memory-free routing schemes:
Only one type of labels

Goal: Route messages between x, y relying solely on $Label(x), Label(y)$ (+ labels of nodes along route) without any additional information

Routing labeling schemes for trees

Traditional:
$3\log n$ bit addresses,
$O(\min\{deg(v)\log n, \sqrt{n}\log n\})$ bit routing tables
[Cowen,01]

Lower bound:
$\forall$ routing scheme with addresses $\in [1, n]$
requires $\Omega(\sqrt{n})$ bit local routing tables
[Eilam,Gavoille,P,98]

Memory-free:
1. Routing labeling schemes with $c\log n$ bit labels for small const c *[Frigniaud,Gavoille,01]*

2. Improved to $c = 1 + O(1/\log\log n)$ *[TZ,01]*

Contrasting with $\Omega(\sqrt{n})$ lower bound of *[EGP,98]*:
$\Rightarrow$ Variation of $O(\frac{\log n}{\log\log n})$ additive term on address size affects routing table size

Dynamic distributed setting

Motivation: Applications in
distributed systems / telecomm networks:

- Topology changes dynamically;
- online scheme must reflect current situation

Goal: localized labeling schemes using
distributed adaptive marker protocols

Setting: Dynamic tree networks

- Tree topology

- Each node is a processor

Possible events:

Leaf addition:

New leaf u added;
Parent is informed;
assigns unique Child# to u

Leaf removal:

Leaf u is deleted:
Parent is informed

Main results

Result 1: $\exists$ *distributed* distance labeling scheme for dynamic trees using $O(\log^2 n)$ bit labels with amortized message complexity $O(\log^2 n)$ per change

Result 2: $\exists$ general transformation extending a *static* labeling scheme on trees to the *dynamic* setting with *polylogarithmic* overheads

Applicable to a number of natural tree functions (*distance, separation level, flow*)

Settings & complexity measures

Static topology, distributed scheme:

Fixed n-node tree

Label Size:

$\mathcal{LS}(\mathcal{M}, n)$ = max label size

Message Complexity:

$\mathcal{MC}(\mathcal{M}, n)$ = max # messages sent by $\mathcal{M}$ during labeling

Semi-dynamic topology

Only node *additions* are allowed

Marker protocol $\mathcal{M}$ activated after each node addition.
May modify existing labels.

n_0 = initial tree size
n_+ = # leaf additions
$n_f = n_0 + n_+$

Assume: Topological changes are sufficiently spaced

- $\mathcal{LS}(\mathcal{M}, n)$ remains as for static setting
- $\mathcal{MC}(\mathcal{M}, n_0, n_+)$ = max # messages sent

Dynamic topology

Both leaf *additions* and *deletions* allowed

$\bar{n} = (n_1 = 1, n_2, \ldots, n_t)$
n_i = tree size after ith topology change

- $\mathcal{LS}(\mathcal{M}, n)$ remains as for static setting

- $\mathcal{MC}(\mathcal{M}, \bar{n})$ = max # messages sent

Dynamic distance labeling scheme

Step 1: Static scheme StatDL
$\mathcal{LS}(\mathsf{StatDL}, n) = O(\log^2 n)$
$\mathcal{MC}(\mathsf{StatDL}, n) = O(n \log n)$

Step 2: Semi-dynamic scheme SemDL
$\mathcal{LS}(\mathsf{SemDL}, n) = O(\log^2 n)$
$\mathcal{MC}(\mathsf{SemDL}, n_0, n_+) = O(n_f \log^2 n_f)$

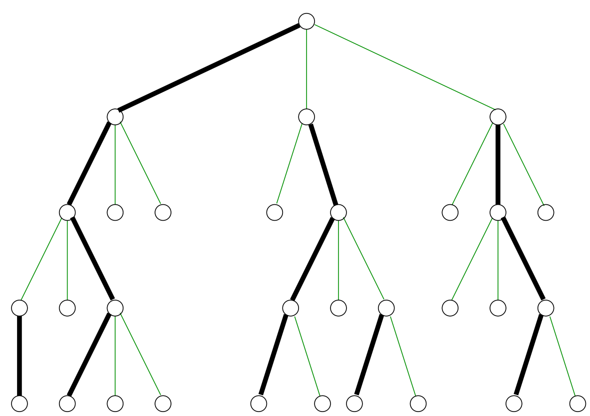
Step 3: Dynamic scheme DL
$\mathcal{LS}(\mathsf{DL}, n) = O(\log^2 n)$
$\mathcal{MC}(\mathsf{DL}, \bar{n}) = O(\sum_i \log^2 n_i)$

