Economics and Computation

Yingkai Li

EC4501/EC4501HM Semester 2, AY2024/25

Logistics

Instructor: Yingkai Li

Office: AS2 05-21

Office hour: by appointment.

Reading Lists

1. Aleksandrs Slivkins. *Introduction to Multi-Armed Bandits.* `https://arxiv.org/abs/1904.07272`
2. Jason Hartline. *Mechanism Design and Approximation.* `https://jasonhartline.com/MDnA/`
3. Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory.* `https://timroughgarden.org/notes.html`

Additional readings:

- Noam Nisan, Tim Roughgarden, Éva Tardos, Vijay V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press.
- Federico Echenique, Nicole Immorlica, Vijay V. Vazirani. *Online and Matching-Based Market Design.* Cambridge University Press.

Prerequisite

Required: Basics in probabilities, calculus, and how to prove formal theorems.

Not required: solid background knowledge about algorithm design (CS), mechanism design (Econ), or game theory (Econ). Coding is also not required.

Evaluations

- Two assignments (40%); due on Sep 29th, Nov 7th.
- Course project (30%); due on Oct 31th, mid-term review on Oct 6th.
- Final exam (30%); scheduled on Nov 21th, 5pm.
- Survey paper (25%); due on Nov 10th; only for HM students.

Syllabus

Week 1:	Preview of the course
Week 2/3/4:	Learning: bandits, experts, calibration
Week 5/6:	Learning in games
Week 7/8:	Mechanism design: welfare, revenue
Week 9/10:	Robust mechanism design
Week 11/12:	Topic courses: fairness, contracts, etc.
Week 13:	Project presentation by students

Course Philosophy

Course Philosophy

Practical problems are way too complex to solve!

Course Philosophy

Practical problems are way too complex to solve!

Classic economic approach: build a simple and representative model that is easy to solve.

Course Philosophy

Practical problems are way too complex to solve!

Classic economic approach: build a simple and representative model that is easy to solve.

Computer science approach: provide simple and practical solutions that are "good enough" for complex problems.

Course Philosophy

Practical problems are way too complex to solve!

Classic economic approach: build a simple and representative model that is easy to solve.

Computer science approach: provide simple and practical solutions that are "good enough" for complex problems.

This Course: focus on the computer science approaches in various economic problems.

A Simple Example: Complexity of Optimal Solutions

Knapsack problems:
n tasks, each task $i \in [n]$ requires a resource of c_i, and generates a value of v_i.

Objective: find a set of tasks to maximize the total value subject to a budget B on resources.

A Simple Example: Complexity of Optimal Solutions

Knapsack problems:
n tasks, each task $i \in [n]$ requires a resource of c_i, and generates a value of v_i.

Objective: find a set of tasks to maximize the total value subject to a budget B on resources.

It is NP-hard to compute the optimal solution.

- naïvely, the optimal solution can be found by enumerating all possible subsets, taking time $\exp(n)$, not practical.

A Simple Example: Complexity of Optimal Solutions

Knapsack problems:
n tasks, each task $i \in [n]$ requires a resource of c_i, and generates a value of v_i.

Objective: find a set of tasks to maximize the total value subject to a budget B on resources.

It is NP-hard to compute the optimal solution.

- naïvely, the optimal solution can be found by enumerating all possible subsets, taking time $\exp(n)$, not practical.
- NP-hard means that "no algorithm" can find the optimal solution much faster than that.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.
- choose either *greedy* or *max-value* solution to maximize the selected value.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.
- choose either *greedy* or *max-value* solution to maximize the selected value.

Theorem

The not so naïve greedy algorithm gives a solution that is at least half of the optimal value.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.
- choose either *greedy* or *max-value* solution to maximize the selected value.

Theorem

The not so naïve greedy algorithm gives a solution that is at least half of the optimal value.

An upper bound of the optimal: allowing partial allocation.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.
- choose either *greedy* or *max-value* solution to maximize the selected value.

Theorem

The not so naïve greedy algorithm gives a solution that is at least half of the optimal value.

An upper bound of the optimal: allowing partial allocation.

- greedy algorithm is optimal for allowing partial allocation.
- only the last task in the greedy algorithm may get a partial allocation.

A Simple Example: Simple Algorithms

Let us give up the optimal solution and only look for a "good enough" one.

