# The $(h, k)$-Server Problem on Bounded-Depth Trees

Nikhil Bansal[1], Marek Eliáš[1], Łukasz Jeż[2], and Grigorios Koumoutsos[1]

[1] TU Eindhoven
[2] University of Wrocław

# The $k$-server problem

- One of the central problems in Online Algorithms
- Introduced by Manasse, McGeoch, Sleator '90

# The $k$-server problem

- One of the central problems in Online Algorithms
- Introduced by Manasse, McGeoch, Sleator '90

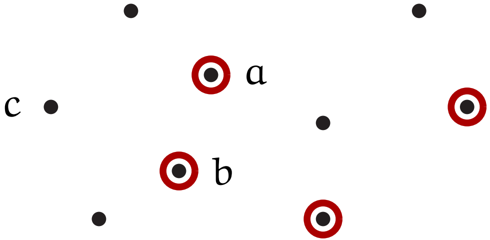
**Definition of the problem:**

- We have $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$
- Target: minimize the total distance traveled by the servers

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 0$:

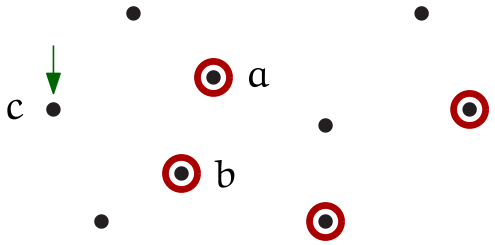



$$\sigma = (\ ?\ ,\ ?\ ,\ ?\ ,\ ?\ ,\ \dots\ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 1$:

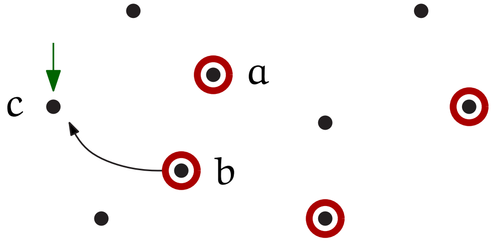



$$\sigma = ( \ c \ , \ ? \ , \ ? \ , \ ? \ , \ \ldots \ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 1$:



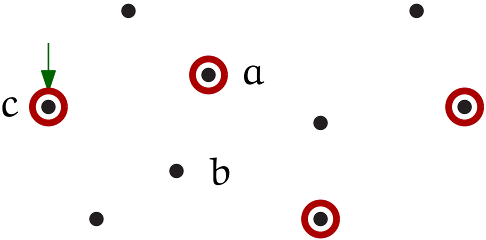

$$\sigma = ( \ c \ , \ ? \ , \ ? \ , \ ? \ , \ \ldots \ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 1$:

cost:
$\mathrm{dist}(c, b)$



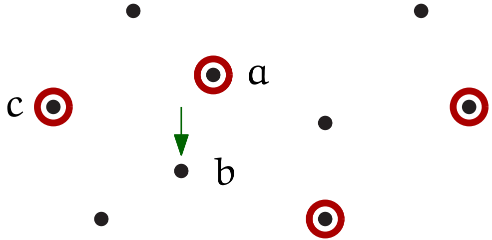

$$\sigma = (\ c\ ,\ ?\ ,\ ?\ ,\ ?\ ,\ \ldots\ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 2$:

cost:
$\mathrm{dist}(c, b)$



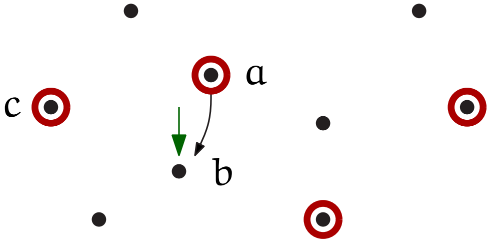

$$\sigma = ( \ c \ , \ b \ , \ ? \ , \ ? \ , \ \ldots \ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 2$:

cost:
$\mathrm{dist}(c, b)$



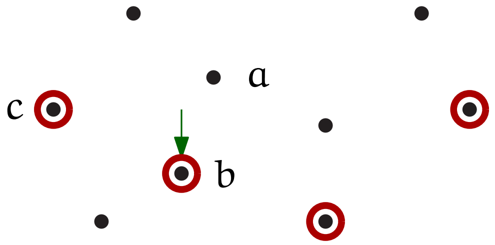

$$\sigma = ( \ c \ , \ b \ , \ ? \ , \ ? \ , \ \ldots \ )$$

# The $k$-server problem

- $k$ servers in a metric space $S$
- At time $t$:
  - Request arrives at some point $\sigma_t \in S$
  - We have to decide what server to move to $\sigma_t$

