Compact and Localized Labeling Schemes for Graphs

David Peleg

The Weizmann Institute

Traditional graph representations

Store adjacency information in data structure

- Requires much storage

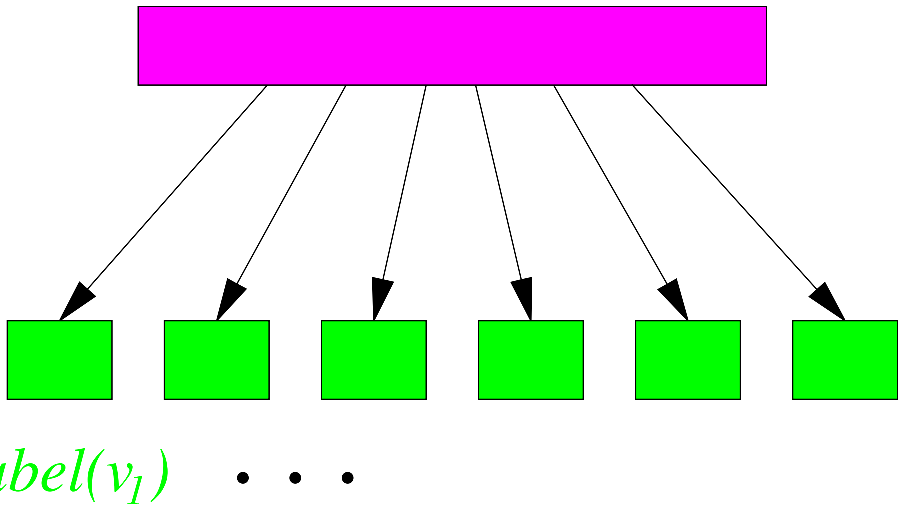
- Node labels contain no information,
  serve only as “pointers” to data structure

	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:

$$u, v \text{ adjacent} \quad \Updownarrow \quad \text{Hamming\_dist}(Label(u), Label(v)) \le T$$

Lemma:
$\forall$ n-node graph
$\exists$ Hamming distance adjacency labeling with
$m = 2n\Delta$, $T = 4\Delta - 4$
where Δ = max node degree
[Breuer,Folkman,67]

Question: Can this be done with short labels?

Example Revived:
Adjacency Labeling on Trees

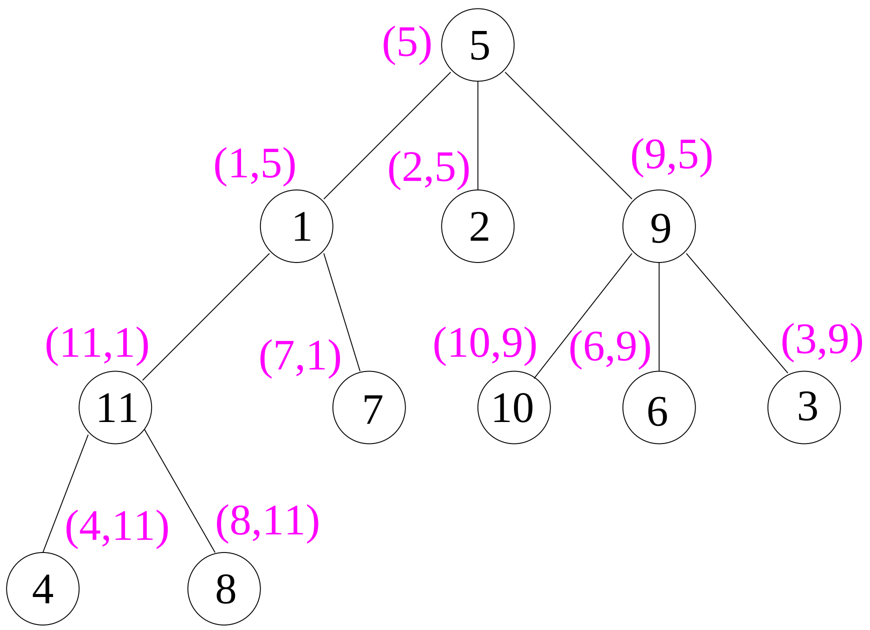
[Kannan,Naor,Rudich,88] *What's in a label?*

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

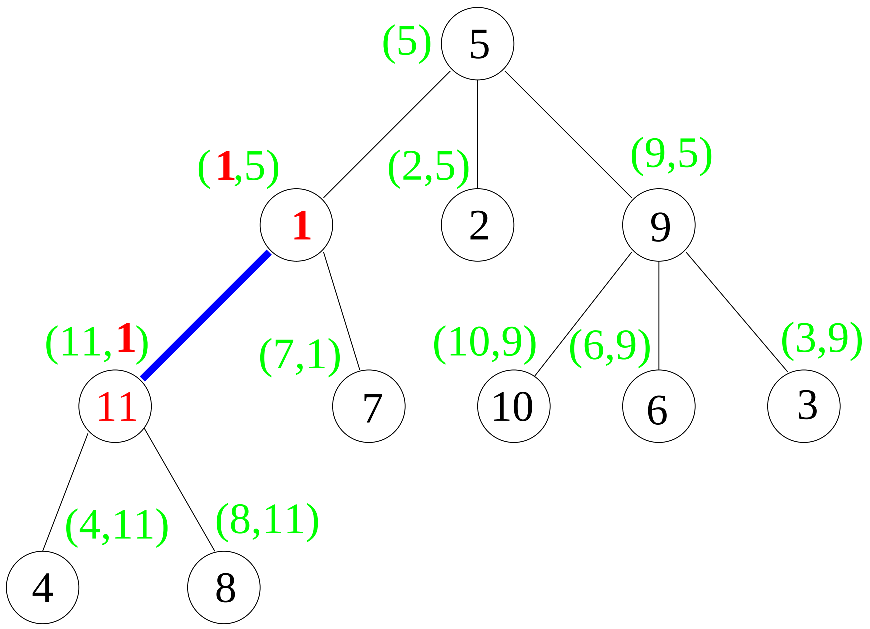
3. Label each nonroot node v with parent w by $Label(v) = (I(v), I(w))$.



