To buy or not to buy: deterministic rent-or-buy problems on node-weighted graphs

STACS 2026

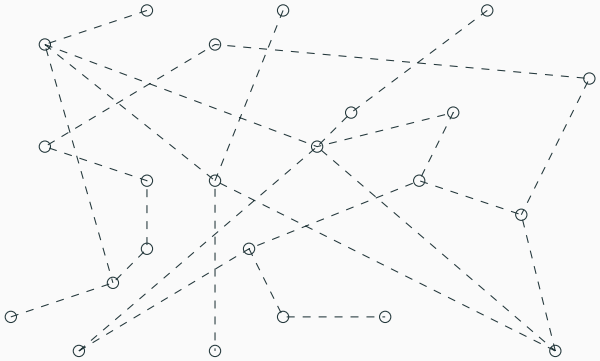
Sander Borst

MPI for Informatics, Saarbrücken

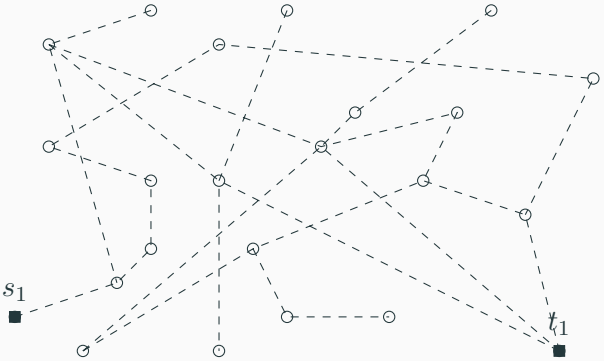
Moritz Venzin

Bocconi University, Milan

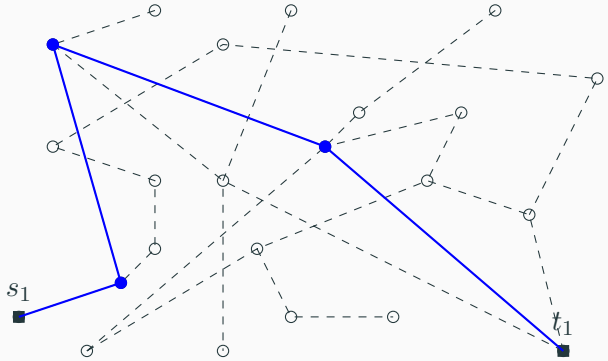
Online Steiner Forest



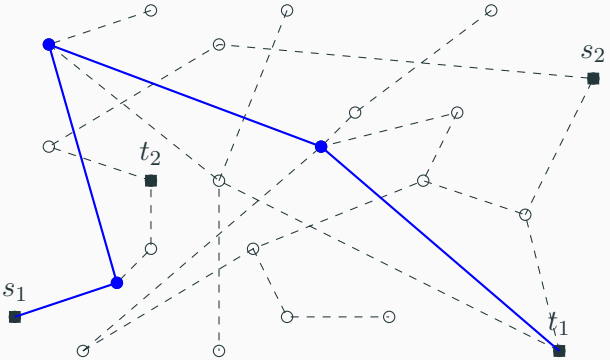
Online Steiner Forest



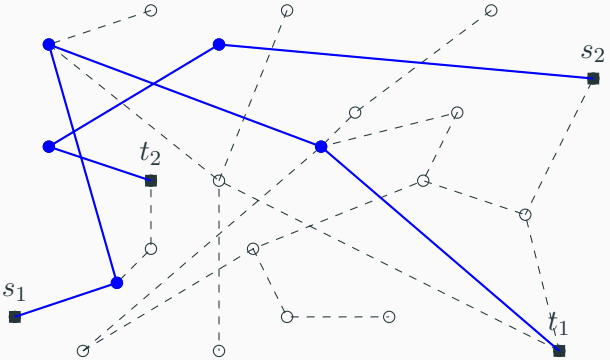
Online Steiner Forest



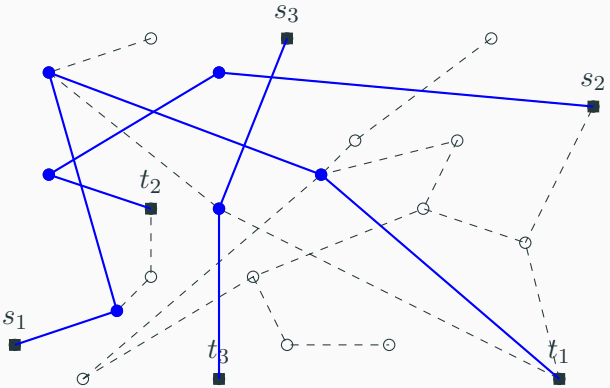
Online Steiner Forest



