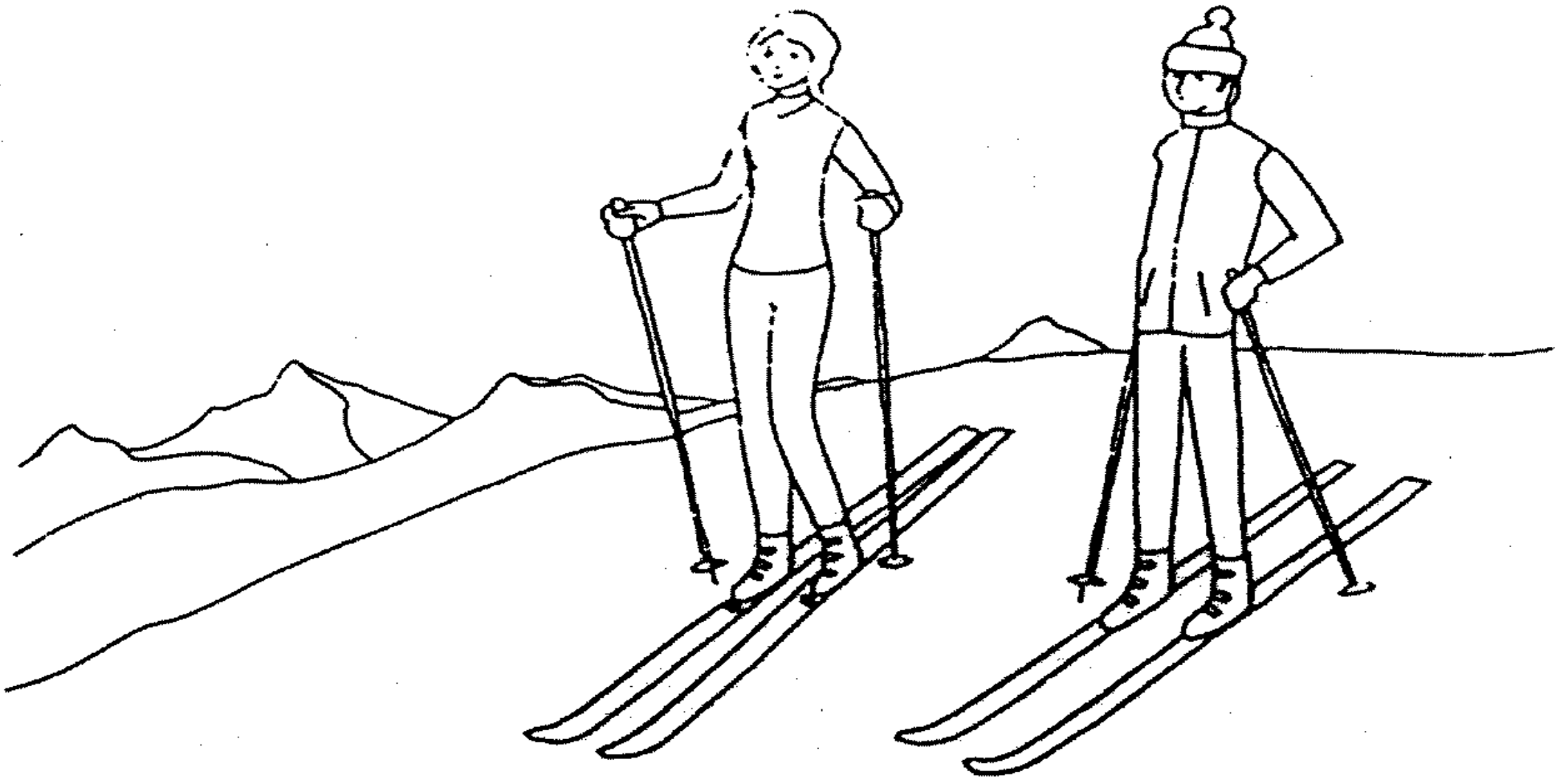


Online Algorithms for Network Problems

Susanne Albers
University of Freiburg
Germany

Ski rental

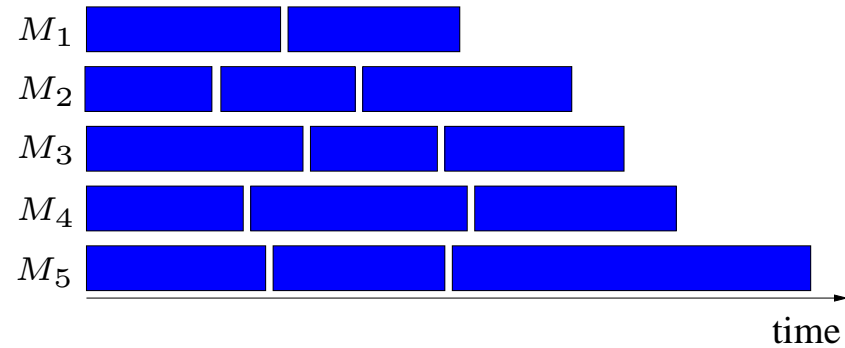
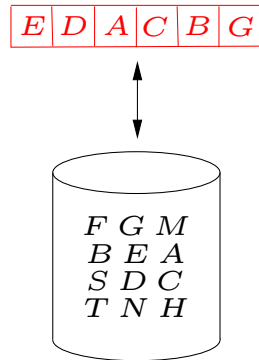


Areas of research

Data structures: Self-organizing lists and trees



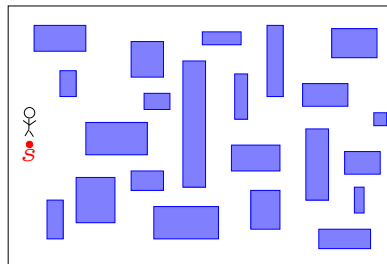
