max planck institut
informatik

Learning-Augmented Online Selection Algorithms

Themis Gouleakis

Joint work with: **Antonios Antoniadis, Pieter Kleer and Pavel Kolev**

September 2, 2020

Online selection problem

- Elements $E = \{e_1, \ldots, e_n\}$ arrive online.
  - Uniformly random arrival order σ of elements in E
- Element e_i has value $v_i \geq 0$ (revealed upon arrival).
- Upon arrival of element e_i: *Select or reject it (irrevocably).*

Goal: Select feasible set S of elements that maximizes

$$f(S) = \sum_{j \in S} v_j.$$

Focus is on (constant-factor) approximation algorithms.

Examples:

- Online (bipartite) matching,
- Matroid secretary problem.

Learning augmentation

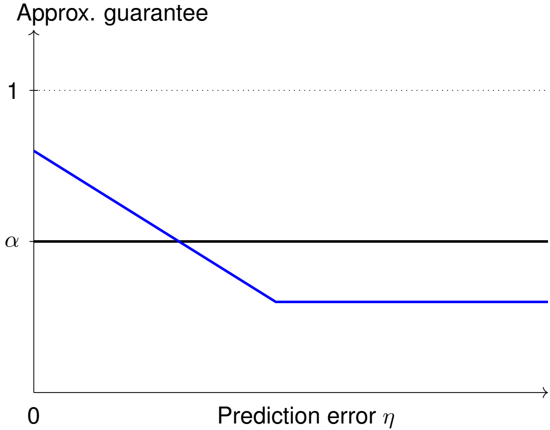
- Machine learning oracle predicts aspect of
  - Input that has not yet arrived.
  - (Offline) optimal solution.
- We do not know quality of prediction.
  - Measured in terms of prediction error η.

Goal: Include predictions in existing α-approximation such that:

- Improved approximation guarantee if η is small.
- Minor loss in approximate guarantee if η is large.

"Best of both worlds"-scenario:

- *Improved guarantees if ML oracle is accurate.*
- *Still guarantee in worst-case when oracle is inaccurate.*


Approx. guarantee
1
α
0
Prediction error η

(Some) related work

Machine learned advice:

- Ski rental
  - [Purohit-Svitkina-Kumar, NIPS 2018], [Wang-Wang, 2020].
- Scheduling
  - [Purohit-Svitkina-Kumar, NIPS 2018], [Mitzenmacher, 2019], [Lattanzi-Lavastida-Moseley-Vassilvitskii, SODA 2020].
- Caching
  - [Lykouris-Vassilvitskii, ICML 2018], [Rothagi, SODA 2020].
- Metric Algorithms
  - [Antoniadis-Coester-Eliás-Polak-Simon, ICML 2020].

Online selection problems with distributional information: $v_i \sim \mathcal{F}_i$.

- Prophet inequalities (adversarial arrival order)
  - Single item: [Krengel-Sucheston, 1978].
  - Matroid prophet inequality: [Kleinberg-Weinberg, 2012].
  - Unknown distribution: e.g., [Correa-Dütting-Fischer-Schewior, '19].

Secretary problem

- Elements (secretaries) $\{e_1, \ldots, e_n\}$ arrive over time.
  - Uniform random arrival order $\sigma = (e_1, \ldots, e_n)$.
- Value v_i revealed upon arrival of e_i.

Goal: Select secretary with maximum value $v^* = \max_i v_i$.

Secretary algorithm [Lindley, 1961]/[Dynkin, 1963]

Phase I:

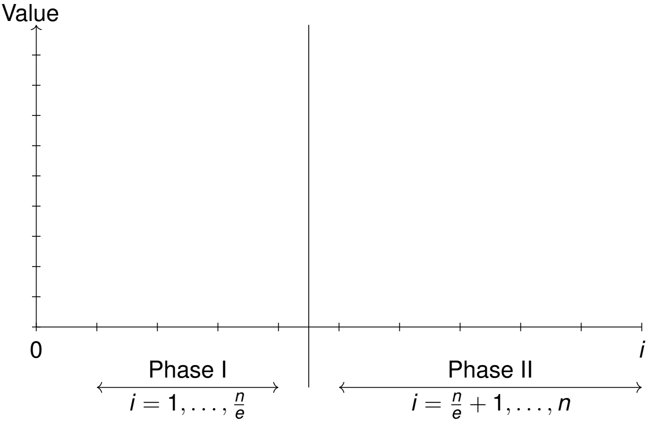
- For $i = 1, \ldots, \frac{n}{e}$: Select nothing.

Phase II:

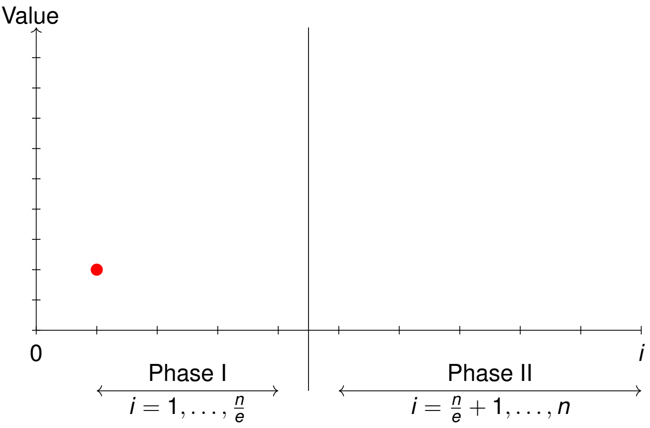
- Set threshold $t = \max_{j=1,\ldots,\frac{n}{e}} v_j$.
- For $i = \frac{n}{e} + 1, \ldots, n$: If $v_i > t$, select e_i and STOP.

Gives $\frac{1}{e}$-approximation for maximum value v^*, i.e., $\mathbb{E}_\sigma[\bar{v}] \geq \frac{1}{e} \cdot v^*$.