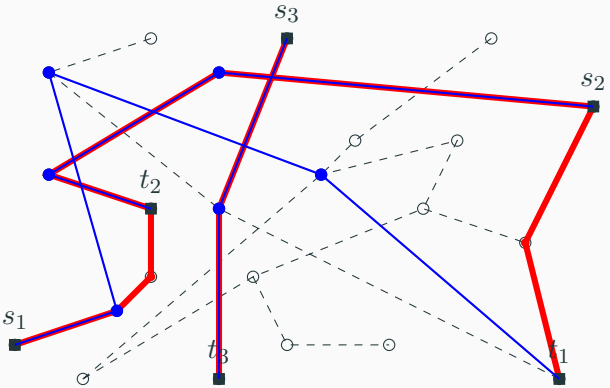
Online Steiner Forest



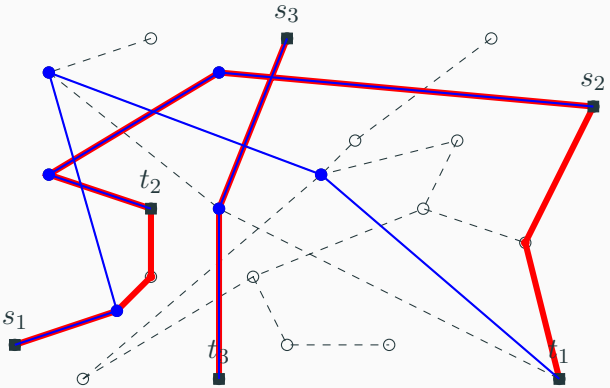
Online Steiner Forest



Online Steiner Forest

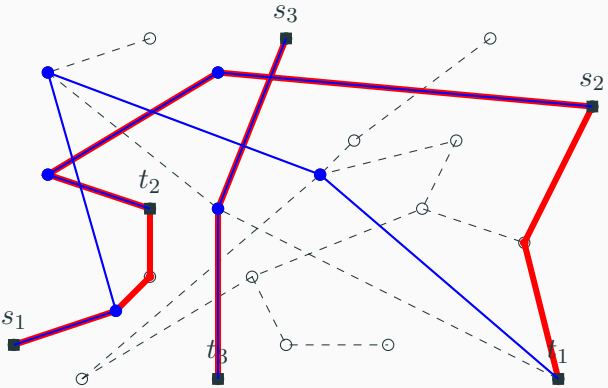


Online Steiner Forest



- Objective: buy edges/nodes to connect s_i and t_i for all i.
- Algorithm is α-competitive if cost(ALG) $\leq \alpha \cdot$ OPT for all inputs.
- Special case: online Steiner tree

Online Steiner Forest

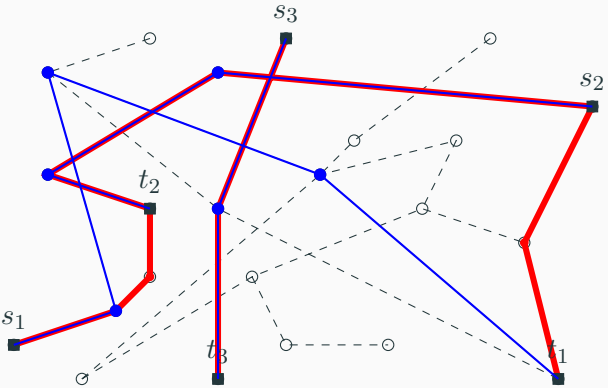


- Edge weights: $O(\log(k))$
  k: # node pairs

  (Imase, Waxman 1991)