$t = 2$:

cost:
$\mathrm{dist}(c, b)$
$+\mathrm{dist}(a, b)$




$$\sigma = (\ c\ ,\ b\ ,\ ?\ ,\ ?\ ,\ \ldots\ )$$

# Competitive ratio

**Performance evaluation:**

$$R(\sigma) = \frac{ALG_k(\sigma)}{OPT_k(\sigma)}; \qquad \text{Competitive ratio} = \max_{\sigma}(R(\sigma))$$

- $ALG_k(\sigma)$: cost of the online algorithm with $k$ servers
- $OPT_k(\sigma)$: cost of the optimal offline solution for $k$ servers

# Competitive ratio

**Performance evaluation:**

$$R(\sigma) = \frac{ALG_k(\sigma)}{OPT_k(\sigma)}; \qquad \text{Competitive ratio} = \max_{\sigma}(R(\sigma))$$

- $ALG_k(\sigma)$: cost of the online algorithm with $k$ servers
- $OPT_k(\sigma)$: cost of the optimal offline solution for $k$ servers

**Achievable ratio is of order $\Theta(k)$ (in deterministic case):**

- LB: $k$ for any metric space of at least $k+1$ points (Manasse, McGeoch, Sleator '90)
- UB: $k$ for DC in tree metrics (Chrobak et al. '91; Chrobak, Larmore '91)
- UB: $2k-1$ for the Work Function Algorithm by (Koutsoupias, Papadimitriou '95)

# The $(h, k)$-server problem

**Servers as a precious resource:**

- Does adding new servers help to decrease the cost?
- Standard setting:
  - If we add more servers to ALG, we also compare it to OPT with more servers
  - Does not tell us whether the cost of ALG decreased or not

# The $(h, k)$-server problem

**Servers as a precious resource:**

- Does adding new servers help to decrease the cost?
- Standard setting:
  - If we add more servers to ALG, we also compare it to OPT with more servers
  - Does not tell us whether the cost of ALG decreased or not

**The $(h, k)$-server problem:**

- $h$: # of servers of OPT; $k$: # of servers of ALG
- Fix $h$ and add new servers only to ALG, i.e. $k > h$:

# The $(h, k)$-server problem

**Servers as a precious resource:**

- Does adding new servers help to decrease the cost?
- Standard setting:
  - If we add more servers to ALG, we also compare it to OPT with more servers
  - Does not tell us whether the cost of ALG decreased or not

**The $(h, k)$-server problem:**

- $h$: # of servers of OPT; $k$: # of servers of ALG
- Fix $h$ and add new servers only to ALG, i.e. $k > h$:

$$\frac{\mathrm{ALG}_h(\sigma)}{\mathrm{OPT}_h(\sigma)} \approx h \qquad\qquad \frac{\mathrm{ALG}_k(\sigma)}{\mathrm{OPT}_h(\sigma)} \approx \text{ constant?}$$

# Known results for the $(h, k)$-server problem

**Uniform metrics:**

- Corresponds to the paging problem
- Tight bounds of $k/(k-h+1)$ by Sleator, Tarjan '85
  - This equals 2 for $k = 2h$ and approaches 1 for $k \to \infty$
  - Later generalized to weighted star metrics by Young '94

# Known results for the $(h, k)$-server problem

**Uniform metrics:**

- Corresponds to the paging problem
- Tight bounds of $k/(k - h + 1)$ by Sleator, Tarjan '85
  - This equals 2 for $k = 2h$ and approaches 1 for $k \to \infty$
  - Later generalized to weighted star metrics by Young '94

**General metrics:**

- Lower bound of 2 irrespective of $k$ by Bar Noy and Schieber for the line
- Upper bound for WFA by Koutsoupias '99:
  - $2h$ for general metrics and $h + 1$ for line irrespective of $k$
- No $o(h)$ result known, even if $k \to \infty$

