On Approximating Degree-Bounded Network Design Problems

Xiangyu Guo
joint work with Guy Kortsarz, Bundit Laekhanukit,
Shi Li, Daniel Vaz, and Jiayi Xian

Aug 17, 2020

Network Design Problems

- **Input:** an graph $G = (V, E)$ with edge cost $c \in \mathbb{R}^E_{\geq 0}$
- **Output:** A min-cost subgraph S of G satisfying certain requirements:
  - Connectivity requirement
    - Minimum spanning tree
    - Minimum Steiner tree
    - Minimum k-edge-connected subgraph
  - Degree bound $d \in \mathbb{R}^V_{\geq 0}$: $\deg_S(v) \leq d_v, \forall v \in V$
- **This talk:** degree-bounded Directed Steiner Tree (DB-DST) and degree-bounded Group Steiner Tree on trees (DB-GST-on-trees)

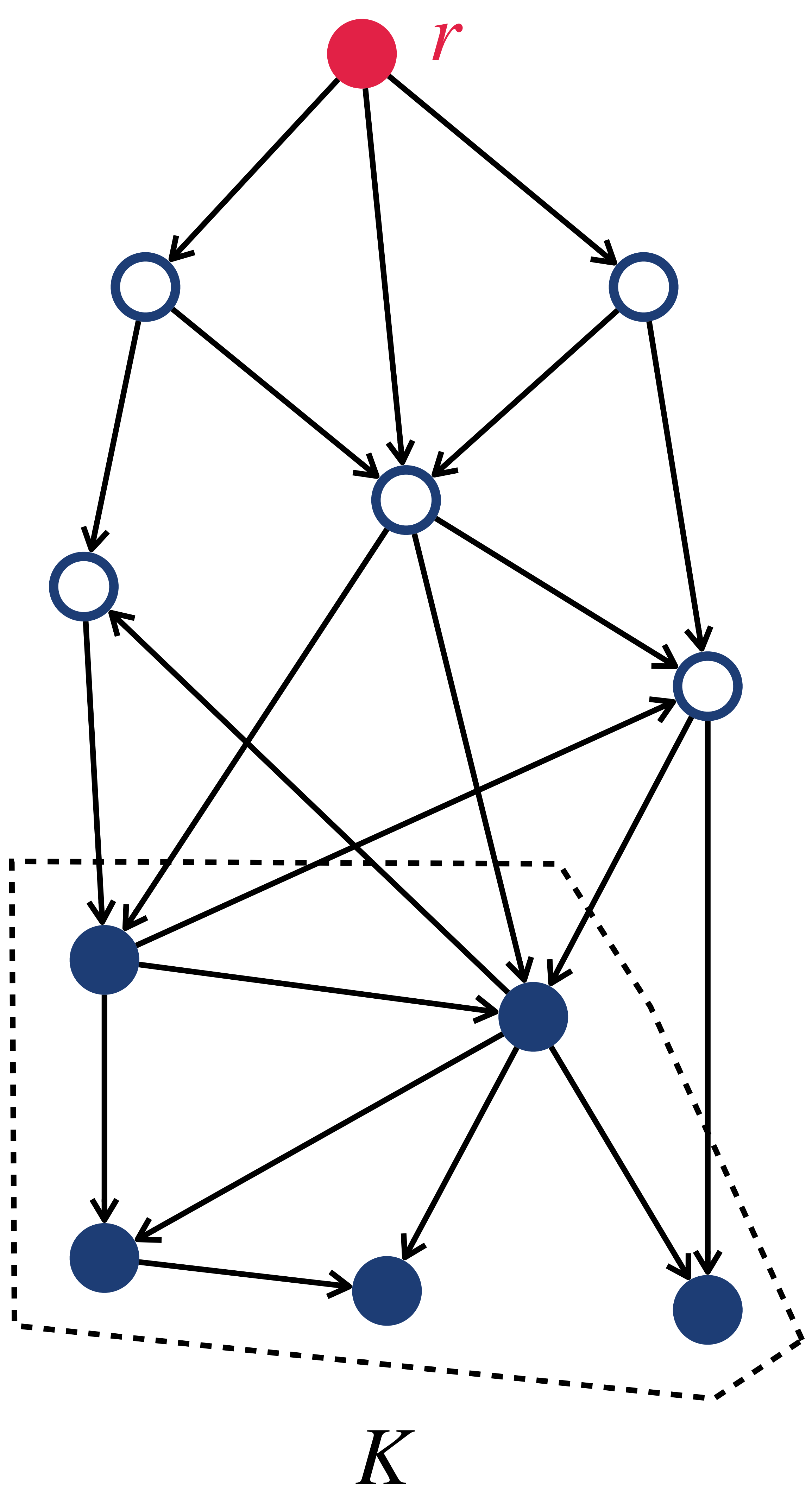
Degree-bounded DST

Input: directed graph $G = (V, E)$ with

- edge cost $c \in \mathbb{R}^E_{\geq 0}$, degree bound $d \in \mathbb{R}^V_{\geq 0}$,
- root $r \in V$, k terminals $K \subseteq V$,

Output: min-cost tree $T \subseteq G$ rooted at r s.t.

- contain $r \to t$ path for every $t \in K$,
- $\forall\, v \in T, \deg^+_T(v) \leq d_v$



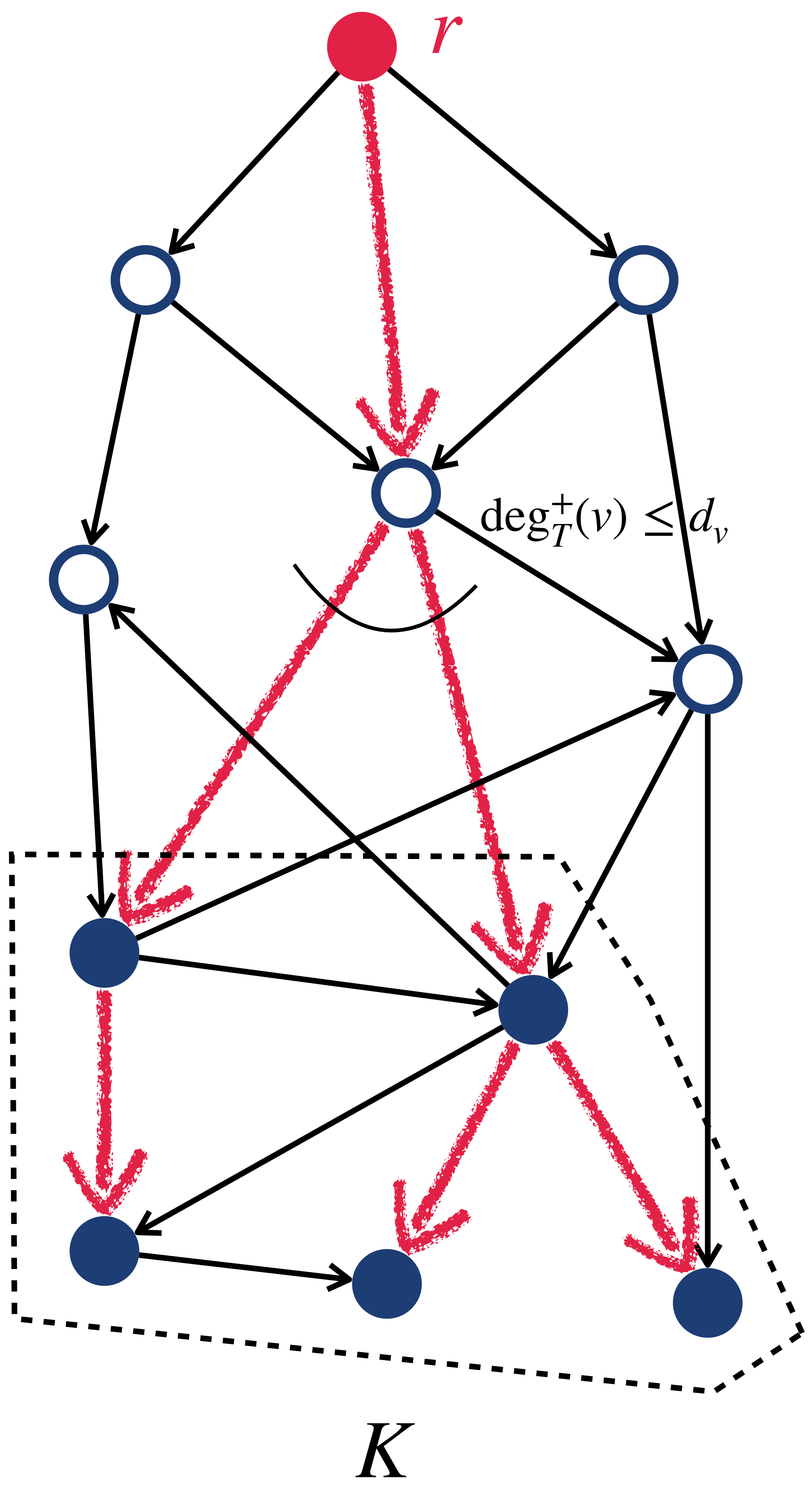
Degree-bounded DST

Input: directed graph $G = (V, E)$ with

- edge cost $c \in \mathbb{R}^E_{\geq 0}$, degree bound $d \in \mathbb{R}^V_{\geq 0}$,
- root $r \in V$, k terminals $K \subseteq V$,

Output: min-cost tree $T \subseteq G$ rooted at r s.t.

- contain $r \to t$ path for every $t \in K$,
- $\forall\, v \in T, \deg^+_T(v) \leq d_v$



Related work

- Degree-bounded network design in *undirected* graphs
  - $(1, d_v + 1)$-apx for DB-MST [Singh-Lau'07]
  - $(2, \min\{d_v + 3, 2d_v + 2\})$-apx for DB-Steiner forest [Lau-Zhou'15, Louis-Vishnoi'09]
- Directed Steiner Tree:
  - $\Omega(\log^{2-\epsilon} k)$-hard [Halperin-Krauthgamer'03]
  - k^{ϵ}-apx in polynomial time [Zelikovsky'97]
  - $O\left(\frac{\log^2 k}{\log\log k}\right)$-apx in quasi-polynomial time [Grandoni-Laekhanukit-Li'19][Ghuge-Nagarajan'19]

Our result

Main Theorem. There's a randomized $(O(\log n \log k), O(\log^2 n))$-bicriteria approx algorithm for the degree-bounded directed Steiner tree (DB-DST) problem, with $n^{O(\log n)}$ running time.