A not so naïve greedy algorithm

- **greedy:** pick the tasks with decreasing order $\frac{v_i}{c_i}$ until budget runs out.
- **max-value:** pick one task with $c_i \leq B$ that maximizes the value.
- choose either *greedy* or *max-value* solution to maximize the selected value.

Theorem

The not so naïve greedy algorithm gives a solution that is at least half of the optimal value.

An upper bound of the optimal: allowing partial allocation.

- greedy algorithm is optimal for allowing partial allocation.
- only the last task in the greedy algorithm may get a partial allocation.

$$\text{Upper-Bound} \leq \text{Greedy} + \text{Max-Val} \Rightarrow \max\{\text{Greedy}, \text{Max-Val}\} \geq \frac{1}{2} \cdot \text{Upper-Bound}.$$

A Simple Example: Effects of Strategic Behavior

A Simple Example: Effects of Strategic Behavior

A naïve thinking: adding more roads should reduce the congestion (total driving time of the drivers).

A Simple Example: Effects of Strategic Behavior

A naïve thinking: adding more roads should reduce the congestion (total driving time of the drivers).

Braess's paradox [Pigou '20; Braess '68]

- adding more roads could lead to more severe congestions in strategic routing.

A Simple Example: Effects of Strategic Behavior

A naïve thinking: adding more roads should reduce the congestion (total driving time of the drivers).

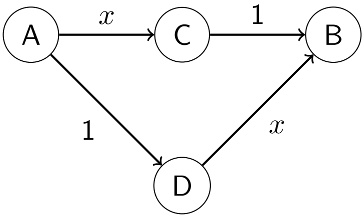
Braess's paradox [Pigou '20; Braess '68]

- adding more roads could lead to more severe congestions in strategic routing.

Example: agents travel from A to B.

- A $\rightarrow$ C, D $\rightarrow$ B: travel time x, fraction of travelers.
- A $\rightarrow$ D, C $\rightarrow$ B: travel time 1.

Network Before Adding Shortcut



A Simple Example: Effects of Strategic Behavior

A naïve thinking: adding more roads should reduce the congestion (total driving time of the drivers).

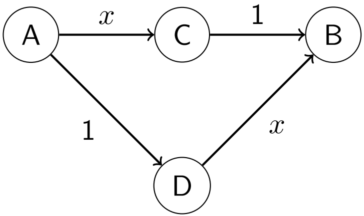
Braess's paradox [Pigou '20; Braess '68]

- adding more roads could lead to more severe congestions in strategic routing.

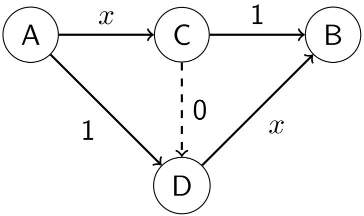
Example: agents travel from A to B.

- A $\to$ C, D $\to$ B: travel time x, fraction of travelers.
- A $\to$ D, C $\to$ B: travel time 1.
- New road in network: open a portal from C to D with zero travel time.