# Our results

**Lower bound of** $\Omega(h)$ **for** DC **and** WFA**:**

- In depth-2 trees for DC and depth-3 trees for WFA

# Our results

**Lower bound of $\Omega(h)$ for DC and WFA:**

- In depth-2 trees for DC and depth-3 trees for WFA

**General lower bound for depth-2 trees:**

- No algorithm can achieve a ratio better than 2.4 irrespective of $k$

# Our results

**Lower bound of $\Omega(h)$ for DC and WFA:**

- In depth-2 trees for DC and depth-3 trees for WFA

**General lower bound for depth-2 trees:**

- No algorithm can achieve a ratio better than 2.4 irrespective of $k$

**New (deterministic) algorithm for bounded-depth trees:**

- Its competitive ratio in a depth-$d$ tree is

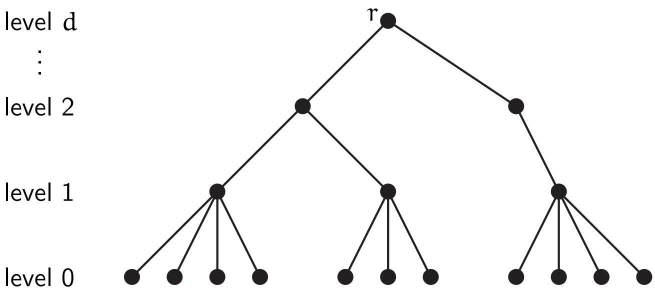
$$O\big(d \cdot 2^{d+1}\big) \qquad \text{for } k \to \infty$$
$$O\big(d \cdot (2d/\epsilon)^{d+1}\big) \qquad \text{for } k = (1+\epsilon)h$$

- First bound sublinear in $h$ for a metric which is not uniform

# Roadmap for the rest of this talk

1. Definition of the depth-$d$ trees
2. Lesson to learn from the lower bounds for DC and WFA
3. Description of our algorithm
4. New potential function based on Excess and Deficiency
5. Open problems

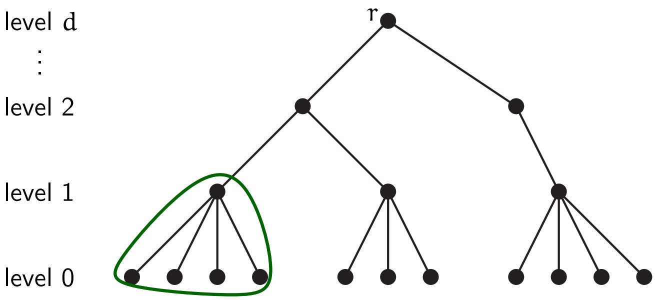
# Trees of bounded depth




**Depth-$d$ tree:**

- Rooted tree
- Each path from root to leaf has $d$ edges
- Requests only in leaves

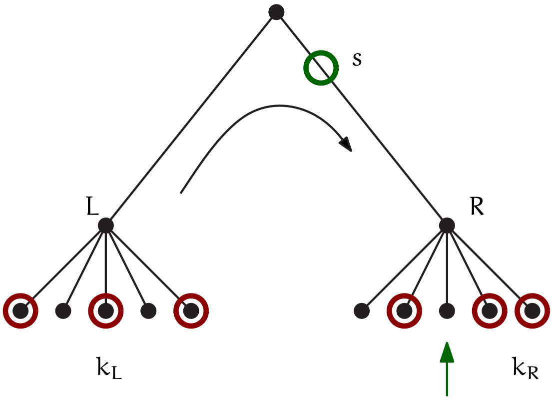
# Trees of bounded depth



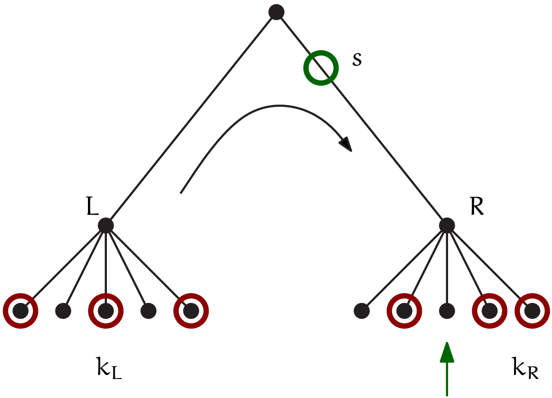

**Elementary subtrees:**

- Subtrees of depth one
- They form a uniform metric
- In uniform metric: if $k \geqslant 2h$, we can be 2-competitive (ST'85)