Paging/caching: Operating systems, distributed systems



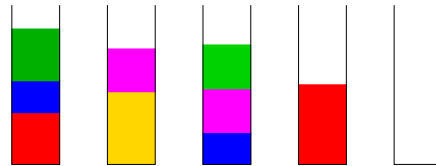
Scheduling: Makespan, flow time, ...

Areas of research

Robotics: Navigation, exploration and localization



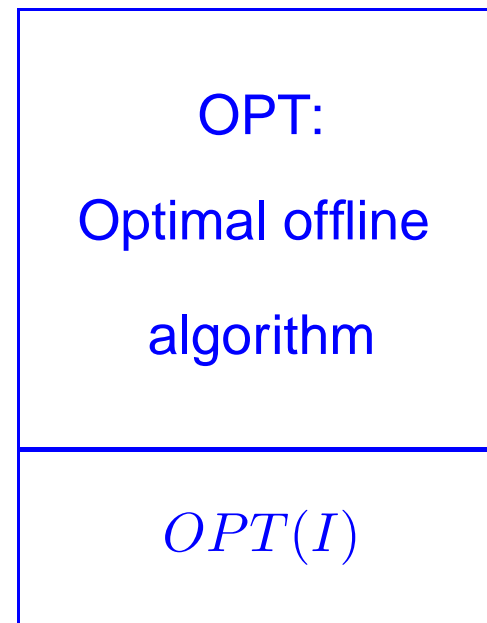
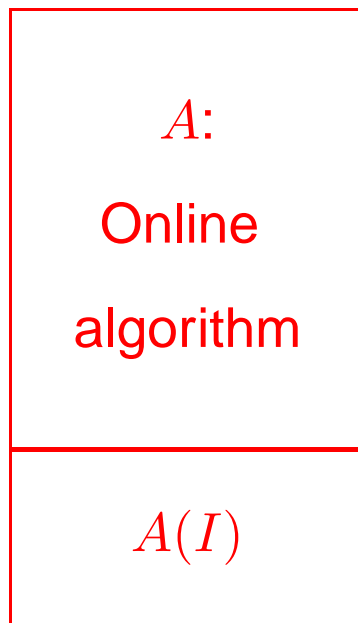
Bin packing: One- or multi-dimensional packing problems