- Objective: buy edges/nodes to connect s_i and t_i for all i.
- Algorithm is α-competitive if cost(ALG) $\leq \alpha \cdot$ OPT for all inputs.
- Special case: online Steiner tree

Online Steiner Forest



- Edge weights:
  $O(\log(k))$
  k: # node pairs
  (Imase, Waxman 1991)
- Node weights:
  $O(\log(n)^2)$
  n: # nodes
  (B., Eliáš, Venzin, 2025)

- Objective: buy edges/nodes to connect s_i and t_i for all i.
- Algorithm is α-competitive if cost(ALG) $\leq \alpha \cdot$ OPT for all inputs.
- Special case: online Steiner tree

Rent or Buy

- Rent an edge/node at cost c_e or buy at cost $M \cdot c_e$ for $M \geq 1$.
- Reusing bought edges/nodes for later requests is free.

Rent or Buy

- Rent an edge/node at cost c_e or buy at cost $M \cdot c_e$ for $M \geq 1$.
- Reusing bought edges/nodes for later requests is free.

Black-box reduction to buy-only (Awerbuch, Azar, Bartal 2004)

With probability $1/M$ buy using algorithm for buy-only, otherwise rent.

- If buy-only algorithm is c-competitive then the above is $O(c)$-competitive.
- Resulting algorithm is randomized.

How to derandomize?

- Deterministic c-competitive buy-only algorithm $\implies$ deterministic $O(c^2)$-competitive algorithm for rent-or-buy (Bartal, Charikar, Indyk 2001)
  - $O(\log(k)^2)$ for edge-weighted graphs.
  - $O(\log(n)^4)$ for node-weighted graphs.

How to derandomize?

- Deterministic c-competitive buy-only algorithm $\implies$ deterministic $O(c^2)$-competitive algorithm for rent-or-buy (Bartal, Charikar, Indyk 2001)
  - $O(\log(k)^2)$ for edge-weighted graphs.
  - $O(\log(n)^4)$ for node-weighted graphs.

- For edge-weighted graphs: deterministic $O(\log(k))$-competitive algorithm (Umboh 2014)

How to derandomize?

- Deterministic c-competitive buy-only algorithm $\implies$
  deterministic $O(c^2)$-competitive algorithm for rent-or-buy (Bartal, Charikar, Indyk 2001)
  - $O(\log(k)^2)$ for edge-weighted graphs.
  - $O(\log(n)^4)$ for node-weighted graphs.

- For edge-weighted graphs: deterministic $O(\log(k))$-competitive algorithm

  (Umboh 2014)

- Our main result:
  For node-weighted graphs: deterministic $O(\log(n)^2)$-competitive algorithm

This presentation

- Restrict to online Steiner tree for simplicity.

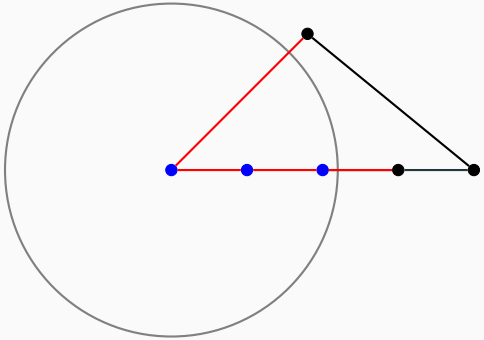
This presentation

- Restrict to online Steiner tree for simplicity.

1. Algorithm for edge-weighted case (Umboh 2014).
2. What makes node-weighted case more challenging?
3. Our algorithm for deterministic rent-or-buy on node-weighted graphs.
4. Open questions

Deterministic algorithm for edge-weighted graphs

- Let $B(v, r)$ be the set of all nodes within distance r of v ($B(v, r) = \{u \in V : d(u, v) \leq r\}$).
- Let $B_{\text{edge}}(v, r)$ be the set of all edges within distance r of v ($B(v, r) = \{(u, w) \in E : \min(d(u, v), d(w, v)) \leq r\}$).