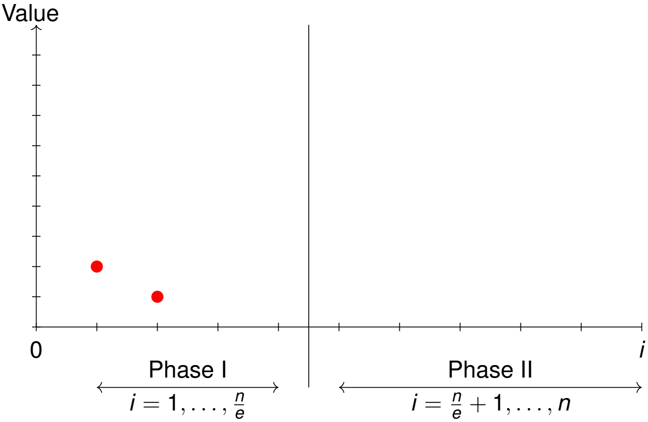
Example



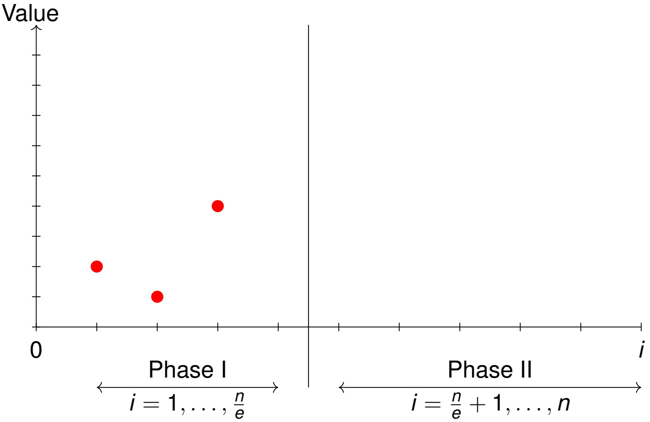
Example



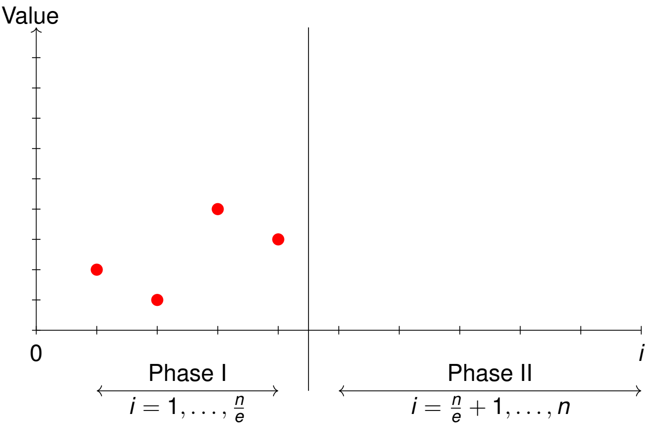
Example



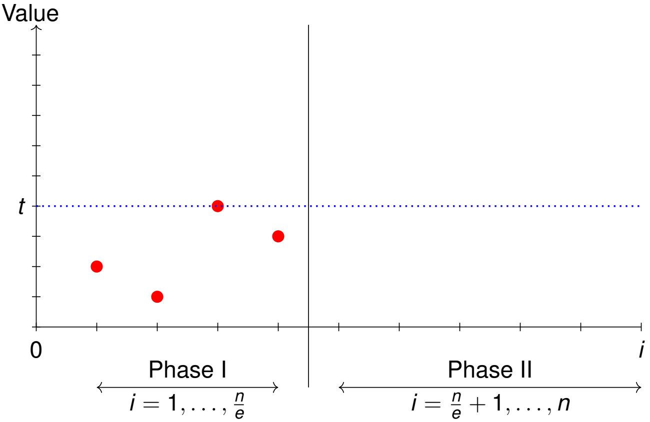
Example



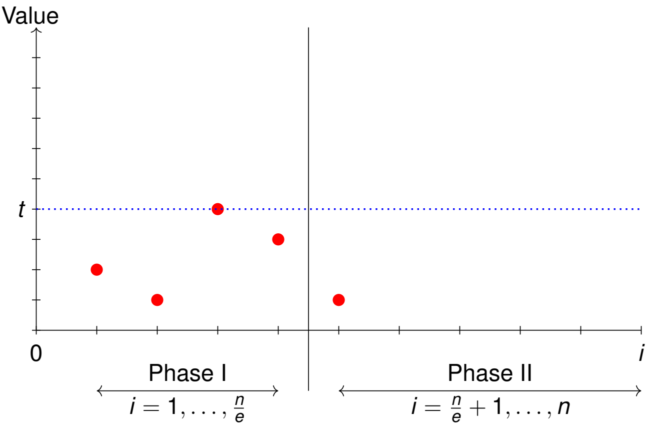
Example



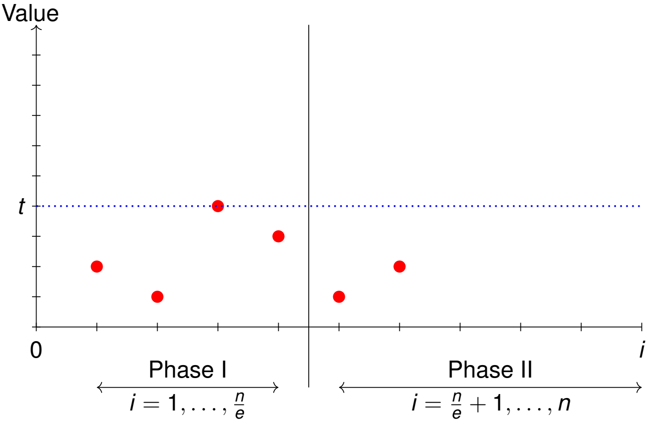
Example



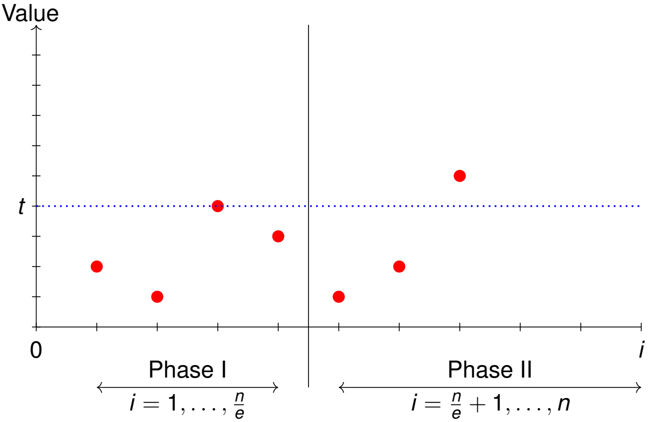
Example



Example



Example



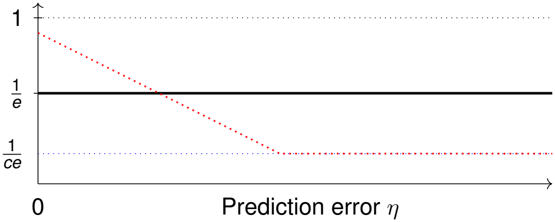
Prediction

We include prediction p^* for optimal value v^*.

- Prediction error $\eta = |p^* - v^*|$.

Goal (informal): Design (deterministic) algorithm such that:

- Approximation guarantee $> \frac{1}{e}$ when η is small.
- Approximation guarantee $\approx \frac{1}{ce}$ when η is large.
  - For some constant $c > 1$.



What to do when prediction is good?

What to do when prediction is good?

Choose element with value ‘close’ to prediction:

What to do when prediction is good?

Choose element with value ‘close’ to prediction:

- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.

What to do when prediction is good?

