Computer Science 136: Economics and Computation

Austin Li
awli@college.harvard.edu

Fall 2021

Abstract

These are notes[1] for Harvard's *Computer Science 136*, a class on algorithmic game theory, as taught by Professor David C. Parkes[2] in Fall 2021.

Course description: The interplay between economic thinking and computational thinking as it relates to electronic commerce, social networks, collective intelligence and networked systems. Topics covered include: game theory, peer production, reputation and recommender systems, prediction markets, crowd sourcing, network influence and dynamics, auctions and mechanisms, privacy and security, matching and allocation problems, computational social choice and behavioral game theory. Emphasis will be given to core methodologies, with students engaged in theoretical, computational and empirical exercises.

Contents

[1] With thanks to Eric K. Zhang for the template.

[2] With teaching fellows Annie Zhu, Mark York, Jeff Jiang, Esther Plotnick, Sai Srivatsa Ravindranath, Zhou Fan, Kevin Huang, Alex Iansiti, Richard Xu, and Jimmy Lin.

1 September 1st, 2021

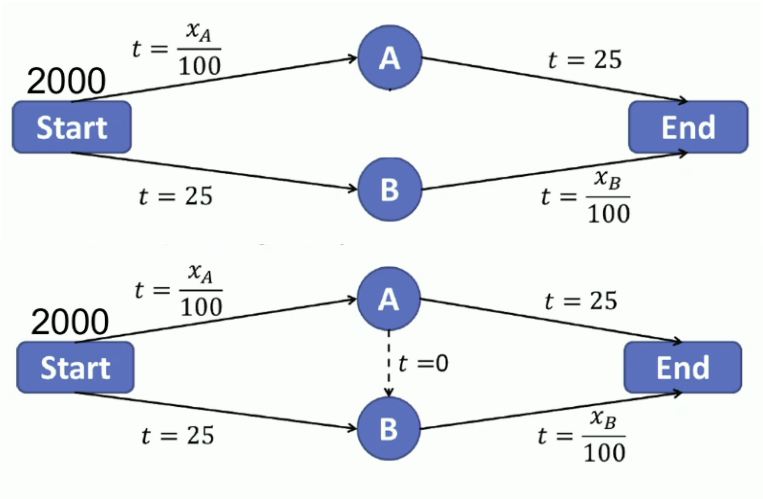
This course emphasizes fundamental principles. By the end of the course we will learn how to model, analyze, and design online platforms where incentives matter.

The class uses a multi-disciplinary approach, combining computer science, economics, and applied math. It is formal, using theoretical proof-based reasoning. And it is collaborative! Teamwork makes the dream work.

We will return to the idea that **incentives matter**. The success or failure of a system depends on how it is used.

Example 1.1 (Braess's paradox)**.** The observation that adding one or more roads to a road network can slow down overall traffic flow through it. Can model as a network flow problem.

Consider the following two situations:



The *optimum* flow **in both systems** is half going to node A and the other half going to node B – the inclusion of a "super highway" does not affect the optimal solution that costs 35.

Equilibrium in the first situation is the same as the optimum. In the second case, the equilibrium is SABE, which costs 40 minutes. This illustrates the paradox; equilibrium increases cost by 5 under rational behavior.

Example 1.2 (Access to PDP-1 computer at Harvard)**.** See lecture notes!

1.1 Syllabus

The full syllabus can be found here.

We will go over theory in the first third of the semester and then move onto more applied material in the next two thirds.

Grading will be based on

- Assignments: 45% with the lowest programming assignment dropped
- Midterms: 30%
- Final project: 15%
- Participation: 10%

We are given **six late days**, with two-day maximum on any problem set.

Assignments are due **Fridays at 5pm EST**. There will be **three programming assignments**, problem sets 0, 2, and 6. Sets 2 and 6 will be pair assignments and all three will have an in-class tournament. Winners will get a prize!

Final projects will be done in groups of 1-3, and will be exploration based. Project proposals are due Wednesday November 17 with short presentations on Monday December 1 and Wednesday December 3. Final papers are due Friday December 6 at 5pm EST.

Sections are **optional** and will be held in person. Office hours will be a hybrid of in-person and Zoom.

1.2 Game theory: the prisoners' dilemma

Game theory is useful in modeling rational software/computer agents and less relevant for actual people, who are not completely rational.

Two agents are held in prison and can either cooperate (C) or defect (D). The payoff matrix is given by

		Player 2 C	Player 2 D
Player 1	C	$(-2,-2)$	$(-5,0)$
	D	$(0,-5)$	$(-4,-4)$

We note that the Nash equilibrium is the outcome (D, D), but this is **not** the optimum payoff.

We are interested answering in two questions:

- How do we **represent** a game with a large number of players?
- How do we **compute** a Nash equilibrium?

1.3 Mechanism design

Mechanism design is inverse game theory. We are trying to design the rules of the game such that the equilibrium of the game has certain properties, e.g. we choose the payoffs.

Example 1.3 (Locating a student center)**.** If we position the student center at the mean of student preferences. Players/students can over estimate their preferred location to move the mean to their actual desired location. This mechanism is **not strategy-proof**.

