Online Network Design with Outliers

Aris Anagnostopoulos

Dept. of Computer and System Sciences

Sapienza University of Rome

Joint work with:

Fabrizio Grandoni, Stefano Leonardi, and Piotr Sankowski

Background: Online Stochastic Network Design

Solve a network design problem **on the fly** as requests arrive according to a stochastic process.

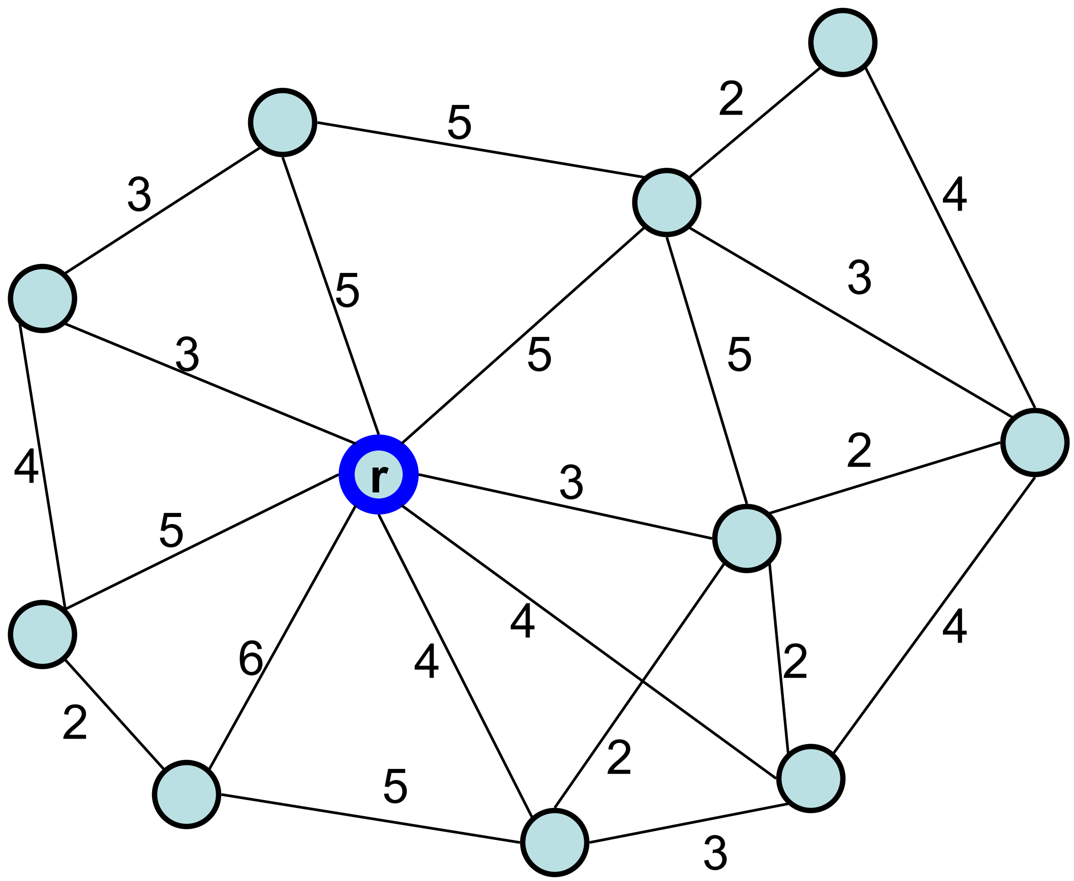
This talk: Focus on **Steiner tree**.

Model:

- Metric space (M, d) and a root $r \in M$
- Input distribution (known or unknown) D on M.
- t requests arrive IID from D (t is known)
- When each request arrives it should be connected to an ongrowing Steiner tree

