Competitive Online Algorithms

Susanne Albers
MPI Saarbrücken
Germany

Overview

Lecture 1:

General concepts
Randomization
} Paging problem

k-server problem

Lecture 2:

Selforganizing data structures
→ List update problem

Data compression

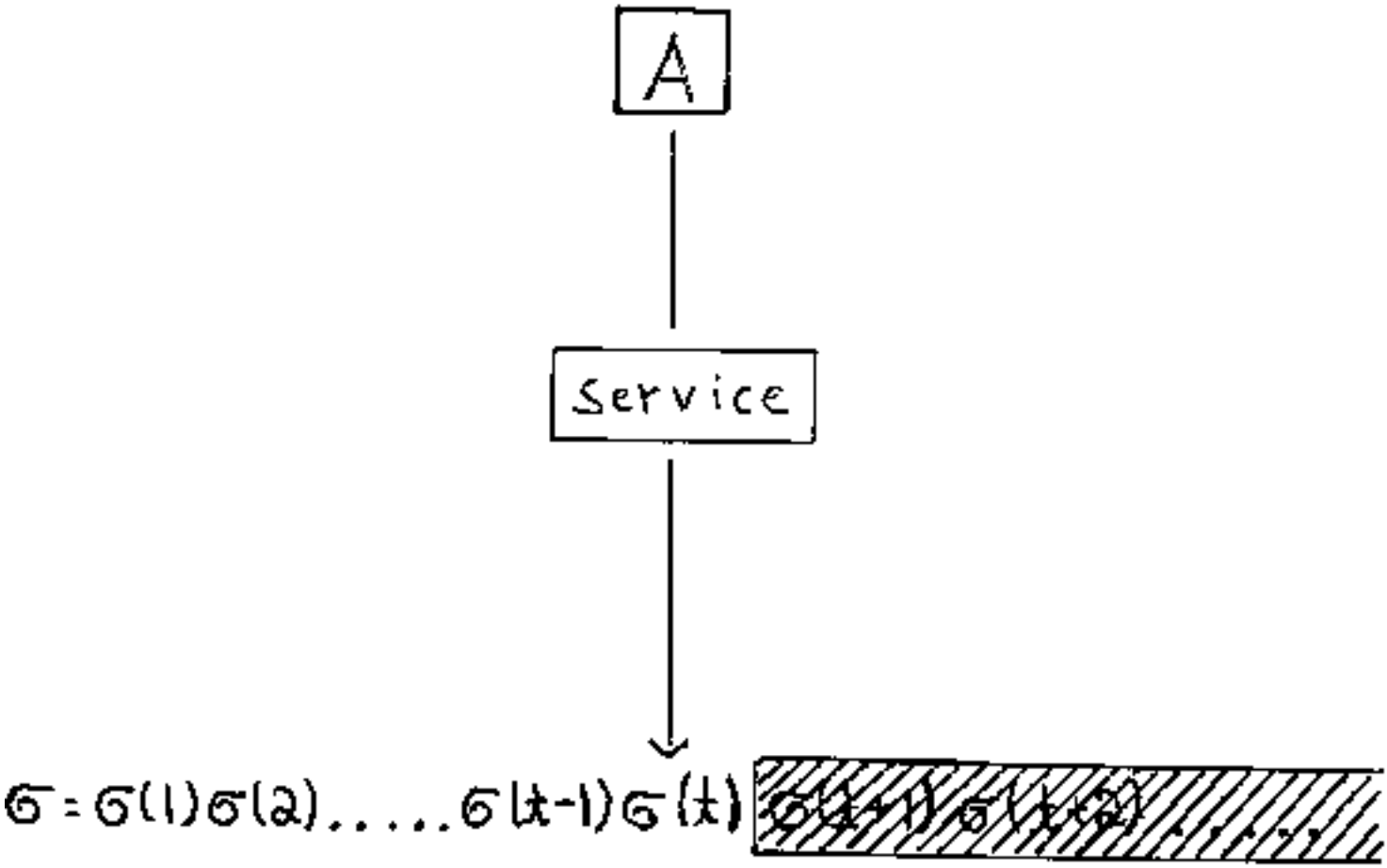
Lecture 3:

Distributed computing

Scheduling, load balancing

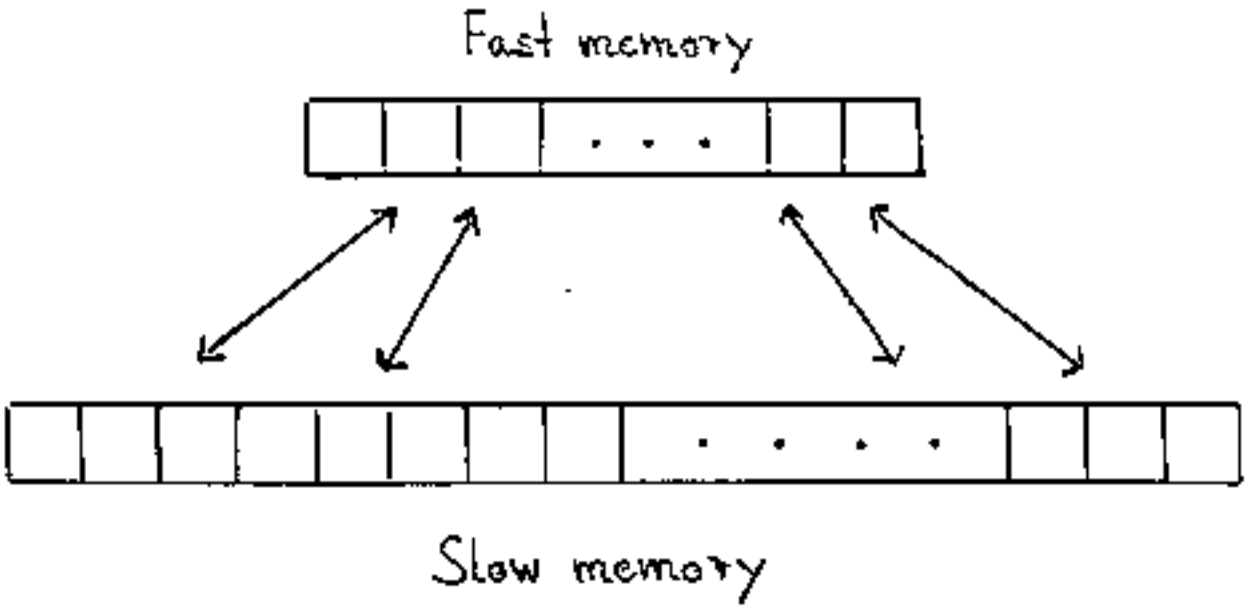
Navigation, exploration

Online algorithms



Goal: Minimize total cost incurred in serving σ.