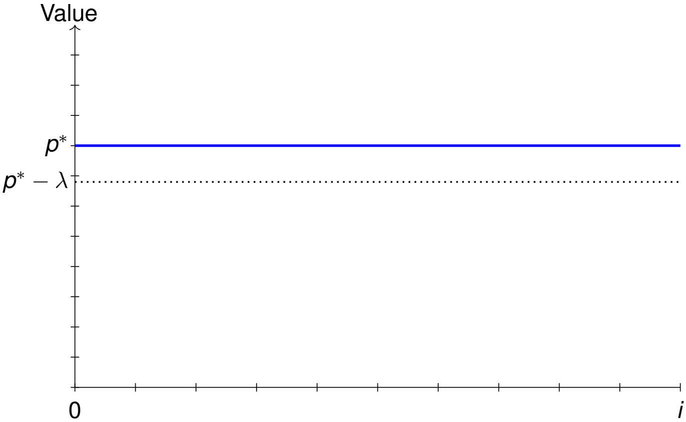
Choose element with value 'close' to prediction:

- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.

What to do when prediction is good?

Choose element with value ‘close’ to prediction:

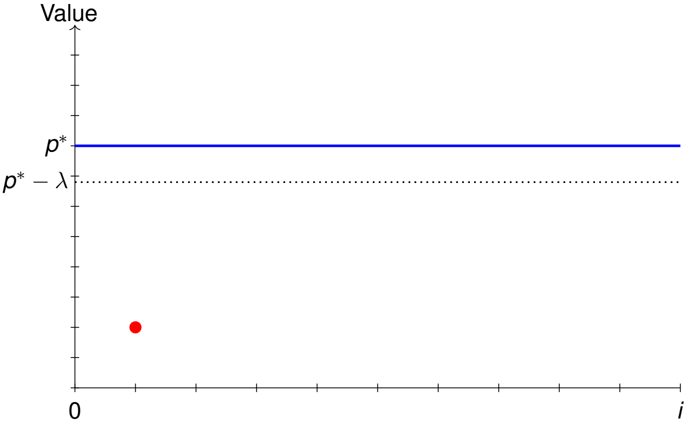
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

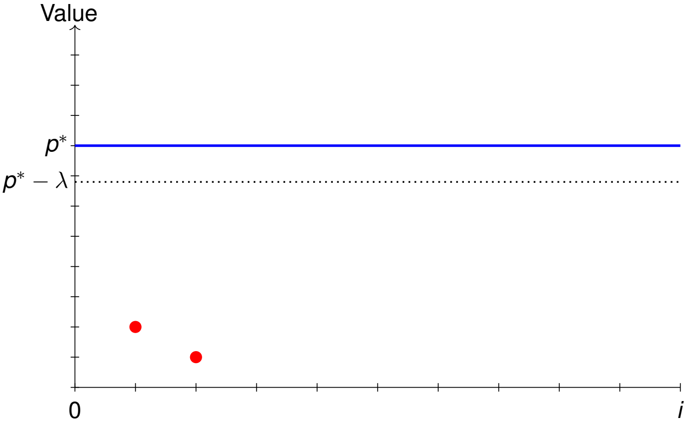
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

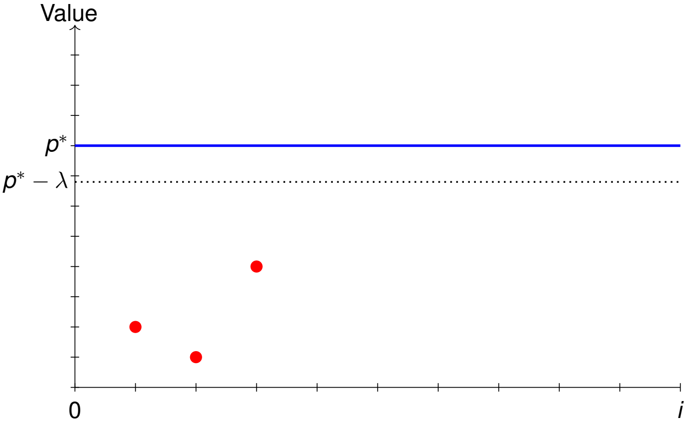
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

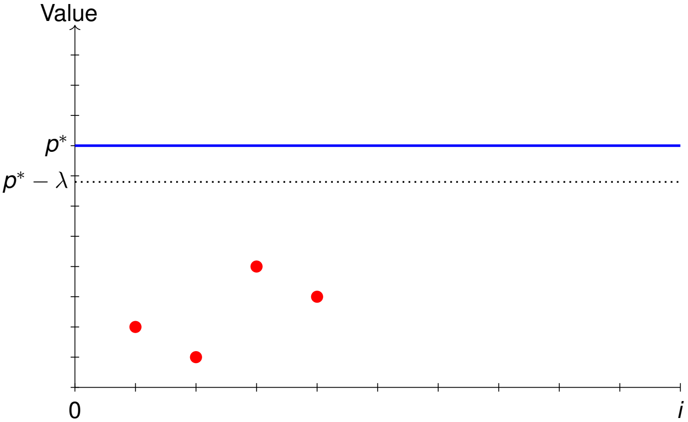
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

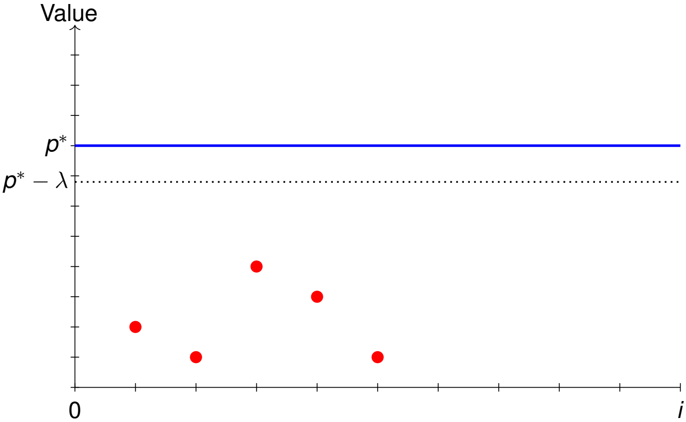
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

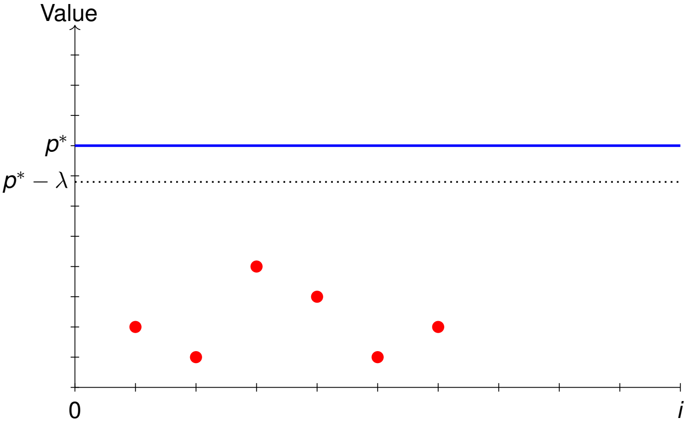
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

