Centrality of Trees for Capacitated *k*-Center

Hyung-Chan An

École Polytechnique Fédérale de Lausanne

July 29, 2013

Joint work with Aditya Bhaskara & Ola Svensson
Independent work of Chandra Chekuri, Shalmoli Gupta & Vivek Madan

Network Location Problems

- Given a metric on nodes (called *servers* and *clients*)
  - Need to connect every client to a server
  - Need to choose a subset of servers to be used

---

k-center

k-median

facility location

Network Location Problems

- Given a metric on nodes (called *servers* and *clients*)
  - Need to connect every client to a server
  - Need to choose a subset of servers to be used

k-center	minimize *maximum* connection cost
k-median	
facility location	

Network Location Problems

- Given a metric on nodes (called *servers* and *clients*)
  - Need to connect every client to a server
  - Need to choose a subset of servers to be used

k-center	minimize *maximum* connection cost
k-median	minimize *average* connection cost
facility location	

Network Location Problems

- Given a metric on nodes (called *servers* and *clients*)
  - Need to connect every client to a server
  - Need to choose a subset of servers to be used

k-center	minimize *maximum* connection cost
k-median	minimize *average* connection cost
facility location	minimize *average* connection cost *opening cost* instead of hard budget

Network Location Problems

- Uncapacitated problems
  - Assumes an open server can serve unlimited # clients

	complexity-theoretic lower bound	approximation ratio
k-center	2	2
k-median	1.735	2.733
facility location	1.463	1.488

[Gonzales 1985] [Hochbaum & Shmoys 1985] [Jain, Mahdian & Saberi 2002] [Li & Svensson 2013] [Guha & Khuller 1999] [Li 2011]

Network Location Problems

- Capacitated problems

	complexity-theoretic lower bound	approximation ratio
k-center	3	O(1)
k-median	1.735	
facility location	1.463	5

[Cygan, Hajiaghayi & Khuller 2012] [Jain, Mahdian & Saberi 2002]
[Guha & Khuller 1999] [Bansal, Garg & Gupta 2012]

Bridging this discrepancy

- How does the capacity impact the problem structure?
- How can we use mathematical programming relaxations?

The problem

- Capacitated *k*-center
  - Very good understanding of the uncapacitated case
  - Reduced to a combinatorial problem on unweighted graphs

Problem

Given k and a metric cost c on V with vertex capacities L, choose k centers to open, along with an assignment of every vertex to an open center that:

- minimizes longest distance between a vertex & its server
- each open center v is assigned at most $L(v)$ clients

Main result

- Simple algorithm with clean analysis
  - Improvement in approximation ratio & integrality gap (9-approximation)
  - Tree instances

Reduction to unweighted graphs

- Guess the optimal solution value T
- Consider a graph G representing admissible assignments: G has an edge (u, v) iff $c(u, v) \leq T$

- Will either
  - certify that G has no feasible assignment
  - find an assignment that uses paths of length $\leq \rho$

  $\Rightarrow$ ρ-approximation algorithm

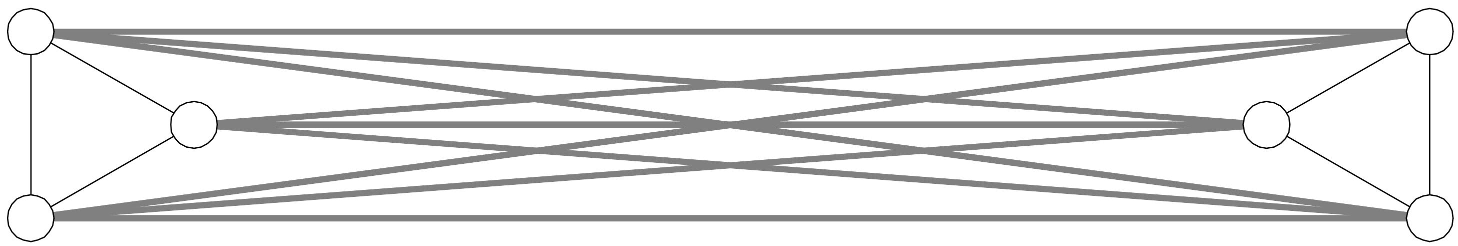
Standard LP relaxation

- Feasibility LP
- Assignment variables x_{uv}
- Opening variables y_u

$$
\begin{aligned}
\sum_{u \in V} y_u &= k; & \\
x_{uv} &\leq y_u, & \forall u, v \in V; \\
\sum_{v:(u,v) \in E} x_{uv} &\leq L(u) \cdot y_u, & \forall u \in V; \\
\sum_{u:(u,v) \in E} x_{uv} &= 1, & \forall v \in V; \\
0 \leq x, y &\leq 1. &
\end{aligned}
$$

Standard LP relaxation

- Unbounded integrality gap



k = 3, uniform capacity of 2

Standard LP relaxation

- Unbounded integrality gap

k = 3, uniform capacity of 2

Lemma (Cygan et al.)

It suffices to solve this combinatorial problem only for connected graphs.