Financial problems: Currency conversion, rent-or-buy problems

Competitive analysis

Online problem: input I



A is c -competitive if $\exists b$ s.t. for all inputs I

$$A(I) \leq c \cdot OPT(I) + b.$$

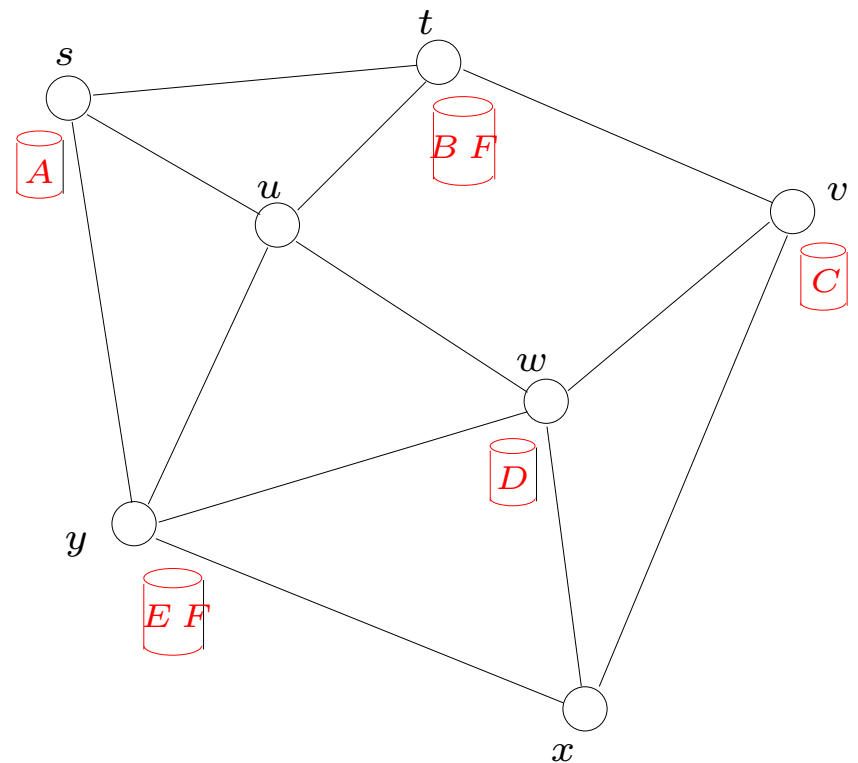
Replication, migration

Request: (x, F) x requests
data item from file F

F not available at x : $dist(x, y)$
 y closest node storing F

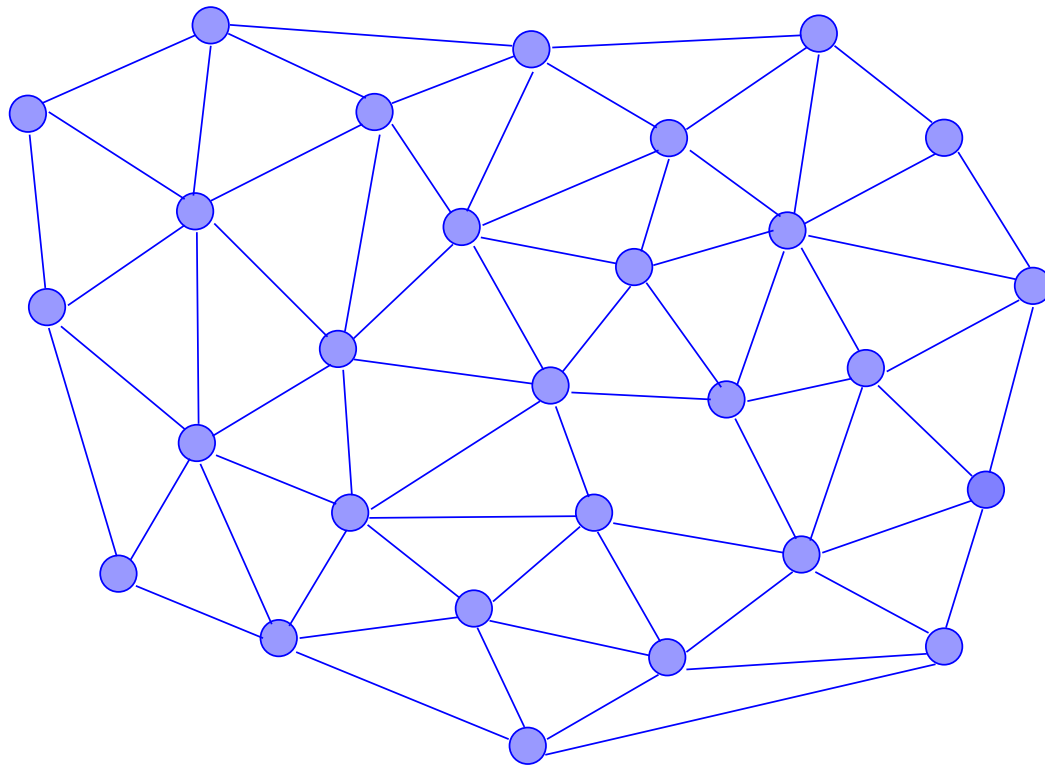
Migration, replication of F from x to y :
 $dist(x, y) \text{Size}(F)$

Goal: Minimize total cost



$$\sigma = (x, F) (y, E) (z, A) (v, C) (w, B) \dots$$

Large networks



Replication, migration:

Maintain multiple copies of a file.