The paging problem

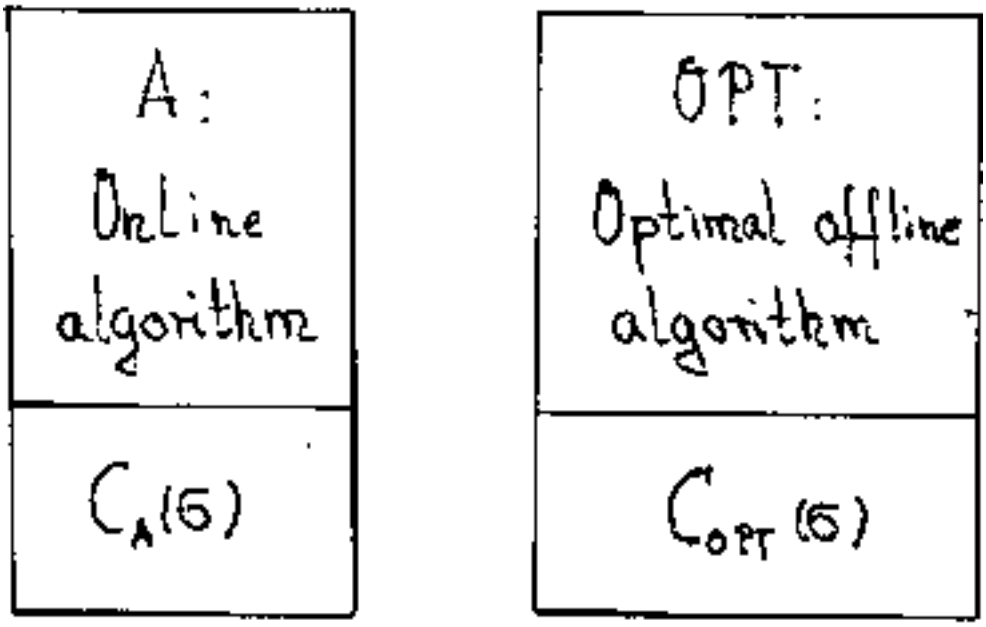


Request: access to a page in the memory system

Page fault: requested page is not in fast memory

Goal: minimize the number of page faults

Competitive analysis



A is c-competitive if $\exists a$:

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + a$$

for all request sequences σ.

Deterministic paging algorithms

LRU (Least Recently Used)

Evict page that was requested least recently.

FIFO (First In First Out)

Evict page that has been in fast memory longest

LFU (Least Frequently Used)

Evict page that has been requested least freque

MIN

Evict page whose next request occurs furtest in the future.

(Belady 66)

Optimal offline algorithm

Theorem: LRU, FIFO are k-competitive.

k = # pages that can be stored in fast memory

Theorem: No deterministic online paging algorithm can be better than k-competitive.

Sleator, Tarjan 85

Randomized online algorithms

Algorithm A

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

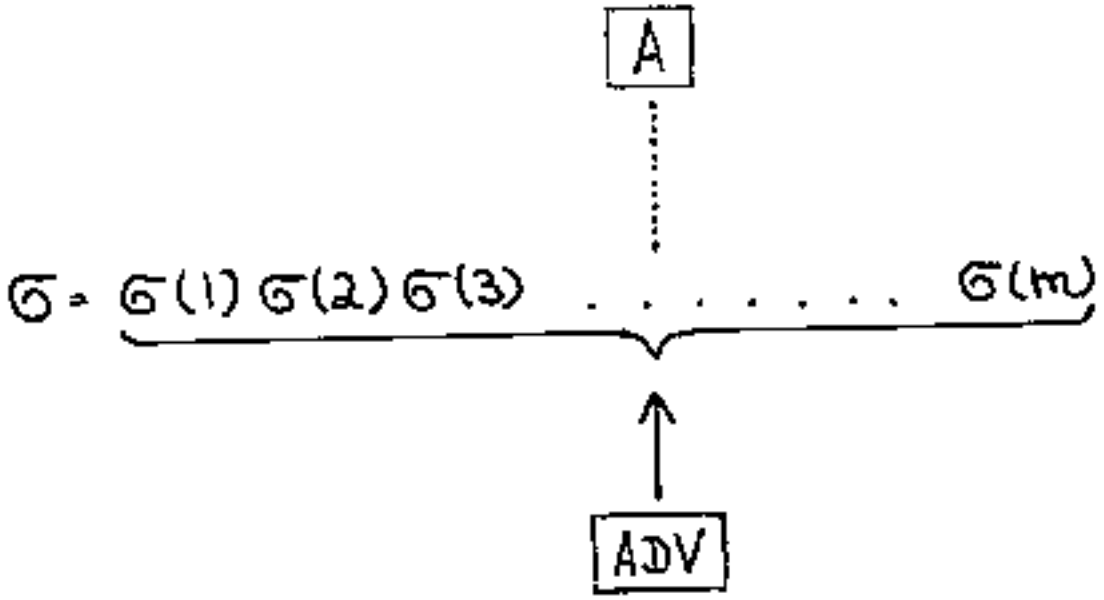
Competitiveness is defined with respect to an adversary

- It constructs σ.
- It also serves σ.

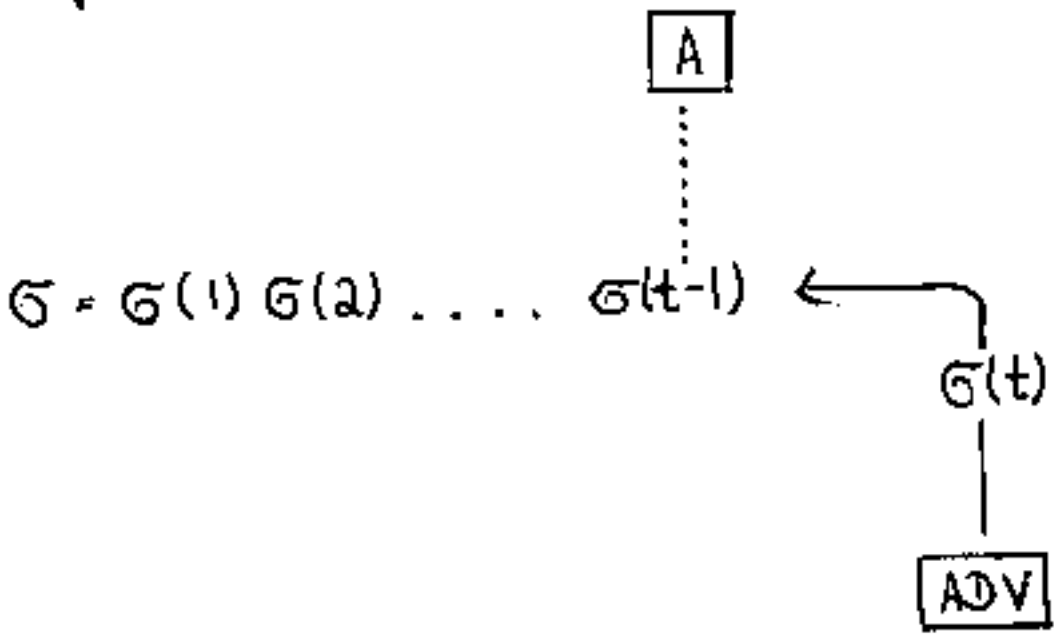
Adversary knows the description of A.

