Online algorithms with predictions

Spyros Angelopoulos

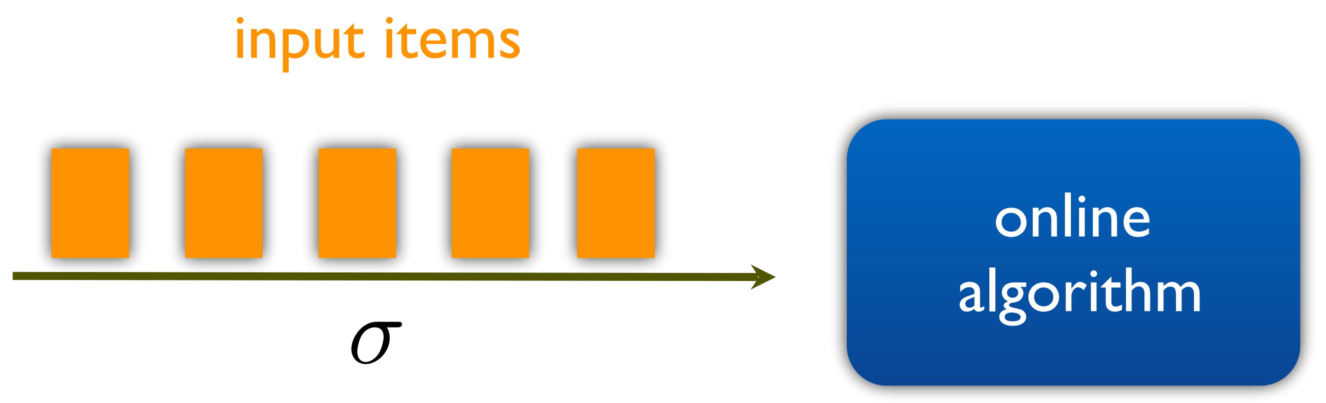
CoreLab Seminar, NTUA, March 29 2021

Leveraging predictions in a status of uncertainty

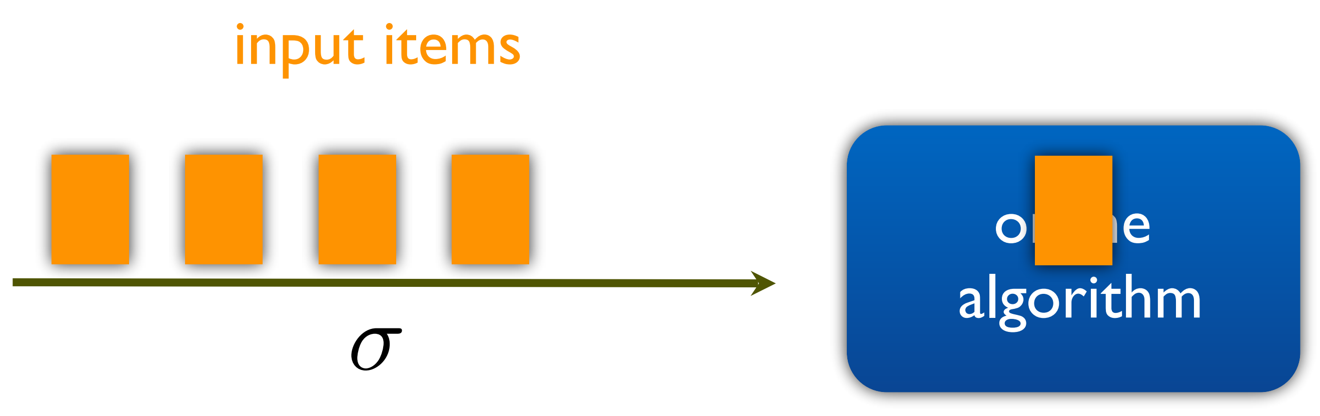
Online computation: the standard model

Online computation: the standard model

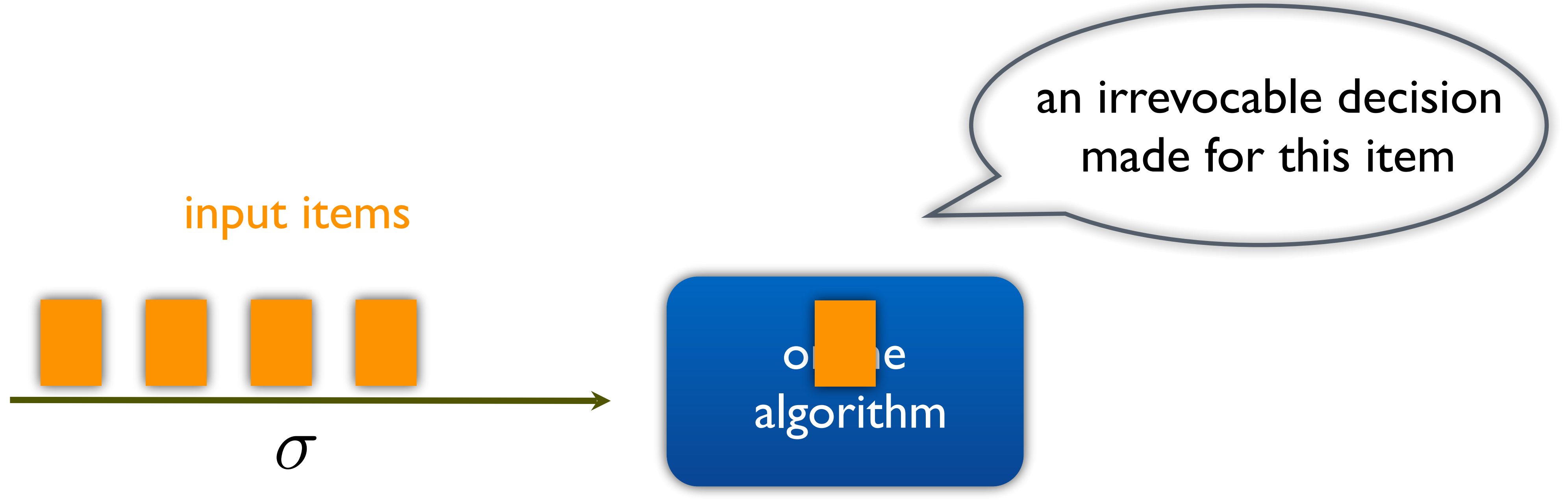
Online computation: the standard model



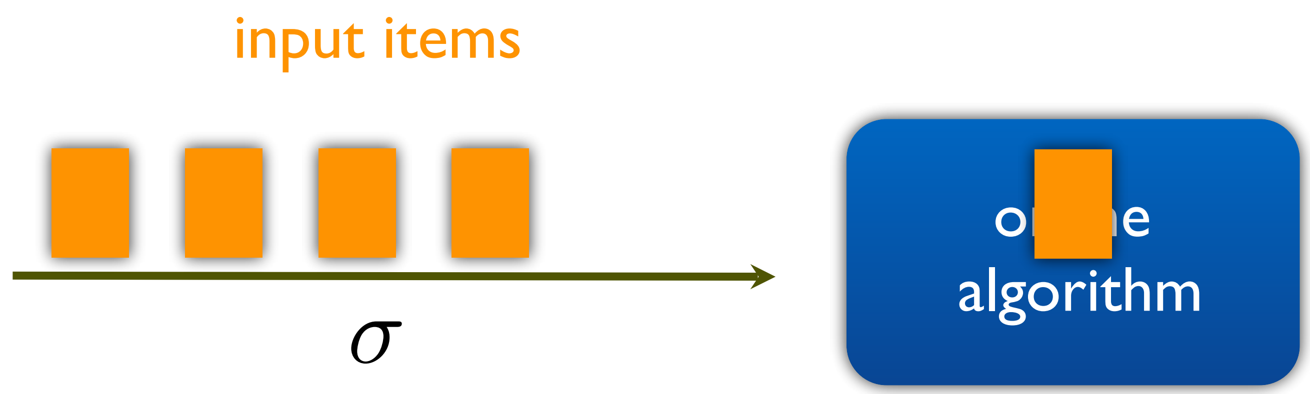
Online computation: the standard model



Online computation: the standard model



Online computation: the standard model



Online computation: the standard model

Online computation: the standard model

In the standard model: No assumptions about the future input items

Online computation: the standard model

In the standard model: No assumptions about the future input items

Competitive analysis: main analysis technique since the mid 80s [Sleator and Tarjan 85]

Competitive ratio of algorithm A:

$$\sup_{\sigma} \frac{\text{cost of } A \text{ on } \sigma}{\mathrm{OPT}(\sigma)}$$

How to enhance the standard model of
online computation so as to deal with predictions
concerning the input?

First approach: online computation with *advice*

First approach: online computation with *advice*

First approach: online computation with *advice*

First approach: online computation with *advice*

- If ϕ(σ) = empty => Standard online computation
- If ϕ(σ) encodes the optimal decisions => Optimal offline performance

First approach: online computation with *advice*

- If φ(σ) = empty => Standard online computation
- If φ(σ) encodes the optimal decisions => Optimal offline performance

General question: what lies between these two extremes ?

Advice complexity of online problems

Advice complexity of online problems

Definition [Dobrev et al. 2009, Böckenhauer et al. 2009, Emek et al. 2011]

An online problem P is c-competitive with advice of size f(n) if there is a c-competitive algorithm for P with advice tape of size at most f(n), where n is the length of the request sequence σ

Advice complexity of online problems

Definition [Dobrev et al. 2009, Böckenhauer et al. 2009, Emek et al. 2011]

An online problem P is c-competitive with advice of size f(n) if there is a c-competitive algorithm for P with advice tape of size at most f(n), where n is the length of the request sequence σ

Applications in many problems: paging, list update, makespan scheduling, k-server, bin packing, graph colouring, Steiner trees and many, many others

[Komm: An introduction to online computation, Springer 2016]

[Boyar et al. Online computation with advice: A survey, ACM Computing Surveys, 2017]

Advice complexity model: mostly theoretical

- Focus on size of the encoded advice
- Advice oracle can be overly powerful
- Advice is guaranteed to be error-free and trustworthy

Advice in the real world

advice

/ədˈvʌɪs/

noun

1. guidance or recommendations offered with regard to prudent future action.
   "my advice is to see your doctor"
   synonyms: guidance, advising, counselling, counsel, help, direction, instruction, information, enlightenment; More
2. a formal notice of a financial transaction.
   "remittance advices"

Second approach: machine-learned predictions

Second approach: machine-learned predictions

A **prediction** associated with the input which is inherently **erroneous**

The prediction has error η (unknown to the algorithm)

Second approach: machine-learned predictions

A **prediction** associated with the input which is inherently **erroneous**

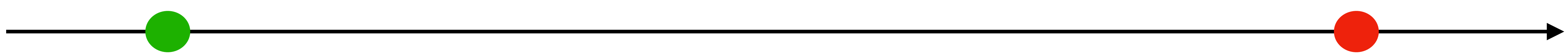
The prediction has error η (unknown to the algorithm)

Second approach: machine-learned predictions

A **prediction** associated with the input which is inherently **erroneous**

The prediction has error η (unknown to the algorithm)

Robustness : competitive ratio
with *adversarial* error

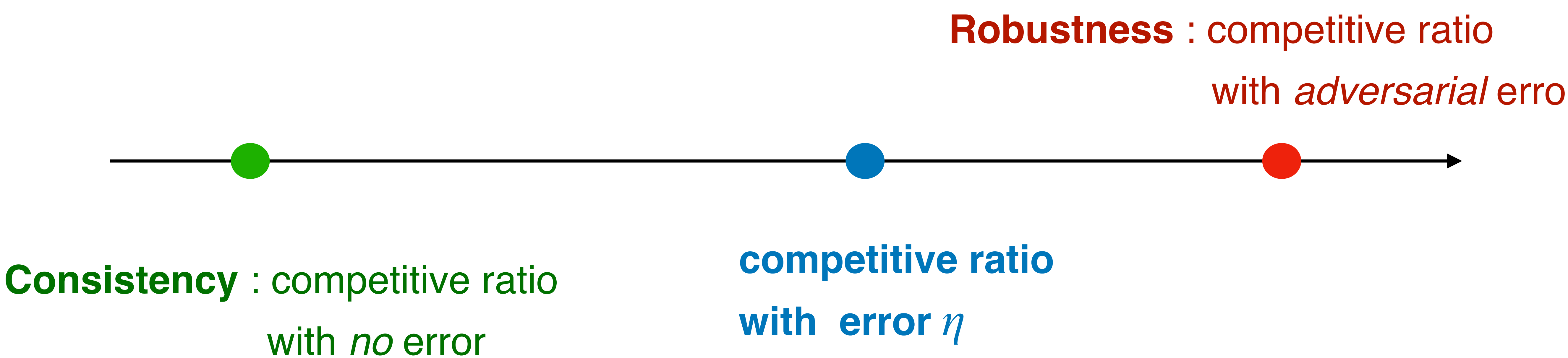


Consistency : competitive ratio
with *no* error

Second approach: machine-learned predictions

A **prediction** associated with the input which is inherently **erroneous**

The prediction has error η (unknown to the algorithm)

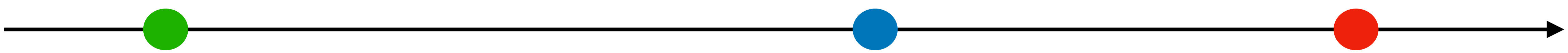


Second approach: machine-learned predictions

A **prediction** associated with the input which is inherently **erroneous**

The prediction has error η (unknown to the algorithm)

Robustness : competitive ratio with *adversarial* error



Consistency : competitive ratio with *no* error

competitive ratio with error η

objectives

Mostly upper bounds on the competitive ratio with error

"Smooth" degradation with error

Experimental validation

Related works

[Lykouris and Vassilvitskii, ICML 2018] : Introduced consistency, robustness in paging

[Purohit, Svitkina, Kumar, NeurIPS 2018] : Other online problems

Many other recent works…

[Mitzenmacher and Vassilvitskii 2020] : survey of (some) recent results

Related works

[Lykouris and Vassilvitskii, ICML 2018] : Introduced consistency, robustness in paging

[Purohit, Svitkina, Kumar, NeurIPS 2018] : Other online problems

Many other recent works…

[Mitzenmacher and Vassilvitskii 2020] : survey of (some) recent results

Broader direction: Analysis of algorithms **beyond the worst case**

Summary of our work

[A, Dürr, Jin, Kamali, Renault: ITCS 2020]
Online computation with untrusted advice
https://arxiv.org/pdf/1905.05655.pdf

[A.: ITCS 2021]
Online search with a hint
https://arxiv.org/pdf/2008.13729.pdf

[A. and Kamali : AAAI 2021]
Contract scheduling with predictions
https://arxiv.org/pdf/2011.12439.pdf

[A., Kamali and Shadkami 2021]
Online bin packing with predictions
https://arxiv.org/pdf/2102.03311.pdf

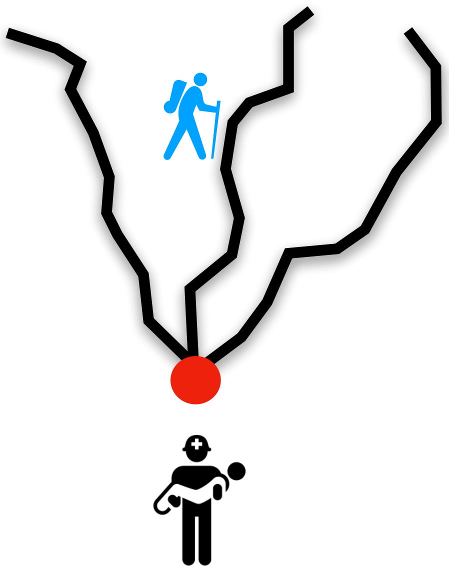
Summary of our work

[A, Dürr, Jin, Kamali, Renault: ITCS 2020]
Online computation with untrusted advice
https://arxiv.org/pdf/1905.05655.pdf

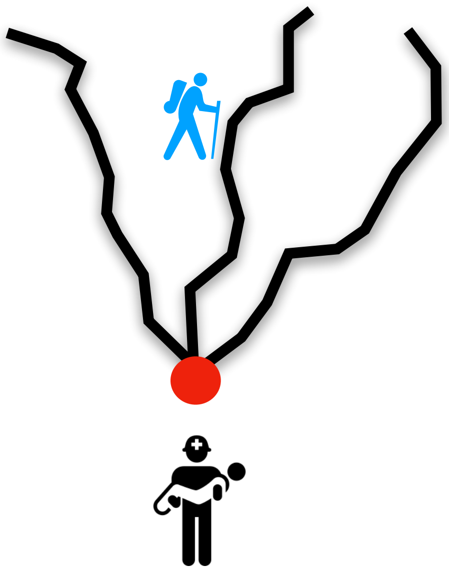
[A.: ITCS 2021]
Online search with a hint
https://arxiv.org/pdf/2008.13729.pdf

[A. and Kamali : AAAI 2021]
Contract scheduling with predictions
https://arxiv.org/pdf/2011.12439.pdf

[A., Kamali and Shadkami 2021]
Online bin packing with predictions
https://arxiv.org/pdf/2102.03311.pdf