Solution: Use the **median** mechanism.

The is a new problem though, and it turns out that the optimization problem for mechanisms is **NP-hard**, so we need strategy proofness and scaleable algorithms.

2 September 8th, 2021

2.1 Game theory

Computer scientists are interested in game theory for modeling, for computational complexity. Parkes stresses the agent, design, and the theoretical perspective of the equilibrium objects in game theory.

In this course, N will be the set of agents, A_i will be the set of actions for agent i, and $u_i(a_i, a_{-i}) \in \mathbb{R}$ will be the utility function for agent i. a_{-i} is the set of actions for all agents **not** i. In this case, $N = \{1, 2\}, A_1 = A_2 = \{W, G\}$.

We will denote equilibria by using asterisks and think of the equilibrium action as a vector $\boldsymbol{a}^* = (a_1^*, a_2^*)$.

Definition 2.1 (Pure strategy Nash equilibrium (PSNE))**.** An action $\boldsymbol{a}^*$ is a PSNE if

$$u_1(a_1^*, a_2^*) \geq u_1(a_1, a_2^*), \qquad \forall a_1 \in A_1 \tag{1}$$

and

$$u_2(a_1^*, a_2^*) \geq u_2(a_1^*, a_2), \qquad \forall a_2 \in A_2. \tag{2}$$

For the existence of a Nash equilibrium, we require:

- All agents are rational
- Knowledge of the payoff matrix (everyone knows that everyone knows what the payoffs are)
- No collusion

Example 2.2 (Chicken)**.** Consider a game of two players with two actions, wait (W) or go (G), where the payoff matrix is given by

		Player 2	
		W	G
Player 1	W	$(0, 0)$	$(0, 2)$
	G	$(2, 0)$	$(-4, -4)$

In this game, the PSNE are (W, G) and (G, W) outcomes. Recall that this game is written in **normal form representation** where the first coordinate corresponds to the row player and the second coordinate corresponds to the column player.

2.2 Randomization

Definition 2.3 (Mixed strategy Nash equilibrium (MSNE))**.** A pair of mixed strategies (s_1^*, s_2^*) such that

$$u_1(s_1^*, s_2^*) \geq u_1(s_1, s_2^*), \qquad u_2(s_1^*, s_2^*) \geq u_1(s_1^*, s_2) \tag{3}$$

for all s_1 and s_2, respectively.

We can think of another way of solving this game.

2.3 Best-response correspondence

We can plot the effect of the strategy of player one on the response of player two. We can do this for the other player as well, what the best correspondence of player one is to the strategy of player two. The Nash equilibria are precisely the **points of intersection**.

See slides.

Theorem 2.4 (Nash, 1950)**.** *There exists MSNE in every finite simultaneous-move game.*

2.4 Correlated equilibrium

There are two stories that can describe a correlated equilibrium. We can consider our game of chicken.

2.4.1 Signal and contingent action

Suppose we introduce a signal that is sampled uniformly at random, say a traffic light and we condition our actions on the signal.

For example, if player 1 goes on 0 and waits on 1 and vice versa for player 2. We can write this as a joint distribution, denoted p^*. We will see a correlation and the distribution is given by

		Player 2	
		W	G
Player 1	W	0	0.5
	G	0.5	0

2.4.2 Mediator

Consider a mediator that samples p^* and recommends actions. Based on recommendations, each player's actions will be correlated.

Definition 2.5 (Correlated equilibrium)**.** For all a_1 in the support of p^* and all $a_1' \in A_1$

$$\sum_{a_2 \in A_2} p_2^*(a_2|a_1)u_1(a_1, a_2) \geq \sum_{a_2 \in A_2} p_2^*(a_2|a_1)u_1(a_1', a_2). \tag{4}$$

This also holds similarly for player 2.

Theorem 2.6. *Correlated equilibrium form a convex set. Given $p \in CE$ and $\tilde{p} \in CE$, then $\alpha p + (1-\alpha)\tilde{p} \in CE$ where $\alpha \in [0,1]$. Here, $CE = \{$correlated equilibria$\}$.*

See Figure 2.30 for example of a game where there are convex equilibria that are outside the convex combination of mixed strategy and pure strategy equilibria.

2.5 Representations

If you take a game theory course, you will see mostly Normal form representations of games, think matrices and n-cubes.

Let there be n agents and m actions. In Normal form, we need $n \times m^n$ cells to represent the game. This is the **space complexity** of Normal form. Observe that it scales *exponentially* in the number of players.

2.5.1 Congestion game

Let R be the set of resources. Let actions be subsets of R. Let $c_r(x)$ be a cost function where x is the number of players that use resource r. The cost for any player is linear in the resources used.

Example 2.7 (Parties game)**.** Let there be three types of parties: dance parties, quiet parties, and Yardfest. We can plot the cost as a function of the number of people that attend.

Your action A_i is any subset, say you go to a dance party and Yardfest. We can define utility as

$$u_i(a_1, \cdots, a_n) = -\sum_{r \in a_i} c_r(x_{ra}), \qquad ra = \#\text{ of people who used the resources.} \tag{5}$$

Note. The congestion game representation is **not fully expressive** representation, but it is compact representation. Moreover, this type of representation is size $|R| \times (n+1)$. There are $|R|$ resources and $(n+1)$ numbers to describe the function.

