A Lower bound advice for the competitiveness in the server problem in a k-star with k-servers with advice

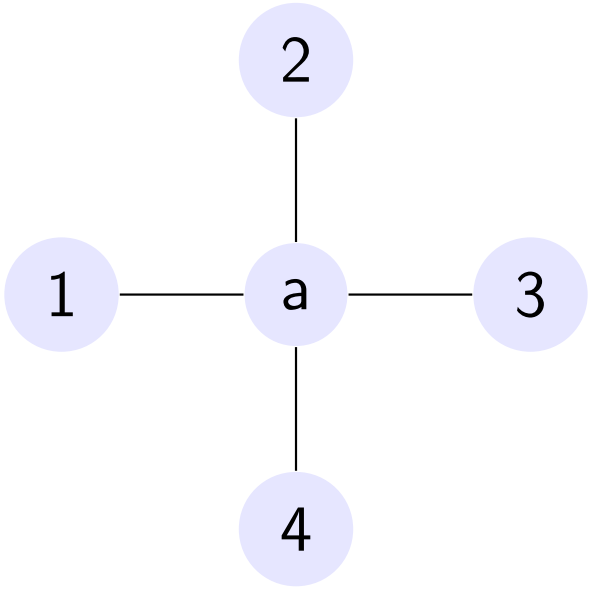
Juraj Hromkovič, Xavier Muñoz and Elisabet Burjons

Departament de Matemàtica Aplicada IV
Universitat Politècnica de Catalunya

Index

1. Introduction
2. Lower bound advice for r-competitiveness

k-server problem in a k-star graph



Request sequence example: $a, 1, 3, 4, a, a, 4, 2, a, \ldots$
Competitive ratio:

$$r = \frac{\text{"number of steps made by the algorithm"}}{\text{"optimal number steps to fulfill the request"}}$$

Advice

- The advice consists of some bits that give information to the algorithm about the request sequence
- They allow the algorithm to be more optimal than it could possibly be without advice. (They decrease the competitive ratio).
- What we present here is a lower bound on the advice bits needed to guarantee that there exists a deterministic algorithm that will be at most r-competitive.

Advice

- The advice consists of some bits that give information to the algorithm about the request sequence
- They allow the algorithm to be more optimal than it could possibly be without advice. (They decrease the competitive ratio).
- What we present here is a lower bound on the advice bits needed to guarantee that there exists a deterministic algorithm that will be at most r-competitive.

Advice

- The advice consists of some bits that give information to the algorithm about the request sequence
- They allow the algorithm to be more optimal than it could possibly be without advice. (They decrease the competitive ratio).
- What we present here is a lower bound on the advice bits needed to guarantee that there exists a deterministic algorithm that will be at most r-competitive.

Lower bound advice for *r*-competitiveness

Theorem

Considering the k-server problem in a k-star. Any deterministic algorithm with at most l advice bits, has a competitive ratio of at least:

$$r \geq \frac{k + 2^l - 1}{2^l}$$

Lower bound advice for r-competitiveness

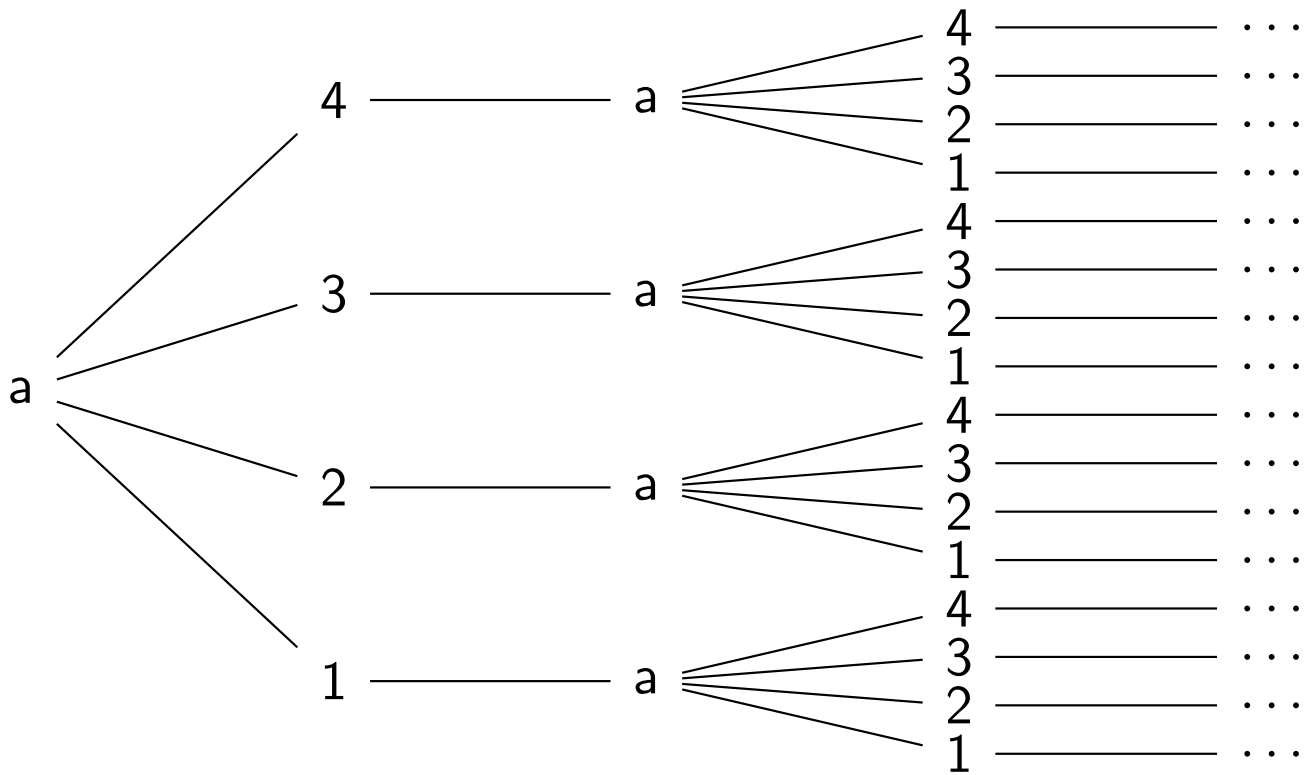
Theorem

Considering the k-server problem in a k-star. Any deterministic algorithm with at most l advice bits, has a competitive ratio of at least:

$$r \geq \frac{k + 2^l - 1}{2^l}$$

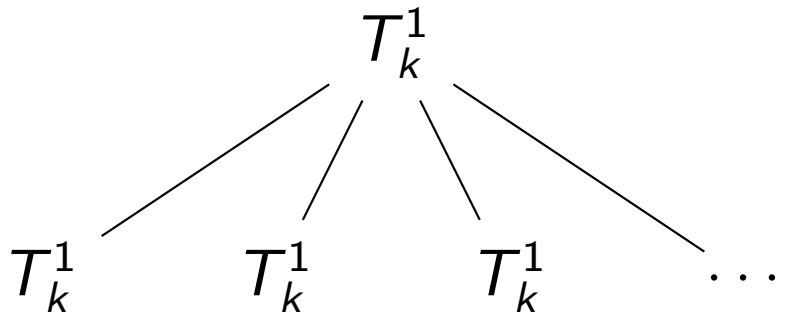
Case with $l = 0$

Set of request sequences S_k:
$a, i_1, a, i_2, a, \ldots$ where i_j are leaves of the star tree.
Let us build a tree that visualizes the request sequences in S_k:



Case with $l = 1$

If we name the request sequence with heigh $2k$ as T_k^1, then the request tree with heigh $4k$, T_k^2 can be represented as:



Meaning that at each of the leafs of the upper T_k^1 hangs another T_k^1.

Case with $l = 1$

Goal: $r \geq \frac{k+2^l-1}{2^l} = \frac{k+1}{2}$ Now we can interpret the advice bit as a color 0 or 1 that is assigned to each request sequence in the tree. One of these will happen:

- One of the son trees T_k^1 has been advised the same color for all leaves (advise useless in the lower subtree, so in the worst case at least $2k + 2$ moves must be made).
- Each of the son trees T_k^1 has been advised both colors at least once (advice useless in the upper subtree, so in the worst case at least $2k + 2$ moves must be made).

Case with $l = 1$

Goal: $r \geq \frac{k+2^l-1}{2^l} = \frac{k+1}{2}$ Now we can interpret the advice bit as a color 0 or 1 that is assigned to each request sequence in the tree. One of these will happen:

- One of the son trees T_k^1 has been advised the same color for all leaves (advise useless in the lower subtree, so in the worst case at least $2k + 2$ moves must be made).
- Each of the son trees T_k^1 has been advised both colors at least once (advice useless in the upper subtree, so in the worst case at least $2k + 2$ moves must be made).

Case with $l = 1$

Goal: $r \geq \frac{k+2^l-1}{2^l} = \frac{k+1}{2}$ Now we can interpret the advice bit as a color 0 or 1 that is assigned to each request sequence in the tree. One of these will happen:

- One of the son trees T_k^1 has been advised the same color for all leaves (advise useless in the lower subtree, so in the worst case at least $2k + 2$ moves must be made).
- Each of the son trees T_k^1 has been advised both colors at least once (advice useless in the upper subtree, so in the worst case at least $2k + 2$ moves must be made).

Case with $l = 1$

Goal: $r \geq \frac{k+2^l-1}{2^l} = \frac{k+1}{2}$ Now we can interpret the advice bit as a color 0 or 1 that is assigned to each request sequence in the tree. One of these will happen:

- One of the son trees T_k^1 has been advised the same color for all leaves (advise useless in the lower subtree, so in the worst case at least $2k + 2$ moves must be made).
- Each of the son trees T_k^1 has been advised both colors at least once (advice useless in the upper subtree, so in the worst case at least $2k + 2$ moves must be made).

The induction step follows straightforward with the ideas given for the case with one bit of advice.

Bound for the number of advice bits

Now, taking the reciprocal of the theorem we have just proved:

Corollary

Given a deterministic algorithm such that:

$$r < \frac{k + 2^{l-1} - 1}{2^{l-1}} \implies \textit{it uses at least } l \textit{ advice bits.}$$

Bound for the number of advice bits

Now, taking the reciprocal of the theorem we have just proved:

Corollary

Given a deterministic algorithm such that:

$$r < \frac{k + 2^{l-1} - 1}{2^{l-1}} \implies \textit{it uses at least } l \textit{ advice bits.}$$

This provides a lower bound on the number of bits needed to have a given competitive ratio.

Bound for the number of advice bits

Now, taking the reciprocal of the theorem we have just proved:

Corollary

Given a deterministic algorithm such that:

$$r < \frac{k + 2^{l-1} - 1}{2^{l-1}} \implies \textit{it uses at least } l \textit{ advice bits.}$$

This provides a lower bound on the number of bits needed to have a given competitive ratio.
This bound is independent of the length of the sequence.

Thank you for your attention!