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?

Talk plan

1. General framework

2. Distance labeling schemes

   (a) Scheme for trees with $O(\log^2 n)$-bit labels

   (b) General graphs: approx-distance scheme

   (c) Two example applications

3. Other functions

   (a) Ancestors, separation level, LCA

   (b) centers and Steiner trees

   (c) Flow and connectivity

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)

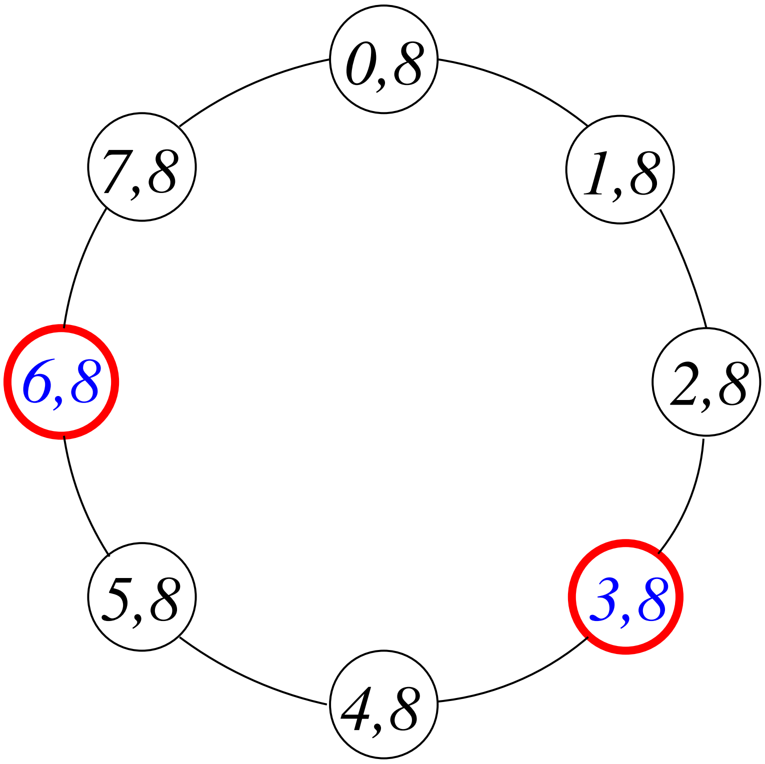
Simple examples:

Highly regular graph classes

- rings, meshes, tori, hypercubes -

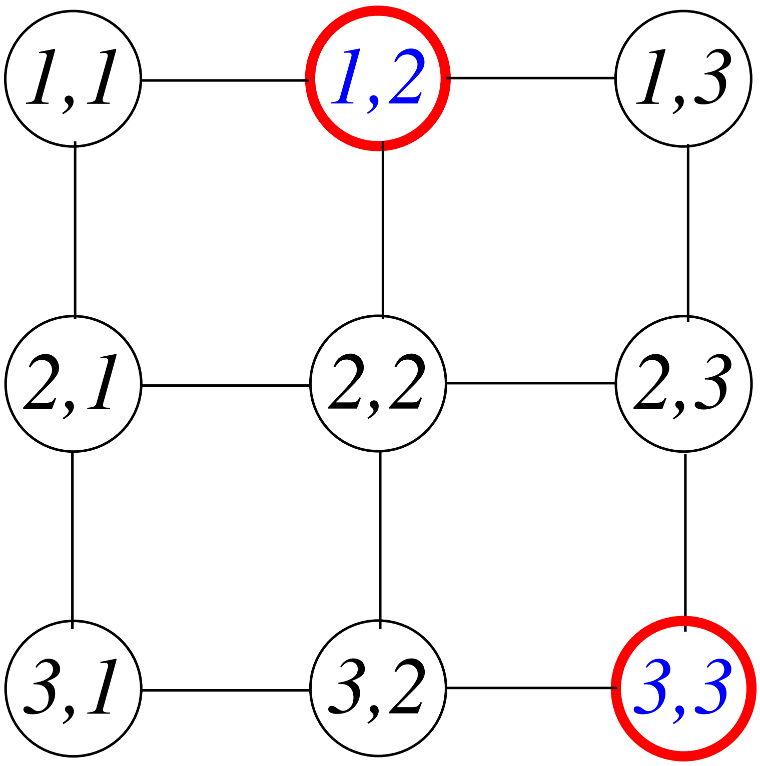
enjoy $O(\log n)$-bit distance labeling schemes

Ring



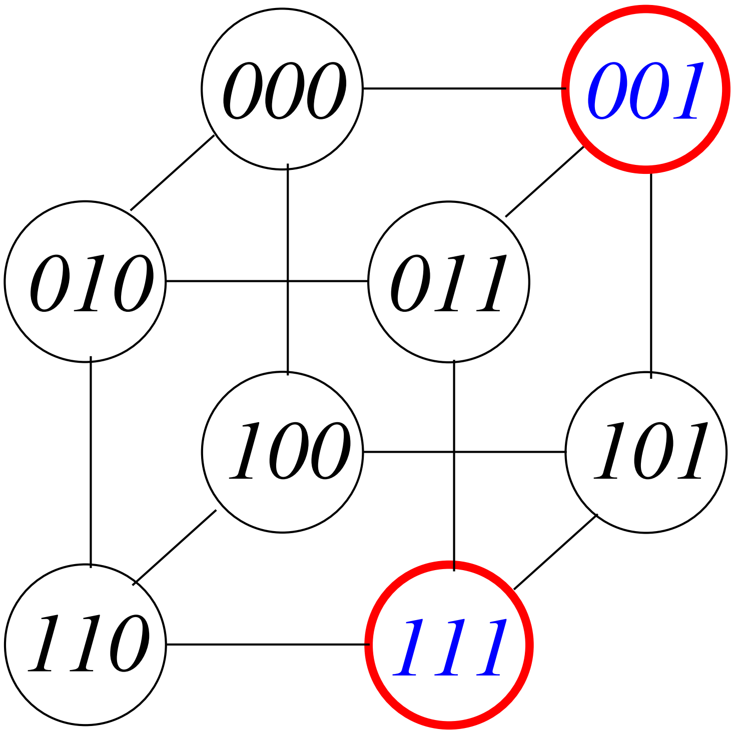
$$dist((3,8),(6,8)) =$$
$$min(6-3, 8+3-6) =$$
$$min(3,5) = 3$$