Network Before Adding Shortcut

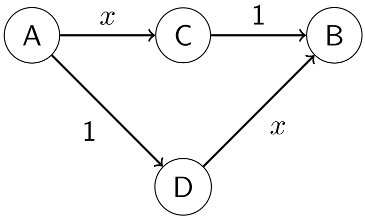


Network After Adding Shortcut

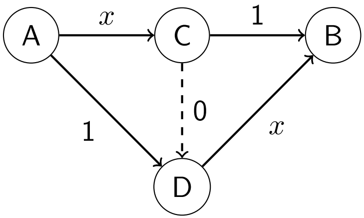


Braess's Paradox

Network Before Adding Shortcut

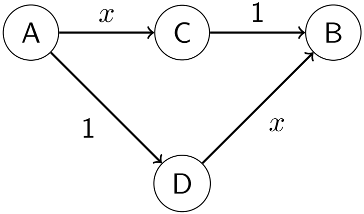


Network After Adding Shortcut

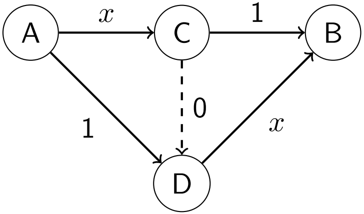


Braess's Paradox

Network Before Adding Shortcut



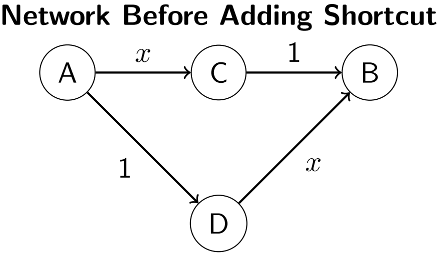
Network After Adding Shortcut

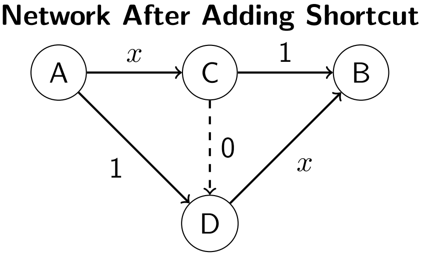


Equilibrium before shortcut: $\frac{1}{2}$ chooses A $\rightarrow$ C $\rightarrow$ B, $\frac{1}{2}$ chooses A $\rightarrow$ D $\rightarrow$ B.

- total travel time is $\frac{3}{2}$ for all agents.

Braess's Paradox





Equilibrium before shortcut: $\frac{1}{2}$ chooses A $\rightarrow$ C $\rightarrow$ B, $\frac{1}{2}$ chooses A $\rightarrow$ D $\rightarrow$ B.

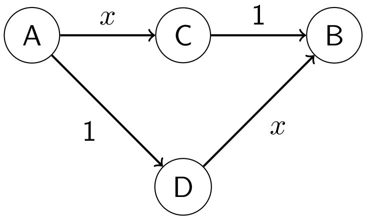
- total travel time is $\frac{3}{2}$ for all agents.

Equilibrium after shortcut: all agents choose A $\rightarrow$ C $\rightarrow$ D $\rightarrow$ B.

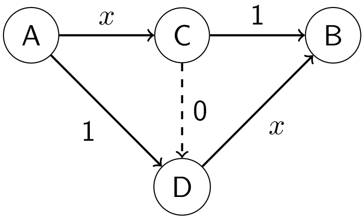
- total travel time is 2 for all agents.

Braess's Paradox

Network Before Adding Shortcut



Network After Adding Shortcut



Equilibrium before shortcut: $\frac{1}{2}$ chooses A $\to$ C $\to$ B, $\frac{1}{2}$ chooses A $\to$ D $\to$ B.

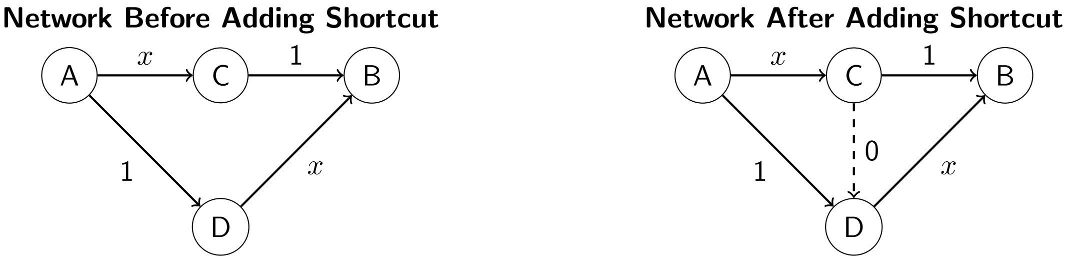
- total travel time is $\frac{3}{2}$ for all agents.

Equilibrium after shortcut: all agents choose A $\to$ C $\to$ D $\to$ B.

- total travel time is 2 for all agents.

$2 > \frac{3}{2}$: everyone suffers from having an additional shortcut!

Braess's Paradox



Equilibrium before shortcut: $\frac{1}{2}$ chooses A $\to$ C $\to$ B, $\frac{1}{2}$ chooses A $\to$ D $\to$ B.

- total travel time is $\frac{3}{2}$ for all agents.

Equilibrium after shortcut: all agents choose A $\to$ C $\to$ D $\to$ B.

- total travel time is 2 for all agents.

$2 > \frac{3}{2}$: everyone suffers from having an additional shortcut!

A sample question: how to design "good" mechanisms in complex strategic environments.

Methodologies

Methodology Overview

Economic analysis using algorithmic tools.

- approximation analysis: design and analysis of simple mechanisms in complex environments where finding the optimal is infeasible or undesirable.
- robust analysis: design robust mechanisms in the absence of detailed knowledge about the environment.
- data analysis: how to design good mechanisms with access to historical data.