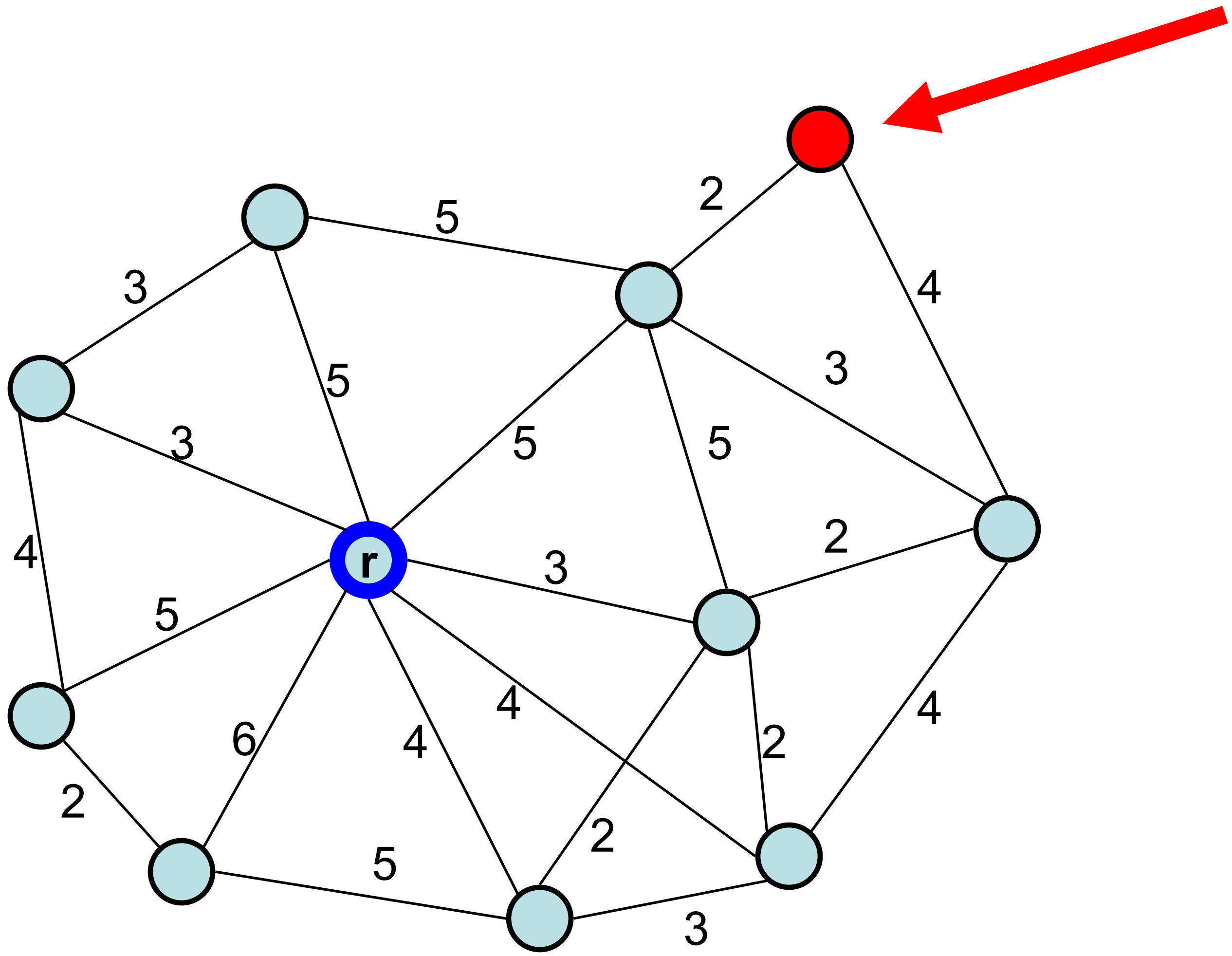
Example

***t* = 4**



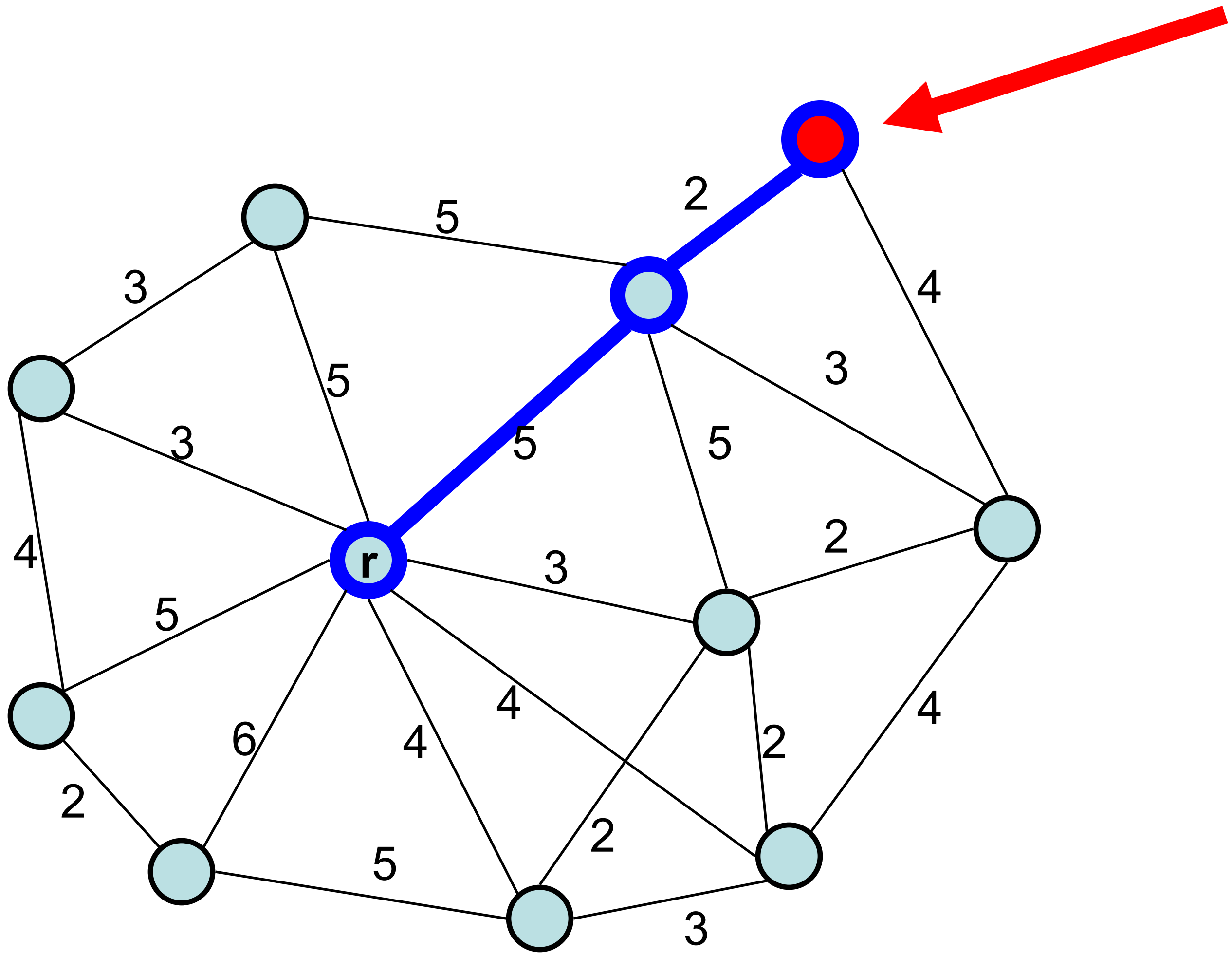
Example

***t* = 4**



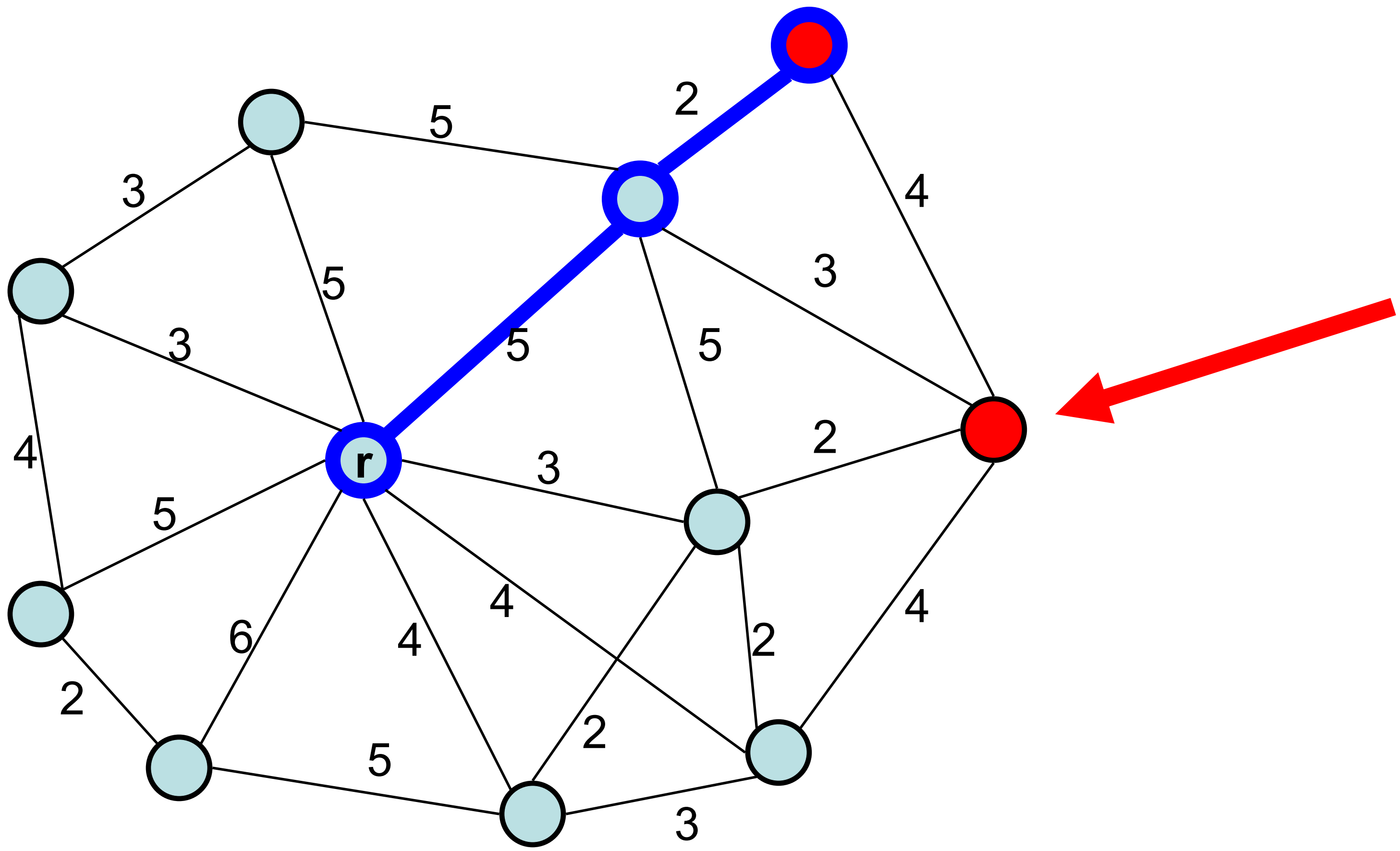
Example

***t* = 4**



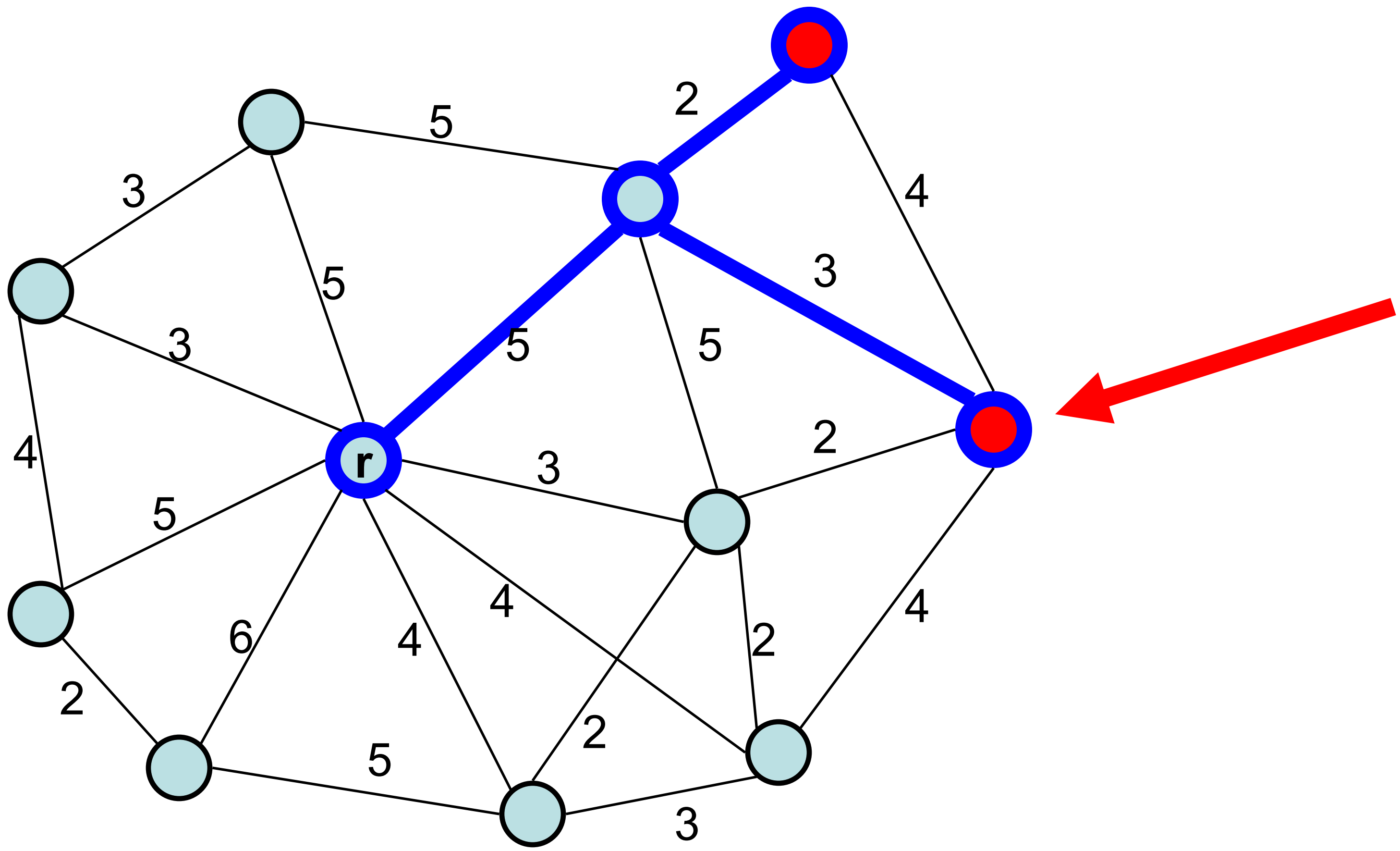
Example

***t* = 4**



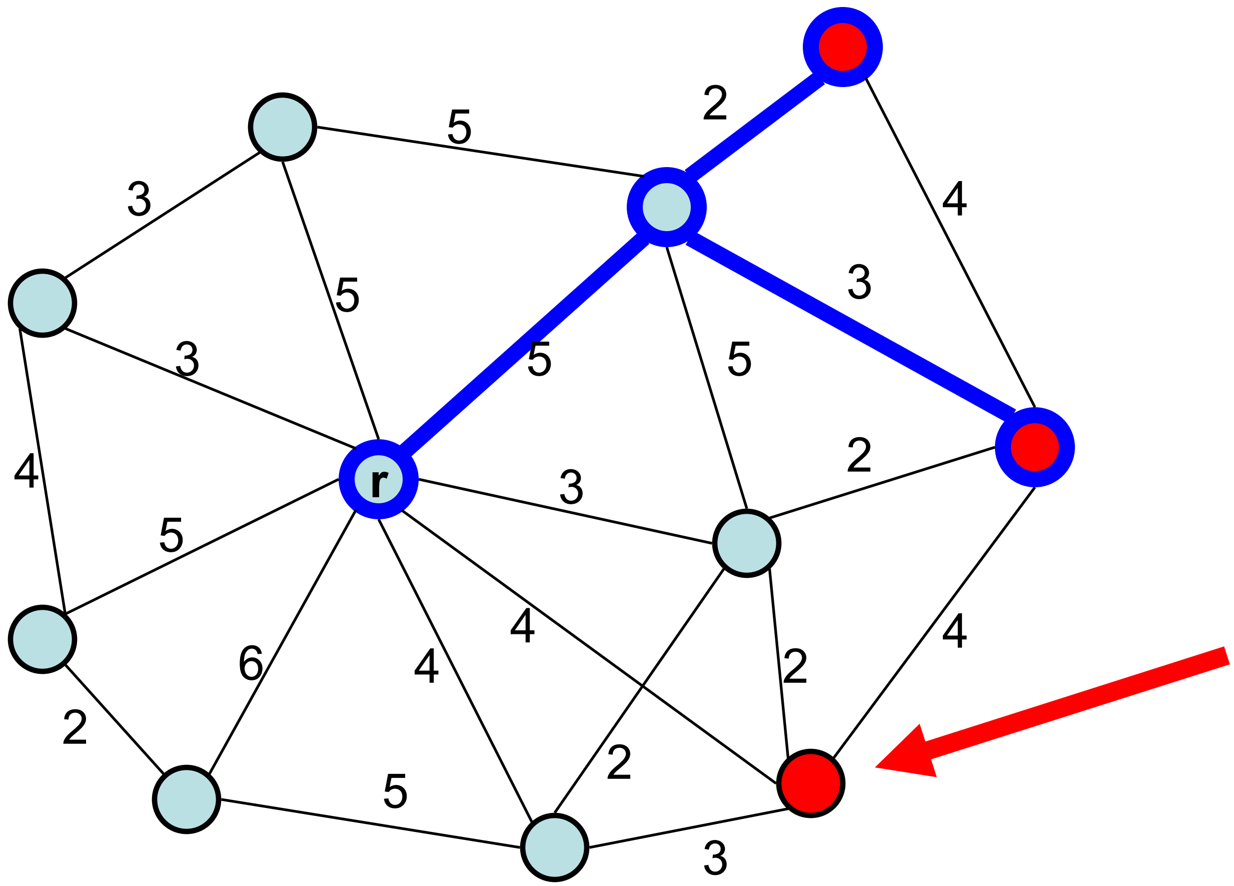
Example

t **= 4**



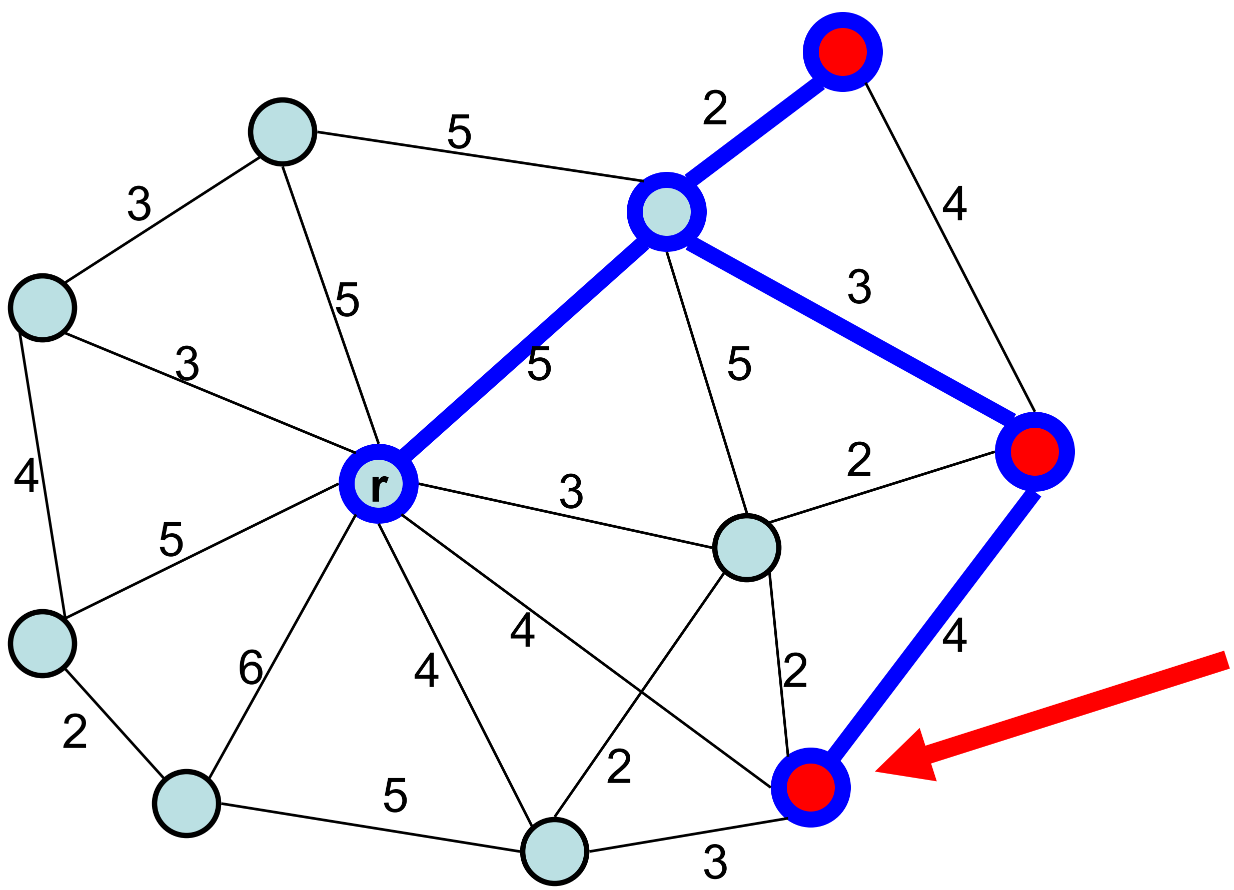
Example

***t* = 4**



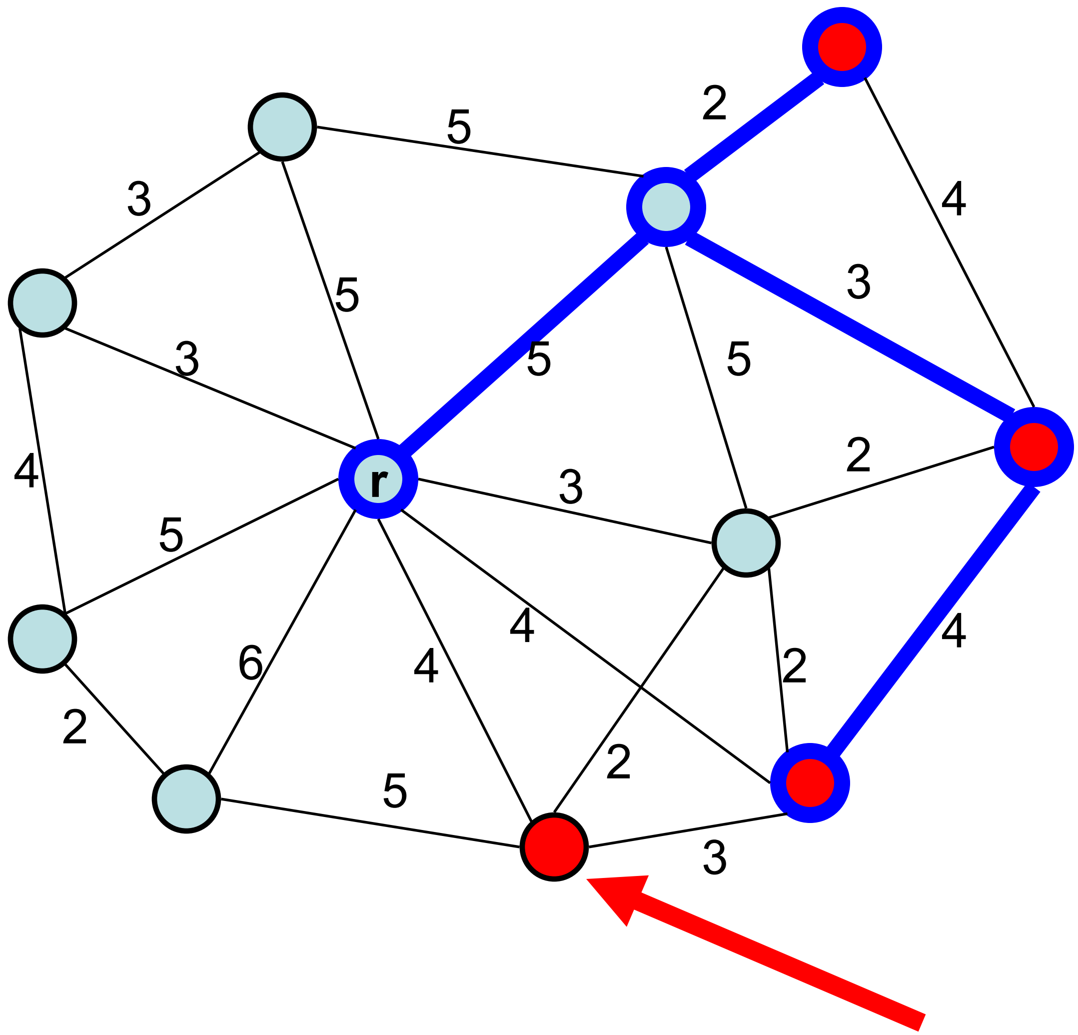
Example

***t* = 4**



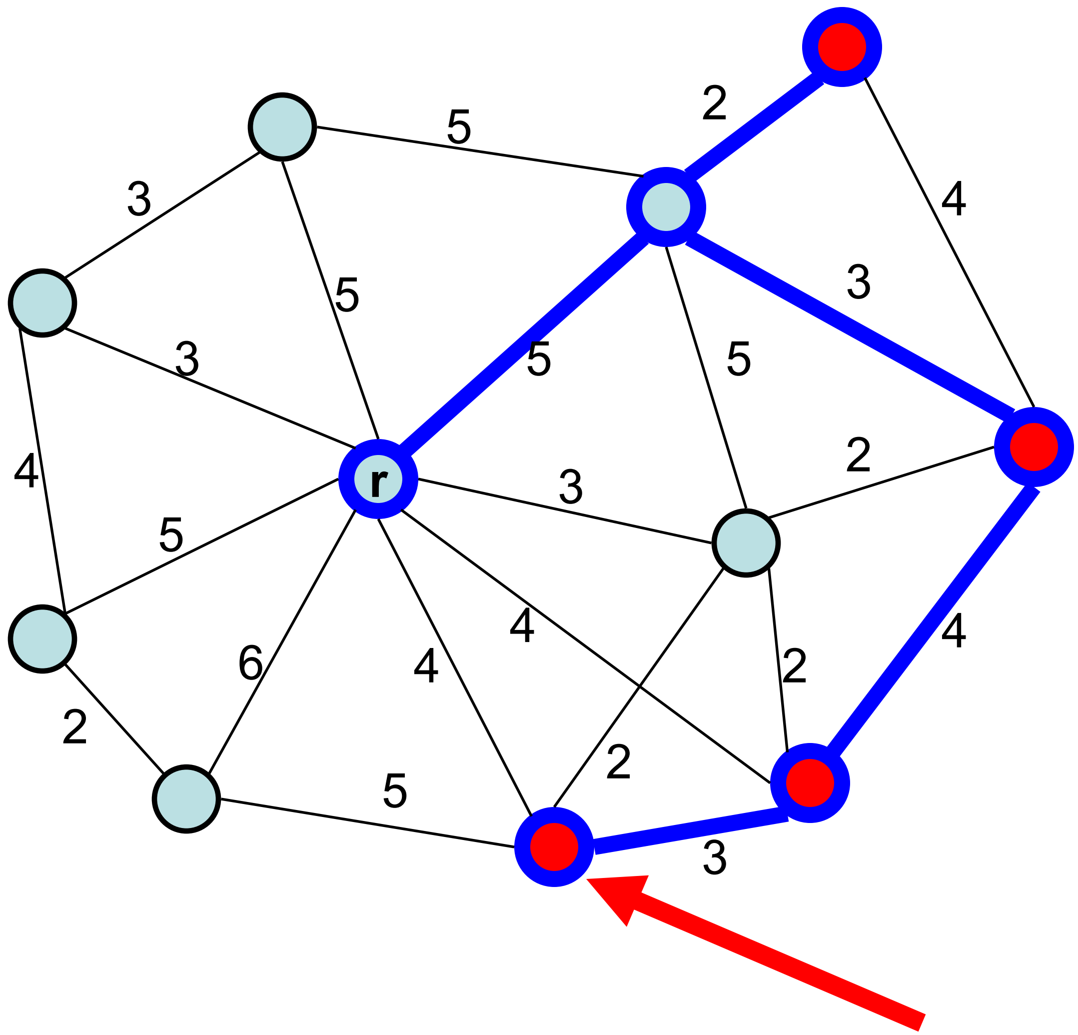
Example

***t* = 4**



Example

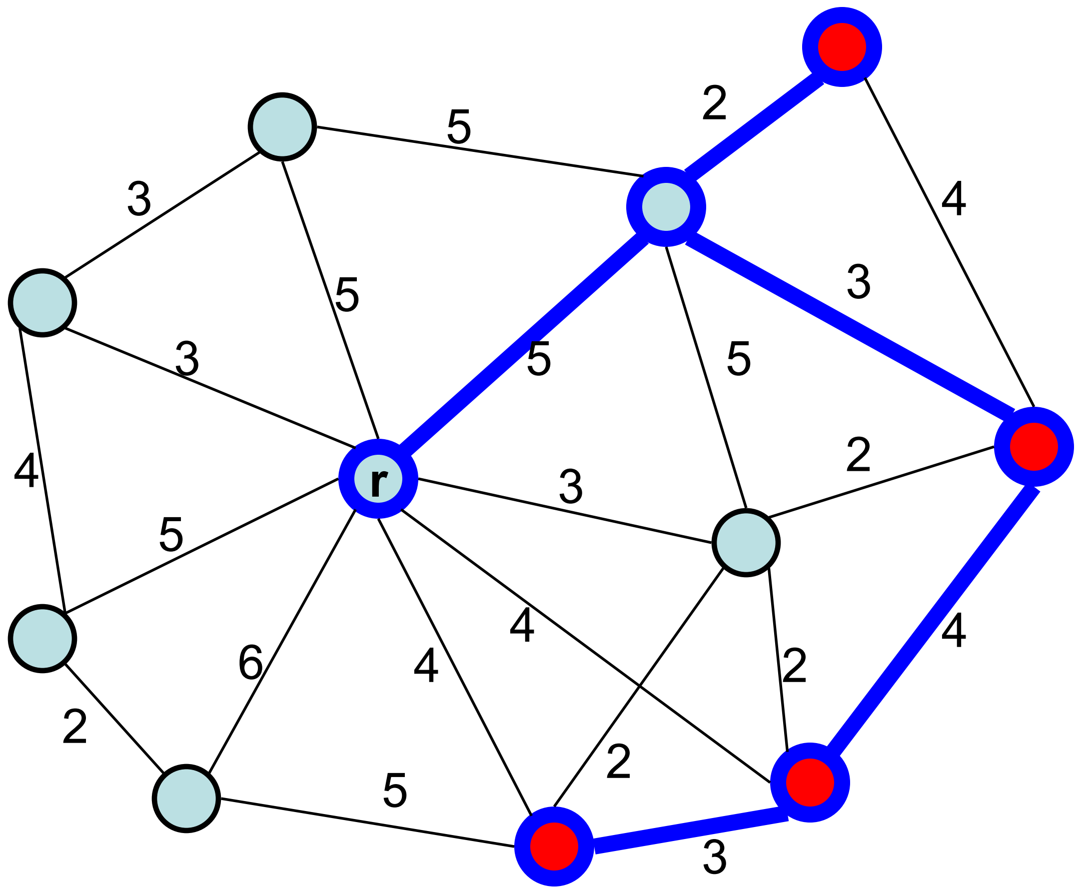
***t* = 4**



Example

ALG Cost = 17

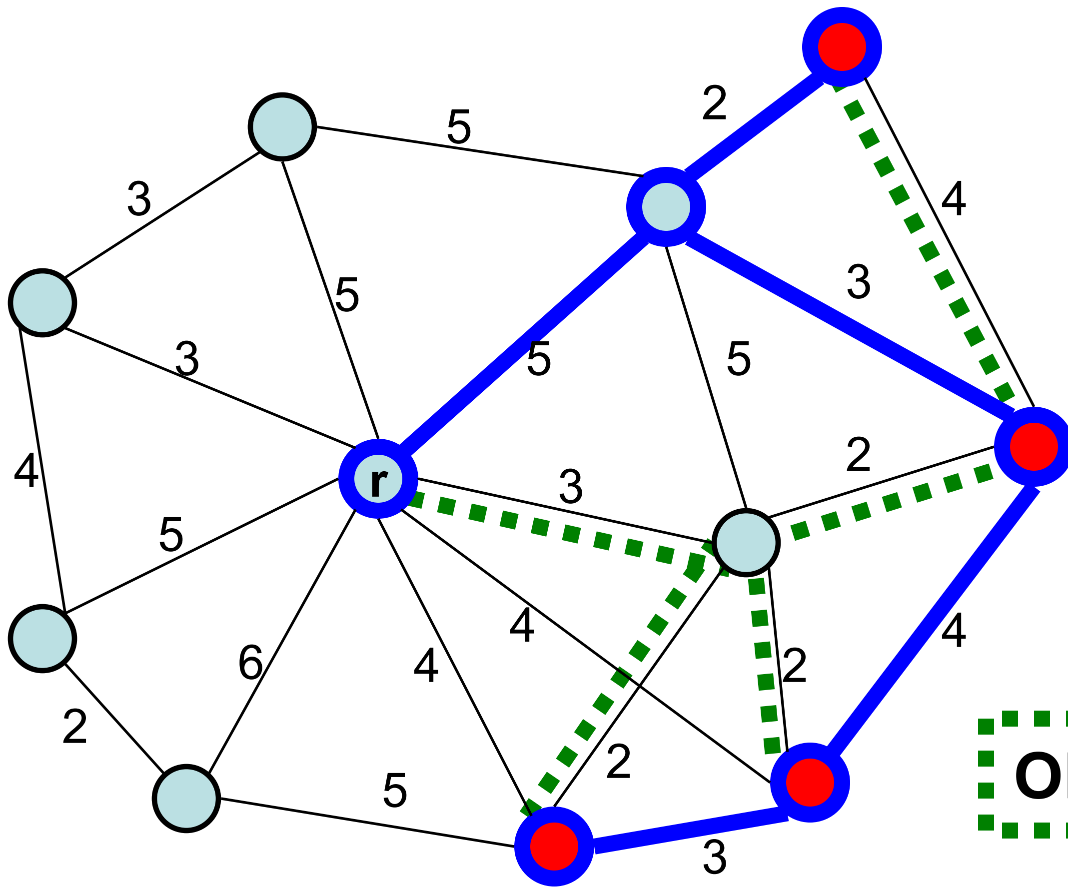
***t* = 4**



Example

ALG Cost = 17

***t* = 4**



OPT Cost = 13

Algorithm Performance

Let:

- $\omega \in D^t$: Sample of t IID requests from D
- ALG(ω, r): Cost of (online) ALG for input ω and random choices r
- OPT(ω): Optimal (offline) cost for input ω

Performance measure (others are possible):

$$\text{Ratio of Expectations} = \sup_{D,t} \frac{\mathbf{E}_{\omega \in D^t, r}[\text{ALG}(\omega, r)]}{\mathbf{E}_{\omega \in D^t}[\text{OPT}(\omega)]}$$

Algorithm Performance

Theorem [Garg, Gupta, Leonardi, Sankowski, SODA 08]

There is an algorithm with

Ratio of Expectations = O(1).

