Learning-Augmented Bidding in Stochastic Settings

Spyros Angelopoulos Bertrand SIMON

CNRS & ILLS Montréal CNRS & LIG

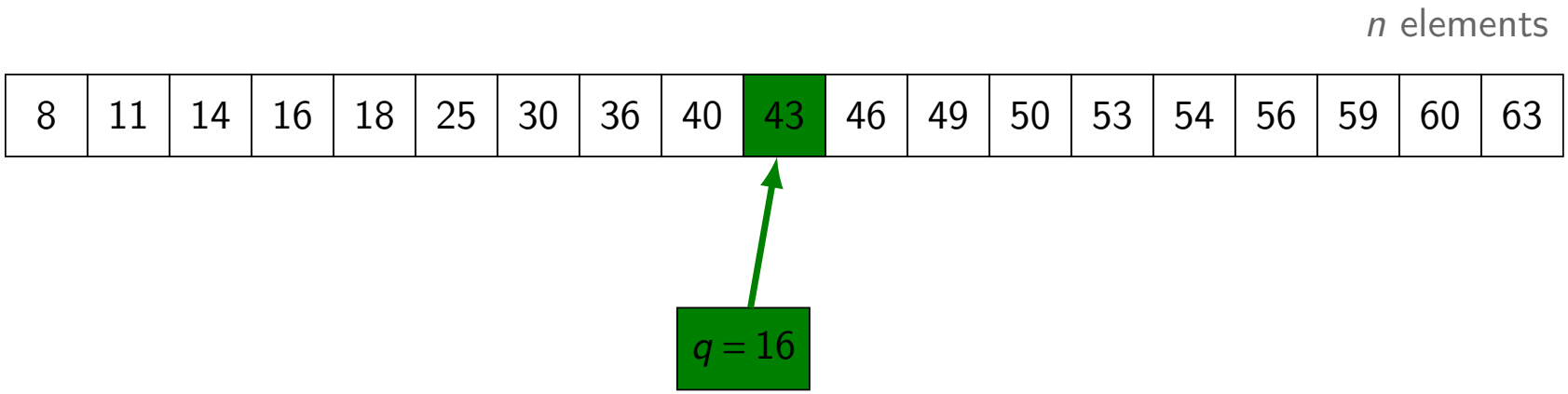
Algorithm with Predictions example : binary search

n elements

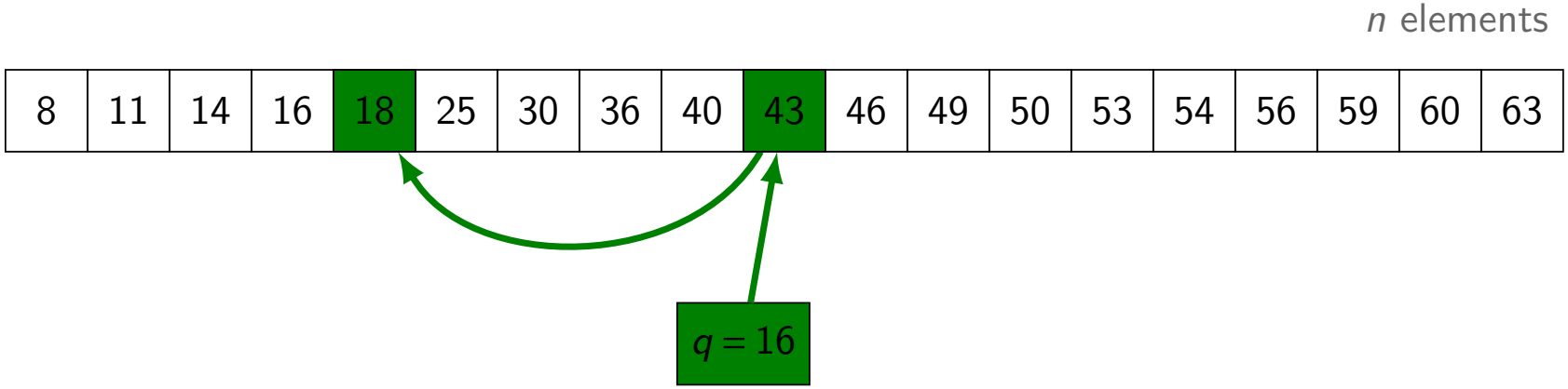
8	11	14	16	18	25	30	36	40	43	46	49	50	53	54	56	59	60	63

$q = 16$

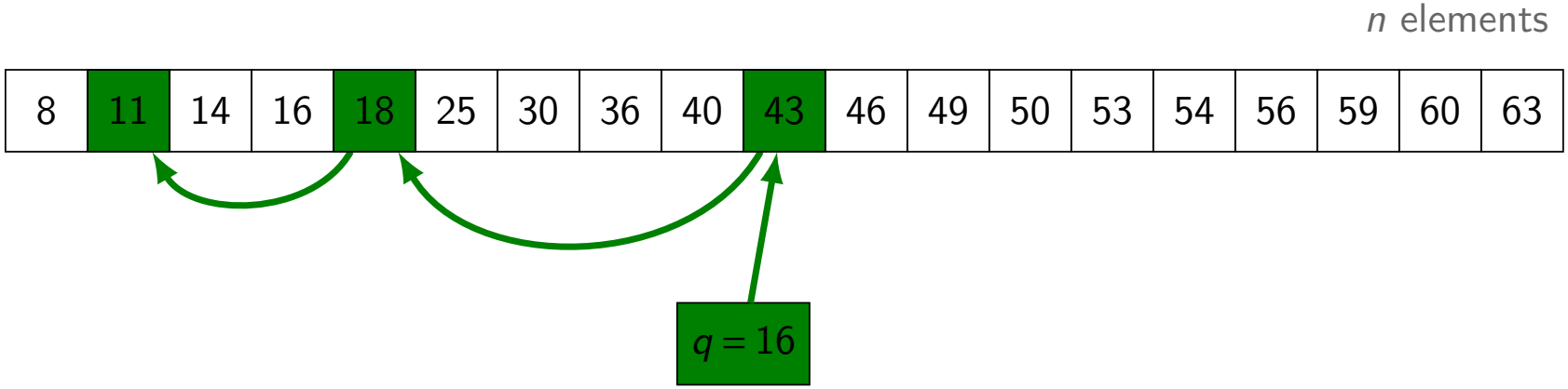
Algorithm with Predictions example : binary search



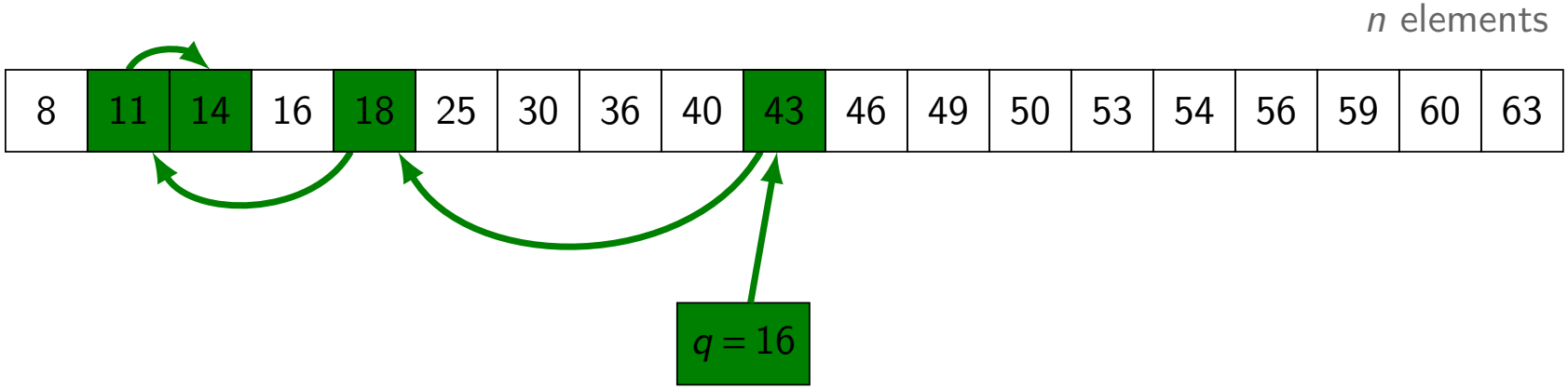
Algorithm with Predictions example : binary search



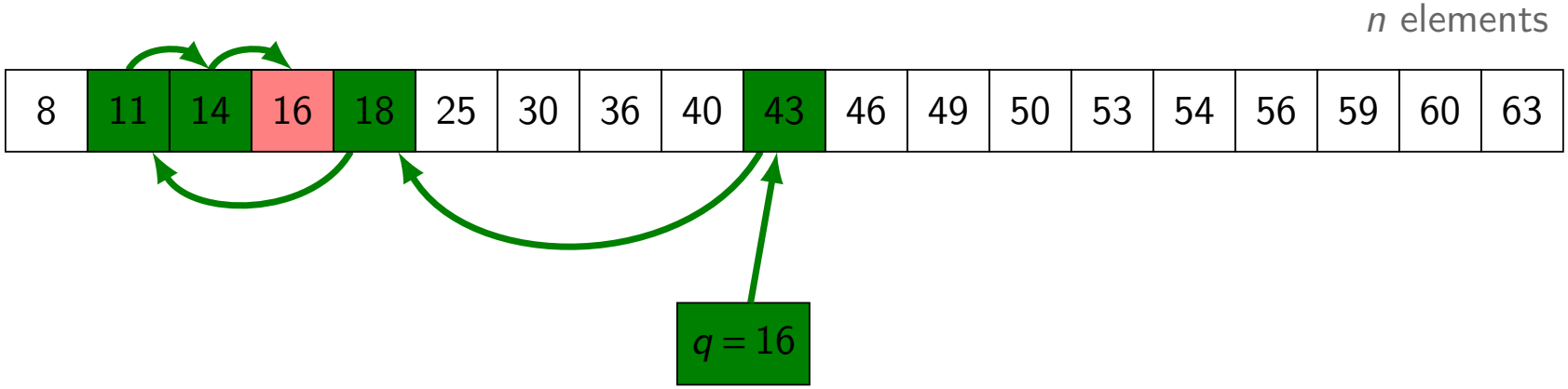
Algorithm with Predictions example : binary search