# Drawbacks of DC and WFA

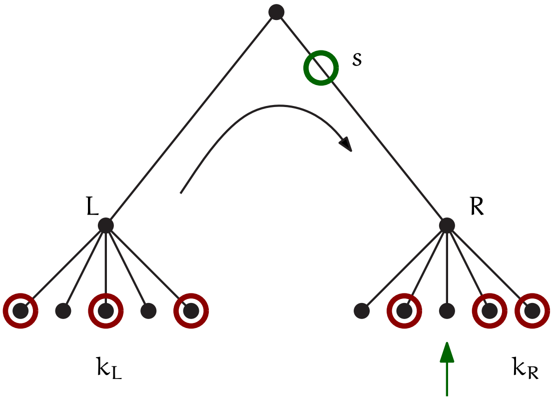


# Drawbacks of DC and WFA



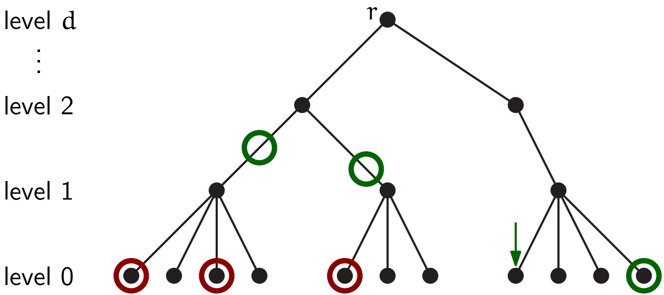

- DC and WFA bring help too slowly
- If cost of $x$ is incurred in the subtree R, DC moves $s$ by $\approx x/k_R$

# Drawbacks of DC and WFA




- DC and WFA bring help too slowly
- If cost of $x$ is incurred in the subtree R, DC moves $s$ by $\approx x/k_R$
- Our algorithm is more aggressive: we move $s$ by $\approx x$

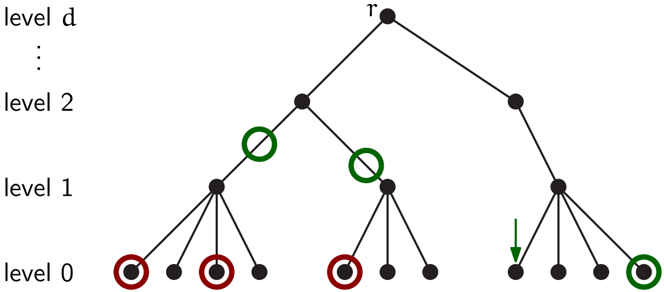
# New algorithm: Paradigm




**Basic paradigm of the algorithm:**

- Similar to DC by Chrobak et al.
- Servers are allowed to stay anywhere
- We move servers adjacent to the request