This Work: Online Stochastic Network Design with Outliers

Model:

- Metric space (M, d) and a root $r \in M$
- Input distribution (known or unknown) D on M.
- t requests arrive from D (t is known)
- $k \leq t$ requests must be satisfied (k is known)
- When each request arrives, the algorithm
  - Decides whether to satisfy the request
  - If yes, then it connects it to an ongrowing Steiner tree
- At the end k requests must have been satisfied

Algorithm Performance

Let:

- $\omega \in D^t$: Sample of t IID requests from D
- ALG(ω, r): Cost of (online) ALG for input ω and random choices r
- OPT(ω): Optimal (offline) cost for input ω

Performance measure (others are possible):

$$\text{Ratio of Expectations} = \sup_{D,t} \frac{\mathbf{E}_{\omega \in D^t, r}[\text{ALG}(\omega, r)]}{\mathbf{E}_{\omega \in D^t}[\text{OPT}(\omega)]}$$

Algorithm Performance

Let:

- $\omega \in D^t$: Sample of t IID requests from D
- ALG(ω, r): Cost of (online) ALG for input ω and random choices r
- OPT(ω): Optimal (offlin

ALG and OPT might select different points

Performance measure (others are possible):

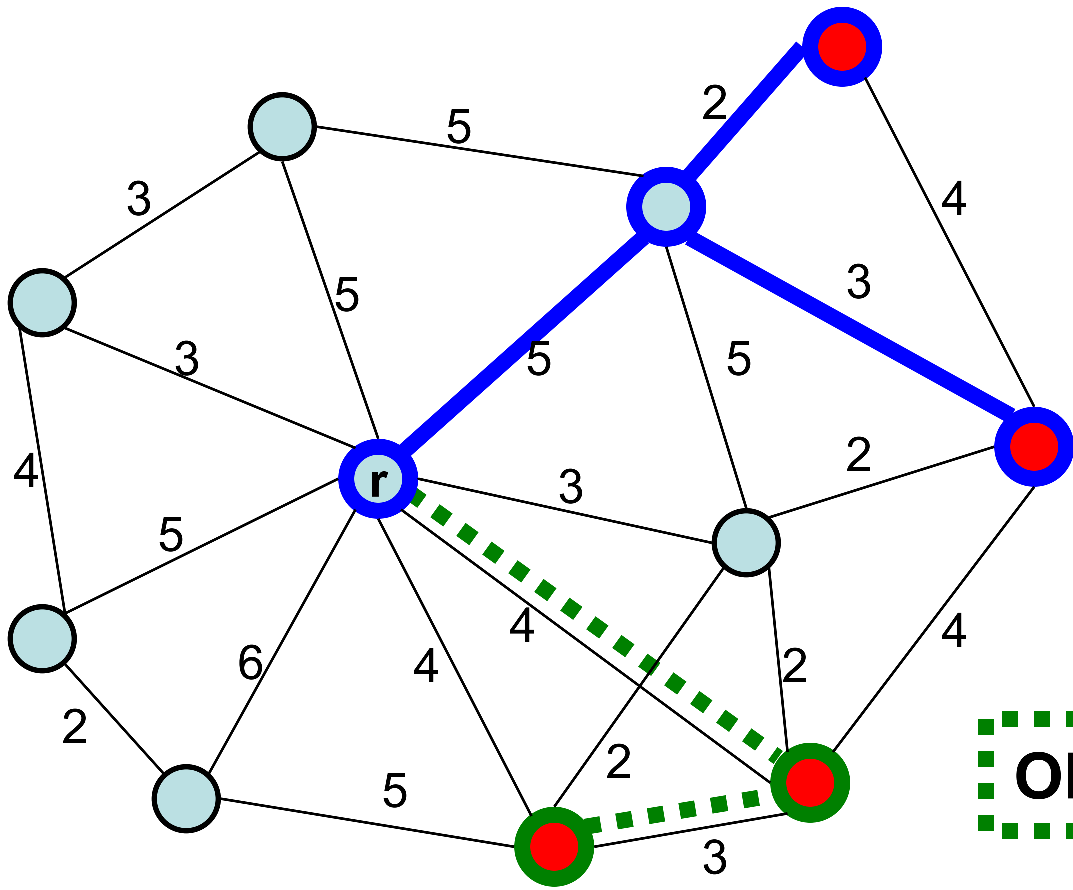
$$\text{Ratio of Expectations} = \sup_{D,t} \frac{\mathbf{E}_{\omega \in D^t, r}[\text{ALG}(\omega, r)]}{\mathbf{E}_{\omega \in D^t}[\text{OPT}(\omega)]}$$

Example

ALG Cost = 10

***t* = 4**

***k* = 2**



OPT Cost = 7

General Network Design

Online Stochastic Network Design with Outliers

- We need to solve an network design problem
- ***t*** requests arrive online
- We need to satisfy ***k*** out of ***t*** requests
- Online Facility location, Online TSP, …

Motivation

Might not want to satisfy everybody if it is too expensive but say 90% of the population

Results

Inapproximability:

- **$k = t - 1$:** Arbitrarily bad (contrast with $k = t$).

Results

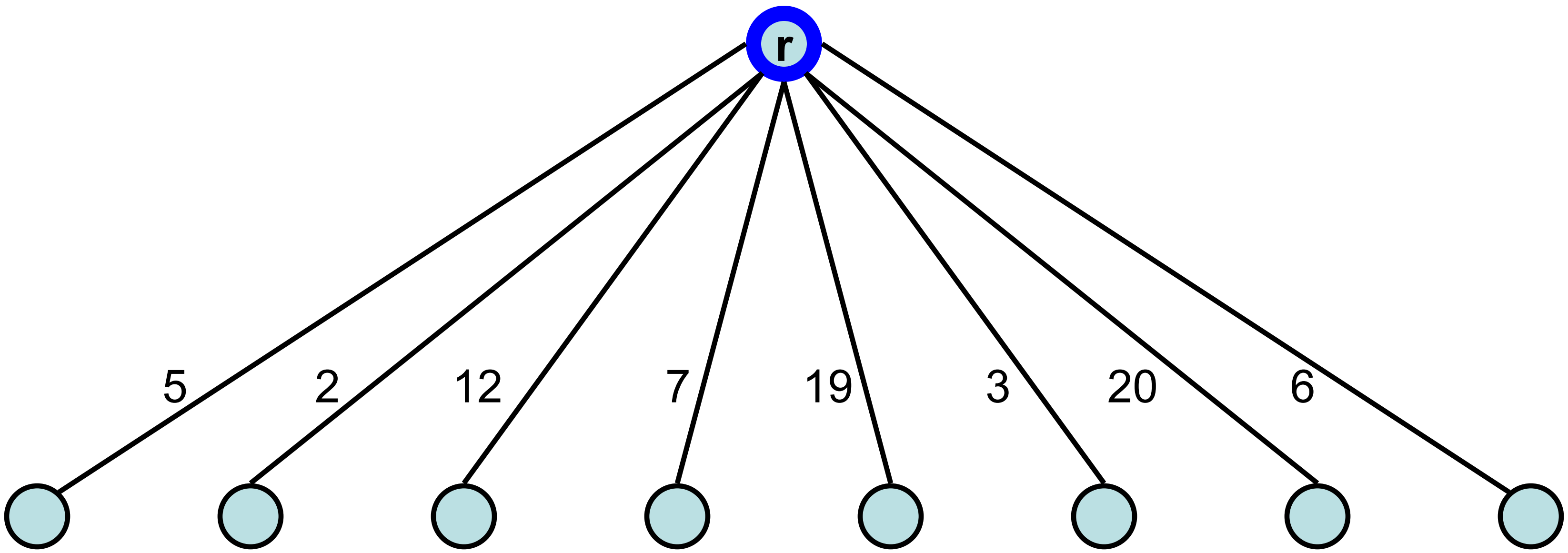
Inapproximability:

- ***k* = *t* – 1:** Arbitrarily bad (contrast with $k = t$).
- ***k* = 1:** Exponentially worse (contrast with secretary)

Results

Inapproximability:

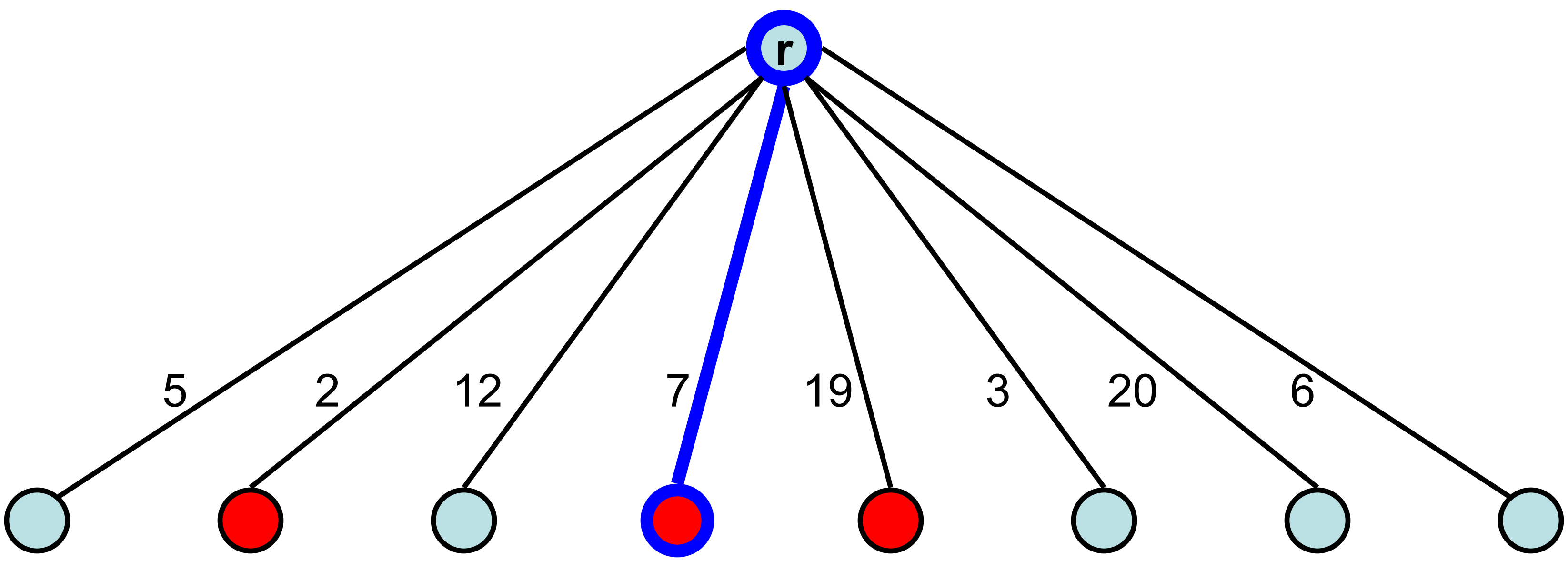
- ***k* = *t* – 1:** Arbitrarily bad (contrast with *k* = *t*).
- ***k* = 1:** Exponentially worse (contrast with secretary)



Results

Inapproximability:

- ***k* = *t* – 1:** Arbitrarily bad (contrast with *k* = *t*).
- ***k* = 1:** Exponentially worse (contrast with secretary)



Results

So we look into bicriteria:

- Allow online ALG to select $(1-\varepsilon)k$ request
- Compare with OPT that selects k requests

Results

So we look into bicriteria:

- Allow online ALG to select $(1-\varepsilon)k$ request
- Compare with OPT that selects k requests

Lower bound: $\mathrm{RoE} = \Omega\left(\frac{\ln n}{\ln \ln n}\right)$

Upper bound: $\mathrm{RoE} = O(\ln^2 n)$

if $k = \Theta(t)$: $\mathrm{RoE} = O(\ln n \ln \ln n)$

Upper Bound

Theorem. There exists an algorithm such that for every $0 \leq \varepsilon \leq 1$ selects $(1 - \varepsilon)k$ request whp. with

$$\text{Ratio of Expectations} = O(\ln^2 n).$$

Algorithm

Algorithm ALG

Assumes Uniform distribution (can generalize)

1. Embed metric space into a Bartal tree
2. Group the nodes into groups of size $s = \frac{n}{t} \ln n$
3. Sample *t* (**light blue**) nodes
4. Create approximate *k*-Steiner tree of *k* out of *t* points (**blue** points)
5. **Blue** group = group with at least one **blue** point
6. Receive *t* input points (**orange**)
7. Accept a point if it is in a blue group – color it **red**
8. Connect it to current Steiner tree

Algorithm

Algorithm ALG

Assumes Uniform distribution (can generalize)

1. Embed metric space into a Bartal tree
2. Group the nodes into groups of size $s = \frac{n}{t} \ln n$
3. Sample *t* (**light blue**) nodes
4. Create approximate *k*-Steiner tree of *k* out of *t* points (**blue** points)
5. **Blue** group = group with at least one **blue** point

Sample and build anticipatory solution

6. Receive *t* input points (**orange**)
7. Accept a point if it is in a blue group – color it **red**
8. Connect it to current Steiner tree

Algorithm

Algorithm ALG

Assumes Uniform distribution (can generalize)

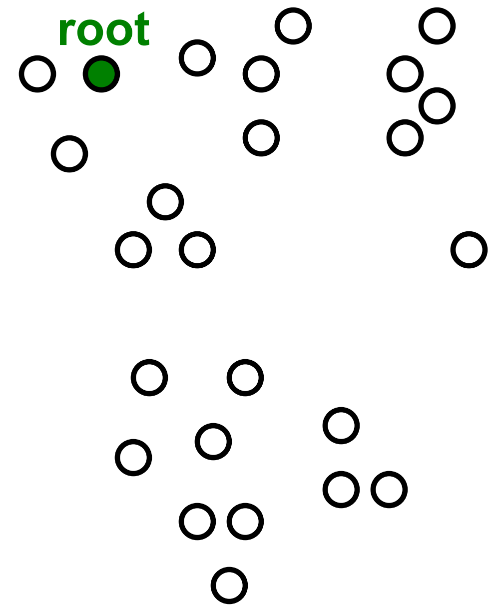
1. Embed metric space into a Bartal tree
2. Group the nodes into groups of size $s = \frac{n}{t}\ln n$
3. Sample t (**light blue**) nodes
4. Create approximate k-Steiner tree of k out of t points (**blue** points)
5. **Blue** group = group with at least one **blue** point
6. Receive t input points (**orange**)
7. Accept point if it is in a blue group – color it **red**
8. Connect it to current Steiner tree

Sample and build anticipatory solution

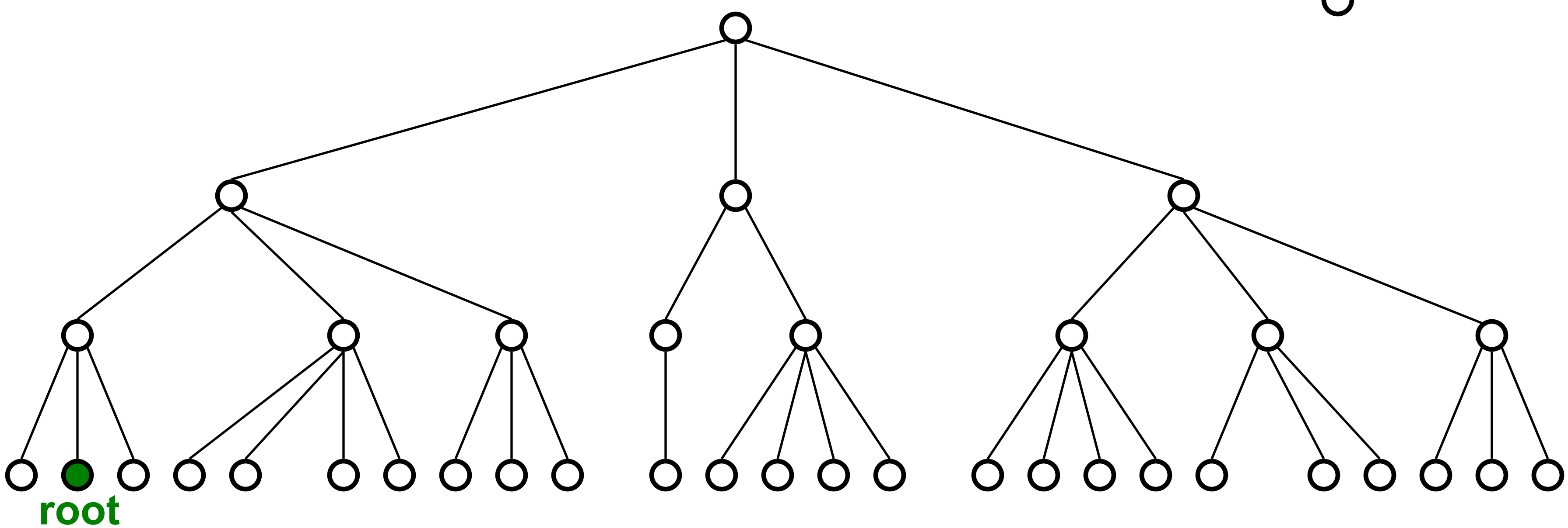
Receive actual requests

Algorithm

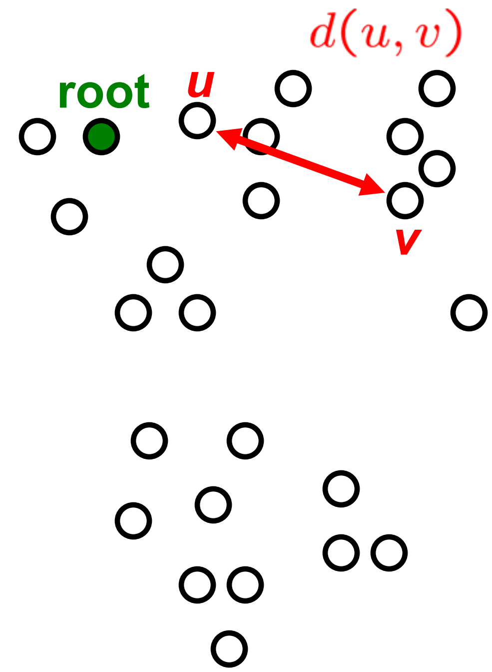
$t = 7, k = 4$



1. **Create the Bartal tree**
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