Amortized message complexity per topology change:
$O(\log^2 n)$

Static scheme StatDL

Def: $\omega(u) = \#$ nodes in u's subtree

Marked & common children:
Each node u marks edge to *heaviest* child

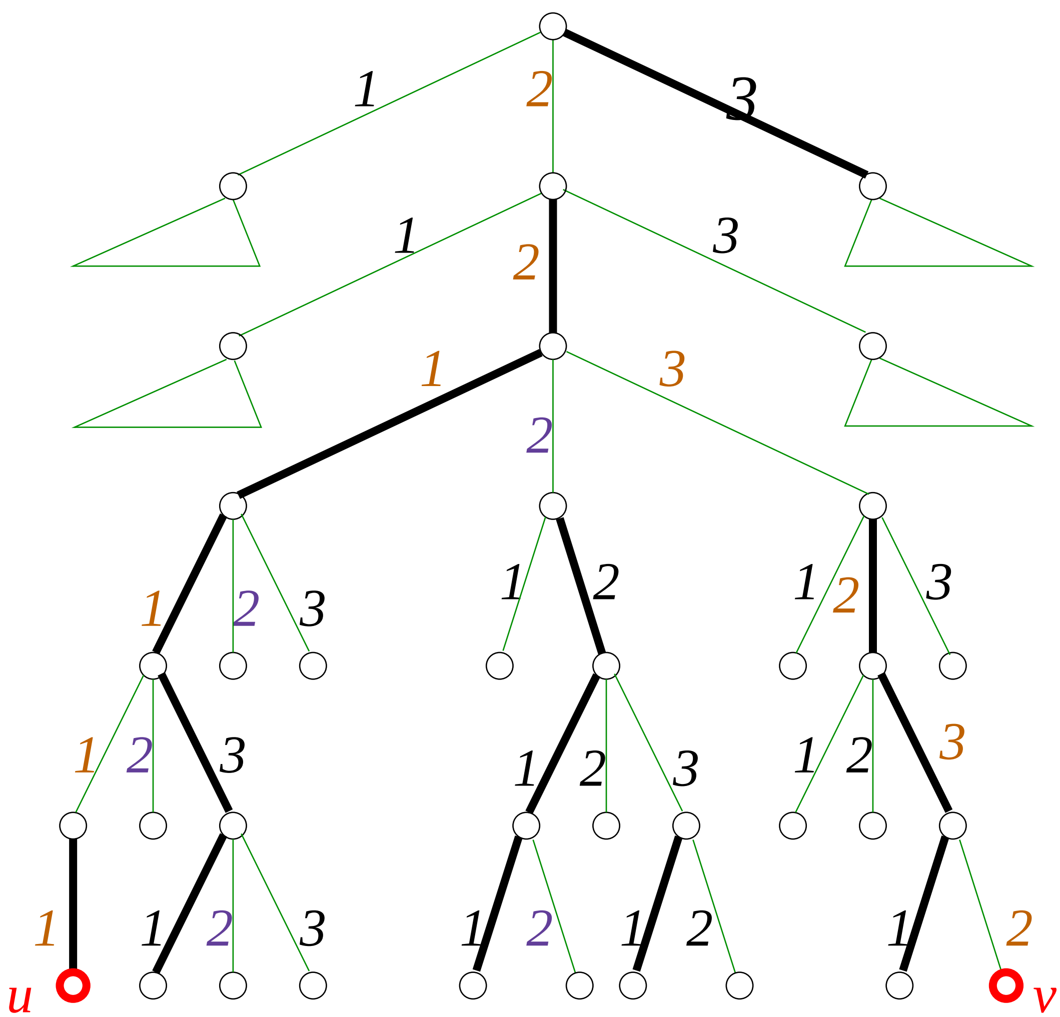