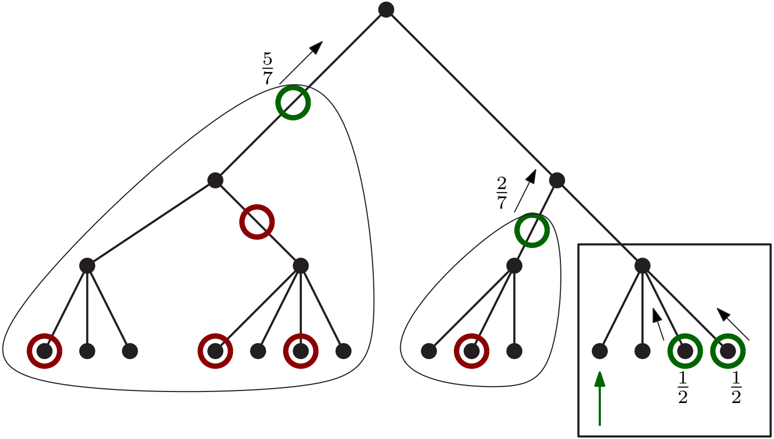
# New algorithm: Paradigm




**Basic paradigm of the algorithm:**

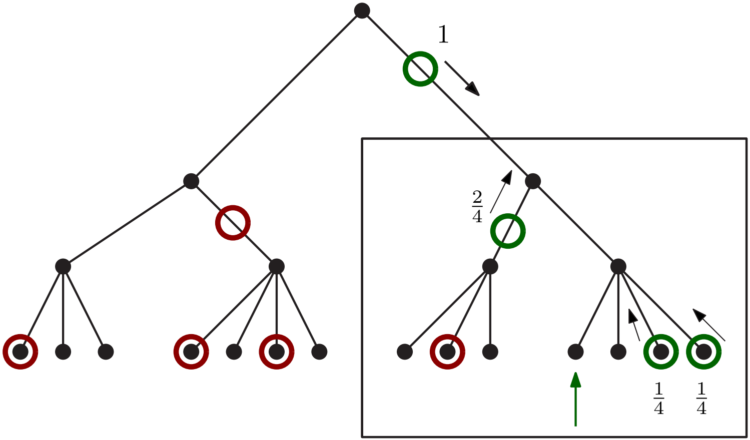
- Each adjacent server keeps moving towards the request until
  - its path to the request is blocked by some other server
  - request is served
- Key part of our algorithm: careful choice of the speeds

# Algorithm: Phase 1




- One unit of speed divided between the servers in the elementary subtree containing the requested point
- Second unit of speed is divided to the other incoming servers proportionally to the number of servers they represent

# Algorithm: Phase 2

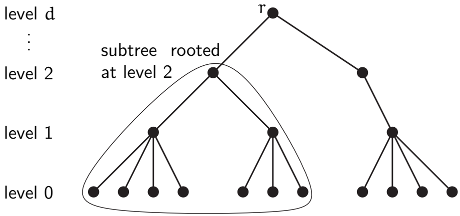



- The descending server is moving with the speed 1
- Second unit of speed is divided to the other incoming servers proportionally to the number of servers they represent

# Analysis: New potential function

**Excess and deficiency:**

- Let $T$ be a subtree rooted at a vertex at level $i$
- $k_T$: number of ALG's servers in $T$
- $h_T$: number of OPT's servers in $T$
- If $k_T \geqslant \beta h_T$, $T$ is excessive; otherwise it is deficient
- Excess threshold $\beta$ depends on $i$ and on $k/h$
  - $\beta = 2$ for elementary subtrees if $k/h$ is large
  - For subtrees rooted in higher levels $\beta$ increases geometrically



# Analysis: New potential function

**The new potential:**

$$\Phi = \sum_{i=1}^{d} (\alpha_i^E E_i + \alpha_i^D D_i)$$

- $E_i$ and $D_i$ measure the excess/deficiency in subtrees rooted at level $i$
- $\alpha_i^E$ and $\alpha_i^D$ are coefficients (dependent only on $i$ and $k/h$)

**Core of the analysis:**

- Proof that ALG always decreases enough excess or deficiency

# Open problems

**Is dependence on $d$ necessary?**

- A constant independent on $d$ would imply a randomized $O(\log n)$-competitive algorithm for any metric on $n$ points

**Can we achieve a similar result for other metrics?**

- e.g. line would be very interesting
  - Current best is $\frac{k}{k+1}(h+1)$ for DC

**Is there a metric which allows a lower bound bigger than 2.4?**

- Maybe even for trees of higher depth

Thank You

# Can the ratio become worse with $k$ increasing?

# Can the ratio become worse with $k$ increasing?

**Answer: Yes!**

# Can the ratio become worse with $k$ increasing?

**Answer: Yes!**

**Performance of DC algorithm in the line and trees:**

- In the standard setting optimal for line and trees
- Competitive ratio for DC is precisely

$$\frac{k}{k+1}(h+1)$$

- Equals $h$ for $k = h$, and increases towards $h + 1$ for $k \to \infty$

# Can the ratio become worse with $k$ increasing?

**Answer: Yes!**

**Performance of** DC **algorithm in the line and trees:**

- In the standard setting optimal for line and trees
- Competitive ratio for DC is precisely

$$\frac{k}{k+1}(h+1)$$

- Equals $h$ for $k = h$, and increases towards $h + 1$ for $k \to \infty$

**Competitive ratio of** WFA **in the line:**

- Equals $h$ for $k = h$, but increases to $h + 1/3$ for $k = 2h$.