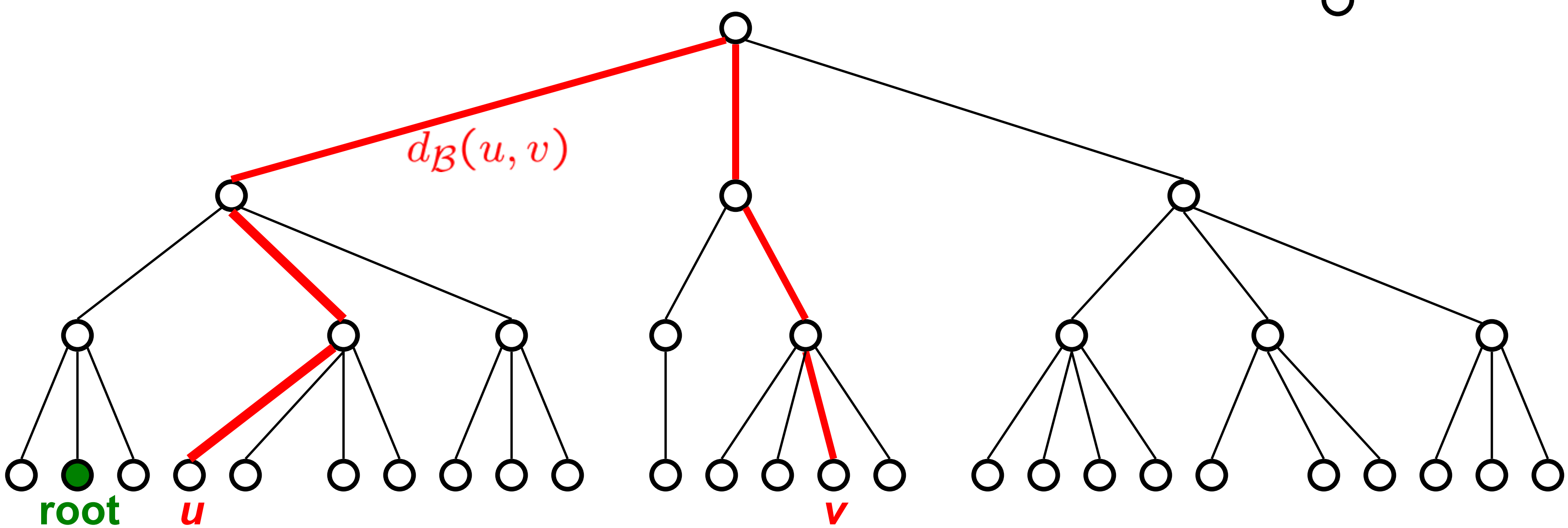
Embedding



Embed the metric space to a **Bartal tree** (2-HST)

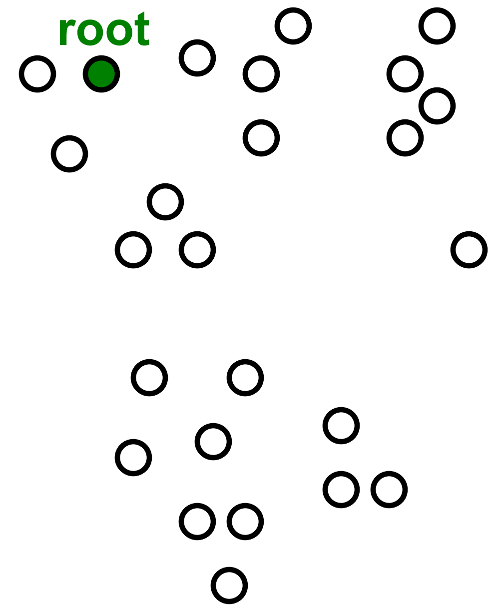
Properties:

1. Points in the space = leaves in Bartal tree
2. $d(u,v) \leq d_{\mathcal{B}}(u,v)$
3. $\mathbf{E}[d_{\mathcal{B}}(u,v)] \leq O(\ln n) \cdot d(u,v)$

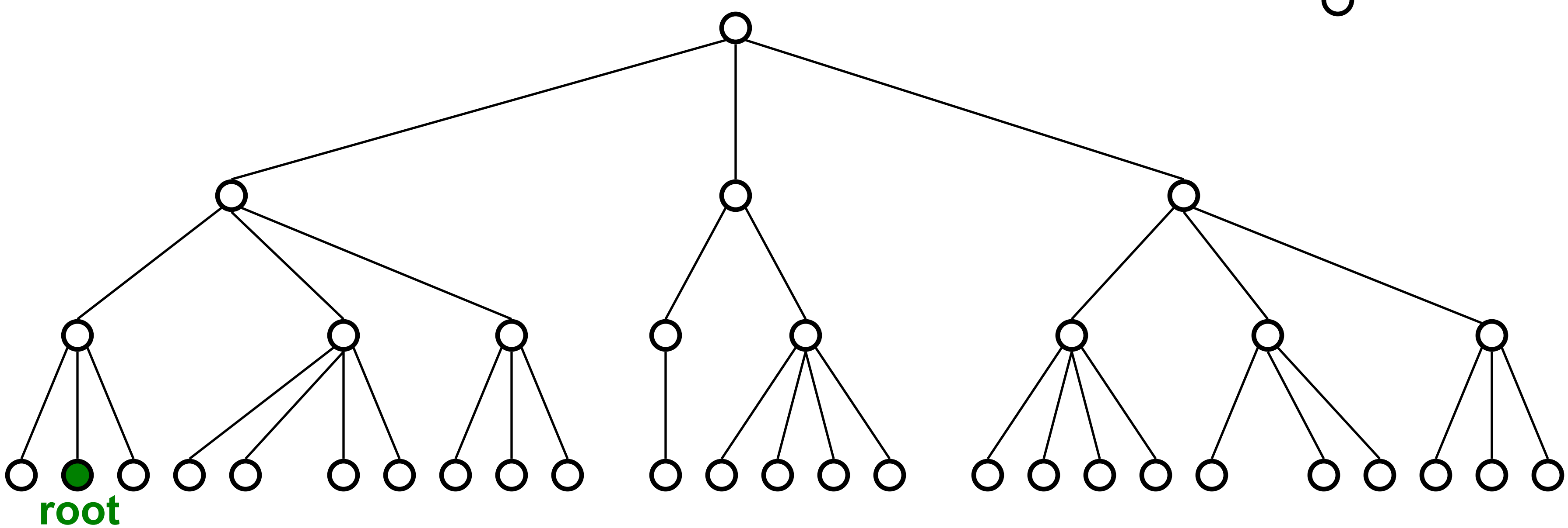


Algorithm

t = 7, k = 4

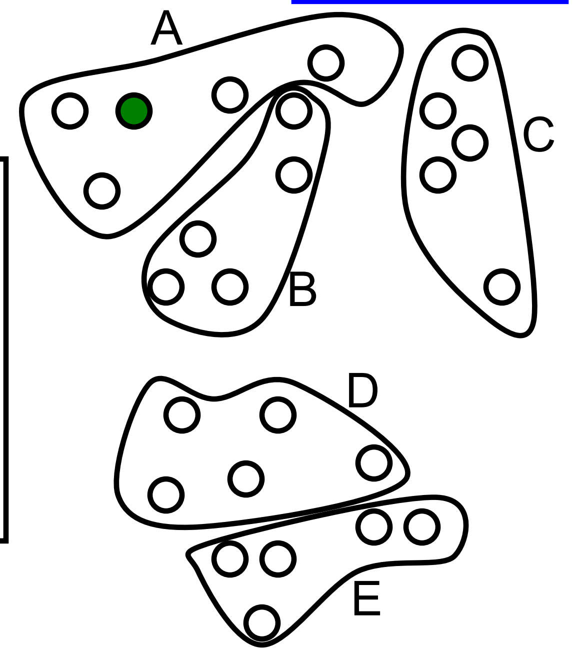


1. **Create the Bartal tree**
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

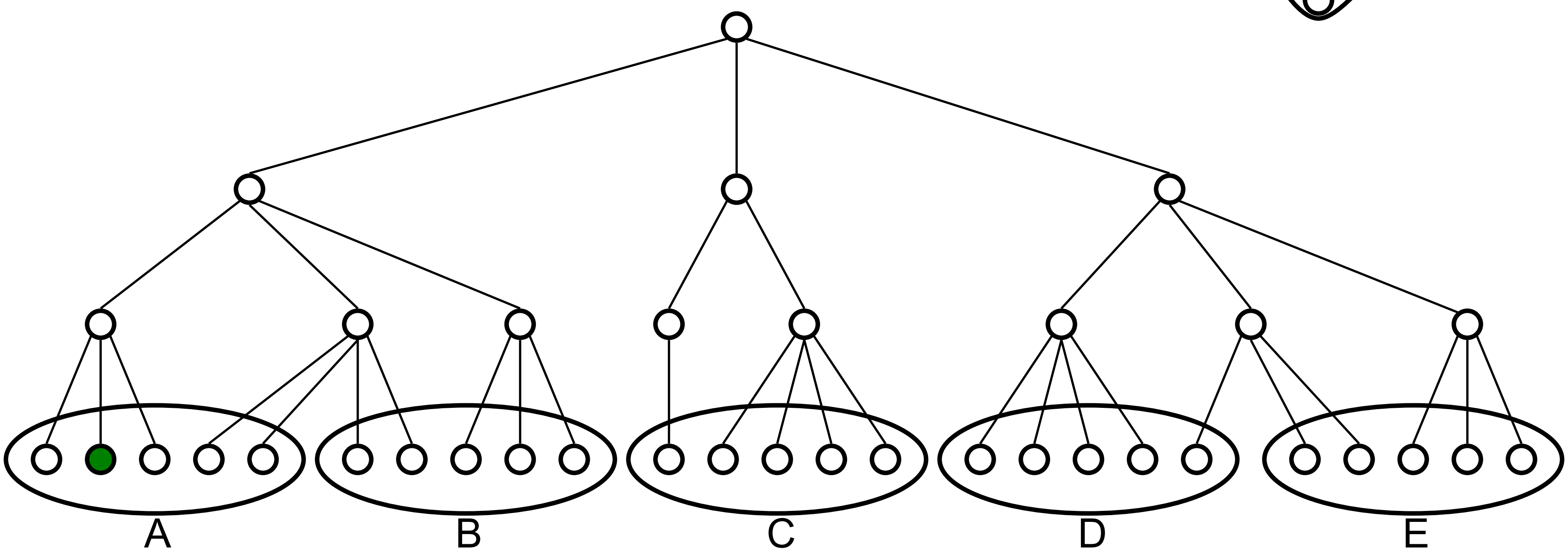


Algorithm

$t = 7, k = 4$

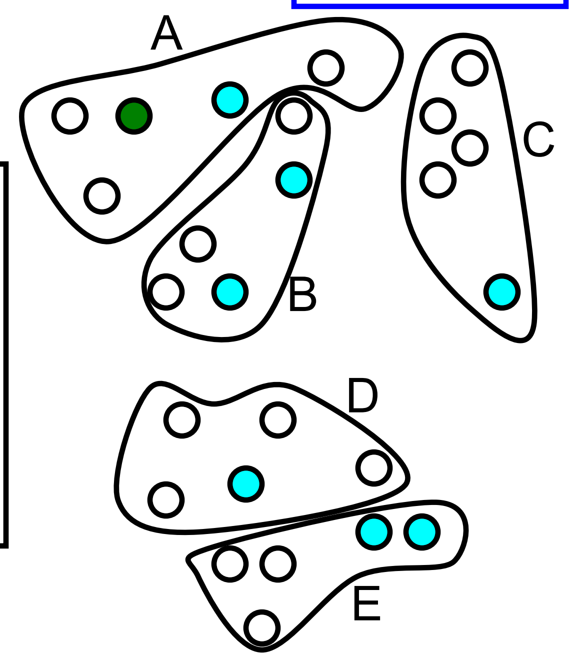


1. Create the Bartal tree
2. **Group the leaves into groups of size** $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

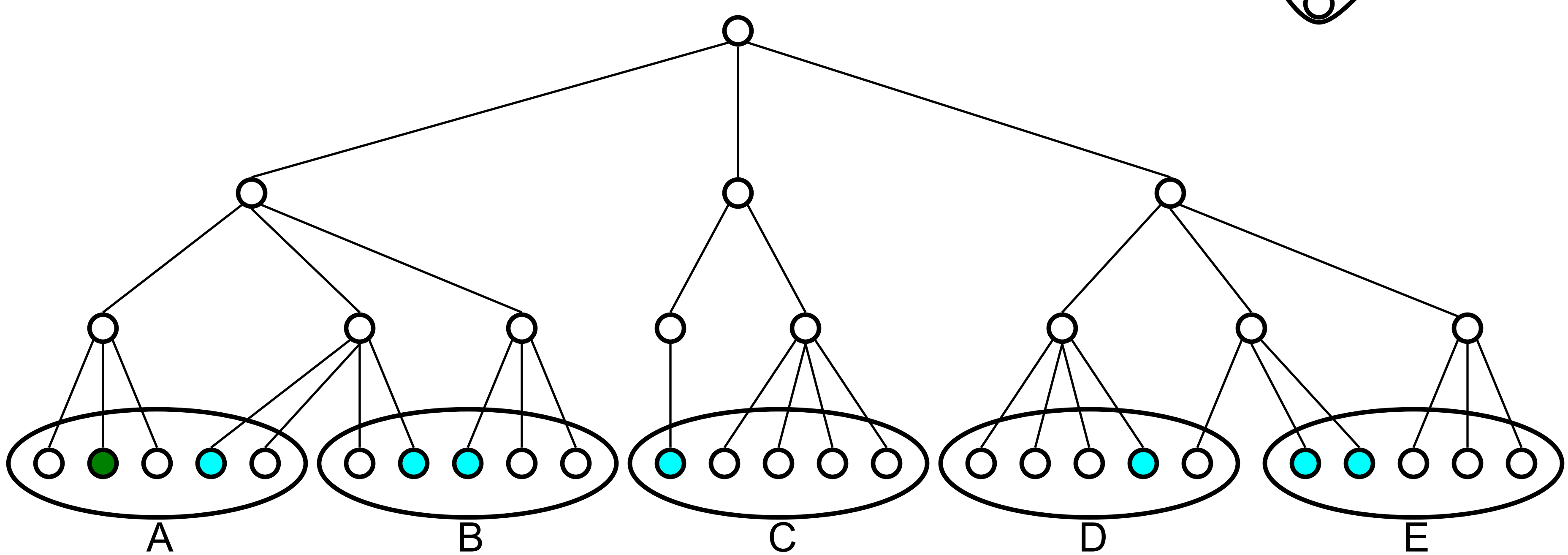


Algorithm

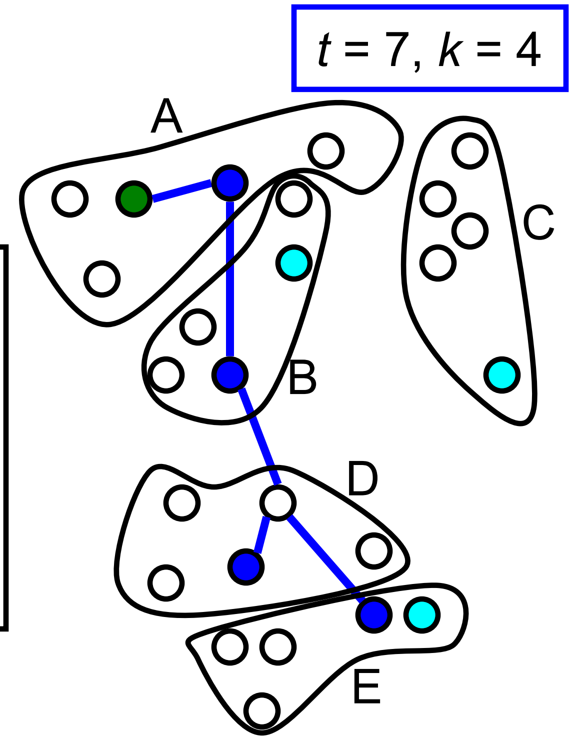
t = 7, k = 4



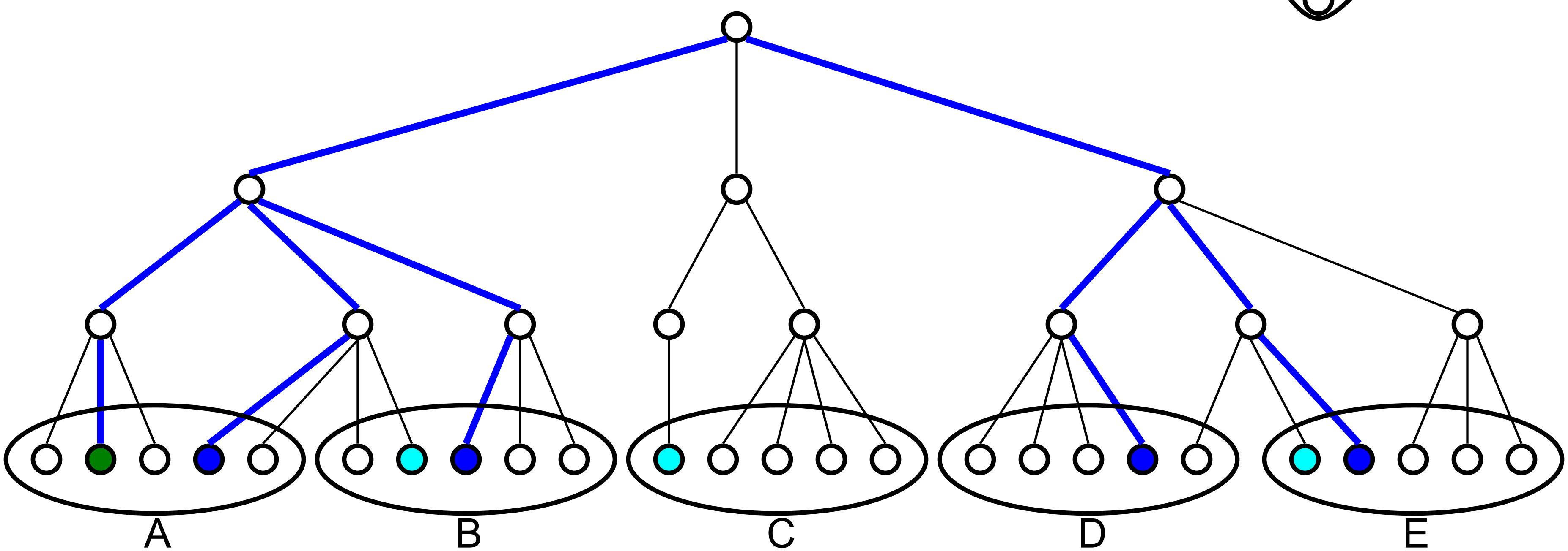
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. **Sample *t* (light blue) nodes**
4. Create approximate *k*-Steiner tree of *k* out of *t* points (blue points)
5. Blue group = group with at least one blue point
6. Receive *t* input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