Note. Exists a PSNE in a congestion game.

3 September 13th, 2021

Recall different representations of games, where there are n agents, m actions and $|R|$ resources. Note that

1. Normal form — size nm^n and is **expressive**
2. Congestion — size $|R|(n+1)$ and is **not** expressive
3. Agent-graph — size $nm^{(d+1)}$ (see below) and is **expressive**
4. Action-graph — $m(n+1)^d$ (see below) and is **expressive**

3.1 Agent-graph representation

Note that the *agent-graph representation* is a game represented by an **undirected graph** where agents are vertices and where the utility function for on agent (as a function of all agents) is a function of the agent and any connected agents on the graph. For example, in a game

$$1 - 2 - 3 - 4, \tag{6}$$

the utility (or payoff) of player 1 is given by

$$u_1(a_1, a_2, a_3, a_4) = g_1(a_1, a_2) \tag{7}$$

where g_1 is some function. Note that if the maximum degree of any vertex on the graph is d, the matrix of the utility function for node 1 for example has size m^2. Then in general, an agent-graph representation of a game has size $nm^{(d+1)}$.

3.2 Action-graph representation

Each agent plays an action, where the utility of each agent depends on **the number of agents that play the same action** and **the number of agents who play adjacent actions on the graph**. Each player cares about the **count** of players that play the other actions.

For example, in class, we have

$$u_1(\text{sandwich}, a_2, \ldots, a_n) = u_{\text{sandwich}}(\#\text{ sandwich}, \#\text{ cake}). \tag{8}$$

Note that we need a matrix size $(n+1)^2$ to represent u_{sandwich} because at most $n+1$ players can choose each option. It is actually less than this, but asymptotic behavior is preserved. Then if the maximum degree is d, the size of the representation is size $m(n+1)^d$.

In the action-graph representation, you can construct actions that are **unique** to each player, which allows you to capture any game that you want.

Note. You can derive an action graph from any agent graph. Different representation graphs are natural in different settings.

3.3 Sequential move games

Consider a **new entrant** game where player one chooses to go **in** or **out**. Player two can enter and either **fight** or **cooperate**, where the payoff is defined at each vertex.

Note that we can represent this game as a **tree** – we will naturally recover the "natural" equilibrium by following the tree and maximizing payoff for both players.

We can write the Normal-form representation of this game as

$$\begin{pmatrix} -1,-1 & 2,2 \\ 0,5 & 0,5 \end{pmatrix} \tag{9}$$

We can define the **history** of a game H as a set of all possible given by

$$H = \{ \underbrace{\varepsilon, (\text{in})}_{P(n)\in\{1,\dots,n\}}, \underbrace{(\text{out}), (\text{in}, F), (\text{out}, C)}_{\text{terminal } Z, u_i(n)} \}. \tag{10}$$

Definition 3.1 (Strategy and utility)**.** We define the *strategy* of player i at history h is given by

$$s_i(h) \in A_i(n) \forall n : P(h) = i \tag{11}$$

and the **utility** u for player i as

$$u_i(s_1, \dots, s_n). \tag{12}$$

Definition 3.2 (Nash equilibrium (NE))**.** A *Nash equilibrium* s^* is a strategy such that

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \tag{13}$$

for all i, for all s_i. Note that s_{-i} is the strategies of all other players.

Definition 3.3 (Subgame perfect equilibrium (SPE))**.** Let Γ_h be a subgame of game Γ at history h for non-terminal h and $u_i(\boldsymbol{s}|\Gamma_h)$ be the utility of player i with strategy vector $\boldsymbol{s}$. Then a *subgame perfect equilibrium* $s^* \iff s^*$ is a Nash Equilibrium in every subgame.

We have removed "incredible threads."

3.4 The single deviation principle

Definition 3.4 (Single deviation (SD))**.** s_i' is a single deviation (SD) at history h if s_i' differs from s_i **only at** h.

Definition 3.5 (Usefulness)**.** A SD s_i' is *useful* if $u_i(s_i', s_{-i}|\Gamma_h) > u_i(s_i, s_{-i}|\Gamma_h)$.

Theorem 3.6 (Single deviation principle)**.** *s^* is a SPE in a finite extensive form game $\iff \nexists$ no useful SD for any player.*

Proof. ($\implies$) By definition of SPE.

($\impliedby$) Suppose no useful SD, but a useful multi-step deviation at some h. Consider the minimal multi-step deviation s_i' and consider the longest history at which $s_i'(h) \neq s_i^*(h)$. This is a useful SD at that history h, a contradiction. □

4 September 15th, 2021

4.1 Repeated games

Definition 4.1 (Stage game)**.** A stage game $G = (N, \tilde{A}, \tilde{u})$ is the game that we repeat over and over again in our repeated game.

Note. We use tildes to differentiate between the actions and utility of the stage game and the repeated game.