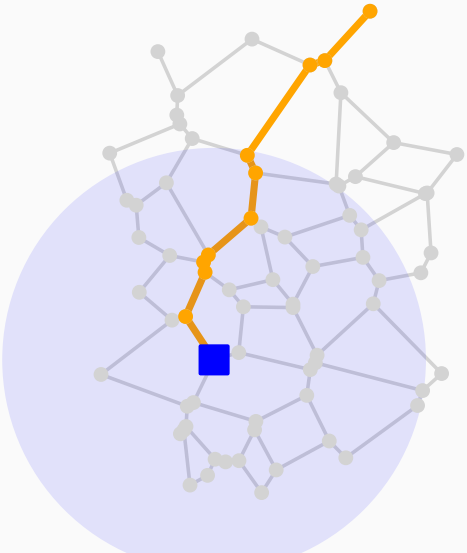


Deterministic algorithm for edge-weighted graphs

Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

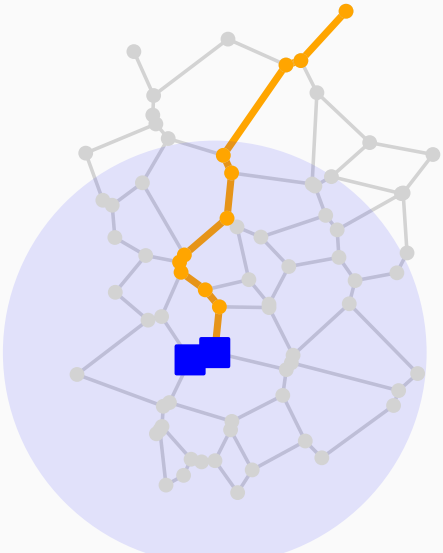
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

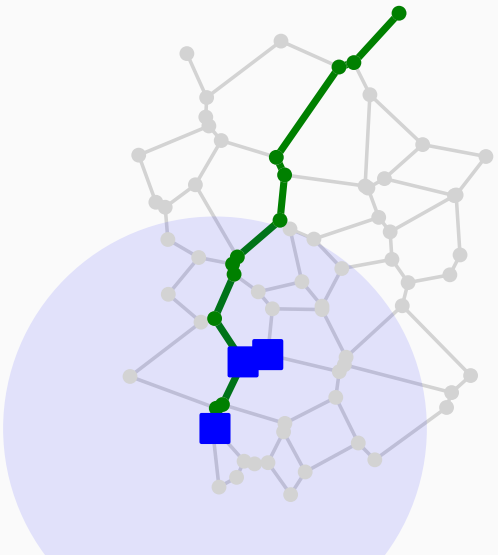
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

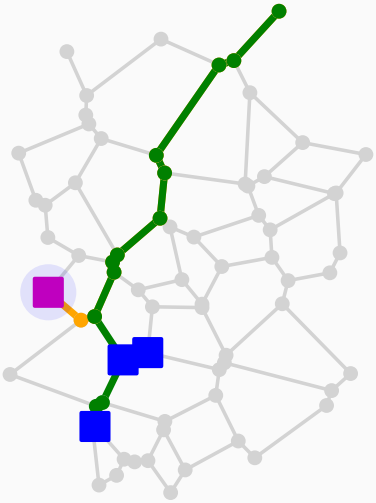
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

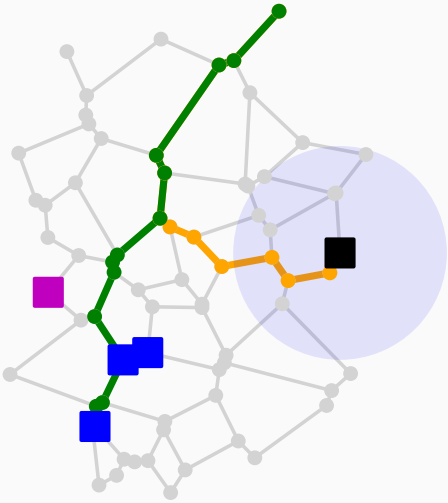
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

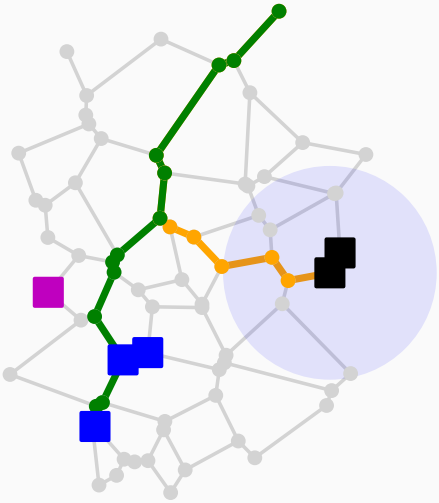
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