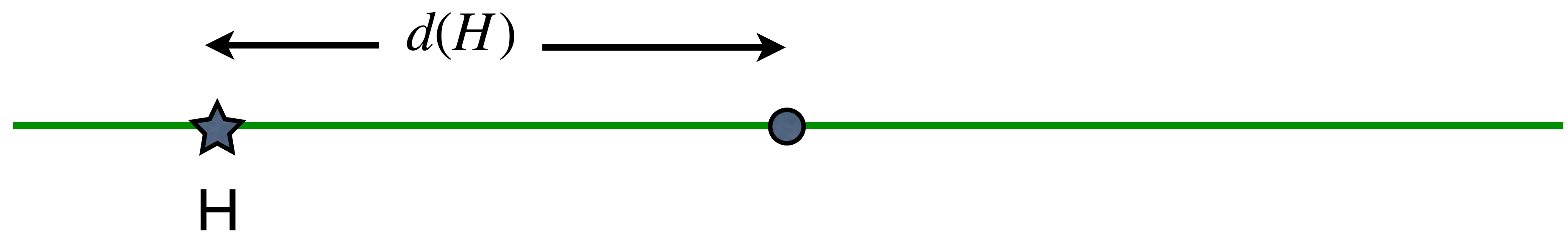
Part 1 : Searching with a hint



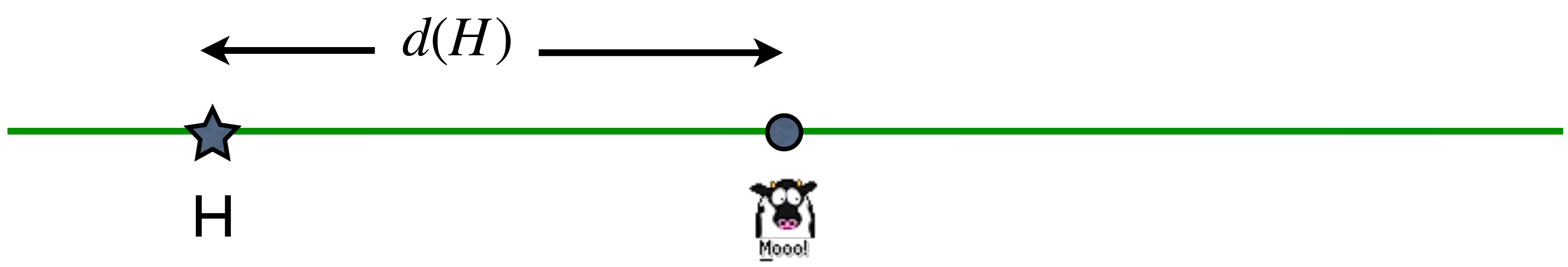
Start simple: searching on the line (with no hints)

Start simple: searching on the line (with no hints)

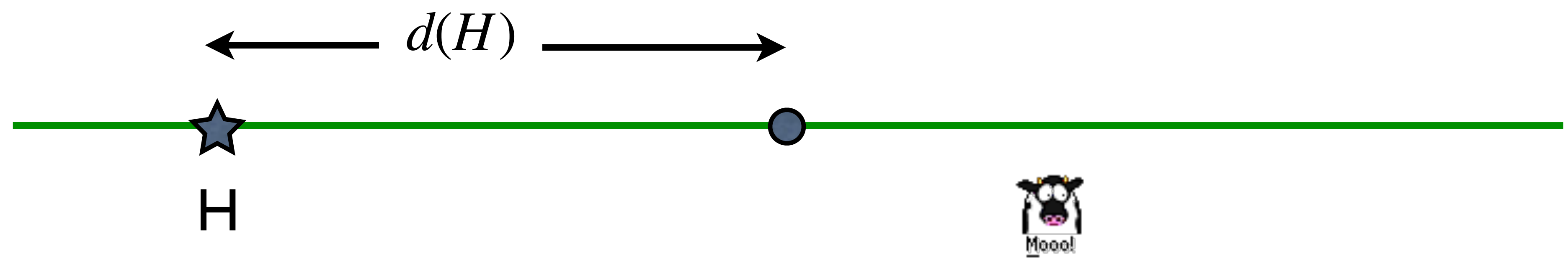
Start simple: searching on the line (with no hints)



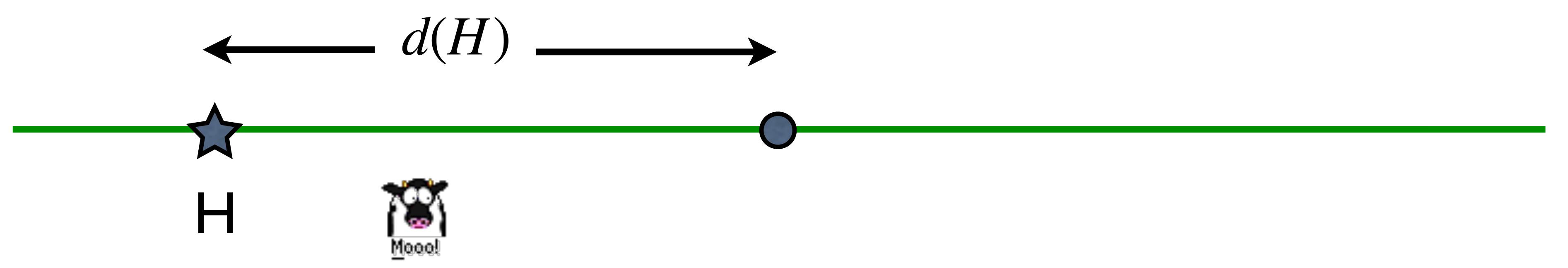
Start simple: searching on the line (with no hints)



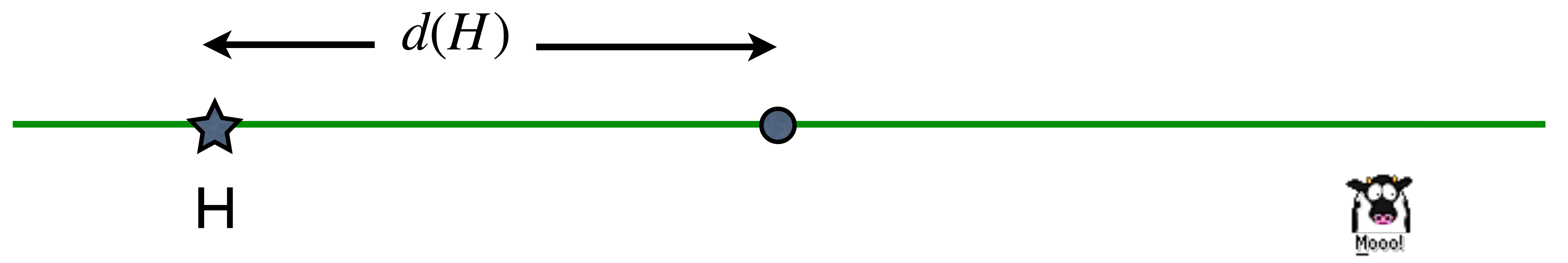
Start simple: searching on the line (with no hints)



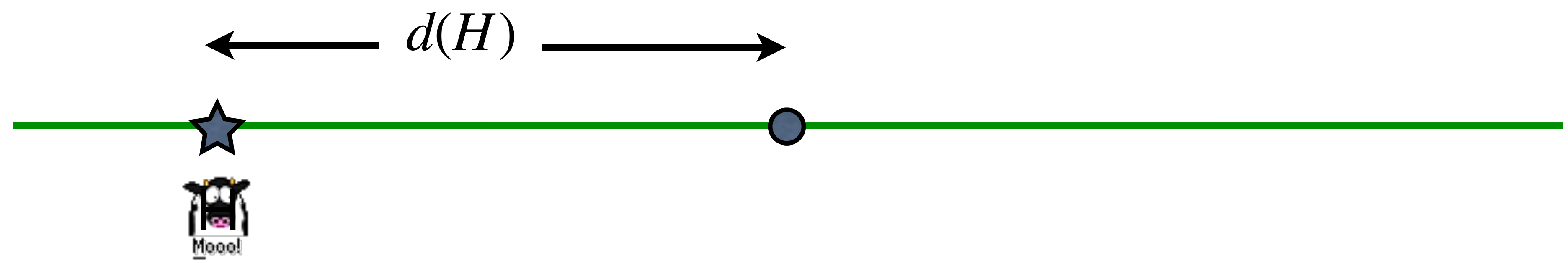
Start simple: searching on the line (with no hints)



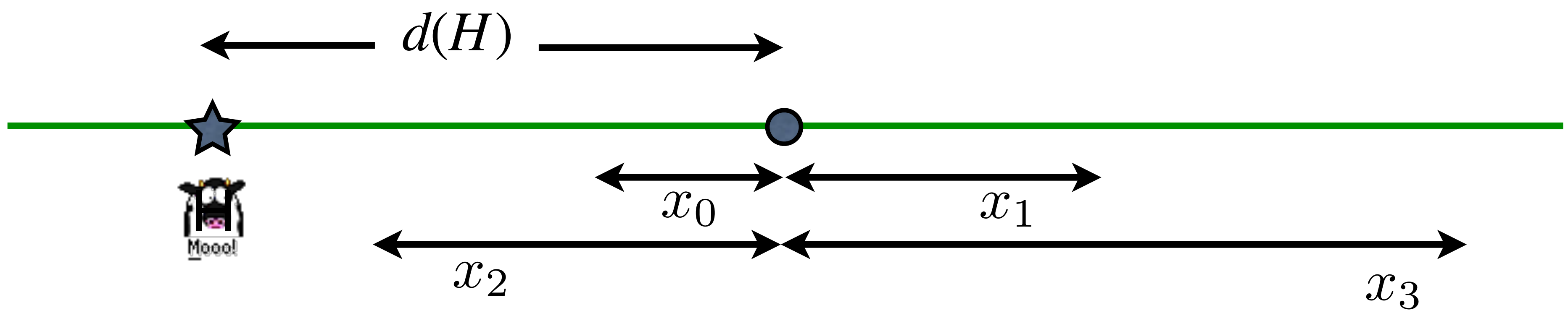
Start simple: searching on the line (with no hints)



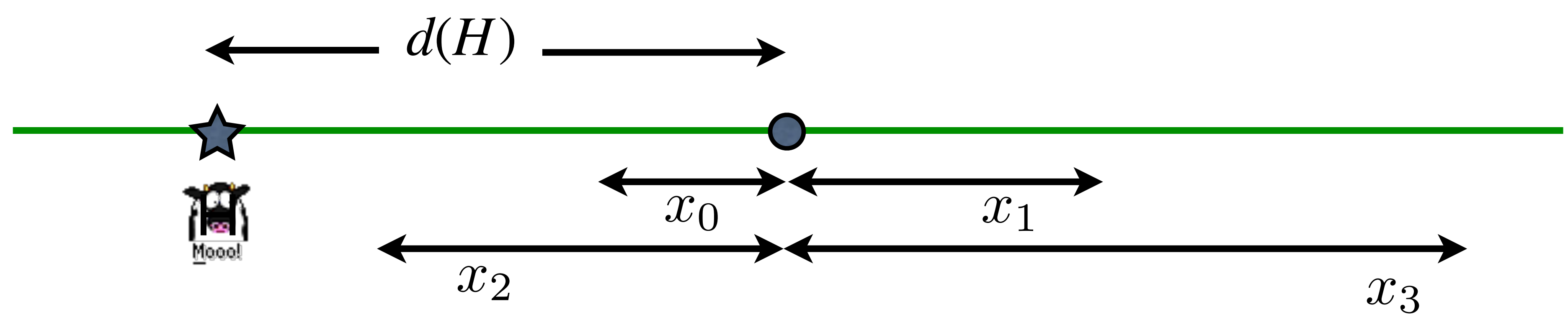
Start simple: searching on the line (with no hints)



Start simple: searching on the line (with no hints)



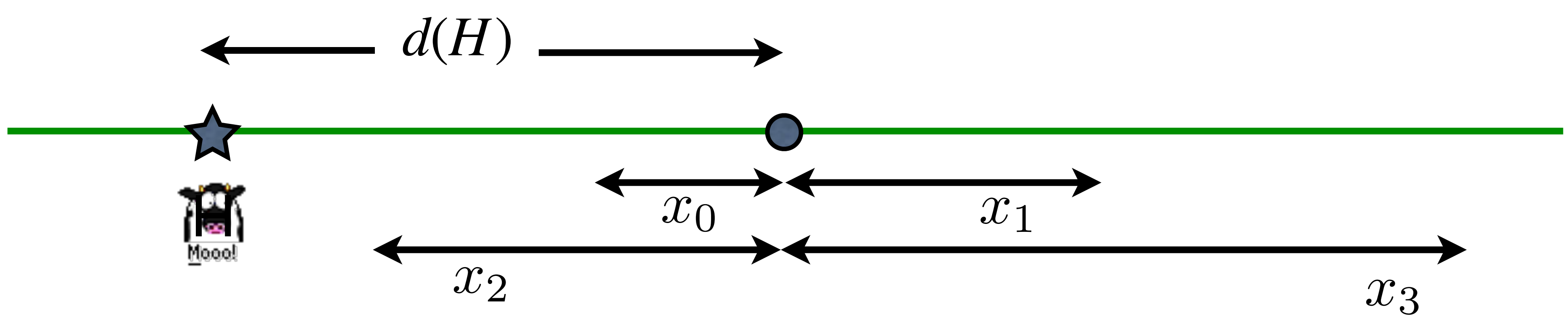
Start simple: searching on the line (with no hints)



- Searcher aims to minimize the competitive ratio of its strategy S

$$\text{cr}(S) = \sup_H \frac{\text{distance traversed by the searcher using } S}{d(H)}$$

Start simple: searching on the line (with no hints)

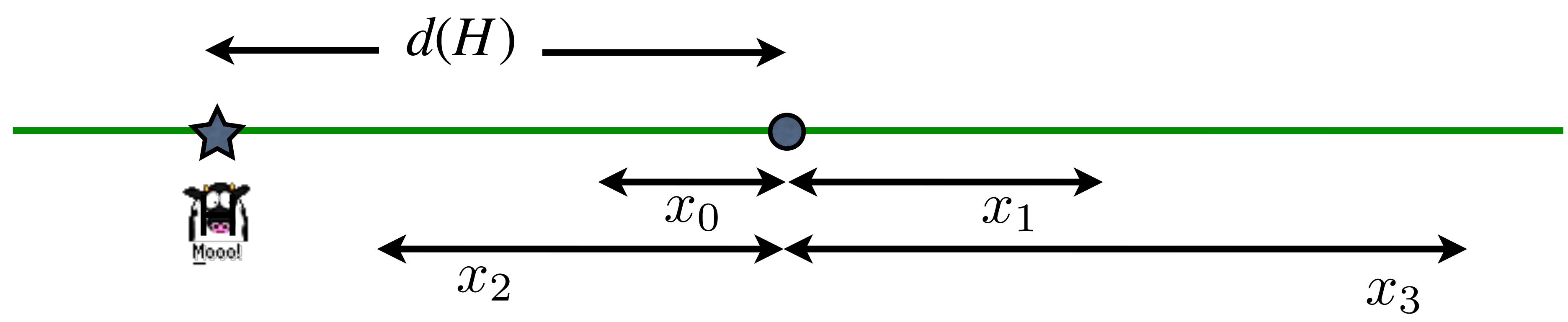


- Searcher aims to minimize the competitive ratio of its strategy S

$$\text{cr}(S) = \sup_H \frac{\text{distance traversed by the searcher using } S}{d(H)}$$

- Optimal deterministic competitive ratio = 9 using $x_i = 2^i$ [Beck and Newman 70]

Start simple: searching on the line (with no hints)



- Searcher aims to minimize the competitive ratio of its strategy S

$$\mathrm{cr}(S) = \sup_H \frac{\text{distance traversed by the searcher using } S}{d(H)}$$

- Optimal deterministic competitive ratio = 9 using $x_i = 2^i$ [Beck and Newman 70]

- Many studies of extensions [Alpern and Gal, *The Theory of Search Games and Rendevous*, 2003]

Hints in online search

Hints in online search

Hint h: some information that is given to the searcher

The search strategy $S(h)$ is now a function of the hint

- If hint is trusted, then it is guaranteed to be correct
- If hint is untrusted, then it is generated adversarially

Hints in online search

Hint h: some information that is given to the searcher

The search strategy $S(h)$ is now a function of the hint

- If hint is trusted, then it is guaranteed to be correct
- If hint is untrusted, then it is generated adversarially

$$\text{Competitiveness of S(h)} = (c_{S,h}, r_{S,h})$$

Hints in online search

Hint h: some information that is given to the searcher

The search strategy $S(h)$ is now a function of the hint

- If hint is trusted, then it is guaranteed to be correct
- If hint is untrusted, then it is generated adversarially

Consistency: c.r. if hint is correct

$$\text{Competitiveness of S(h)} = (c_{S,h}, r_{S,h})$$

Hints in online search

Hint h: some information that is given to the searcher

The search strategy $S(h)$ is now a function of the hint

- If hint is trusted, then it is guaranteed to be correct
- If hint is untrusted, then it is generated adversarially