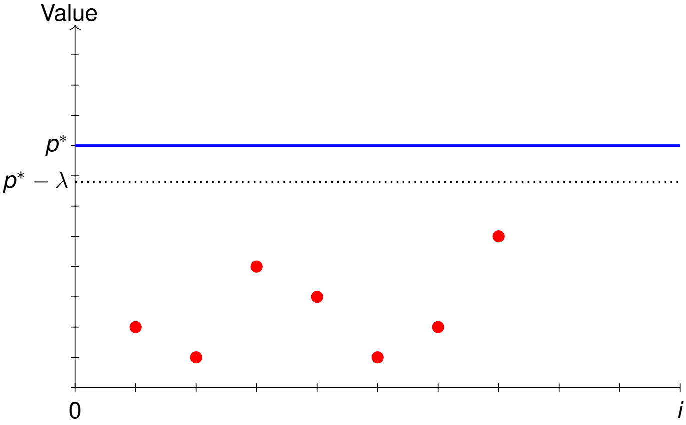
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

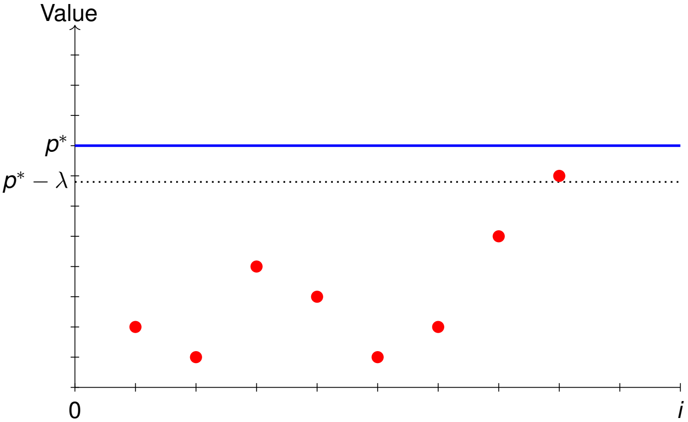
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

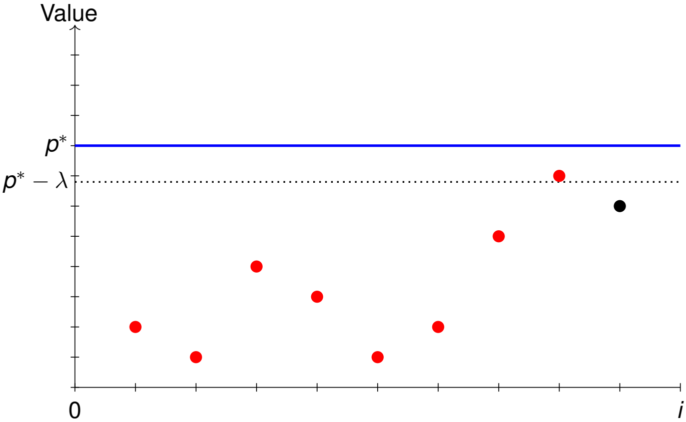
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value 'close' to prediction:

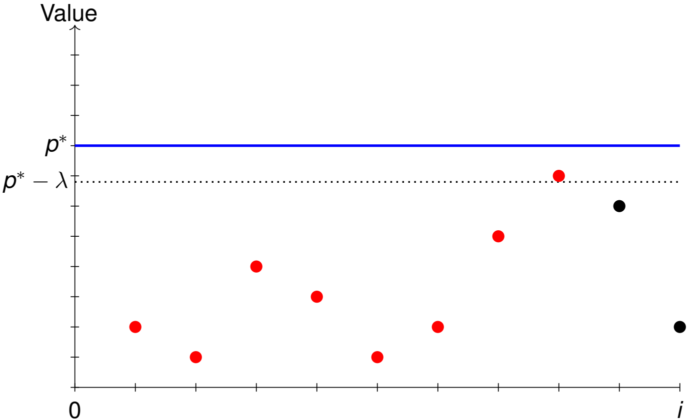
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

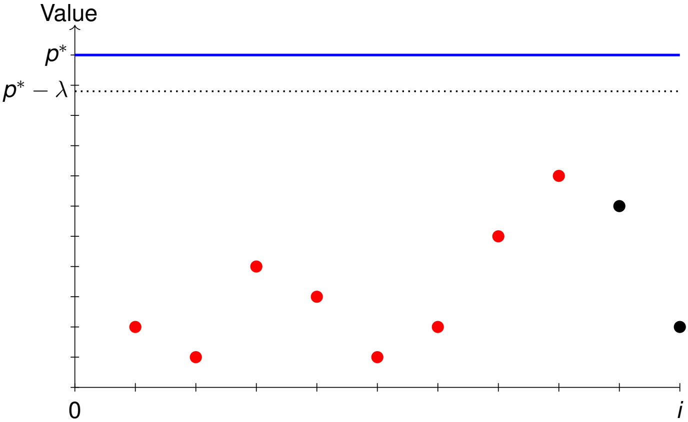
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

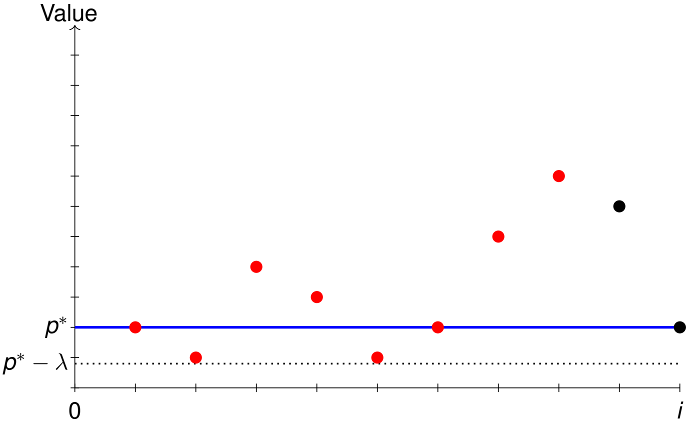
- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



What to do when prediction is good?

Choose element with value ‘close’ to prediction:

- Fix $\lambda > 0$, and select first element with $v_i > p^* - \lambda$.
- Parameter λ can be seen as estimator for η.



Algorithm with prediction

Input: Parameters $0 < \gamma \leq \delta \leq 1$ and $\lambda > 0$; prediction p^*.

Our algorithm

Phase I (*Observation*):

- For $i = 1, \ldots, \gamma n$: Select nothing.

