How much information about the future is needed ?

Stefan Dobrev[1], *Rastislav Královič*[2], Dana Pardubská[2]

Institute of Mathematics,
Slovak Academy of Sciences
`Stefan.Dobrev@savba.sk`

Department of Computer Science,
Comenius University, Bratislava, Slovakia
`{kralovic,pardubska}@dcs.fmph.uniba.sk`

Sofsem 2008, January 21, 2008

ski-rental problem

dilemma of a skier:

- rent equipment for 10 EUR per day
- buy equipment for $10c$ EUR

doesn't know how many days will be skiing

ski-rental problem

dilemma of a skier:

- rent equipment for 10 EUR per day
- buy equipment for 10c EUR

doesn't know how many days will be skiing

online algorithm $\mathcal{A}$

input $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$
output $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) = \langle y_1, y_2, \ldots, y_n \rangle$, where $y_i = f(x_1, \ldots, x_i)$.

ski-rental problem

dilemma of a skier:

- rent equipment for 10 EUR per day
- buy equipment for 10c EUR

doesn't know how many days will be skiing

online algorithm $\mathcal{A}$

input $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$
output $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) = \langle y_1, y_2, \ldots, y_n \rangle$, where $y_i = f(x_1, \ldots, x_i)$.

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

ski-rental problem

dilemma of a skier:

- rent equipment for 10 EUR per day
- buy equipment for 10c EUR

doesn't know how many days will be skiing

online algorithm $\mathcal{A}$

input $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$
output $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) = \langle y_1, y_2, \ldots, y_n \rangle$, where $y_i = f(x_1, \ldots, x_i)$.

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x})) \Leftarrow$ offline

ski-rental problem

dilemma of a skier:

- rent equipment for 10 EUR per day
- buy equipment for $10c$ EUR

doesn't know how many days will be skiing

Theorem [Karp 1992]

optimal worst-case strategy:
rent $c-1$ days, then buy
competitive ratio:

$$\frac{\mathrm{cost}(\mathcal{A}(\boldsymbol{x}))}{\mathrm{cost}(OPT(\boldsymbol{x}))} \leq \frac{2c-1}{c} = 2 - \frac{1}{c}$$

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: OPT vs. k-times "more powerfull" online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: *OPT* vs. k-times “more powerfull” online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: *OPT* vs. k-times "more powerfull" online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: *OPT* vs. k-times “more powerfull” online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy

c-competitive algorithm

for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: *OPT* vs. *k*-times “more powerfull” online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy

c-competitive algorithm

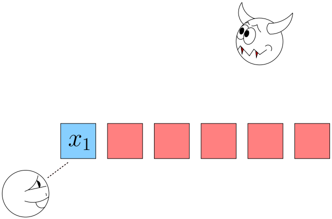
for each input $\boldsymbol{x}$: $\mathrm{cost}(\mathcal{A}(\boldsymbol{x})) \leq c \cdot \mathrm{cost}(OPT(\boldsymbol{x}))$

Definition: problem complexity

Best attainable competitive ratio.

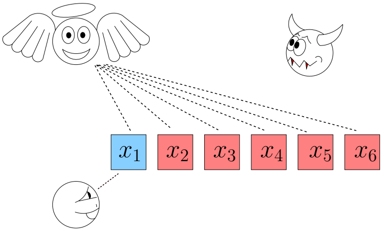
other approaches

- loose competitiveness,...: taylored for Paging
- resource augmentation: *OPT* vs. *k*-times “more powerfull” online
- look-ahead: alg. can see a number of future inputs
- online vs online: Max/Max ratio, relative worst-order ratio,...
- limited adversary: access graph, statistical, diffuse, ...
- entropy



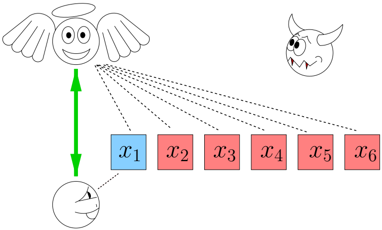
idea

- algorithm can see part of the input
- oracle sees the whole input
- they can communicate