- First non-trivial approximation for the DB-DST problem.
- Close to the $(\Omega(\log^{2-\epsilon} k), \Omega(\log n))$ lower bound
- Based on rounding a novel LP formulation.
- Can handle other constraints: e.g., length bound, buy-at-bulk

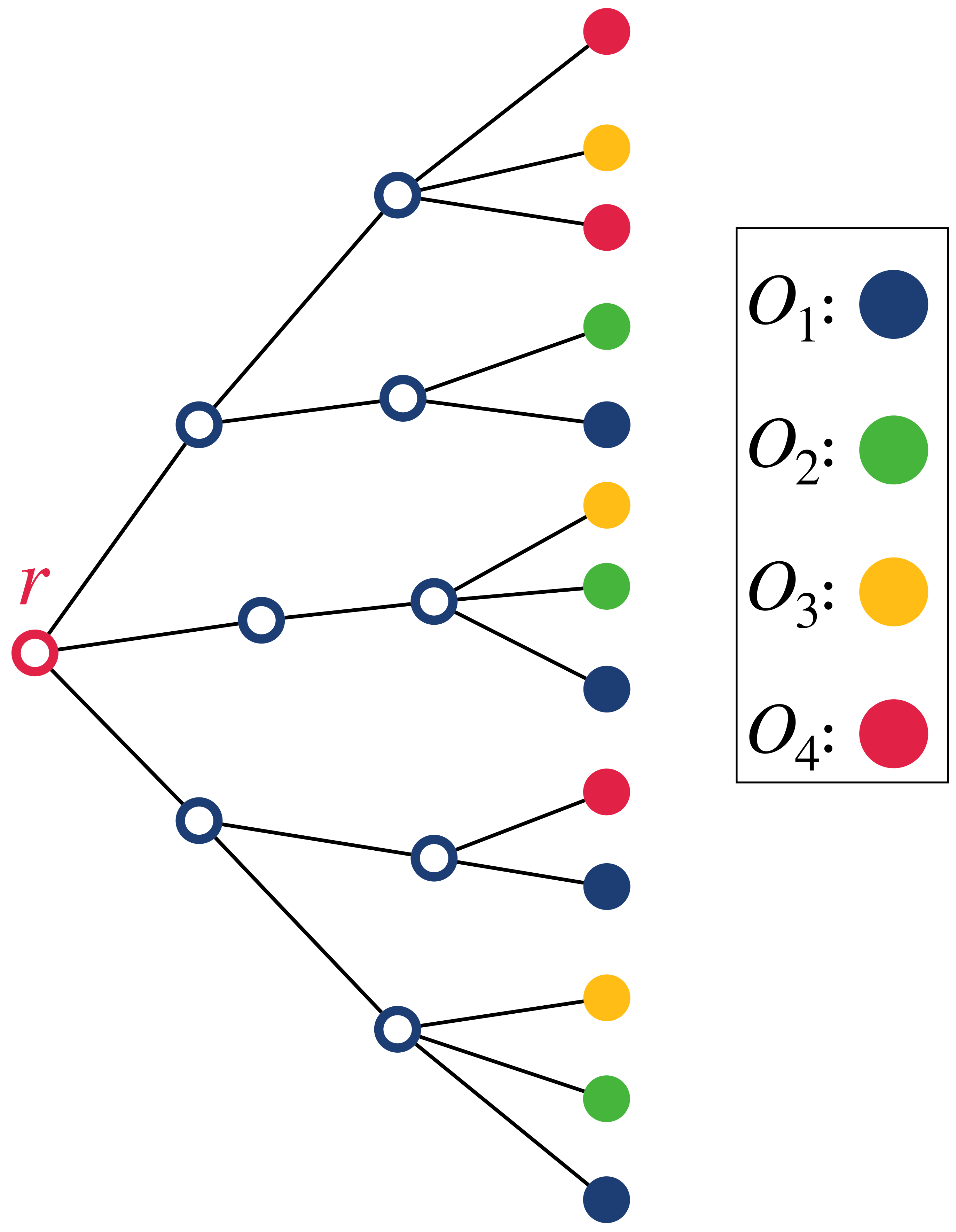
Degree-bounded GST-on-trees

Input: undirected tree $G = (V, E)$ rooted at $r \in V$, with

- edge cost $c \in \mathbb{R}^{E}_{\geq 0}$, degree bound $d \in \mathbb{R}^{V}_{\geq 0}$
- k terminal groups $O_1, O_2, \ldots, O_k \subseteq V$.

Output: min-cost tree $T \subseteq G$ s.t.

- contains a path from r to every terminal group,
- $\forall\, v \in T, \deg_T(v) \leq d_v$.



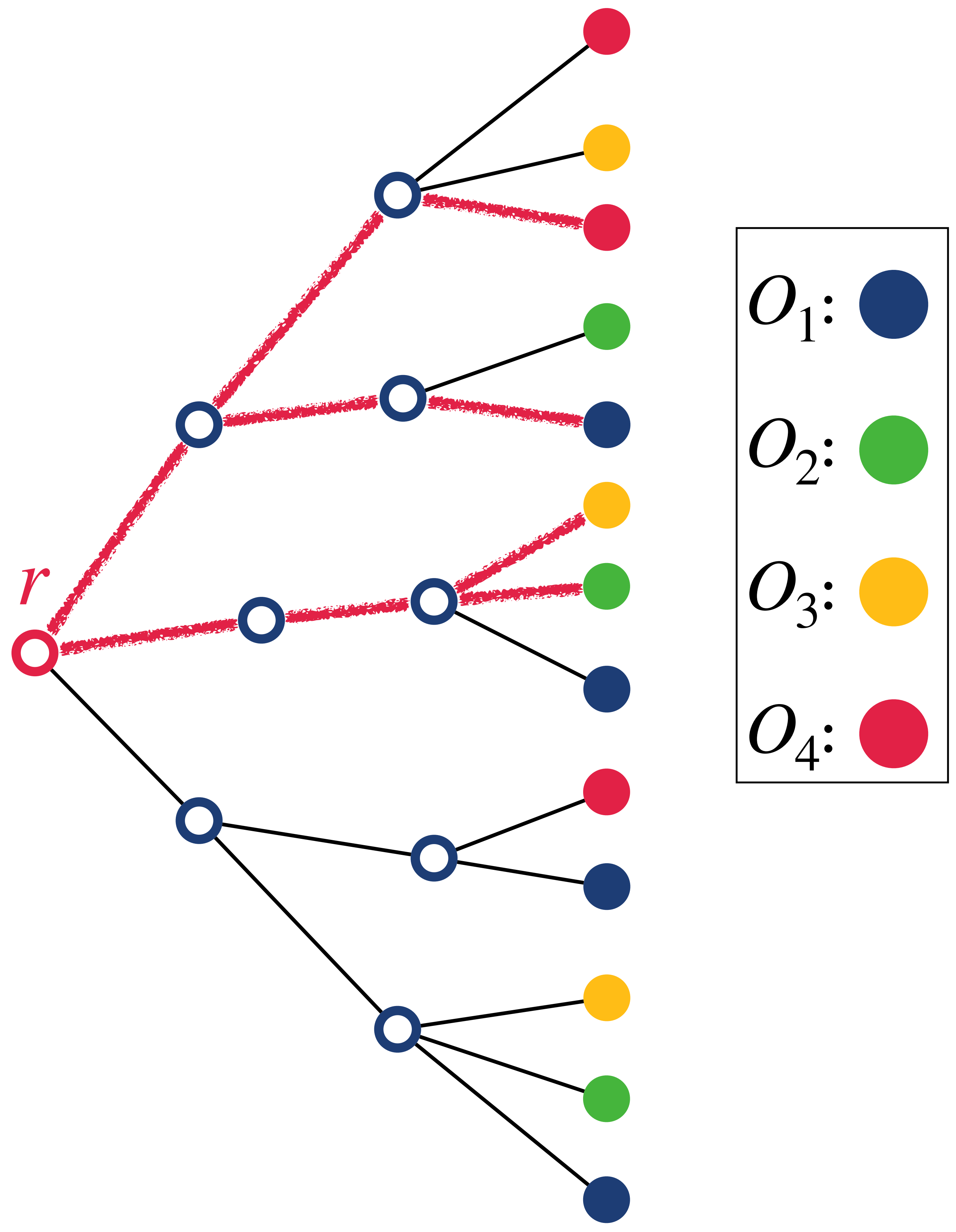
Degree-bounded GST-on-trees

Input: undirected tree $G = (V, E)$ rooted at $r \in V$, with

- edge cost $c \in \mathbb{R}^{E}_{\geq 0}$, degree bound $d \in \mathbb{R}^{V}_{\geq 0}$
- k terminal groups $O_1, O_2, \ldots, O_k \subseteq V$.

Output: min-cost tree $T \subseteq G$ s.t.

- contains a path from r to every terminal group,
- $\forall\, v \in T, \deg_T(v) \leq d_v$.



- Why study GST-on-trees?
  - Source for the $\Omega(\log^{2-\epsilon} n)$-hardness of DST [Halperin-Krauthgamer'03]
  - Our DB-DST alg converts the input to a GST-on-trees instance
- Our result:

  A *polynomial-time* $(O(\log n \log k), O(\log n))$-apx algorithm for DB-GST-on-trees
  - (almost) tight on both the cost ratio and degree violation
  - Improves upon the $(O(\log n \log k), O(\log^2 n))$-apx of [Kortsarz-Nutov'20]

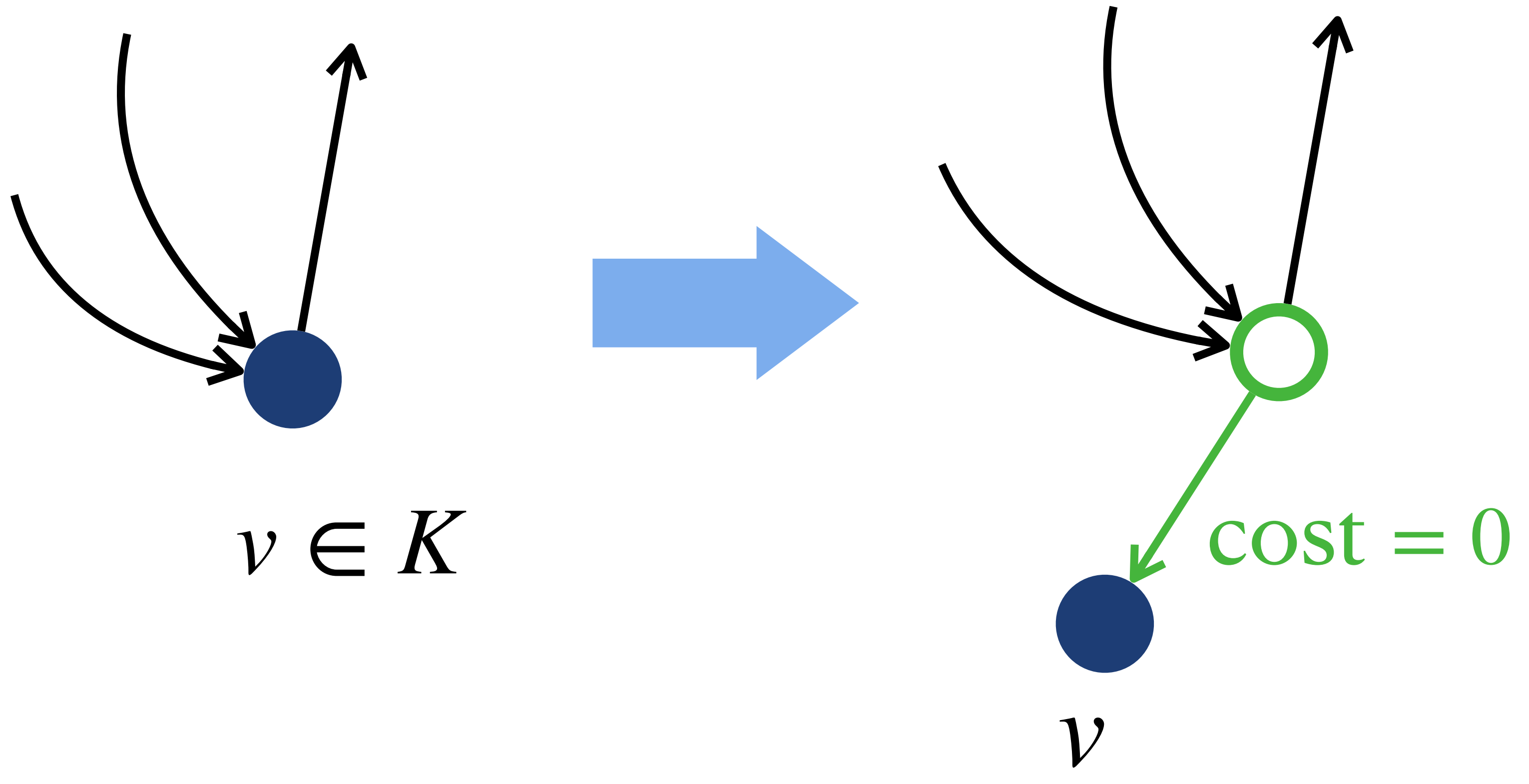
Rest of the talk

The main algorithm for the DB-DST result:

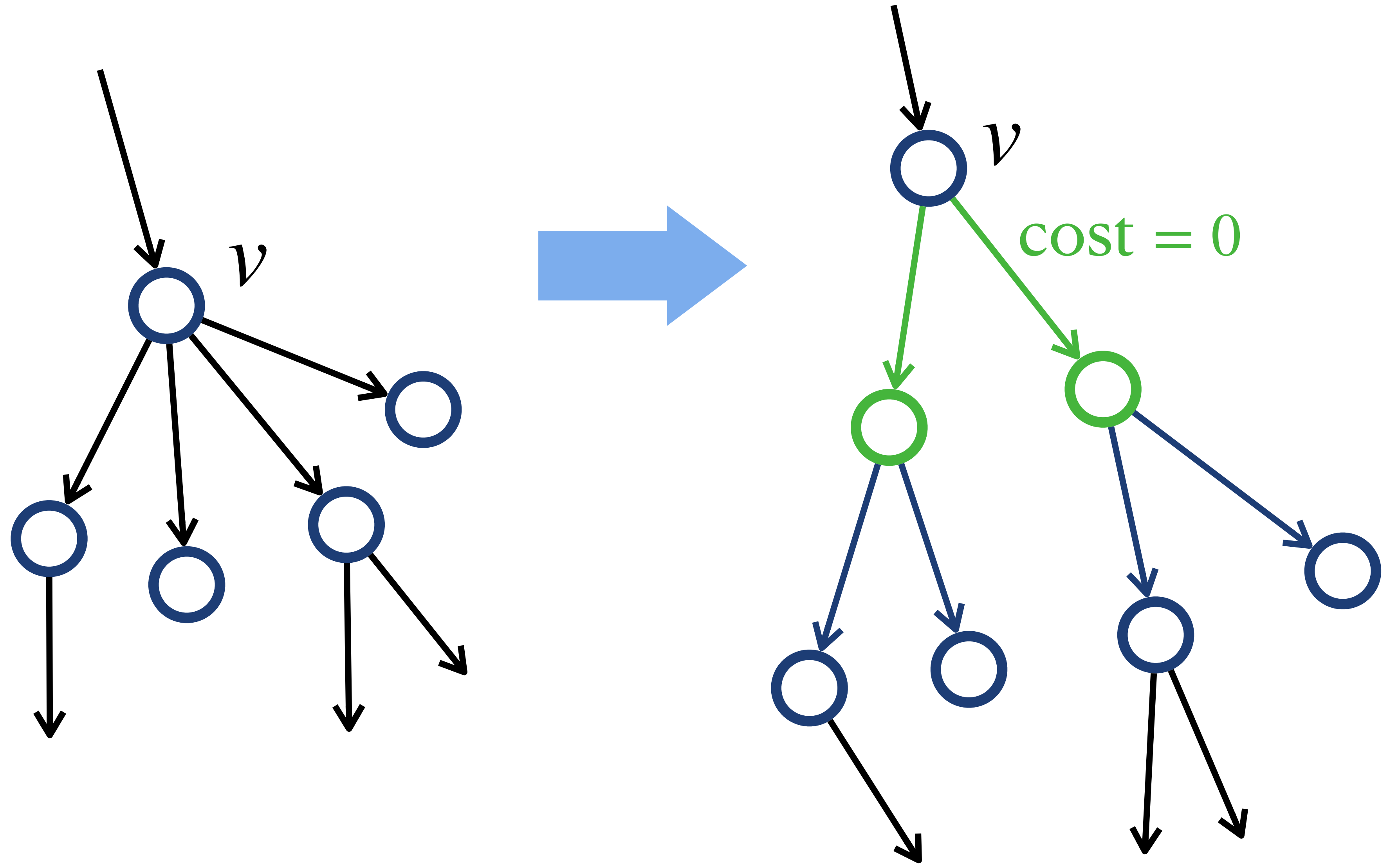
1. Encoding DSTs
   - Encoding as a *decomposition tree*
   - From decomposition trees to *state trees*
2. Handling degree bound
3. Rounding

Preprocessing

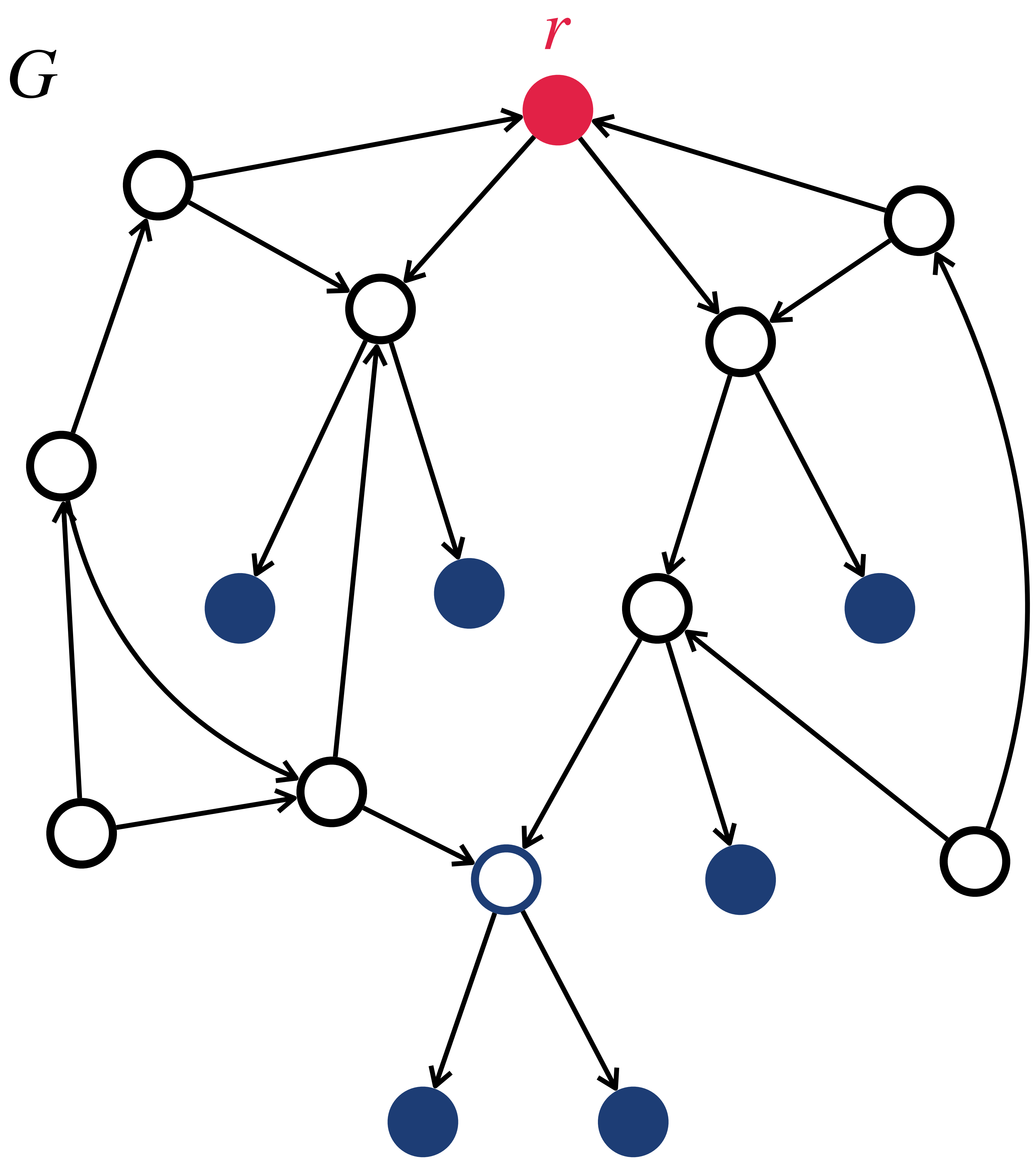
- Make every terminal v a leaf:



- Make every vertex v have out-degree ≤ 2

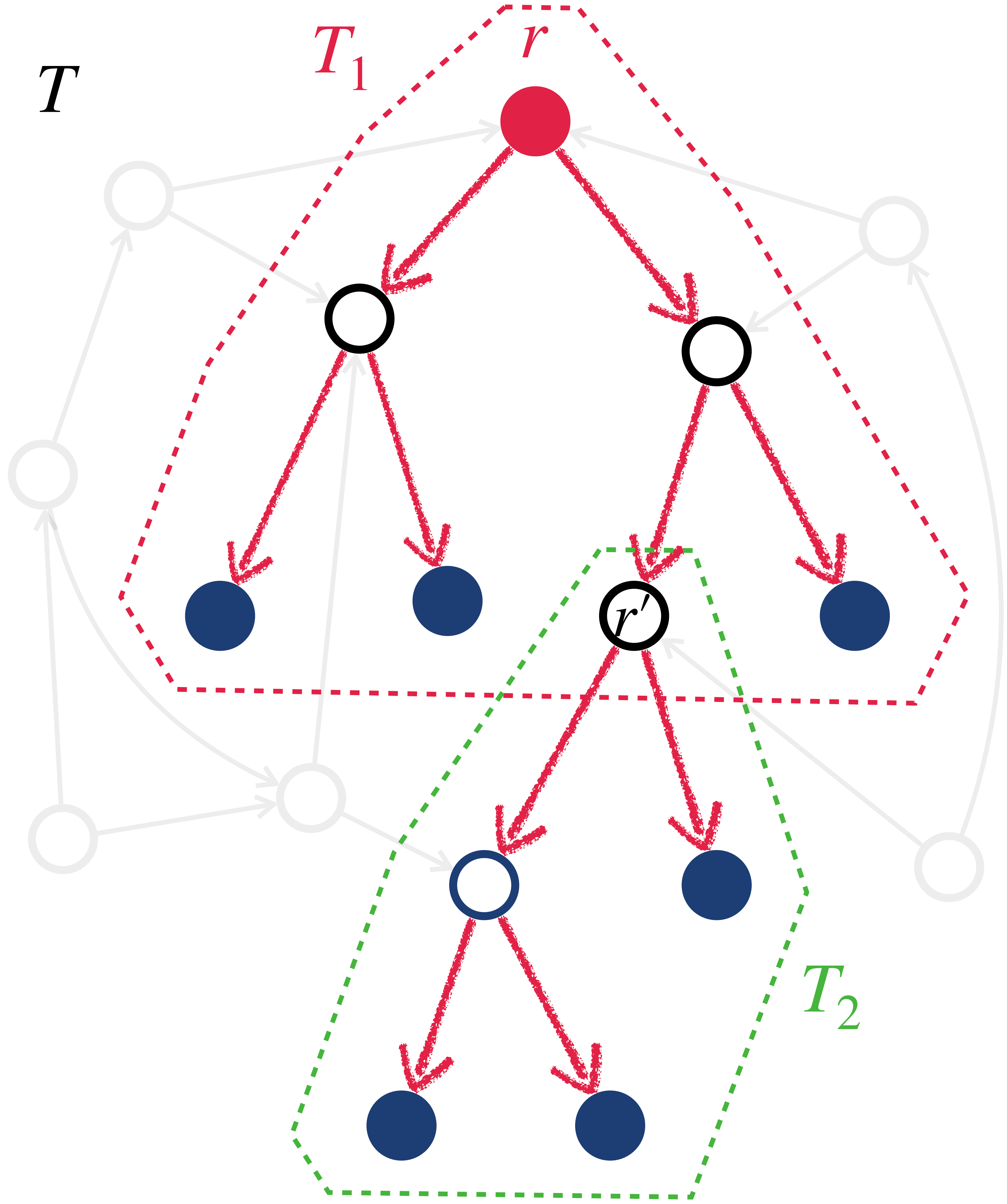


Decomposition Tree



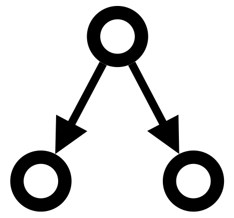
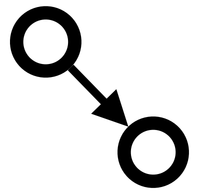
(● : terminal)