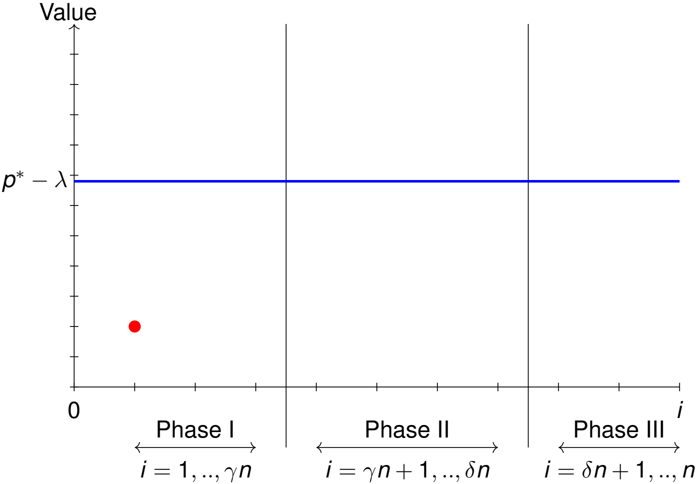
Phase II (*Exploiting prediction*):

- Set threshold $t = \max\left\{p^* - \lambda, \max_{j=1,\ldots,\gamma n} v_j\right\}$.
- For $i = \gamma n + 1, \ldots, \delta n$: If $v_i > t$, select e_i and STOP.

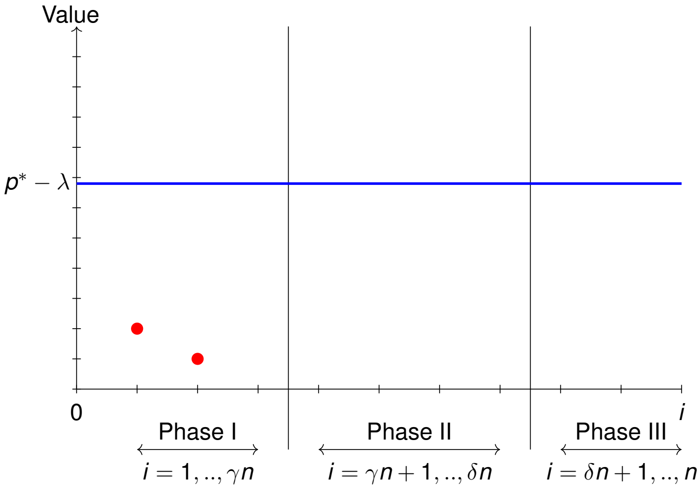
Phase III (*Classical algorithm*):

- (Re)set threshold $t = \max_{j=1,\ldots,\delta n} v_j$.
- For $i = \delta n + 1, \ldots, n$: If $v_i > t$, select e_i and STOP.

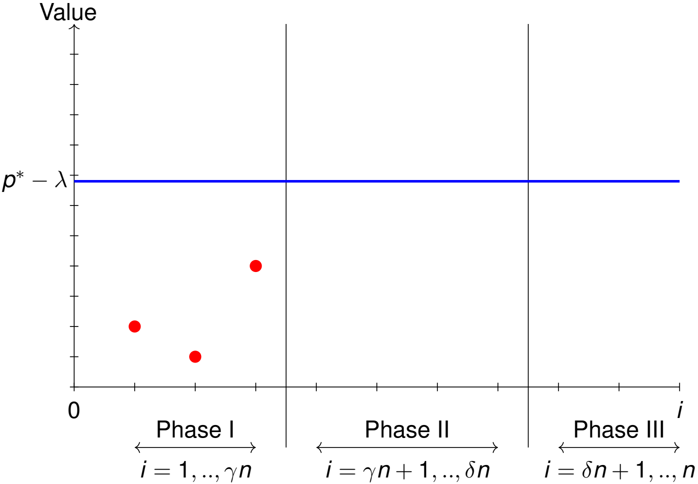
Example


Value
$p^* - \lambda$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

Example

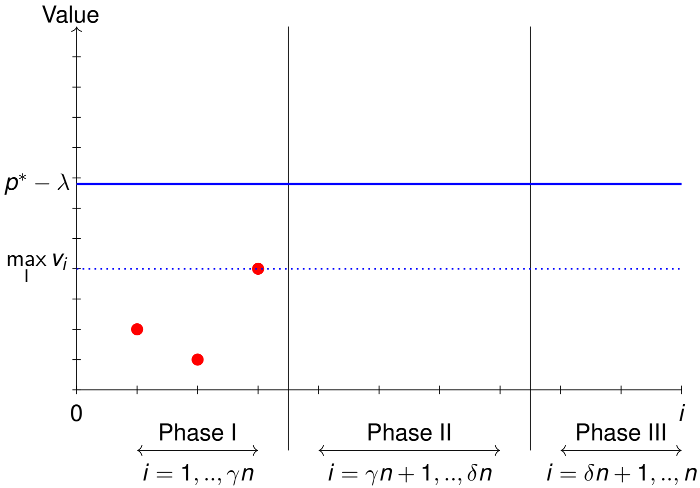

Value
$p^* - \lambda$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

Example

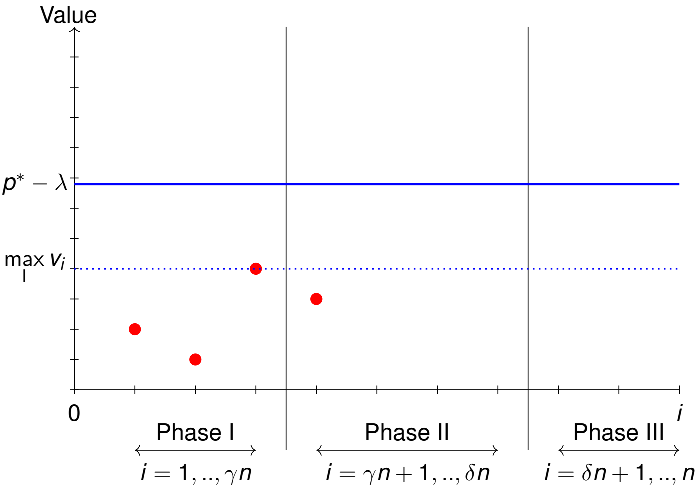


Value
$p^* - \lambda$
0
i
Phase I
Phase II
Phase III
$i = 1, .., \gamma n$
$i = \gamma n + 1, .., \delta n$
$i = \delta n + 1, .., n$