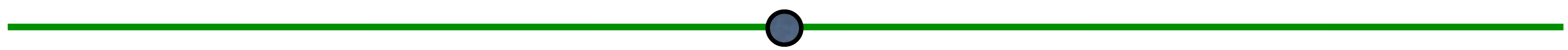
Consistency: c.r. if hint is correct

$$\text{Competitiveness of S(h)} = (c_{S,h}, r_{S,h})$$

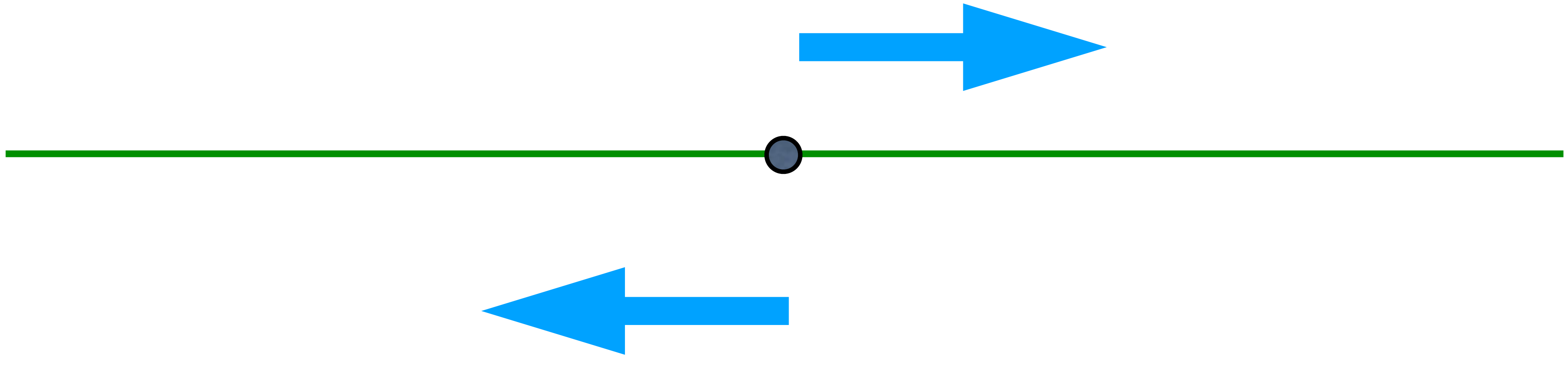
Robustness : c.r. if hint is adversarial

An example

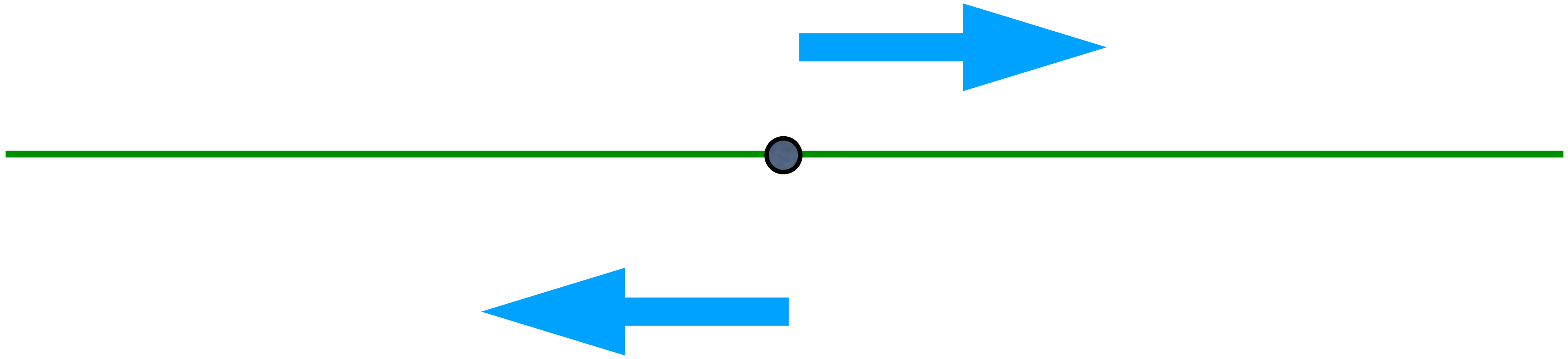
An example



An example

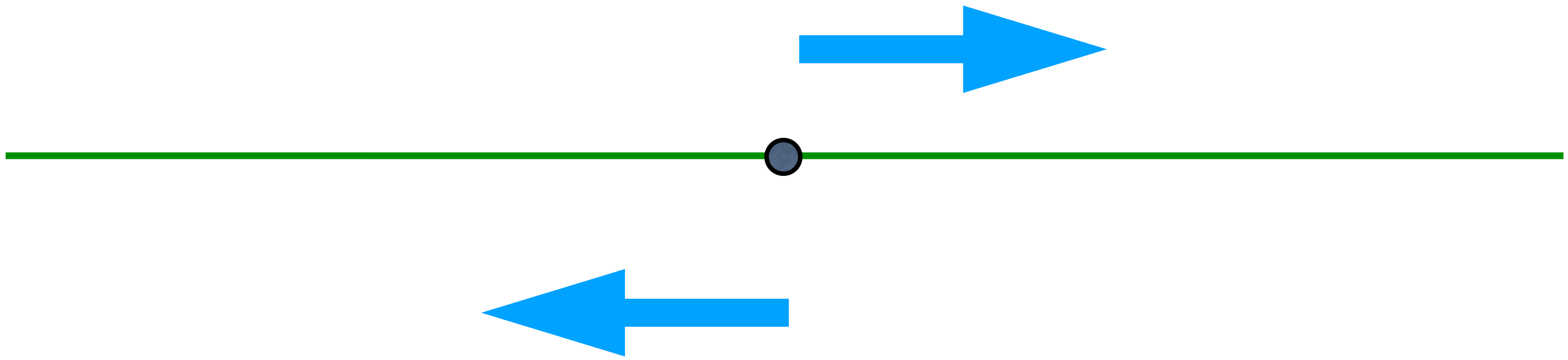


An example



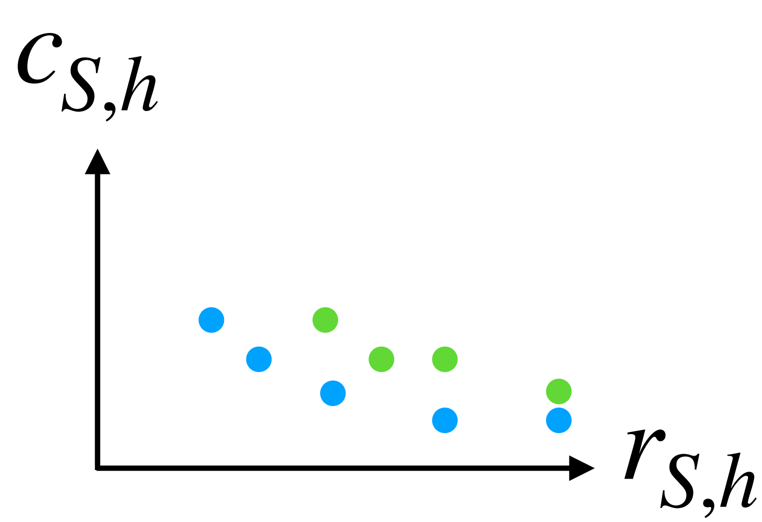
- A strategy that always trusts the hint is $(1, \infty)$ competitive
- The doubling strategy that ignores the hint is $(9, 9)$ competitive

An example



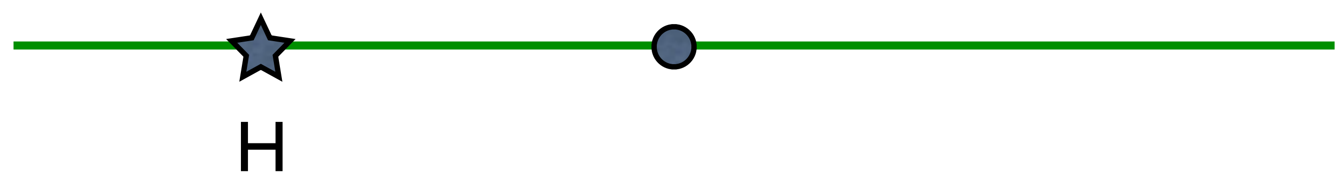
- A strategy that always trusts the hint is $(1,\infty)$ competitive
- The doubling strategy that ignores the hint is $(9,9)$ competitive

Pareto efficiency

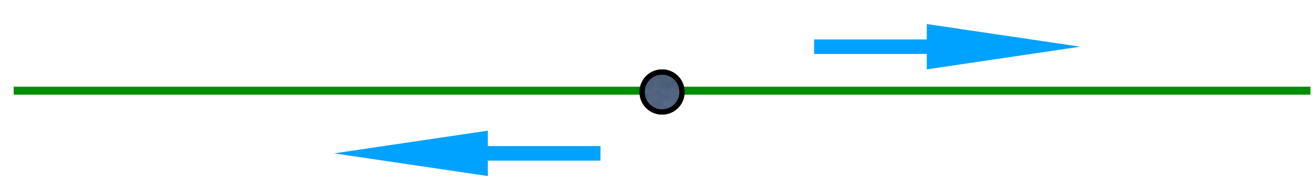


Types of hints

1 The hint is the exact **position** of the target



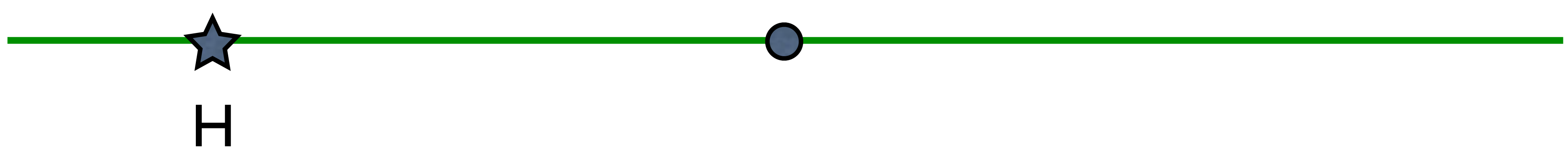
2 The hint is the **direction** of the search (left or right)

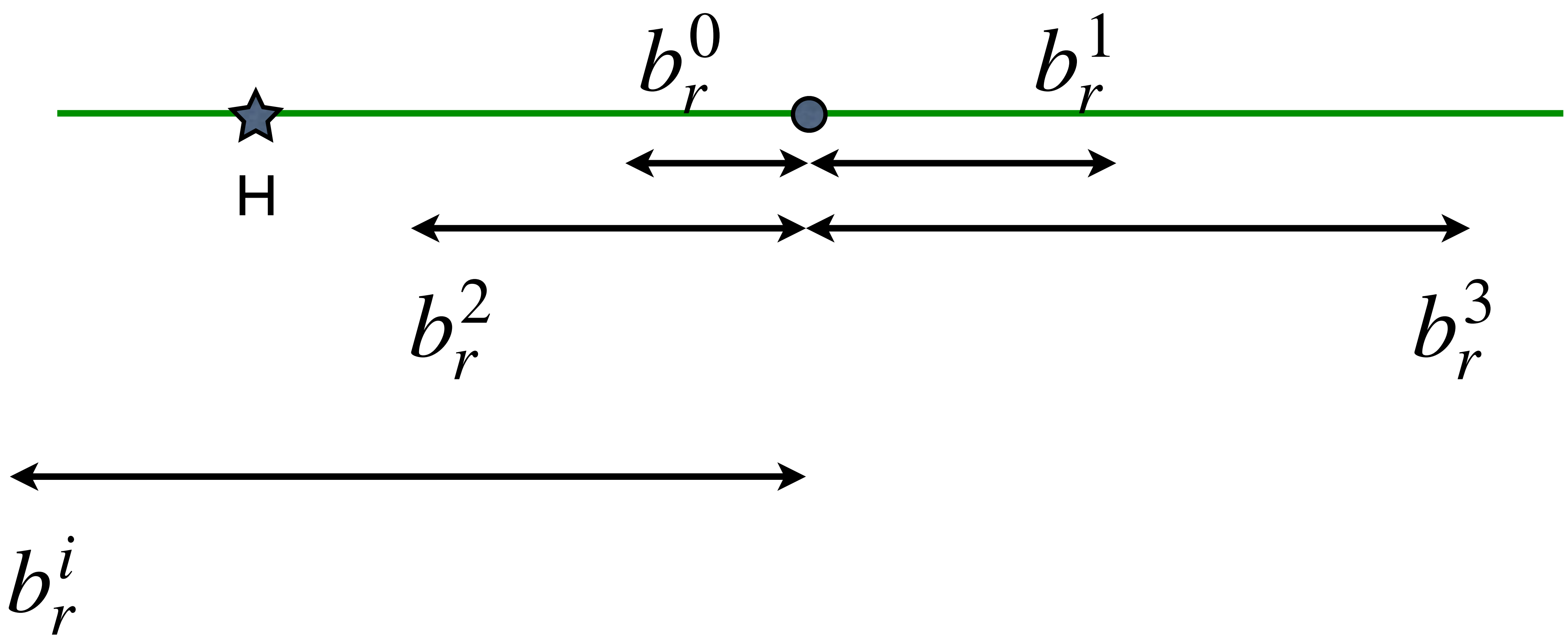


3 The hint is a **k-bit string**

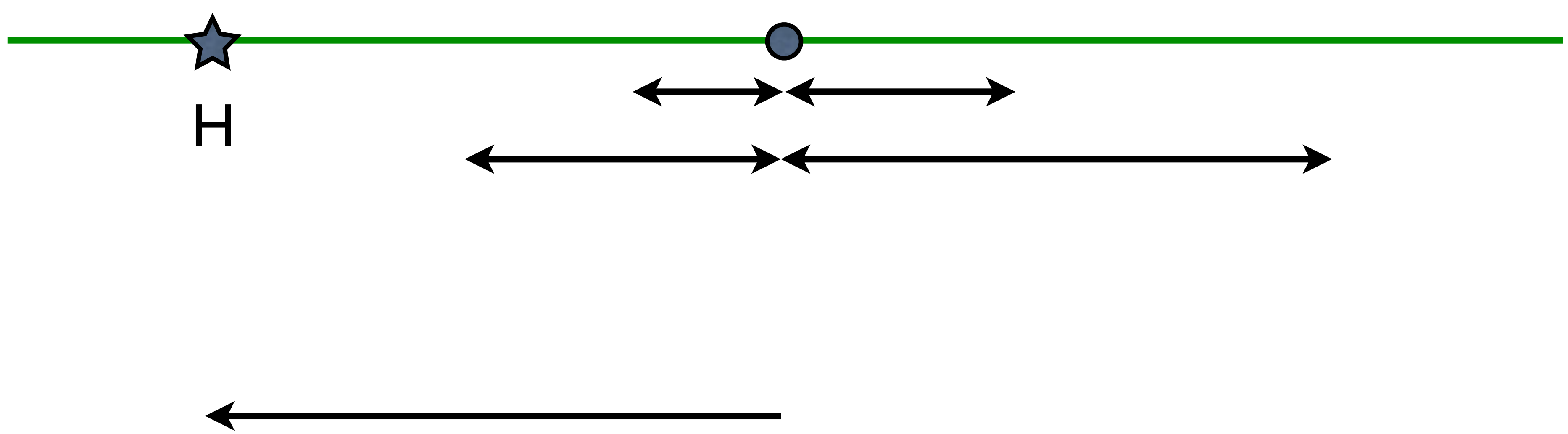
`01101...1`

1. The hint is the exact **position** of the target





$$b_r = \frac{r + \sqrt{r^2 - 4r}}{2}$$


H

Results for hint as position

Upper bound: The previous strategy is $\left(\frac{b_r + 1}{b_r - 1}, r\right)$-competitive

Lower bound: No other strategy is better (Pareto optimality)

Helpful lemma. For every r-robust strategy it holds that

$$x_i \leq \left(b_r + \frac{b_r}{i+1}\right) x_{i-1}, \text{ for every i}$$

Summary of results for the other settings

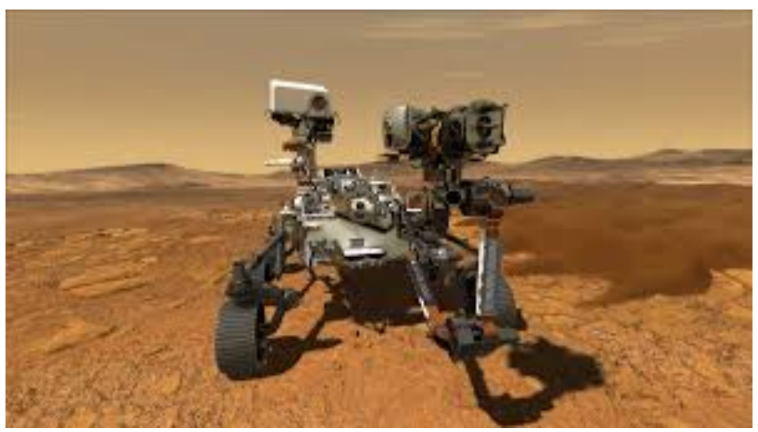
		Upper bounds	Lower bounds	Techniques
Hint= direction	Pareto-optimal strategies			Functional theorems [Schuierer 2001] and [Gal 1974]
Hint= k-bit string	k=1	$(1+4\sqrt{2},9)$ Upper bound for general r	No better than $(5,9)$ $(1+2\frac{b_r}{b_r-1},r)$ for restricted strategies	Information-theoretic arguments Adversary-algorithm games Relate the hint to multi-searcher strategies
	k>1	Upper bound for general r	$c \geq 3$	