$Common(u)$ = set of unmarked children

Observ: For common child v of u, $\omega(v) \leq \omega(u)/2$

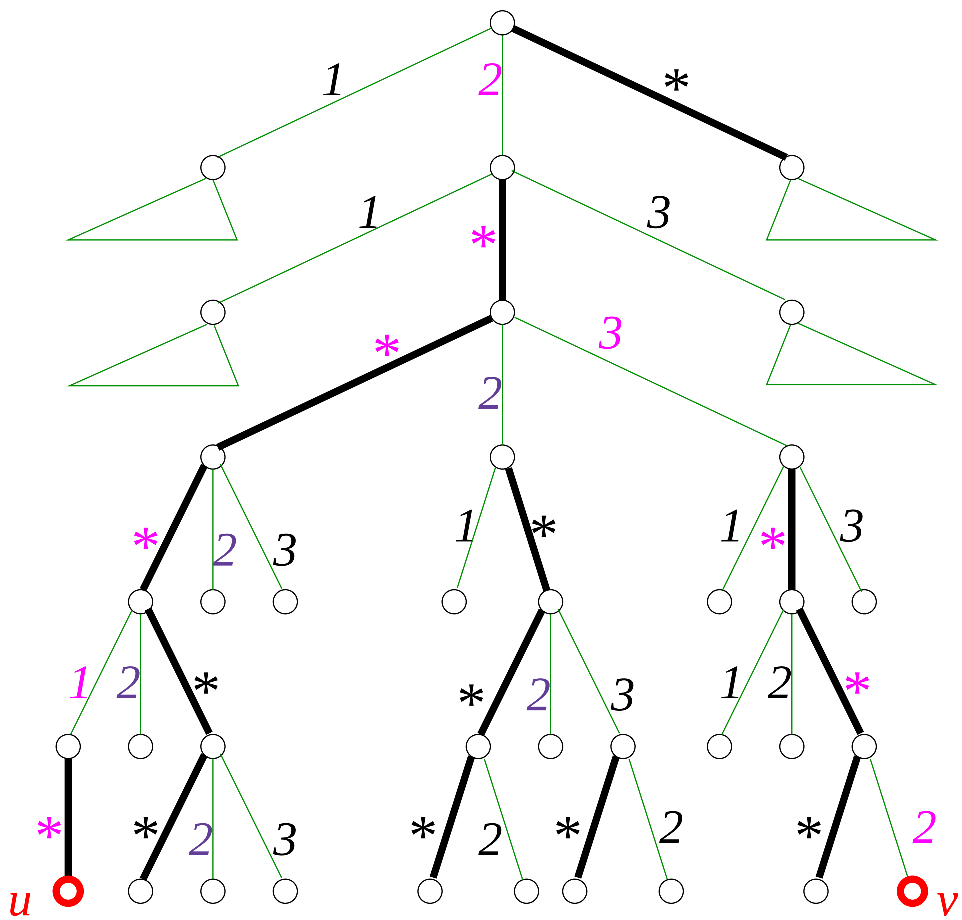
Balance property: $\forall\ u$, $path(u)$ (from root) contains $O(\log n)$ unmarked edges