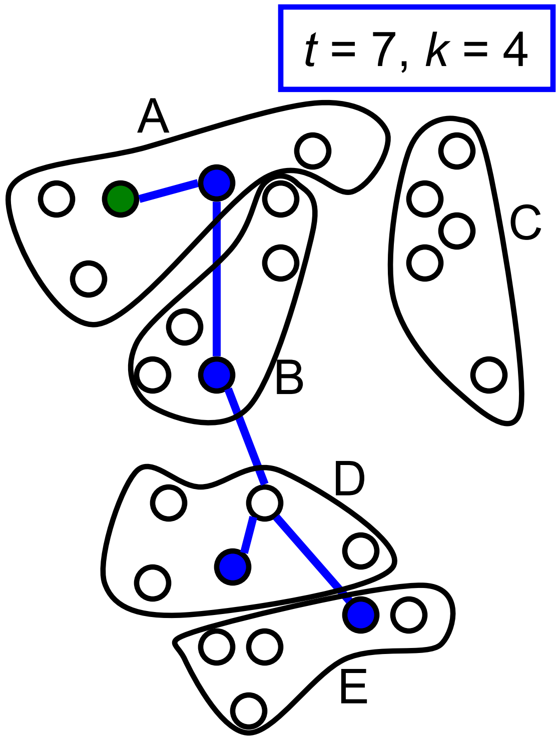
Algorithm



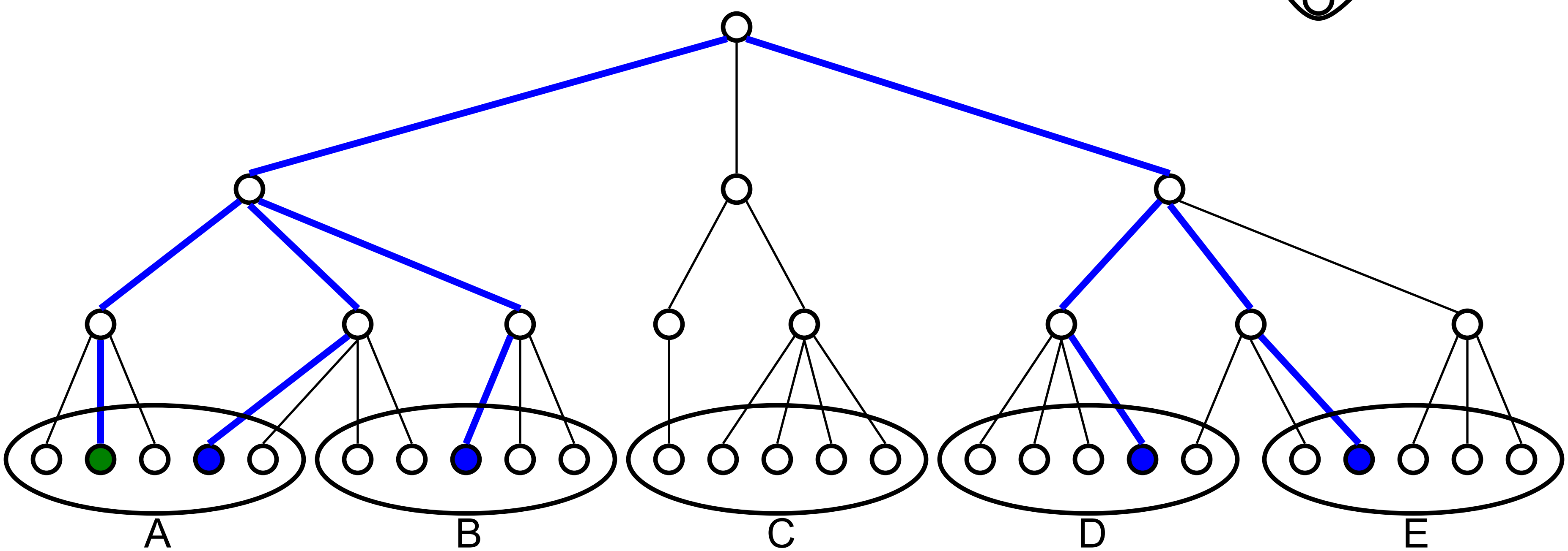
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. **Create approximate *k*-Steiner tree of *k* out of *t* points (blue points)**
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



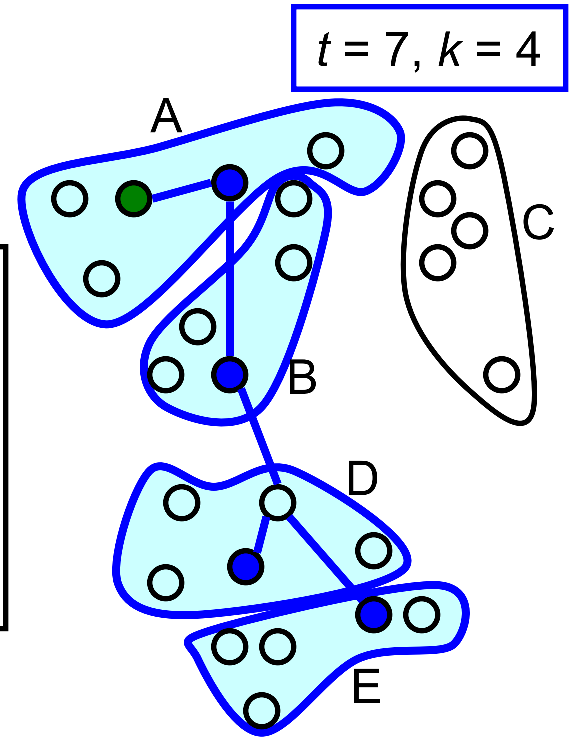
Algorithm



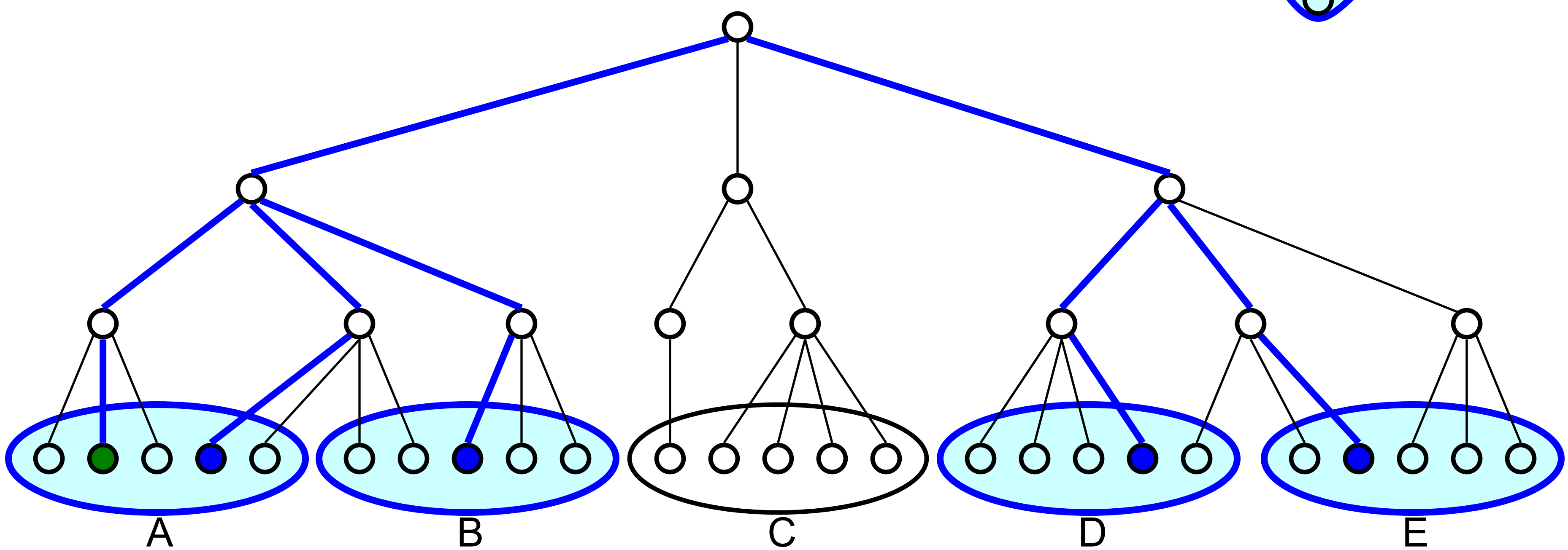
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. **Create approximate k-Steiner tree of k out of t points (blue points)**
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



Algorithm

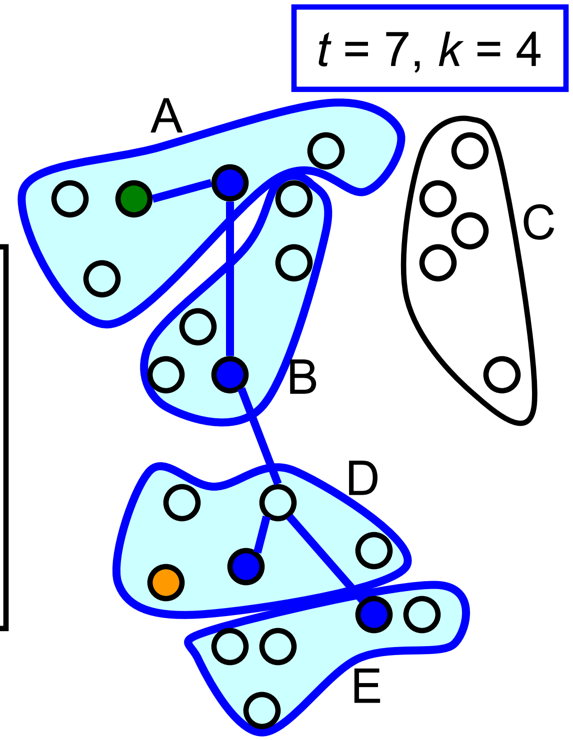


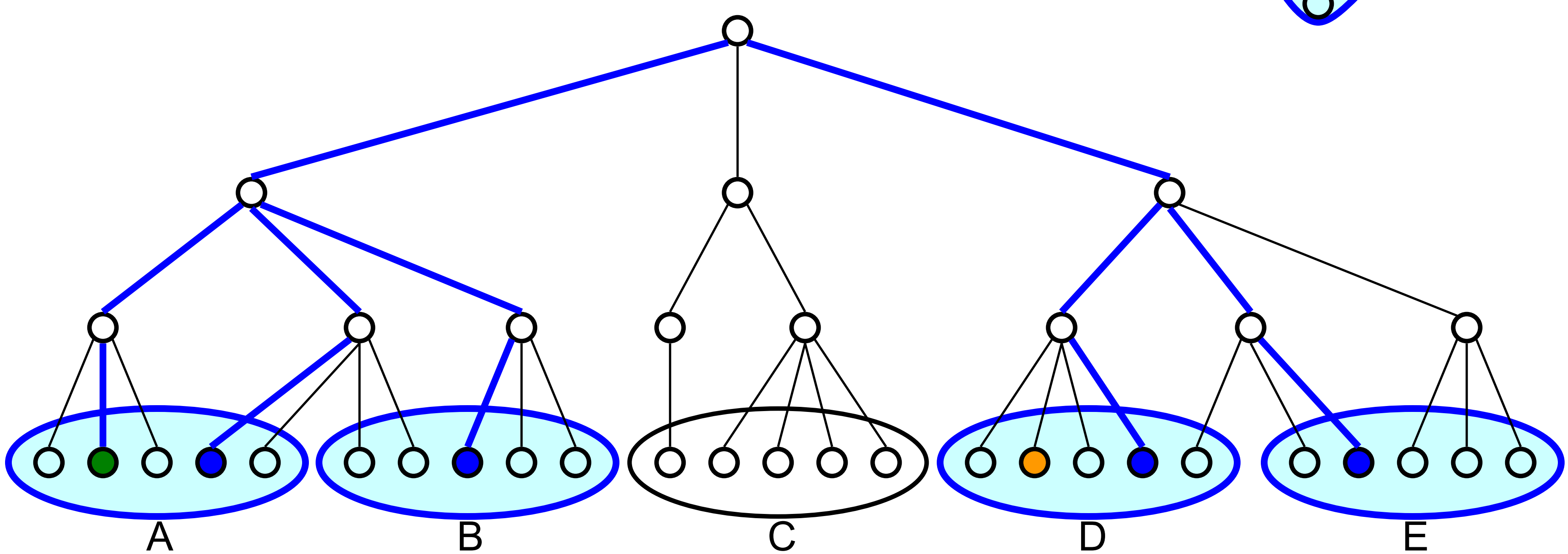
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. **Blue group = group with at least one blue point**
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



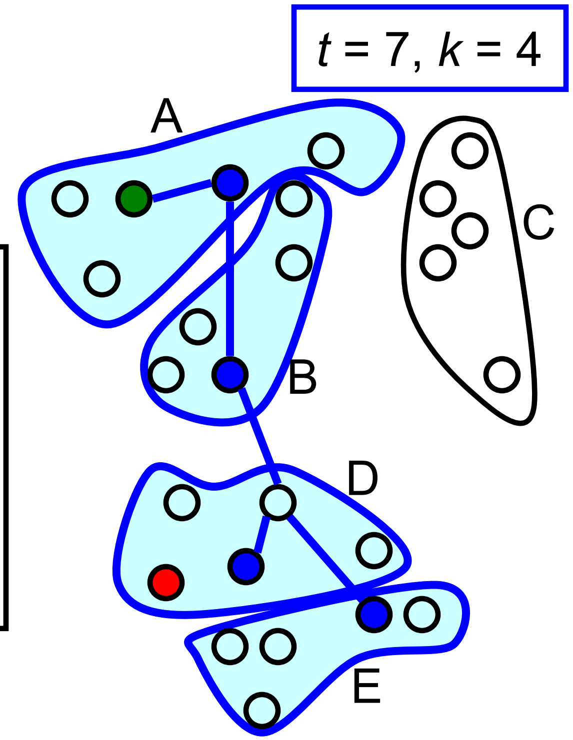
Algorithm

1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. **Receive t input points (orange)**
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

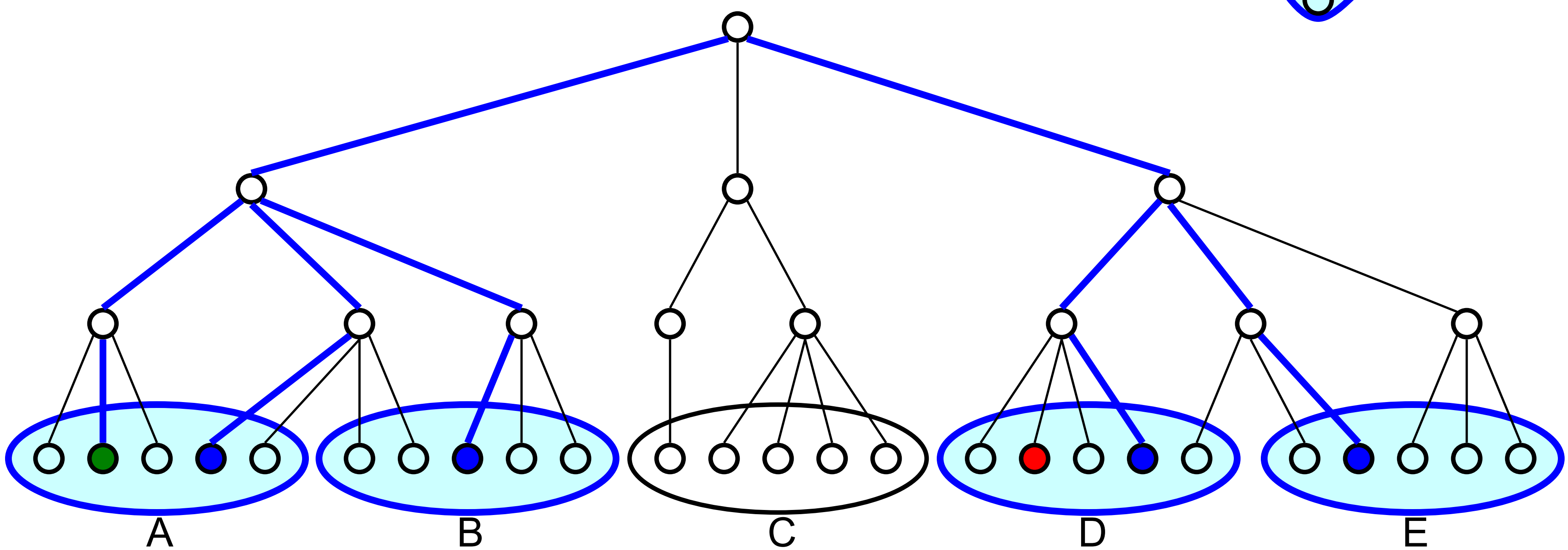




Algorithm

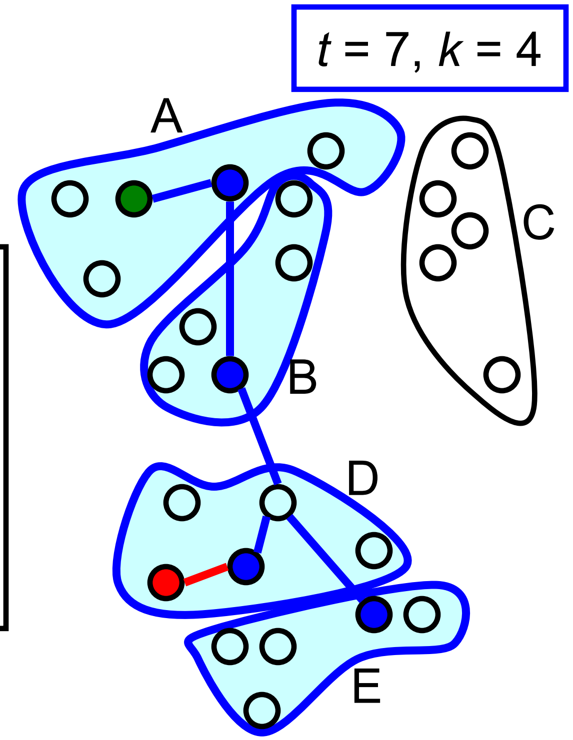


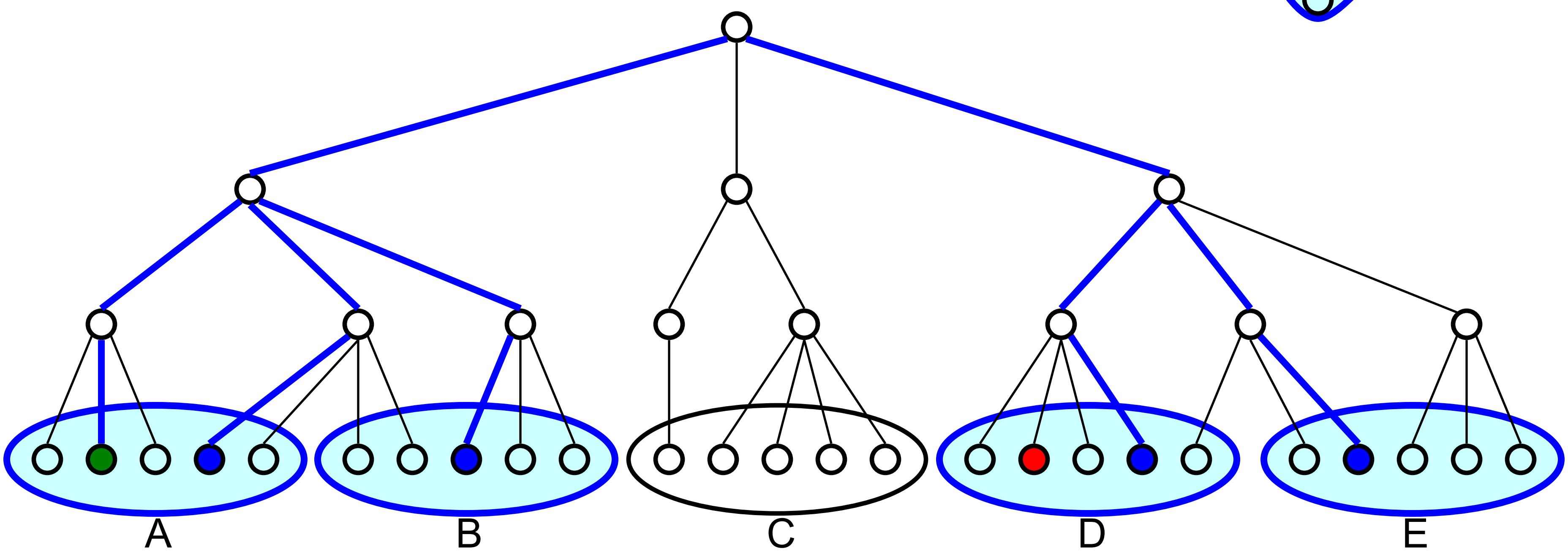
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. **Accept a point if it is in a blue group – color it red**
8. Connect it to current Steiner tree