We will repeat G a finite or infinite number of times. We will use *histories* $h = (\boldsymbol{a}^{(0)}, \boldsymbol{a}^{(1)}, \cdots, \boldsymbol{a}^{(k)}, \cdots)$ to keep track of everything that has happened, where $\boldsymbol{a}^{(i)} \in \tilde{A}$ are action profiles. We will allow for mixed strategies, where $s_i(h) \in \Delta(\tilde{A}_i)$ is the strategy of player i at strategy h and Δ represents a distribution simplex over the actions.

We can talk about finite games G^T where we repeat G a T number of times, or infinite games G^∞.

4.1.1 Finite games

The utility function for player 1 is given by

$$u_1(s_1, s_2) = \sum_{k=0}^{T-1} \tilde{u}_1(\boldsymbol{a}^{(k)}). \tag{14}$$

Note that the overall utility is the sum of the utilities of the stage game over all time periods.

4.1.2 Infinitely repeated games

We will use the idea of discounting. The utility is given by

$$u_1(s_1, s_2) = \sum_{k=0}^{\infty} \delta^k u_1(\boldsymbol{a}^{(k)}), \qquad \delta \in (0, 1). \tag{15}$$

The discount factor δ can be interpreted as uncertainty of future events. Mathematically, this is convenient for the infinite sum to converge.

Theorem 4.2. *A subgame perfect equilibrium s^* $\iff$ no useful single deviation. This holds for both finite and infinitely repeated games.*

Definition 4.3 (Open loop strategy)**.** Strategy s_i is *open loop* if $s_i(h) = s_i(h')$ for all $|h| = |h'|$.

Example 4.4 (Prisoners' dilemma open loop strategy)**.** Consider the action profiles given by

$$(D, D), \qquad (C, C), \qquad (D, D), \qquad (C, C), \qquad \cdots \tag{16}$$

Definition 4.5 (Stage-Nash)**.** A strategy profile s is *stage-Nash* if $s(h) \in \mathrm{NE}(G)$ for all h.

Example 4.6 (Prisoners' dilemma open loop stage-Nash strategy)**.** Consider the action profiles given by

$$(D, D), \qquad (D, D), \qquad (D, D), \qquad (D, D), \qquad \cdots \tag{17}$$

Theorem 4.7. *Open loop and stage Nash $\implies$ subgame perfect equilibrium.*

Proof. Via single deviation principle. Want to argue that there is no useful single deviation. Note that

1. Does not affect future utility because we are playing a Nash of the entire game
2. Does not help now because we are at a stage Nash!

□

Note. Open loop and stage Nash $\implies$ SPE $\implies$ NE.

Definition 4.8 (Enforceable)**.** Action profile $\boldsymbol{a} \in \tilde{A}$ is *enforceable* if there exists a (mixed) Nash equilibrium x^* of stage game such that $v_i = \tilde{u}_i(\boldsymbol{a}) > \tilde{u}_i(x^*) = e_i$

Theorem 4.9 (Nash threats folk theorem)**.** *There exists a subgame perfect equilibrium of G^∞ for some δ_0 and $\delta \geq \delta_0$: an enforceable profile $\boldsymbol{a}$ is played in equilibrium.*

Proof. Consider strategy

$$s_i = \begin{cases} \text{plays } a_i & \text{if } \boldsymbol{a} \text{ played so far} \\ \text{plays } x_i^* & \text{otherwise} \end{cases}. \tag{18}$$

By SDP and case analysis.

Case 1: No deviation. We can calculate the utility given by $v_{\max} + \frac{\delta e_i}{1-\delta} \leq \frac{v_i}{1-\delta}$. □

4.2 Peer-to-peer (P2P) systems

In traditional file-sharing systems, as the number of users increases, the download rate decreases. In P2P systems, the solution is much more scalable and download rate ideally does not decrease.

P2P systems are scalable, resilient to failure, fair, have good performance.

Some definitions:

Definition 4.10 (Protocol)**.** A *protocol* is the "rules of the game," e.g. messages that can be sent, available upload/download actions.

Definition 4.11 (Peer)**.** A *peer* is a participant in the network, an agent.

Definition 4.12 (Client)**.** A *client* is code that sends messages, takes actions. A client is a strategy.

Definition 4.13 (Reference client)**.** A *reference client* is the recommended client design, the strategy that people would like you to play.

4.2.1 Gnutella (2000)

The Gnutella network is truly decentralized. Peer gets a list of IP addresses of peers from some set of known peers. You can send a "query message" to peers and your peers send a "query response" who has a desired file. This is a distributed infrastructure to get files you want.

Note. Gnutella has a **free rider problem**, where agents may download from others, but may not let others download from you in order to protect bandwidth or security reasons.

4.2.2 BitTorrent (2001)

The **key innovation** is to split up documents into pieces and turn this into a repeated game.