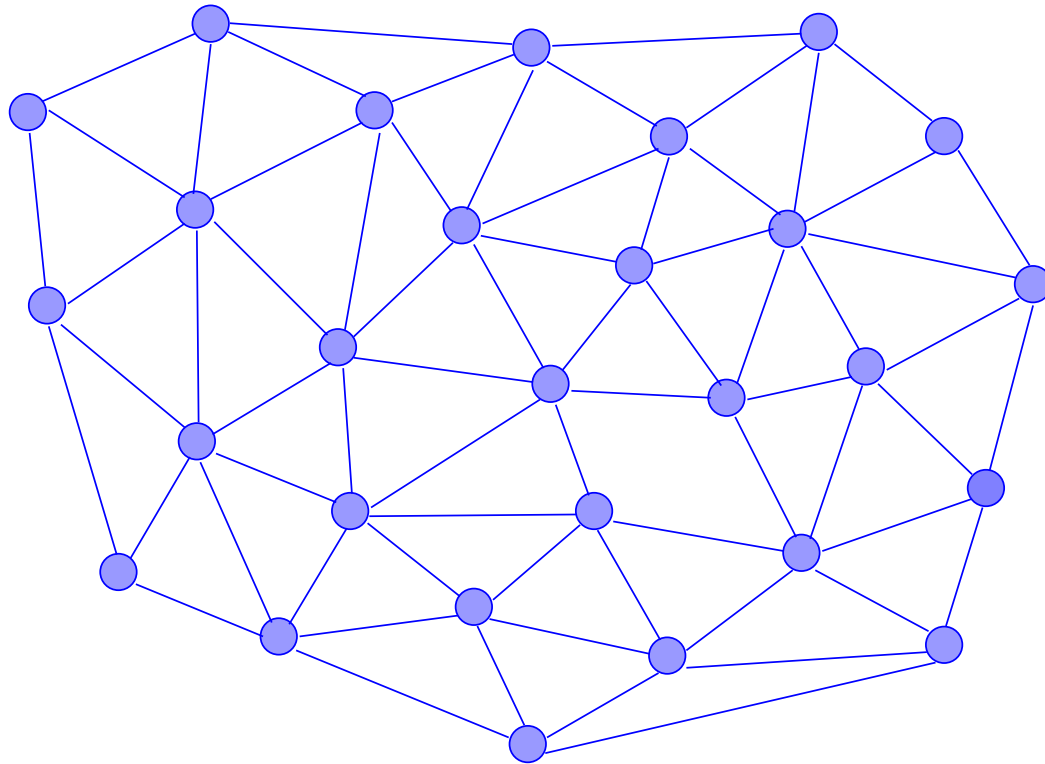
Buffer management:

Maintain buffers in routers and switches.

Web caching:

Maintain caches of web documents.

Large networks



TCP connection caching:

Maintain set of open connections.

TCP acknowledgement:

Acknowledge arrival of data packets.

Online routing:

Determine transmission paths.

Replication

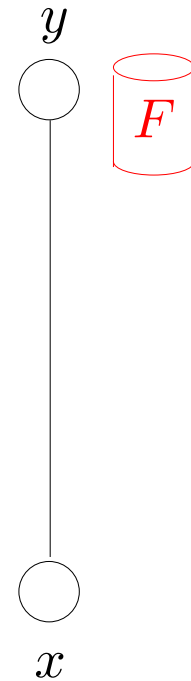
Request: x requests data item of F

F not available at x : $dist(x, y)$

Replication: $R \cdot dist(x, y)$

$R = \text{Size}(F)$

Goal: Minimize total cost



Replication

Algorithm R-Edge:

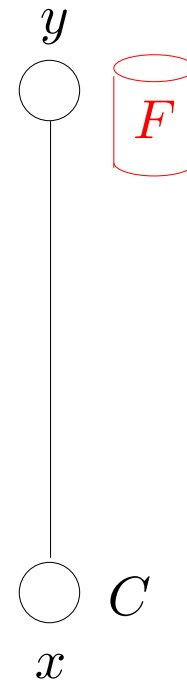
Maintain counter C at x , initially 0.