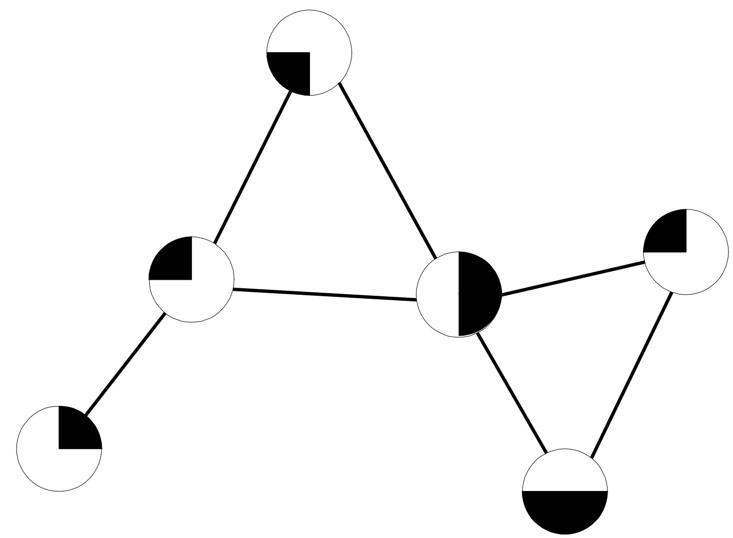
Outline

- Basic definitions
  - distance-*r* transfer
  - tree instance
- Solving a tree instance
- Applications
- Future directions

What does it mean to round an LP soln?

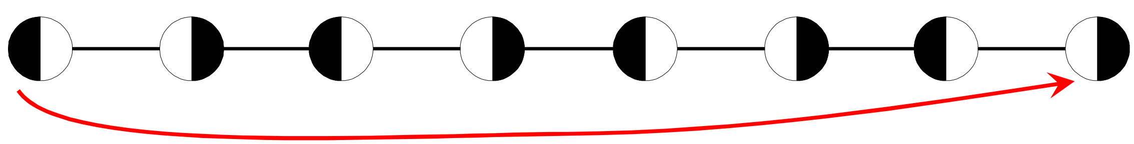
(x^*, y^*): LP solution

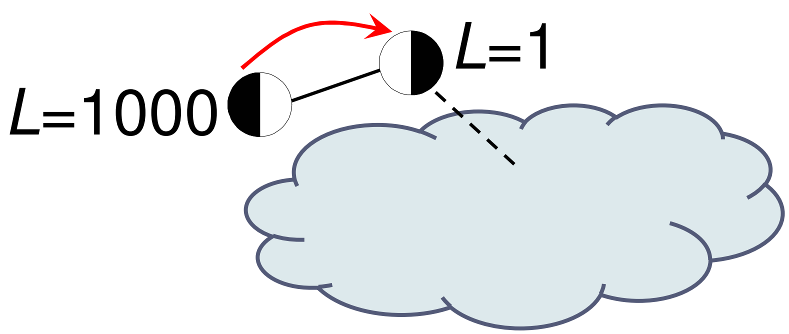
- y^* *fractionally* opens vertices
- If y^* integral, done
- We will “transfer” openings between vertices to make them integral
  - No new opening created
  - Need to ensure that a small-distance assignment exists



What does it mean to round an LP soln?

- We will “transfer” openings between vertices to make them integral
  - Need to ensure that a small-distance assignment exists
    - transfers in small vicinity
    - locally available capacity does not decrease





Distance-r transfer

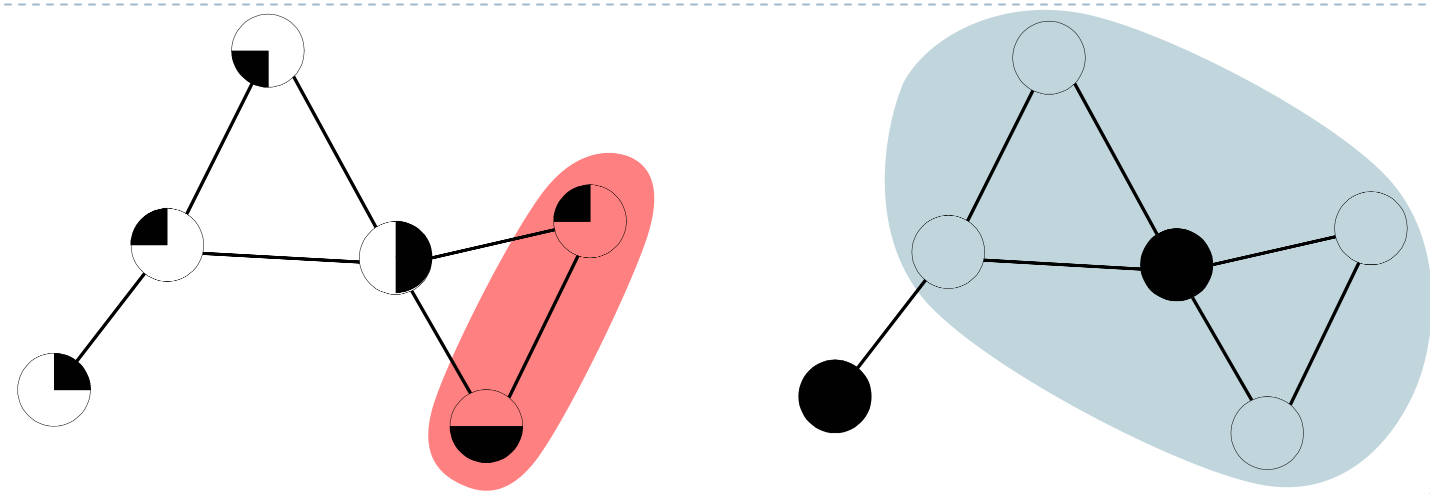
- Fractionally open vertex u has "fractional capacity" $L(u)y_u$
- Our rounding procedure "redistributes" these frac. cap.
- A distance-r transfer give a redistribution where *locally available capacity does not decrease*