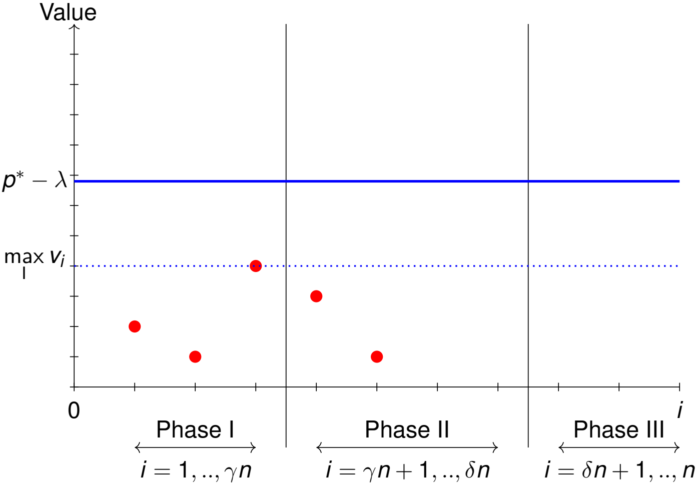
Example


Value
$p^* - \lambda$
$\max_I v_i$
0
i
Phase I
Phase II
Phase III
$i = 1, .., \gamma n$
$i = \gamma n + 1, .., \delta n$
$i = \delta n + 1, .., n$

Example

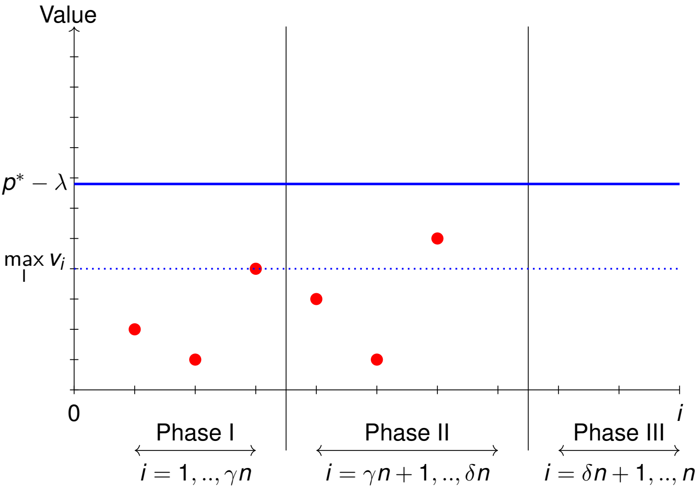

Value
$p^* - \lambda$
$\max_l v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

Example

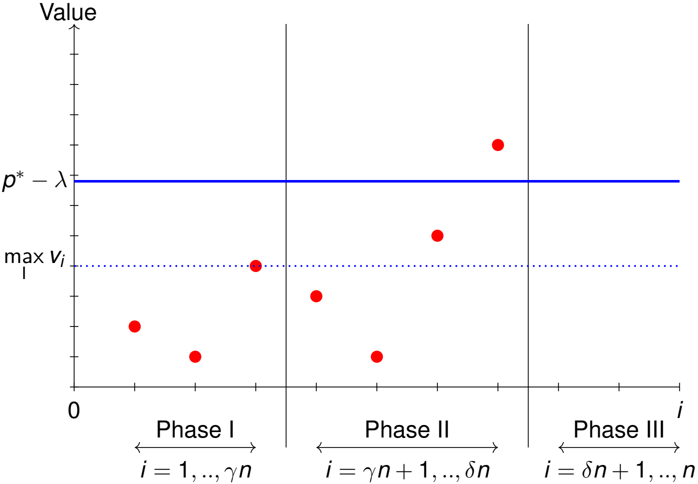


Value
$p^* - \lambda$
$\max_l v_i$
0
i
Phase I
Phase II
Phase III
$i = 1, .., \gamma n$
$i = \gamma n + 1, .., \delta n$
$i = \delta n + 1, .., n$

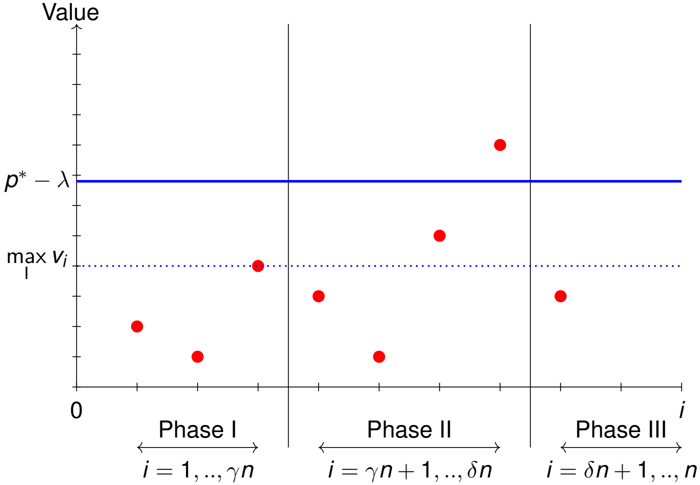
Example


Value
$p^* - \lambda$
$\max_{I} v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

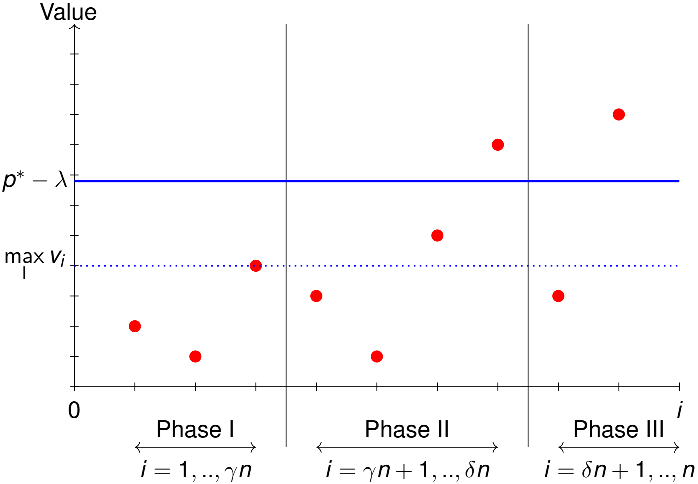
Example


Value
$p^* - \lambda$
$\max_I v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

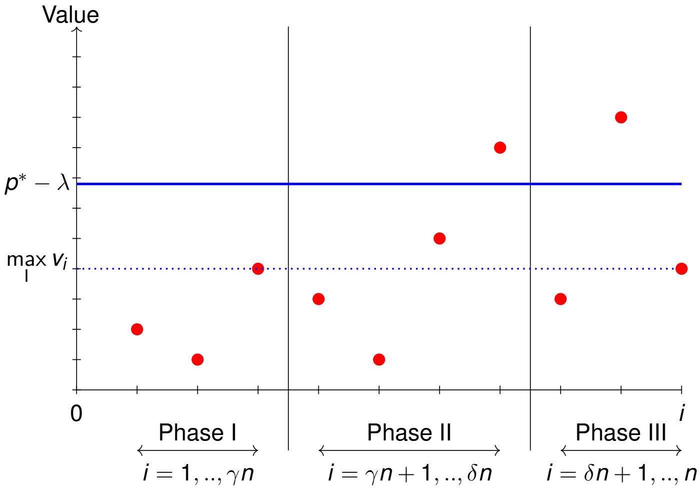
Example


Value
$p^* - \lambda$
$\max_{\mathrm{I}} v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