Algorithm with Predictions example : binary search



Algorithm with Predictions example : binary search



Algorithm with Predictions example : binary search

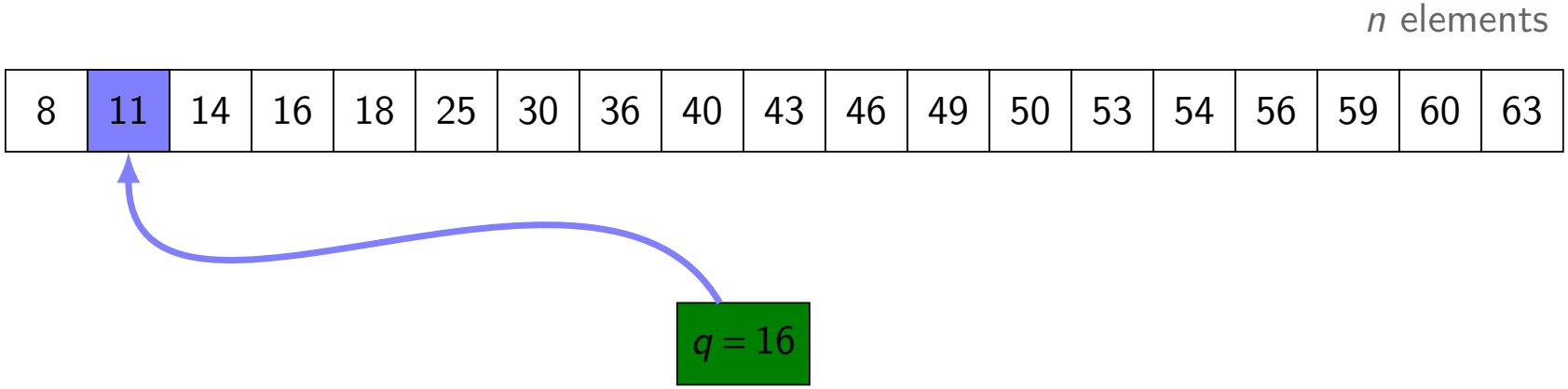
n elements

8	11	14	16	18	25	30	36	40	43	46	49	50	53	54	56	59	60	63

$q = 16$

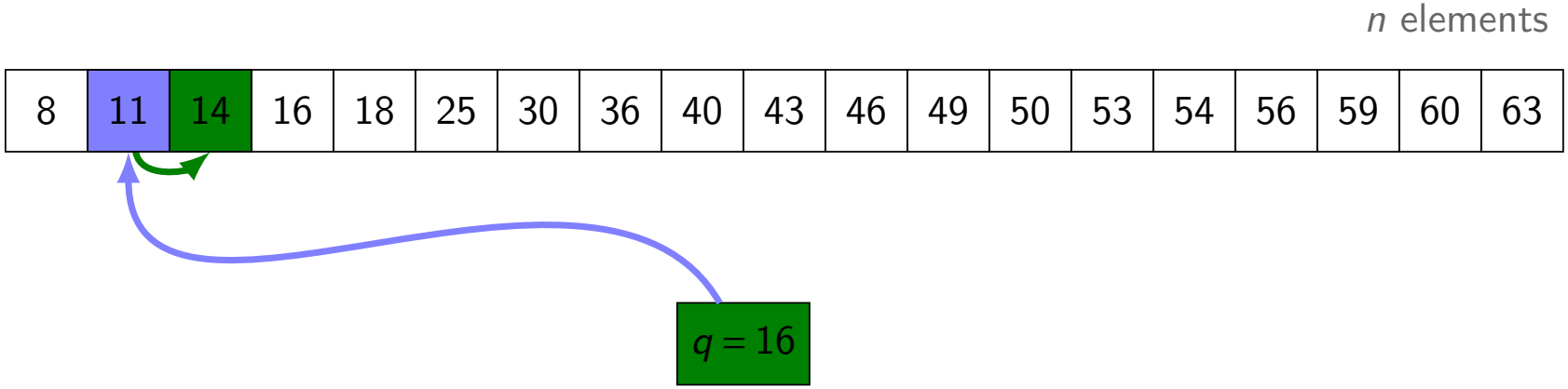
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



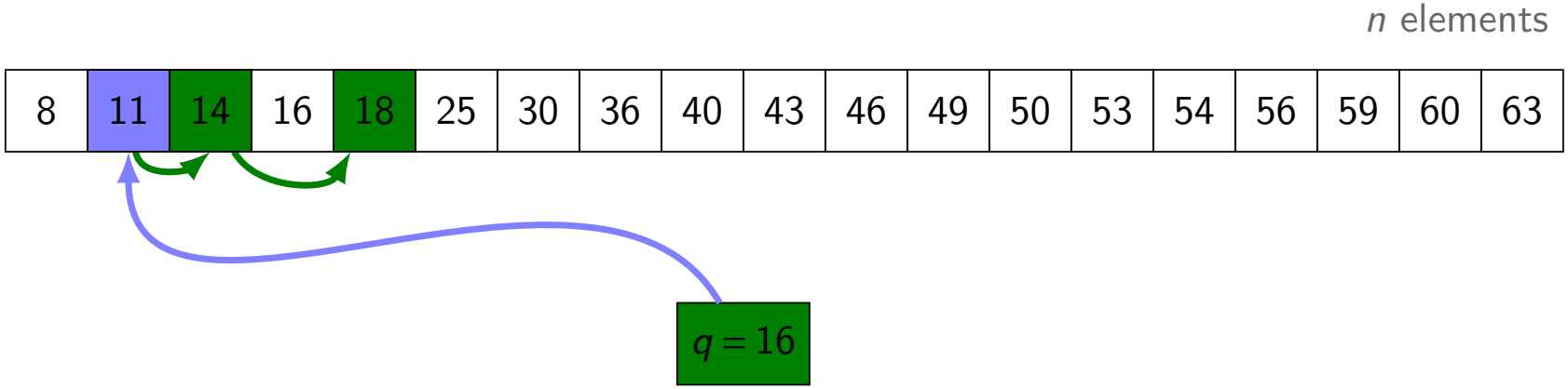
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



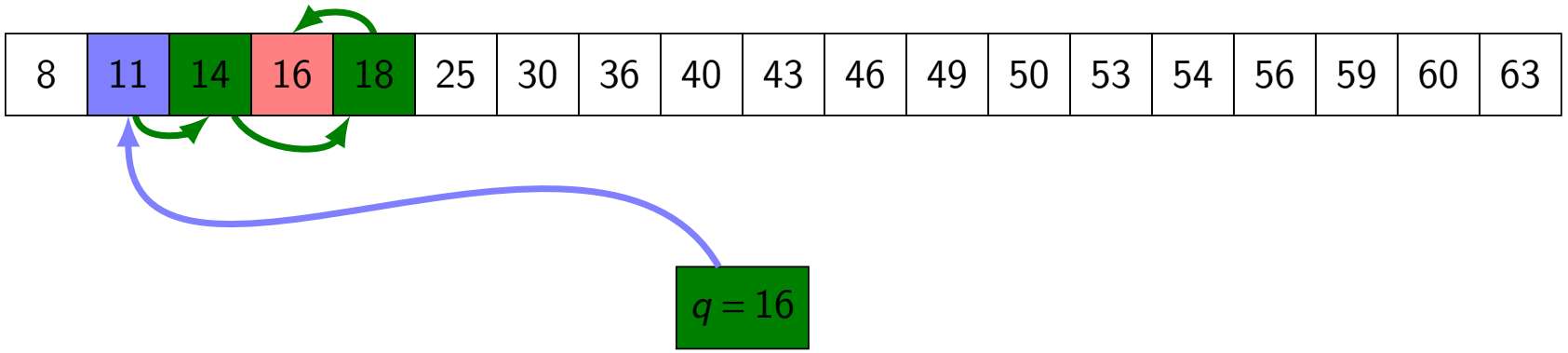
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



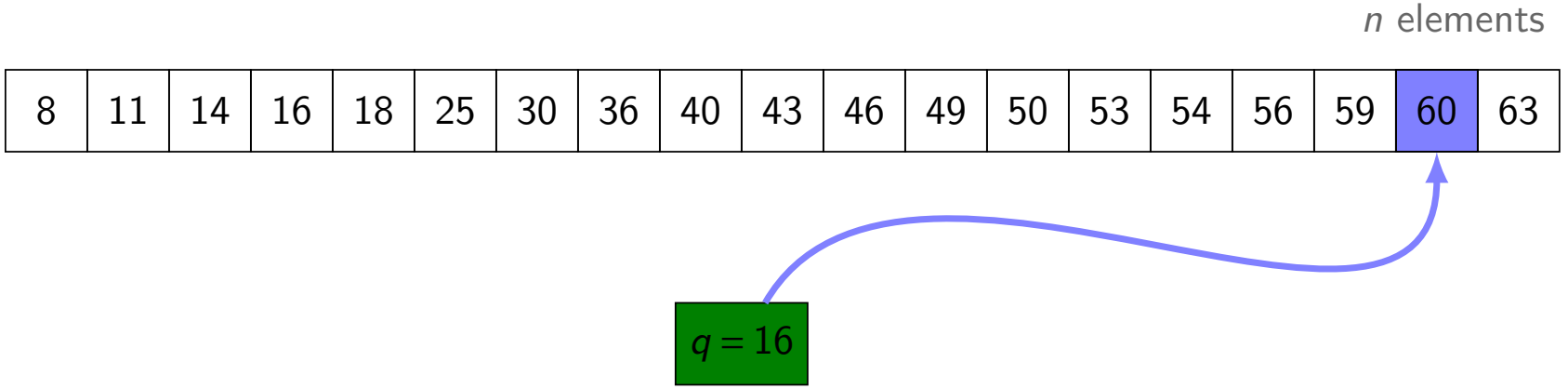
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