Grid



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

Hypercube



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

Lower bound

For a very large graph class $\mathcal{F}$:

$\mathcal{F}$ has $\Omega(\exp(n^{1+\epsilon}))$ non-isomorphic graphs

$\Rightarrow$ labeling scheme uses total label length $\Omega(n^{1+\epsilon})$

$\Rightarrow$ some label must have $\Omega(n^{\epsilon})$ bits

Results - Distance labeling schemes

For weighted trees with μ-bit edge weights:

$O(\mu \log n + \log^2 n)$ bit labels *[P,99]*

$\Omega(\mu \log n + \log^2 n)$ lower bound *[GPPR,99]*

For unweighted trees:

$\Theta(\log^2 n)$ bit labels

For interval and permutation graphs:

$O(\log^2 n)$ bit labels *[Katz,Katz,P,00]*

k-separator graphs:

$O(k \log n + \log^2 n)$ bit labels *[GPPR,99]*

$\Downarrow$

$O(\log^2 n)$ bits for *bounded treewidth* graphs

For planar graphs:

$O(\sqrt{n} \log n)$ bit labels *[GPPR,99]*

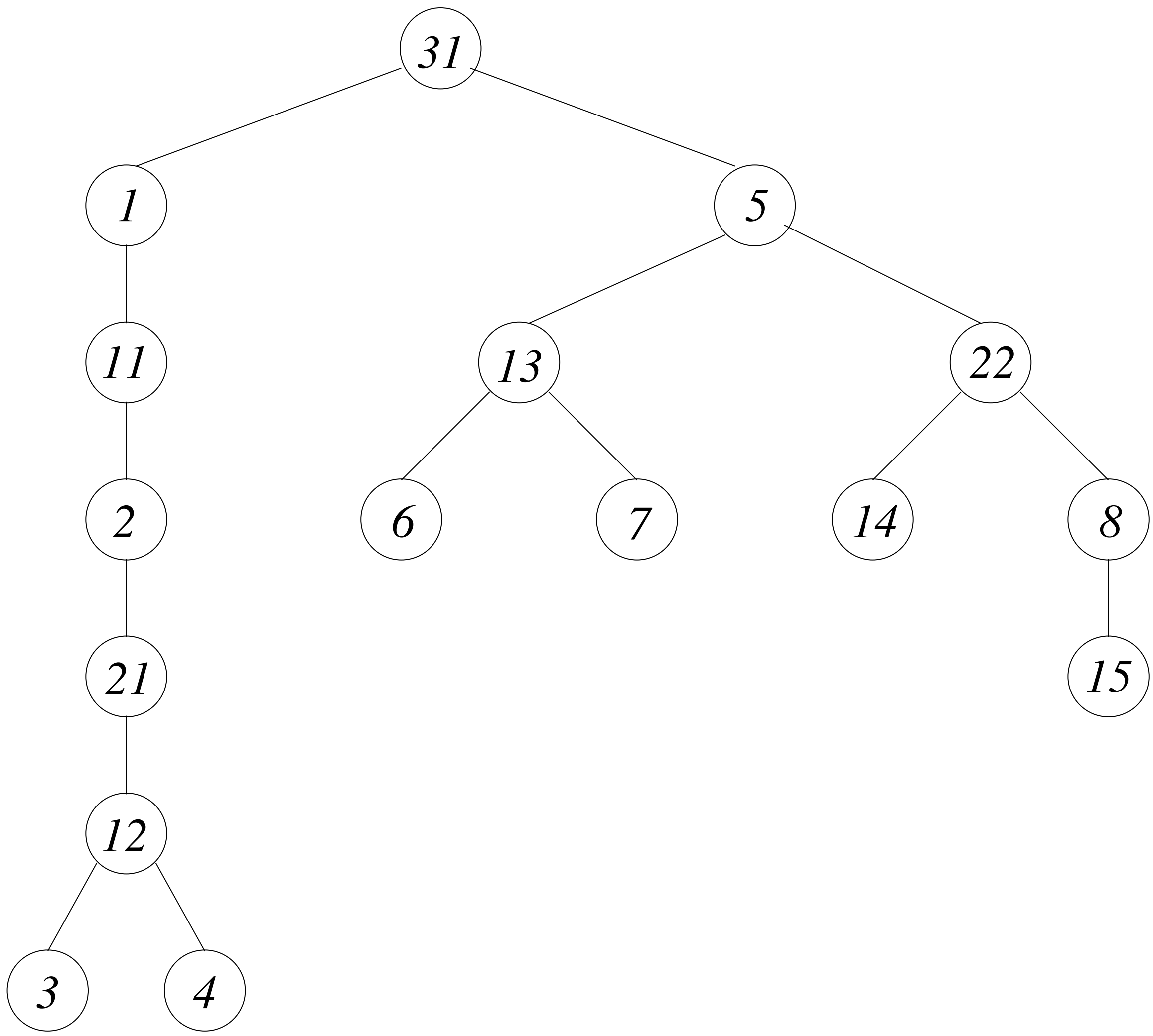
$\Omega(n^{1/3})$ lower bound *[GPPR,99]*

For arbitrary graphs:

$\Theta(n)$ bit labels *[GPPR,99]*

Distance labeling on trees

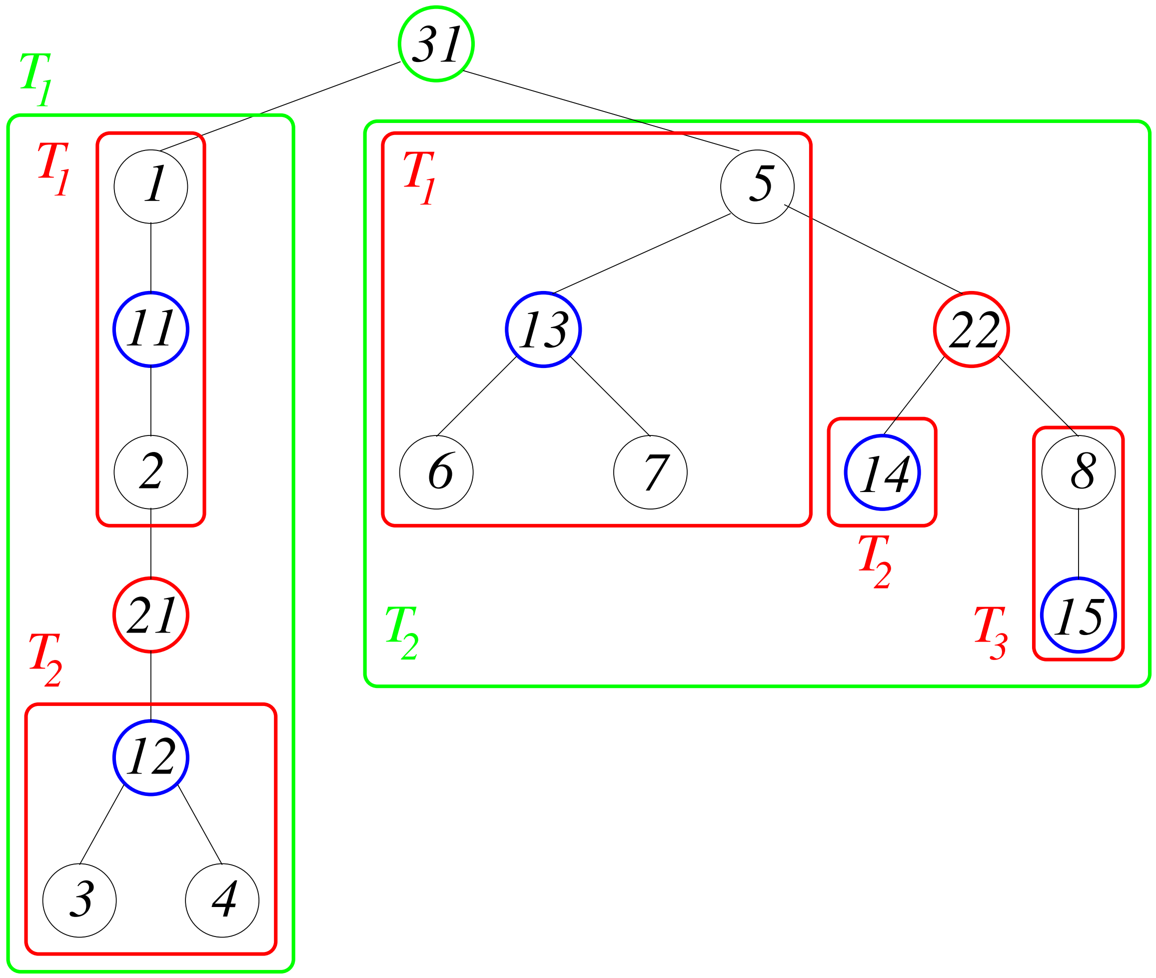
Preprocessing: Arbitrarily prelabel each node v of T with distinct integer $I(v)$ from $[1, n]$



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

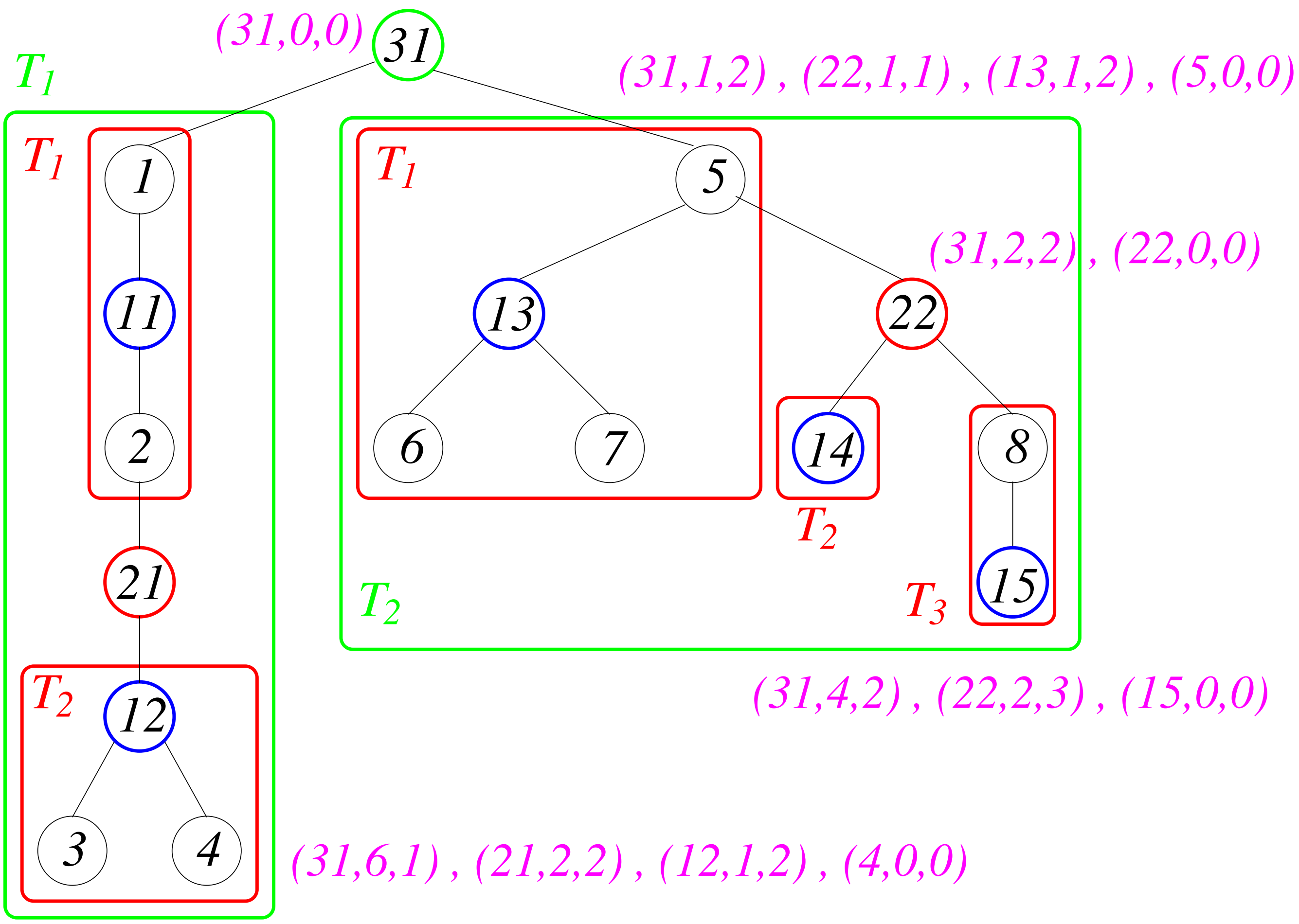
2. */* T' contains > 1 node */*
   Find separator v_0 for T' and remove it.
   / Breaks T' into $T_1, \ldots$ of size $\leq n/2$ */*