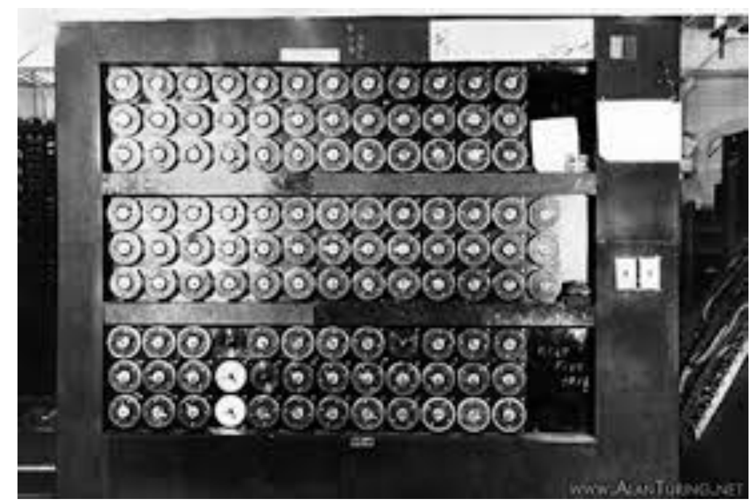
Part 2 : Contract scheduling with predictions

Motivation

Design of systems that are robust to interruptions

Integral requirement of many real-time and anytime applications





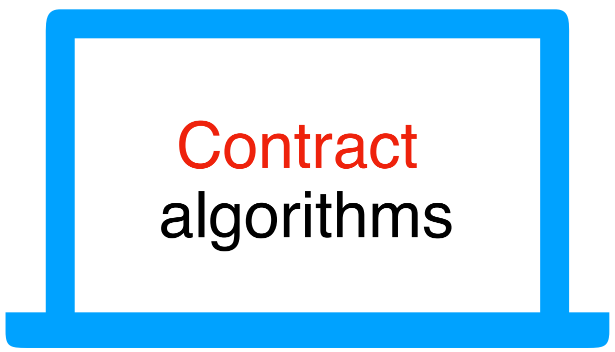
Anytime computation: contract and interruptible algorithms

Anytime computation: contract and interruptible algorithms

Two different types of anytime algorithms [Russell and Zilberstein 1991]

Anytime computation: contract and interruptible algorithms

Two different types of anytime algorithms [Russell and Zilberstein 1991]



Execution time T given as input

If allowed to run up to T: output is correct

If interrupted prior to T, output
may be meaningless

Anytime computation: contract and interruptible algorithms

Two different types of anytime algorithms [Russell and Zilberstein 1991]

Contract algorithms

Execution time T given as input

If allowed to run up to T: output is correct

If interrupted prior to T, output may be meaningless

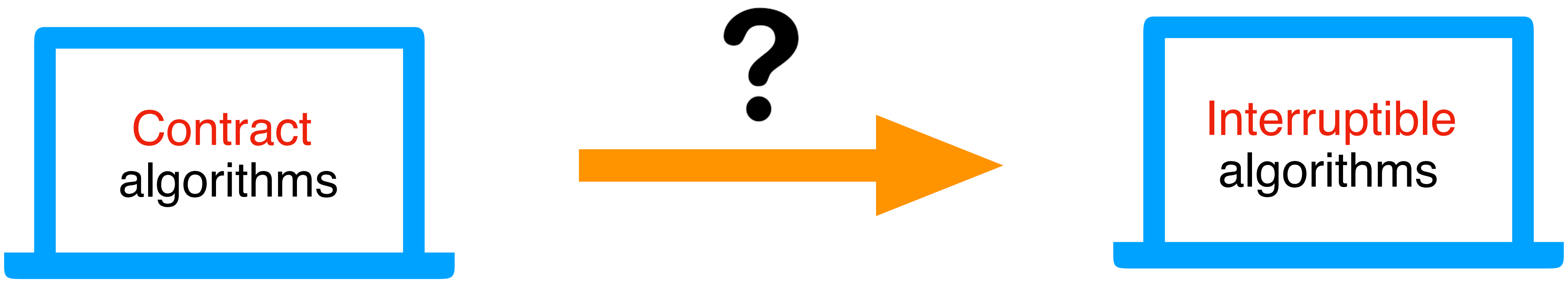
Interruptible algorithms

Return progressively better output as function of time

If interrupted they output "meaningful" results

Anytime computation: contract and interruptible algorithms

Two different types of anytime algorithms [Russell and Zilberstein 1991]



Execution time T given as input

If allowed to run up to T: output is correct

If interrupted prior to T, output may be meaningless

Return progressively better output as function of time

If interrupted they output "meaningful" results

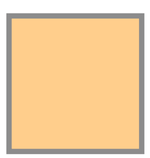
From contract algorithms to interruptible algorithms

From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times

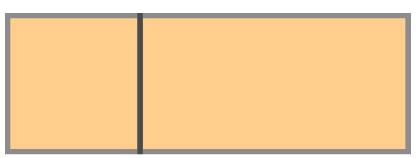
From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



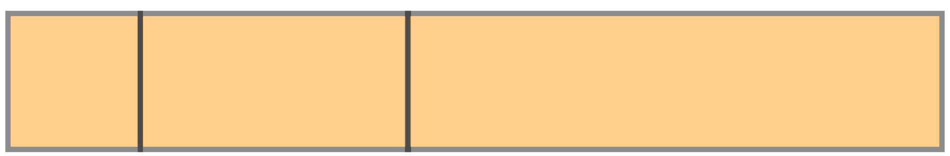
From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



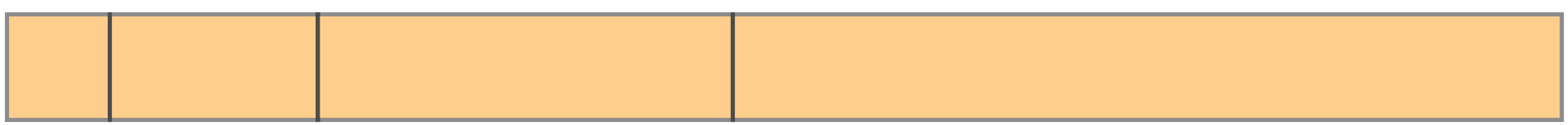
From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



From contract algorithms to interruptible algorithms

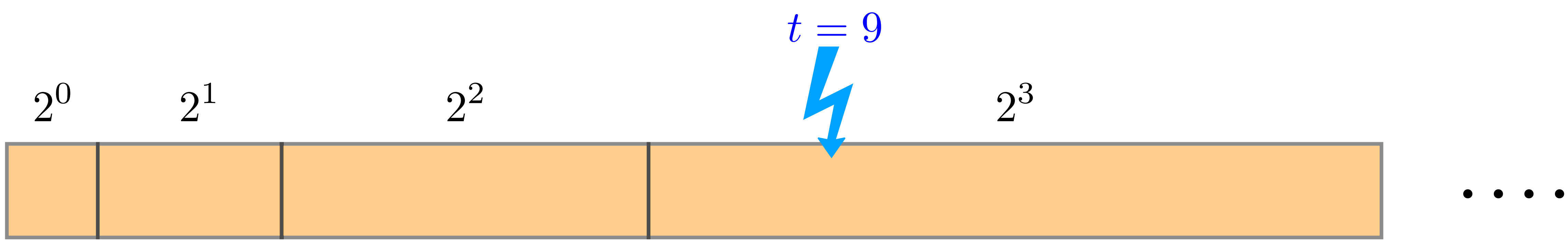
Idea: Schedule executions of the contract algorithm with increasing running times

From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times

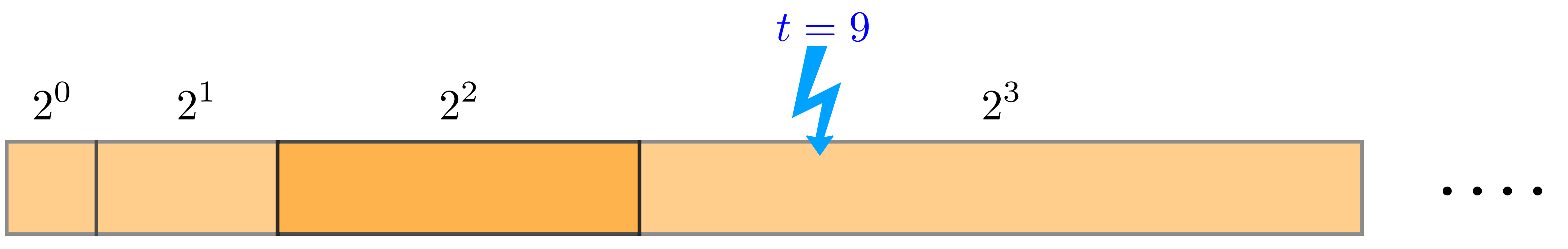
From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



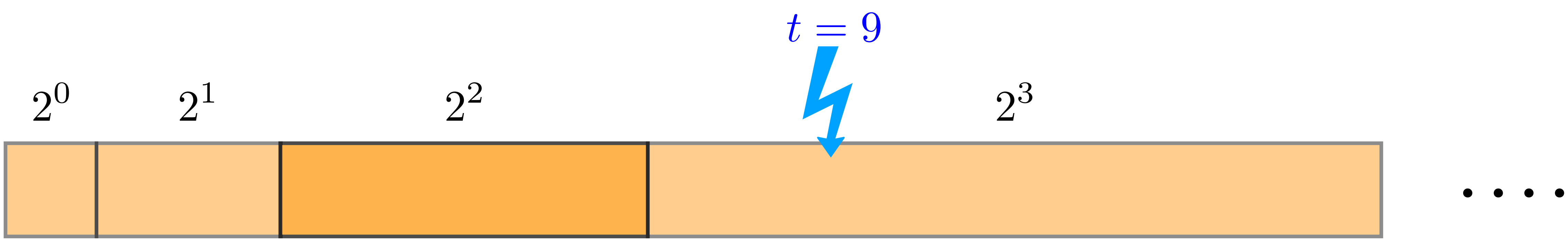
From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



From contract algorithms to interruptible algorithms

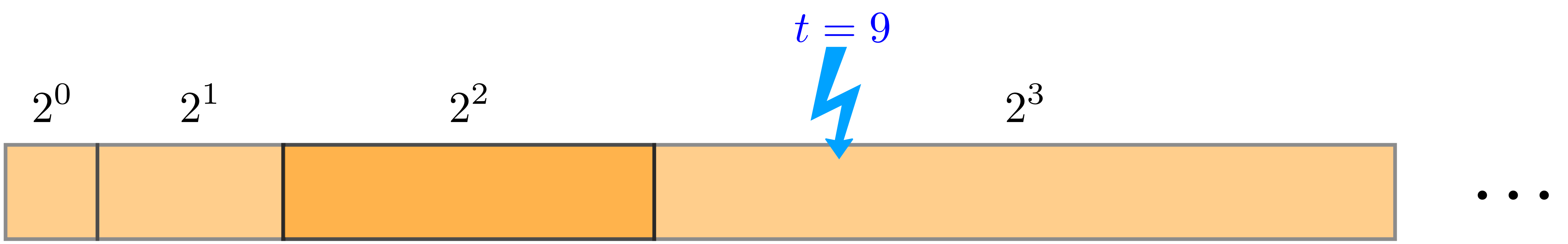
Idea: Schedule executions of the contract algorithm with increasing running times



$$\texttt{acceleration ratio} = \sup_t \frac{t}{\text{largest contract completed by } t}$$

From contract algorithms to interruptible algorithms

Idea: Schedule executions of the contract algorithm with increasing running times



$$\texttt{acceleration ratio} = \sup_t \frac{t}{\text{largest contract completed by } t} = 4(\texttt{optimal})$$

[Russell and Zilberstein 1991]

Related work on contract scheduling

Setting	Reference
1 instance	Russell and Zilberstein 1991
n instances	Zillberstei, Charpillet and Chassaing 2003
1 instance, m processors	Bernstein, Perkins, Finkelstein and Zilberstein 2002
n instances, m processors	Bernstein Finkelstein and Zilberstein 2003
n instances, m processors	López-Ortiz, A, and Hamel 2014
Soft interruptions	A. and López-Ortiz 2017
Alternative measures	A. and López-Ortiz 2009
Connections to searching	Bernstein Finkelstein and Zilberstein 2003 and A. 2015
End guarantees	A. and Jin, 2019

Our setting: Contract scheduling with predictions

Our setting: Contract scheduling with predictions

Information related to interruption

Our setting: Contract scheduling with predictions

Information related
to interruption

- Prediction is **the interruption**

- Prediction is the answer to **n binary queries**

Our setting: Contract scheduling with predictions

Information related to interruption

- Prediction is **the interruption**

actual interruption

$$\tau(1-\eta) \leq T \leq \tau(1+\eta)$$

prediction

error $\in [0,1]$

- Prediction is the answer to **n binary queries**

Our setting: Contract scheduling with predictions

Information related
to interruption

- Prediction is **the interruption**

actual interruption

$$\tau(1-\eta) \leq T \leq \tau(1+\eta)$$

error $\in [0,1]$

prediction

- Prediction is the answer to **n binary queries**

$\eta \in [0,1]$: fraction of the erroneous bits

Our setting: Contract scheduling with predictions

Information related to interruption

- Prediction is **the interruption**

actual interruption

$$\tau(1-\eta) \leq T \leq \tau(1+\eta)$$

prediction

error $\in [0,1]$

- Prediction is the answer to **n binary queries**

$\eta \in [0,1]$: fraction of the erroneous bits

- H : upper bound on η. Distinction between H-**aware** and H-**oblivious** schedules

Prediction is the interruption time

Start with an **ideal** setting: the prediction has **no error ($\eta = 0$)**

This is similar to searching on the line with hint being the position of the target :
we will use the schedule $(b_r^i)_i$

Prediction is the interruption time

Start with an **ideal** setting: the prediction has **no error ($\eta = 0$)**

This is similar to searching on the line with hint being the position of the target :

we will use the schedule $(b_r^i)_i$

For desired robustness r,

this schedule has consistency

$$c = \frac{r - \sqrt{r^2 - 4r}}{2}$$

E.g., for $r = 4$, c= 2

Prediction is the interruption time

Start with an **ideal** setting: the prediction has **no error ($\eta = 0$)**

This is similar to searching on the line with hint being the position of the target :

we will use the schedule $(b_r^i)_i$

Pareto optimal

For desired robustness r,

this schedule has consistency

$$c = \frac{r - \sqrt{r^2 - 4r}}{2}$$

E.g., for $r = 4$, c= 2

Prediction is the interruption time

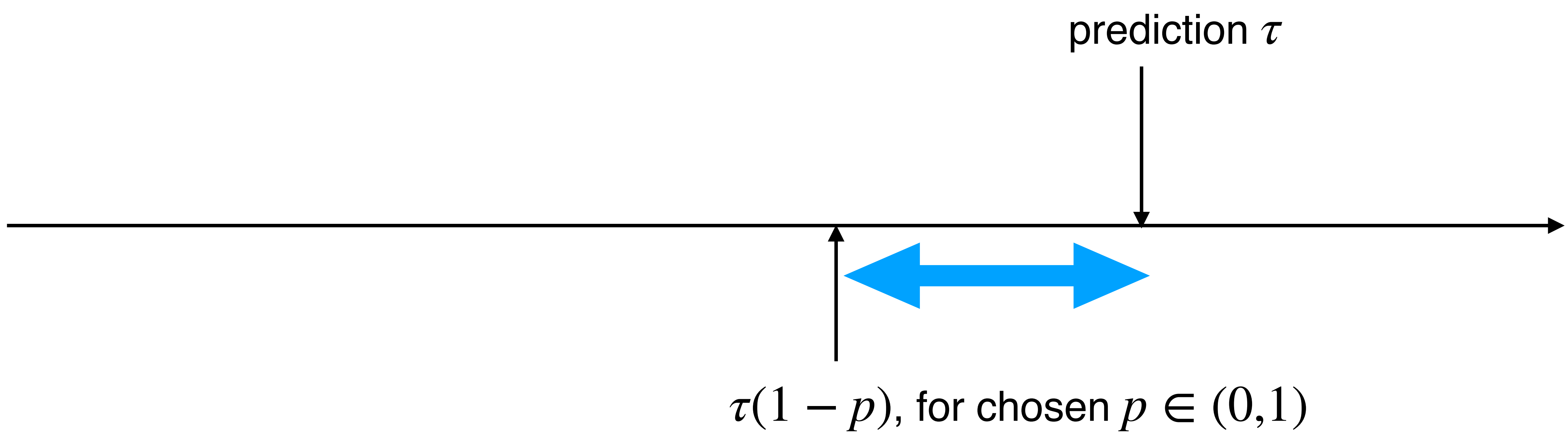
The general setting: The prediction has **error** η