Decomposition Tree



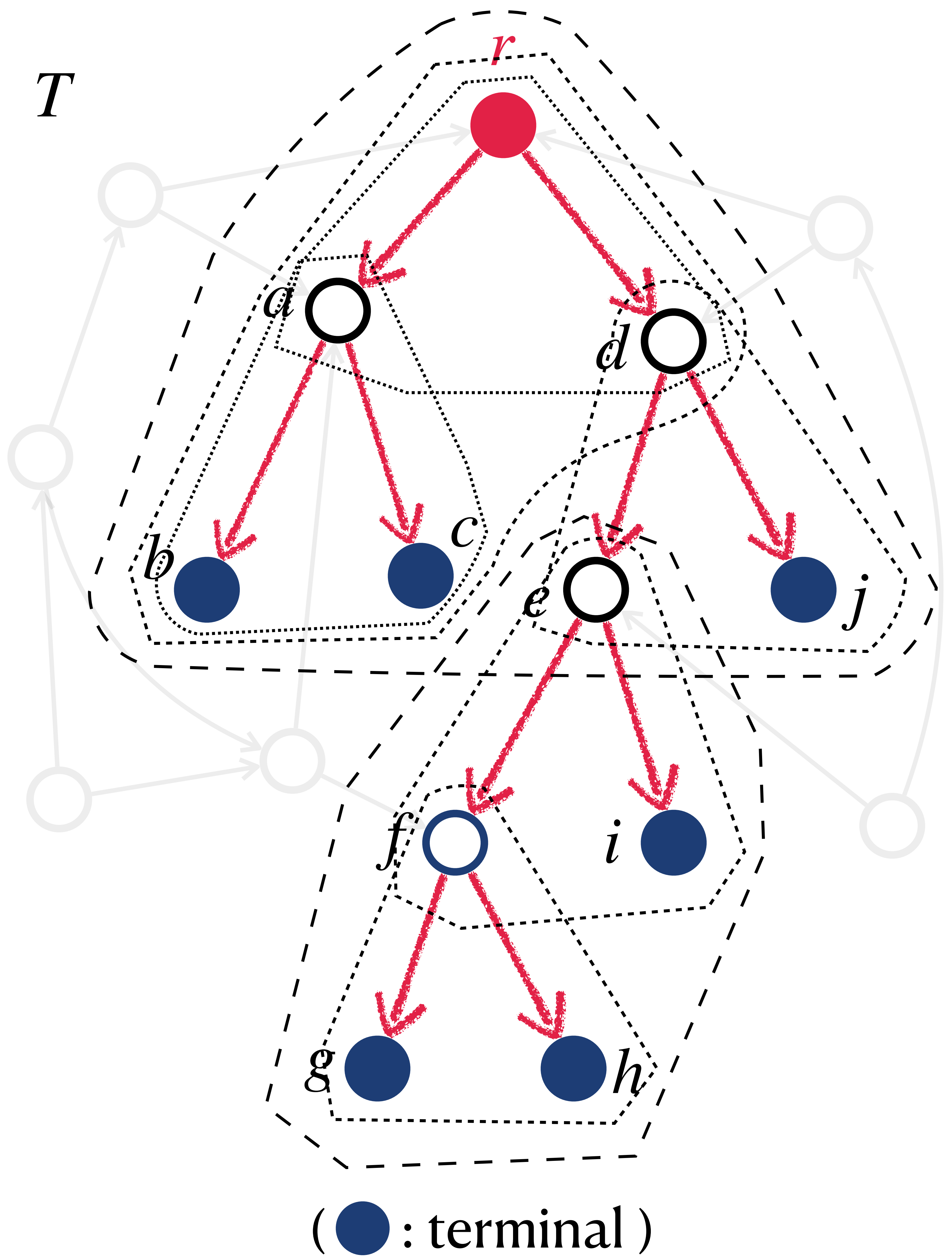
(● : terminal)

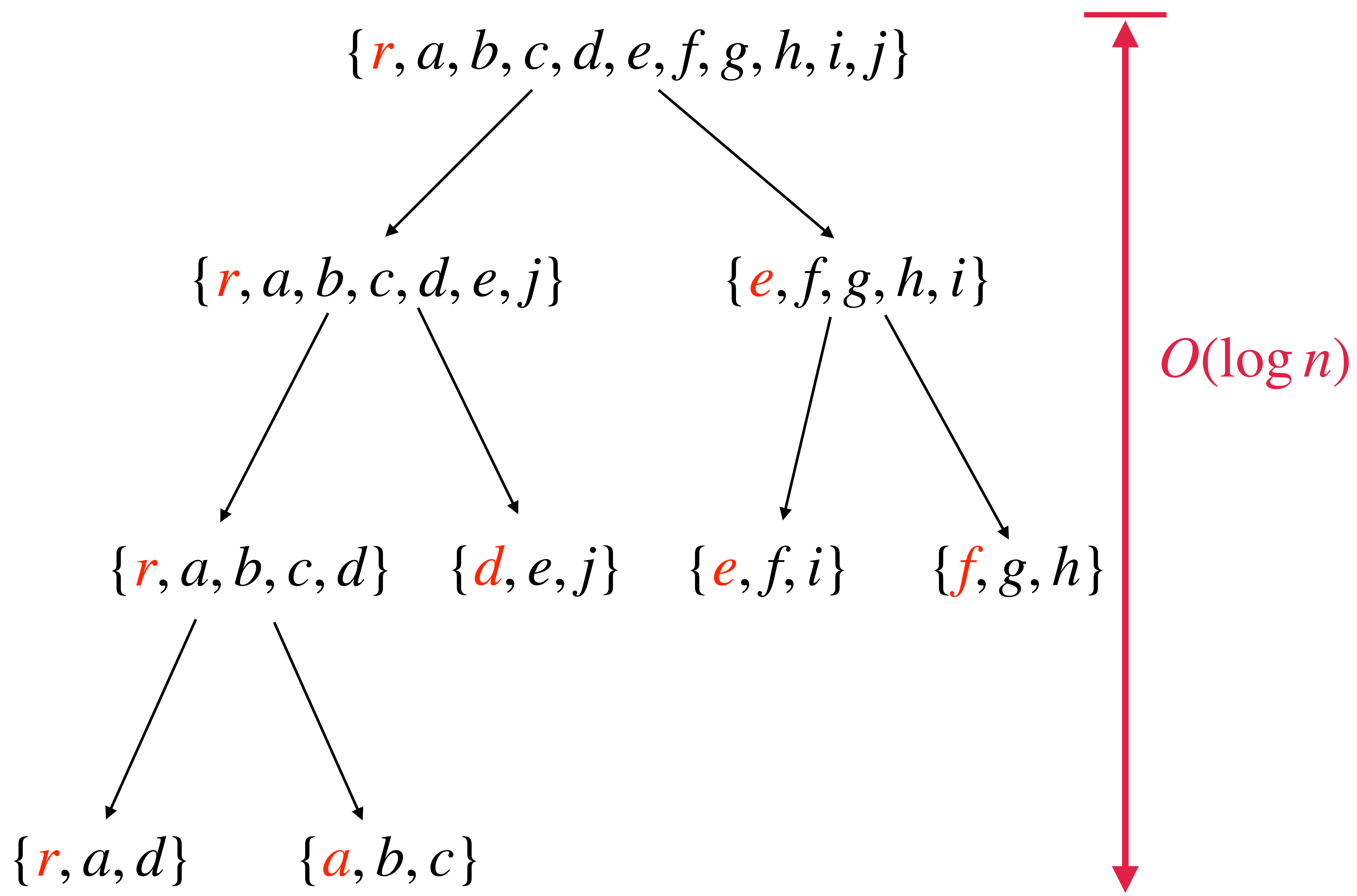
Balanced Partition Thm

For any n-vertex *binary* tree T that's not  or , we can split it into two subtrees T_1 and T_2 such that

- $T_1 \cup T_2 = T$
- $|T_1|, |T_2| < \frac{2}{3}n + 1$
- $|T_1 \cap T_2| = 1$,

Decomposition Tree





decomposition tree of T

Decomposition Tree

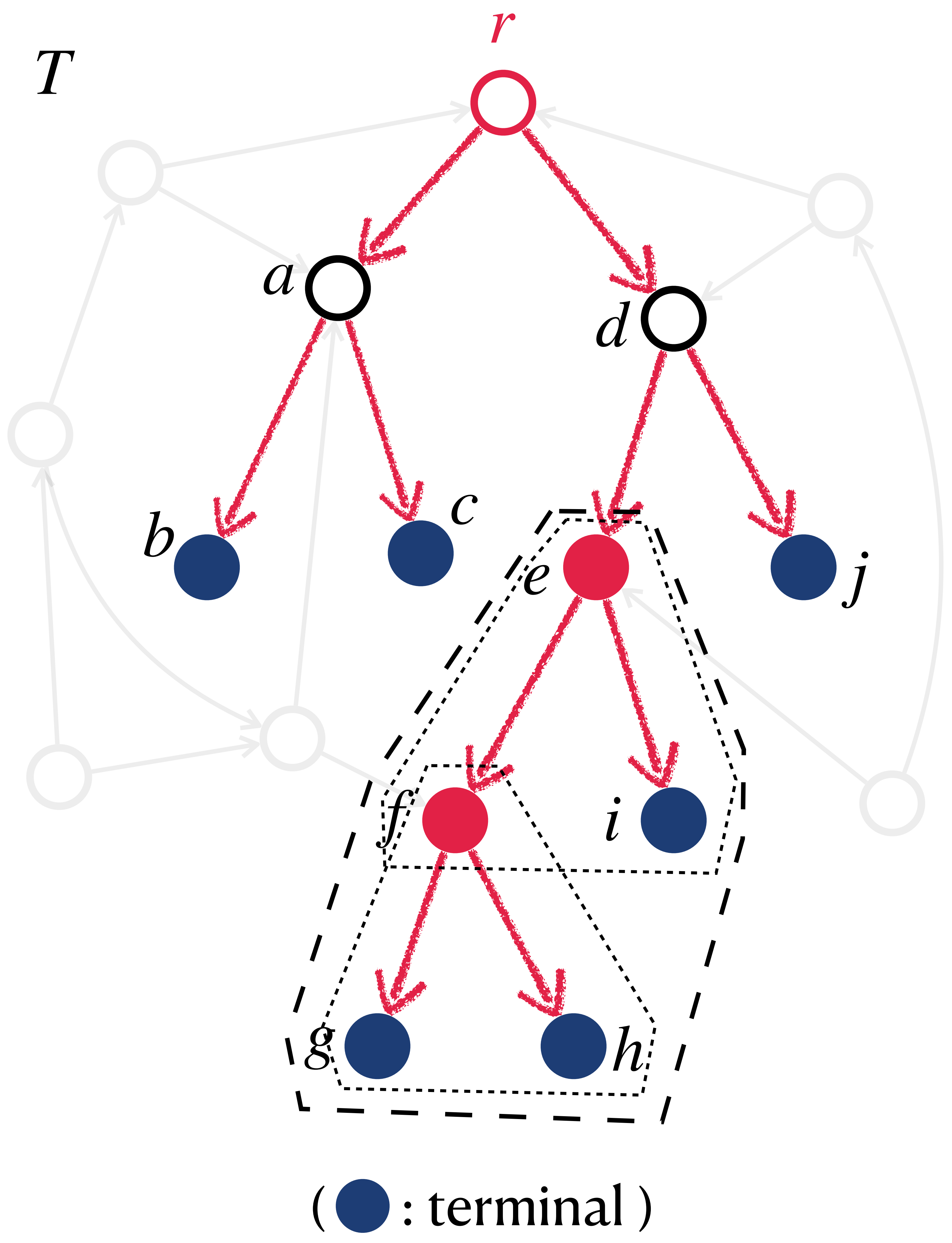
- An encoding of feasible DSTs
- Well-structured: $O(\log n)$-depth full binary tree
- Goal: find the decomposition tree encoding the optimal DST