1. The user goes to a directory service that gives a URL that points to a .torrent file (metadata)

2. This metadata includes a URL to a tracker (a computer that keeps tracks of participants in a swarm — a group of users who are sharing a file with each other). The client will request 50 IP addresses of users in the swarm

3. Each file is broken up into pieces and blocks. Each peer is keeping a *bitfield* that tracks the pieces it has received

4. Between each user, we exchange bitfields and engage in a download/upload protocol. They will also upload "have" statements to let other users know which pieces the user has

   Download protocol: Ask everyone and the reference client will say "download the rarest pieces first"

   Upload: In reference client, divide upload capacity into four pieces. One piece is used for *optimistic unchoke* — randomly unchoke (allow one peer to download from you). Every 10 seconds, prioritize the top three peers for you (who are letting you download from them) in the other three upload slots

As the number of users increases, the capacity increases.

5 September 20th, 2021

5.1 Algorithmic game theory

Using algorithms to solve games.

We have **polynomial time results** for two player zero sum games, pure Nash strategies, and correlate equilibrium

We also have that FindNash is PPAD-complete. Conjecture that FindNash is an intractable computational problem.

Definition 5.1 (Big-Oh)**.** We say that $T(z) \in O(g(z))$ if $\exists k, z_0 > 0 : T(z) \leq kg(z) \forall z > z_0$.

Definition 5.2 (Linear programs)**.** Recall from CS 124: LPs can be solved in polynomial time in the number of variables and constraints.

We will sometimes talk about linear *feasibility* programs (LFPs), a system of linear equations without an optimization constraint where we seek to find a solution.

5.1.1 Two player zero sum games

We can solve games using a *maximin* or *minimax* strategy.

Maximin is given by

$$\bar{x} \in \text{argmax}_{x \in \Delta A_1} \left[\min_{a_2} u_1(x, a_2) \right] \tag{19}$$

and minimax is given by

$$\underline{x} \in \text{argmin}_{x \in \Delta A_1} \left[\max_{a_2} u_1(x, a_2) \right]. \tag{20}$$

The first case we are trying to minimize player 2's harm and the second case we are trying to prevent player 2 from doing very well.

Theorem 5.3. *Any maximun (minimax) for player 1 and player 2 is a Nash equilibrium, and they will all have the same value.*

5.2 Noisy Prisoner's dilemma results

We then went over the results of the in-class tournament for the Prisoners' dilemma game.

Note. Cooperative agents tended to perform better than the non-cooperative agents. We note that the automaton that performed worse never leave the (D, D) loop, so the payoff is not maximized.

5.3 Calculating pure strategy Nash equilibria

We have our input a Normal form game, and an output either a PSNE or FALSE.

For a two player game with m actions, there is a $m \times m$ Normal form representation. To check if one action profile is a PSNE, we have $2(m-1)$ actions across the row and column, and we need to check m^2 profiles. The time complexity, therefore is $O(m^2 [2(m-1)])$. The input is size m^2.

For an n player game, we similarly have an input size nm^n and the checking taking $m^n [n(m-1)]$ time, which is *polynomial* in the size of the input.

5.4 Calculating mixed strategy Nash equilibria

Our input is a Normal form game, and our output is a mixed s^*.

Recall the **support enumeration method**, where we enumerate action-set pairs $X \in A_1, Y \in A_2$ and call CheckNash(X, Y) for MSNE.

Time: there are $(2^m - 1)^2$ subsets to check, with some polynomial function for CheckNash $\implies (2^m - 1)^2 \times$ poly(m) time. Our input is size $2m^2$.

5.4.1 CheckNash(X, Y)

Output is a mixed strategy profile (x, y):

1. Support$(x) \subseteq X$ and Support$(y) \subseteq Y$
2. Given Y, player 1 is indifferent for all $j \in X$ and weakly prefers $j' \notin X$
3. Given X, player 2 is indifferent for all $k \in Y$ and weakly prefers $k' \notin Y$
4. or FALSE

Theorem 5.4. *CheckNash will return a Nash equilibrium or FALSE. Moreover, it **will** return a Nash equilibrium if $\exists$ a Nash equilibrium with support X, Y. And finally, this runs in polynomial time.*

Consider the algorithm for y, analogous for x. We have an LFP given by:

1. $\sum_{k \in A_2} y_k = 1, y_k \geq 0$ and $y_k = 0 \forall k \notin Y$
2. $\sum_{k \in A_2} u_1(j, k) y_k = v_1 \forall j \in X$ and $\sum_{k \in A_2} u_1(j, k) y_k \leq v_1 \forall j \notin X$

Note property 2 says that the expected utility for player 1 for any action j in X given the probabilities picked by player 2 is equal to v_1, and is weakly smaller than v_1 for all actions outside of X.

Note. We have $m + 1$ variables and $2m + 1$ constraints. The input size for the two player game is $2m^2$.

This tells us the reason that calculating a MSNE is hard is that it is **hard to find the right support pair**. CheckNash runs quickly, but the combinatorics of finding the correct pair makes it difficult.

Note. For notation, we will use index i for agents, j for actions of player 1, and k for actions of player 2.

5.4.2 CheckNash, $n > 2$

Observe that equation 2 in our LFP becomes

$$\sum_{k \in A_2} \sum_{l \in A_3} u_1(j, k, l) y_k z_l = v_1 \forall j \in X \tag{21}$$

Note that we have a product of y, z, which is **not** linear. The CheckNash problem ceases to be a LFP for $n > 2$. This can be solved quickly, but not in polynomial time. We will see that finding Nash is intractable, even for $n = 2$.

