Pandora's Box with Correlations: Learning and Approximation

Shuchi Chawla[1], *Evangelia Gergatsouli*[1], Yifeng Teng[1], Christos Tzamos[1], Ruimin Zhang[1]

[1]University of Wisconsin-Madison

ACAC, August 2020

A Search Problem

Find the best out of n alternatives!

A Search Problem

Find the best out of n alternatives!

$\mathcal{D}_1$ $\mathcal{D}_2$ $\mathcal{D}_3$ $\mathcal{D}_4$ $\mathcal{D}_5$

- Stochastic information on price

A Search Problem

Find the best out of n alternatives!

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$
1	4	2	7	3

- Stochastic information on price
- Information is not free!

A Search Problem

Find the best out of n alternatives!

42	??	17	13	??
1	4	2	7	3

- Stochastic information on price
- Information is not free!

A Search Problem

Find the best out of n alternatives!

42	??	17	13	??
1	4	2	7	3

- Stochastic information on price
- Information is not free!
- Open boxes until decide to stop (*stopping rule*).
- Keep best price seen so far

Instantiation of prices = *scenario*

A Search Problem

Find the best out of n alternatives!

42	??	17	13	??
1	4	2	7	3

- Stochastic information on price
- Information is not free!

Maximization version: *max price - information cost*
Minimization version: *min price + information cost*

This paper: focus on minimization

A Search Problem - What do we know

Pandora's Box [Weitzman '79] greedy gives optimal!

- Assign an index to every box
- Search boxes in order of index until: current price better than index of next box

A Search Problem - What do we know

Pandora's Box [Weitzman '79] greedy gives optimal!

- Assign an index to every box
- Search boxes in order of index until: current price better than index of next box

Crucial assumption: distributions are **independent**!

A Search Problem - What do we know

Pandora's Box [Weitzman '79] greedy gives optimal!

- Assign an index to every box
- Search boxes in order of index until: current price better than index of next box

Crucial assumption: distributions are **independent**!

What about **correlation?**

Our setting: sample access, arbitrarily correlated $\mathcal{D}$'s

A Search Problem - What do we know

Pandora's Box [Weitzman '79] greedy gives optimal!

- Assign an index to every box
- Search boxes in order of index until: current price better than index of next box

Crucial assumption: distributions are **independent**!

What about **correlation?**

Our setting: sample access, arbitrarily correlated $\mathcal{D}$'s

Related but different: Optimal Decision Tree (require small support/explicit distributions)

Approximating the Optimal

Hard Problem: encode location of best box in prices of other boxes

Example: prices 4 and 2 means go to box 42 to find best price

Approximating the Optimal

Hard Problem: encode location of best box in prices of other boxes

Box i *Box j* ... *Box 42*

4 2 0

Example: prices 4 and 2 means go to box 42 to find best price

Cannot learn arbitrary mapping with finitely many samples!

Approximating the Optimal

Hard Problem: encode location of best box in prices of other boxes

Box i	*Box j*		*Box 42*
		...	
4	2		0

Example: prices 4 and 2 means go to box 42 to find best price

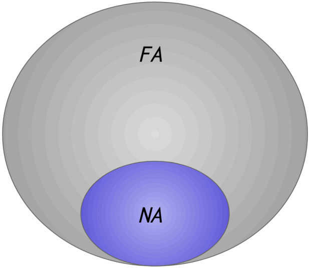
Cannot learn arbitrary mapping with finitely many samples!

Best Strategy: decide next box after seen prices. Other strategies?

Strategies

Strategy: (1) What is next box? (2) When do I stop?

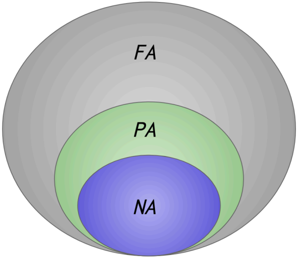
- *Fully Adaptive*: next box/stopping rule **both** adaptive
- *Non-Adaptive*: fixed order **and** stopping time
  Fixed stopping time: fix a set of boxes to open all at once, decide which to pick



Strategies

Strategy: (1) What is next box? (2) When do I stop?

- *Partially Adaptive*: fixed order, adaptive stopping time (for independent $\mathcal{D}$ this gives optimal policy!)