Prediction is the interruption time

The general setting: The prediction has **error** η

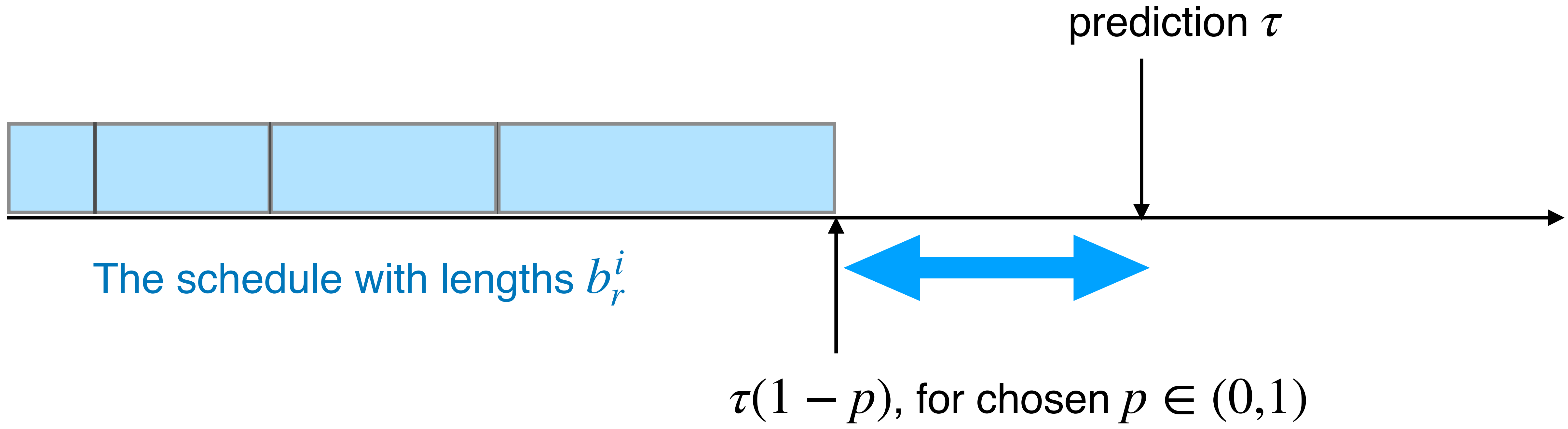
Prediction is the interruption time

The general setting: The prediction has **error** η



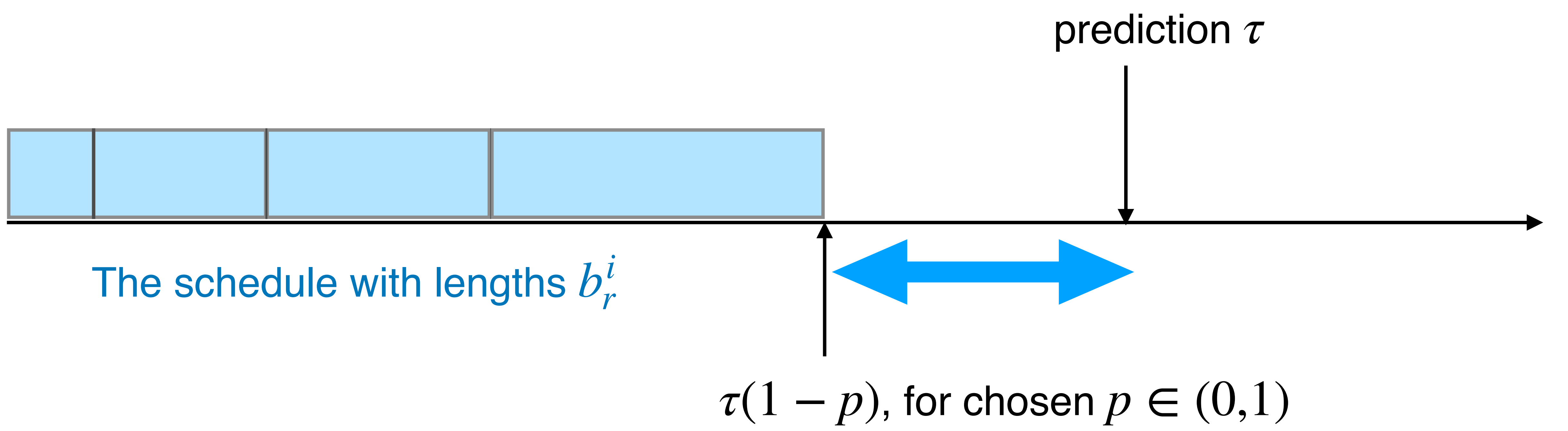
Prediction is the interruption time

The general setting: The prediction has **error** η



Prediction is the interruption time

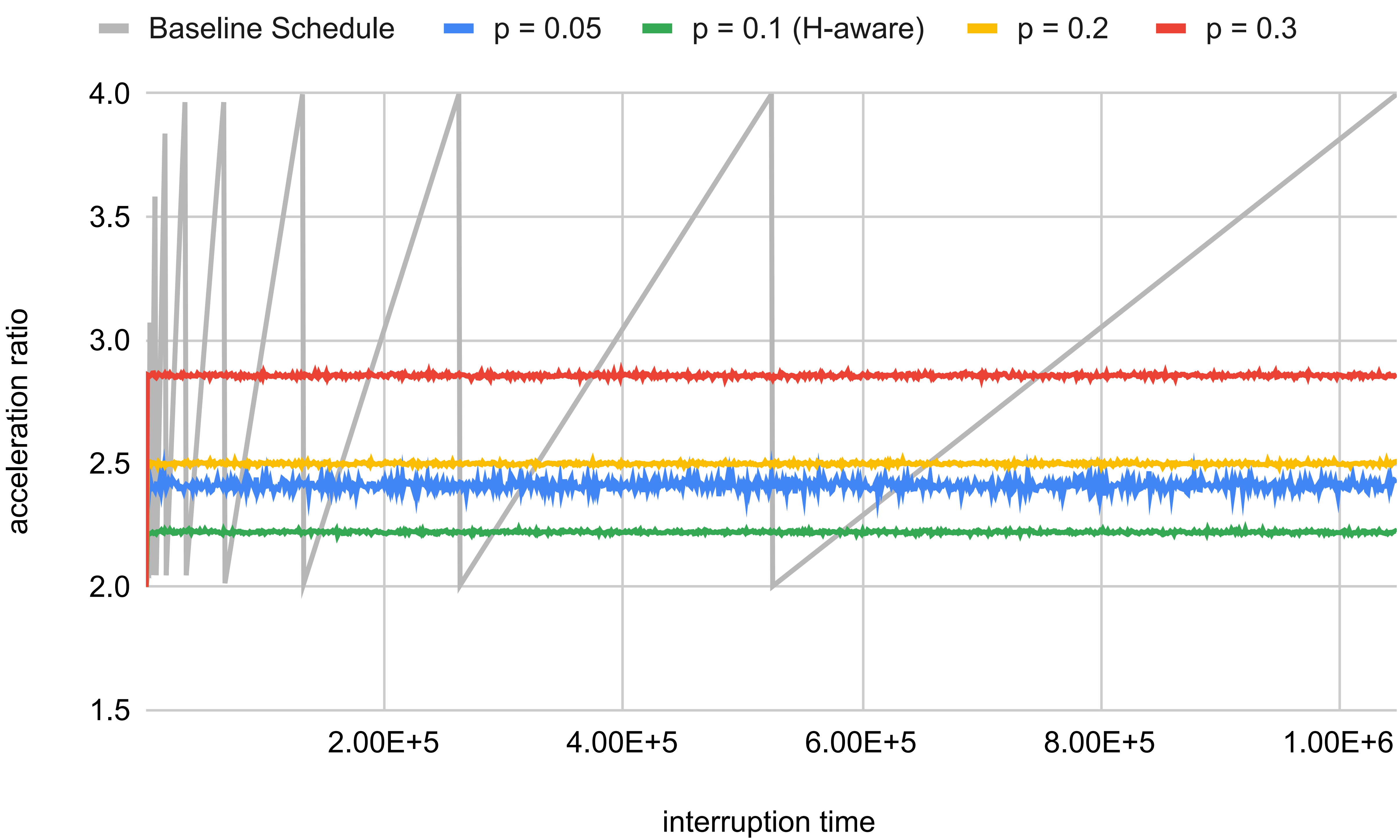
The general setting: The prediction has **error** η



Results

- For H-**oblivious** schedules, this is near-optimal
- For H-**aware** schedules, choosing $p = H$ is near-optimal

Experimental results for r=4, H=0.1



Prediction comes from n binary queries

Start with an **ideal** setting: the prediction has **no error (**$\eta = 0$**)**

Prediction comes from n binary queries

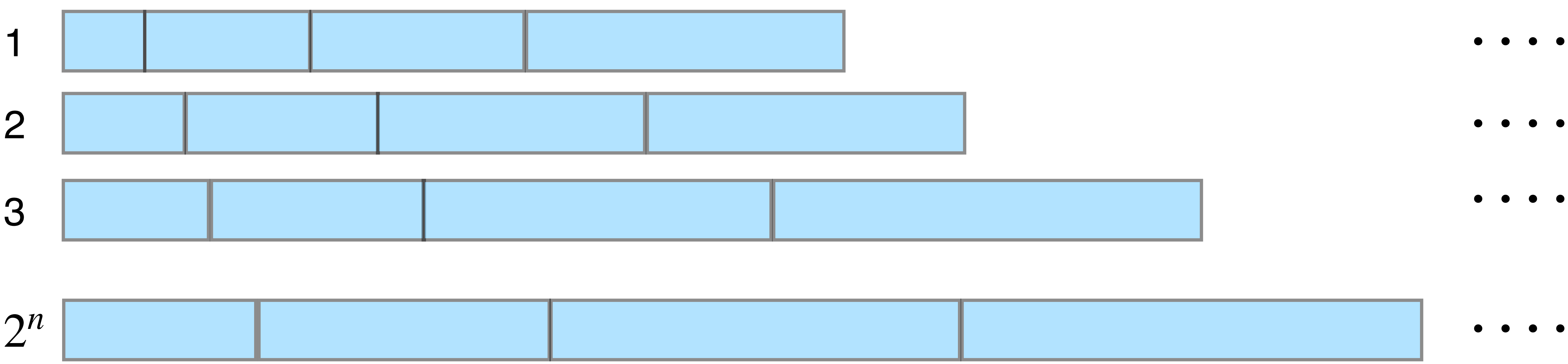
Start with an **ideal** setting: the prediction has **no error ($\eta = 0$)**

Idea: With n bits, we can choose the best among a collection of 2^n schedules

Prediction comes from n binary queries

Start with an **ideal** setting: the prediction has **no error ($\eta = 0$)**

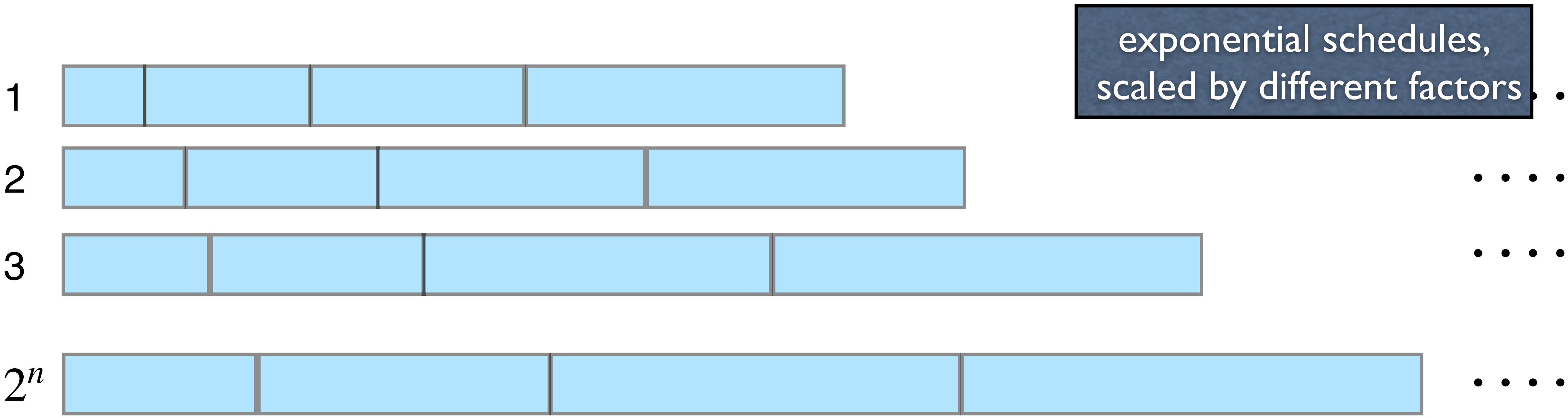
Idea: With n bits, we can choose the best among a collection of 2^n schedules



Prediction comes from n binary queries

Start with an **ideal** setting: the prediction has **no error** ($\eta = 0$)

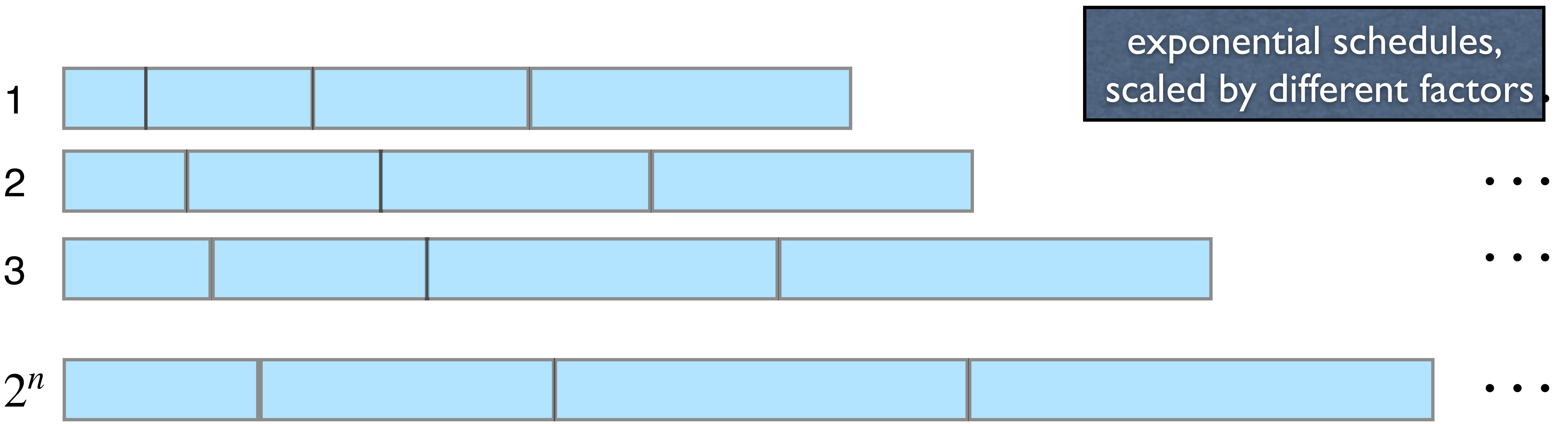
Idea: With n bits, we can choose the best among a collection of 2^n schedules



Prediction comes from n binary queries

Start with an **ideal** setting: the prediction has **no error** ($\eta = 0$)

Idea: With n bits, we can choose the best among a collection of 2^n schedules



Results

- Tradeoff between robustness r and consistency c in terms of n
- E.g., for $r = 4$, we obtain $c = 2^{1+\frac{1}{2n}}$
- This is **Pareto-optimal** for $r = 4$

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Use the n queries to choose the best among n possible schedules

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Use the n queries to choose the best among n possible schedules

Use some ``buffer" $p \in (0,1)$ on how much error we can tolerate

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Use the n queries to choose the best among n possible schedules

Use some ``buffer" $p \in (0,1)$ on how much error we can tolerate

The i-th query is of the form: "Is the best schedule among the i first ones?"

Prediction comes from n binary queries

The general setting: the prediction has **error** η

Outline of the approach

Use the n queries to choose the best among n possible schedules

Use some ``buffer'' $p \in (0,1)$ on how much error we can tolerate

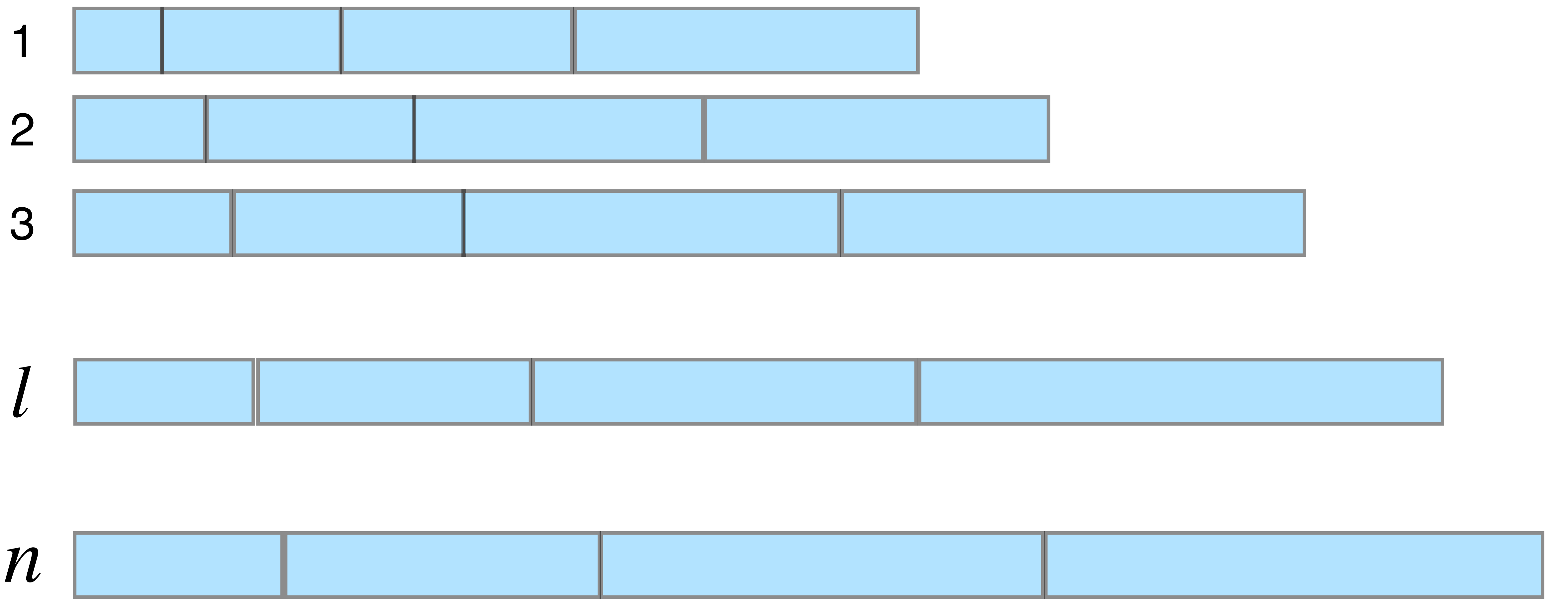
The i-th query is of the form: "Is the best schedule among the i first ones?"