Next time, we will show that we can write a LP to calculate correlated equilibria directly $\implies$ correlated equilibrium is polynomial.

6 September 22nd, 2021

6.1 Correlated equilibria

Why is it that we can compute correlated equilibria in polynomial time?

The input is a Normal form game, the output a correlated equilibrium.

Some notation:

1. $p^*(j,k)$ is the joint probability
2. $p_1^*(j|k), p_2^*(k|j)$ are conditional probabilities
3. $p_1^*(j), p_2^*(k)$ are marginal probabilities. E.g. thee marginal probability of player 1 for W is the sum of the rows.

Definition 6.1 (Correlated equilibrium (CE))**.** The correlated equilibrium for player 1 satisfies

$$\sum_{k\in A_2} u_1(j,k)p_2^*(k|j) \geq \sum_{k\in A_2} u_1(j',k)p_2^*(k|j) \qquad \forall j : p_1^*(j) > 0, \forall j' \tag{22}$$

Example 6.2 (Chicken)**.** Consider a game of chicken

		Player 2 W	Player 2 G
Player 1	W	$(0,0)$	$(0,2)$
	G	$(2,0)$	$(-4,-4)$

with a correlated equilibrium p^*

		Player 2 W	Player 2 G
Player 1	W	$4/18$	$11/18$
	G	$2/18$	$1/18$

Consider player 1 playing G. We have

$$p_2^*(W|G)u_1(G,W) + p_2^*(G|G)u_1(G,G) \geq p_2^*(W|G)u_1(W,W) + p_2^*(G|G)u_1(W,G). \tag{23}$$

Substituting values gives

$$\frac{2}{3}\times 2 + \frac{1}{3}\times(-4) \geq \frac{2}{3}\times 0 + \frac{1}{3}\times 0. \tag{24}$$

We are going to multiply through by $p_1^*(j)$ and obtain

$$\sum_{k\in A_2} u_1(j,k)\underbrace{p_1^*(j)p_2^*(k|j)}_{p^*(j,k)} \geq \sum_{k\in A_2} u_1(j',k)\underbrace{p_1^*(j)p_2^*(k|j)}_{p^*(j,k)}, \qquad \forall j, j' \tag{25}$$

Note. We do not use the condition that $p_1^*(j) > 0$ because in this case, both sides vanish automatically by the equation.

Moreover, recall $P(B|A)P(A) = P(A,B)$ and we can simplify this as in the above.

6.2 CEs as LFPs

We can represent the correlated equilibrium problem as a linear feasibility program.

The variables are p_{jk}. Note there are $m \times m$ variables. Then we have for player 1

$$\sum_k u_1(j,k)p_{jk} \geq \sum_k u_1(j',k)p_{jk} \qquad \forall j, j' \tag{26}$$

and for player 2, we have

$$\sum_j u_2(j,k)p_{jk} \geq \sum_j u_2(j,k')p_{jk} \qquad \forall k, k' \tag{27}$$

with conditions

$$\sum_j \sum_k p_{jk} = 1, \qquad p_{jk} \geq 0. \tag{28}$$

Note that this is a linear program. In order to find the correlated equilibrium that is maximally fair or maximizes the value of the participants, we can do this by solving the LFP.

Note. There are m^2 variables and $3m^2 + 1$ constraints. The input size is $2m^2$. We can solve this LFP in polynomial time.

For an n-player correlated equilibrium, we would have m^n variables and $nm^2 + 1 + m^n$. The input size in the normal form is nm^n.

Note. If we were looking for Nash equilibria, instead of the p_{jk} in our equations above, we would have $p_{jk} \to x_j y_k$ because we have row mixing and column mixing. We have introduced additional decision variables, making our problem **nonlinear**.

In our case, looking at correlated equilibrium and introducing more variables makes the problem simpler to solve. We can think of this as a *relaxation* of a Nash equilibrium. NEs insist that actions of agents are **independent**. CEs relax this constraint and alternatively insist that actions between players are correlated.

The correlated set is a *superset* of the Nash set!

6.3 Complexity theory primer

We are going to discuss the complexity of the FindNash problem.

We reviewed the complexity class NP. In short, it is a complexity class of *decision problems* where we can check *certificates*, or solutions to the problem, in polynomial time. We cannot solve these problems on a one-tape Turing machine in polynomial time.

We also reviewed the class NP-Hard. Colloquially, NP-hard problems are "as hard as any problem in NP." We show that problems are NP-hard via polynomial-time reductions **to** $X, X \in$ NP-hard. E.g. 3-$COLORABLE \xrightarrow{\text{poly}} X$. Take an instance of 3-$COLOR$ write it as an instance of X.

Moreover, X is NP-complete if $X \in$ NP and X NP-hard.

We also review the complexity class $P = \bigcup_k \text{TIME}(O(n^k))$.

It is conjectured that $P \overset{?}{\neq} NP$.