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



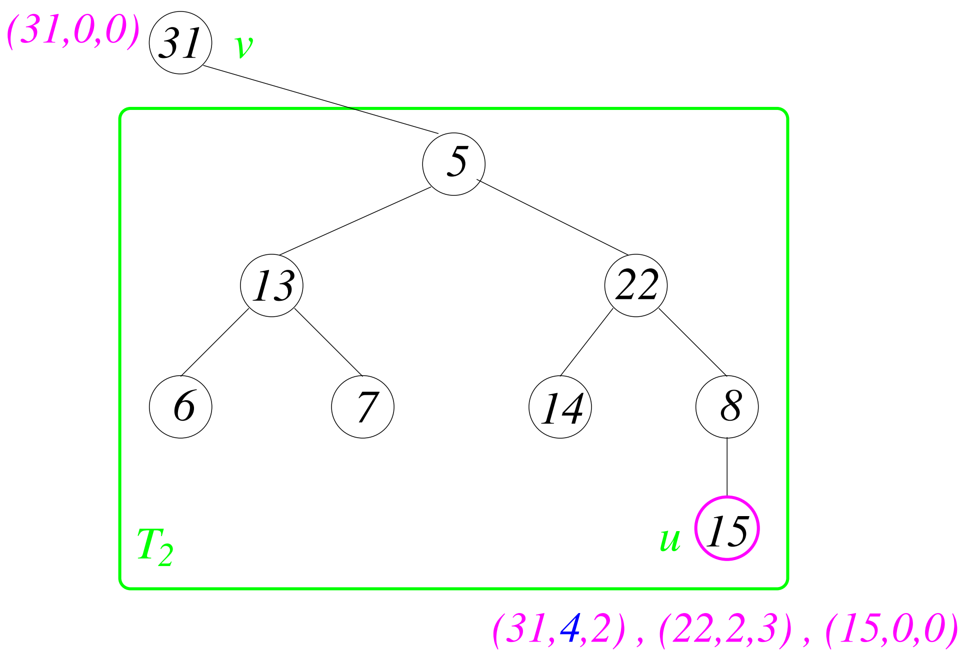
Fields: *(separator , distance , tree #)*