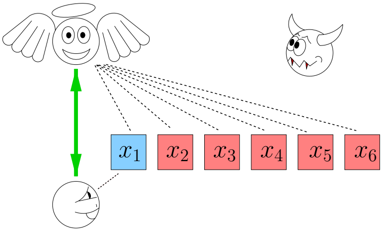
idea

- algorithm can see part of the input
- oracle sees the whole input
- they can communicate



idea

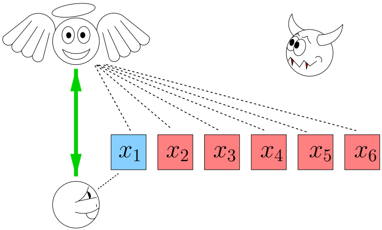
- algorithm can see part of the input
- oracle sees the whole input
- they can communicate



idea

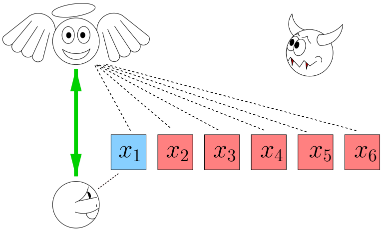
- algorithm can see part of the input
- oracle sees the whole input
- they can communicate

problem complexity ≈ # bits to achieve *OPT*



communication

- answerer: algorithm asks, gets an answer
- helper: oracle can give spontanuous advice

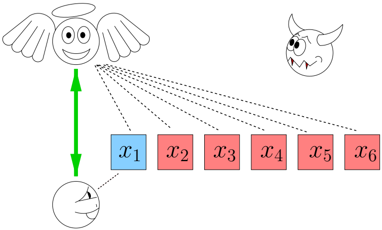


communication

- answerer: algorithm asks, gets an answer
- helper: oracle can give spontanuous advice

trivial bounds

- encode whole input
- encode output



Definition: helper

input $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$

helper sequence: $\mathcal{O}(\boldsymbol{x}) = \langle \boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n} \rangle$

output: $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle$, $y_i = f(x_1, \ldots, x_i, \boldsymbol{a_1}, \ldots, \boldsymbol{a_i})$

advice (bit) complexity:

$$B^H_{(\mathcal{A},\mathcal{O})} = \limsup_{n \mapsto \infty} \max_{|\boldsymbol{X}|=n} \frac{\sum_{i=1}^n |\boldsymbol{a_i}|}{n}$$

our results

	competitive ratio	helper	answerer
PAGING	K	$(0.1775, 0.2056)$	$(0.4591, 0.5+\varepsilon)$
DIFFSERV	≈ 1.281	$\frac{1}{K}$	$\left(\frac{\log K}{2K}, \frac{\log K}{K}\right)$

4	4	3	7	....

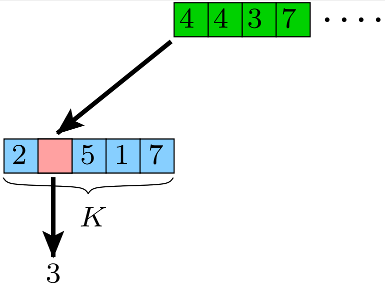
$$\underbrace{\boxed{2}\boxed{3}\boxed{5}\boxed{1}\boxed{7}}_{K}$$

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

Theorem [Sleator, Tarjan 1985]

Every deterministic algorithm is at least K competitive.



Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

Theorem [Sleator, Tarjan 1985]

Every deterministic algorithm is at least K competitive.