Definition

y' is a distance-r transfer of y if

- $\sum_u y'_u = \sum_u y_u$
- $\sum_{u \in U} L(u) y_u \leq \sum_{v : d(v,U) \leq r} L(v) y'(v)$ for all $U \subset V$

Distance-r transfer



Dist-2 transfer
All cap. 4

Definition

y' is a distance-r transfer of y if

- $\sum_u y'_u = \sum_u y_u$
- $\sum_{u \in U} L(u) y_u \leq \sum_{v : d(v,U) \leq r} L(v) y'(v)$ for all $U \subset V$

Distance-r transfer

Lemma

If we can find a distance-8 transfer of an LP solution, we obtain a 9-approximation solution

Definition

y' is a distance-r transfer of y if

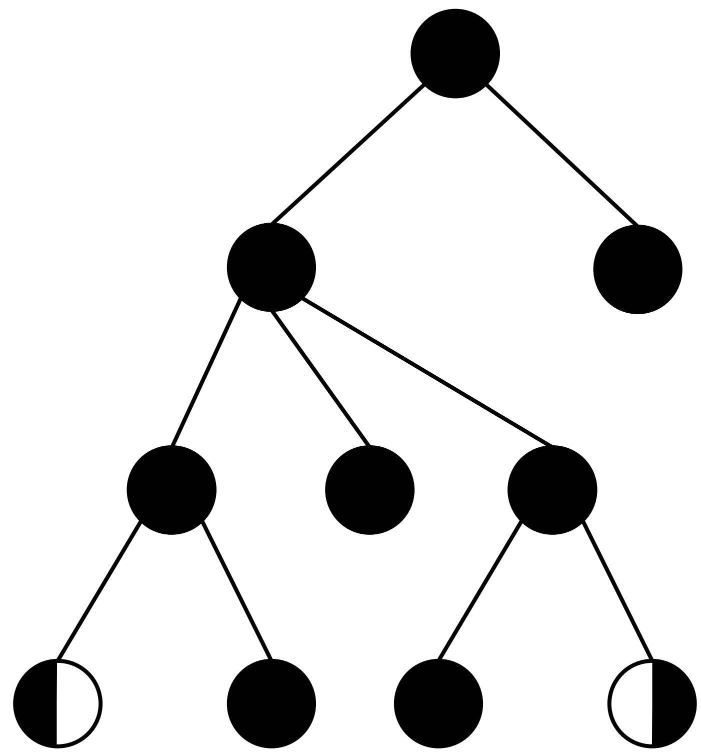
- $\sum_u y'_u = \sum_u y_u$
- $\sum_{u \in U} L(u) y_u \leq \sum_{v: d(v,U) \leq r} L(v) y'(v)$ for all $U \subset V$

Tree instance

Definition

A tree instance is a rooted tree of fractionally open vertices where every internal node *v* is fully open: i.e. $y_v = 1$