Decomposition Tree

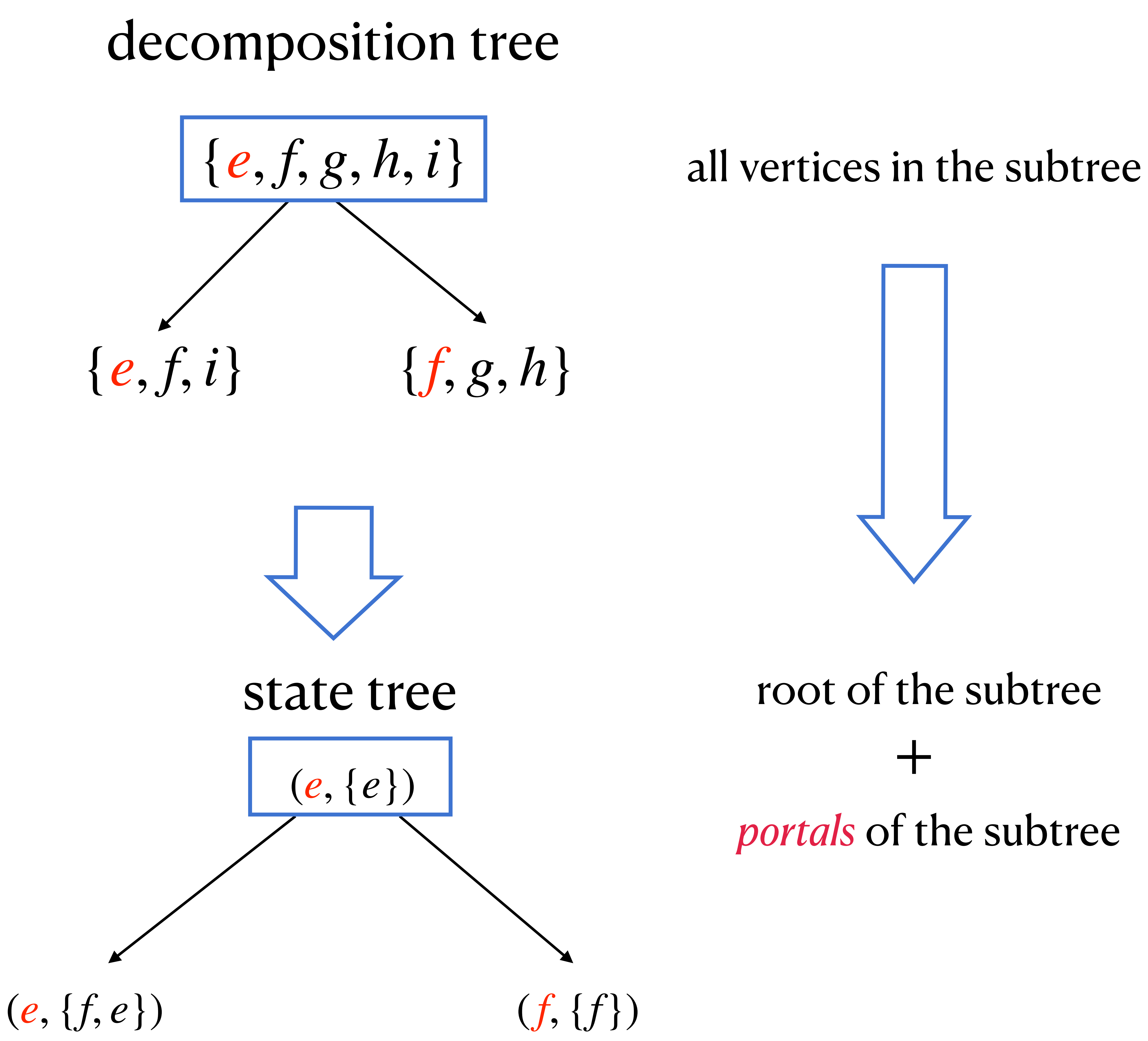
- An encoding of feasible DSTs
- Well-structured: $O(\log n)$-depth full binary tree
- Goal: find the ~~decomposition~~ state tree encoding the optimal DST

State tree: a more succinct (but lossy) encoding

State Tree

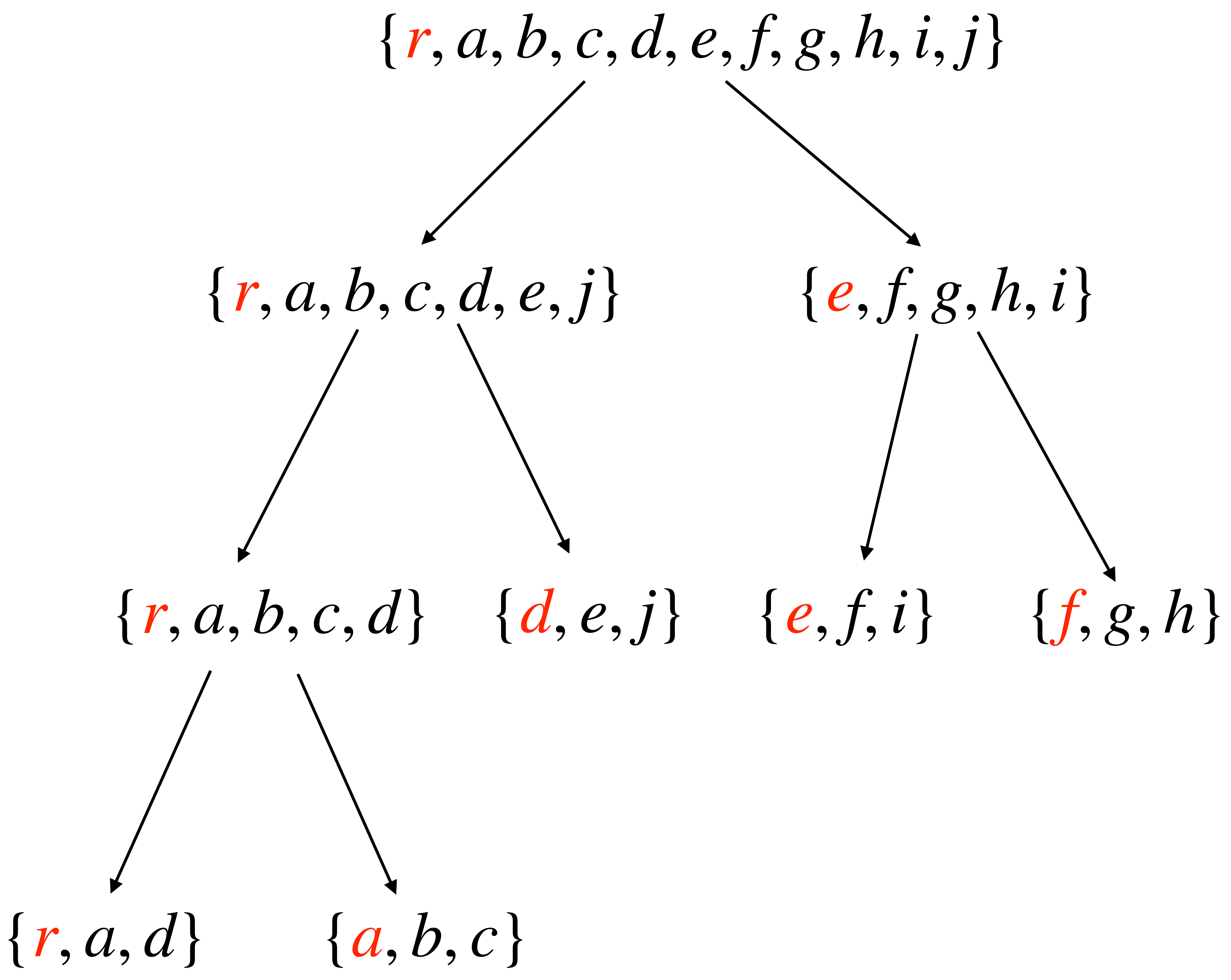


T
r
a
d
b
c
e
j
f
i
g
h
(: terminal)

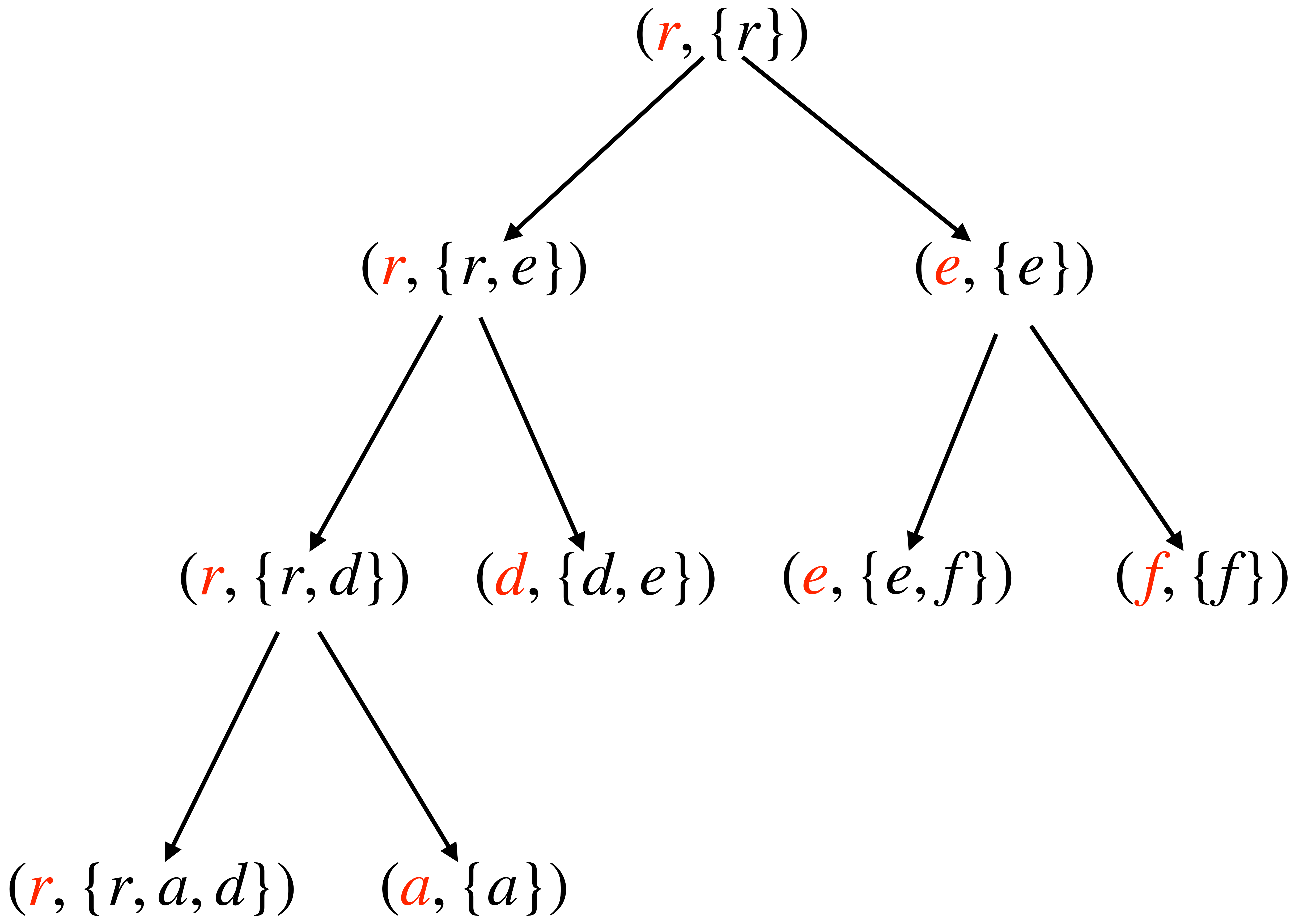


decomposition tree
$\{e, f, g, h, i\}$
$\{e, f, i\}$
$\{f, g, h\}$
all vertices in the subtree
state tree
$(e, \{e\})$
$(e, \{f, e\})$
$(f, \{f\})$
root of the subtree
+
portals of the subtree