Results

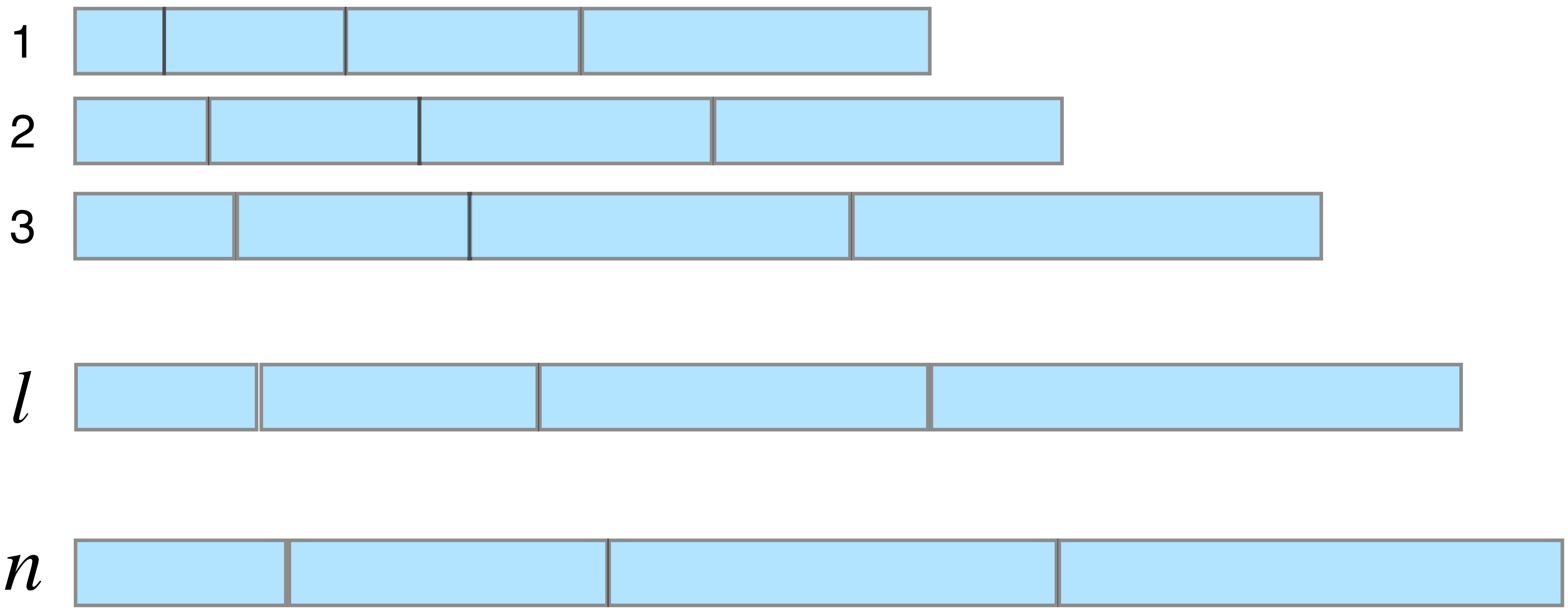
- Tradeoff between robustness r and consistency c, in terms of n and p
- E.g., for $r = 4$, we obtain $c = 2^{1+\frac{1}{n}+2p}$

Some details in choosing the schedule

Some details in choosing the schedule

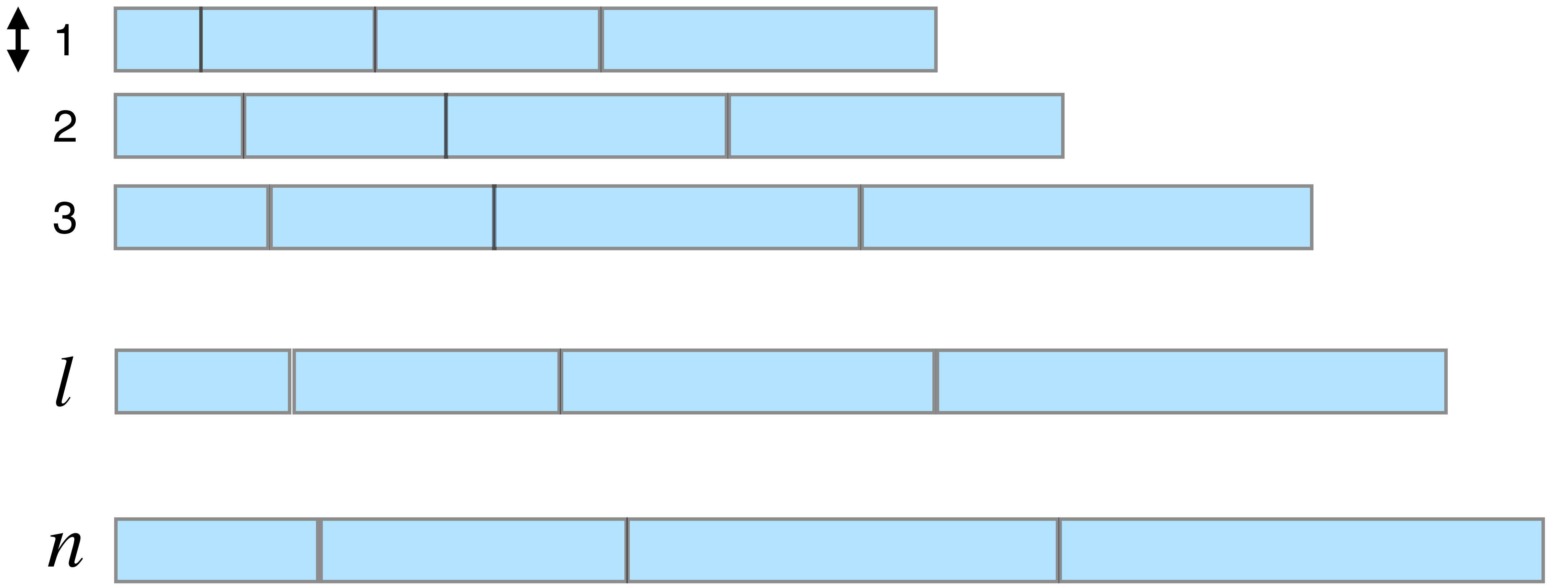


Some details in choosing the schedule



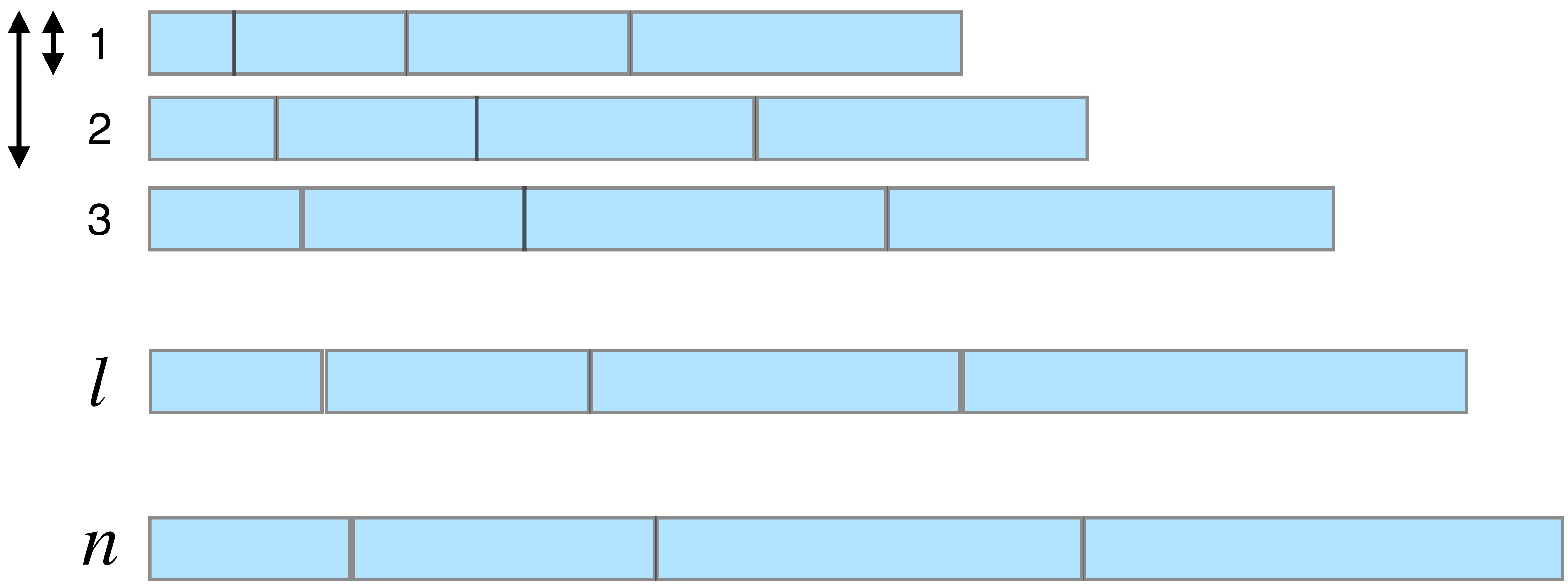
index of best schedule : l

Some details in choosing the schedule



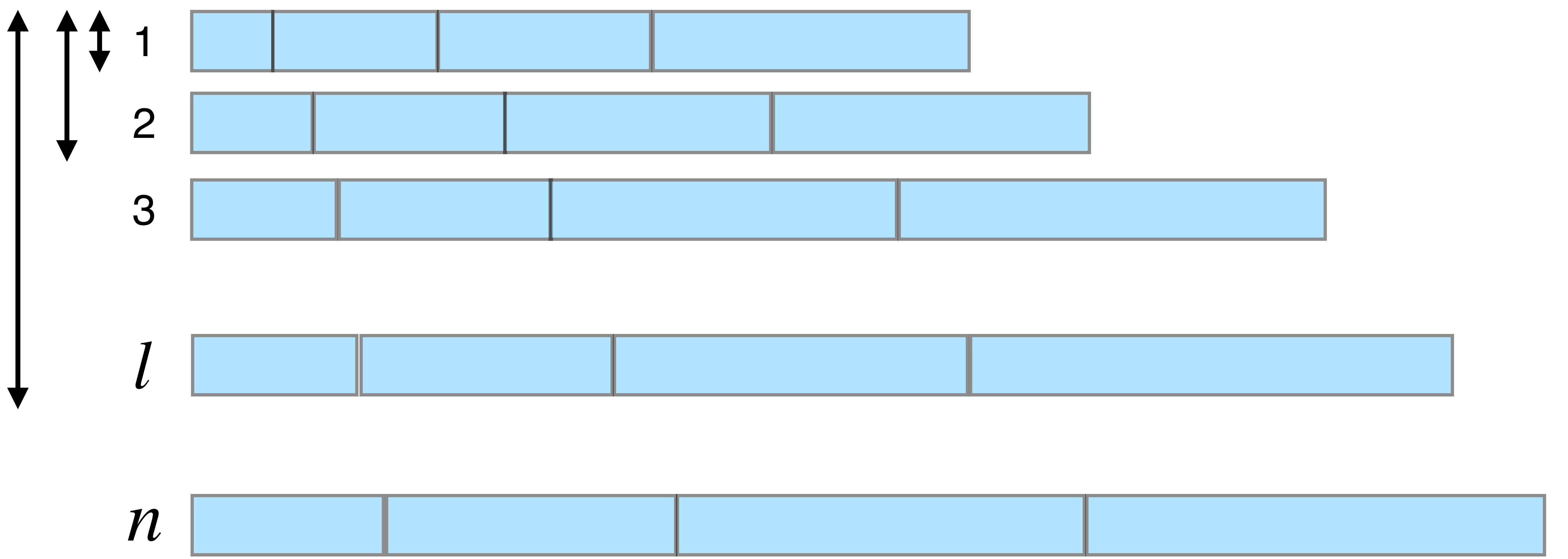
index of best schedule : l

Some details in choosing the schedule



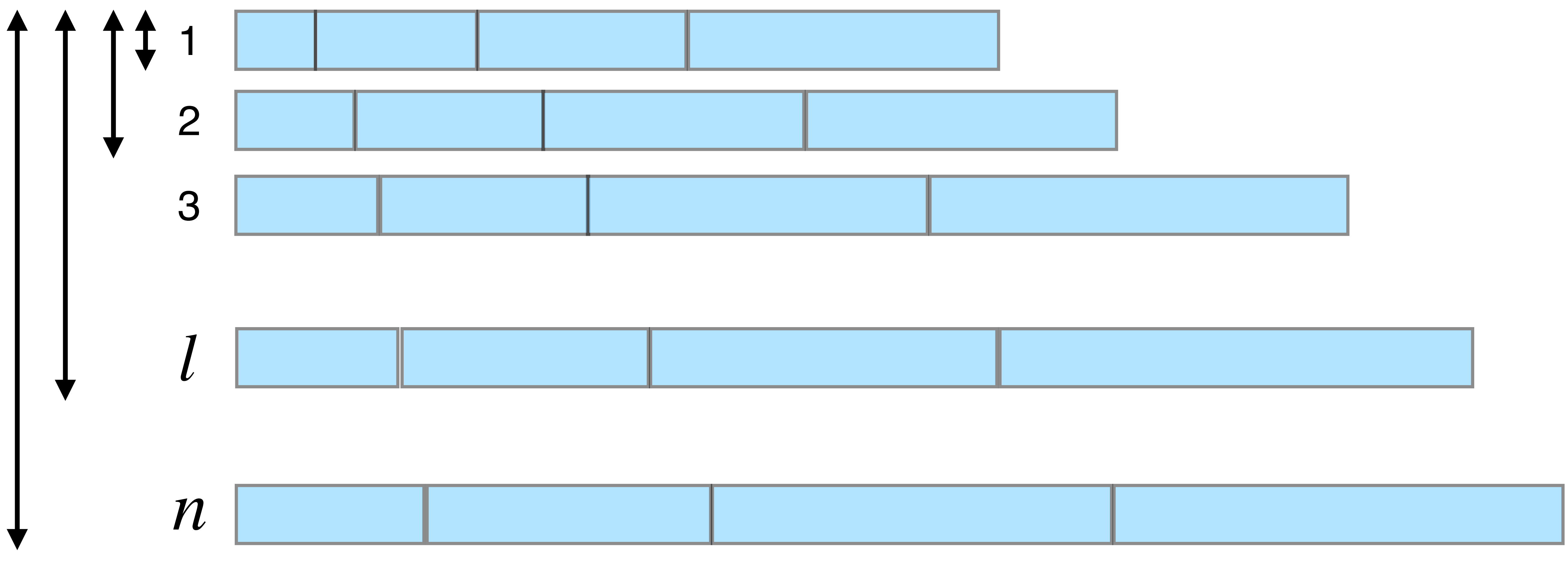
index of best schedule : l

Some details in choosing the schedule



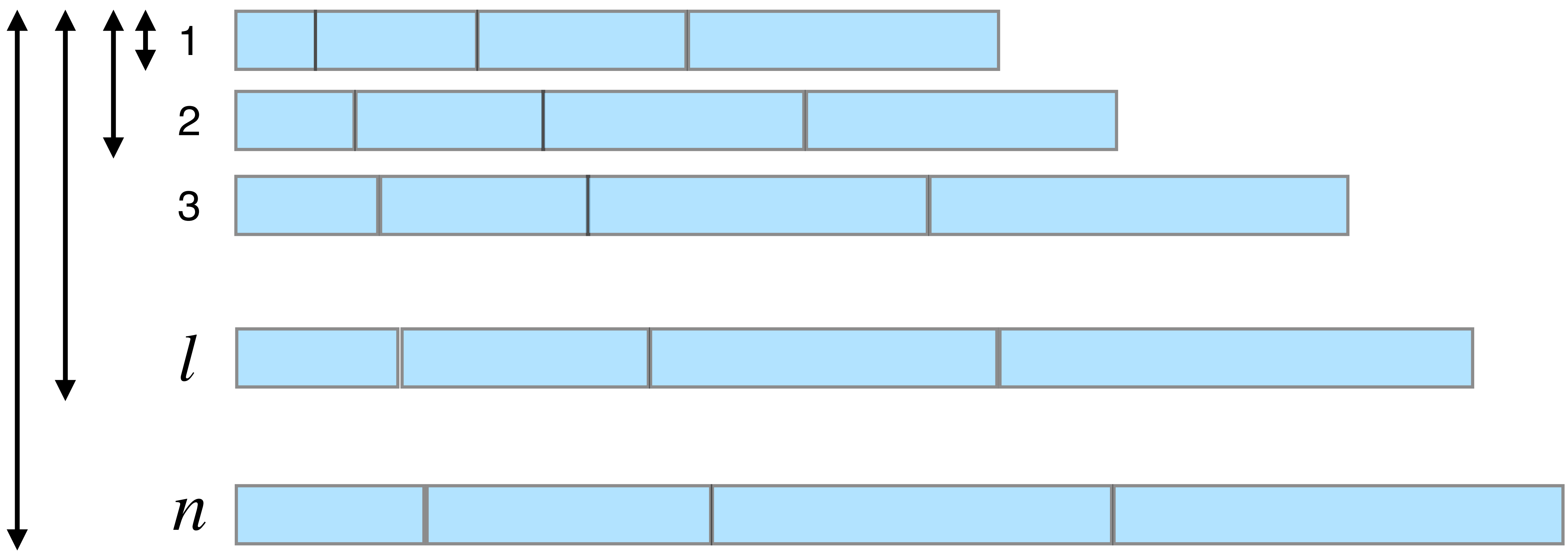
index of best schedule : l

Some details in choosing the schedule



index of best schedule : l

Some details in choosing the schedule



index of best schedule : l

index of chosen schedule : $N + 1 - p \cdot n$

N: number of ``no'' responses

p: tolerance (buffer)