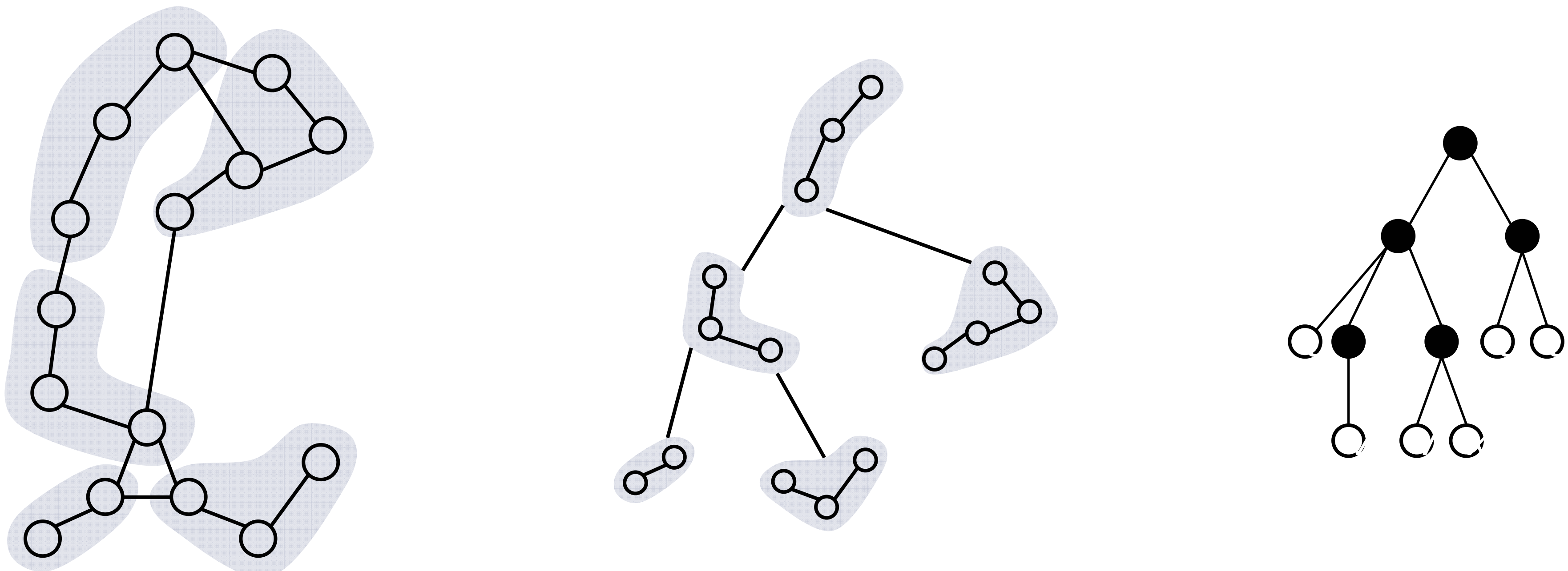
- Focusing on servers only
- Why is this interesting?



Reduction to a tree instance

Lemma (Khuller & Sussmann, informal)

A connected graph can be partitioned into small-diameter clusters



Reduction to a tree instance

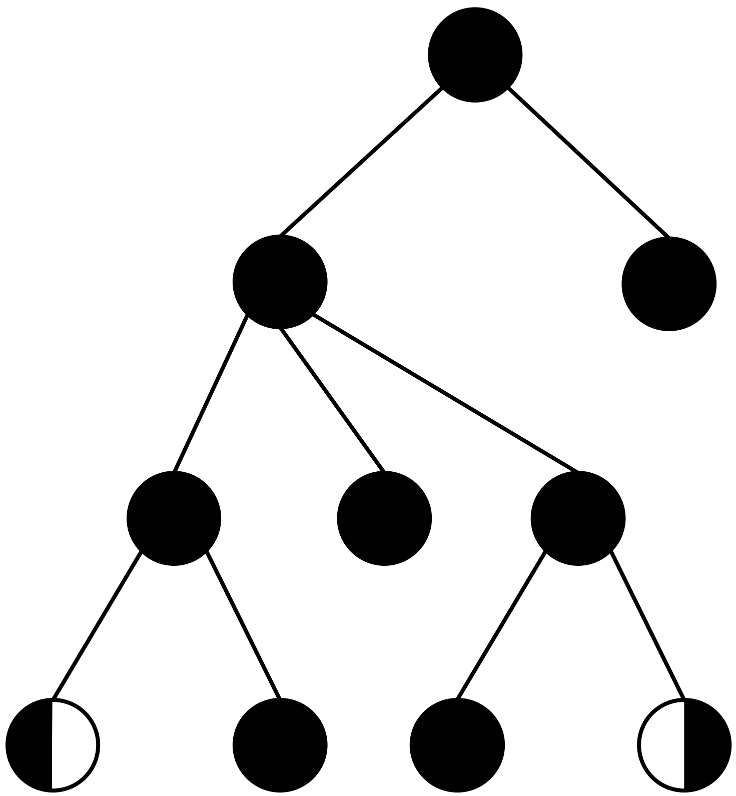
Lemma

If we can find an integral distance-r transfer of a tree instance, we obtain a $(3r+3)$-approximation algorithm for capacitated k-center