Methodology Overview

Economic analysis using algorithmic tools.

- approximation analysis: design and analysis of simple mechanisms in complex environments where finding the optimal is infeasible or undesirable.
- robust analysis: design robust mechanisms in the absence of detailed knowledge about the environment.
- data analysis: how to design good mechanisms with access to historical data.

Goal: understand the design of good mechanisms in practical applications.

- online platforms (Google/Meta);
- resource allocations (FCC Spectrum/Land Resource/Cloud Computing);
- blockchains and cryptocurrencies (Bitcoin);
- recommendation system (Yelp/Netflix);
- etc.

Predicting the Future Without Knowing the Future

Weather forecast: in each day, predict the probability of rain tomorrow.

Predicting the Future Without Knowing the Future

Weather forecast: in each day, predict the probability of rain tomorrow.

A criterion for good forecasts: for each probability p, looking at all the days that the forecast is p, the actual frequency of rain almost matches p.

prediction	50%	50%	33.3%	50%	33.3%	33.3%	50%
outcome	rain	sunny	sunny	rain	rain	sunny	sunny

Predicting the Future Without Knowing the Future

Weather forecast: in each day, predict the probability of rain tomorrow.

A criterion for good forecasts: for each probability p, looking at all the days that the forecast is p, the actual frequency of rain almost matches p.

prediction	50%	50%	33.3%	50%	33.3%	33.3%	50%
outcome	rain	sunny	sunny	rain	rain	sunny	sunny

In future lectures, we will see an algorithm for making good forecasts without any information about the future.

Worst-case Approximations

- Algorithm / Mechanism M;
- Benchmark B;
- Set of possible inputs $\mathcal{F}$.

Worst-case approximation:

$$\mathrm{APX}(M) \triangleq \max_{F \in \mathcal{F}} \frac{B(F)}{M(F)}$$

$B(F)$ and $M(F)$ are the performance of the benchmark B and algorithm M respectively given input F.

Worst-case Approximations

- Algorithm / Mechanism M;
- Benchmark B;
- Set of possible inputs $\mathcal{F}$.

Worst-case approximation:

$$\mathrm{APX}(M) \triangleq \max_{F \in \mathcal{F}} \frac{B(F)}{M(F)}$$

$B(F)$ and $M(F)$ are the performance of the benchmark B and algorithm M respectively given input F.

Example: B: optimal value for the knapsack problem; M: not so naïve greedy algorithm.

$$\mathrm{APX}(M) = 2.$$

Worst-case Approximations

- Algorithm / Mechanism M;
- Benchmark B;
- Set of possible inputs $\mathcal{F}$.

Worst-case approximation:

$$\mathrm{APX}(M) \triangleq \max_{F \in \mathcal{F}} \frac{B(F)}{M(F)}$$

$B(F)$ and $M(F)$ are the performance of the benchmark B and algorithm M respectively given input F.

Example: B: optimal value for the knapsack problem; M: not so naïve greedy algorithm.

$$\mathrm{APX}(M) = 2.$$

How do we evaluate this approximation? Is 2 "good enough"?

Parametrized Instances

In often cases, we can consider the worst-case approximations in parameterized instances.

Parametrized Instances

In often cases, we can consider the worst-case approximations in parameterized instances.

Example: in the knapsack problem, one possible parameterization is the number of tasks n.

- This parameterization captures the size of the instance / input.

Parametrized Instances

In often cases, we can consider the worst-case approximations in parameterized instances.

Example: in the knapsack problem, one possible parameterization is the number of tasks n.

- This parameterization captures the size of the instance / input.

Asymptotic analysis: understand how the worst-case approximation guarantee scales with the instance size.

- $\mathrm{APX}(M; n)$: worst-case approximation of M when input size is n.

Asymptotic Analysis

- $f(n) = O(g(n)) : \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$;
- $f(n) = \Omega(g(n)) : \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$.
- $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$;
- $f(n) = o(g(n))$ if $f(n) = O(g(n))$ and $f(n) \neq \Omega(g(n))$;
- $f(n) = \omega(g(n))$ if $f(n) \neq O(g(n))$ and $f(n) = \Omega(g(n))$;

It measures the relative growth of the function with respect to n.