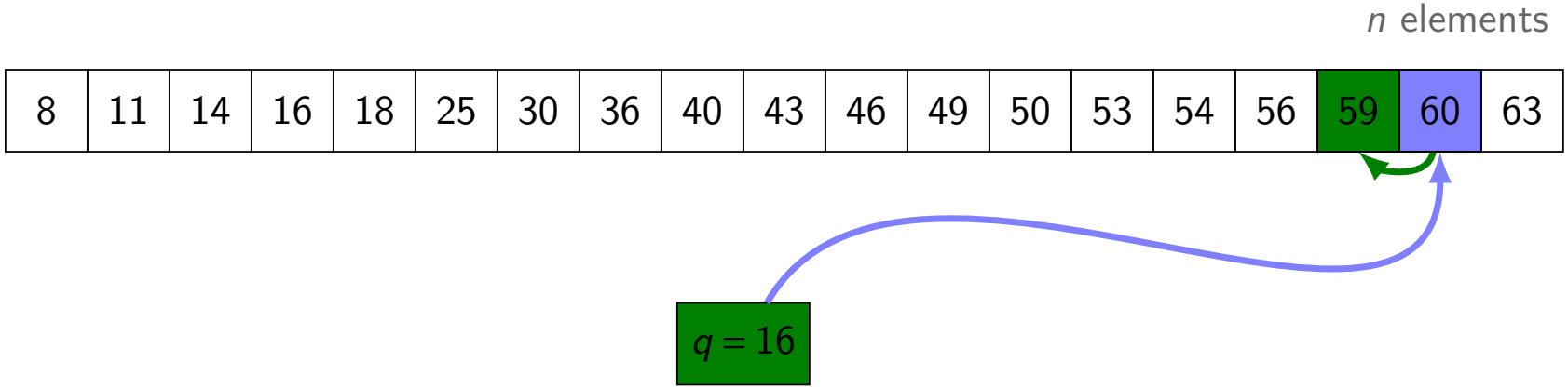
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



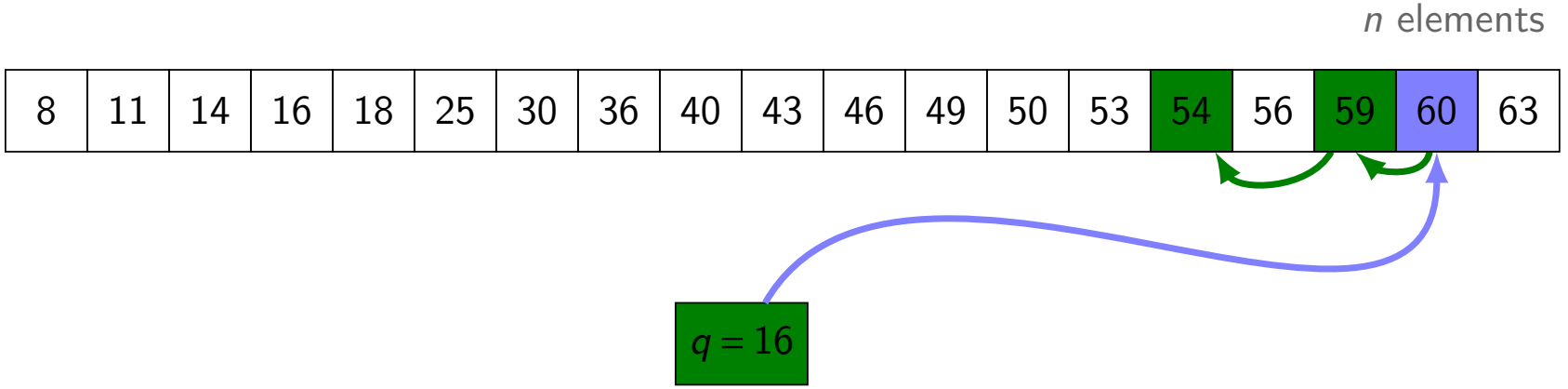
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



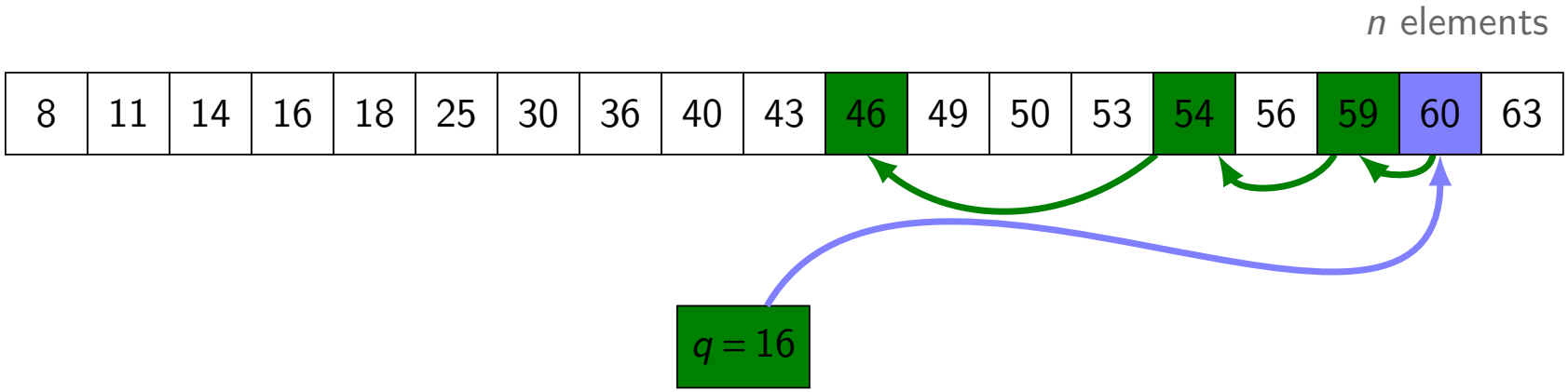
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



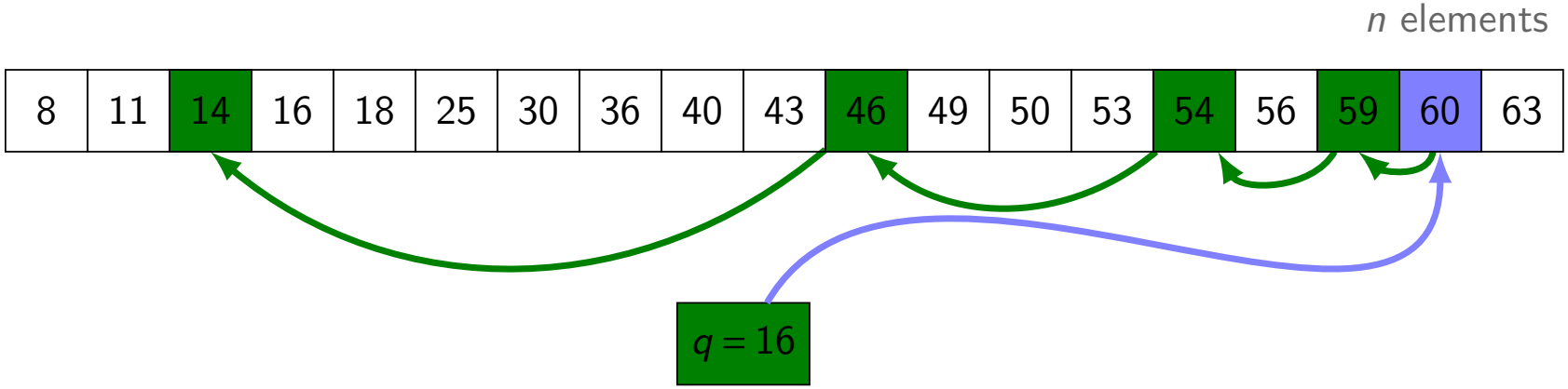
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



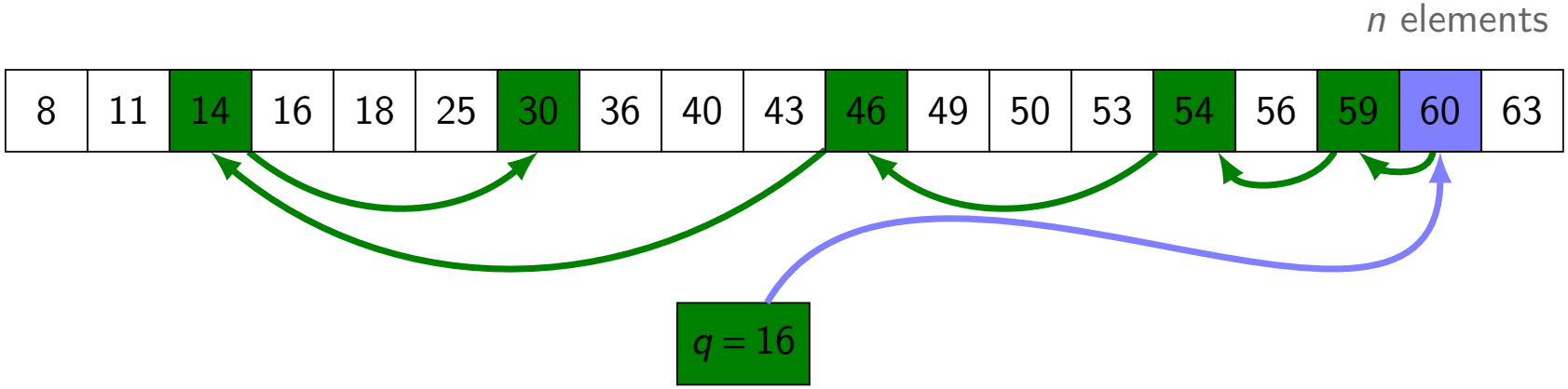
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