Want: distance-2 transfer of a tree instance

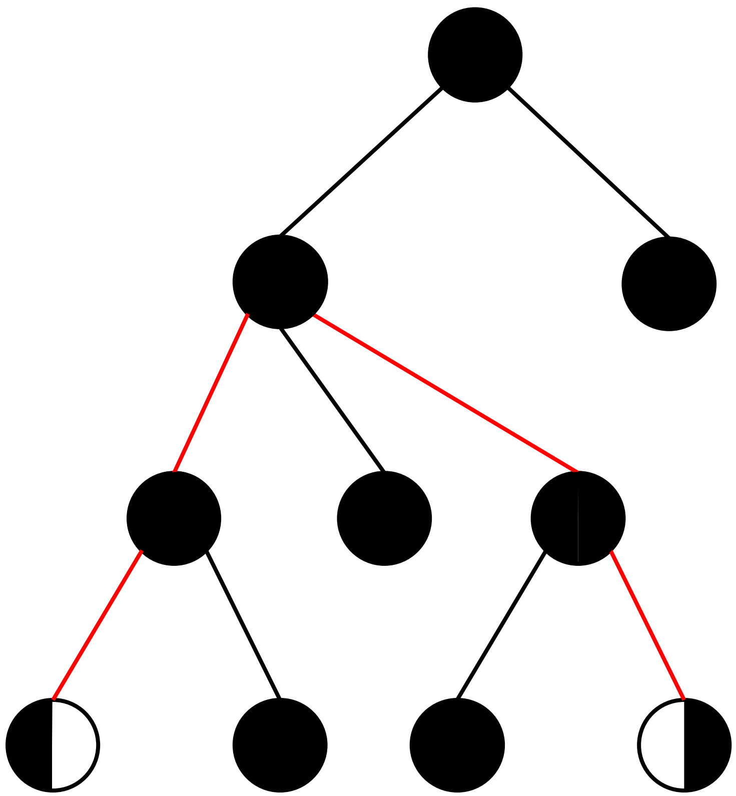
Solving a tree instance

- Example (uniform capacity)



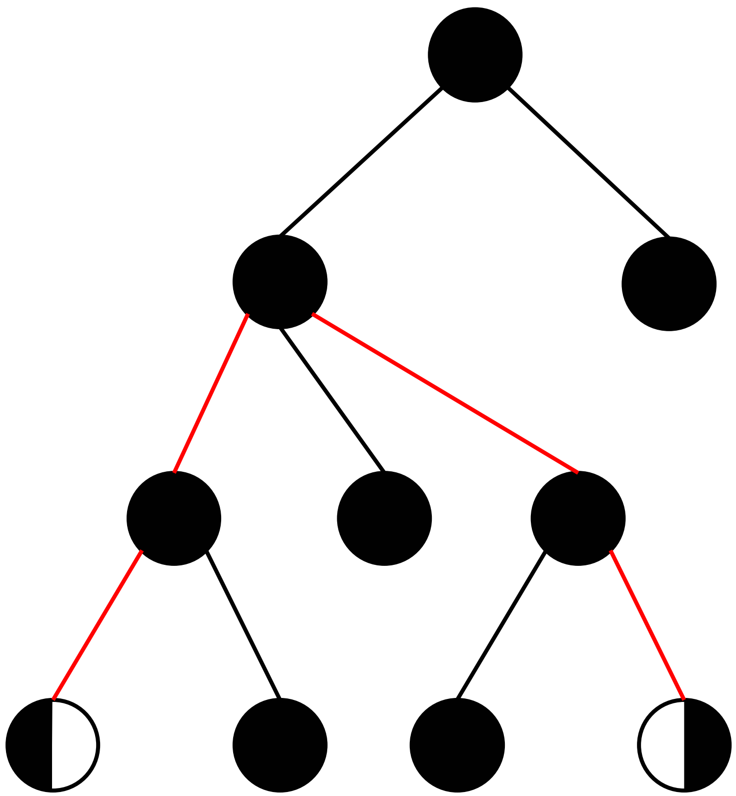
Solving a tree instance

- Example (uniform capacity)



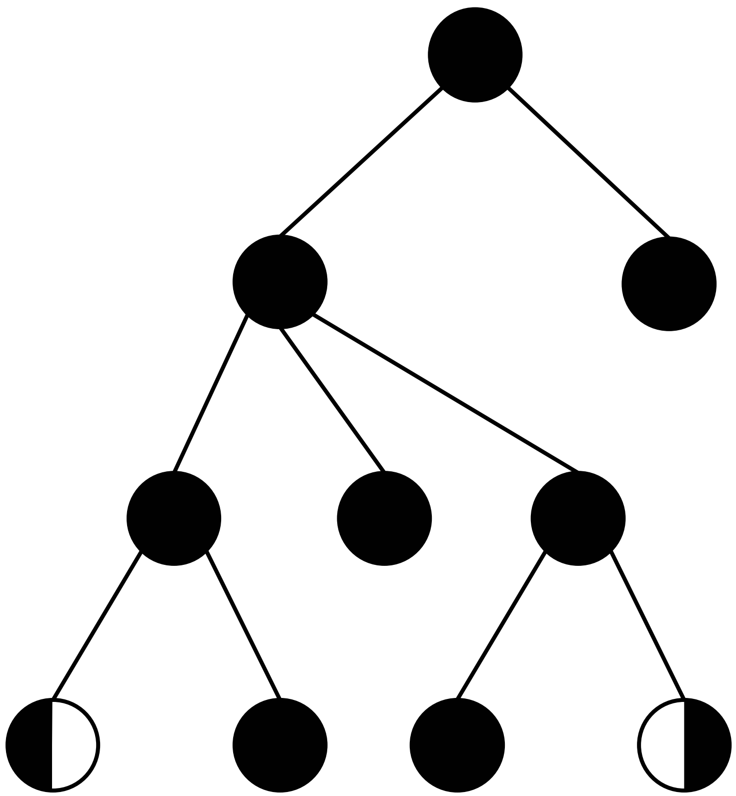
Solving a tree instance

- Example (the two nodes have capacity 10, others 1000)