Approximating Other Strategies

- Fully Adaptive: Learning/Approximation: Hard!
  Example: encoded location of best box
- Non-Adaptive:
  - Learning: Hard!: tiny probability scenario has price$=\infty$ on all boxes but one$\rightarrow$either query all boxes **or** sample this scenario
  - Approximation: As hard as Set Cover! For $0/\infty$ prices $\rightarrow$ find a 0 for every scenario $\rightarrow$ hitting set formulation of set cover
- Partially Adaptive: Can Learn & Efficiently approximate!

Main Theorem

Using polynomially in n sampled scenarios we can efficiently find a Partially Adaptive strategy that is $O(1)$-competitive against the optimal Partially Adaptive strategy.

Roadmap to Main Result

Space of PA strategies can be large! $\rightarrow$ Scenario-aware PA

SPA: Fix order $\rightarrow$ scenario is revealed$\rightarrow$decide stopping time

Algorithm:

1. Draw samples of scenarios
2. Design good SPA strategy using samples
3. Find stopping rule that performs well

Roadmap to Main Result

Space of PA strategies can be large! $\rightarrow$ Scenario-aware PA

SPA: Fix order $\rightarrow$ scenario is revealed$\rightarrow$decide stopping time

Algorithm:

1. Draw samples of scenarios (Learning Lemma)
2. Design good SPA strategy using samples (Main Algorithm)
3. Find stopping rule that performs well (Myopic Stopping Lemma)

Roadmap to Main Result

Algorithm:

1. Draw samples of scenarios (Learning Lemma)
2. Design good SPA strategy using samples (Main Algorithm)
3. **Find stopping rule that performs well (Myopic Stopping Lemma)**

Lemma (Myopic Stopping)

For any order, there is an adaptive stopping rule that 2-approximates the optimal Scenario-aware stopping rule.

Roadmap to Main Result

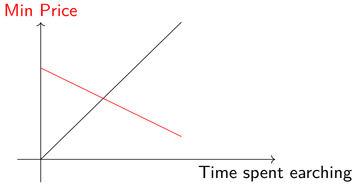
Algorithm:

1. Draw samples of scenarios (Learning Lemma)
2. Design good SPA strategy using samples (Main Algorithm)
3. **Find stopping rule that performs well (Myopic Stopping Lemma)**

Lemma (Myopic Stopping)

For any order, there is an adaptive stopping rule that 2-approximates the optimal Scenario-aware stopping rule.

Proof Sketch: Assume a SPA order→ need to find a stopping rule for PA. Stop when best price seen so far is at most time spent until now[a].



[a]Argument is equivalent to Ski-Rental→ can get 1.58 using ski rental algorithm.

Roadmap to Main Result

Algorithm:

1. Draw samples of scenarios (Learning Lemma)
2. Design good SPA strategy using samples (Main Algorithm)
3. **Find stopping rule that performs well (Myopic Stopping Lemma)**

Lemma (Myopic Stopping)

For any order, there is an adaptive stopping rule that 2-approximates the optimal Scenario-aware stopping rule.

Focus on SPA then convert to PA losing a factor of 2.

Roadmap to Main Result

Algorithm:

1. **Draw samples of scenarios (Learning Lemma)**
2. Design good SPA strategy using samples (Main Algorithm)
3. Find stopping rule that performs well (Myopic Stopping Lemma)

Lemma

Near-Optimal SPA Strategies can be efficiently learned from poly(n) number of samples.

Roadmap to Main Result

Algorithm:

1. **Draw samples of scenarios (Learning Lemma)**
2. Design good SPA strategy using samples (Main Algorithm)
3. Find stopping rule that performs well (Myopic Stopping Lemma)

Lemma

Near-Optimal SPA Strategies can be efficiently learned from poly(n) number of samples.

Proof Sketch.

Possible permutations: $n!$

Each permutation has bounded cost$\rightarrow$can learn with few samples$\rightarrow$ union bound on all $n!$ permutations. $\square$

Roadmap to Main Result

Algorithm:

1. **Draw samples of scenarios (Learning Lemma)**
2. Design good SPA strategy using samples (Main Algorithm)
3. Find stopping rule that performs well (Myopic Stopping Lemma)

Lemma

Near-Optimal SPA Strategies can be efficiently learned from poly(n) number of samples.

Enough to find good SPA strategies!