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

Oblivious adversary



Adaptive adversary



1) Adversary serves each request online.

2) Adversary serves σ offline.

Relating the adversaries

Theorem: If there exists randomized algorithm that is c-competitive against any adaptive offline adversary
$\Longrightarrow \exists$ c-competitive deterministic algorithm.

Theorem: If

- A c-competitive against any adaptive online adversary
- $\exists$ d-competitive algorithm against any oblivious adversary

$\Longrightarrow$ A is $(c \cdot d)$-competitive against any adaptive offline adversary.

Corollary: A c-competitive against any adaptive online adversary
$\Longrightarrow \exists$ c^2-competitive deterministic algorithm.

Three types of adversaries

Oblivious adversary: Does not see A's random choices

serves σ offline.

A is c-competitive if $\exists a$ s.t. for all σ

$$E[C_A(\sigma)] \leq c \cdot C_{OPT}(\sigma) + a$$

Adaptive adversaries:

A ADV

$E[C_A]$: Expected cost incured by A in serving request sequence generated by ADV.

$E[C_{ADV}]$: ——— " ——— ADV ——— " ———

——— " ——— ADV

Adaptive online adv.: Sees A's random choices, serves σ online

A is c-competitive if $\exists a$ s.t. for all ADV

$$E[C_A] \leq c \cdot E[C_{ADV}] + a$$

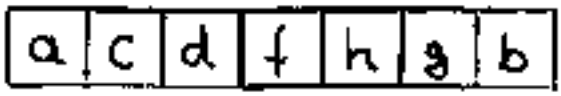
Adaptive offline adv.: Sees A's random choices, serves σ offline.

---

oblivious ≤ adaptive online ≤ adaptive offline

Ben-David, Borodin, Karp, Tardos, Wigderson 90

Paging against oblivious adversaries



σ = ...i a b b a. d. e

MARKING:

Phases:

- When a page is requested it is marked.
- On a fault, choose a page uniformly at random from among the unmarked pages and evict it.

Phase is over when all pages are marked.

Theorem: MARKING is $2H_k$-competitive.

$H_k = \sum_{i=1}^{k} 1/i \approx \ln k$ k-th Harmonic number

Theorem: $c \geq H_k$

Fiat, Karp, Luby, McGeoch, Sleator, Young 91

Proof technique for lower bounds against oblivious adversaries

Yao's minimax principle

$\mathcal{P}$: Probability distribution for choosing σ.

$\mathcal{C}_A^{\mathcal{P}}$: Competitiveness of deterministic online algorithm A under $\mathcal{P}$

Infimum of all c such that

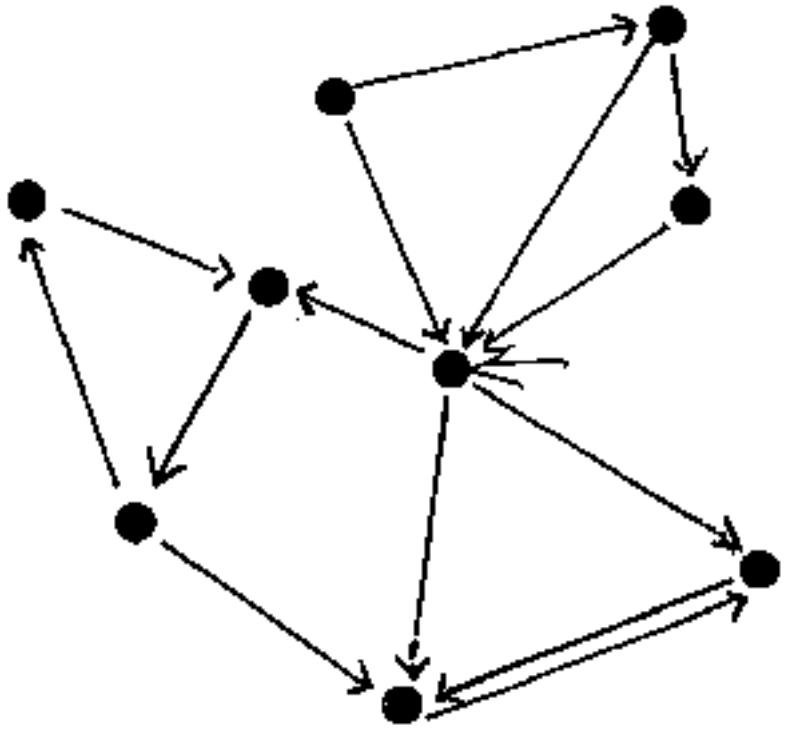
$$E[C_A(\sigma)] \leq c \cdot E[C_{OPT}(\sigma)] + a$$

for σ generated according to $\mathcal{P}$

$$\inf_R C_R = \sup_{\mathcal{P}} \inf_A \mathcal{C}_A^{\mathcal{P}}$$

Construct $\mathcal{P}$, develop lower bound on $\inf_A \mathcal{C}_A^{\mathcal{P}}$

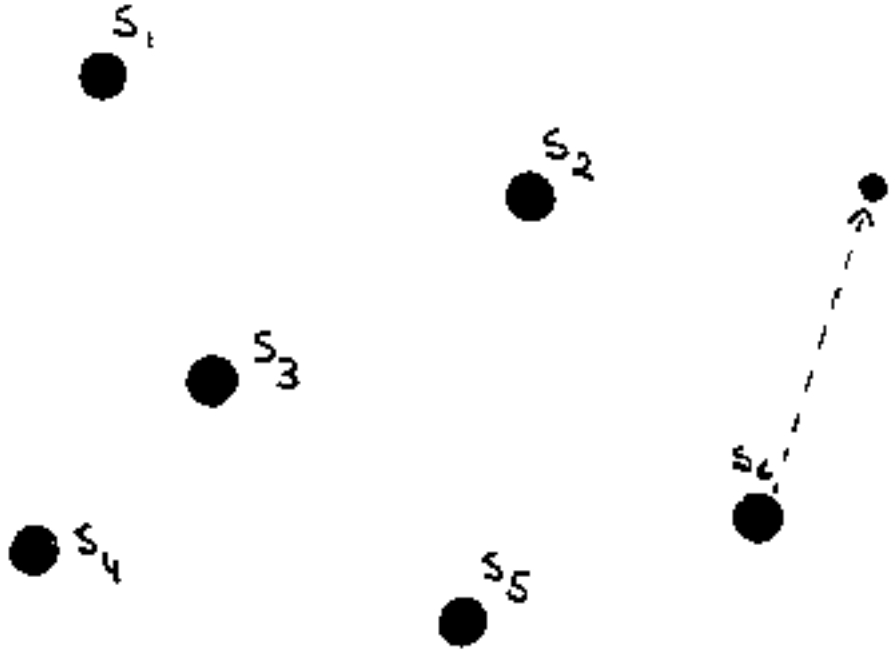
Paging with access graphs



There are access graphs on which LRU is better than FIFO.