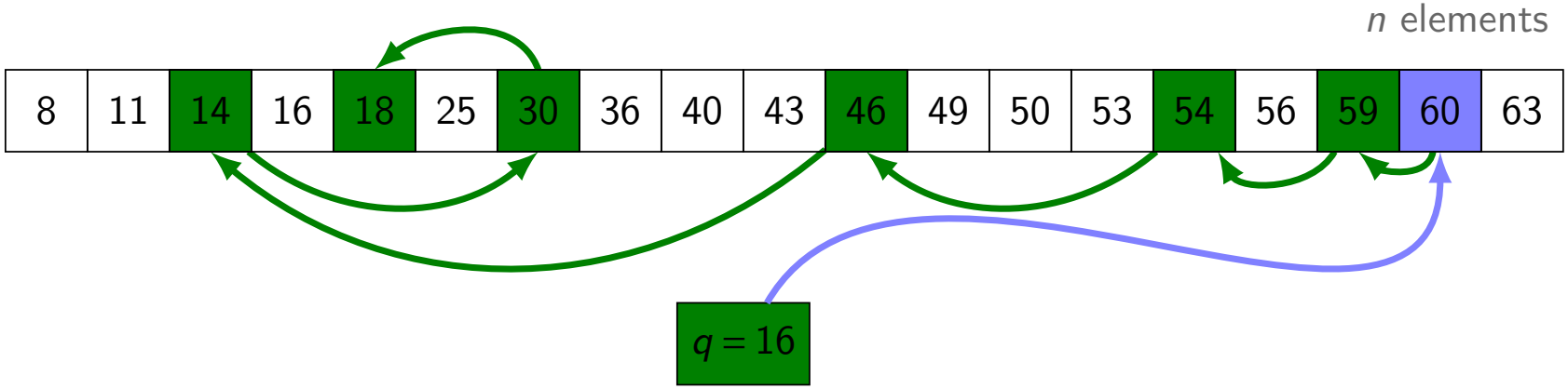
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



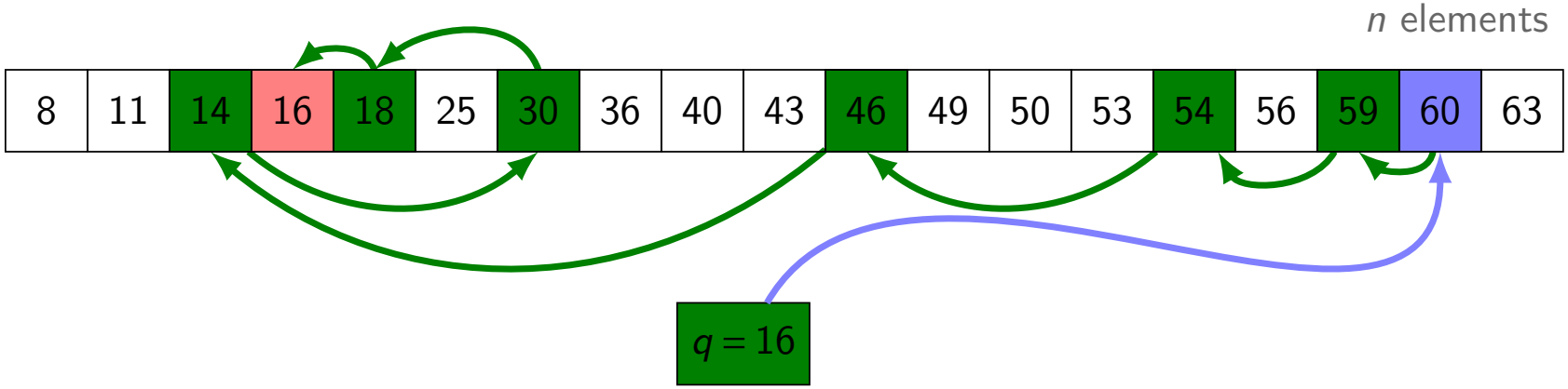
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



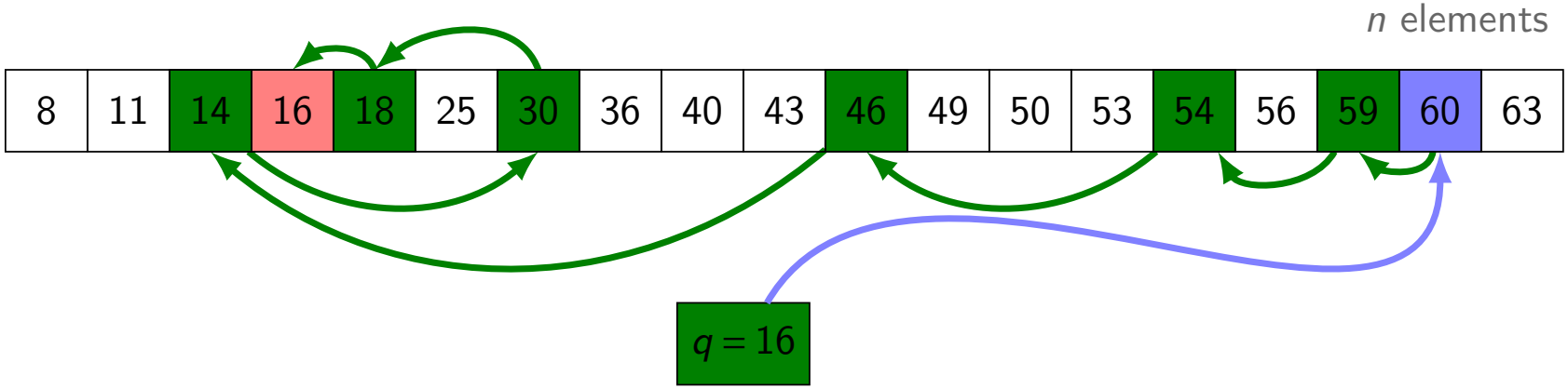
Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search

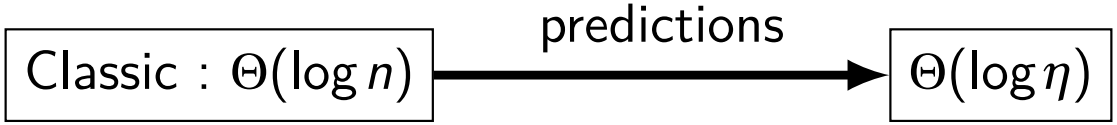


Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$

Algorithm with Predictions example : binary search



Prediction : position $h(q)$ Error : $\eta = |h(q) - index(q)|$



Practical applications [KraskaBCDP'18] – 400+ papers since 2020

Our work in a nutshell [Angelopoulos, Simon, NeurIPS 25]

- Learning-augmented algorithms for a classic online resource allocation problem
- **Stochastic** settings
    - Distributional prediction oracles
    - Randomized online algorithms
- **Upper** and **lower** bounds on performance metrics
- Theoretical / experimental evaluation

Online bidding

Online bidding

Online bidding

Online bidding

u

Online bidding

x_0

u

Online bidding

x_0

x_1

u

Online bidding

u

x_0

x_1

$\vdots$

$x_j : x_{j-1} < u \leq x_j$

Online bidding

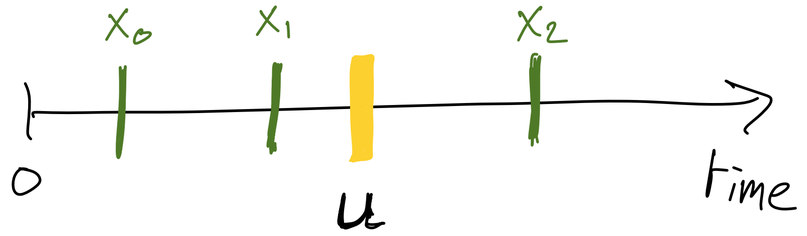
u

x_0

x_1

$\vdots$

$x_j : x_{j-1} < u \leq x_j$



Online bidding

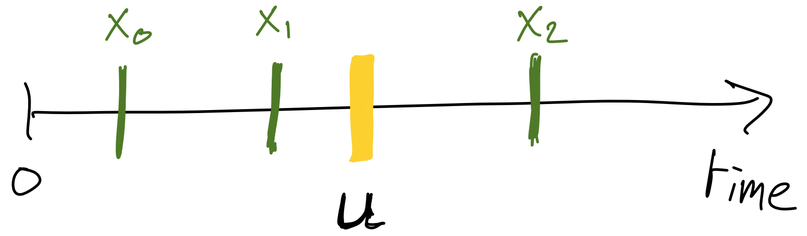
u

x_0

x_1

$\vdots$

$x_j : x_{j-1} < u \leq x_j$



Competitive ratio of $X : (x_0, x_1, \ldots)$ $= \sup_u \frac{\text{cost}(X, u)}{u} = \sup_u \frac{x_0 + x_1 + \ldots x_j}{u}$