The label structure



$L^{\text{simple}}(u)=(2.2.1.1.1.1)$ $L^{\text{simple}}(v)=(2.2.3.2.3.2)$

Compacting the labels



$L^{\text{full}}(u)=(2,*,*,*,1,*)$

$L^{\text{full}}(v)=(2,*,3,*,*,2)$

$L^{\text{comp}}(u)=(2,*^3,1,*)$

$L^{\text{comp}}(v)=(2,*,3,*^2,2)$

Distributed marker protocol

Def: $\omega(u)$ = # nodes in u's subtree

1. Assign Child# $1, 2, \ldots$ to children in arbitrary order

2. Calculate node weights bottom-up by convergecast

3. Node u marks edge at heaviest child

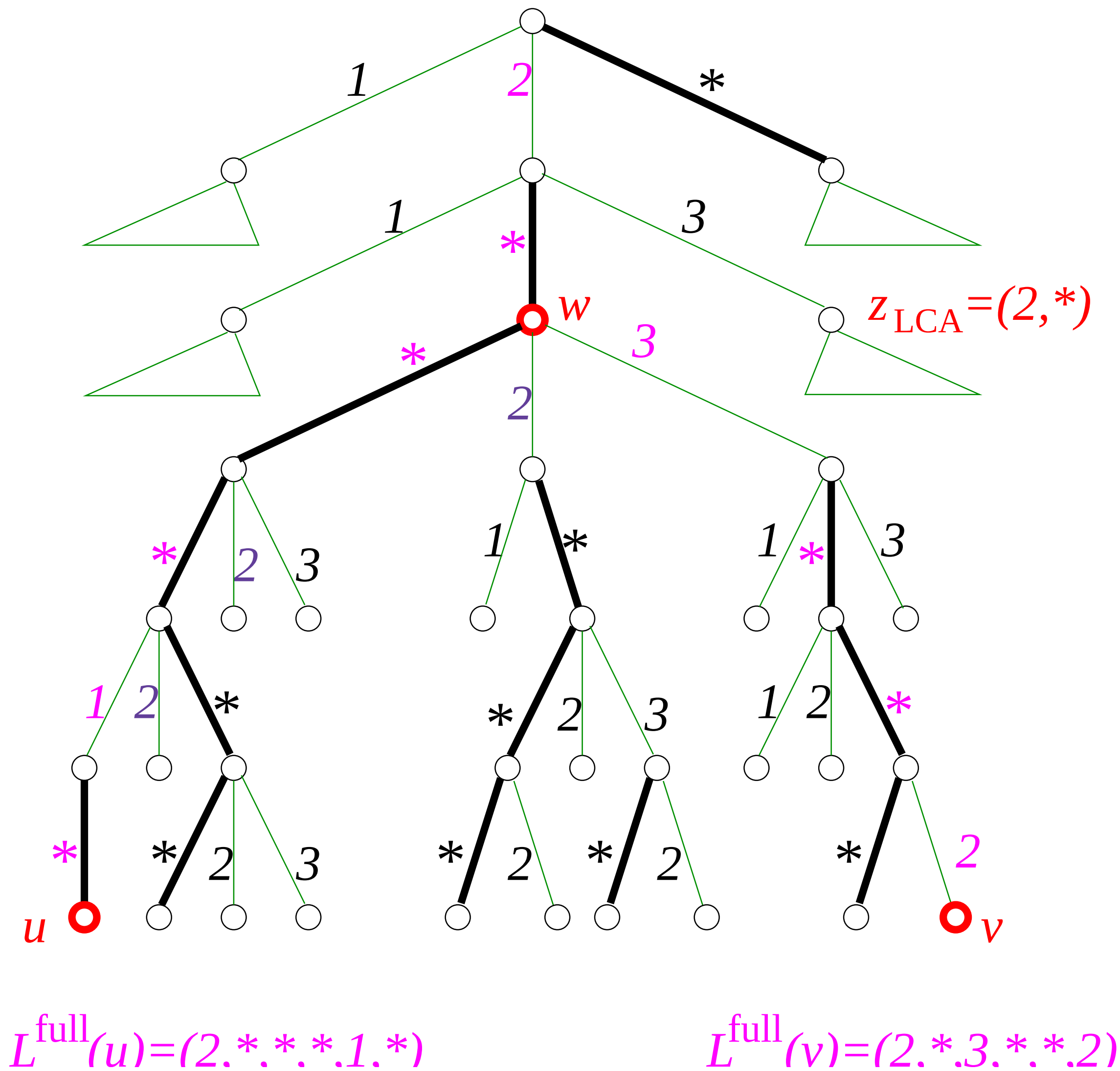
4. Assign distance labels top-down:

   (a) **For** v=root: $L^{\text{full}}(v) \leftarrow \langle\rangle$ /* empty list
   For v child of u with Child# $= s$:

   $$L^{\text{full}}(v) \leftarrow \begin{cases} L^{\text{full}}(u) \cdot \langle\star\rangle, & \text{if } v \text{ is marked,} \\ L^{\text{full}}(u) \cdot \langle s\rangle, & \text{otherwise.} \end{cases}$$