State Tree



decomposition tree of T

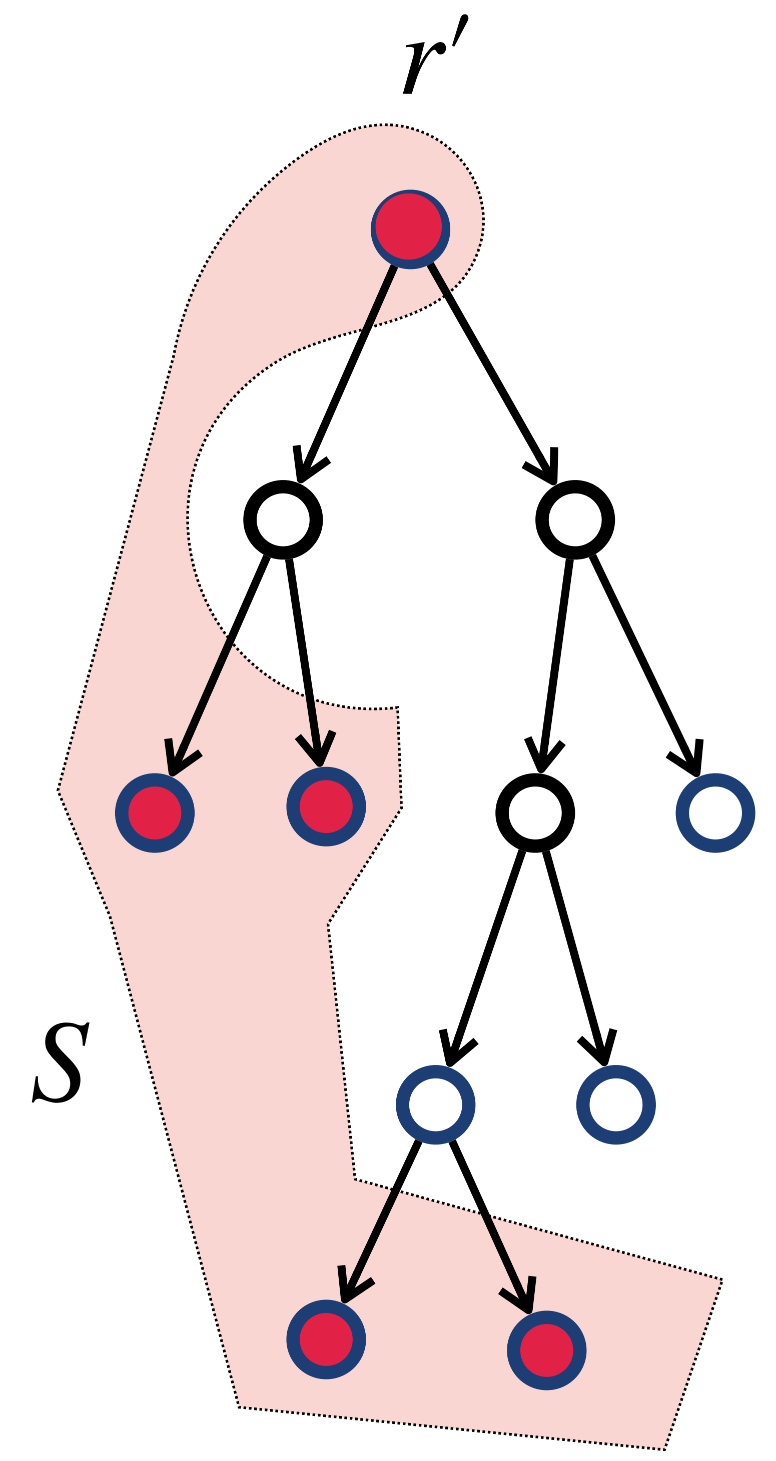


state tree of T

Obs: every node of the optimal state tree has at most $O(\log n)$ portals

Proof:

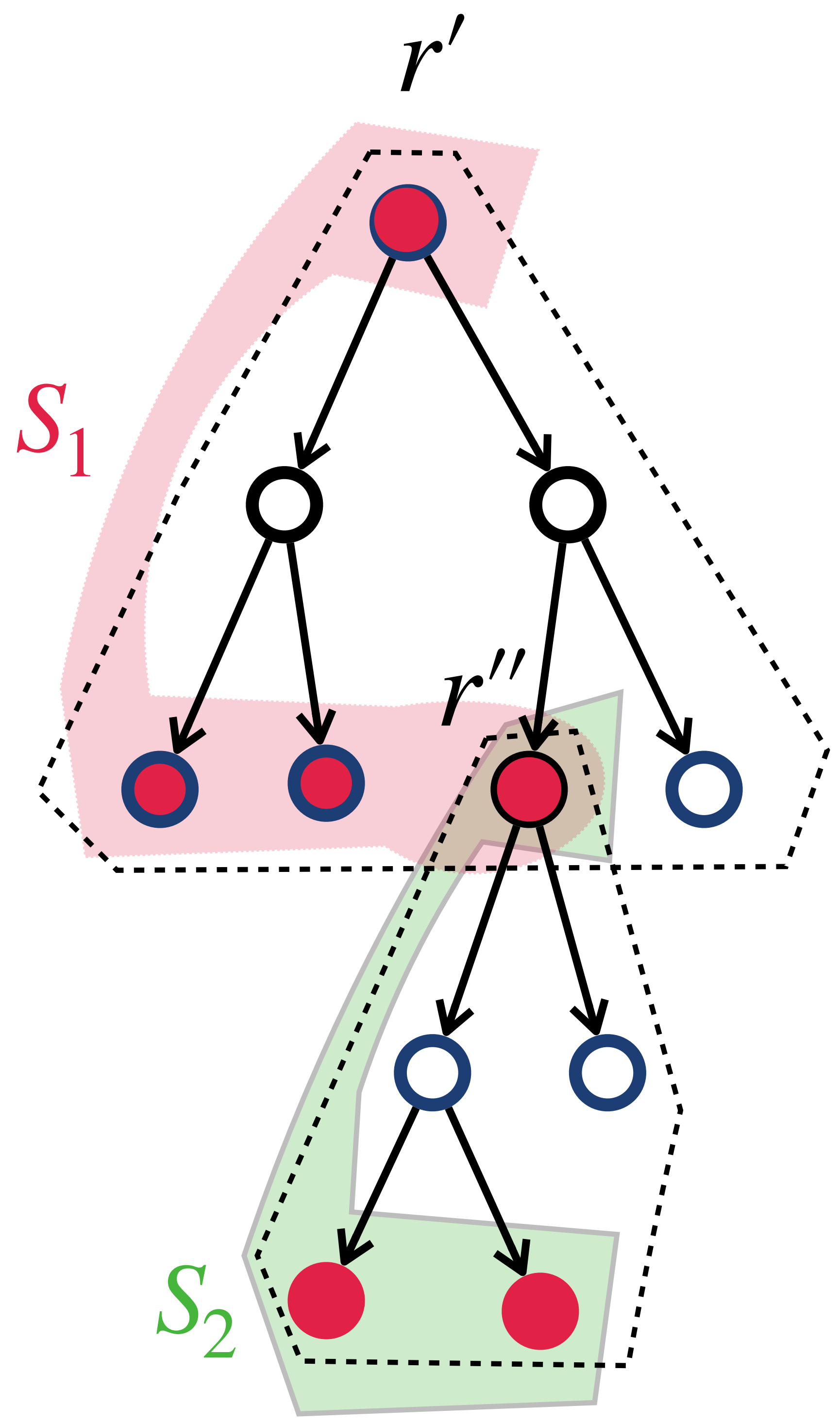
- Consider partitioning a subtree with state (r', S)



Obs: every node of the optimal state tree has at most $O(\log n)$ portals

Proof:

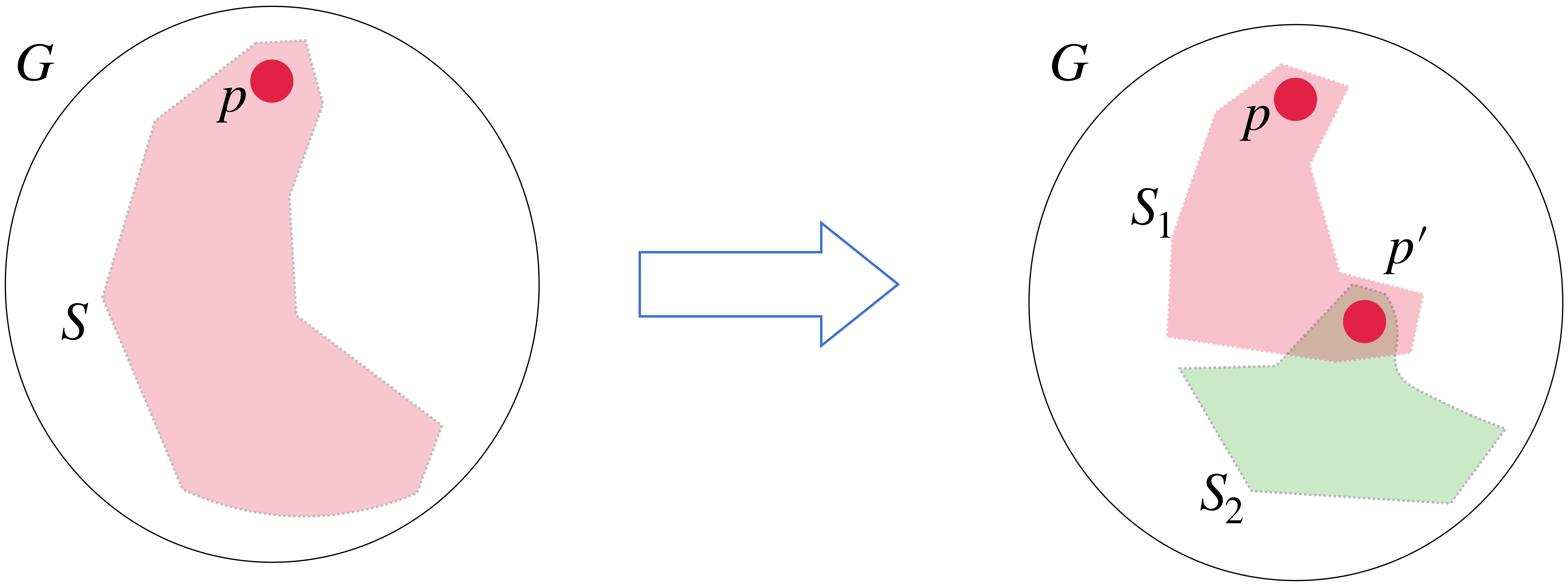
- Consider partitioning a subtree with state (r', S)
- Suppose we partition it at vertex r'' and get two subtrees (r', S_1) and (r'', S_2)
- Will introduce *one* new portal (r'') in each partition
- Recall the root state is $(r, \{r\})$, and the state tree is of depth $O(\log n)$. QED