Online bidding

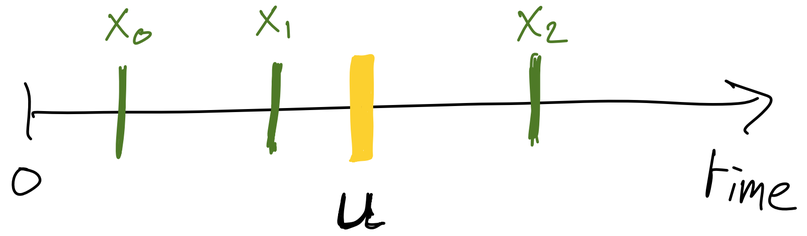
u

x_0

x_1

$\vdots$

$x_j : x_{j-1} < u \leq x_j$



first bid in X that is at least u

Competitive ratio of $X : (x_0, x_1, \ldots)$ $= \sup_u \frac{\text{cost}(X, u)}{u} = \sup_u \frac{x_0 + x_1 + \ldots x_j}{u}$

Online bidding

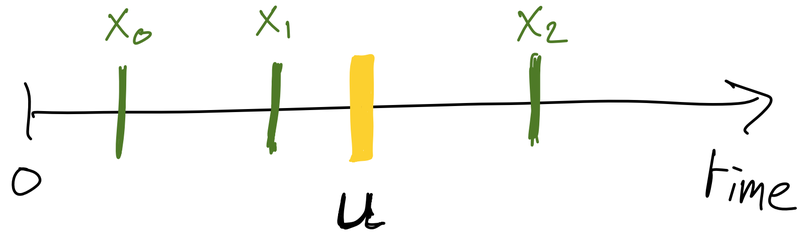
u

x_0

x_1

$\vdots$

$x_j : x_{j-1} < u \leq x_j$



first bid in X that is at least u

Competitive ratio of $X : (x_0, x_1, \ldots)$ $= \sup_u \frac{\text{cost}(X, u)}{u} = \sup_u \frac{x_0 + x_1 + \ldots x_j}{u}$

$$= \sup_{j \geq 1} \frac{\sum_{i=0}^{j} x_i}{x_{j-1}}$$

Previous work : the standard setting

Previous work : the standard setting

- Tight **deterministic** competitive ratio = **4**, using the strategy $(2^i)_{i\geq 0}$

- Tight **randomized** competitive ratio = e, using the strategy $X = (e^{i+s})_{i\geq 0}$, $s \sim U[0,1]$ [Chrobak, Mathieu, Noga, and Young 2008]

Previous work : the standard setting

- Tight **deterministic** competitive ratio = **4**, using the strategy $(2^i)_{i\geq 0}$
- Tight **randomized** competitive ratio = e, using the strategy $X = (e^{i+s})_{i\geq 0}$, $s \sim U[0,1]$ [Chrobak, Mathieu, Noga, and Young 2008]
- Connections to **searching** for a hidden target **on the infinite line**
  - Tight deterministic competitive ratio = 9 [Beck and Newman 1970]
  - Tight randomized competitive ratio ≈ 4.6 [Gal 1980], [Kao, Reif and Tate 2001]

Previous work : the standard setting

- Tight **deterministic** competitive ratio = **4**, using the strategy $(2^i)_{i\geq 0}$
- Tight **randomized** competitive ratio = e, using the strategy $X = (e^{i+s})_{i\geq 0}$, $s \sim U[0,1]$ [Chrobak, Mathieu, Noga, and Young 2008]
- Connections to **searching** for a hidden target **on the infinite line**
  - Tight deterministic competitive ratio = 9 [Beck and Newman 1970]
  - Tight randomized competitive ratio ≈ 4.6 [Gal 1980], [Kao, Reif and Tate 2001]
- Applications in many online and offline problems [Chrobak and Mathieu 2008]

Previous work : the learning-augmented setting

Previous work : the learning-augmented setting

- Point prediction $\hat{u}$ on the real target u, hence error $\eta = |\hat{u} - u|$

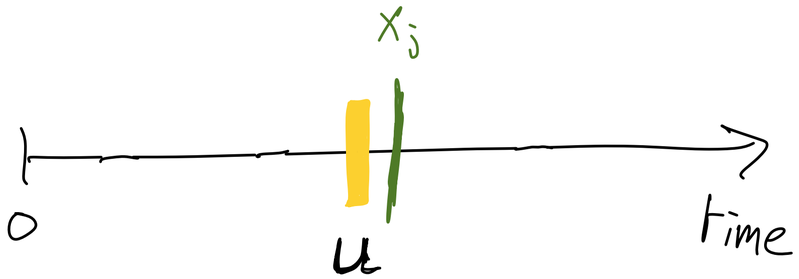
$$\textbf{Consistency} = \sup_{\hat{u}} \frac{\text{cost}(X_{\hat{u}}, \hat{u})}{\hat{u}} \quad \text{whereas} \quad \textbf{robustness} = \sup_{\hat{u},u} \frac{\text{cost}(X_{\hat{u}}, u)}{u}$$

Previous work : the learning-augmented setting

- Point prediction $\hat{u}$ on the real target u, hence error $\eta = |\hat{u} - u|$

$$\textbf{Consistency} = \sup_{\hat{u}} \frac{\text{cost}(X_{\hat{u}}, \hat{u})}{\hat{u}} \quad \text{whereas} \quad \textbf{robustness} = \sup_{\hat{u},u} \frac{\text{cost}(X_{\hat{u}}, u)}{u}$$

- Deterministic Pareto-optimal algorithms: $\frac{b}{b-1}$ - consistency and $\frac{b^2}{b-1}$ robustness, $b > 1$ [Angelopoulos, Durr, Jin, Kamali, Renault 2020]

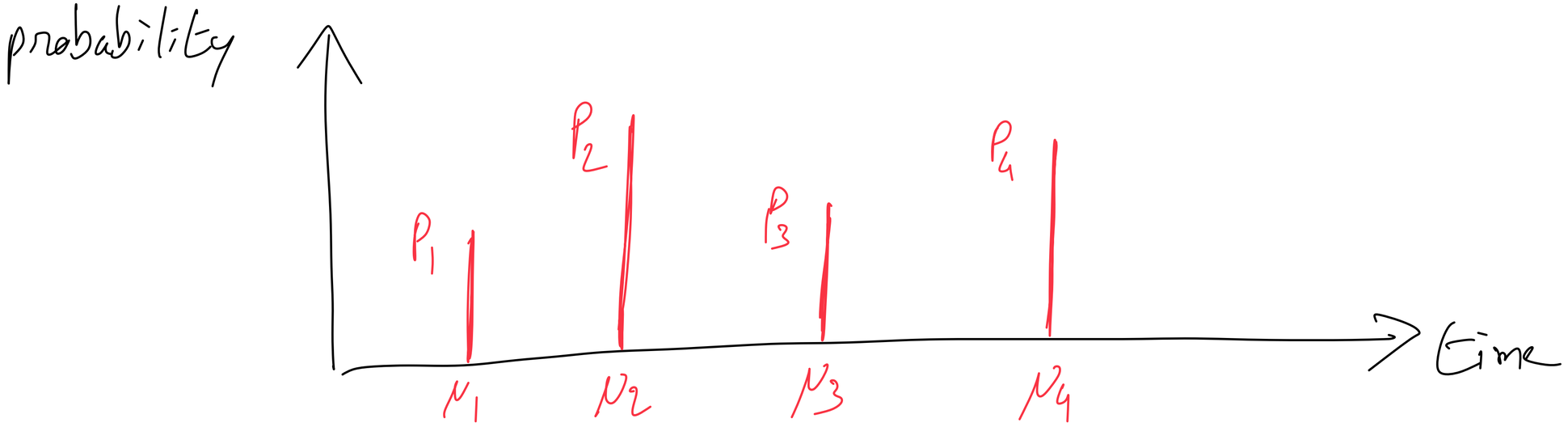


- Pareto-optimal algorithms are **brittle** [Elenter, Angelopoulos, Dürr, and Lefki 2024], tradeoffs between consistency, robustness and smoothness, also in [Benomar, Perchet 2025]

First stochastic setting: Distributional predictions

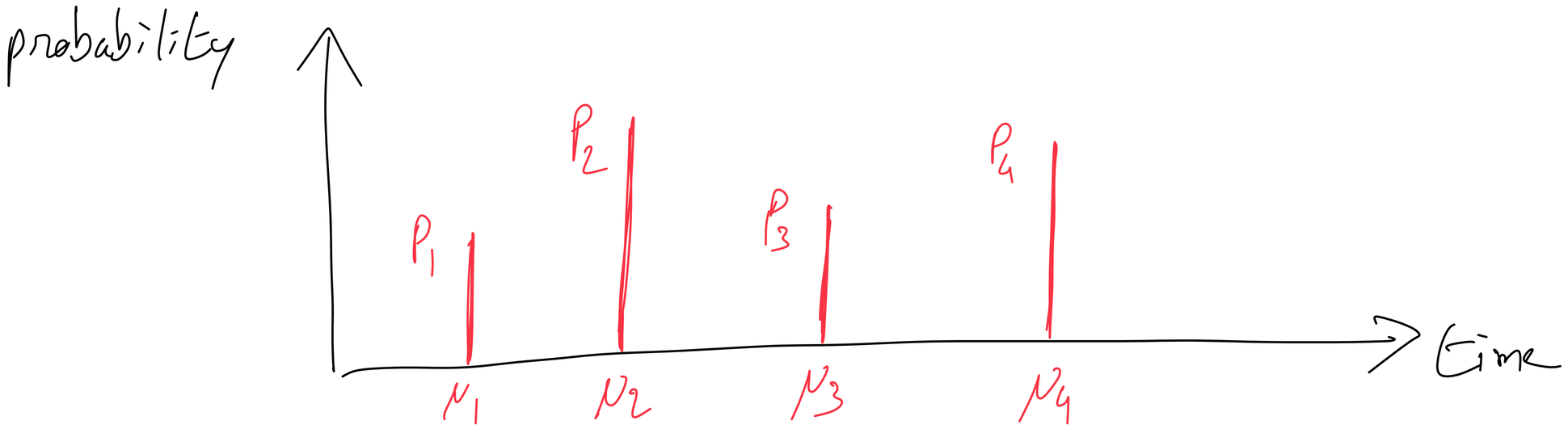
First stochastic setting: Distributional predictions