Definition 6.3 (Polynomial parity argument, directed graph (PPAD))**.** $X \in$ PPAD if $X \xrightarrow{\text{poly}} EOTL$. where $EOTL$ is the end of the line problem.

X is PPAD-hard if $EOTL \xrightarrow{\text{poly}} X$.

6.4 End of the line problem

The EOTL problem is defined on a directed graph with the "parity property" and with a "standard source" $\underline{o}$.

The parity property says that each vertex has 1 parent or 1 child $\implies$ no isolated vertices. A standard source is a source that is also in the input.

The input is a directed graph an the output is either a sink or a non-standard source.

Note. Because of the parity property, having a source $\implies$ a solution to the problem. Moreover, we note that there are an *odd* number of solutions. We note that cycles do not contribute any solutions. Any other source $\to$ sink provides an additional two solutions.

It is conjectured that PPAD $\overset{?}{\neq}$ P.

To formulate the EOTL problem more rigorously.

Input. There are labels $\ell \in \{0,1\}^r$. We are given the following polynomial time functions

1. IsVertex(ℓ) returns TRUE or FALSE
2. Succ(ℓ)
3. Pred(ℓ)
4. The standard source $\underline{o}$

We cannot directly find vertices. We can only generate a label ℓ in the set of labels and check if it is a vertex. If so, we can query ℓ for its successor and predecessor. E.g. we can find a vertex and walk in either direction.

Note that the set of vertices V is related to the set of labels L by $V \subseteq L$.

These functions provided are represented as "circuits." We insist that the size of the circuits is polynomial in r.

Some intuition. This problem is **hard** because the number of labels is *exponential in* r, so we cannot check all the labels. Moreover, there exists a path from $\underline{o}$ to a solution, but this path may be exponentially long.

Theorem 6.4. *FindNash is PPAD-complete.*

We need to show that FindNash $\xrightarrow{\text{poly}}$ $EOTL$ and $EOTL \xrightarrow{\text{poly}} BROUWER \xrightarrow{\text{poly}} FindNash$. We will be using gadgets to transform $BROUWER \to FindNash$.

Theorem 6.5 (Brouwer fixed point theorem)**.** *Let* $f : [0,1]^3 \to [0,1]^3, f$ *continuous. Then* $\exists x : f(x) = x$.

7 September 27th, 2021

We recall the complexity diagram. We have the relationships P $\subseteq$ PPAD $\subseteq$ NP. NP-hard problems are at least as hard as NP and PPAD problems.

We note the correction that NP-hard $\subseteq$ PPAD-hard.

Recall the conjectures NP $\neq$ P, PPAD $\neq$ P, and PPAD $\subsetneq$ NP.

7.1 End of the line (EOTL)

Recall EOTL is defined on a directed parity-property graph with a standard source $\underline{o}$. We seek to return a sink or a nonstandard source.

We want to show that

1. FindNash $\xrightarrow{\text{poly}}$ EOTL
2. EOTL $\xrightarrow{\text{poly}}$ FindNash

Example 7.1. Consider the normal form game

		Player 2	
		L	R
Player 1	U	$0, 1$	$6, 0$
	M	$2, 0$	$5, 2$
	D	$3, 4$	$3, 3$

Recall we have (D, L) a pure-Nash equilibria.

Let m_i be the number of actions for player i, $\sigma(s_i) \subseteq A_i$ be the supports of the strategies of player i, and $br_i(s_j) \in A_i$ be the best response of player i to the strategy of player j.

Definition 7.2 (Labels)**.** A direct product (ℓ_1, ℓ_2) is a label if

$$\ell_1 \subseteq A_1 \cup A_2, \qquad |\ell_1| = m_1, \qquad \ell_2 \subseteq A_1 \cup A_2, \qquad |\ell_2| = m_2 \tag{29}$$

Then a *label* $\boldsymbol{\ell}$ is a vector length $m_1 + m_2$ where the first m_1 entries correspond to actions of player 1 in A_1 and the last m_2 entries correspond to the actions of player 2 in A_2.

Definition 7.3 (Vertices)**.** Given a base action a_b, a label is a *vertex* if

1. (Almost) completeness: $\ell_1 \cup \ell_2 \cup \{a_b\} = A_1 \cup A_2$
2. $\ell_1 = A_1$ or $\exists s_1 : \sigma(s_1) = A_1 \setminus \ell_1$ and $br_2(s_1) = A_2 \cap \ell_1$
3. $\ell_2 = A_2$ or $\exists s_2 : \sigma(s_2) = A_2 \setminus \ell_2$ and $br_1(s_2) = A_1 \cap \ell_2$

We can define the *standard source* $\underline{o} = (A_1, A_2)$. Vertices ℓ and ℓ' are *neighbors* if they differ in exactly one action.

We discussed an example of drawing a PPAD representation of this game. We note that there is **exactly one** vertex away from our standard source. Following this construction, we will end and start at the Nash equilibria.