Properties of the optimum state tree

- Root: state $(r, \{r\})$
- Depth: $O(\log n)$
- Simple state: $\forall$ state (p, S) in the tree, $|S| \leq O(\log n)$

Key idea: we can "enumerate" such state trees in quasi-polynomial time

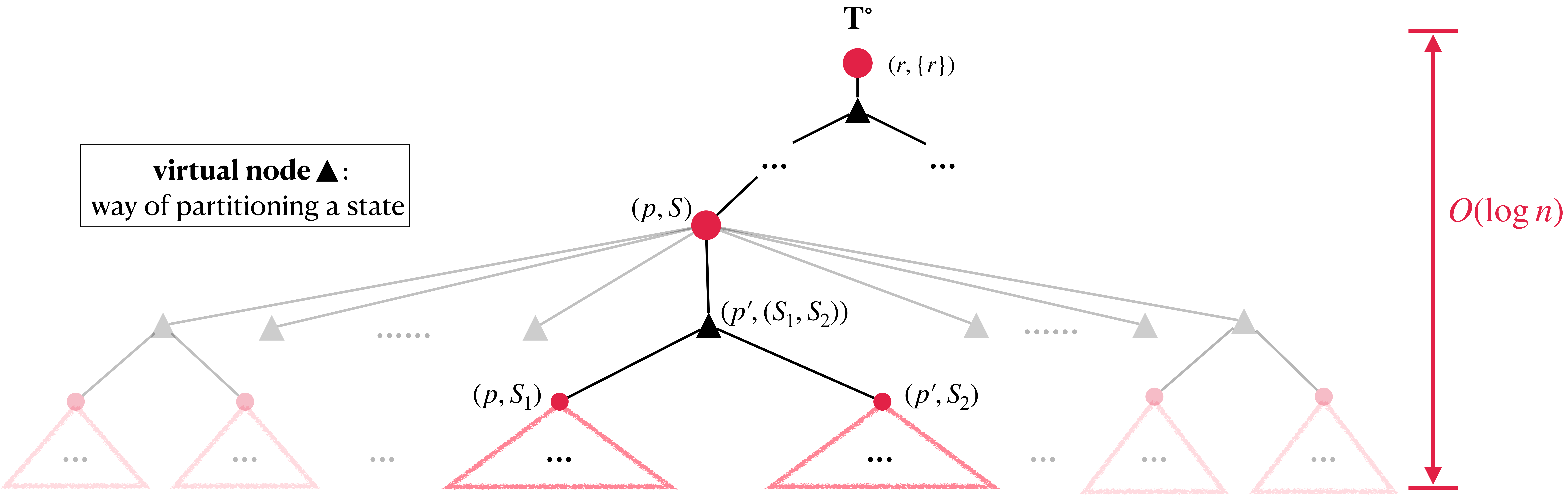


- **Question**: Number of possible ways to partition a state (p, S) ?
- **Ans** = #{ choices of p' } × #{ choice of (S_1, S_2) }

$$\leq \quad |V| \quad \times \quad 2^{|S|}$$

$$\leq \quad n \quad \times \quad 2^{O(\log n)} \quad = \quad \text{poly}(n)$$

- **Def:** Let $\mathbf{T}^\circ$ be the union of all possible state trees rooted at $(r, \{r\})$ with depth $O(\log n)$.
- Size of $\mathbf{T}^\circ = \text{poly}(n)^{O(\log n)} = n^{O(\log n)}$



- The optimal state tree is a subtree of $\mathbf{T}^\circ$
- For every $v \in \mathbf{T}^\circ$, let $x_v := \mathbf{1}[v \text{ in the optimal state tree}]$
- Can be captured by a LP of size $\leq \text{poly}(\text{size}(\mathbf{T}^\circ)) = n^{O(\log n)}$

$$\min_{x \in [0,1]^{\mathbf{V}^\circ}} \sum_{o\,:\,\text{base state}} x_o c(o)\,,$$

$$\sum_{q\,:\,\text{child of } p} x_q = x_p, \quad \forall \text{state node } p \quad (1)$$

$$\sum_{\substack{o\,:\,\text{base state involving } t \\ o \text{ is descendant of } p}} x_o \leq x_p, \quad \forall p \in \mathbf{T}^\circ, t \in K \quad (2)$$

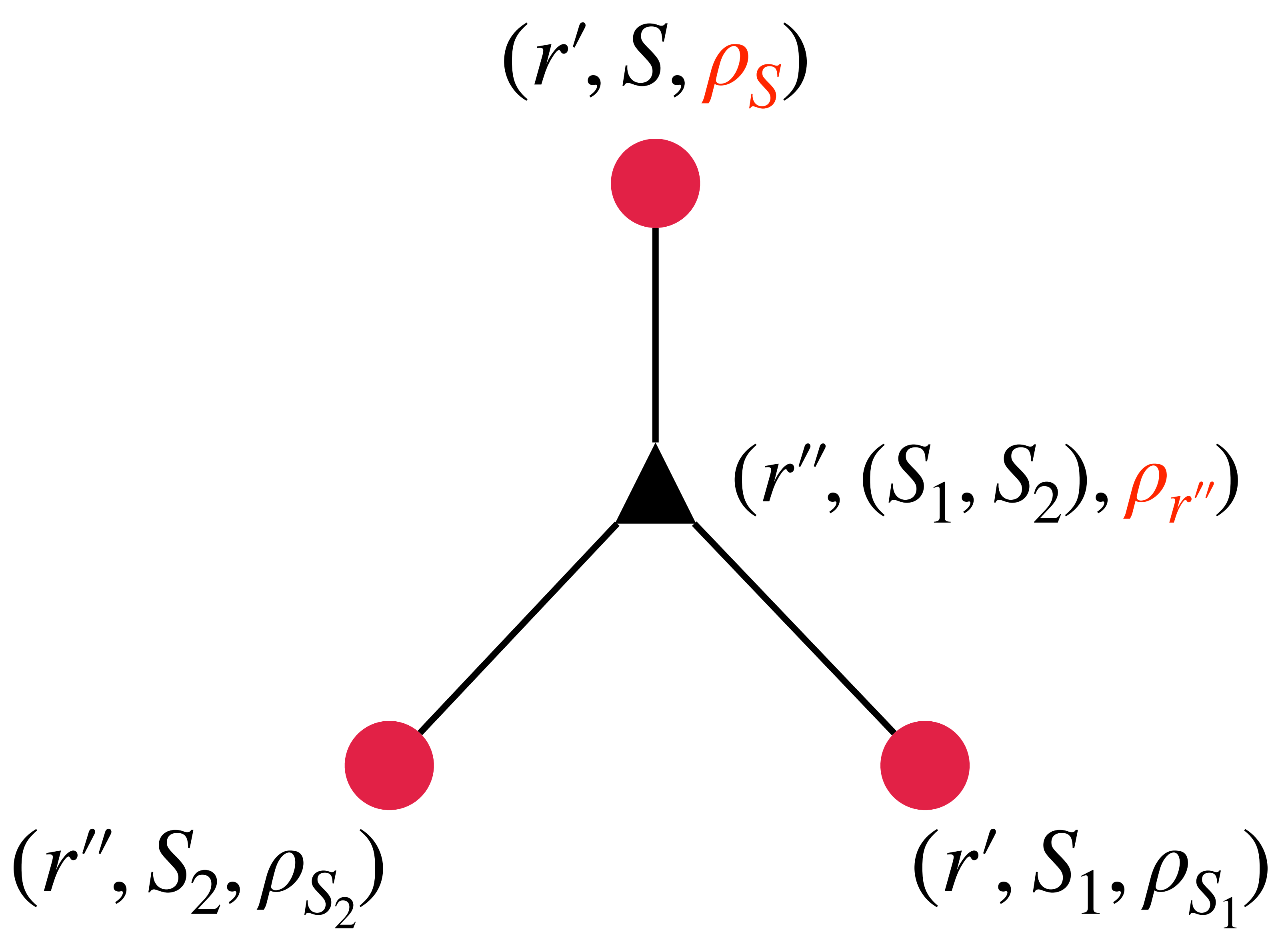
$$x_p = x_q, \quad \forall \text{virtual node } q, p \text{ child of } q \quad (3)$$

$$\sum_{o\,:\,\text{base state involving } t} x_o = 1, \quad \forall t \in K \quad (4)$$

Handling the degree bound

- What about the degree bound?
- **Ans:** add degree information to states

State node
root of the subtree
+
set of portals
+
out-degree of each portal



Virtual node
new portal
+
portal set partition
+
out-degree of the new portal

Handling the degree bound

- **Question**: Number of possible ways to partition a state (p, S, ρ_S) ?
- **Ans** = #{ choices of p' } $\times$ #{ choice of (S_1, S_2) } $\times$ #{ choices of $\rho_{p'}$ }

$$\leq \quad |V| \quad \times \quad 2^{|S|} \quad \times \quad d_{p'}$$

$$\leq \quad n \quad \times \quad 2^{O(\log n)} \quad \times \quad n \quad = \quad \text{poly}(n)$$

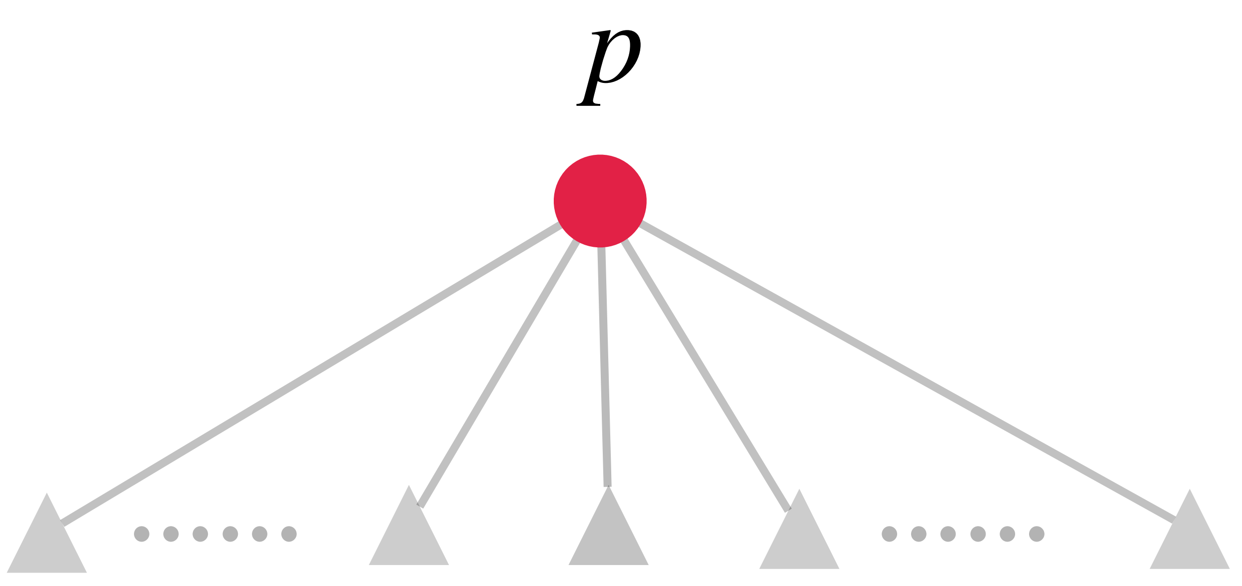
- Size of $\mathbf{T}^\circ \leq \text{poly}(n)^{O(\log n)} = n^{O(\log n)}$