Roadmap to Main Result

Algorithm:

1. Draw samples of scenarios (Learning Lemma)
2. **Design good SPA strategy using samples (Main Algorithm)**
3. Find stopping rule that performs well (Myopic Stopping Lemma)

Main Result: SPA vs PA

This talk: Focus on SPA vs NA

PA vs NA - LP Formulation

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i\in\mathcal{B}} x_i \quad + \quad \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{B}, s\in\mathcal{S}} c_{is} z_{is} && \text{(LP-NA)}\\
\text{subject to} \quad & \sum_{i\in\mathcal{B}} z_{is} \quad = \quad 1, && \forall s\in\mathcal{S} \qquad (1)\\
& z_{is} \quad \le \quad x_i, && \forall i\in\mathcal{B}, s\in\mathcal{S}\\
& x_i, z_{is} \quad \in \quad [0,1] && \forall i\in\mathcal{B}, s\in\mathcal{S}
\end{aligned}
$$

x_i: indicates whether box i is opened

z_{is}: indicates whether box i is assigned to scenario s

c_{is}: price in box i for scenario s

PA vs NA - Algorithm

Given: Solution $\boldsymbol{x}, \boldsymbol{z}$ to LP, scenario s

1. Open box i wp $\frac{x_i}{\sum_{i \in \mathcal{B}} x_i}$
2. If box i is opened, select the box and stop wp $\frac{z_{is}}{x_i}$

Analysis: Bound probing cost + price

- Part 1: bound probing cost

$$\mathbf{Pr}\left[\text{stop at step } t\right] = \sum_{i \in \mathcal{B}} \frac{x_i}{\sum_{i \in \mathcal{B}} x_i} \frac{z_{is}}{x_i} = \frac{\sum_{i \in \mathcal{B}} z_{is}}{\sum_{i \in \mathcal{B}} x_i} = \frac{1}{\mathrm{OPT}_t},$$

Probing cost is optimal on expectation

PA vs NA - Analysis

- Part 2: bound the price
  For scenario s

$$\begin{aligned}
\mathbf{E}\left[\mathrm{ALG}_{c,s}\right] &= \sum_{i \in \mathcal{B}, t} \mathbf{Pr}\left[\text{select } i \text{ at } t \mid \text{stop at } t\right] \mathbf{Pr}\left[\text{stop at } t\right] c_{is} \\
&\leq \sum_{i \in \mathcal{B}, t} \frac{z_{is}}{\sum_{i \in \mathcal{B}} z_{is}} \mathbf{Pr}\left[\text{stop at } t\right] c_{is} \\
&= \sum_{i \in \mathcal{B}} z_{is} c_{is} \\
&= \mathrm{OPT}_{c,s}
\end{aligned}$$

  Take expectation over all scenarios $\mathbf{E}\left[\mathrm{ALG}_c\right] \leq \mathrm{OPT}_c$

  SPA Approximates NA $\rightarrow$ lose a 2-factor to convert to PA

Summary - Extensions

Showed: Can approximate NA with PA within 2.

Summary - Extensions

Showed: Can approximate NA with PA within 2. **Other settings?**

	Choose 1	Choose k	Matroid rank k
PA vs PA (Upper-bound)	9.22	$O(1)$	$O(\log k)$
FA vs NA (Lower-bound)	1.27	1.27	$\Omega(\log k)$

Table: Summary of Results

Summary - Extensions

Showed: Can approximate NA with PA within 2. **Other settings?**

	Choose 1	Choose k	Matroid rank k
PA vs PA (Upper-bound)	9.22	$O(1)$	$O(\log k)$
FA vs NA (Lower-bound)	1.27	1.27	$\Omega(\log k)$

Table: Summary of Results

Main Result: related to Min Sum Set Cover [Feige et al. 2002]

Choose k, matroid: Related to Generalized Min Sum Set Cover [Bansal et al. 2010 & Skutella, Williamson 2011]

Summary - Extensions

Showed: Can approximate NA with PA within 2. **Other settings?**

	Choose 1	Choose k	Matroid rank k
PA vs PA (Upper-bound)	9.22	$O(1)$	$O(\log k)$
FA vs NA (Lower-bound)	1.27	1.27	$\Omega(\log k)$

Table: Summary of Results

Main Result: related to Min Sum Set Cover [Feige et al. 2002]

Choose k, matroid: Related to Generalized Min Sum Set Cover [Bansal et al. 2010 & Skutella, Williamson 2011]

Maximization: Cannot approximate the Non-Adaptive using a Fully Adaptive within any constant.

Future directions

Our work: tradeoff adaptivity vs computational complexity

Future Directions:

- What can we approximate by fully adaptive strategies?
- Can we get adaptive algorithms for more general combinatorial problems?

Future directions

Our work: tradeoff adaptivity vs computational complexity

Future Directions:

- What can we approximate by fully adaptive strategies?
- Can we get adaptive algorithms for more general combinatorial problems?

Thank you!