On each request, **increment C by 1.**

When $C = R$, **replicate** file.

Thm: R-Edge is **2-competitive.**

Thm: **No** det. algorithm can achieve competitive ratio **smaller than 2.**



Upper bound

Thm: R-Edge is 2-competitive.

Proof: Request sequence σ of m requests.

$d = \text{dist}(x, y)$

- $m < R$

$$R - \text{Edge}(\sigma) = md$$

$$\text{OPT}(\sigma) = md$$

- $m \geq R$

$$R - \text{Edge}(\sigma) = Rd + Rd = 2Rd$$

$$\text{OPT}(\sigma) = Rd$$

Lower bound

Thm: Any deterministic online alg. A has competitive ratio of **at least 2**.

Proof: When A replicates to x , adversary stops request sequence.

Suppose m requests are issued.

$$A(\sigma) = md + Rd$$

$$\text{OPT}(\sigma) = \min\{m, R\}d$$

Replication

Tree with root s ; initially F only at s

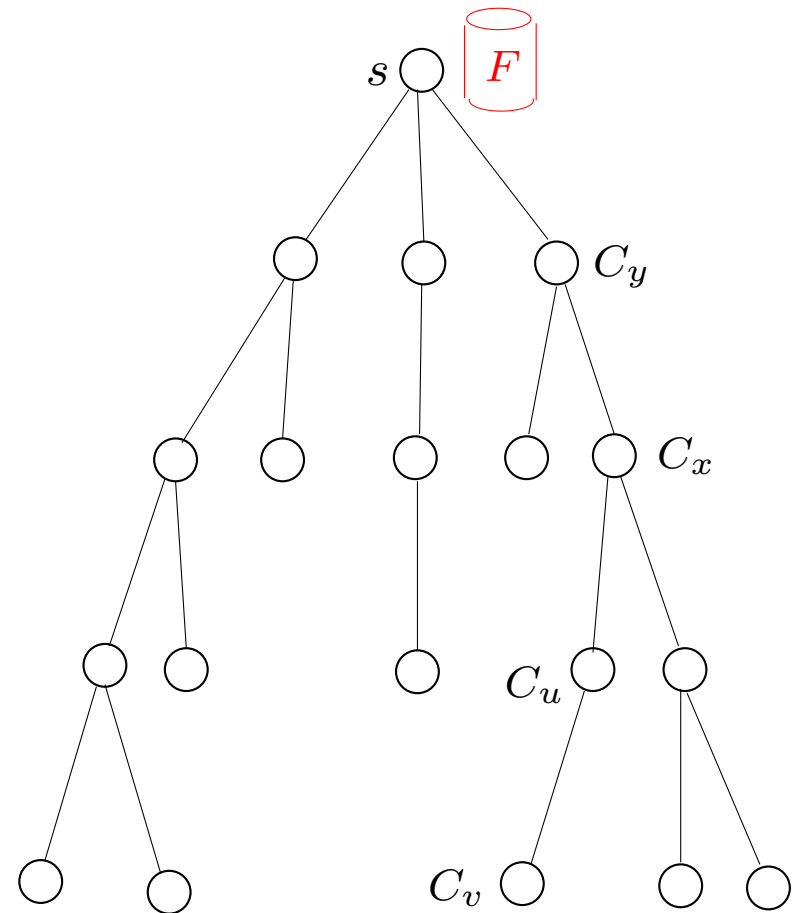
Algorithm R-Tree:

For any $v \in T$ maintain counter C_v ; initially 0.

On each request at a node v , increment all counters on **path from v to closest node w holding F** .

Whenever $C_v = R$, **replicate F** to v and all nodes on **path from v to closest node with F** .

Thm: R-Tree is **2-competitive**.