Asymptotic Analysis

- $f(n) = O(g(n)) : \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$;
- $f(n) = \Omega(g(n)) : \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$.
- $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$;
- $f(n) = o(g(n))$ if $f(n) = O(g(n))$ and $f(n) \neq \Omega(g(n))$;
- $f(n) = \omega(g(n))$ if $f(n) \neq O(g(n))$ and $f(n) = \Omega(g(n))$;

It measures the relative growth of the function with respect to n.

Example:

- $2n^2 + 8n + 100 = O(n^2)$;
- $16n^3 = o(2^n)$.
- $4n - 32 = \Theta(n)$.
- $\log(n) = o(n^\epsilon)$ for any constant $\epsilon > 0$.

Asymptotic Analysis

An algorithm M has a constant approximation if $\mathrm{APX}(M;n) = O(1)$.

- usually we view constant approximation as a good approximation since the worst-case performance does not degrade as the problem instance grows large ($n \to \infty$).

Asymptotic Analysis

An algorithm M has a constant approximation if $\mathrm{APX}(M;n) = O(1)$.

- usually we view constant approximation as a good approximation since the worst-case performance does not degrade as the problem instance grows large ($n \to \infty$).

Usually, an approximation is not ideal if it is a super-constant, i.e., $\mathrm{APX}(M;n) = \omega(1)$.

- E.g., $\mathrm{APX}(M;n) = \Theta(\log(n))$, or $\mathrm{APX}(M;n) = \Theta(n^2)$.

Asymptotic Analysis

An algorithm M has a constant approximation if $\mathrm{APX}(M;n) = O(1)$.

- usually we view constant approximation as a good approximation since the worst-case performance does not degrade as the problem instance grows large ($n \to \infty$).

Usually, an approximation is not ideal if it is a super-constant, i.e., $\mathrm{APX}(M;n) = \omega(1)$.

- E.g., $\mathrm{APX}(M;n) = \Theta(\log(n))$, or $\mathrm{APX}(M;n) = \Theta(n^2)$.

2-approximation to the optimal: Great! Same rate as the optimal!

Online Selection Problems

Problem: n items arriving online.

- item i has value $v_i \sim F_i$;
- the agent knows $F_1, \ldots, F_n$ at time 0.
- at time $i \leq n$, the agent observes value v_i and decides whether to select item i (if the selection has not been made).

Note: the arrival order of the items is unknown to the agent.

Online Selection Problems

Problem: n items arriving online.

- item i has value $v_i \sim F_i$;
- the agent knows $F_1, \ldots, F_n$ at time 0.
- at time $i \leq n$, the agent observes value v_i and decides whether to select item i (if the selection has not been made).

Note: the arrival order of the items is unknown to the agent.

How to make good decision without knowing the future?

Application: Job Hiring Process

A firm wants to hire for a vacant position.

- optimal policy: interview all the candidates, and selects the best one after the interviews.
- may not be feasible in certain scenarios, e.g., some candidates cannot wait long for the decisions.

Application: Job Hiring Process

A firm wants to hire for a vacant position.

- optimal policy: interview all the candidates, and selects the best one after the interviews.
- may not be feasible in certain scenarios, e.g., some candidates cannot wait long for the decisions.

Online Interview Process

- the candidates arrive in an online order;

Application: Job Hiring Process

A firm wants to hire for a vacant position.

- optimal policy: interview all the candidates, and selects the best one after the interviews.
- may not be feasible in certain scenarios, e.g., some candidates cannot wait long for the decisions.

Online Interview Process

- the candidates arrive in an online order;
- the firm observes the true quality of the current candidate, but not the quality of future candidates;
- the firm needs to make an immediate hiring decision for each candidate.

Application: Job Hiring Process

A firm wants to hire for a vacant position.

- optimal policy: interview all the candidates, and selects the best one after the interviews.
- may not be feasible in certain scenarios, e.g., some candidates cannot wait long for the decisions.

Online Interview Process

- the candidates arrive in an online order;
- the firm observes the true quality of the current candidate, but not the quality of future candidates;
- the firm needs to make an immediate hiring decision for each candidate.

Question: how to design good online hiring policies? What is the loss of adhering to online policies?

- designing the online policy is the same as the previous online selection problem.

Prophet Inequalities

How to evaluate the performance of an online policy?

Prophet Inequalities

How to evaluate the performance of an online policy?

Compare to a prophet who can foresee all future values.

- the prophet can guarantee an expected value of $\mathbf{E}[\max_i v_i]$. This is the benchmark.

Prophet Inequalities

How to evaluate the performance of an online policy?