Model: Oracle provides distributional advice of the form $\mu = \cup_{i=1}^{k} (\mu_i, p_i)$, with $\mu_i \in \mathbb{R}^+$ and $\sum_{i=1}^{k} p_i = 1$



First stochastic setting: Distributional predictions

Model: Oracle provides distributional advice of the form $\mu = \cup_{i=1}^{k} (\mu_i, p_i)$, with $\mu_i \in \mathbb{R}^+$ and $\sum_{i=1}^{k} p_i = 1$

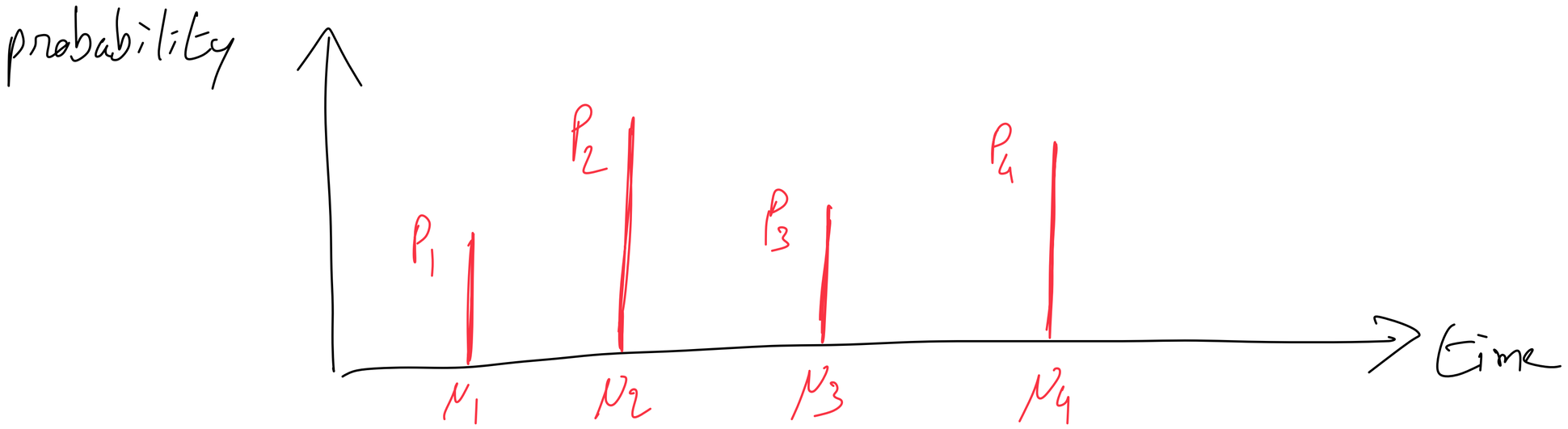


Objective : Given $r \geq 4$ find a deterministic $r-$robust algorithm X that minimizes the consistency wrt μ

$$\mathrm{cons}(X, \mu) = \frac{\mathbb{E}_{z\sim\mu}[\mathrm{cost}(X, z)]}{\mathbb{E}_{z\sim\mu}[z]}$$

First stochastic setting: Distributional predictions

Model: Oracle provides distributional advice of the form $\mu = \cup_{i=1}^{k} (\mu_i, p_i)$, with $\mu_i \in \mathbb{R}^+$ and $\sum_{i=1}^{k} p_i = 1$

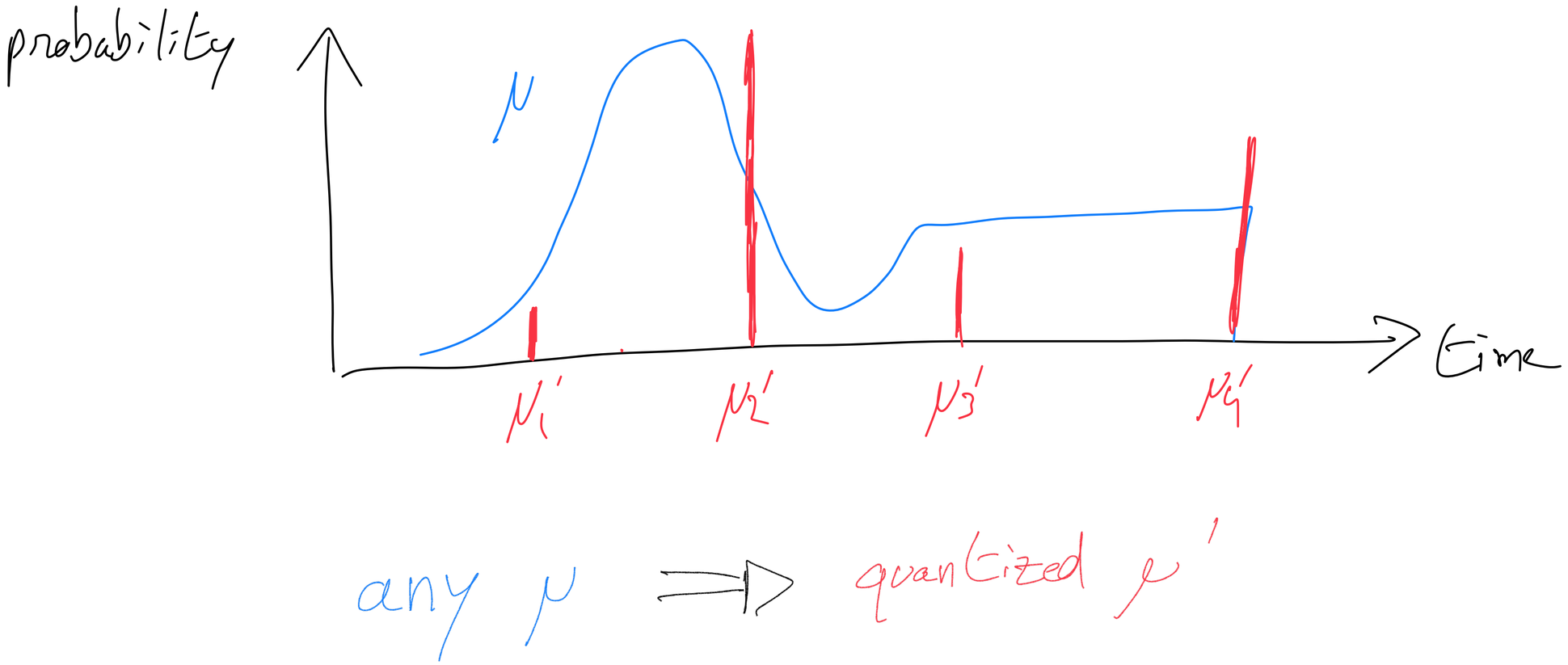


Objective : Given $r \geq 4$ find a deterministic $r-$robust algorithm X that minimizes the consistency wrt μ

$$\mathrm{cons}(X, \mu) = \frac{\mathbb{E}_{z\sim\mu}[\mathrm{cost}(X, z)]}{\mathbb{E}_{z\sim\mu}[z]}$$

Main result: Pareto-optimal algorithm for this setting

Handling general distributions via quantization



any μ ⇒ quantized μ'

Outline of the approach

Challenge: Pareto-optimal strategies may have a non-obvious structure

Theorem: Even if $k = 2$, every geometric strategy $(\lambda \cdot b_r^i)_{i \geq 0}$ has *unbounded consistency* as $r \to \infty$

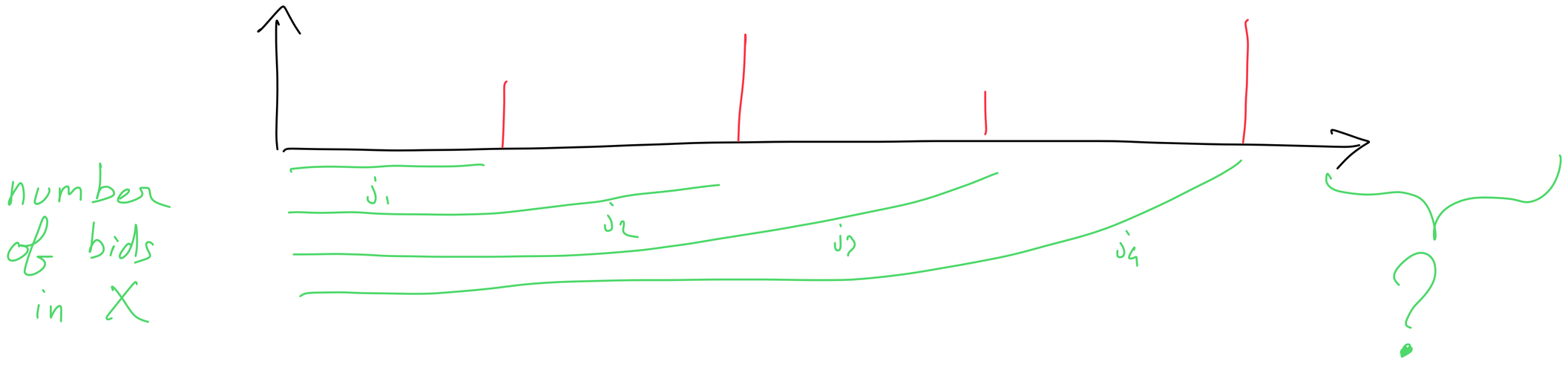
Outline of the approach

Challenge: Pareto-optimal strategies may have a non-obvious structure

Theorem: Even if $k = 2$, every geometric strategy $(\lambda \cdot b_r^i)_{i \geq 0}$ has *unbounded consistency* as $r \to \infty$