Queries can be made “natural”

Queries can be made “natural”

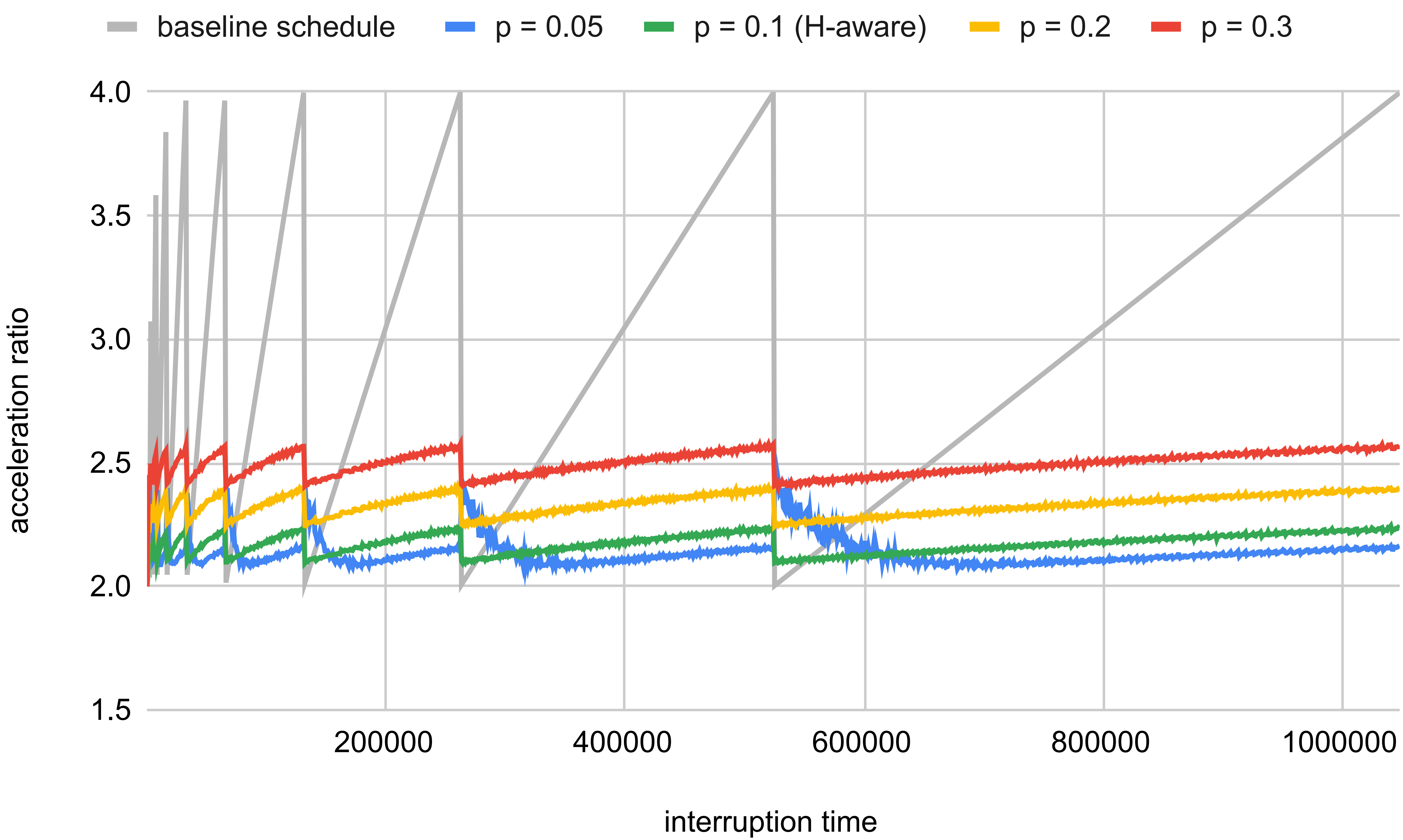
- In the previous discussion, queries are not very intuitive….
- … but we can interpret each query as **a partition of the timeline**

Queries can be made “natural”

- In the previous discussion, queries are not very intuitive….
- … but we can interpret each query as **a partition of the timeline**

“Does the interruption occur in the red partition on in the orange partition?”

Experimental results for r=4, n=100, H=0.1



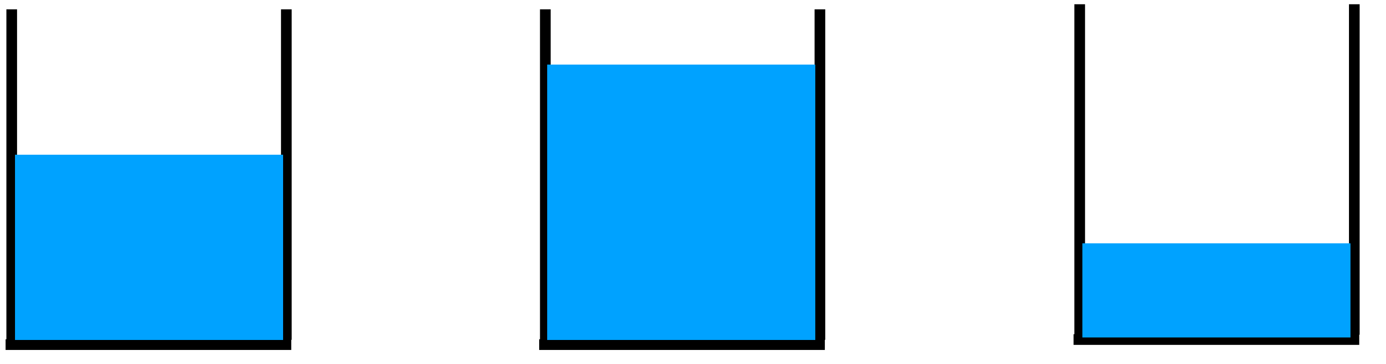
Part 3 : Online bin packing with predictions

Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity

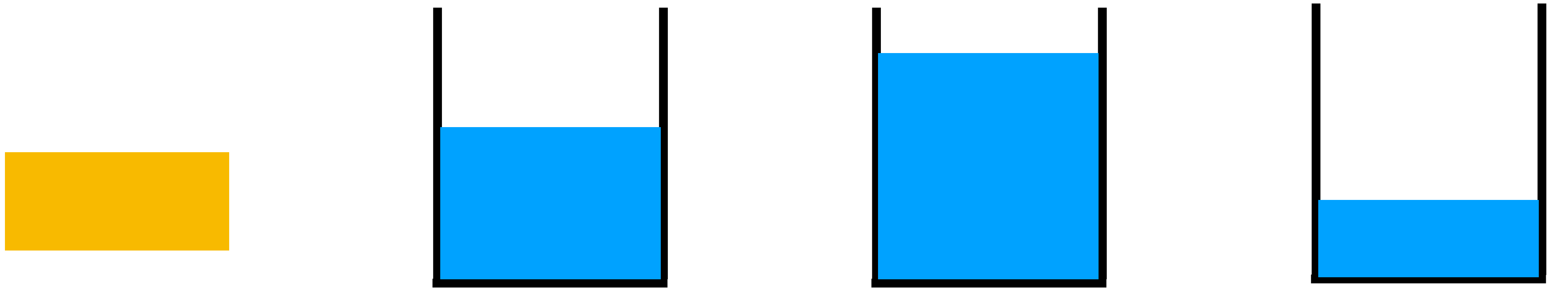
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



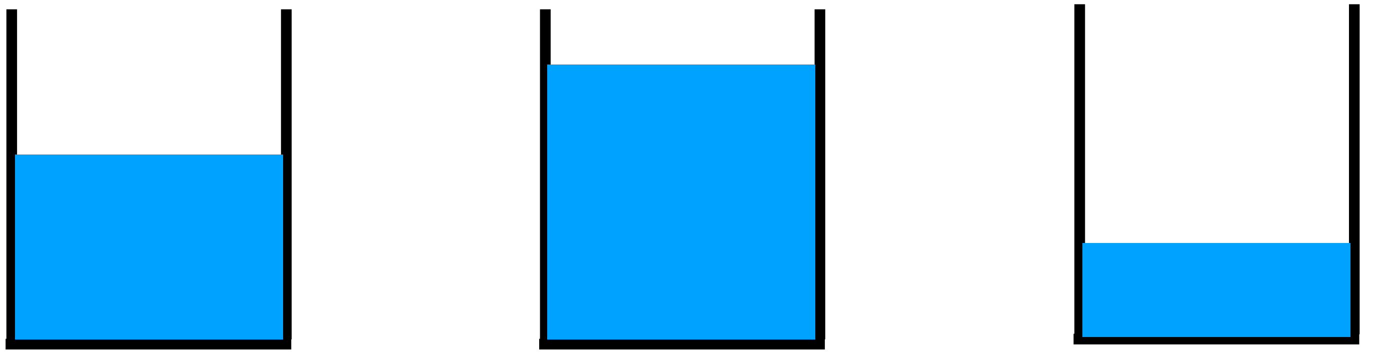
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



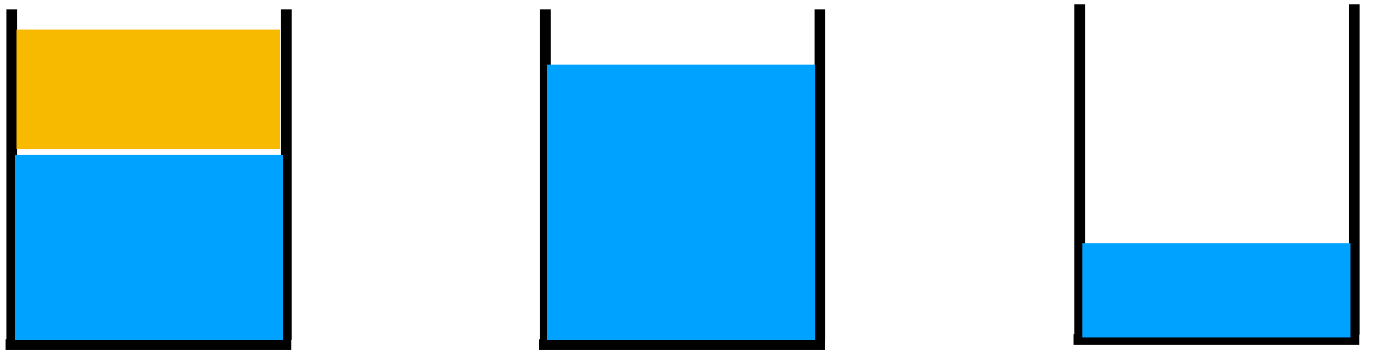
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



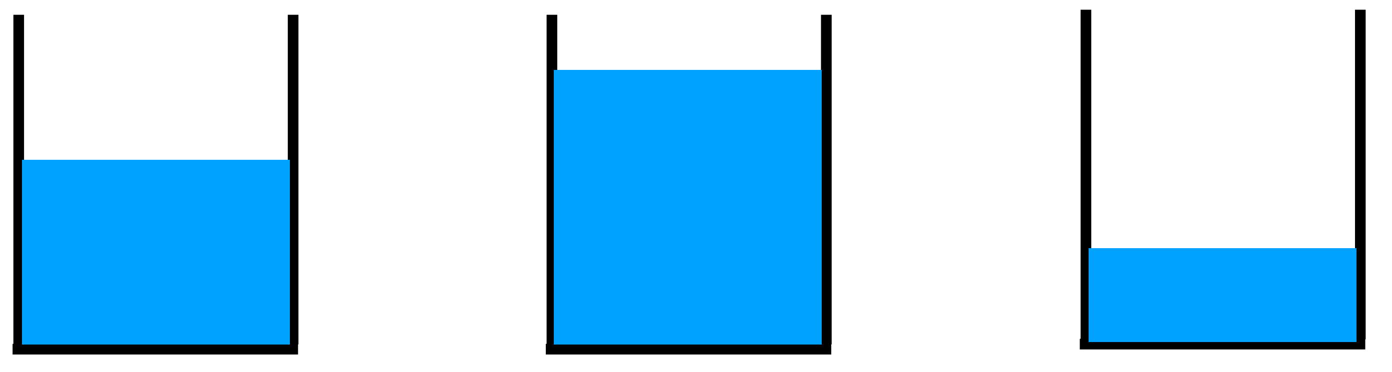
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



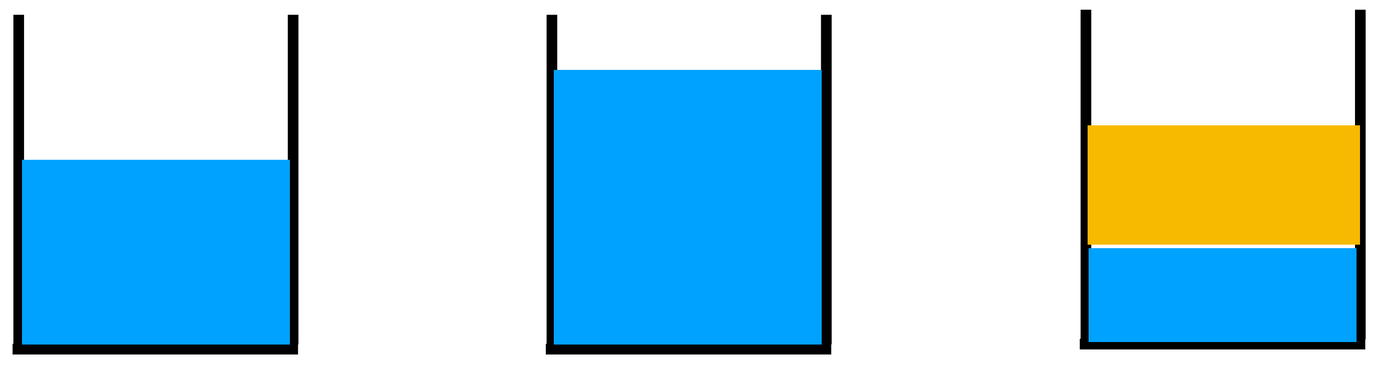
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



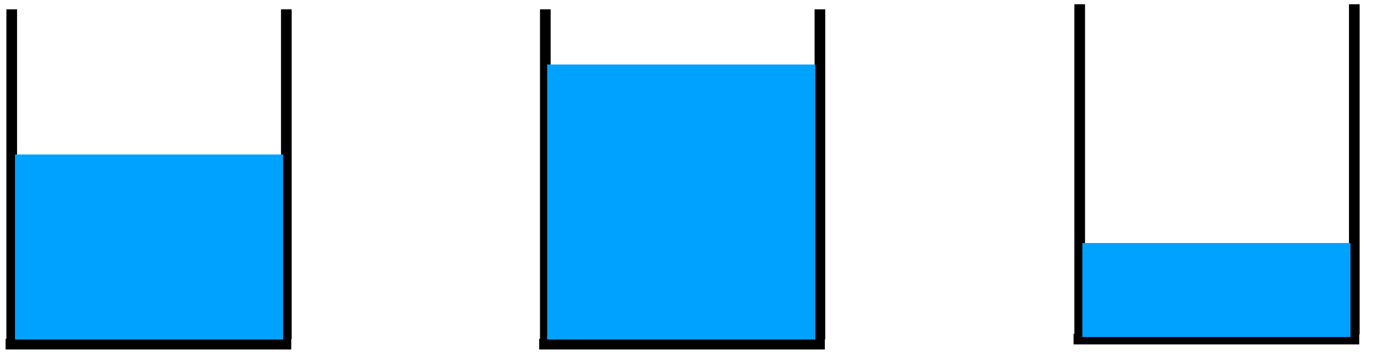
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



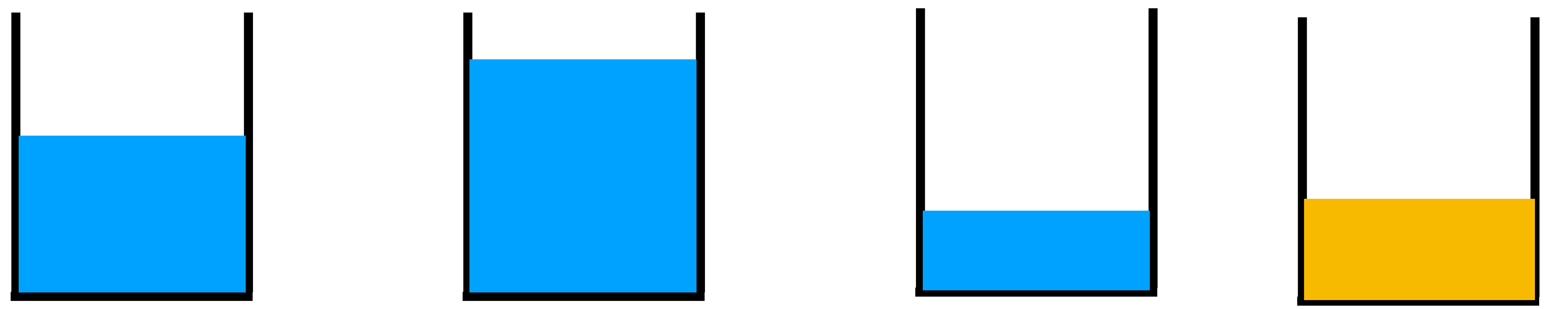
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