Compare to a prophet who can foresee all future values.

- the prophet can guarantee an expected value of $\mathbf{E}[\max_i v_i]$. This is the benchmark.

Question: what are the performance guarantees using online policies compared to the prophet?

Prophet Inequalities

How to evaluate the performance of an online policy?

Compare to a prophet who can foresee all future values.

- the prophet can guarantee an expected value of $\mathbf{E}[\max_i v_i]$. This is the benchmark.

Question: what are the performance guarantees using online policies compared to the prophet?

Naive solution: randomly select a value (RS).

- the probability of choosing the highest value is $\frac{1}{n} \Rightarrow \mathrm{APX}(\mathrm{RS}) = n$.
- can we do better?

Prophet Inequalities

How to evaluate the performance of an online policy?

Compare to a prophet who can foresee all future values.

- the prophet can guarantee an expected value of $\mathbf{E}[\max_i v_i]$. This is the benchmark.

Question: what are the performance guarantees using online policies compared to the prophet?

Naive solution: randomly select a value (RS).

- the probability of choosing the highest value is $\frac{1}{n} \Rightarrow \mathrm{APX}(\mathrm{RS}) = n$.
- can we do better?

The designer cannot foresee the future values. How would she know whether to select the current value or not?

Threshold Policies

The designer knows the distribution of values and can predict the expected gain from the future if the current value is not selected.

- Intuitively, the designer should stop if the current value exceeds the predicted future value.

Threshold Policies

The designer knows the distribution of values and can predict the expected gain from the future if the current value is not selected.

- Intuitively, the designer should stop if the current value exceeds the predicted future value.

Simple policy in practice: threshold policies

- set threshold τ;
- at time i, selects item i if and only if $v_i \geq \tau$.

τ is an approximation of what the designer can gain in the future.

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2-approximation, i.e., it achieves expected value at least $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2*-approximation, i.e., it achieves expected value at least* $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

Consider threshold τ and let p_τ be the probability that an item is selected given τ.

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2-approximation, i.e., it achieves expected value at least $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

Consider threshold τ and let p_τ be the probability that an item is selected given τ.
The expected performance of the algorithm is

$$
\begin{aligned}
\mathrm{ALG}_\tau &= p_\tau \cdot \tau + \sum_{i \le n} \Pr[v_j < \tau, \forall j < i] \cdot \mathbf{E}\left[(v_i - \tau)^+\right] \\
&\ge p_\tau \cdot \tau + (1 - p_\tau) \cdot \sum_{i \le n} \mathbf{E}\left[(v_i - \tau)^+\right] \\
&\ge p_\tau \cdot \tau + (1 - p_\tau) \cdot \left(\mathbf{E}\left[\max_i v_i\right] - \tau \right)
\end{aligned}
$$

Last inequality holds since $\max_i v_i \le \tau + \max_i (v_i - \tau)^+ \le \tau + \sum_i (v_i - \tau)^+$.

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2*-approximation, i.e., it achieves expected value at least* $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

$$\mathrm{ALG}_\tau \geq p_\tau \cdot \tau + (1 - p_\tau) \cdot \left(\mathbf{E}\left[\max_i v_i\right] - \tau \right).$$

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2-approximation, i.e., it achieves expected value at least $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

$$\mathrm{ALG}_\tau \geq p_\tau \cdot \tau + (1 - p_\tau) \cdot \left(\mathbf{E}\left[\max_i v_i\right] - \tau \right).$$

- Mean Rule: Let $\tau = \frac{1}{2}\mathbf{E}[\max_i v_i]$. We have

$$\mathrm{ALG}_\tau \geq p_\tau \cdot \frac{1}{2}\mathbf{E}\left[\max_i v_i\right] + (1 - p_\tau) \cdot \frac{1}{2}\mathbf{E}\left[\max_i v_i\right] = \frac{1}{2}\mathbf{E}\left[\max_i v_i\right].$$

Prophet Inequalities

Theorem

There exists a threshold policy that achieves a 2-approximation, i.e., it achieves expected value at least $\frac{1}{2}\mathbf{E}[\max_i v_i]$.

$$\mathrm{ALG}_\tau \geq p_\tau \cdot \tau + (1 - p_\tau) \cdot \left(\mathbf{E}\left[\max_i v_i\right] - \tau \right).$$