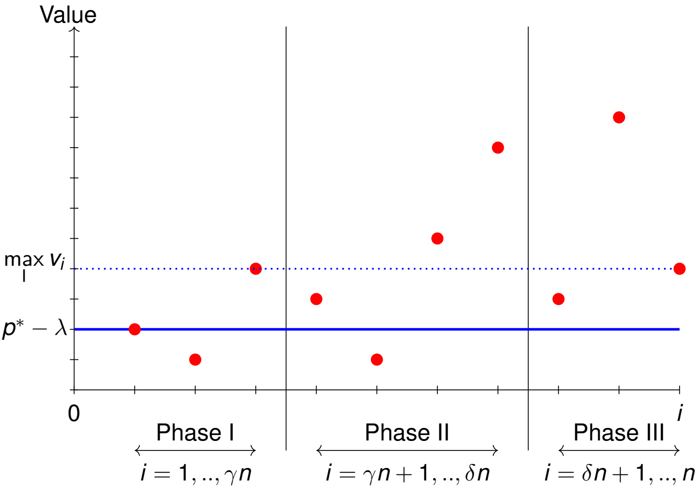
Example


Value
$p^* - \lambda$
$\max_I v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

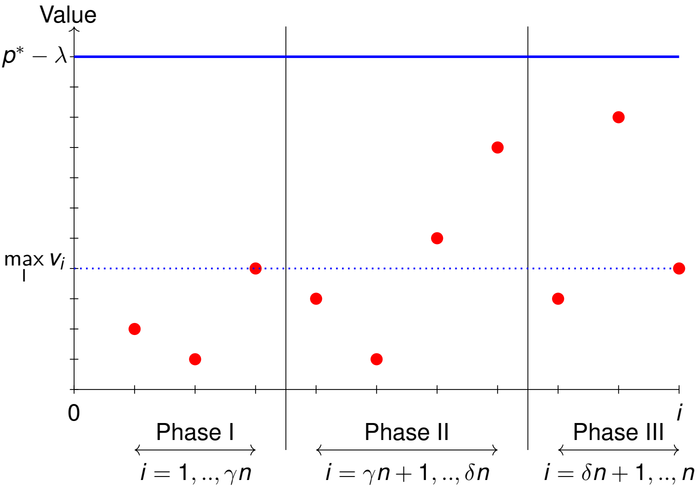
Example


Value
$p^* - \lambda$
$\max_i v_i$
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

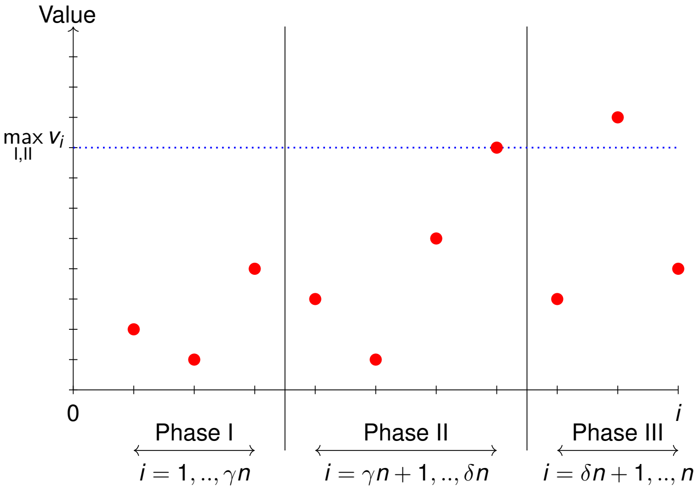
Example


Value
$\max_I v_i$
$p^* - \lambda$
0
i
Phase I
Phase II
Phase III
$i = 1, .., \gamma n$
$i = \gamma n + 1, .., \delta n$
$i = \delta n + 1, .., n$

Example


Value
$p^* - \lambda$
$\max_{I} v_i$
0
i
Phase I
Phase II
Phase III
$i = 1, .., \gamma n$
$i = \gamma n + 1, .., \delta n$
$i = \delta n + 1, .., n$

Example


Value
max v_i
I,II
0
i
Phase I
$i = 1, .., \gamma n$
Phase II
$i = \gamma n + 1, .., \delta n$
Phase III
$i = \delta n + 1, .., n$

Tunable parameters:

Tunable parameters:

- Confidence parameter $\lambda > 0$;

Tunable parameters:

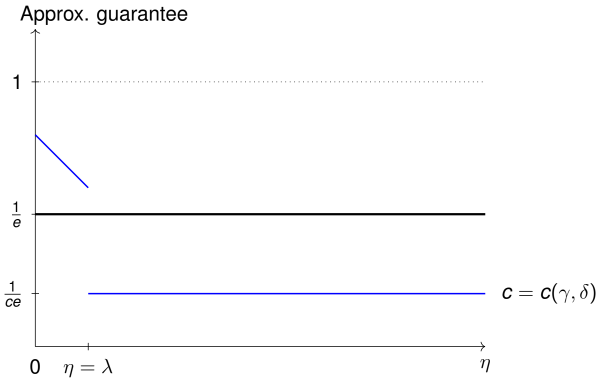
- Confidence parameter $\lambda > 0$;
- Phase lengths determined by $0 < \gamma \leq \delta \leq 1$.

Tunable parameters:

- Confidence parameter $\lambda > 0$;
- Phase lengths determined by $0 < \gamma \leq \delta \leq 1$.

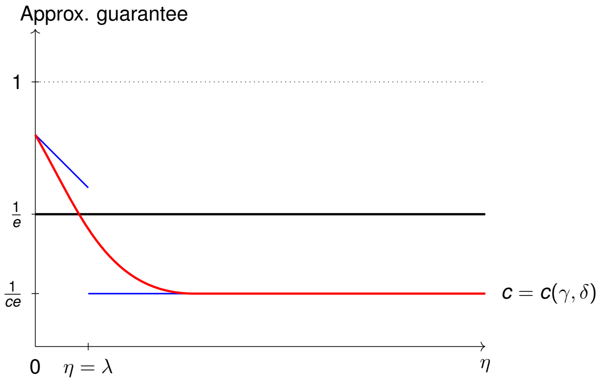
Tunable parameters:

- Confidence parameter $\lambda > 0$;
- Phase lengths determined by $0 < \gamma \leq \delta \leq 1$.



Tunable parameters:

- Confidence parameter $\lambda > 0$;
- Phase lengths determined by $0 < \gamma \leq \delta \leq 1$.



High-level challenge:

- Different algorithms for different parts of the element stream.
  - One for exploiting predictions.
  - One for worst-case theoretical guarantee.
- Make sure they do not conflict (too much) with each other.
  - "Bad choices" in one part should not affect other part too much.

Often (seems) non-trivial to achieve deterministically.

Online bipartite matching

Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

- Nodes in L arrive online.

Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

- Nodes in L arrive online.
  - Uniform random arrival order.