Main idea: **LP-approach**

Configuration of X : vector $(j_1, \ldots j_k) \in \mathbb{N}^k$ such that $x_{j_i} < \mu_i \leq x_{j_i+1}$ for all $i \in [1,k]$



Outline of the approach

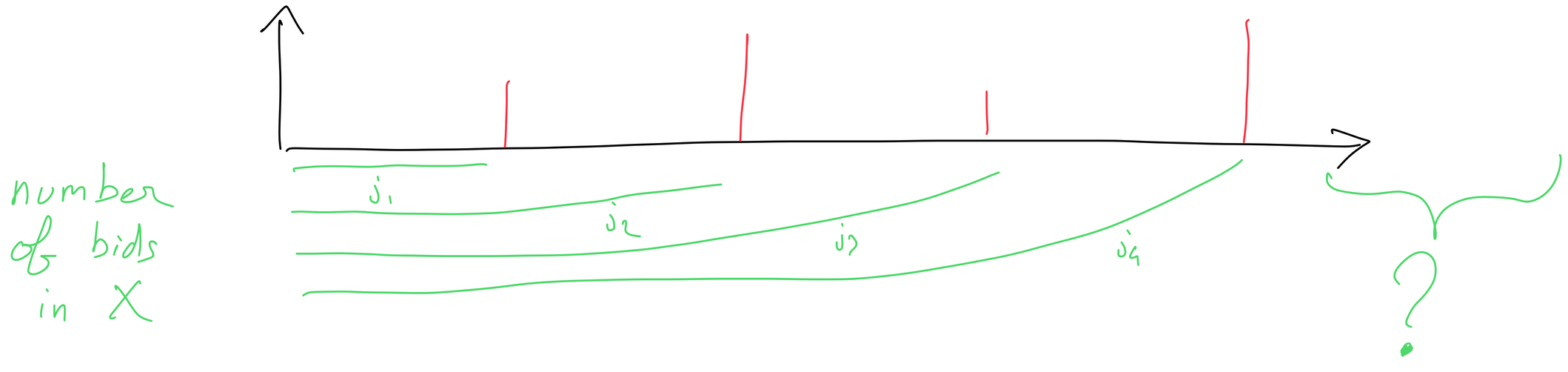
Challenge: Pareto-optimal strategies may have a non-obvious structure

Theorem: Even if $k = 2$, every geometric strategy $(\lambda \cdot b_r^i)_{i \geq 0}$ has *unbounded consistency* as $r \to \infty$

Main idea: **LP-approach**

Configuration of X : vector $(j_1, \ldots j_k) \in \mathbb{N}^k$ such that $x_{j_i} < \mu_i \leq x_{j_i+1}$ for all $i \in [1,k]$

- We can write the objective as: $\mathbb{E}_{u \sim \mu} \text{cost}(X, u) = \sum_{i=1}^{k} p_i \sum_{q=0}^{j_i+1} x_q$



Building the LP (for a given configuration j)

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1,k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0,j_k]$$

monotonicity

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$\text{subj. to} \quad x_i \leq r x_{i-1} - \sum_{j=0}^{i-1} x_j, i \in [1, j_k + 1]$$

robustness “up to the prediction”

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1, k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0, j_k]$$

monotonicity

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$\text{subj. to} \quad x_i \leq r x_{i-1} - \sum_{j=0}^{i-1} x_j, i \in [1, j_k + 1]$$

robustness “up to the prediction”

robustness “everywhere else”

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1, k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0, j_k]$$

monotonicity

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$\text{subj. to} \quad x_i \leq r x_{i-1} - \sum_{j=0}^{i-1} x_j, i \in [1, j_k + 1]$$

robustness “up to the prediction”

Theorem: A partial $r-$robust strategy can be extended to an infinite one if subsequent robustness constraints are saturated

robustness “everywhere else”

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1, k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0, j_k]$$

monotonicity

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$\text{subj. to} \quad x_i \leq r x_{i-1} - \sum_{j=0}^{i-1} x_j, i \in [1, j_k + 1]$$

robustness “up to the prediction”

robustness “everywhere else”

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1, k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0, j_k]$$

monotonicity

Building the LP (for a given configuration j)

$$\min \quad \sum_{i=1}^{k} p_i \cdot \sum_{q=0}^{j_i+1} x_q$$

objective

$$\text{subj. to} \quad x_i \leq r x_{i-1} - \sum_{j=0}^{i-1} x_j, i \in [1, j_k + 1]$$

robustness “up to the prediction”

$$\sum_{i=0}^{j_k+1} x_i \leq \frac{r + \sqrt{r(r-4)}}{2} x_{j_k+1}$$

robustness “everywhere else”

$$x_{j_i} \leq \mu_i \leq x_{j_i+1}, i \in [1, k]$$

configuration definition

$$x_i \leq x_{i+1}, i \in [0, j_k]$$

monotonicity

Complexity, brittleness, and error

Complexity, brittleness, and error

Complexity: $O(c^c)$ LPs, $c = O(\log^k \mu_k)$

We can improve via exponential quantization of μ

Complexity, brittleness, and error

Complexity: $O(c^c)$ LPs, $c = O(\log^k \mu_k)$

We can improve via exponential quantization of μ

Theorem: Let $[m, M]$ be the support of μ. Then the algorithm runs in time polynomial in $\max\{M/m, \log m\}$ and yields a $1 + \varepsilon$ - approximation of the consistency.

Complexity, brittleness, and error

Complexity: $O(c^c)$ LPs, $c = O(\log^k \mu_k)$

We can improve via exponential quantization of μ

Theorem: Let $[m, M]$ be the support of μ. Then the algorithm runs in time polynomial in $\max\{M/m, \log m\}$ and yields a $1 + \varepsilon$ - approximation of the consistency.

Brittleness and error:

- We can detect brittleness around the k probability masses

  If needed, apply techniques from [Elenter et al. 2024], [Perchet and Benomar 2024]

Complexity, brittleness, and error

Complexity: $O(c^c)$ LPs, $c = O(\log^k \mu_k)$

We can improve via exponential quantization of μ

Theorem: Let $[m, M]$ be the support of μ. Then the algorithm runs in time polynomial in $\max\{M/m, \log m\}$ and yields a $1 + \varepsilon$ - approximation of the consistency.

Brittleness and error:

- We can detect brittleness around the k probability masses

  If needed, apply techniques from [Elenter et al. 2024], [Perchet and Benomar 2024]

- EMD-based analysis is likely possible using techniques from contract scheduling [Angelopoulos, Bienkowski, Dürr, and Simon 2024]

  (but some assumptions on the distribution will be required)

Second stochastic setting: randomized algorithms

Second stochastic setting: randomized algorithms

Model: Oracle provides a single point prediction $\hat{u}$, but the algorithm is now **randomized**

Second stochastic setting: randomized algorithms

Model: Oracle provides a single point prediction $\hat{u}$, but the algorithm is now **randomized**

Objective : Given $r \geq e$ find a randomized $r-$robust strategy X of minimum consistency

Second stochastic setting: randomized algorithms

Model: Oracle provides a single point prediction $\hat{u}$, but the algorithm is now **randomized**

Objective : Given $r \geq e$ find a randomized $r-$robust strategy X of minimum consistency

$$\text{cons}(X) = \frac{\mathbb{E}[cost(X, \hat{u})]}{\hat{u}} \quad \text{and} \quad \text{rob}(X) = \sup_{u} \frac{\mathbb{E}[cost(X, u)]}{u}.$$

Second stochastic setting: randomized algorithms

Model: Oracle provides a single point prediction $\hat{u}$, but the algorithm is now **randomized**

Objective : Given $r \geq e$ find a randomized $r-$robust strategy X of minimum consistency

$$\mathrm{cons}(X) = \frac{\mathbb{E}[cost(X, \hat{u})]}{\hat{u}} \quad \text{and} \quad \mathrm{rob}(X) = \sup_u \frac{\mathbb{E}[cost(X, u)]}{u}.$$

Recall: $X = (e^{i+s})_{i \geq 0}$ with $s \sim U[0,1]$ has optimal randomized competitive ratio = e

Upper bound

Upper bound

Parameterized strategy $R_{a,\delta}$ with $a > 1$, $\delta \in [0,1]$

$R_{a,\delta} = (\lambda a^{i+s})_{i \geq 0}$ where $\lambda \in [1,a]$ is such that $\hat{u} = \lambda a^{j+\delta}$ for some $j \in \mathbb{N}$ and $s \sim [\delta,1]$

Upper bound

regulates the “randomness” of the strategy

Parameterized strategy $R_{a,\delta}$ with $a > 1$, $\delta \in [0,1]$

$R_{a,\delta} = (\lambda a^{i+s})_{i \geq 0}$ where $\lambda \in [1,a]$ is such that $\hat{u} = \lambda a^{j+\delta}$ for some $j \in \mathbb{N}$ and $s \sim [\delta,1]$