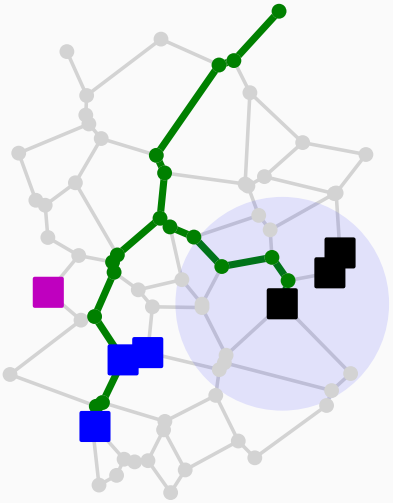
Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

Deterministic algorithm for edge-weighted graphs



Algorithm for edge-weighted case

1: **when** terminal i arrives **do**
2: Set j s.t. distance to bought tree $\approx 2^j$.
3: **if** $R_j \cap B(i, 2^{j-1}) \geq M$ **then**
4: Buy path to bought tree.
5: **else**
6: Rent path to bought tree.
7: Add i to R_j.

Analysis for edge-weighted graphs

Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every edge $e \in E$ is contained in at most M balls $B_{\text{edge}}(s, y_s)$, then:

$$\sum_{s \in S} y_s \leq \text{OPT}$$

Analysis for edge-weighted graphs

Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every edge $e \in E$ is contained in at most M balls $B_{\text{edge}}(s, y_s)$, then:

$$\sum_{s \in S} y_s \leq \text{OPT}$$

- Our construction ensures: $|B(s, 2^{j-1}) \cap R_j| \leq M$ for each $s \in R_j$.
- For each j, setting $y_s := 2^{j-2}\mathbf{1}_{s \in R_j}$ satisfies the above condition.
- So $2^{j-2}|R_j| \leq \text{OPT}$ for each j.

Analysis for edge-weighted graphs

Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every edge $e \in E$ is contained in at most M balls $B_{\text{edge}}(s, y_s)$, then:

$$\sum_{s \in S} y_s \leq \text{OPT}$$

- Our construction ensures: $|B(s, 2^{j-1}) \cap R_j| \leq M$ for each $s \in R_j$.
- For each j, setting $y_s := 2^{j-2}\mathbf{1}_{s \in R_j}$ satisfies the above condition.
- So $2^{j-2}|R_j| \leq \text{OPT}$ for each j.

- Total rental cost: $\sum_j 2^{j+1}|R_j| \leq 8\log(k) \cdot \text{OPT}$

Analysis for edge-weighted graphs

Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every edge $e \in E$ is contained in at most M balls $B_{\text{edge}}(s, y_s)$, then:

$$\sum_{s \in S} y_s \leq \text{OPT}$$

- Our construction ensures: $|B(s, 2^{j-1}) \cap R_j| \leq M$ for each $s \in R_j$.
- For each j, setting $y_s := 2^{j-2} \mathbf{1}_{s \in R_j}$ satisfies the above condition.
- So $2^{j-2}|R_j| \leq \text{OPT}$ for each j.

- Total rental cost: $\sum_j 2^{j+1}|R_j| \leq 8 \log(k) \cdot \text{OPT}$
- Charge bought paths to rental paths: we buy after renting $\geq M$ nearby paths.

Analysis for edge-weighted graphs

Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every edge $e \in E$ is contained in at most M balls $B_{\text{edge}}(s, y_s)$, then:

$$\sum_{s \in S} y_s \leq \text{OPT}$$

- Our construction ensures: $|B(s, 2^{j-1}) \cap R_j| \leq M$ for each $s \in R_j$.
- For each j, setting $y_s := 2^{j-2}\mathbf{1}_{s \in R_j}$ satisfies the above condition.
- So $2^{j-2}|R_j| \leq \text{OPT}$ for each j.

- Total rental cost: $\sum_j 2^{j+1}|R_j| \leq 8\log(k) \cdot \text{OPT}$
- Charge bought paths to rental paths: we buy after renting $\geq M$ nearby paths.