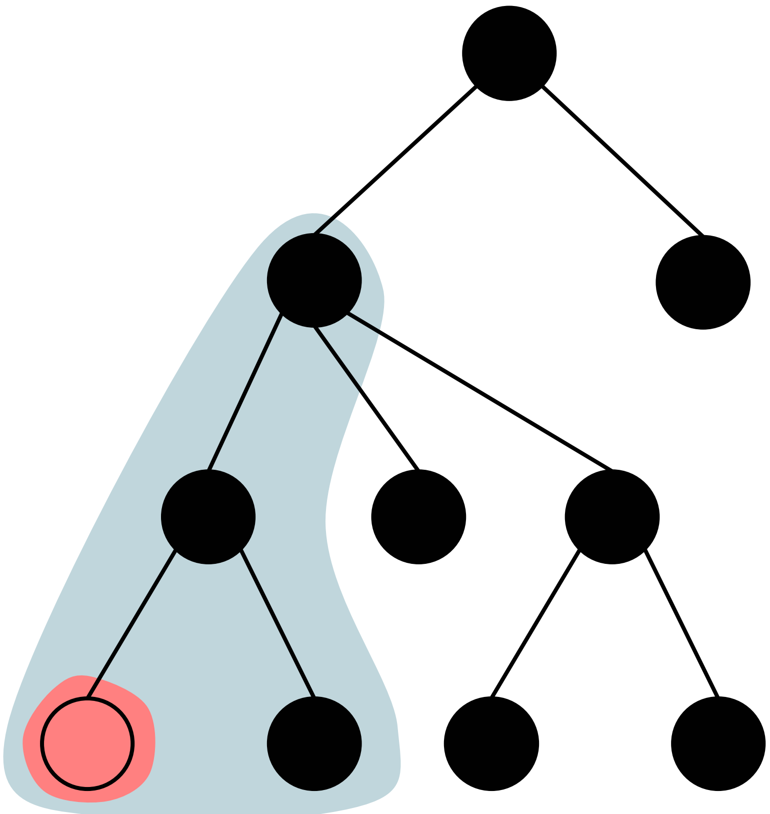
Solving a tree instance

- Example (the two nodes have capacity 1000, others 10)



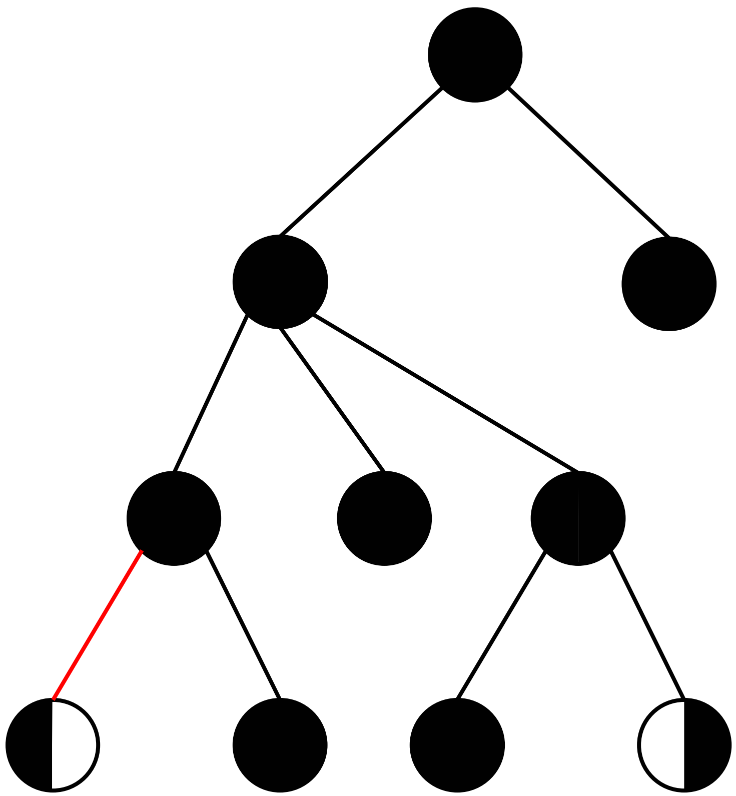
Solving a tree instance

- Example (the two nodes have capacity 1000, others 10)