Irani 91

The k-server problem



Metric space S k mobile servers

Request : point $x \in S$

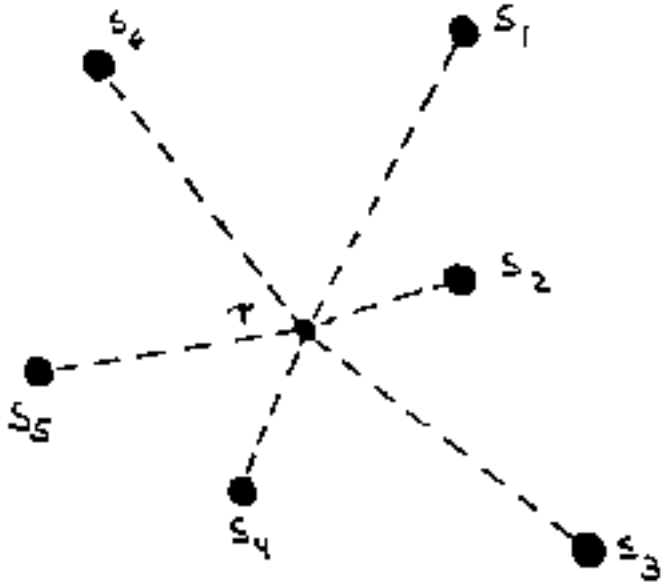
Moving server from x to y costs dist (x, y)

Goal: minimize total distance traveled by the servers.

Special cases : - Paging

- Font caching in printers
- Vehicle routing problem (repair service)

The HARMONIC algorithm



Move server S_i with probability

$$\frac{1/d(S_i, r)}{\sum_{j=1}^{k} 1/d(S_j, r)}$$

$$\frac{k(k+1)}{2} \leq C \leq 2^k$$

Grove 91

Results

Deterministic algorithms

$$k \leq c \leq 2k-1$$

Manasse, McGeoch, Sleator 90

Koutsoupias, Papadimitriou 95

* The greedy algorithm is not competitive.

Randomized algorithms against oblivious adversaries

$$\Omega\left(\sqrt{\frac{\log k}{\log\log k}}\right) = c$$

Blum, Karloff, Rabani, Saks 92

Open problems

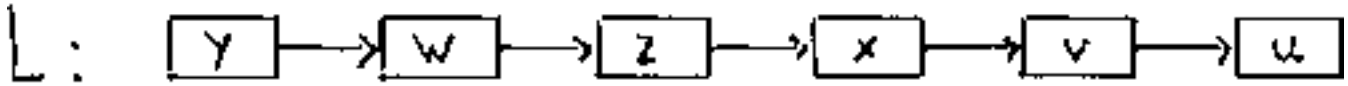
Develop randomized algorithm with $c < k$

Tight analysis for HARMONIC.

The list update problem

The list update problem

Unsorted list



$\sigma = \sigma(1)\,\sigma(2)\,\sigma(3) \ldots \sigma(m)$

- access (x)
- insert (x)
- delete (x)

Standard cost model

Access to i-th item costs i.

Deletion of i-th item costs i.

Insertion of new item costs n+1.

Paid exchange costs 1.

Applications

Data structures

- small data sets
- computational geometry

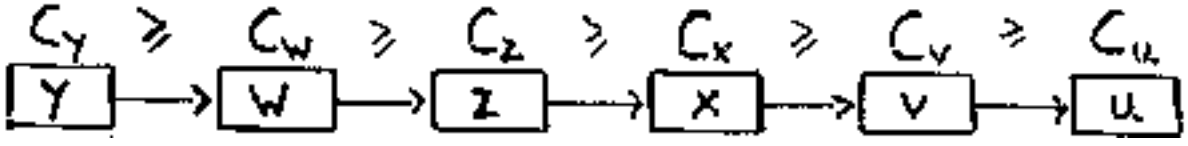
Data compression

Deterministic online algorithms

MOVE-TO-FRONT: Move requested item to the front of the list.

TRANSPOSE: Move requested item one position ahead.

FREQUENCY-COUNT: Maintain counter c_x for each item x. Whenever x is accessed, increase its counter by 1.

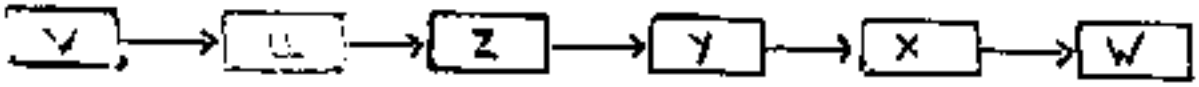


Theorem: MOVE-TO-FRONT is 2-competitive.

Sleator, Tarjan 85

Theorem: TRANSPOSE and FREQUENCY-COUNT are not c-competitive for any constant c.

A more recent deterministic algorithm

L: 

TIMESTAMP(0): Move requested item x in front of the first item in the list that was requested at most once since the last request to x.

σ = ... x u u z v v x
↑

Theorem: TIMESTAMP(0) is 2-competitive.
Albers 95

Lower bounds

Theorem: No deterministic online algorithm for List update can be better than 2-competiti

Proof technique for upper bounds

$$\sigma = \sigma(1)\,\sigma(2)\,\ldots\ldots\,\sigma(m)$$

$C_A(\sigma)$ $\qquad$ $C_{OPT}(\sigma)$

Potential function Φ

$\Phi(t) \geq 0 \qquad \Phi(0) = 0$

$$C_A(\sigma(t)) + \underbrace{\Phi(t) - \Phi(t-1)}_{\Delta\Phi} \leq c \cdot C_{OPT}(\sigma(t)) \quad \|$$

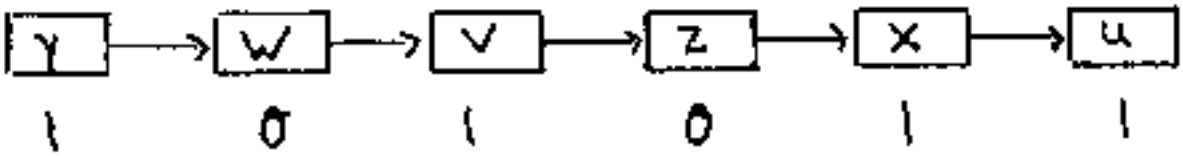
$$\Longrightarrow \sum_{t=1}^{m} C_A(\sigma(t)) + \Phi(m) - \Phi(0) \leq c \cdot \sum_{t=1}^{m} C_{OPT}(\sigma(t))$$

$$\Longleftrightarrow \qquad C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) - \Phi(m)$$

A is c-competitive

$$E[\, C_A(\sigma(t)) + \Delta\Phi \,] \leq \begin{array}{l} c \cdot C_{OPT}(\sigma(t)) \\ c \cdot E[C_{ADV}(\sigma(t))] \end{array}$$

Randomized online algorithms against oblivious adversa



BIT: Maintain bit box for each item x.
Complement box whenever x is accessed.
Move x to the front if box changes to 1.