Replication

Tree with root s ; initially F only at s

Algorithm R-Tree:

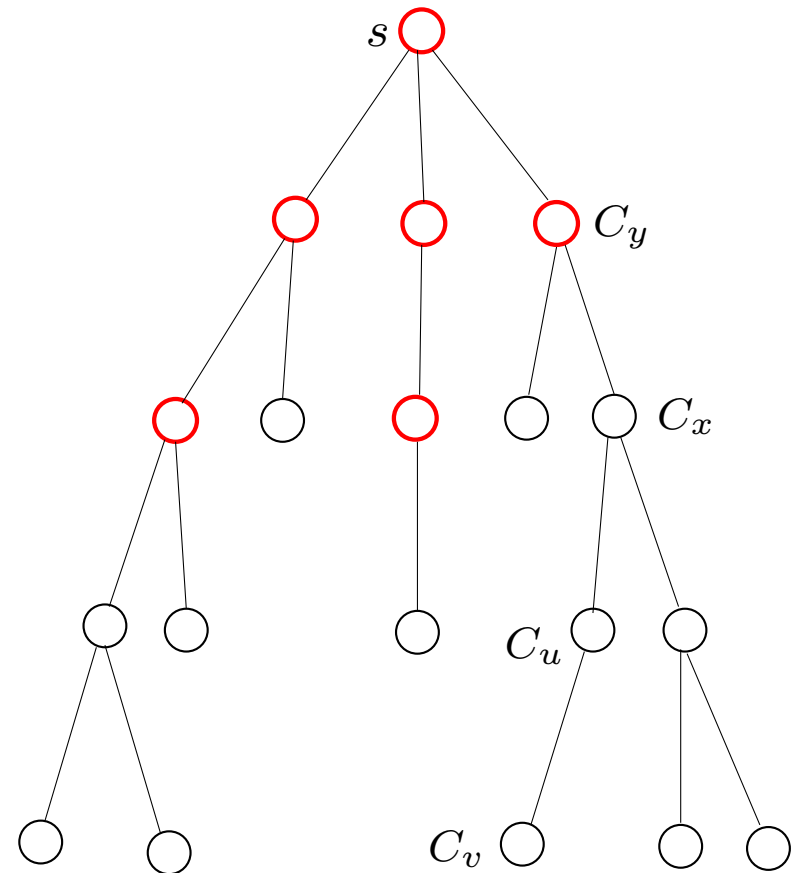
For any $v \in T$ maintain counter C_v ; initially 0.

On each request at a node v , increment all counters on **path from v to closest node w** holding F .

Whenever $C_v = R$, **replicate F** to v and all nodes on **path from v to closest node with F** .

Thm: R-Tree is **2-competitive**.

Black, Sleator 1994



Analysis

Nodes storing F form **one connected component**.

Partition cost of R-Tree, OPT into parts **corresponding** to the **edges** of T .

Edge e incurs cost for an access if edge is on access path.

Edge e incurs cost for replication across it.

Extended notion of “request”. At any time a request occurs at v if there is a **request at v or its decendents**.

For any node R-Tree and OPT experience the same requests.

For any edge **R-Tree works like R-Edge**.

Randomization

Randomized algorithm A

$A(\sigma)$ is **random variable** for any σ .

Competitive ratio defined with respect to an **adversary**.

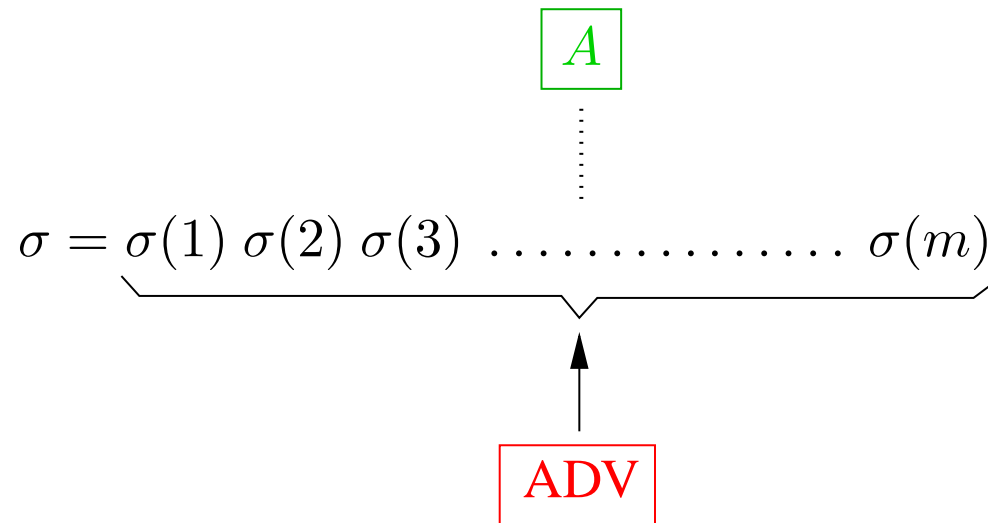
- It **constructs** σ .
- It also **serves** σ .