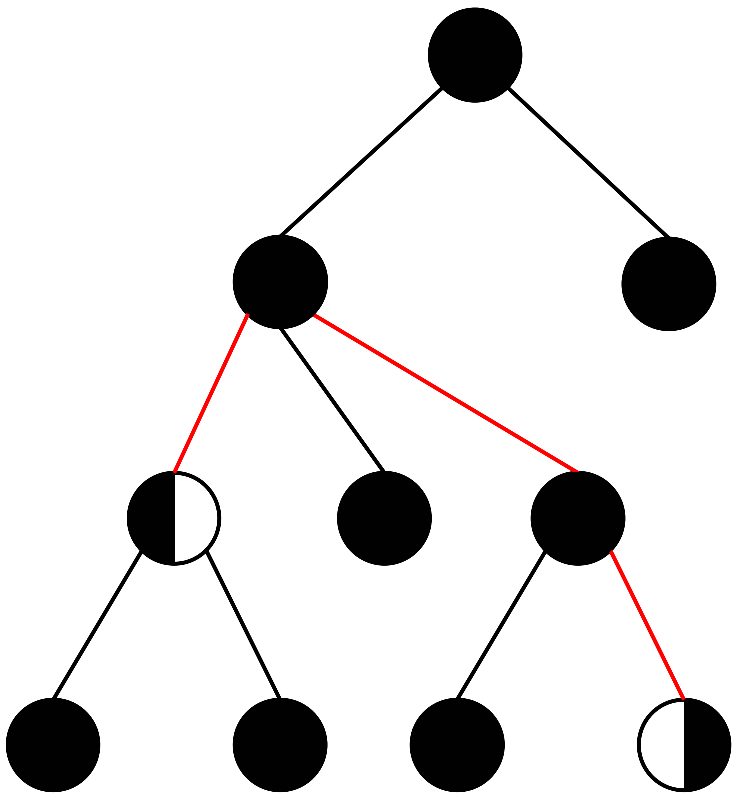
Solving a tree instance

- Example (the two nodes have capacity 1000, others 10)



Solving a tree instance

- Example (the two nodes have capacity 1000, others 10)

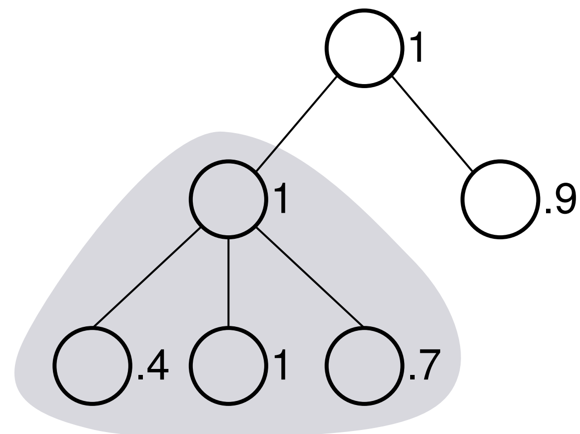


Solving a tree instance

- Closing a fully open center
  - Useful strategy; but its viability depends on the choice of open centers in the neighborhood
  - Our algorithm departs from previous approaches by using a simple *local* strategy for *every* internal node

Solving a tree instance

- Our algorithm
  - Locally round a height-2 subtree to obtain a smaller instance
  - Would want to open Y+1 centers in the subtree
    - Instead will open either $\lfloor Y \rfloor$+1 or $\lceil Y \rceil$+1 centers
    - Choose $\lfloor Y \rfloor$+1 centers and commit now to open them
    - Choose one additional candidate for which the decision is postponed

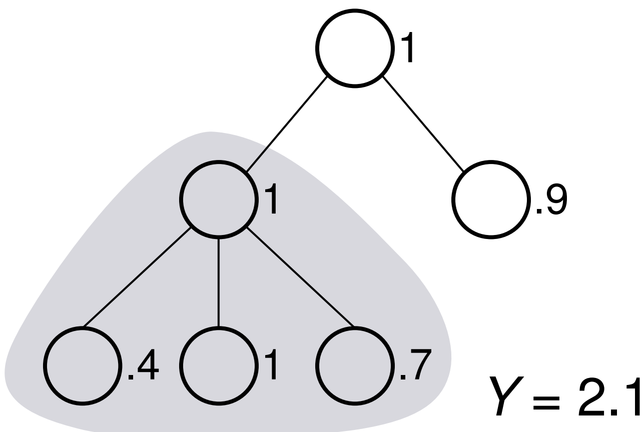


Y: total opening of children

(2.1)

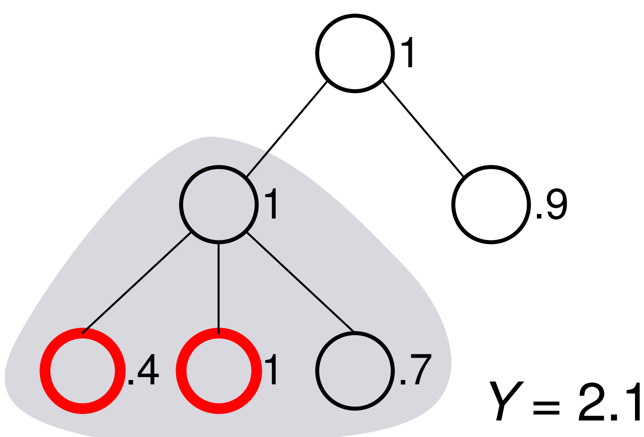
Solving a tree instance

- Our algorithm
  - $\lfloor Y \rfloor + 1$ centers to commit



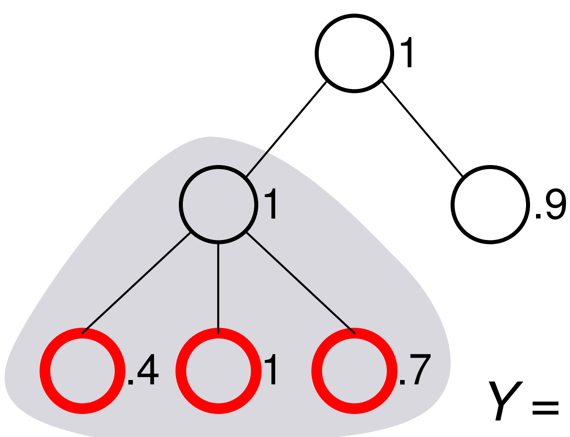
Solving a tree instance

- Our algorithm
  - $\lfloor Y \rfloor + 1$ centers to commit
    - Choose $\lfloor Y \rfloor$ children of highest capacities