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

- a page is active, if shall be used by OPT
- replace only inactive pages
- with each input helper tells if the page is active

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

- a page is active, if shall be used by OPT
- replace only inactive pages
- with each input helper tells if the page is active

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

- a page is active, if shall be used by OPT
- replace only inactive pages
- with each input helper tells if the page is active

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

Replace the farthest-requested page.

online algorithm with helper

- a page is active, if shall be used by OPT
- replace only inactive pages
- with each input helper tells if the page is active

1 bit per input request $\rightarrow$ can be optimized

Paging

- input: logical pages $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, $x_i > 0$
- buffer: physical memory $B = \{b_1, \ldots, b_K\}$
- if $x_i \notin B \Rightarrow$ page fault, a victim has to be found

optimal offline algorithm

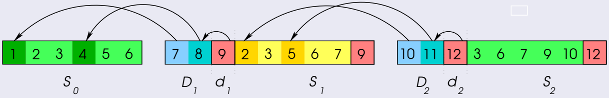
Replace the farthest-requested page.

online algorithm with helper

- a page is active, if shall be used by OPT
- replace only inactive pages
- with each input helper tells if the page is active

1 bit per input request $\rightarrow$ can be optimized

lower bound



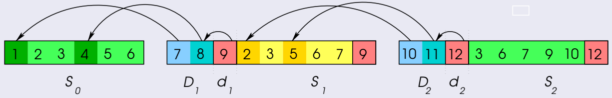
- input: S_0, blocks of length $3K$
- optimal cannot generate fault in S_i
- distinct inputs have distinct advice sequences
- the number of distinct inputs is

$$Y = \binom{3K-1}{2K-1}^i = \left[\frac{2}{3}\binom{3K}{K}\right]^i$$

- the number of advice sequences with at most s bits is X:

$$\log X \leq s\left(\log(1+\alpha) + 1 + \frac{1}{\ln 2}\right) + \frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right) + \log s\right] + c$$

lower bound



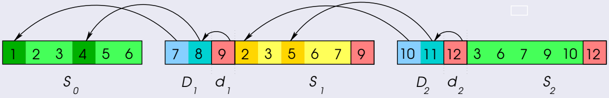
- input: S_0, blocks of length $3K$
- optimal cannot generate fault in S_i
- distinct inputs have distinct advice sequences
- the number of distinct inputs is

$$Y = \binom{3K-1}{2K-1}^i = \left[\frac{2}{3}\binom{3K}{K}\right]^i$$

- the number of advice sequences with at most s bits is X:

$$\log X \le s\left(\log(1+\alpha) + 1 + \frac{1}{\ln 2}\right) + \frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right) + \log s\right] + c$$

lower bound



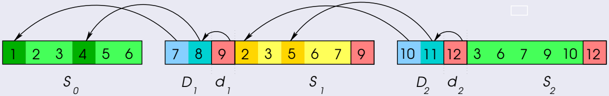
- input: S_0, blocks of length $3K$
- optimal cannot generate fault in S_i
- distinct inputs have distinct advice sequences
- the number of distinct inputs is

$$Y = \binom{3K-1}{2K-1}^i = \left[\frac{2}{3}\binom{3K}{K}\right]^i$$

- the number of advice sequences with at most s bits is X:

$$\log X \leq s\left(\log(1+\alpha) + 1 + \frac{1}{\ln 2}\right) + \frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right) + \log s\right] + c$$

lower bound



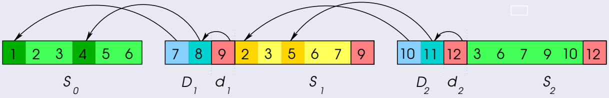
- input: S_0, blocks of length $3K$
- optimal cannot generate fault in S_i
- distinct inputs have distinct advice sequences
- the number of distinct inputs is

$$Y = \binom{3K-1}{2K-1}^i = \left[\frac{2}{3}\binom{3K}{K}\right]^i$$

- the number of advice sequences with at most s bits is X:

$$\log X \leq s\left(\log(1+\alpha) + 1 + \frac{1}{\ln 2}\right) + \frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right) + \log s\right] + c$$