Adversary knows the description of A .

What does **adversary know about random choices made by A ?**

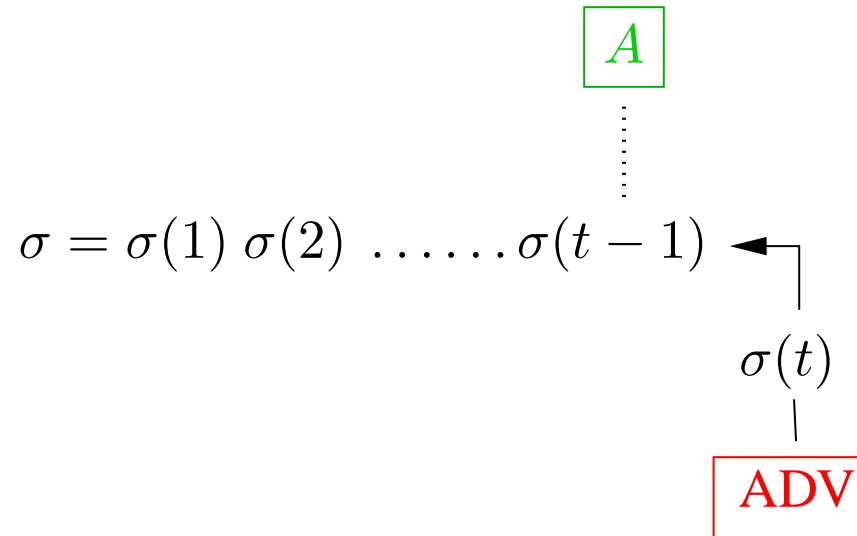
Oblivious adversary

Does not see the outcome of random choices made by A .



Adaptive adversaries

Sees A 's random choices on previous requests when generating next request.



- 1.) Adaptive online adversary serves each request **online**.
- 2.) Adaptive offline adversary serves σ **offline**.

Three types of adversaries

Oblivious adversary:

Does not see A 's random choices; serves σ offline.

A is c -competitive if $\exists b$ such that for all σ $\mathbf{E}[A(\sigma)] \leq c \cdot OPT(\sigma) + b.$

Adaptive online adversary:

Sees A 's random choices; serves σ online.

A is c -competitive if $\exists b$ such that for all σ $\mathbf{E}[A(\sigma)] \leq c \cdot \mathbf{E}[ADV(\sigma)] + b.$

Adaptive offline adversary:

See A 's random choices; serves σ offline.

A is c -competitive if $\exists b$ such that for all σ $\mathbf{E}[A(\sigma)] \leq c \cdot \mathbf{E}[OPT(\sigma)] + b.$

Relating the adversaries

Thm: If \exists rand. alg. that is c -competitive against any adaptive offline adv.
 $\Rightarrow \exists c$ -competitive deterministic algorithm.

Thm: If

- A c -competitive against any adaptive online adv.
 - $\exists d$ -competitive algorithm against any oblivious adv.
- $\Rightarrow A$ is $(c \cdot d)$ -competitive against any adaptive offline adv.

Cor: A c -competitive against any adaptive online adv.
 $\Rightarrow \exists c^2$ -competitive deterministic algorithm.

Randomized replication

$$\rho = (R + 1)/R \quad R = \text{Size}(F)$$

$$\alpha = \frac{\rho - 1}{\rho^R - 1}$$

Algorithm Rand-Edge:

Choose random number I .

$I = i$ with prob. $p_i = \alpha \rho^{i-1}$, $i = 1, \dots, R$.

Maintain counter C at x , initially 0.

On each request, increment C by 1.

When $C = I$, replicate file.

Thm: Rand-Edge has competitive ratio of

$\frac{e}{e-1} \approx 1.58$. against any oblivious adv.



Analysis

Proof: Request sequence of m requests, $1 \leq m \leq R$. $d = \text{dist}(x, y)$

$$\begin{aligned} \mathbb{E}[\text{Rand-Edge}(\sigma)] &= d \left(\sum_{i=1}^m (i + R) p_i + \sum_{i=m+1}^R m p_i \right) \\ &= \frac{\rho^R}{\rho^R - 1} \cdot m \cdot d \end{aligned}$$

Thm: On trees corresp. alg. is $\frac{e}{e-1}$ -competitive against any oblivious adv.