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 $\mathcal{J}_1(v)$

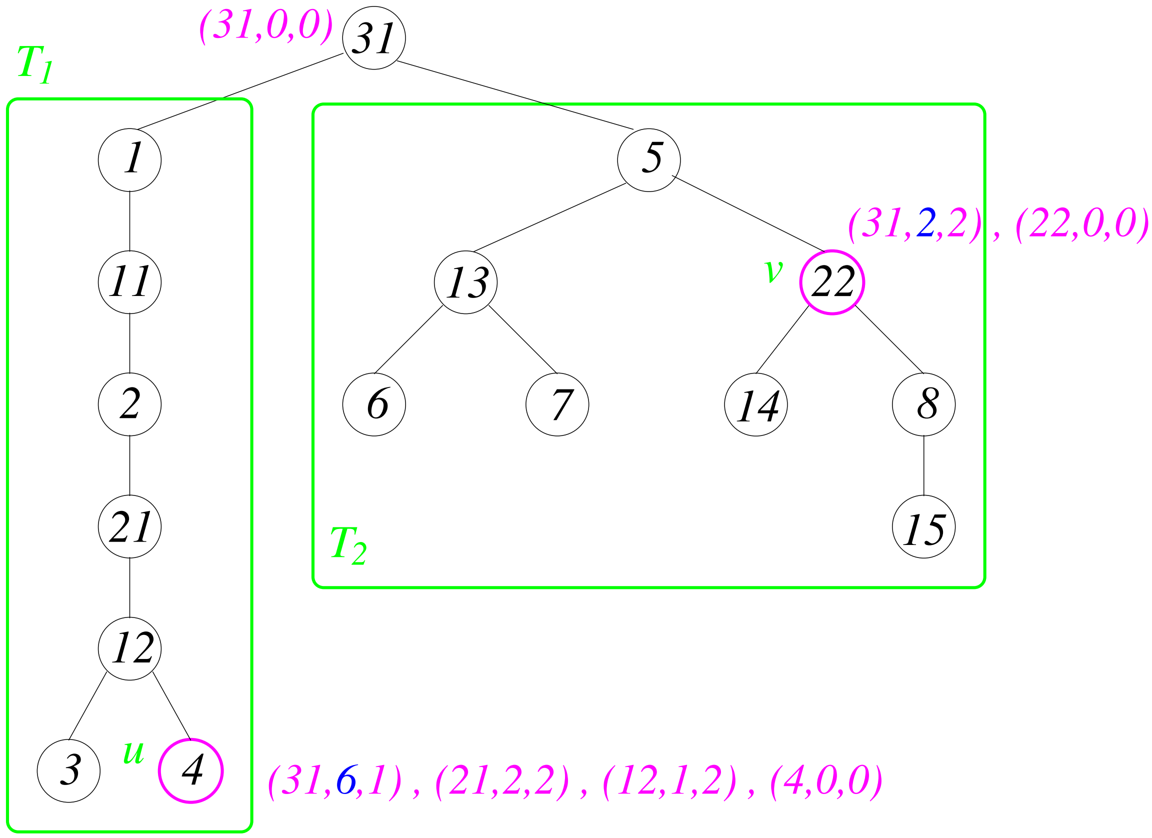
$p, q > 1$: Let

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

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

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

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

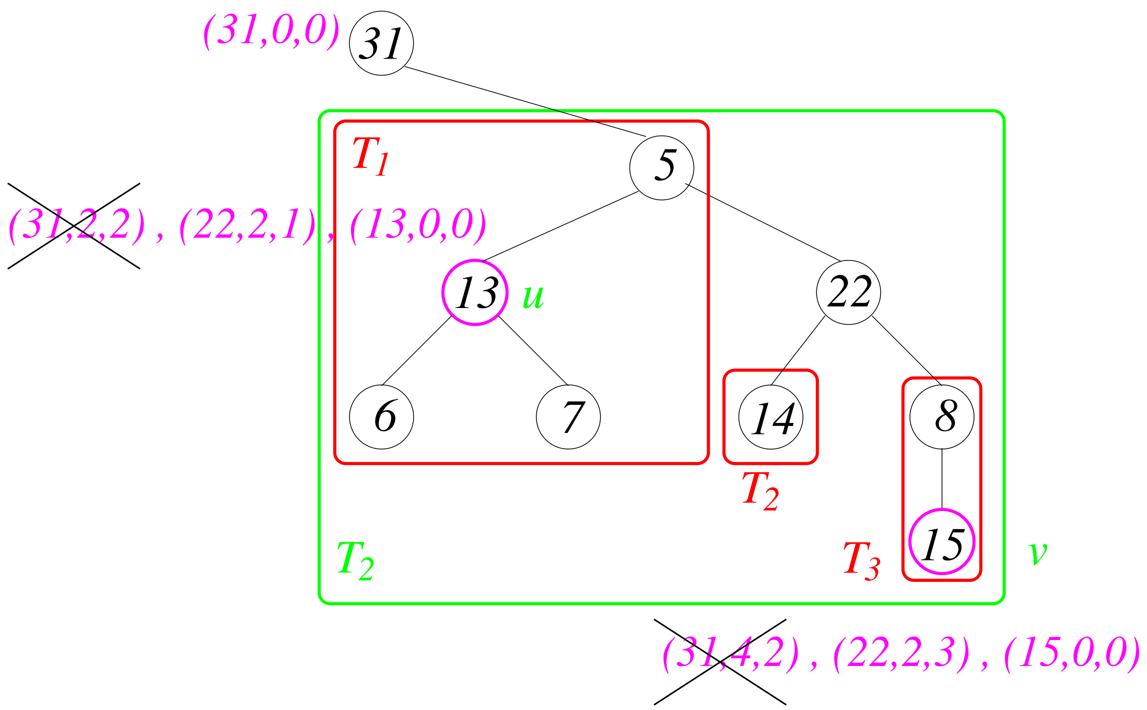


Fields: *(separator , distance , tree #)*

$i = j$:   /* 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

Theorem *[GPPR,99]*: This is tight

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

Approximate distance labeling

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

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

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

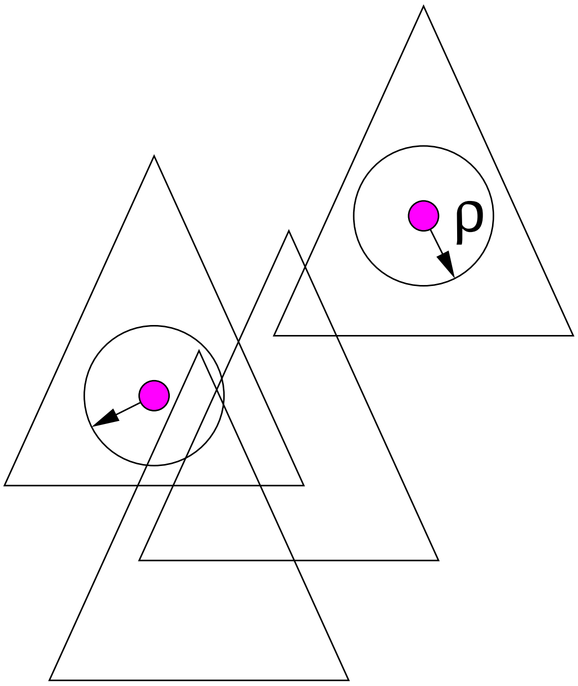
For all weighted graphs:
$(O(\Lambda \log n \cdot \kappa \cdot n^{1/\kappa}), \sqrt{8\kappa})$ approx-distance labeling scheme for $\kappa \ge 1$ *[P,99]*

In particular: /* setting $\kappa = \log n/8$ */
$(O(\Lambda \log^2 n), \sqrt{\log n})$ approx-distance labeling scheme