   (b) Compute $L^{\text{comp}}(v)$ from $L^{\text{full}}(v)$; send to all children

Distance computation



$$
\begin{aligned}
dist(u,v) &= dist(u,w) + dist(w,v) \\
&= Depth(u) + Depth(v) - 2Depth(w)
\end{aligned}
$$

Analysis

Observations:

1. New child-numbers are of size $\log n$ bits.
2. $\forall$ common child v of u: $\omega(v) \leq \omega(u)/2$
3. Balance property is satisfied.

Thm:

$\mathcal{LS}(\mathsf{StatDL}, n) = O(\log^2 n)$

$\mathcal{MC}(\mathsf{StatDL}, n) = O(n \log n)$

Dynamic distributed tools

General idea:

1. Keep track of node weights
2. Change marked edges & labels accordingly

Main difficulty:

Maintaining exact weights incurs high overhead

Solution: Keep weight *estimates*

$\Rightarrow$ Not clear exactly which child is heaviest, yet common children are small

Important: Minimize # marked edge changes (each costs in communication)

Weight-Watch Protocol WW

Modifying the method of
[Afek,Awerbuch,Plotkin,Saks,1989],
derive *weight estimation* protocol WW
for dynamically growing trees, satisfying

Property WW: $\forall$ node v maintains weight estimation $\tilde{\omega}(v)$ s.t. at any time,

$$\omega(v) \ \leq \ \tilde{\omega}(v) \ \leq \ \frac{5}{4} \cdot \omega(v)$$

Note: Modification fails in fully dynamic model

Lemma: $\mathcal{MC}(\mathsf{WW}, n_0, n_+) = O(n_f \log^2 n_f)$

Weight ratios

Exact ratio:

$$\varrho(u) = \frac{\omega(u)}{\omega(parent(u))}$$

Estimated ratio:

$$\tilde{\varrho}(u) = \frac{\tilde{\omega}(u)}{\tilde{\omega}(parent(u))}$$

Property WW guarantees:

Lemma:

$$\frac{4}{5} \cdot \varrho(u) \;\leq\; \tilde{\varrho}(u) \;\leq\; \frac{5}{4} \cdot \varrho(u)$$

Heavy-Child Protocol HC

1. $\forall$ node v keeps estimated weight $\tilde{\omega}(v)$
   $+$ all children's estimated weights

2. Each node calculates $\tilde{\varrho}(u)$ for each child u

3. **Shuffle event $\gamma(v, u', u)$:**
   Whenever node v observes a common child u satisfies

   $$\tilde{\varrho}(u) \ > \ \frac{3}{4} \, ,$$

   it changes its marked edge from u' to u

Properties

1. If $\tilde{\varrho}(u) > 3/4$ then $\varrho(u) > 3/5$
$\Rightarrow$ $\forall v$, at most one child u has $\tilde{\varrho}(u) > 3/4$

2. $\forall$ v, $\forall$ common child u of v, $\varrho(u) < 15/16$

3. Protocol HC satisfies balance property

4. $\mathcal{MC}(\mathsf{HC}, n_0, n_+) \ \leq \ 2 \cdot \mathcal{MC}(\mathsf{WW}, n_0, n_+)$