Theorem: BIT is 1.75-competitive.
Reingold, Westbrook, Sleator 91

Combination: With probability 4/5
serve σ using BIT;
With probability 1/5
serve σ using TIMESTAMP(0).

Theorem: COMBINATION is 1.6-competitive.
Albers, von Stengel, Werchner

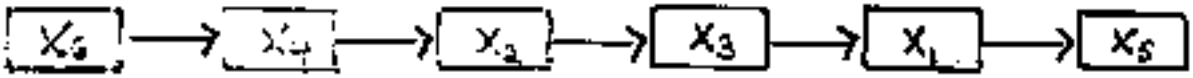
Data compression

Goal: Represent file with fewer bits.

Data compression

$\Sigma = \{ X_1, X_2, \ldots, X_n \}$ alphabet

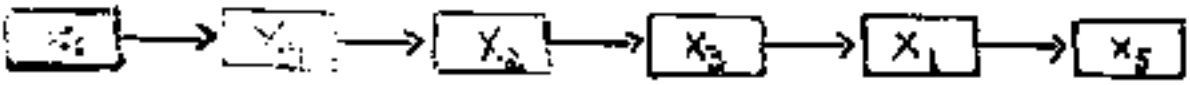
$S = X_1 X_1 X_2 X_1 X_5 X_3 X_4 X_2 X_1 X_2$



Encoder

$I = 5\,1\,4\,2\,6\,4\,5\,2$

Decoder



$S = X_1 X_1 X_2 X_1 X_5 X_3 X_4 X_2 X_1 X_2$

Bentley, Sleator, Tarjan, Wei 86

Variable length prefix codes

Code for integer j

$1 + 2\lfloor \log j \rfloor$ bits $\quad \underbrace{00\ldots0}_{\lfloor \log j \rfloor} 1 0010\ldots11$

$1 + \lfloor \log j \rfloor + 2\lfloor \log(1+\log j) \rfloor$ bits $\quad \underbrace{0\ldots010\ldots1}_{2\lfloor \log(1+\log j) \rfloor} 0010\ldots11$

Upper bounds on compression

S arbitrary

$|S| = m$ $\quad m_i = \#$ occurrences of X_i in S

"Empirical entropy":

$$H(S) = \sum_{i=1}^{n} \frac{m_i}{m} \log\left(\frac{m}{m_i}\right)$$

$A_{MTF}(S)$ = average number of bits needed to encode one symbol in S.

Theorem: $\forall S$

$$A_{MTF}(S) \leq 1 + H(S) + 2 \log(1 + H(S))$$

Bentley, Sleator, Tarjan, Wei 86

$\Sigma = \{X_1, \ldots, X_n\}$

S generated by discrete memoryless source

$$\vec{p} = (p_1, \ldots, p_n)$$

Each symbol is X_i with probability p_i, $1 \leq i \leq n$

$B_A(\vec{p})$ = expected number of bits to encode one symbol using A.

Entropy: $H(\vec{p}) = \sum_{i=1}^{n} p_i \log(1/p_i)$

Theorem: $\forall \vec{p}$

$$B_{HTF}(\vec{p}) \leq 1 + H(\vec{p}) + 2\log(1 + H(\vec{p}))$$

Theorem: $\forall \vec{p}$

$$B_{TS}(\vec{p}) \leq 1 + \bar{H}(\vec{p}) + 2\log(1 + \bar{H}(\vec{p}))$$

$$\bar{H}(\vec{p}) = H(p) + \log\left(1 - \sum_{i<j} p_i p_j \left(\frac{p_i - p_j}{p_i + p_j}\right)^2\right)$$

Albers, Mitzenmacher 96

Extensions

* MTF encoding with secondary lists.
  Grinberg, Rajagopalan, Venkatesan, Wei 95

* Compression algorithm by Burrows and Wheeler

$\Sigma = \{X_1, \ldots, X_n\}$

S generated by discrete memoryless source

$$\vec{p} = (p_1, \ldots, p_n)$$

Each symbol is X_i with probability p_i, $1 \le i \le n$

$B_A(\vec{p})$ = expected number of bits to encode one symbol using A.

Entropy: $H(\vec{p}) = \sum_{i=1}^{n} p_i \log(1/p_i)$

Theorem: $\forall \vec{p}$

$$B_{HTF}(\vec{p}) \le 1 + H(\vec{p}) + 2\log(1 + H(\vec{p}))$$

Theorem: $\forall \vec{p}$

$$B_{TS}(\vec{p}) \le 1 + \bar{H}(\vec{p}) + 2\log(1 + \bar{H}(\vec{p}))$$

$$\bar{H}(\vec{p}) = H(p) + \log\left(1 - \sum_{i<j} p_i p_j \left(\frac{p_i - p_j}{p_i + p_j}\right)^2\right)$$

Albers, Mitzenmacher 96

Implementation

Byte-based compression

Every byte (ASCII character) is a symbol.

Compressed file: 70-80%

Word-based compression

Every word is a symbol.

Compressed file: 20-30%

Experimental results

Byte-based compression

File	TS(0)		MTF		Original	
	Bytes	% Orig.	Bytes	% Orig.	Bytes	% Orig.
bib	99121	89.09	106478	95.70	111261	100.00
book1	581758	75.67	644423	83.83	768771	100.00
book2	473734	77.55	515257	84.35	610856	100.00
geo	92770	90.60	107437	104.92	102400	100.00
news	310003	82.21	333737	88.50	377109	100.00
obj1	18210	84.68	19366	90.06	21504	100.00
obj2	229284	92.90	250994	101.69	246814	100.00
paper1	42719	80.36	46143	86.80	53161	100.00
paper2	63654	77.44	69441	84.48	82199	100.00
pic	113001	22.02	119168	23.22	513216	100.00
progc	33123	83.62	35156	88.75	39611	100.00
progl	52490	73.26	55183	77.02	71646	100.00
progp	37266	75.47	40044	81.10	49379	100.00
trans	79258	84.59	82058	87.58	93695	100.00

Word-based compression

File	TS(0)		MTF		Original	
	Bytes	% Orig.	Bytes	% Orig.	Bytes	% Orig.
bib	34117	30.66	35407	31.82	111261	100.00
book1	286691	37.29	296172	38.53	768771	100.00
book2	260602	42.66	267257	43.75	610856	100.00
news	116782	30.97	117876	31.26	377109	100.00
paper1	15195	28.58	15429	29.02	53161	100.00
paper2	24862	30.25	25577	31.12	82199	100.00
progc	10160	25.65	10338	26.10	39611	100.00
progl	14931	20.84	14754	20.59	71646	100.00
progp	7395	14.98	7409	15.00	49379	100.00