Distance-approx labelings

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

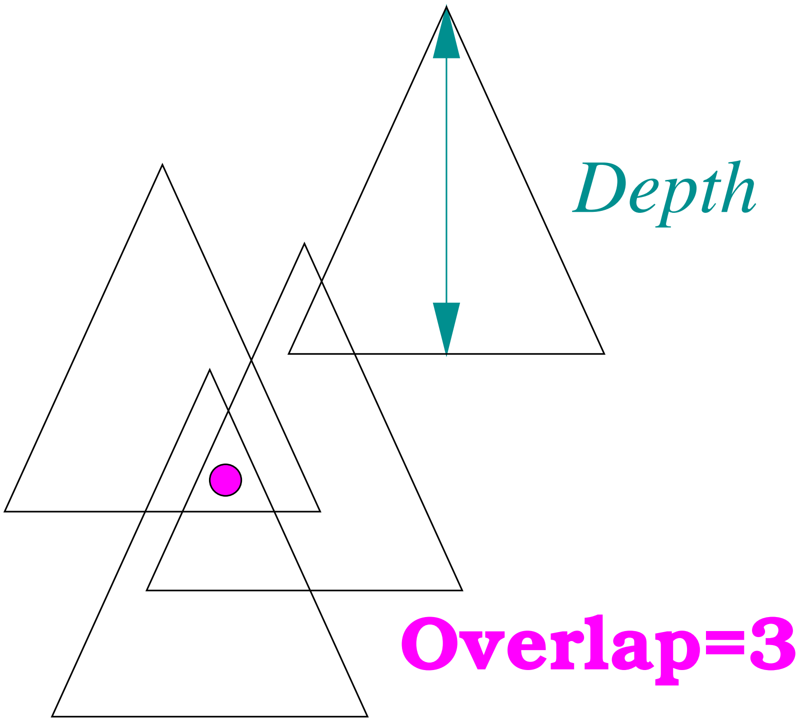
ρ-tree cover: Set $\mathcal{TC}$ of trees in G s.t.
$\forall$ node v $\exists$ tree $T \in \mathcal{TC}$
spanning v's entire ρ-neighborhood



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, $\texttt{Overlap}(\mathcal{TC}) = 1$

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,Peleg,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

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

The marker algorithm

1. $\forall \ 1 \le i \le \Lambda$, 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

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

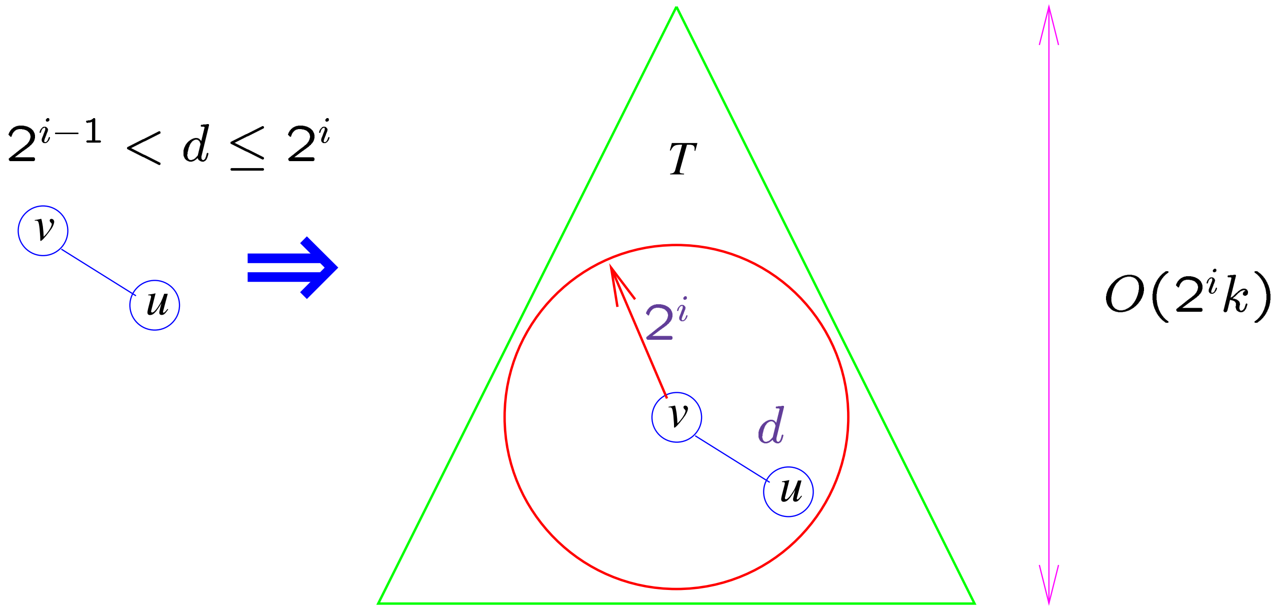
Distance estimation algorithm

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

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 $R = \sqrt{8\kappa}$

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

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]*

Application 1: Connection setup

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

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]*

Ancestry labeling schemes

Easy to encode ancestry in trees with $2\log n$ bit labels using interval-based schemes *[Santoro,Khatib,85]*