$$= \ O(n_f \log^2 n_f)$$

Semi-dynamic Protocol SemDL

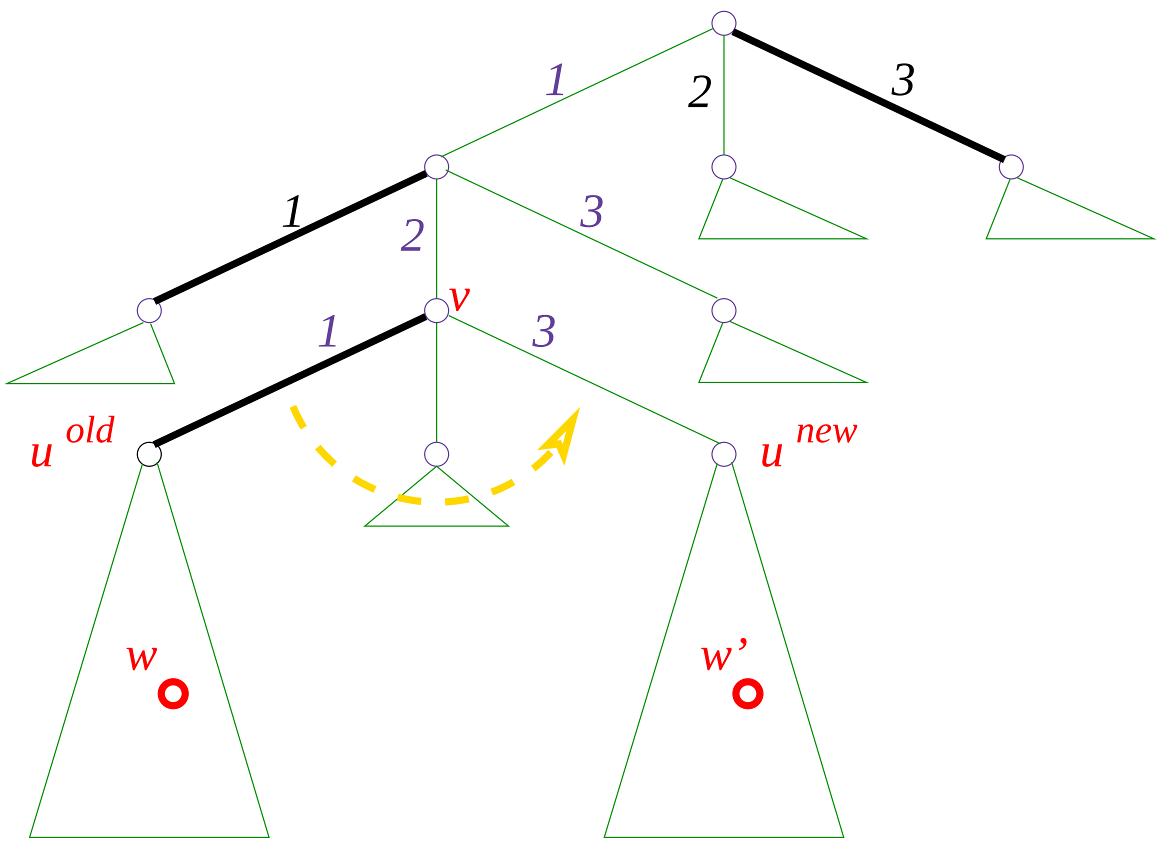
1. Run Protocol StatDL

2. Initiate Protocol HC, monitor its behavior

3. For new v added as child of u
   with Child# $= s$:

$$L^{\mathsf{full}}(v) \ \leftarrow \ \begin{cases} L^{\mathsf{full}}(u) \cdot \langle \star \rangle, & \text{if } v \text{ is marked} \\ L^{\mathsf{full}}(u) \cdot \langle s \rangle, & \text{otherwise.} \end{cases}$$

4. If shuffle $\gamma(v, u^{old}, u^{new})$ occurs in HC
   where v shifts its pointer from u^{old} to u^{new}
   with Child# $= s^{old}$ and s^{new} resp, then:
   (a) $\forall w \in subtree(u^{new})$ changes entry $depth(v)$
       of $L^{\mathsf{full}}(w)$
       from s^{new} to $\star$
   (b) $\forall w \in subtree(u^{old})$ changes entry $depth(v)$
       of $L^{\mathsf{full}}(w)$
       from $\star$ to s^{old}



L(w)=(1,2,,...)*

⇓

L(w)=(1,2,1,...)

L(w')=(1,2,3,...)