Algorithm

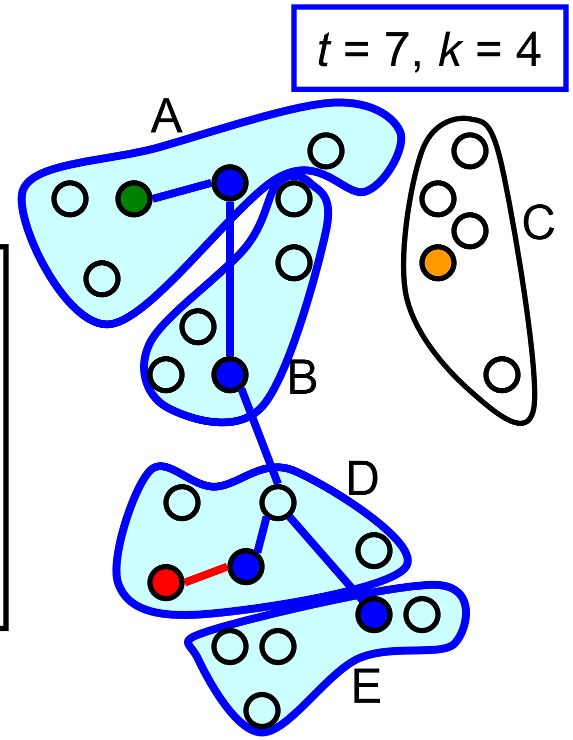
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. **Connect it to current Steiner tree**

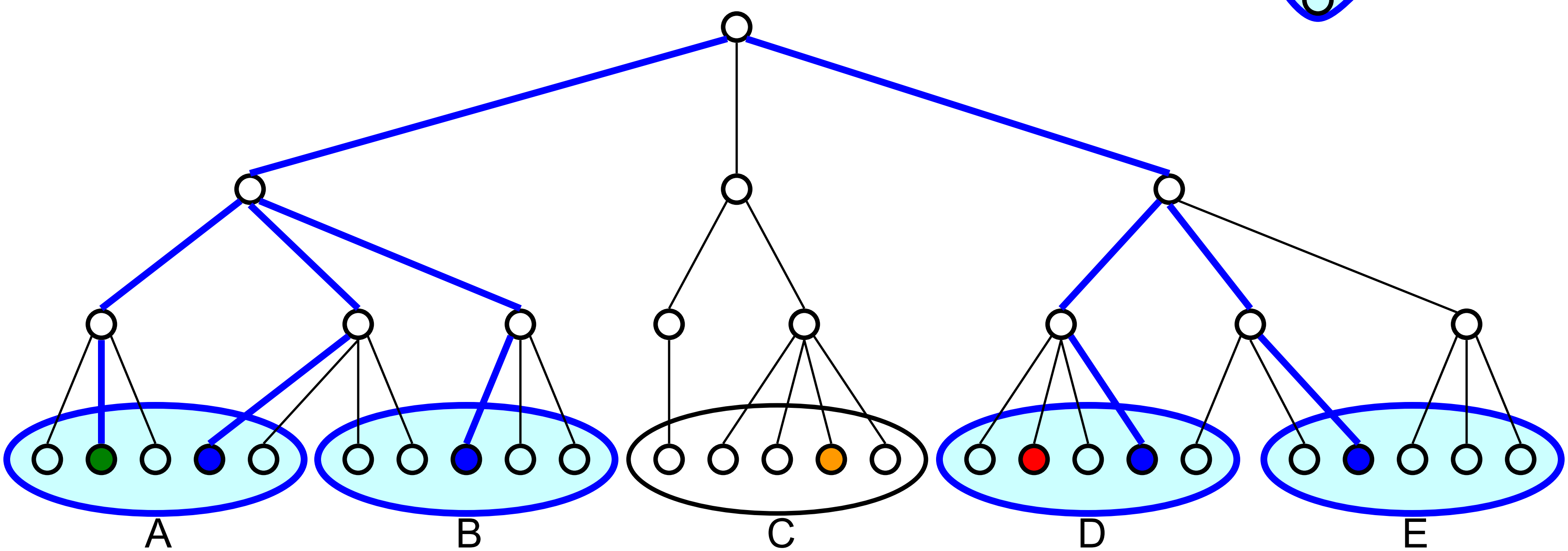




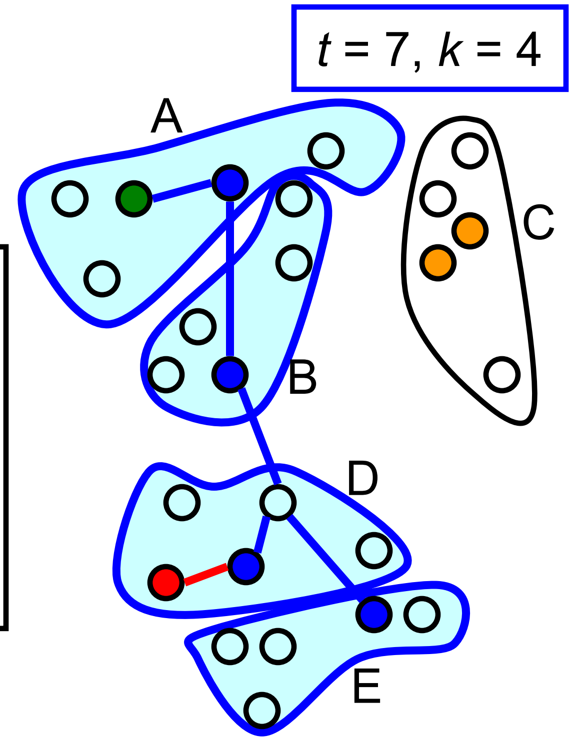
Algorithm

1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. **Receive t input points (orange)**
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

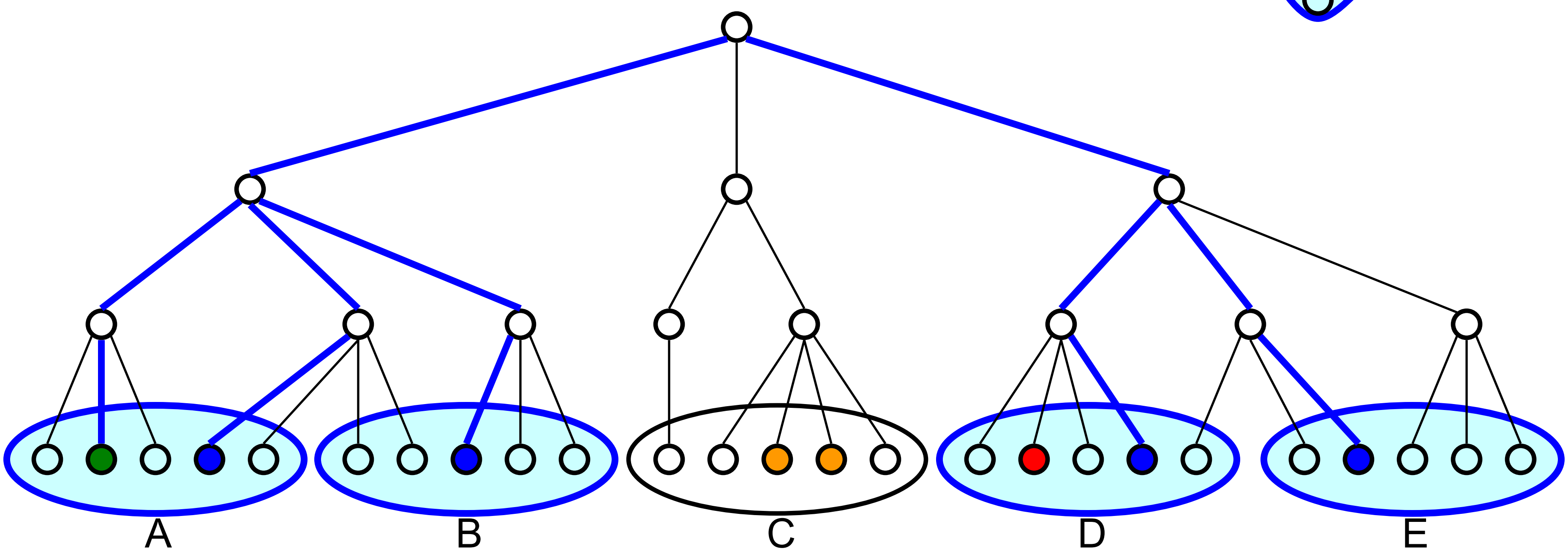




Algorithm

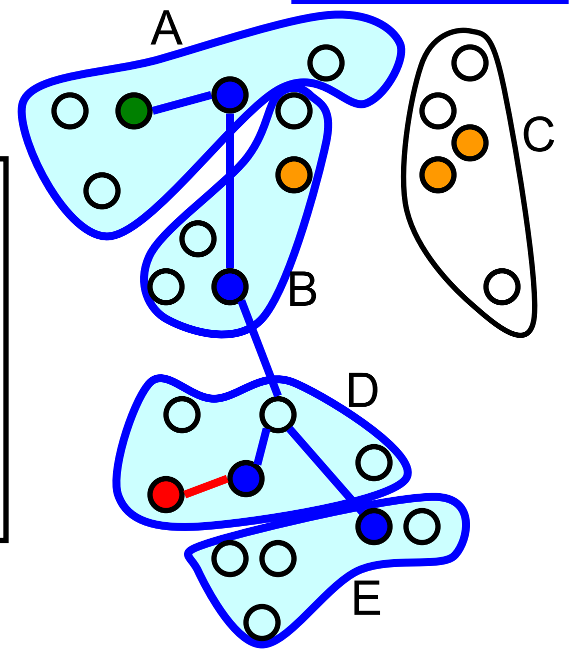


1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample *t* (light blue) nodes
4. Create approximate *k*-Steiner tree of *k* out of *t* points (blue points)
5. Blue group = group with at least one blue point
6. **Receive *t* input points (orange)**
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

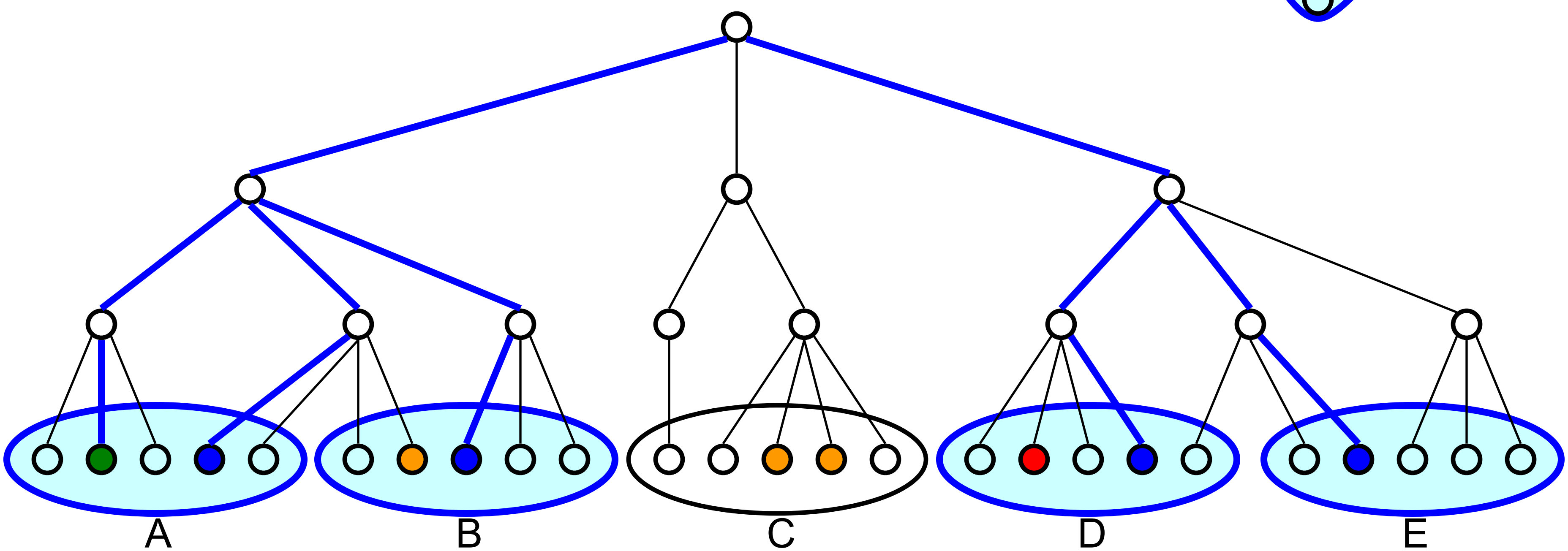


Algorithm

t = 7, k = 4

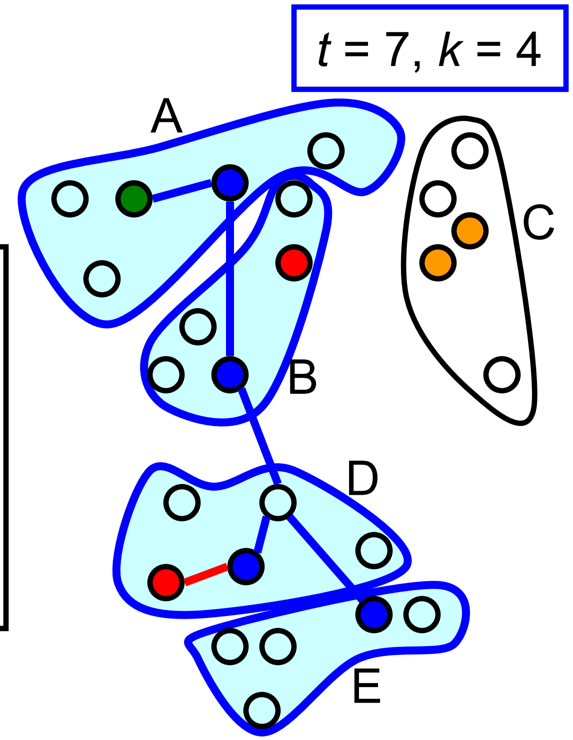


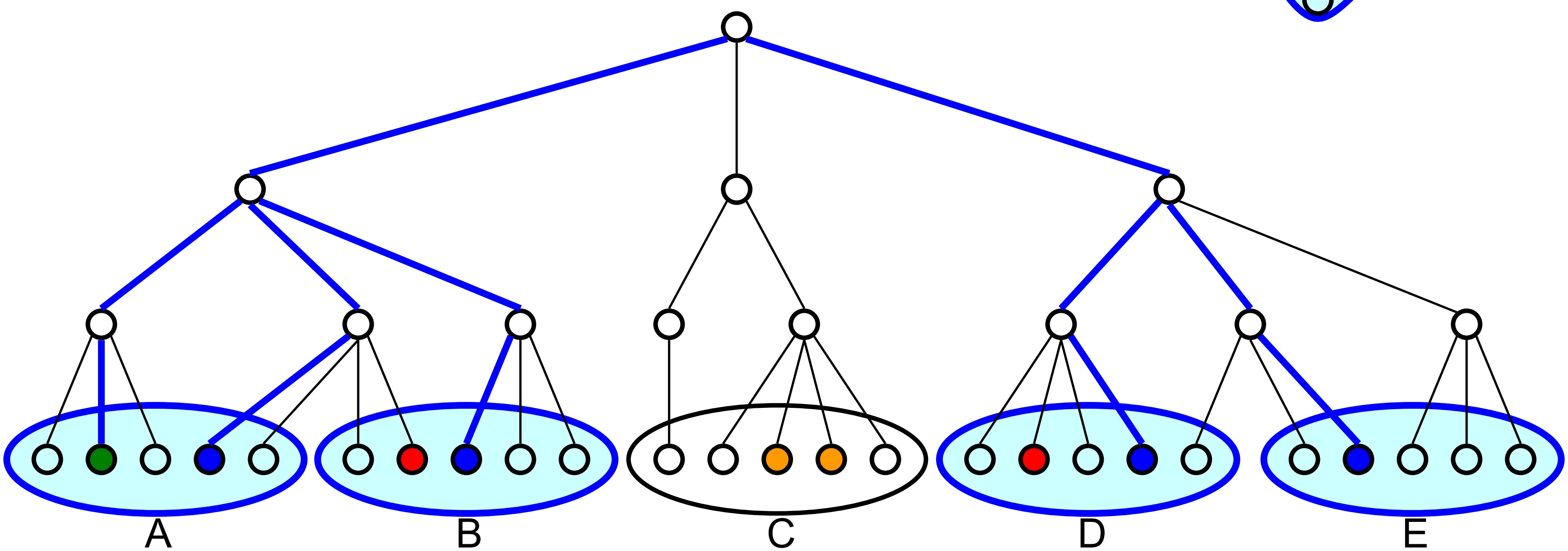
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. **Receive t input points (orange)**
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree



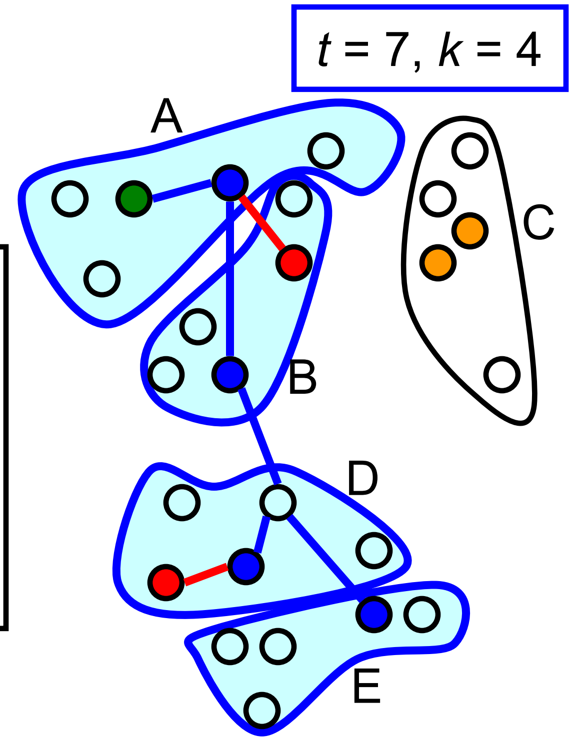
Algorithm

1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. **Accept a point if it is in a blue group – color it red**
8. Connect it to current Steiner tree

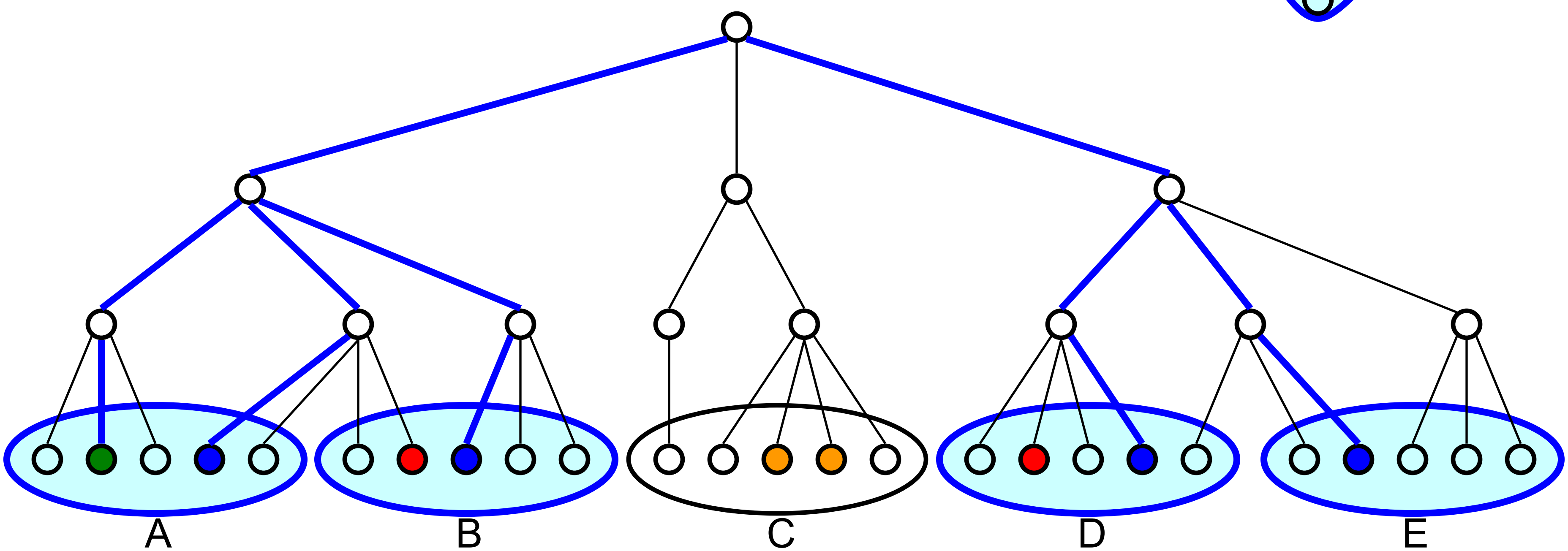




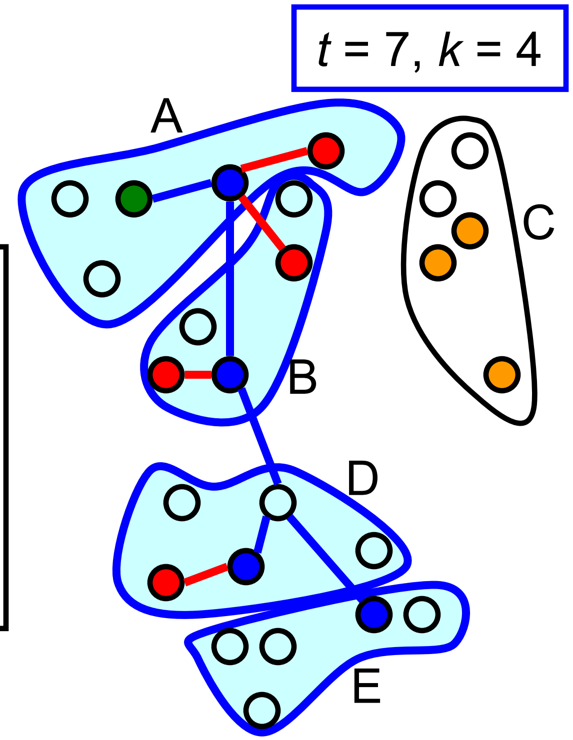
Algorithm



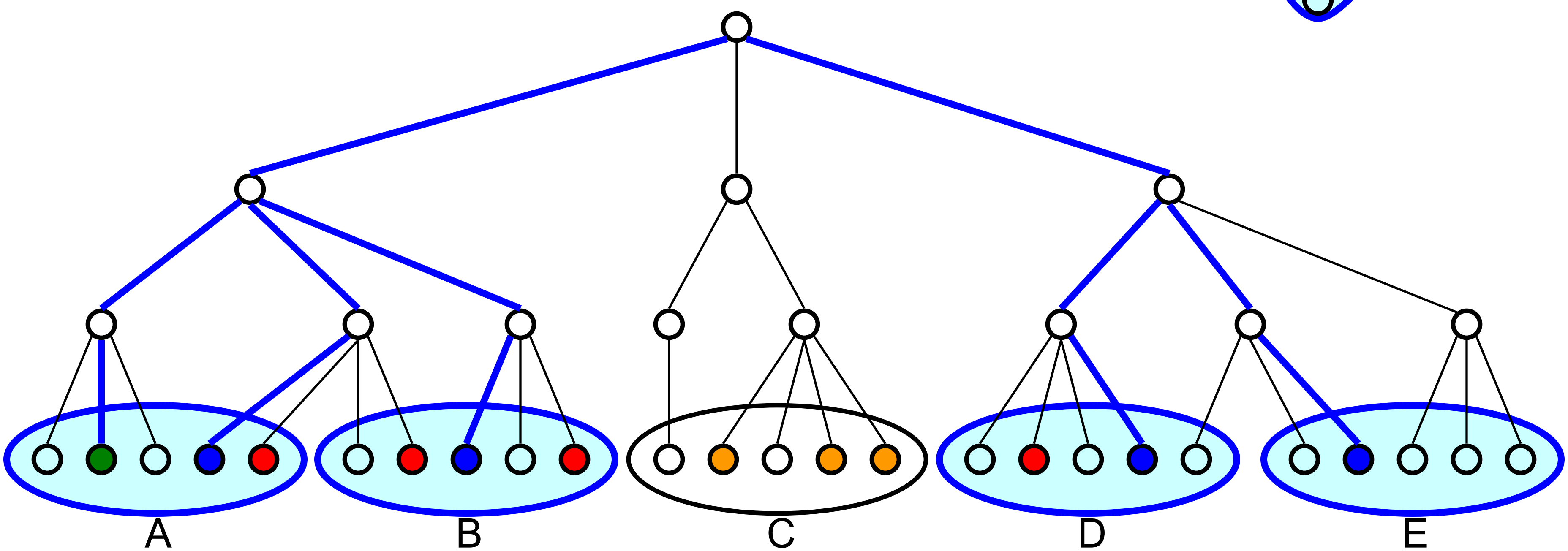
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \dfrac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. Accept a point if it is in a blue group – color it red
8. **Connect it to current Steiner tree**



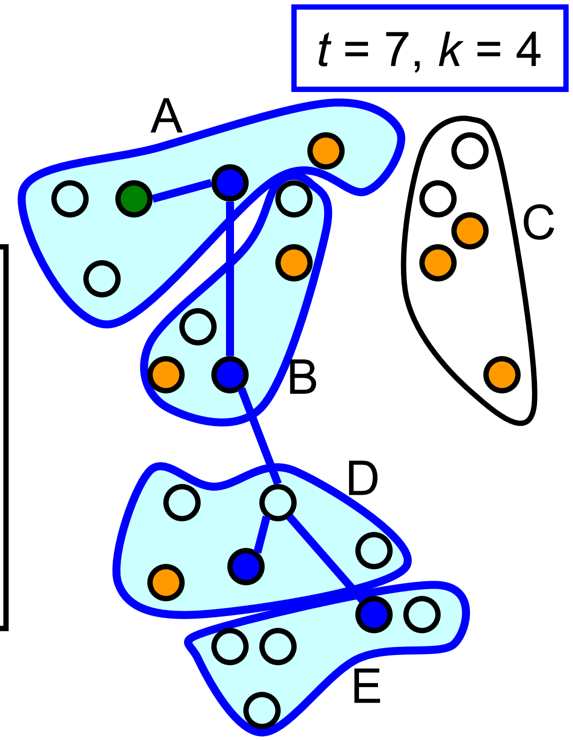
Algorithm



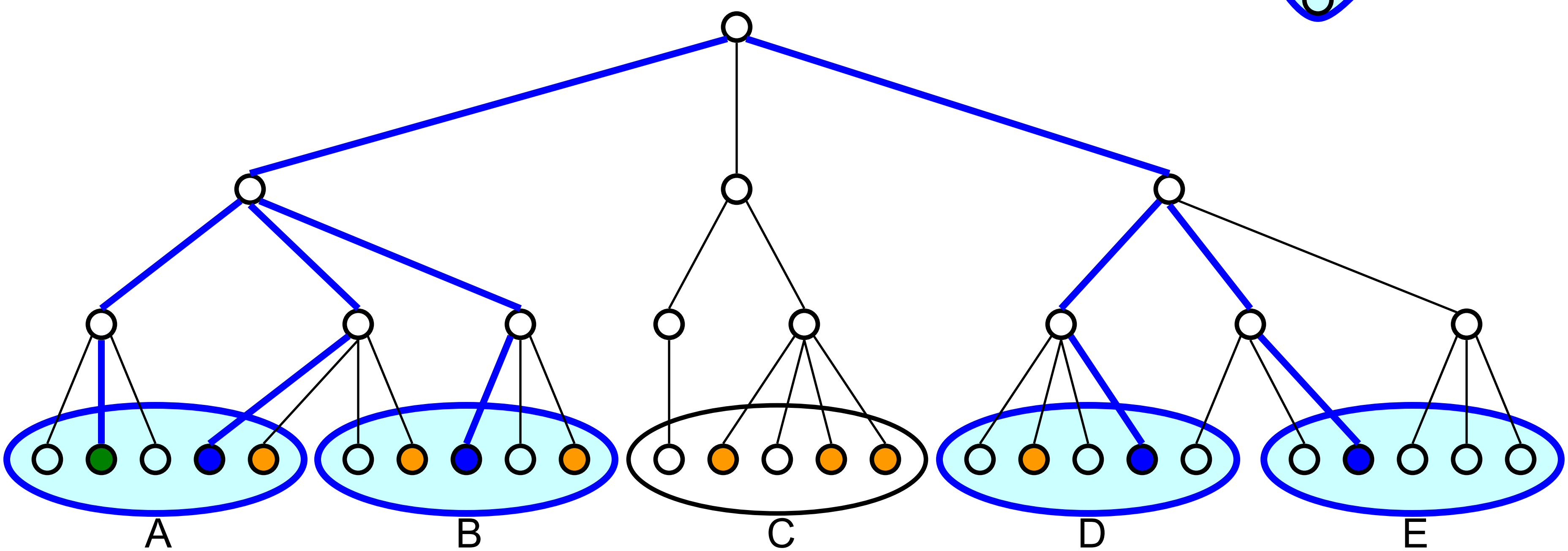
1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. **Accept a point if it is in a blue group – color it red**
8. **Connect it to current Steiner tree**



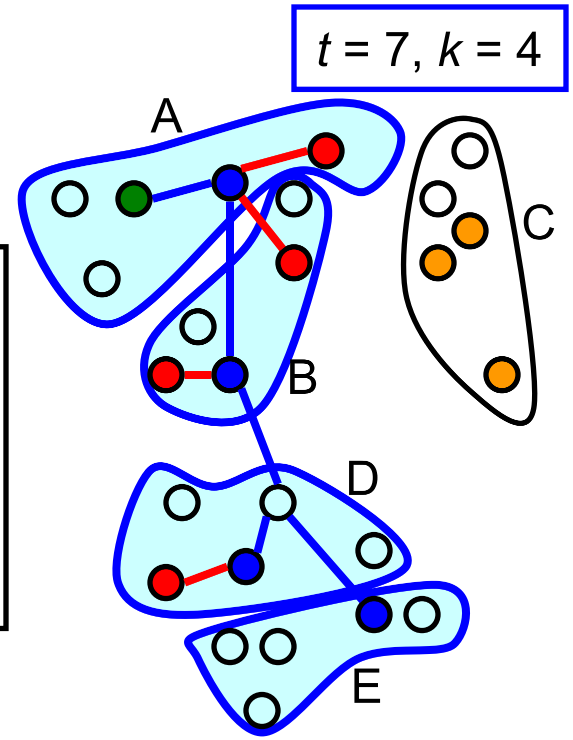
Algorithm



1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample *t* (light blue) nodes
4. Create approximate *k*-Steiner tree of *k* out of *t* points (blue points)
5. Blue group = group with at least one blue point
6. **Receive *t* input points (orange)**
7. Accept a point if it is in a blue group – color it red
8. Connect it to current Steiner tree

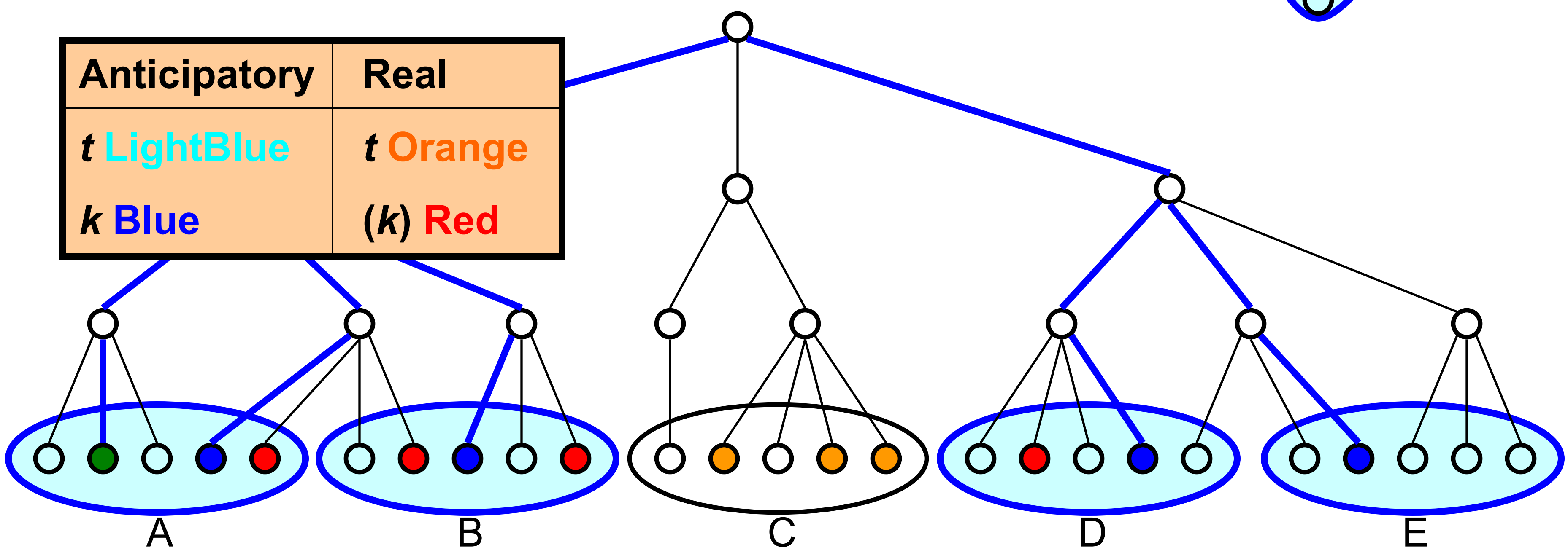


Algorithm



1. Create the Bartal tree
2. Group the leaves into groups of size $s = \frac{n}{t} \ln n$
3. Sample t (light blue) nodes
4. Create approximate k-Steiner tree of k out of t points (blue points)
5. Blue group = group with at least one blue point
6. Receive t input points (orange)
7. **Accept a point if it is in a blue group – color it red**
8. **Connect it to current Steiner tree**

Anticipatory	**Real**
***t* LightBlue**	***t* Orange**
***k* Blue**	**(*k*) Red**



Results

Anticipatory	Real
t **LightBlue**	***t*** **Orange**
k **Blue**	**(*k*)** **Red**

Theorem. There exists an algorithm such that for every $0 \le \varepsilon \le 1$ selects $(1 - \varepsilon)k$ request whp. with

$$\text{Ratio of Expectations} = O(\ln^2 n).$$

We show:

- ALG selects $(1 - \varepsilon)k$ requests (**Red** points) whp.
- $\mathbf{E}[\text{Cost(ALG)}] \le \mathbf{E}[\text{Cost(OPT)}] \cdot O(\ln^2 n)$

Number of Points

Anticipatory	Real
t **LightBlue**	***t*** **Orange**
k **Blue**	**(*k*) Red**

We show: ALG selects (1 – ε)*k* requests (Red points) whp.