Online bipartite matching

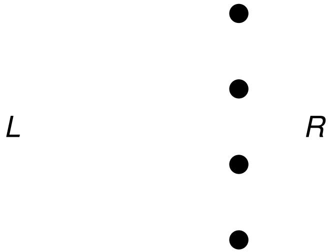
Given is bipartite graph $G = (L \cup R, E)$.

- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).

Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

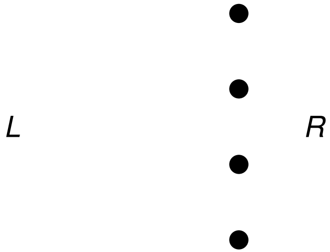
- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

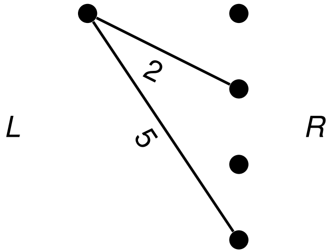
- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

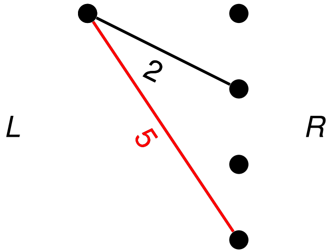
- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

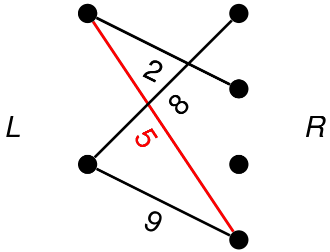
- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

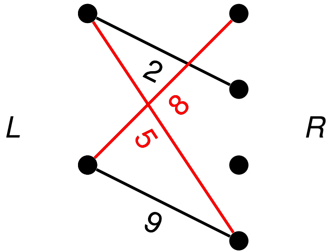
- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Online bipartite matching

Given is bipartite graph $G = (L \cup R, E)$.

- Nodes in L arrive online.
  - Uniform random arrival order.
- Upon arrival, $\ell \in L$ reveals edge-weights w_e to neighbors in R.
- Match up ℓ with currently unmatched node in R (or do nothing).



Goal: Select matching M with maximum weight $\sum_{e \in M} w_e$.

Related work

"Secretary" (online) bipartite matching:

- [Babaioff-Immorlica-Kempe-Kleinberg, 2007]
  - $\frac{1}{16}$-approximation for transversal matroids.
- [Dimitrov-Plaxton, 2008]
  - $\frac{1}{8}$-approximation for transversal matroids.
- [Korula-Pál, 2009]
  - $\frac{1}{8}$-approximation
- [Kesselheim-Radke-Tönnis-Vöcking, 2013].
  - $\frac{1}{e}$-approximation.

Last result best possible.

Predictions

Predictions

Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

Predictions

Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

- There is an offline optimal solution OPT with:

Predictions

Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

- There is an offline optimal solution OPT with:
  - Node $r \in R$ adjacent to edge with weight p_r^* in OPT.

Predictions

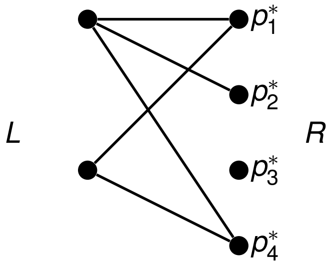
Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

- There is an offline optimal solution OPT with:
  - Node $r \in R$ adjacent to edge with weight p_r^* in OPT.
  - Prediction error $\eta = \max_r |p_r^* - OPT_r|$.

Predictions

Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

- There is an offline optimal solution OPT with:
  - Node $r \in R$ adjacent to edge with weight p_r^* in OPT.
  - Prediction error $\eta = \max_r |p_r^* - OPT_r|$.



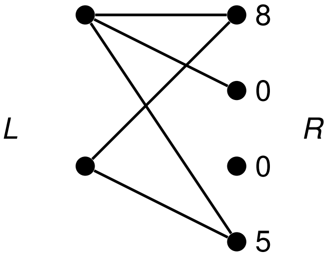
Perfect predictions: Online vertex-weighted bipartite matching.

- [Aggarwal-Goel-Karande-Mehta, 2011]

Predictions

Vector $p = (p_1^*, \ldots, p_{|R|}^*)$.

- There is an offline optimal solution OPT with:
  - Node $r \in R$ adjacent to edge with weight p_r^* in OPT.
  - Prediction error $\eta = \max_r |p_r^* - OPT_r|$.



Perfect predictions: Online vertex-weighted bipartite matching.

- [Aggarwal-Goel-Karande-Mehta, 2011]

Algorithm with predictions

Our algorithm

Construct (online) matching M.

Phase I (*Observation*):

- For $i = 1, \dots, \gamma n$: Select nothing.

Phase II (*[KRTV'13]*):

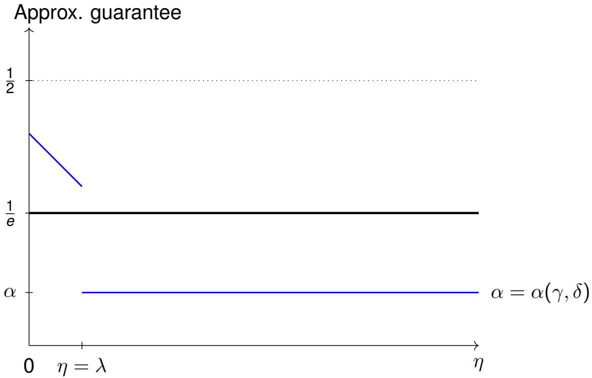
- For $i = \gamma n + 1, \dots, \delta n$:
  - Compute offline optimal solution OPT on $G[\{1, \dots, i\} \cup R]$.
  - If $\{i, r\} \in$ OPT for some $r \in R$, and r unmatched in M: $M \leftarrow M \cup \{i, r\}$.

Phase III (*Exploiting predictions*):

- For $i = \delta n + 1, \dots, n$:
- Run greedy algorithm for vertex-weighted bipartite online matching problem with node weights $p_r^* - \lambda$ for each $r \in R$.

Tunable parameters:

- Confidence parameter $\lambda > 0$;
- Phase lengths determined by $0 < \gamma \leq \delta \leq 1$.

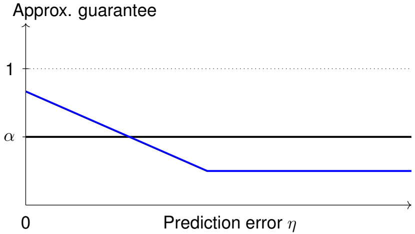


In our prediction model, guarantee of $\frac{1}{2}$ is best we can hope for.

Summary

Include ML predictions in existing α-approximation such that:

- Improved approximation guarantee if η is small.
- Minor loss in approximate guarantee if η is large.



- We study the following problems.
  - Classical secretary problem.
  - Online bipartite matching.
  - Graphic matroid secretary problem.

Thank you.