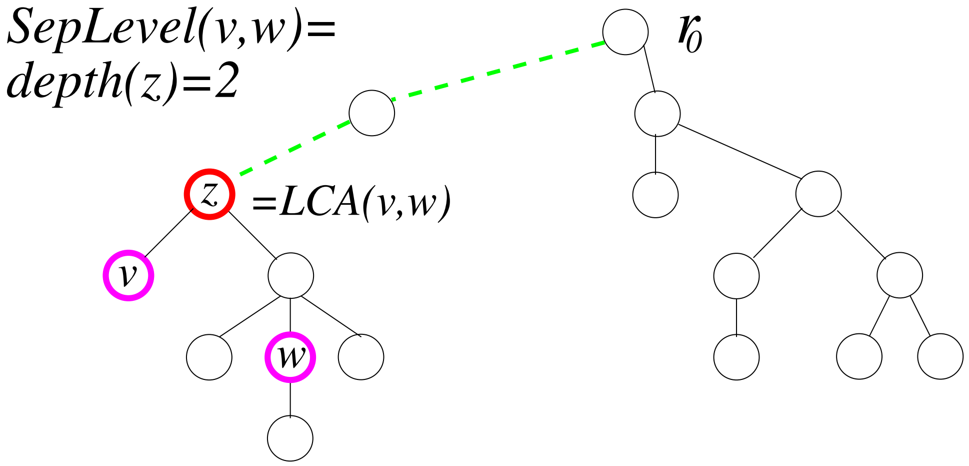
Improvements:

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]*

$SepLevel$ labeling schemes



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

Lemma:

1. $\mathcal{L}(SepLevel, \mathcal{T}_n) \le \mathcal{L}(dist, \mathcal{T}_n) + \log n$

2. $\mathcal{L}(dist, \mathcal{T}_n) \le \mathcal{L}(SepLevel, \mathcal{T}_n) + \log n$

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

LCA labeling schemes

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

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

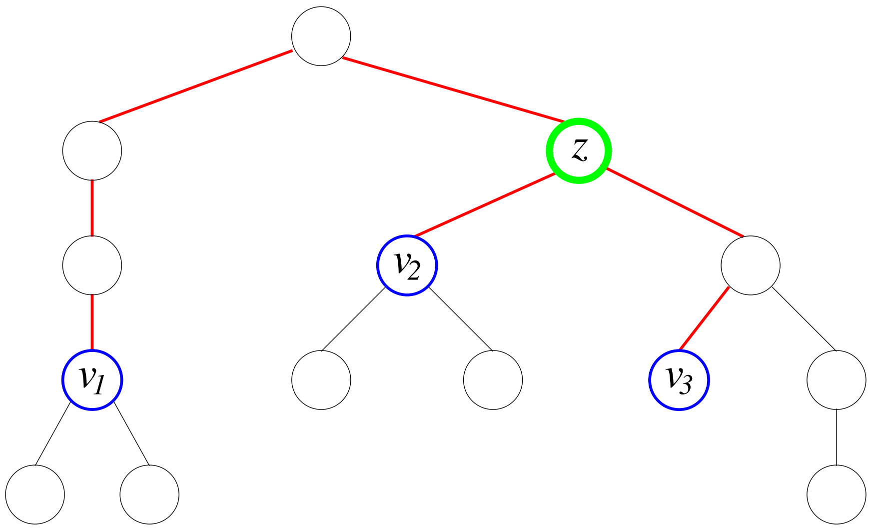
Thm *[P,00]*:
1. $\exists$ LCA labeling scheme with $O(\log^2 n)$-bit labels for trees
2. $\forall$ LCA labeling scheme for trees requires some $\Omega(\log^2 n)$-bit labels

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

$Center$ labeling schemes

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:

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

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

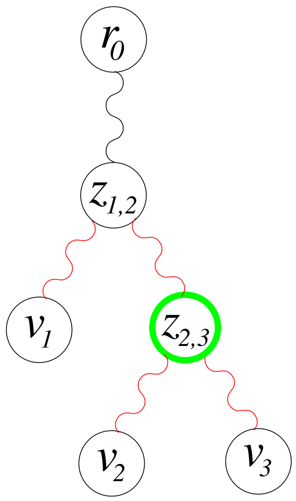
The $Center$-decoder $\mathcal{D}_{Center}$

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

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

2. If *three LCA's coincide* then return one

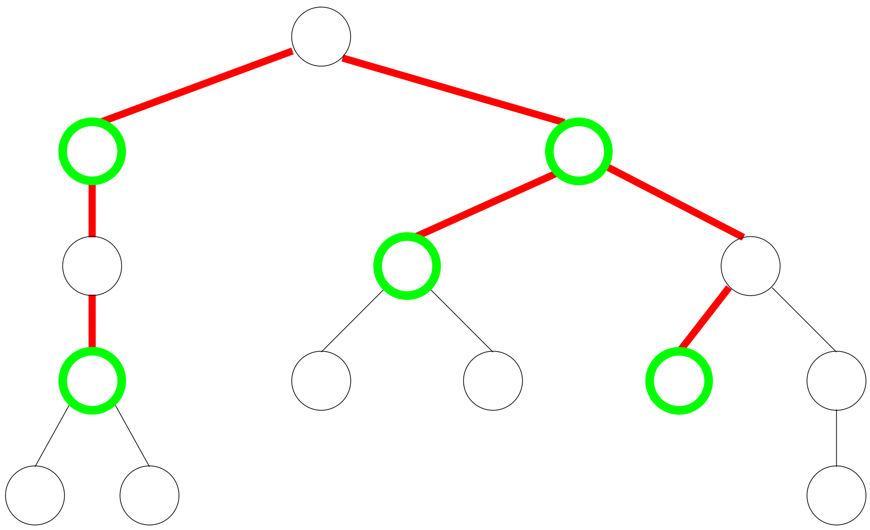
3. If *exactly two LCA's coincide* then *return third*



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

Steiner labeling schemes

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

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]

Flow: $\exists$ scheme with $O(\log n \cdot \log \omega)$ bit labels for general n-vertex graphs with maximum integral capacity ω, and this is tight

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

k-Vertex connectivity:

1. $\exists$ scheme with $O(\log n)$ bit labels for fixed k

2. Lower bound of $\Omega(k \log n)$ for $k = polylogn$

Future applications

- Distance labeling schemes as tool for bounded ("time-to-live") broadcast protocols or topology update mechanisms

- Steiner labeling schemes as tool for optimizing multicast schedules and selection of subtrees for group communication or for certain information representation problems on the Web

- Computational geometry (in context of Euclidean graphs)