Extensions

* MTF encoding with secondary lists.
  Grinberg, Rajagopalan, Venkatesan, Wei 95

* Compression algorithm by Burrows and Wheeler

Compression algorithm by Burrows and Wheeler

S = a b r a . c a

* Compute all cyclic rotations of S and sort them lexicographically

	a	a	b	r	a	c
→	a	b	r	a	c	a
	a	c	a	a	b	r
	b	r	a	c	a	a
	c	a	a	b	r	a
	r	a	c	a	a	b

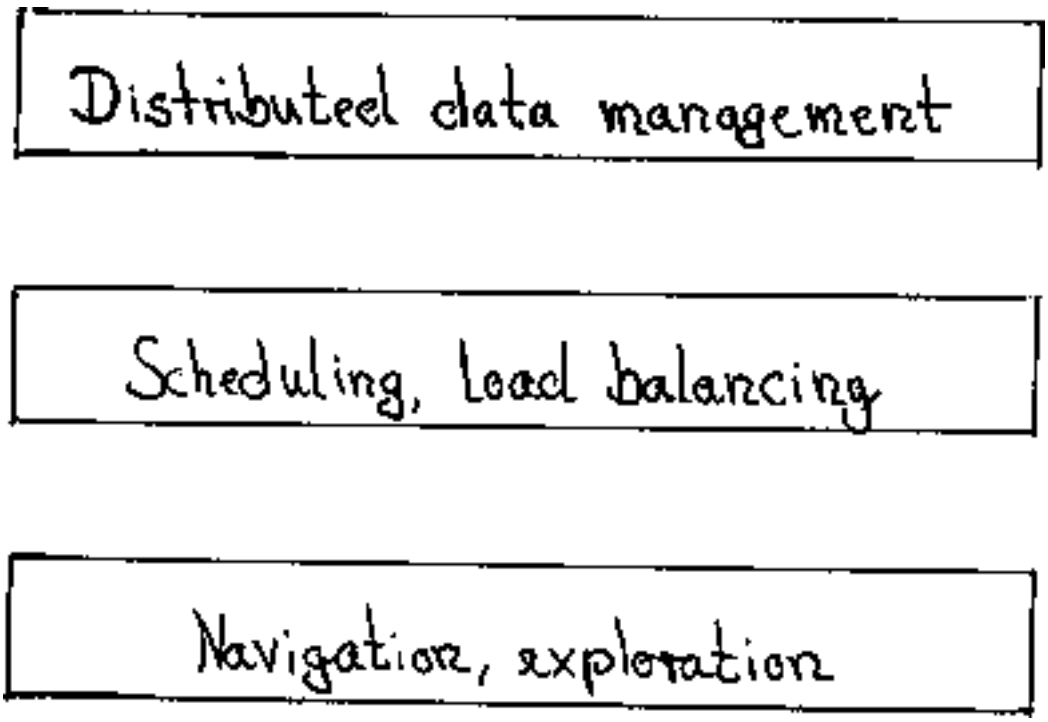
* S' : Extract last character from each rotation

* Encode S' using MTF encoding

Experimental results

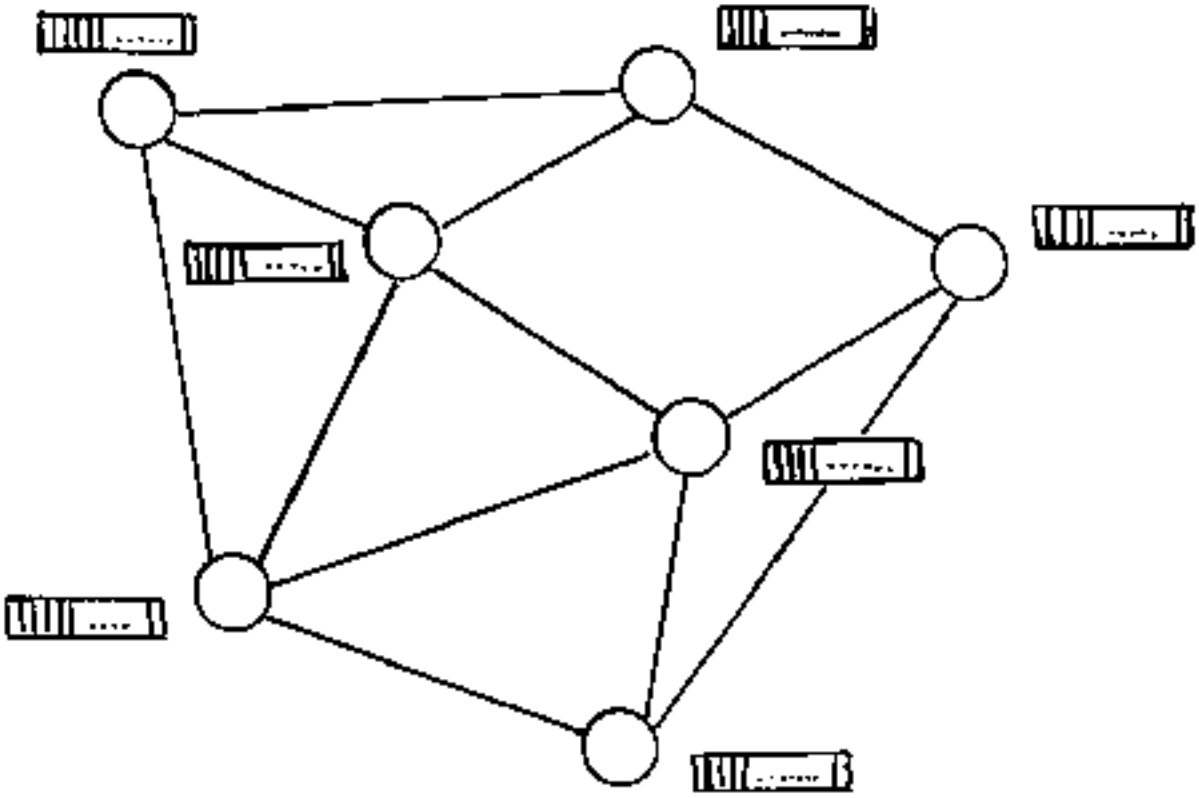
Algorithm achieves better compression than UNIX utilities.

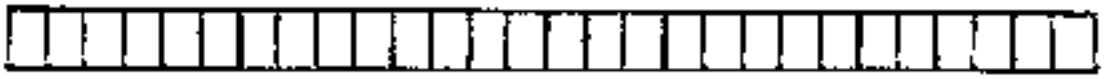
Compressed file : 20 - 25 %


Distributeel data management
Scheduling, load balancing
Navigation, exploration

Multiprocessor systems

Network





Virtual global memory

Migration: one copy of each page

Replication: multiple copies of each page

Migration – general graphs

Deterministic algorithms:

7-competitive, 4.1-competitive

Randomized algorithms:

COUNTER: Maintain counter C, $C \in \{0, \ldots, k$

On each request, decrement C by 1

If $C = 0$, move page to requesting node

reset C to k

Westbrook 92

Theorem: COUNTER is c-competitive

$$c = \max \left\{ 2 + \frac{2d}{k}, 1 + \frac{k+1}{2d} \right\}$$

Optimal choice for k:

$$c \longrightarrow 1 + \phi \approx 2.62$$

Replication

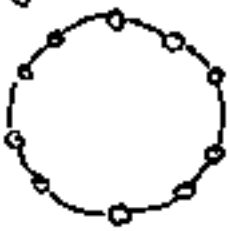
Specific topologies

Trees

Uniform networks

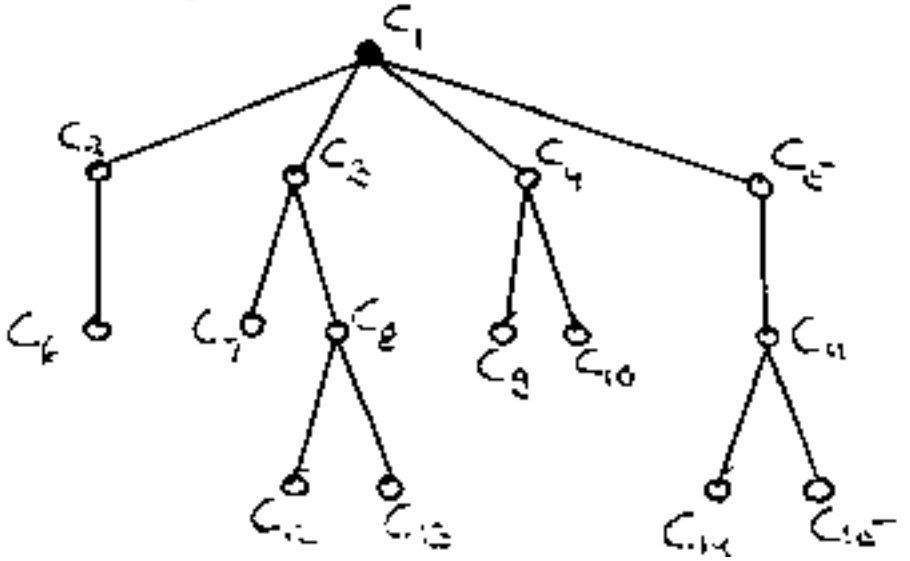
$c = \begin{cases} 2 & \text{deterministic} \quad \text{Black, Sleator 89} \\ \frac{e}{e-1} \approx 1.58 & \text{randomized} \quad \text{Albers, Koga 94} \end{cases}$

Rings



$c = 4$ Albers, Koga 94

COUNTER algorithm for trees



Request at v: Increment counters on path from v to closest node with the page

If a counter becomes C, replicate page to corresponding node.

Initial counter values:

1 2-competitive

$1 \le i \le d$ with probability $\alpha(\frac{\delta+1}{\delta})^{i-1}$ 1.58-competitive

$\alpha = \frac{\delta-1}{\delta^{d+1}-1}$ $\delta = \frac{d+1}{d}$

Replication - general graphs

COINFLIP:

If requesting node v does not have the page, with probability $1/d$ replicate page to v from the closest node holding page.

Theorem: COINFLIP is $\Theta(\log n)$ - competitive

n = # nodes in graph

Bartal, Fiat, Rabani

Combined migration / replication

Cost model:

* migration / replication : $d \cdot dist(u,v)$
* erasing page replica : 0
* read request at v : $dist(u,v)$
  u - closest node with the page
* write request at v : cost of communicati
  from v to all nodes with a page repli

COINFLIP algorithm:

Read request at v: If v does not have the page
then with probability $1/d$, replicate page to v

Write request at v: With probability $1/\sqrt{3}d$,
migrate page to v and erase all other replica

Theorem: COINFLIP is $\Theta(\log n)$-competitive

Scheduling, load balancing

Scheduling

* m machines

* sequence of jobs $\sigma = j_1, j_2, j_3, j_4, \ldots$

Minimize given objective function.

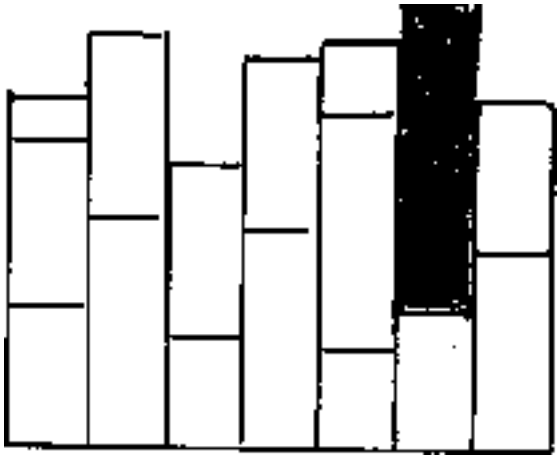
Identical machines

Job j_i has processing time p_i

Minimize completion time of last job, C_{max}

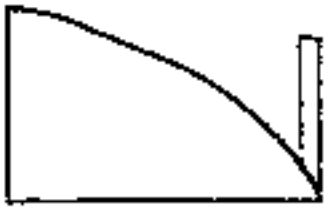
GREEDY: Always assign new job to the least loaded machine.

Theorem: GREEDY is $(2 - 1/m)$-competitive.



Improvements

GREEDY



New approach

Improved competitive ratios

1.986	Bartal, Fiat, Karloff, Vohra 92
1.945	Karger, Phillips, Torng 94

Algorithm:

h_i : height of i-th smallest machine

A_i : average height of the $i-1$ smallest machines

Schedule new job J_t on tallest machine k such that

$$h_k + p_t \leq \alpha A_k$$

Extensions

Identical machines, restricted assignment

Each job can be assigned to one of a subset of admissable machines.

Theorem: GREEDY is $\Theta(\log m)$-competitive.

Related machines

Machine i has speed s_i.

Processing time of J_t on machine i : p_t/s_i

Theorem: GREEDY is $\Theta(\log m)$-competitive.

Unrelated machines

$p_{t,i}$: processing time of J_t on machine i

Theorem: $\exists$ $\Theta(\log m)$-competitive algorithm

Load balancing

* m machines
* sequence of jobs $j_1, j_2, j_3, \ldots$
  each job j_k has weight $w(k)$
  unknown duration

* $l_i(t)$ = load of machine i at time t
  = sum of weights of jobs present on machine i at time t

Goal: Minimize maximum load that ever occurs.

Identical machines

Theorem: GREEDY is $(2-1/m)$-competitive.

Identical machines, restricted assignment

Theorem: GREEDY is $\Theta(m^{2/3})$-competitive.

Theorem: No algorithm can be better than $\Omega(\sqrt{m})$-competitive.

Azar, Broder, Karlin 92

Theorem: ROBIN-HOOD is $O(\sqrt{m})$-competitive

Azar, Kalyansundaram, Plotkin, Pruhs, Waarts 93

ROBIN-HOOD:

Maintain estimate L for OPT satisfying $L \leq OPT$

Machine i is rich if $l_i(t) \geq \sqrt{m}\, L$

poor otherwise

When new job j_k arrives

$$L = \max\left\{ L,\ w(k),\ \frac{1}{m}\left(w(k) + \sum_{i=1}^{m} l_i(t)\right)\right\}$$

If possible, assign j_k on a poor machine.