- Size of group $s = \frac{n}{t} \ln n$
- $\Rightarrow \ \mathbf{Pr}(\text{LightBlue in a given group}) = \frac{s}{n} = \frac{1}{t} \ln n$
- $\Rightarrow \ \mathbf{E}[\#\ \text{LightBlue in a given group}] = \ln n$
- Whp (Chernoff):

$$\#\ \text{LightBlue in a group} \approx \ln n$$

- Whp:

$$\#\ \text{Orange in a group} \approx \ln n$$

Number of Points

Anticipatory	Real
t LightBlue	*t* Orange
k Blue	(*k*) Red

We show: ALG selects (1 – ε)*k* requests (Red points) whp.

- # LightBlue in a group $\approx \ln n$
- # Blue in a group $\leq$ # LightBlue in a group $\approx \ln n$
- k Blue points
- $\Rightarrow$ # Blue groups $\gtrsim \dfrac{k}{\ln n}$
- Also: # Orange in a group $\approx \ln n$
- $\Rightarrow$

$$\begin{aligned}\#\ \text{Red points} &= \#\ \text{Orange points in Blue groups}\\ &\gtrsim \frac{k}{\ln n} \cdot \ln n\\ &\approx k\\ &(\geq (1-\epsilon)k)\end{aligned}$$

Results

Anticipatory	Real
t **LightBlue**	***t*** **Orange**
k **Blue**	**(*k*) Red**

Theorem. There exists an algorithm such that for every $0 \le \varepsilon \le 1$ selects $(1 - \varepsilon)k$ request whp. with

$$\text{Ratio of Expectations} = O(\ln^2 n).$$

We show:

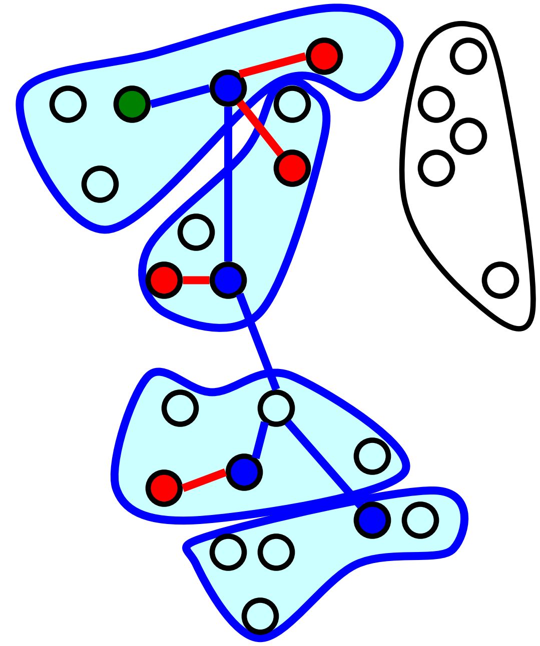
- ALG selects $(1 - \varepsilon)k$ requests (**Red** points) whp. ✓
- $\mathbf{E}[\text{Cost(ALG)}] \le \mathbf{E}[\text{Cost(OPT)}] \cdot O(\ln^2 n)$

Expected Cost

Anticipatory	Real
t LightBlue	*t* Orange
k Blue	(*k*) Red

We show: E[Cost(ALG)] ≤ E[Cost(OPT)] · O(ln² *n*)

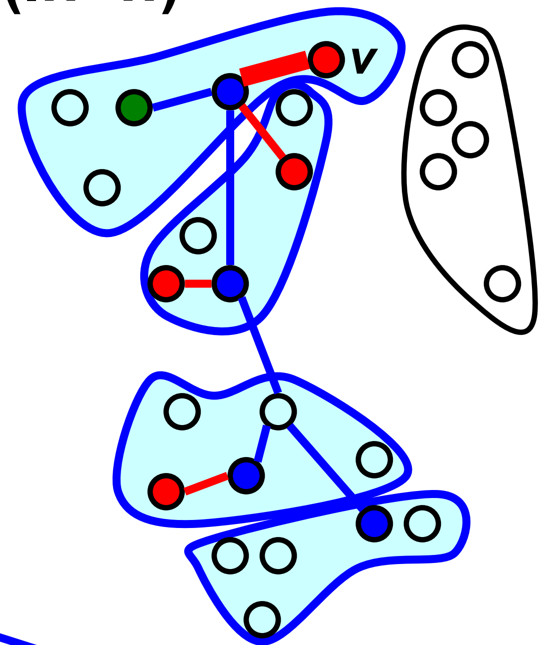
$$\mathbf{E}[\text{Cost}] = \mathbf{E}[\textbf{Blue}\ \text{Cost}] + \mathbf{E}[\textbf{Red}\ \text{Cost}]$$



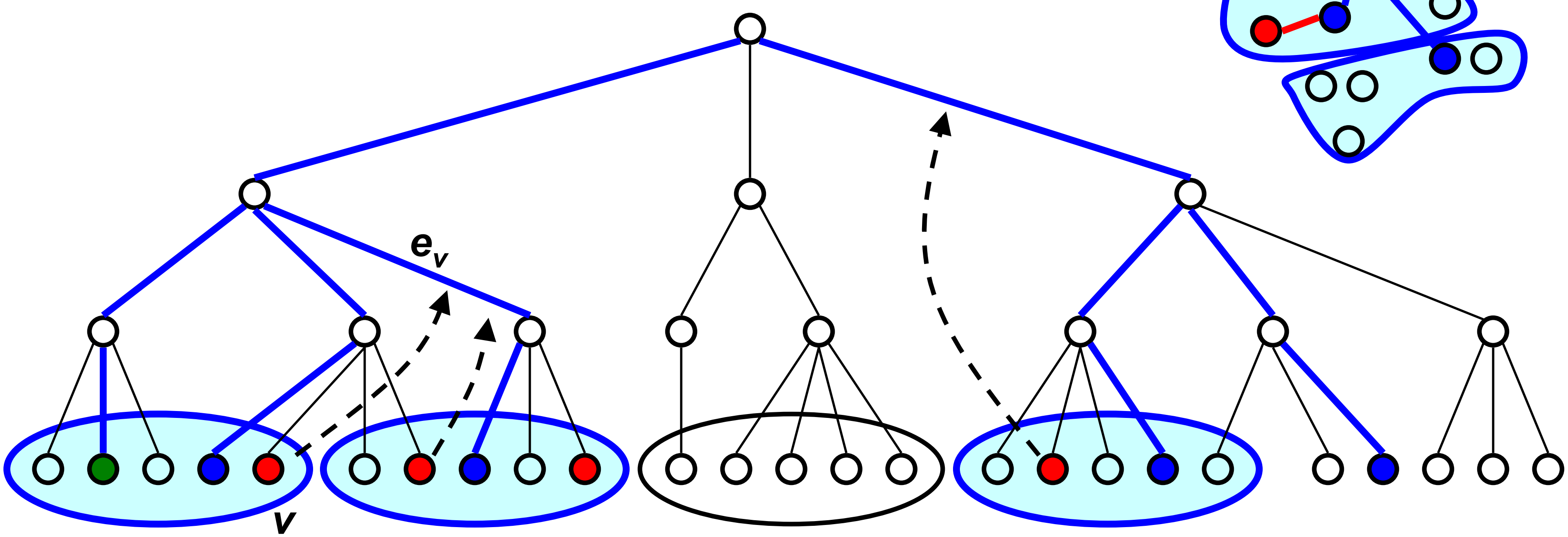
$$= \mathbf{E}[\text{Cost of approximate } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$\leq c \cdot \mathbf{E}[\text{Cost of optimal } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$= c \cdot \mathbf{E}[\text{Cost(OPT)}]$$

Red Cost

We show: E[Cost(ALG)] ≤ E[Cost(OPT)] · O(ln² *n*)



- Charge connecting **red** node ***v*** to an edge $\boldsymbol{e_v}$ of the Bartal tree
  - Connection cost of ***v*** ≤ cost of edge $\boldsymbol{e_v}$
  - Charge at most O(ln *n*) **red** nodes to each Bartal tree edge

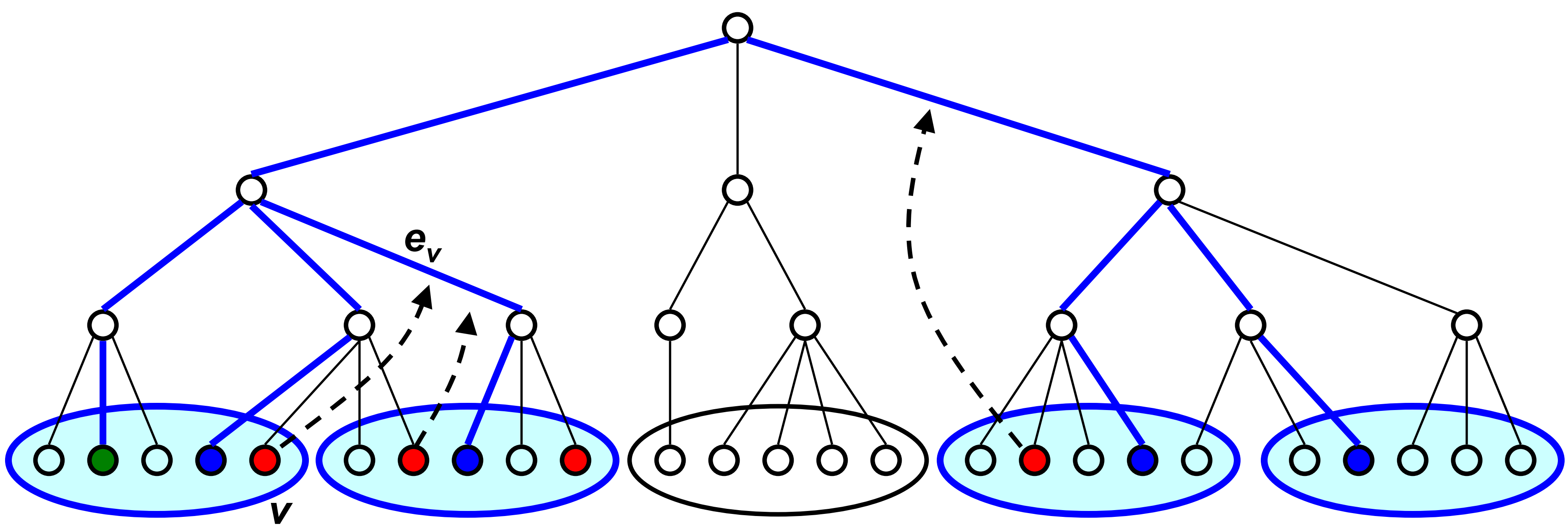


Red Cost

To each tree edge we charge O(ln *n*) **red** nodes

$\Rightarrow$ E[**Red** Cost] = O(ln *n*) · E[**Blue** Cost on Bartal tree)]

= O(ln² *n*) · E[**Blue** Cost (on the metric)]

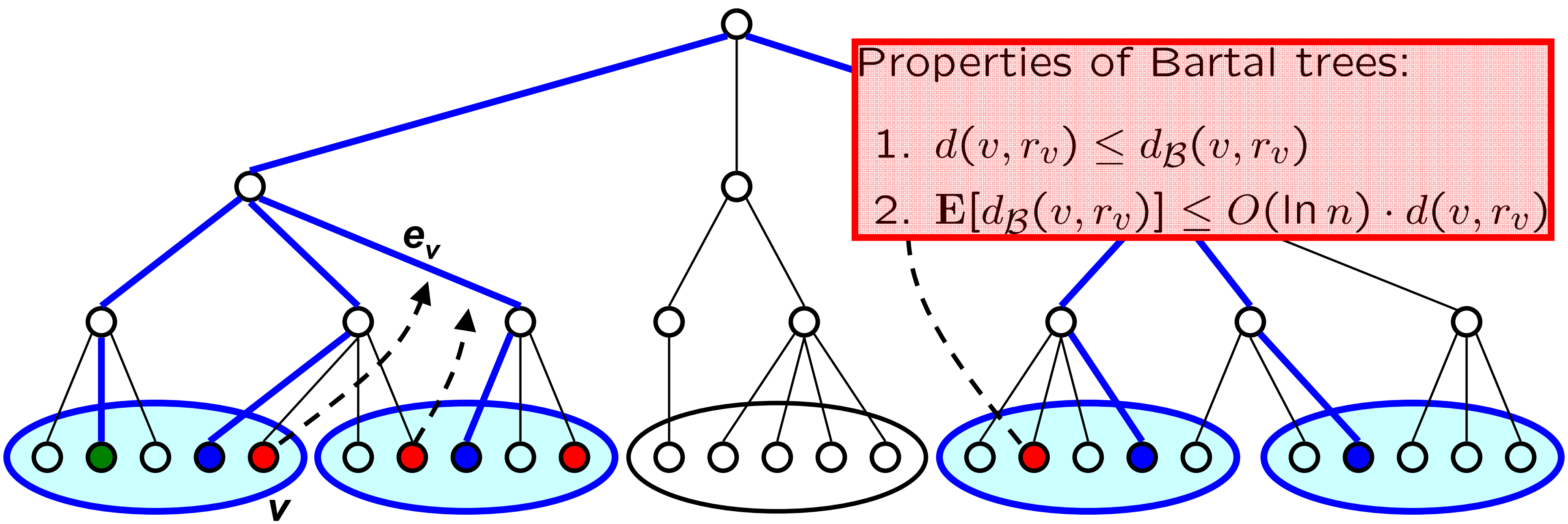


Red Cost

To each tree edge we charge O(ln *n*) **red** nodes

$\Rightarrow$ E[**Red** Cost] = O(ln *n*) · E[**Blue** Cost on Bartal tree)]

= O(ln^2 *n*) · E[**Blue** Cost (on the metric)]



Properties of Bartal trees:

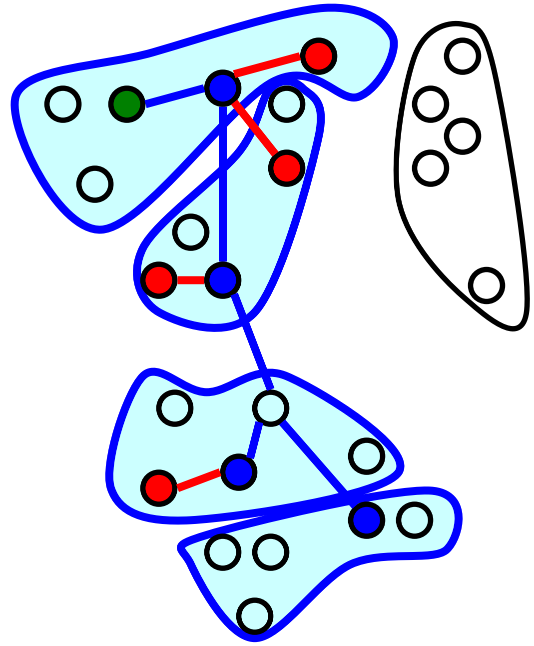
1. $d(v, r_v) \leq d_{\mathcal{B}}(v, r_v)$
2. $\mathbf{E}[d_{\mathcal{B}}(v, r_v)] \leq O(\ln n) \cdot d(v, r_v)$

Expected Cost

Anticipatory	Real
t LightBlue	*t* Orange
k Blue	(*k*) Red

We show: E[Cost(ALG)] ≤ E[Cost(OPT)] · O(ln² *n*)

$$\mathbf{E}[\text{Cost}] = \mathbf{E}[\textbf{Blue}\ \text{Cost}] + \mathbf{E}[\textbf{Red}\ \text{Cost}]$$



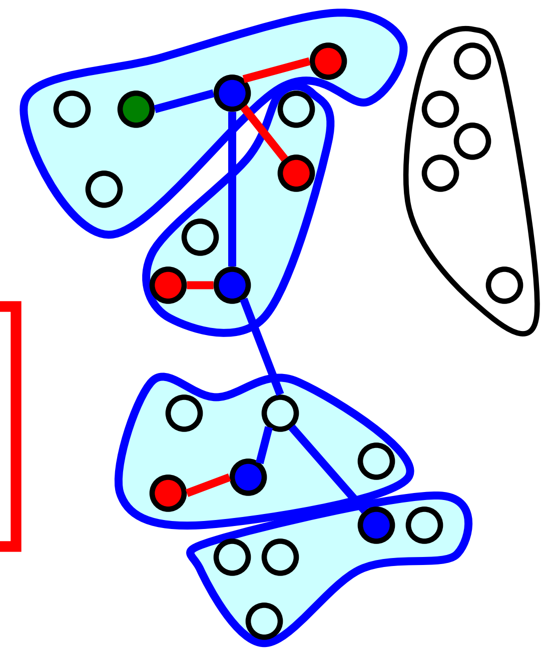
$$=\mathbf{E}[\text{Cost of approximate } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$\leq c \cdot \mathbf{E}[\text{Cost of optimal } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$= c \cdot \mathbf{E}[\text{Cost(OPT)}]$$

Expected Cost

Anticipatory	Real
t LightBlue	*t* Orange
k Blue	(*k*) Red

We show: E[Cost(ALG)] ≤ E[Cost(OPT)] · O(ln² *n*)

$$\mathbf{E}[\text{Cost}] = \mathbf{E}[\textbf{Blue}\ \text{Cost}] + \mathbf{E}[\textbf{Red}\ \text{Cost}]$$



$$= O(\ln^2 n) \cdot \mathbf{E}[\textbf{Blue}\ \text{cost}]$$
$$= O(\ln^2 n) \cdot \mathbf{E}[\text{Cost(OPT)}]$$

$$= \mathbf{E}[\text{Cost of approximate } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$\leq c \cdot \mathbf{E}[\text{Cost of optimal } k\text{-Steiner tree on } \textbf{LightBlue} \text{ points}]$$
$$= c \cdot \mathbf{E}[\text{Cost(OPT)}]$$

Results

Anticipatory	Real
t **LightBlue**	***t*** **Orange**
k **Blue**	**(*k*)** **Red**

Theorem. There exists an algorithm such that for every $0 \leq \varepsilon \leq 1$ selects $(1 - \varepsilon)k$ request whp. with

$$\text{Ratio of Expectations} = O(\ln^2 n).$$

We show:

- ALG selects $(1 - \varepsilon)k$ requests (**Red** points) whp. ✓
- $\mathbf{E}[\text{Cost(ALG)}] \leq \mathbf{E}[\text{Cost(OPT)}] \cdot O(\ln^2 n)$ ✓

Additional Results

For online Steiner tree with outliers:

- if $k = \Theta(t)$: $O(\ln n \ln \ln n)$
- Unknown distribution:
  - Upper bound: Same result ($O(\ln^2 n)$), under assumptions
  - Lower bound: $\Omega(\ln n)$

Same results for Online TSP

Same results for Facility Location

Conclusions

- Introduced the family of problems of **online network design with outliers**
- Much harder than versions without outliers
- Connection with secretary problems:
  - Minimization is strictly harder than maximization
- Studied problems of
  - Steiner tree
  - Facility location
  - TSP
- Gave approximation algorithms and close lower bounds

Open Problems

- Match upper and lower bounds
- Look at other problems
  - Price collecting Steiner tree
  - Set cover

Thanks!

Questions, etc.:

http://aris.me