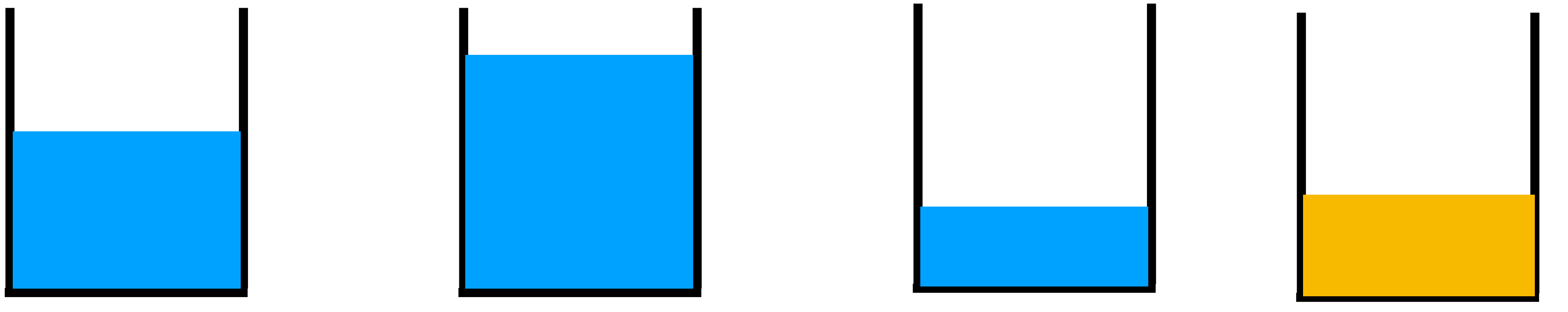
Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



Problem description

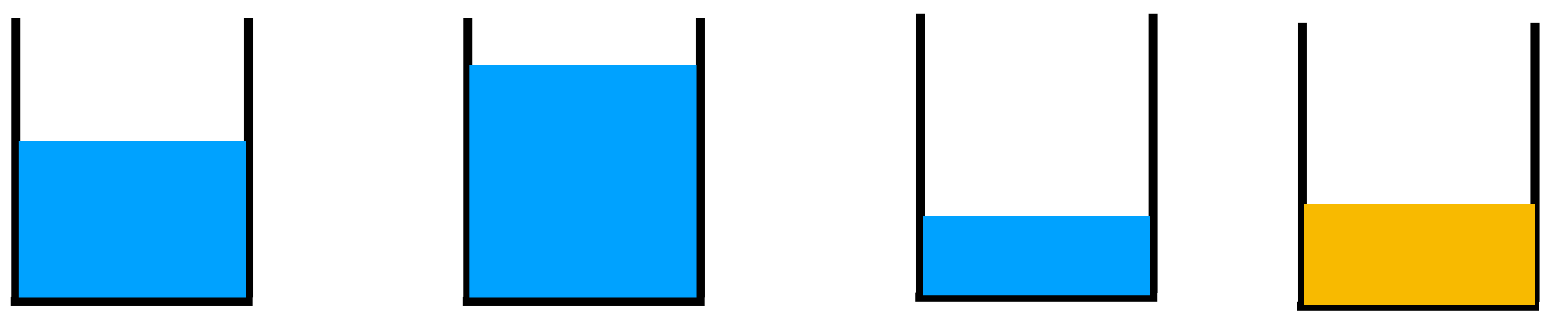
Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



Online setting: Minimize the (asymptotic) competitive ratio

Problem description

Pack a sequence of items (each with its own weight) into the minimum number of bins of a given capacity



Online setting: Minimize the (asymptotic) competitive ratio

Many applications (from inventory management to cloud computing)

e.g., [Cohen et al : Overcommitment in Cloud Services: Bin Packing with Chance Constraints, *Management Science* 2019]

Some known results

Best known **upper** bound : 1.57829 [Balogh et al. 2018]

Best known **lower** bound : 1.54037 [Balogh, Békési and Galambos 2012]

FIRST-FIT, BEST-FIT have competitive ratio 1.7. [Johnson et al. 1974]

Some known results

Best known **upper** bound : 1.57829 [Balogh et al. 2018]

Best known **lower** bound : 1.54037 [Balogh, Békési and Galambos 2012]

FIRST-FIT, BEST-FIT have competitive ratio 1.7. [Johnson et al. 1974]

In practice, FIRST-FIT and BEST-FIT perform very well

In practice, many competitively efficient algorithms do not perform as well as FIRST-FIT

Bin packing with predictions

Bin packing with predictions

We assume a *discrete* model: The bin capacity is a constant k, and each item has integral size in [1,k]

Bin packing with predictions

We assume a *discrete* model: The bin capacity is a constant k, and each item has integral size in [1,k]

Prediction: *Frequencies* at which the items are requested in the sequence

Formally: for each size $x \in [1,k]$, the *frequency* $f_{x,\sigma}$ of x in the sequence σ is the number of items of size x in σ divided by the size of σ

Bin packing with predictions

We assume a *discrete* model: The bin capacity is a constant k, and each item has integral size in [1,k]

Prediction: *Frequencies* at which the items are requested in the sequence

Formally: for each size $x \in [1,k]$, the *frequency* $f_{x,\sigma}$ of x in the sequence σ is the number of items of size x in σ divided by the size of σ

Prediction error: L_1 distance between the actual and the predicted frequencies

Profile Packing

Profile Packing

Fix a (large) constant M. We call the multiset that consists of $\lceil f_{x,\sigma} \cdot M \rceil$ items of size x the **profile** of σ

We can compute the optimal packing of this profile set in $O(1)$ time

Profile Packing

Fix a (large) constant M. We call the multiset that consists of $\lceil f_{x,\sigma} \cdot M \rceil$ items of size x the **profile** of σ

We can compute the optimal packing of this profile set in $O(1)$ time

Example: M=12, $k = 3, f_1 = 0.7, f_2 = 0.2, f_3 = 0.1$

Profile consists of 9 items of size 1, 3 items of size 2 and 2 items of size 3

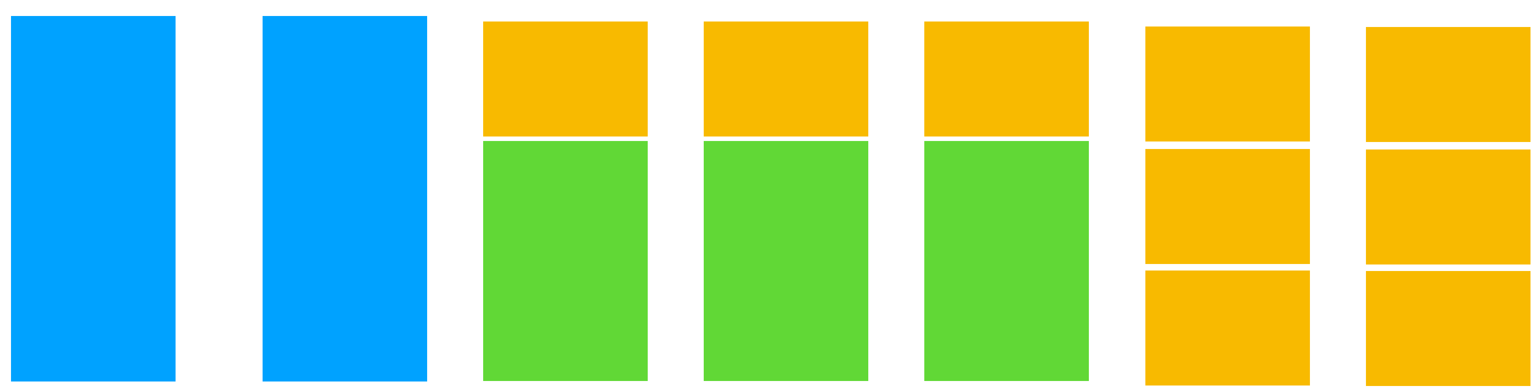
Profile Packing

Fix a (large) constant M. We call the multiset that consists of $\lceil f_{x,\sigma} \cdot M \rceil$ items of size x the **profile** of σ

We can compute the optimal packing of this profile set in $O(1)$ time

Example: M=12, $k = 3, f_1 = 0.7, f_2 = 0.2, f_3 = 0.1$

Profile consists of 9 items of size 1, 3 items of size 2 and 2 items of size 3



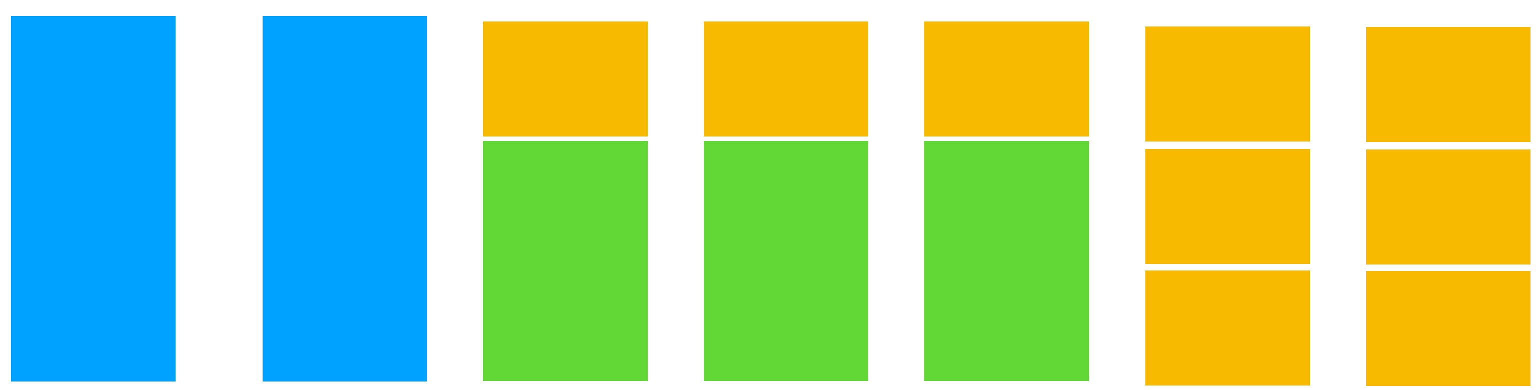
Profile Packing

Fix a (large) constant M. We call the multiset that consists of $\lceil f_{x,\sigma} \cdot M \rceil$ items of size x the **profile** of σ

We can compute the optimal packing of this profile set in $O(1)$ time

Example: M=12, $k = 3, f_1 = 0.7, f_2 = 0.2, f_3 = 0.1$

Profile consists of 9 items of size 1, 3 items of size 2 and 2 items of size 3



Profile Packing: A natural online algorithm based on this concept

Results

Results

Theorem: Profile packing has competitive ratio arbitrarily close to $1 + 2\eta k$

(excellent consistency, bad robustness)

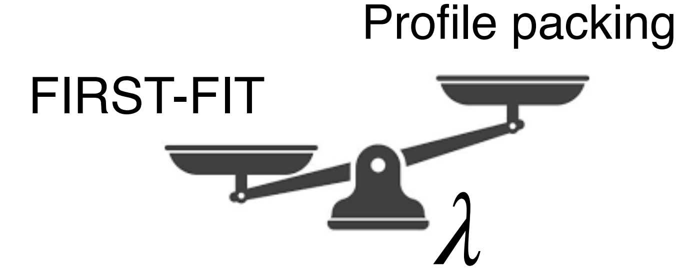
Results

Theorem: Profile packing has competitive ratio arbitrarily close to $1 + 2\eta k$

(excellent consistency, bad robustness)

We propose an algorithm that offers a much better balance, which we call **HYBRID**(λ)**,** where $\lambda \in [0,1]$ is a parameter chosen by the user

Main idea: Some items are served using FIRST-FIT, others using Profile Packing



Results

Theorem: Profile packing has competitive ratio arbitrarily close to $1 + 2\eta k$

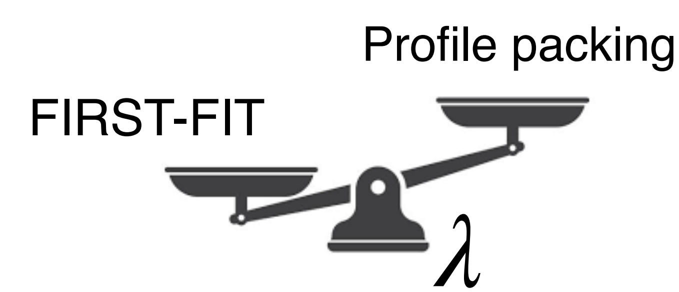
(excellent consistency, bad robustness)

We propose an algorithm that offers a much better balance, which we call **HYBRID**(λ),

where $\lambda \in [0,1]$ is a parameter chosen by the user

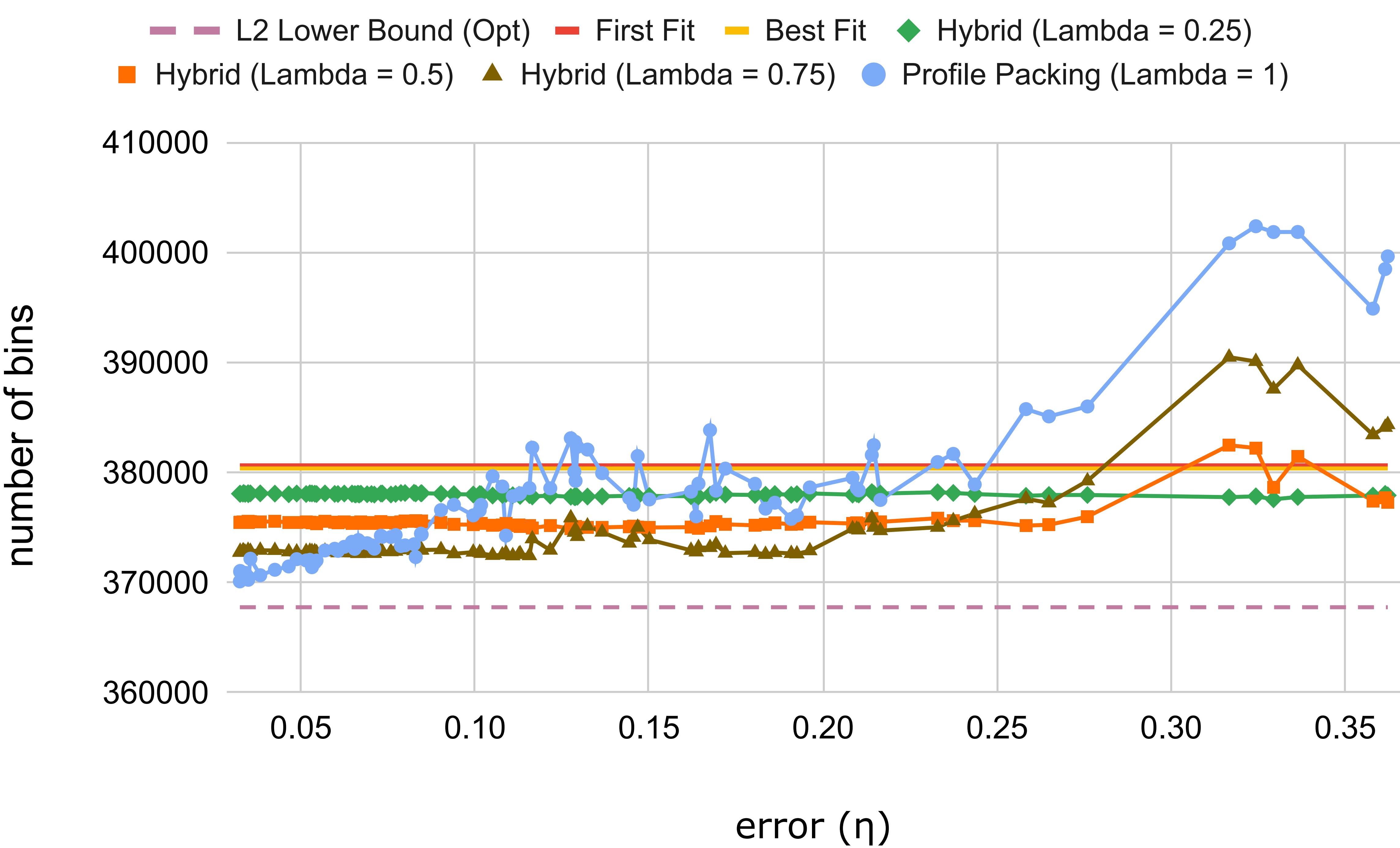
Main idea: Some items are served using FIRST-FIT,

others using Profile Packing



Theorem: HYBRID(λ) has competitive ratio arbitrarily close to $1.7 + \lambda(2\eta k - 0.7)$

Experimental evaluation (Weibull distribution)



Future work

- Bridge the gaps between the upper and the lower bounds for online search and contract scheduling (the upper bounds are likely tight)
- Challenge: information-theoretic lower bounds in the presence of errors
- Analysis beyond the competitive ratio (e.g. search optimization problems)
- Learning aspects of predictions

Potential PhD and postdoc opportunities

- Likely to have PhD opening on this topic (or more broadly on online computation) in 2021
- Potential postdoc opening for 2022

Potential PhD and postdoc opportunities

- Likely to have PhD opening on this topic (or more broadly on online computation) in 2021

- Potential postdoc opening for 2022

Thank you!