⇓

L(w')=(1,2,,...)*

Analysis

Label size:

1. Protocol SemDL maintains correct labels
2. Protocol HC maintains the balance property

Complexity:

- Step 2 takes $O(n_f \log^2 n_f)$ messages
- Step 3 takes one $O(\log^2 n)$-bit message
- Step 4 takes $O(\omega(v)) \equiv O(\omega(\gamma))$ messages
- Tree size satisfies $n < n_f$ at all times

Lemma: $\sum_\gamma \omega(\gamma) \;=\; O(n_f \log n_f)$

Proof:

$\psi(v) = \sum_{u \in Common(v)} \omega(u)$

$\Psi = \sum_v \psi(v)$

Adding v:

$\psi(u)$ increases by 1

$\Leftrightarrow$ $v \in subtree(w)$ for some $w \in Common(u)$

$\Leftrightarrow$ u is on $path(v)$ with unmarked outgoing edge

$path(v)$ has $O(\log n) \leq O(\log n_f)$ unmarked edges

$\Rightarrow$ Ψ increases by $O(\log n_f)$

$\Rightarrow$ all additions contribute $O(n_+ \log n_f)$

Initially: $\Psi \leq O(n_0 \log n_0)$

Shuffle $\gamma(v, u^{old}, u^{new})$**:**

$\varrho(u^{new}) \geq 3/5 \Rightarrow \ \varrho(u^{old}) \leq 2/5 \Rightarrow$

$\psi^{post}(v) - \psi^{pre}(v) = \omega(u^{old}) - \omega(u^{new}) \leq -\frac{1}{5} \cdot \omega(v)$

$\Rightarrow$ shuffle decreases Ψ by $\Omega(\omega(v)) = \Omega(\omega(\gamma))$

$\Rightarrow \ \Psi \ \leq \ O(n_0 \log n_0) + O(n_+ \log n_f) - \Omega(\sum_\gamma \omega(\gamma))$.

$\Psi \geq 0 \Rightarrow \ \sum_\gamma \omega(\gamma) \ \leq \ O(n_f \log n_f)$ ∎

Thm:

1. $\mathcal{LS}(\text{SemDL}, n) = O(\log^2 n)$
2. $\mathcal{MC}(\text{SemDL}, n_0, n_+) = O(n_f \log^2 n_f)$

Algorithm DL - dynamic setting

General idea: Run SemDL ignoring deletions, i.e., treat deleted nodes as if they still exist

Problem: If tree size falls significantly then labels that were *small* before might be *large* now

Accounting for deletions: Parallel to SemDL, run protocol estimating # topology changes

Every $\Theta(n)$ topology changes, restart SemDL

Change-Watch Protocol CW

$\tau = \#$ topology changes during execution

Change-Watch Protocol CW**:** Instance of the protocol of *[AAPS,89]* in which the root maintains estimate $\tilde{\tau}$ of τ satisfying

$$\tau \;\le\; \tilde{\tau} \;\le\; \frac{5}{4}\cdot\tau$$

Lemma *[AAPS,89]***:**
$\mathcal{MC}(\mathsf{CW},\bar{n}) = O((n_1+\tau)\log^2(n_1+\tau))$

(Note: Here n_1 is not necessarily 1)

Protocol DL

1. Root calculates initial # nodes n_0

2. Start Protocols SemDL and CW.

3. If node v is deleted during run:
   $u = parent(v)$ simulates its behavior as if v and its descendents are still in the tree:
   (a) For Protocols WW and HC, simulation is trivial - requires no extra messages
   (b) For Protocol SemDL, u simply avoids passing new labels to v following shuffles
   (c) Protocol CW accounts for node deletions

4. When estimateed count of topology changes grows ($\tilde{\tau} \geq n_0/4$): return to Step 1

Thm:

1. $\mathcal{LS}(\mathsf{DL}, n) = O(\log^2 n)$

2. $\mathcal{MC}(\mathsf{DL}, \bar{n}) = O(\sum_i \log^2 n_i)$

Main proof observations:

1. Restart occurs every τ changes for
$\frac{n_0}{5} \leq \tau \leq \frac{n_0}{4}$

2. During a phase (between restarts),
$\frac{3n_0}{4} \leq n_i \leq \frac{5n_0}{4}$

3. During a phase, the overall communication complexity is
$O(n_0 \log^2 n_0) = O(\sum_i n_i \log^2 n_i)$