Otherwise assign it on rich machine that became rich most recently.

Navigation / exploration

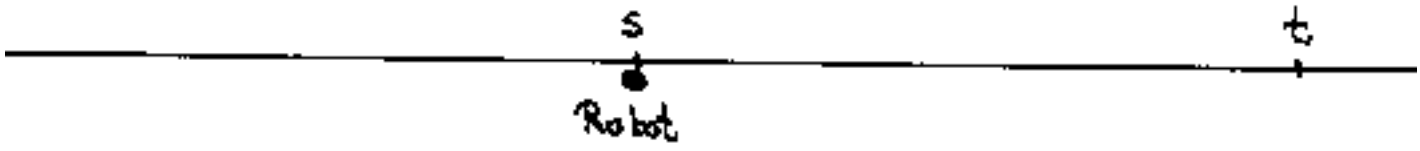
Unfamiliar terrain

Navigation: Starting at s, a robot has to find a path to a target t.

Exploration: Robot has to construct a complete map of the environment.

Goal: Minimize distance traveled by robot

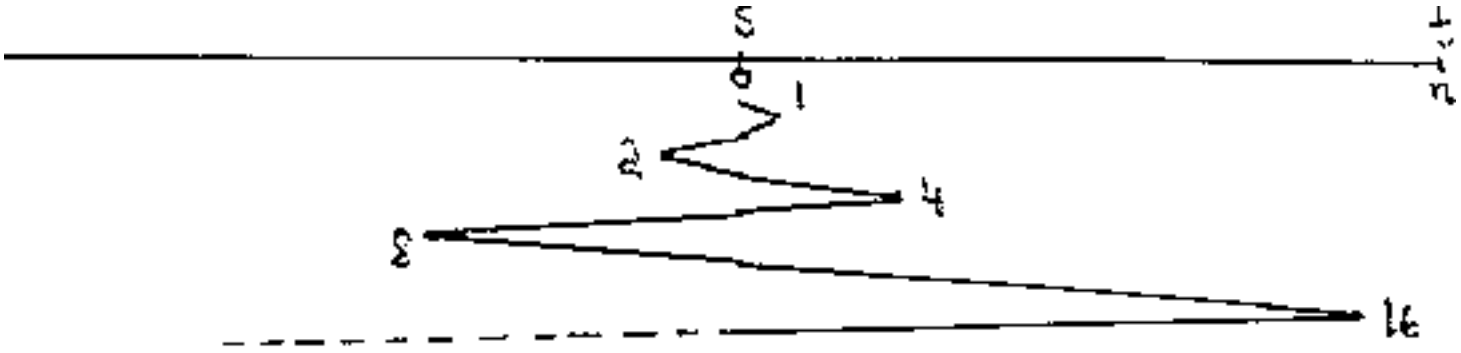
A simple navigation problem



Tactile robot

Robot strategy is c-competitive if

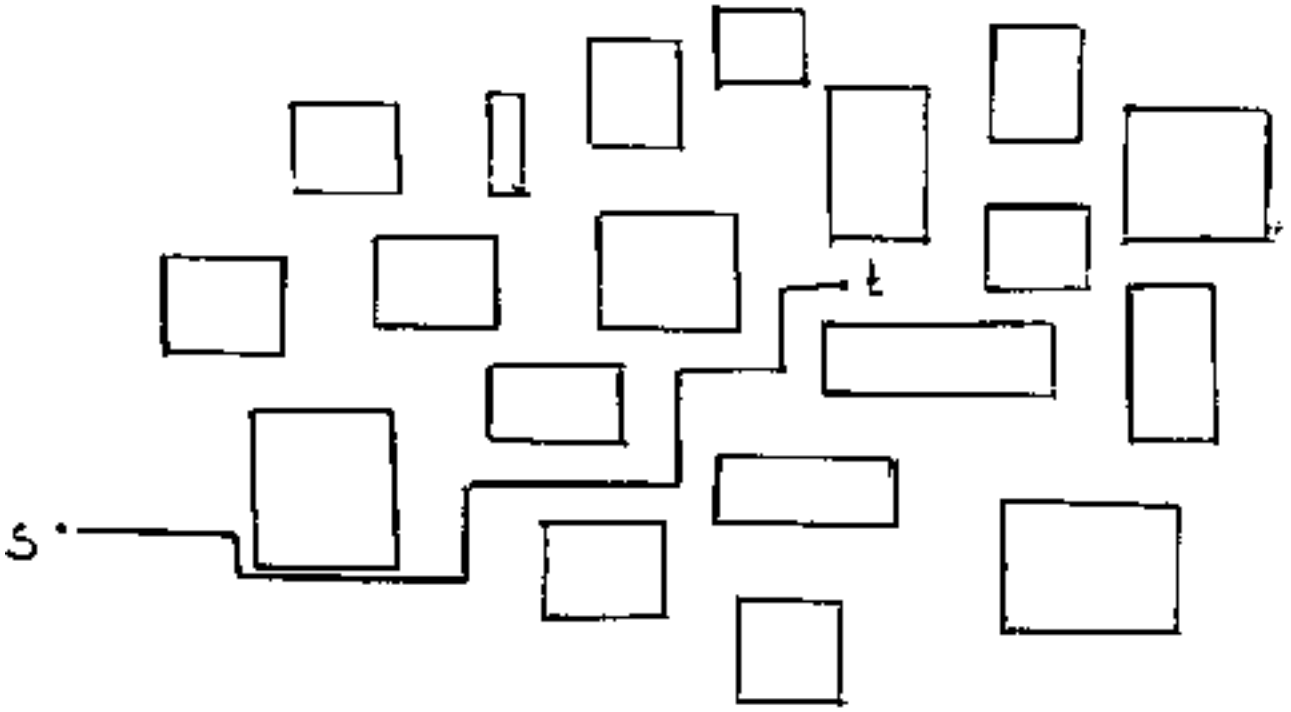
$$L_R \leq c \, L_{OPT}$$



$$L_R = 2\sum_{i=1}^{\lfloor \log_2 n \rfloor + 1} 2^i + n \quad = 9n$$

Baeza-Yates, Culberson, Rawlins

Scenes with obstacles



Robot knows current position and position of t.

does not know positions, extends of obstacles.

Tactile robot: learns about obstacles by "bumping" into it

Robot strategy is c-competitive if

$$L_R(S) \leq c \cdot L_{OPT}(S)$$

for all scenes S.

Obstacles: axis parallel rectangles

Deterministic algorithms

$$c = \Theta(\sqrt{n})$$

n = # obstacles

Papadimitriou, Yannakakis 90
Blum, Raghavan, Schieber 91

Randomized algorithms against oblivious adversaries

$$c = O(n^{4/5} \log n)$$

BBFKRS 96

Open problem:

Better bounds on c.

$$c = \Theta(\log n)?$$

Further applications

* Bin packing

* Graph problems
  - Coloring
  - Matching

* Finance
  - .
  - .
  - .
  - .