Upper bound

regulates the “randomness” of the strategy

Parameterized strategy $R_{a,\delta}$ with $a > 1$, $\delta \in [0,1]$

$$R_{a,\delta} = (\lambda a^{i+s})_{i \geq 0} \text{ where } \lambda \in [1,a] \text{ is such that } \hat{u} = \lambda a^{j+\delta} \text{ for some } j \in \mathbb{N} \text{ and } s \sim [\delta,1]$$

Consistency: $\dfrac{a(a-a^{\delta})}{a^{\delta}(a-1)(1-\delta)\ln a}$

Robustness: $\dfrac{a(a-a^{\delta})}{(a-1)(1-\delta)\ln a}$

For $\delta \in \{0,1\}$ we recover the randomized algorithm and the deterministic Pareto-optimal algorithm

Upper bound

regulates the "randomness" of the strategy

Parameterized strategy $R_{a,\delta}$ with $a > 1$, $\delta \in [0,1]$

$$R_{a,\delta} = (\lambda a^{i+s})_{i\geq 0} \text{ where } \lambda \in [1,a] \text{ is such that } \hat{u} = \lambda a^{j+\delta} \text{ for some } j \in \mathbb{N} \text{ and } s \sim [\delta,1]$$

Consistency: $\dfrac{a(a-a^{\delta})}{a^{\delta}(a-1)(1-\delta)\ln a}$

Robustness: $\dfrac{a(a-a^{\delta})}{(a-1)(1-\delta)\ln a}$

For $\delta \in \{0,1\}$ we recover the randomized algorithm and the deterministic Pareto-optimal algorithm

How much improvement relative to deterministic algorithms ?

Robustness	Randomized consistency	Deterministic consistency
$r = 4$	1.724 ($\delta^* \approx 0.9$)	2
$r \geq 5$	**very similar values** $\delta^* \approx 0.99$	

Lower bound (rough idea)

Lower bound (rough idea)

Main idea: Apply **Yao's principle** on the **scalarized** objective **Cons + $\lambda \cdot$ Rob,** **for all** $\lambda \in [0,1]$

(inspired by two-person zero-sum games [Angelopoulos, Lidbetter, Panagiotou 2024])

Lower bound (rough idea)

Main idea: Apply **Yao's principle** on the **scalarized** objective **Cons + $\lambda \cdot$ Rob,** **for all** $\lambda \in [0,1]$

(inspired by two-person zero-sum games [Angelopoulos, Lidbetter, Panagiotou 2024])

Main steps:

- Adversarial distribution $\mathscr{D}$ with a hyperbolic pdf (also recovers the $e-$competitive ratio)
- Express the expected consistency / robustness of any deterministic strategy over $\mathscr{D}$
- Prove a lower bound on the scalarized objective
- By allowing λ to vary, we obtain consistency/robustness tradeoffs

Lower bound (rough idea)

Main idea: Apply **Yao's principle** on the **scalarized** objective **Cons + $\lambda \cdot$ Rob,** **for all** $\lambda \in [0,1]$

(inspired by two-person zero-sum games [Angelopoulos, Lidbetter, Panagiotou 2024])

Main steps:

- Adversarial distribution $\mathscr{D}$ with a hyperbolic pdf (also recovers the $e-$competitive ratio)
- Express the expected consistency / robustness of any deterministic strategy over $\mathscr{D}$
- Prove a lower bound on the scalarized objective
- By allowing λ to vary, we obtain consistency/robustness tradeoffs

Result: The consistency of any $r-$robust strategy is at least $1 + \dfrac{1}{r \cdot F(r)}$, where $F(r) = \ln(r(\ln r + \ln\ln r))$

Lower bound (rough idea)

Main idea: Apply **Yao's principle** on the **scalarized** objective **Cons + $\lambda \cdot$ Rob,** **for all** $\lambda \in [0,1]$

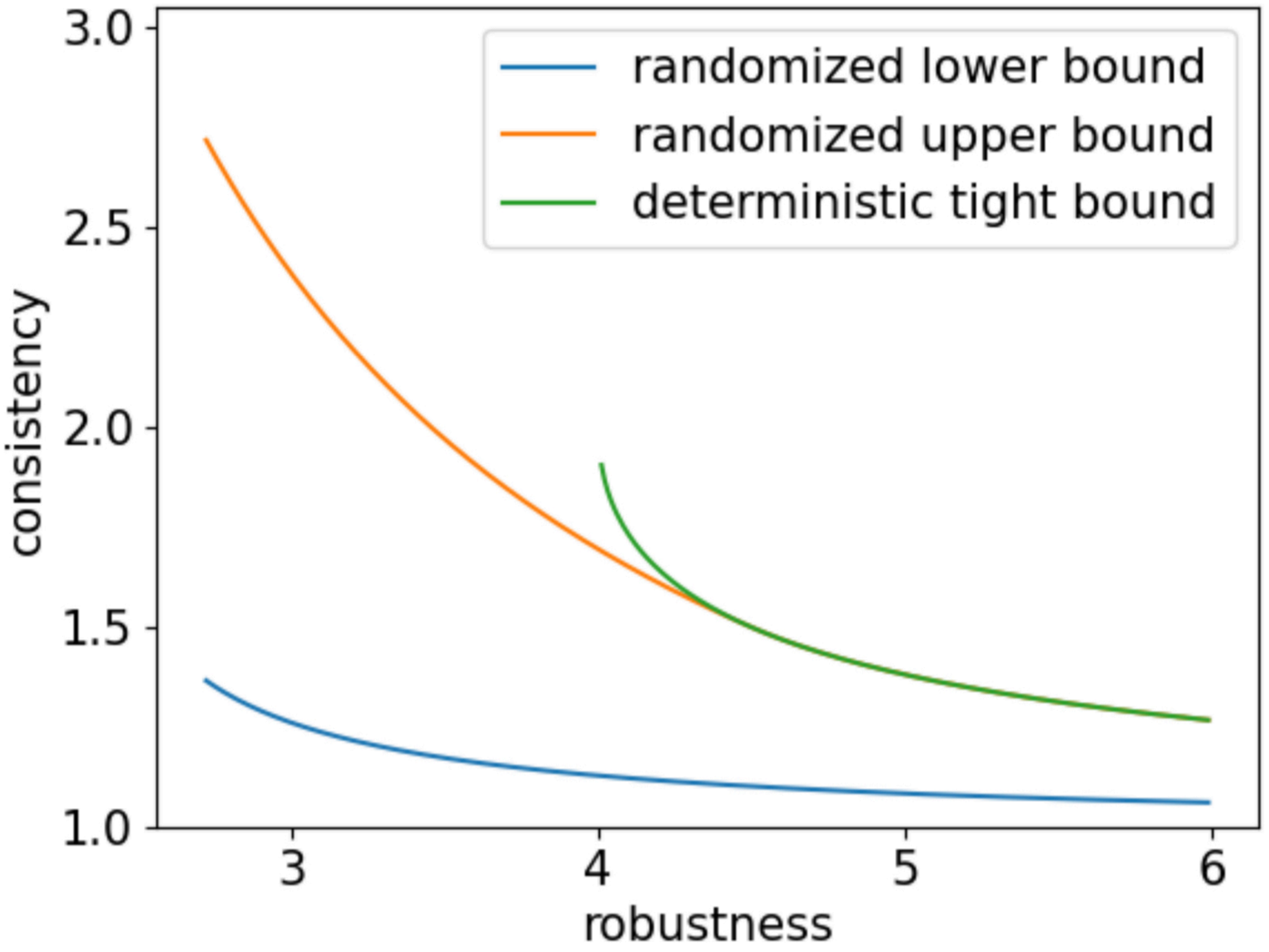
(inspired by two-person zero-sum games [Angelopoulos, Lidbetter, Panagiotou 2024])

Main steps:

- Adversarial distribution $\mathscr{D}$ with a hyperbolic pdf (also recovers the $e-$competitive ratio)
- Express the expected consistency / robustness of any deterministic strategy over $\mathscr{D}$
- Prove a lower bound on the scalarized objective
- By allowing λ to vary, we obtain consistency/robustness tradeoffs

Result: The consistency of any $r-$robust strategy is at least $1 + \dfrac{1}{r \cdot F(r)}$, where $F(r) = \ln(r(\ln r + \ln\ln r)) = \tilde{O}(\ln r)$

Comparison between the bounds



Extensions and further applications

Extensions and further applications

Searching on the infinite line

- Pareto-optimal deterministic algorithms for distributional, **positional** advice

  (robustifies the average-case analysis of [Gal 1980])

- Upper/lower bounds for randomized algorithms with single-point **positional** prediction

  (much harder than directional prediction)

Extensions and further applications

Searching on the infinite line

- Pareto-optimal deterministic algorithms for distributional, **positional** advice

  (robustifies the average-case analysis of [Gal 1980])

- Upper/lower bounds for randomized algorithms with single-point **positional** prediction

  (much harder than directional prediction)

Bidding with dynamic predictions

Setting: Bidding with a sequence of predictions $\hat{u}_1, \hat{u}_2, \ldots$

Objective: Lexicographically Pareto-optimal deterministic algorithm

Result: LP-based polynomial-time solution

Future work

- Improve the time complexity of the LP-based algorithm: is there an FPTAS? Lower bounds?
- Tight(er) upper / lower bounds for randomized algorithms
- Combined settings: Distributional oracles + randomized algorithms (hard)
- Other tools for lower bounds beyond scalarized Yao's principle?

Future work

- Improve the time complexity of the LP-based algorithm: is there an FPTAS? Lower bounds?
- Tight(er) upper / lower bounds for randomized algorithms
- Combined settings: Distributional oracles + randomized algorithms (hard)
- Other tools for lower bounds beyond scalarized Yao's principle?

Thank you!