- Mean Rule: Let $\tau = \frac{1}{2}\mathbf{E}[\max_i v_i]$. We have

$$\mathrm{ALG}_\tau \geq p_\tau \cdot \frac{1}{2}\mathbf{E}\left[\max_i v_i\right] + (1 - p_\tau) \cdot \frac{1}{2}\mathbf{E}\left[\max_i v_i\right] = \frac{1}{2}\mathbf{E}\left[\max_i v_i\right].$$

- Median Rule: Let τ such that $p_\tau = \frac{1}{2}$. We have

$$\mathrm{ALG}_\tau \geq \frac{1}{2}\tau + \frac{1}{2}\left(\mathbf{E}\left[\max_i v_i\right] - \tau \right) = \frac{1}{2}\mathbf{E}\left[\max_i v_i\right].$$

Hard Instances

Can we do better than 2? No!

Hard Instances

Can we do better than 2? No!

Example: two items.

- Item 1: $v_1 = 1$ with probability 1.
- Item 2: $v_2 = z$ w.p. $\frac{1}{z}$, and 0 otherwise.

Hard Instances

Can we do better than 2? No!

Example: two items.

- Item 1: $v_1 = 1$ with probability 1.
- Item 2: $v_2 = z$ w.p. $\frac{1}{z}$, and 0 otherwise.

Any Online Policy:

- If item 1 is chosen, the expected value is $v_1 = 1$.
- If item 1 is not chosen, the expected value is at most $\mathbf{E}[v_2] = 1$.

Hard Instances

Can we do better than 2? No!

Example: two items.

- Item 1: $v_1 = 1$ with probability 1.
- Item 2: $v_2 = z$ w.p. $\frac{1}{z}$, and 0 otherwise.

Any Online Policy:

- If item 1 is chosen, the expected value is $v_1 = 1$.
- If item 1 is not chosen, the expected value is at most $\mathbf{E}[v_2] = 1$.

Prophet: select item 1 if and only if $v_2 = 0$. The expected value of the prophet is $z \cdot \frac{1}{z} + (1 - \frac{1}{z}) \cdot 1 = 2 - \frac{1}{z}$.

Hard Instances

Can we do better than 2? No!

Example: two items.

- Item 1: $v_1 = 1$ with probability 1.
- Item 2: $v_2 = z$ w.p. $\frac{1}{z}$, and 0 otherwise.

Any Online Policy:

- If item 1 is chosen, the expected value is $v_1 = 1$.
- If item 1 is not chosen, the expected value is at most $\mathbf{E}[v_2] = 1$.

Prophet: select item 1 if and only if $v_2 = 0$. The expected value of the prophet is $z \cdot \frac{1}{z} + (1 - \frac{1}{z}) \cdot 1 = 2 - \frac{1}{z}$.

The gap is 2 when $z \to \infty$.

Application: Auctions

Auctions: a single item, n agents.

- each agent i has a private value $v_i \sim F_i$;

Application: Auctions

Auctions: a single item, n agents.

- each agent i has a private value $v_i \sim F_i$;

Efficiency maximization: allocate the item to the agent with the highest value.

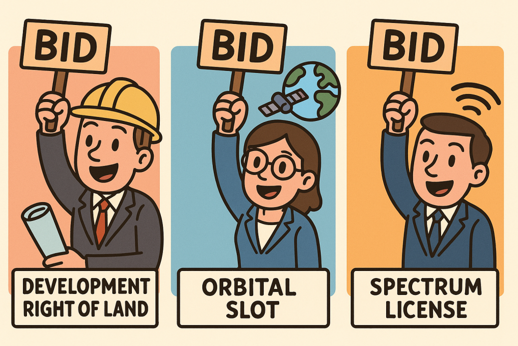
Application: Auctions

Auctions: a single item, n agents.

- each agent i has a private value $v_i \sim F_i$;

Efficiency maximization: allocate the item to the agent with the highest value.

- distributing scarce resource: spectrum license; display of Ad slots; development rights on lands; orbital slots for satellites; pollution permits; ...



Application: Auctions

Incentives matters in this problem!

- agents have incentives to misreport the true values as higher ones to win the item;

Application: Auctions

Incentives matters in this problem!

- agents have incentives to misreport the true values as higher ones to win the item;

A simple method for disciplining the strategic agents: using transfers:

- given price p_i for each agent i, item is sold only when $v_i \geq p_i$.

Application: Auctions

Incentives matters in this problem!