Solving a tree instance

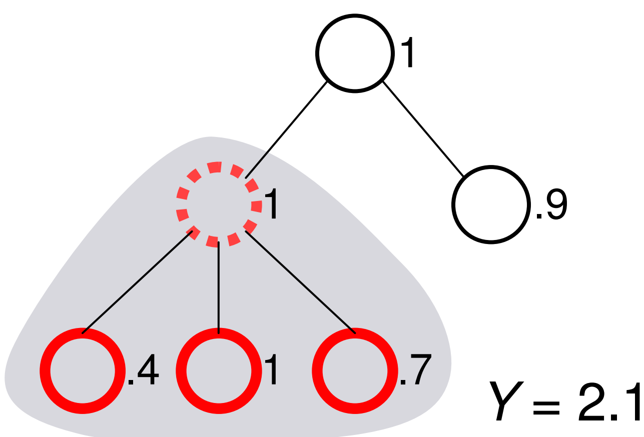
- Our algorithm
  - $\lfloor Y \rfloor + 1$ centers to commit
    - Choose $\lfloor Y \rfloor$ children of highest capacities
    - Between the next highest and the subtree root, choose the higher capacity



$Y = 2.1$

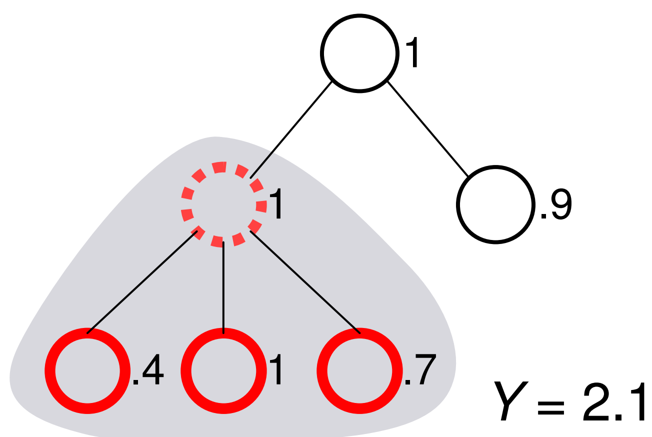
Solving a tree instance

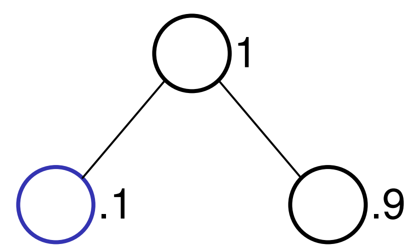
- Our algorithm
  - Additional candidate
    - Would want to fractionally open the other node by $Y - \lfloor Y \rfloor$
    - This node becomes the candidate



Solving a tree instance

- Our algorithm
  - Contract the subtree, replaced with a new node with
    - Capacity equal to the candidate
    - Opening $Y - \lfloor Y \rfloor$
  - Recursively solve the new instance; if the new node gets opened, the candidate gets opened





Solving a tree instance

- Our algorithm
  - Choose highest capacity children, as many as allowed
  - Choose one more: root or next highest child

  - The other becomes the candidate

  - Contract the subtree into a new node
  - Recursively solve the new instance; if the new node gets opened, the candidate gets opened

Solving a tree instance

- Natural algorithm
  - chooses highest-capacity nodes in a small vicinity and opens opportunity to the next highest
- Correctness
  - Candidate may be coming from deep inside the subtree
  - Subtree root either gets opened or becomes the candidate
- Optimal

Main result & applications

Lemma We can find an integral distance-2 transfer of a tree instance

Lemma If we can find an integral distance-*r* transfer of a tree instance, we obtain a (3*r*+3)-approximation algorithm for capacitated *k*-center

Theorem ∃ 9-approximation alg for capacitated k-center

Theorem ∃ 11-approximation alg for capacitated k-supplier

Theorem ∃ 9-approximation alg for budgeted-center w/ uniform cap.

Future directions

- Can we do better?
  - Integrality gap lower bound is 7
  - Our algorithm runs in three phases:
    - Preprocessing (finding connected components)
    - Reduction to a tree instance
    - Solving the tree instance
- {0, *L*}-instances
  - Inapproximability and integrality gap lower bound both comes from this special case
  - Better preprocessing gives a 6-approximation algorithm: improved integrality gap!

Future directions

- Is there a better preprocessing for the general case?
- Is there a notion that incorporates these preprocessings?
- Would such a notion be applicable to other network location problems using similar relaxations?

Thank you.