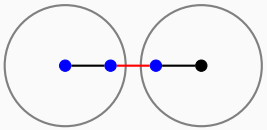
For node-weighted graphs

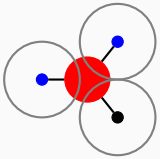
Lemma

For a set of terminals S with radius $y_s \geq 0$ for $s \in S$, if every node $e \in E$ is contained in at most M balls $B(s, y_s)$, then:

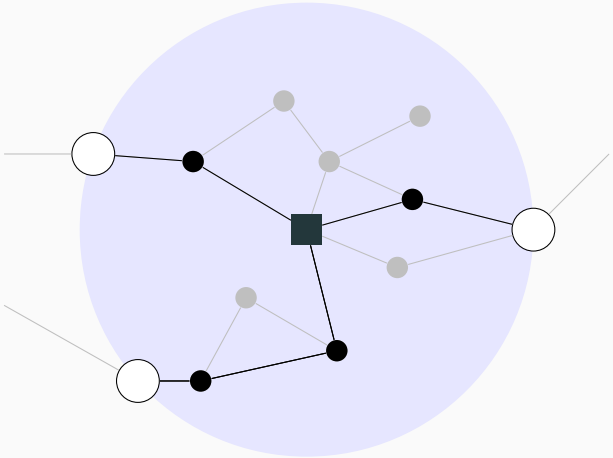
$$\sum_{s \in S} y_s \leq \mathsf{OPT}$$

- Our construction ensures: $|B(s, 2^{j-1}) \cap R_j| \leq M$ for each $s \in R_j$.
- For each j, setting $y_s := 2^{j-2}\mathbf{1}_{s \in R_j}$ no longer satisfies the above condition.





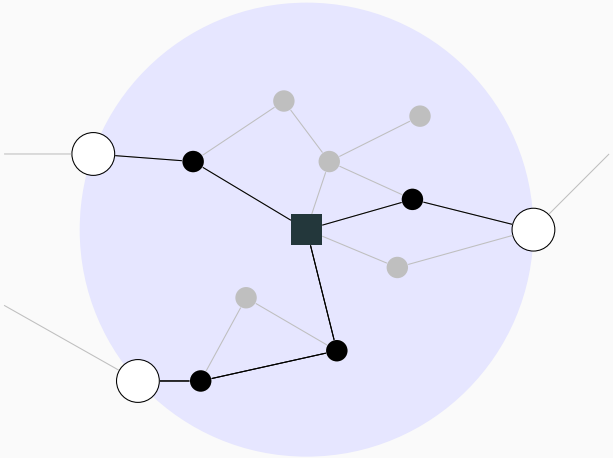
Choosing nodes on the boundary using prize-collecting set cover



- ■ Arriving terminal
- ○ Node on the bounary

- On arrival of terminal ■:
  Either:
  - buy boundary node ○
  - or pay a penalty for renting.

Choosing nodes on the boundary using prize-collecting set cover



- ■ Arriving terminal
- ○ Node on the bounary

- On arrival of terminal ■:
  Either:
  - buy boundary node ○
  - or pay a penalty for renting.
- Prize-collecting set cover
- $\log(n)\log(m)$-competitive online algorithm

(n: # sets, m: # elements)

Main result

Main result

There is a $\log n \log \bar{n}$ competitive deterministic algorithm for node-weighted + edge-weighted rent-or-buy. (n : #nodes, $\bar{n}$: #nodes with nonzero weight)

Main result

Main result

There is a $\log n \log \bar{n}$ competitive deterministic algorithm for node-weighted + edge-weighted rent-or-buy. (n : #nodes, $\bar{n}$: #nodes with nonzero weight)

Open questions

- Better deterministic black-box reduction from rent-or-buy to buy-only (for monotonic task systems)
- Buy-at-bulk (generalizing to concave node costs)
- Oblivious algorithms (not knowing M)

Open questions

- Better deterministic black-box reduction from rent-or-buy to buy-only (for monotonic task systems)
- Buy-at-bulk (generalizing to concave node costs)
- Oblivious algorithms (not knowing M)

Thanks for your attention!
Questions?