- agents have incentives to misreport the true values as higher ones to win the item;

A simple method for disciplining the strategic agents: using transfers:

- given price p_i for each agent i, item is sold only when $v_i \geq p_i$.

Posted pricing mechanism: given prices $\{p_i\}_{i\in[n]}$

- the item is sold to an agent with value $v_i \geq p_i$.
- tie-breaking π when there are multiple agents with high values.

Connection to Auctions

Prophet inequality: n items

- value distributions $F = F_1 \times \cdots \times F_n$;
- threshold τ;
- arrival order π.

Posted pricing mechanism: n agents

- value distributions $F = F_1 \times \cdots \times F_n$;
- price $p_i = \tau$ for each agent i;
- tie breaking rule π.

Connection to Auctions

Prophet inequality: n items

- value distributions $F = F_1 \times \cdots \times F_n$;
- threshold τ;
- arrival order π.

Posted pricing mechanism: n agents

- value distributions $F = F_1 \times \cdots \times F_n$;
- price $p_i = \tau$ for each agent i;
- tie breaking rule π.

Given any valuation profile $v = (v_1, \ldots, v_n)$, the selected value and the optimal value in both problems are the same.

- Posted pricing mechanism has a 2-approximation to the optimal welfare.

Basics on Game Theory

Incomplete Information Games

A static game with incomplete information is denoted as
$\Gamma_I = \left(N, (A_i)_{i\in N}, (u_i)_{i\in N}, (\Theta_i)_{i\in N}, \mu\right)$ where

- N is the set of players;
- A_i is the set of player i's actions; (what the agents can do)
- Θ_i is the set of player i's "types" where $\theta_i \in \Theta_i$ is private information of i; (what the agents know)
- $u_i : A \times \Theta \to \mathbb{R}$ is player i's payoff function (where $A = \times_{i\in N} A_i$, and $\Theta = \times_{i\in N} \Theta_i$).
- $\mu(\theta)$ is the probability that a type profile $\theta \in \Theta$ occurs.

Incomplete Information Games

A static game with incomplete information is denoted as
$\Gamma_I = \left(N, (A_i)_{i\in N}, (u_i)_{i\in N}, (\Theta_i)_{i\in N}, \mu\right)$ where

- N is the set of players;
- A_i is the set of player i's actions; (what the agents can do)
- Θ_i is the set of player i's "types" where $\theta_i \in \Theta_i$ is private information of i; (what the agents know)
- $u_i : A \times \Theta \to \mathbb{R}$ is player i's payoff function (where $A = \times_{i\in N} A_i$, and $\Theta = \times_{i\in N}\Theta_i$).
- $\mu(\theta)$ is the probability that a type profile $\theta \in \Theta$ occurs.

μ is called a common prior.

- Let μ_i denote the marginal distribution of μ on Θ_i, i.e., $\mu_i(\theta_i) \equiv \sum_{\theta_{-i}\in\Theta_{-i}} \mu(\theta_i, \theta_{-i})$.
- Let $\mu(\theta_{-i}|\theta_i)$ be the belief of agent i over θ_{-i} conditional on his type being θ_i.

Strategies and Bayesian Nash Equilibrium

A strategy of player i in Γ_I is a mapping $s_i : \Theta_i \to \Delta(A_i)$.

- s_i is a pure strategy if the mapping is deterministic, i.e., $s_i : \Theta_i \to A_i$. Let S_i be the set of pure strategies for i.

Strategies and Bayesian Nash Equilibrium

A strategy of player i in Γ_I is a mapping $s_i : \Theta_i \to \Delta(A_i)$.

- s_i is a pure strategy if the mapping is deterministic, i.e., $s_i : \Theta_i \to A_i$. Let S_i be the set of pure strategies for i.

Definition (BNE)

A strategy profile s is a Bayesian Nash Equilibrium if for any agent i and any type θ_i (such that $\mu_i(\theta_i) > 0$), for any action a_i^* in the support of $s_i(\theta_i)$, we have

$$a_i^* \in \operatorname*{argmax}_{a_i \in A_i} \sum_{\theta_{-i} \in \Theta_{-i}} \mu(\theta_{-i}|\theta_i) \cdot \mathbf{E}_{a_{-i} \sim s_{-i}(\theta_{-i})}[u_i(a_i, a_{-i}, \theta)] \,.$$

Informal definition of BNE: all agents are doing the best they can given what they think others are doing.