lower bound



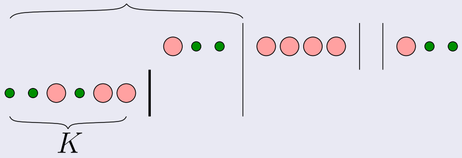
- input: S_0, blocks of length $3K$
- optimal cannot generate fault in S_i
- distinct inputs have distinct advice sequences
- the number of distinct inputs is

$$Y = \binom{3K-1}{2K-1}^i = \left[\frac{2}{3}\binom{3K}{K}\right]^i$$

- the number of advice sequences with at most s bits is X:

$$\log X \leq s\left(\log(1+\alpha) + 1 + \frac{1}{\ln 2}\right) + \frac{1}{2}\left[\log\left(1+\frac{1}{\alpha}\right) + \log s\right] + c$$

DiffServ



- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

skipped

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

Motivation

- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

Motivation

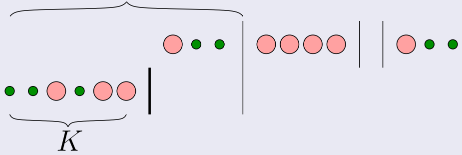
- problem complexity $\approx$ amount of relevant information
- possible applications – semi-online setting

Open directions

- other problems
- bounded advice
- trade-off between approximation and information
- randomization

Thank you for your atention

DiffServ



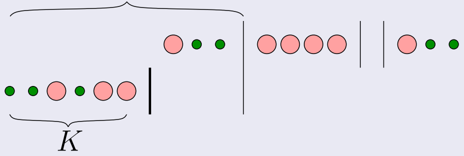
- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

critical input

- buffer starts with small item
- buffer eventually fills with large items

Small items mus be discarded!

DiffServ



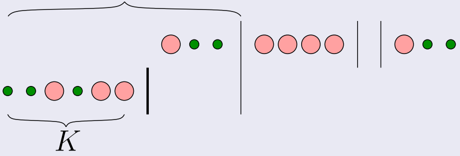
- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

critical input

- buffer starts with small item
- buffer eventually fills with large items

Small items mus be discarded!

DiffServ



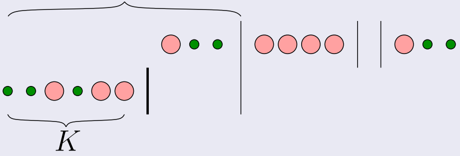
- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

critical input

- buffer starts with small item
- buffer eventually fills with large items

Small items mus be discarded!

DiffServ



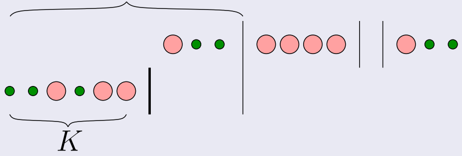
- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

critical input

- buffer starts with small item
- buffer eventually fills with large items

Small items mus be discarded!

DiffServ



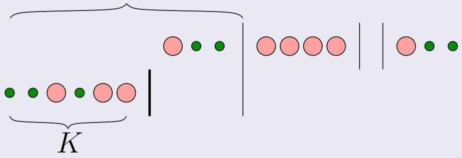
- buffer of size K, input: sequence of large and small items
- process one item, discard some to fit into buffer
- order must be preserved

critical input

- buffer starts with small item
- buffer eventually fills with large items

Small items mus be discarded!

DiffServ



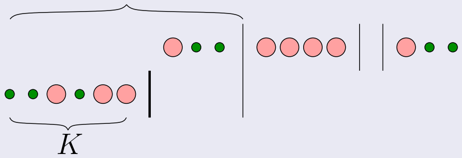
critical input

- buffer starts with small item
- buffer eventually fills with large items

upper bound

critical input at most ever $K+1$ steps $\Rightarrow \frac{1}{K+1}$ bits per input

DiffServ



critical input

- buffer starts with small item
- buffer eventually fills with large items

upper bound

critical input at most ever $K+1$ steps $\Rightarrow \frac{1}{K+1}$ bits per input