Lower bound

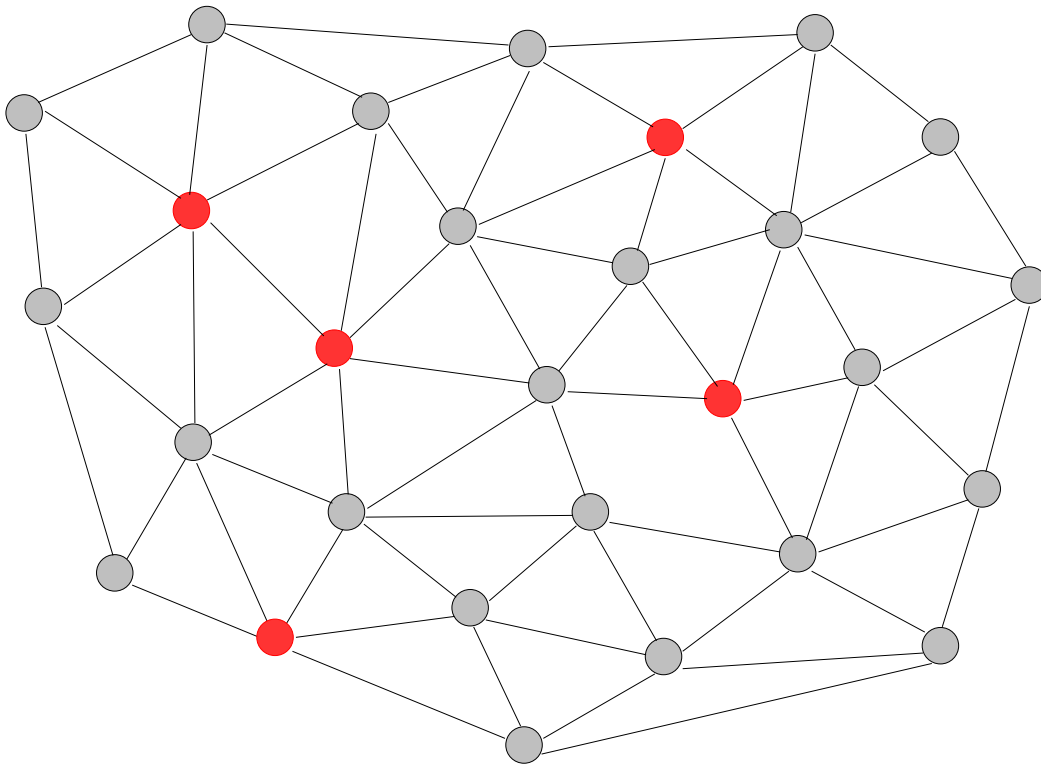
Thm: No rand. alg. A can achieve competitive ratio smaller than $\frac{e}{e-1}$.

Proof: q_i = prob. A replicates after exactly i requests

- Case 1: $\exists m$ with $\sum_{i=1}^m q_i \geq \sum_{i=1}^m p_i$
 $\sigma = m$ requests
- Case 2: $\sum_{i=1}^m q_i < \sum_{i=1}^m p_i$ for all $m = 1, \dots, R$
 $\sigma = 2R$ requests

$$E[A(\sigma)] \geq E[\text{Rand-Edge}(\sigma)]$$

Replication, migration



Replication, migration:

$$\text{Size}(F) \text{dist}(x, y)$$

Deletion of replica: 0

Read request at x :

$$\text{dist}(x, y) \quad y \text{ closest node with replica}$$

Write request at x :

$$\sum_{y \in Y} \text{dist}(x, y)$$

Y nodes with replica

Algorithm FA

Arbitrary network.

- Partition σ into **phases** containing $R = \text{Size}(F)$ write requests.
- In each phase, maintain list L of read requests.
On a read request at v , add to L .
If $|L| > R$, determine smallest **k -neighborhood**, $k = 2^i$, with **R reads**.
If no replica within **$4k$ of v** , **replicate** and remove the R requests from L .
- At the end of the phase, let v_1, \dots, v_R be locations of write requests.
Replicate to **w minimizing** $\sum_{i=1}^R \text{dist}(w, v_i)$ and delete all other copies.

Thm: FA is $O(\log n)$ -competitive.

Awerbuch, Bartal, Fiat Inf. & Comp. 2003