Recursive rounding

- Let $\{x_v\}_{v \in \mathbf{T}^\circ}$ be the LP solution

Alg round(p)

- **if** p is state node:
  - pick child q of p with probability x_q/x_p
  - **return** $\{p\} \cup$ round(q)
- **else if** p is a virtual node:
  - **return** $\{p\} \cup$ round(left child of p) $\cup$ round(right child of p)
- **else return** $\{p\}$

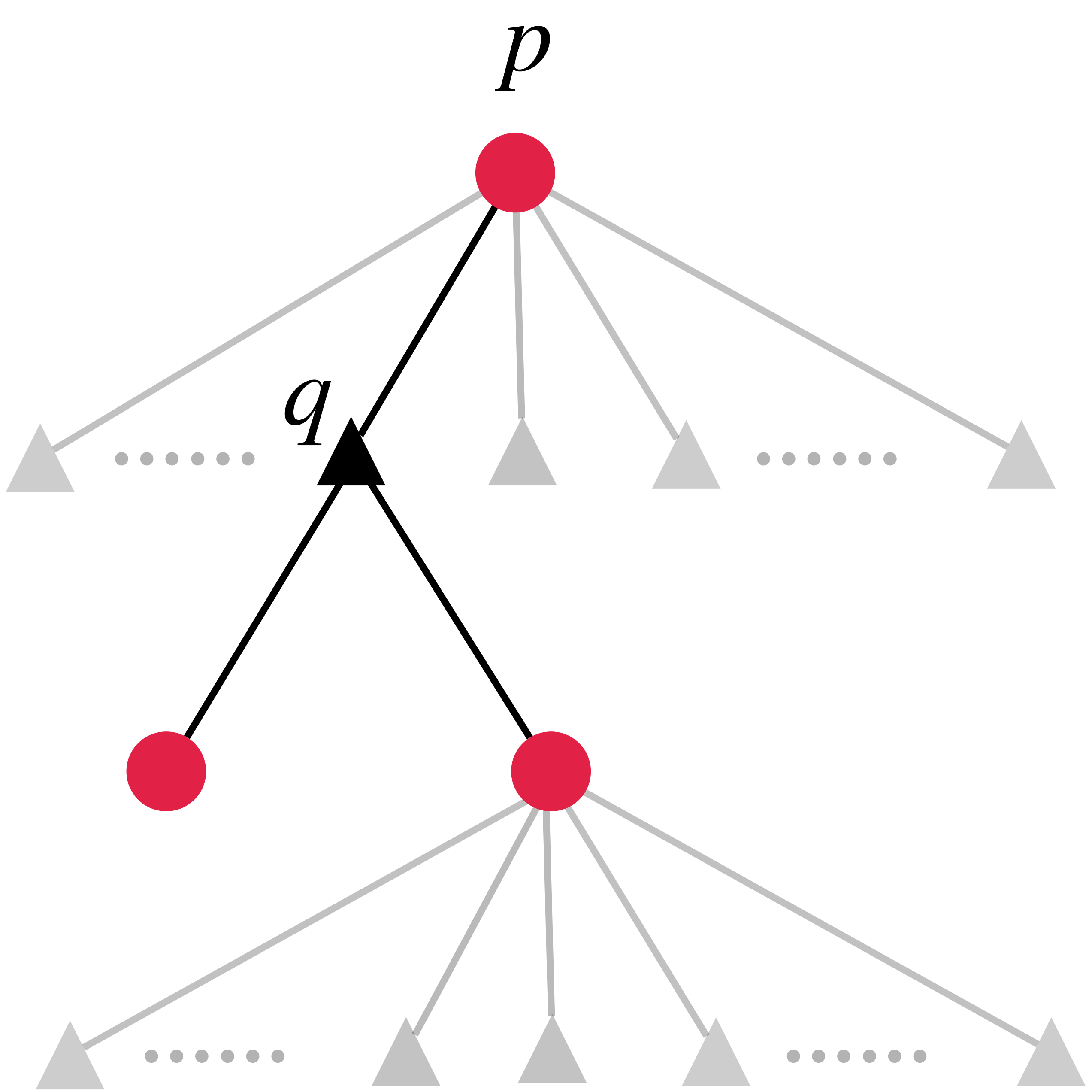


Recursive rounding

- Let $\{x_v\}_{v \in \mathbf{T}^\circ}$ be the LP solution

Alg round(p)

- **if** p is state node:
  - pick child q of p with probability x_q / x_p
  - **return** $\{p\} \cup$ round(q)
- **else if** p is a virtual node:
  - **return** $\{p\} \cup$ round(left child of p) $\cup$ round(right child of p)
- **else return** $\{p\}$

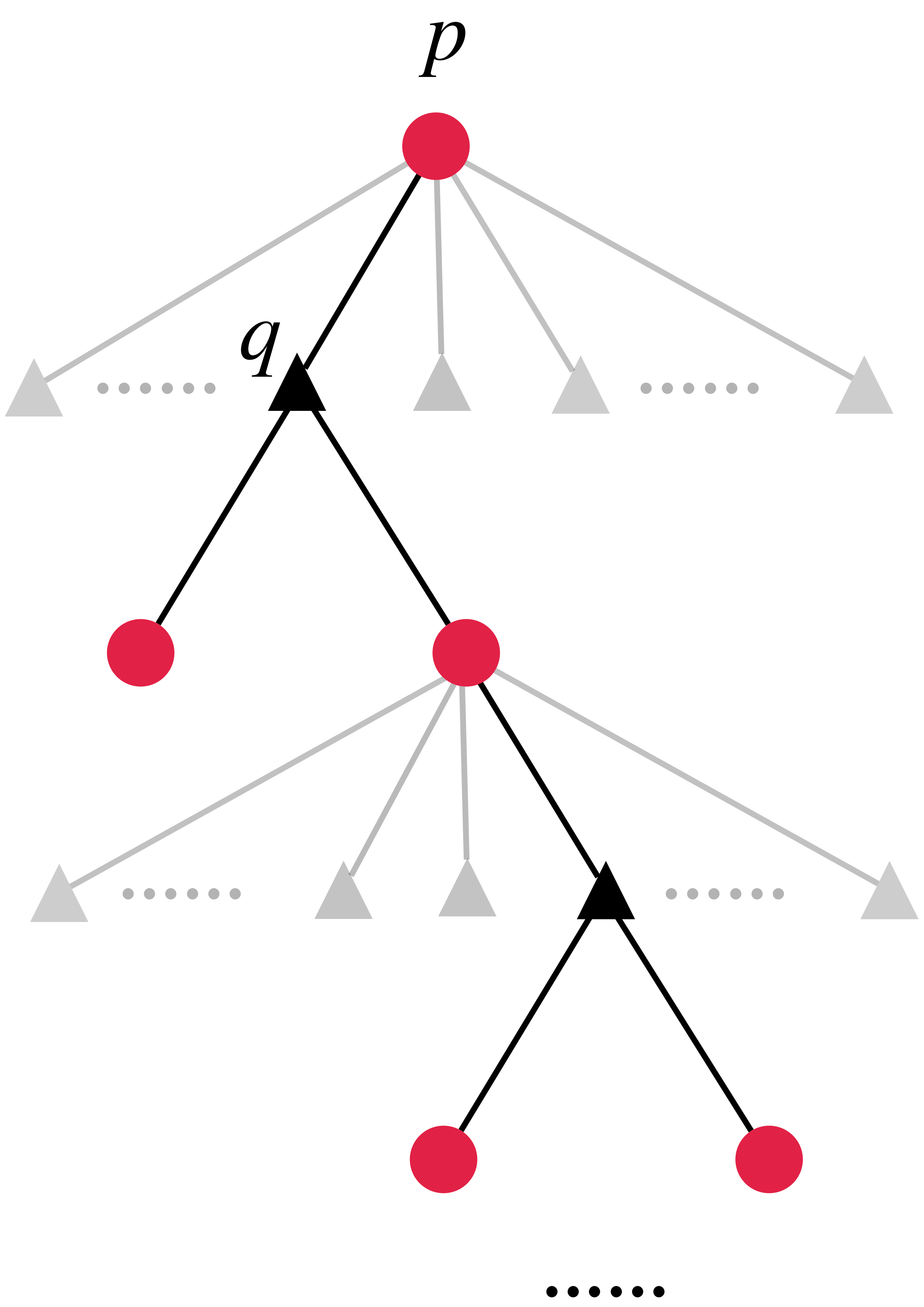


Recursive rounding

- Let $\{x_v\}_{v \in \mathbf{T}^\circ}$ be the LP solution

Alg round(p)

- **if** p is state node:
  - pick child q of p with probability x_q / x_p
  - **return** $\{p\} \cup$ round(q)
- **else if** p is a virtual node:
  - **return** $\{p\} \cup$ round(left child of p) $\cup$ round(right child of p)
- **else return** $\{p\}$



Recursive rounding

- Let $\mathbf{r} \leftarrow$ root of $\mathbf{T}^{\circ}$, $\tau \leftarrow \text{round}(\mathbf{r})$
- **Thm 1** [GKR'00]: Let T_0 be the tree encoded by state tree τ, then
  - $\mathbb{E}[\text{cost}(T_0)] \leq \text{LP cost}$
  - $\forall\, v \in T_0, \deg^{+}_{T_0}(v) \leq d_v$
  - For every terminal $t \in K$, T_0 connects t w.p. $\geq \Omega(1/\log n)$

Main algorithm

- Let $Q = O(\log n \log k)$
- For $i \leftarrow 1 \ldots Q$:
  - $\tau_i \leftarrow \text{round}(\mathbf{r})$
  - $T_i \leftarrow$ tree encoded by τ_i
- return $T = T_1 \cup T_2 \cup \cdots T_Q$

Thm 2: W.p. ≥ 0.9, T connects all terminals, and each $v \in V$ appears in T for at most $O(\log^2 n)$ times.

Thm 1 [GKR'00]: Let T_0 be the tree encoded by state tree τ, then

- $\mathbb{E}[\text{cost}(T_0)] \leq$ LP cost
- $\forall\, v \in T_0, \deg^+_{T_0}(v) \leq d_v$
- For every terminal $t \in K$, T_0 connects t w.p. $\geq \Omega(1/\log n)$

$+$

Thm 2: W.p. ≥ 0.9, T connects all terminals, and each $v \in V$ appears in T for at most $O(\log^2 n)$ times.

$\Downarrow$

$\mathbb{E}[\text{cost}(T)] \leq \text{OPT} \cdot O(\log n \log k)$ and $\forall\, v \in T, \deg^+_T(v) \leq d_v \cdot O(\log^2 n)$

Summarize

- We give a randomized $(O(\log n \log k), O(\log^2 n))$-apx algorithm for the DB-DST problem with $n^{O(\log n)}$ running time.
- Generalizations:
  - The degree bound is handled by simple enumeration.
  - Applicable for constraints that can be enumerated in poly(n) time, e.g., length-bound, buy-at-bulk.
    - In particular, we can reproduce the result of [Ghuge-Nagarajan'20]

Thank you!