We will be able to argue that we can

1. Give an LFP to check for an s_1 that makes ℓ_1 valid $\implies$ be able to find its neighbors quickly
2. Establish the parity property $\implies$ odd number of NE in any game

3. Any sink or source is completely labeled

Theorem 7.4. *Any complete vertex that is* ***not*** *the standard source is a Nash equilibrium.*

Proof. Let $\ell = (\ell_1, \ell_2) \neq \varrho$ be a complete vertex. Then any $a_1 \in A_1$ is present. We have two cases:

Case 1: $a_1 \in \ell_1 \implies a_1 \notin s_1, a_1 \notin \ell_2 \implies a_1 \notin br_1(s_2)$.

Case 2: $a_1 \notin \ell_1 \implies a_1 \in s_1, a_1 \in \ell_@ \implies a_1 \in br_1(s_2)$. □

7.2 Gadgets

We now seek to show that FindNash is PPAD-hard. We want to show EOTL $\xrightarrow{\text{poly}}$ BROUWER $\xrightarrow{\text{poly}}$ FindNash.

In the computational BROUWER problem, we will represent F as a circuit.

We will show that Nash equilibria can do calculations.

The central idea: the Brouwer problem involves calculations involving a circuit. We will convert the calculations of a circuit to solving a game. This is the essence of this reduction.

7.2.1 Multiplication gadget

The multiplication gadget is a four-player agent graph game with actions $A_i = \{0, 1\}$. We write $x_i \in [0, 1]$ to represent the probability that player i players 1.

The utility functions are given by $u_1(a_1) = 0, u_2(a_2) = 0$ because players 1 and 2 do not depend on other agents. For u_3, u_4, we have

$$u_3(\boldsymbol{a}) = \begin{cases} a_1 \wedge a_2, & a_3 = 0 \\ a_4, & a_3 = 1 \end{cases}, \qquad u_4(a_3, a_4) = a_3 \oplus a_4. \tag{30}$$

Our expected utility for player 3 is given by $\langle u_3 \rangle = x_1 x_2 + x_4$.

We claim that

1. Any $x_1, x_2, x_4 = x_1 x_2$ is a Nash equilibrium.

   Proof. Consider $x_1, x_2, x_3 = \frac{1}{2}, x_4 = x_1 x_2$. Players 1 and 2 are indifferent (utility 0) $\implies$ player 3 plays 50/50 $\implies$ player 4 indifferent and can play $x_1 x_2$. □

2. In any Nash equilibrium, $x_4 = x_1 x_2$.

   Proof. Proceed by contradiction. Suppose $x_4 > x_1 x_2 \implies x_3 = 1 \implies x_4 = 0$. Suppose $x_4 < x_1 x_2 \implies x_3 = 0 \implies x_4 = 1$. We have contradictions in both cases. □

7.3 Auctions

There are *single item* or multi-item auctions. There are *sealed bid* or iterative auctions.

There are different kinds of values. We will focus on *private values*, or personal values for items. Others are *common values*, where everyone has same value for an item, but don't know what it's worth – for example, oil. We also have *interdependent values*, where your value is dependent on other people, like art.

There are different design objectives for auctions, including efficiency or revenue. We will focus on efficiency. Efficiency is about allocating items to maximize value in equilibrium.

There are also different *types* of auctions. Second-price sealed-bid (SPSB), First-price sealed-bid (FPSB), AllPay where the highest bidder wins but everyone is charged.

We can define our agents $N = \{1, 2, \cdots, n\}$ with private values $v_i \sim G_i$, sometimes sampled from a distribution function G_i.

There are auction allocation rules, which take bids $x(b_1, \cdots, b_n) \in \{0, 1\}^n$ and payment rules $t(b_1, \cdots, b_n) \in \mathbb{R}^n$ that induce utilities $u_i(b_1, \cdots, b_n) = v_i - p$ where p the price.

7.3.1 Second price sealed bid auctions (SPSB)

Suppose $b_1 \geq b_2 \geq \cdots \geq b_n$ with rules

$$x_i(b) = \begin{cases} 1, & i = 1 \\ 0, & \text{otherwise} \end{cases}, \qquad t_i(b) = \begin{cases} b_2, & i = 1 \\ 0, & \text{otherwise} \end{cases}. \tag{31}$$

Theorem 7.5. *Truthful bidding is a dominant strategy equilibrium for SPSB auctions.*

We are left with the phrase "strategy-proof." SPSB auctions are strategy-proof, meaning that each agent doesn't need to "game" how to bid; rather, you should *want to be truthful* about your private value.

8 September 29th, 2021

One third the way into the semester! Some review: we have done game theory, we will now focus on designing games/systems with *good* equilibrium.

Some takeaways: simultaneous move games, sequential move games, subgame perfect equilibrium, compact representations, complexity classes, solving zero sum games, solving general sum games (we cannot do), solving for correlated equilibria!

8.1 Auctions

Let N be our set of bidders, v_i and b_i be the value and bid for player i.

Definition 8.1 (Strategy)**.** A *strategy* for an auction is a function $b_i = s_i(v_i)$ that takes values to bids. The values may be sampled from some distribution $v_i \sim G_i$.

Definition 8.2 (Dominant strategy equilibrium (DSE))**.** A set of strategies $s_1^*, s_2^*, \cdots, s_n^*$ is a DSE $\iff$ $u_i(s_i^*(v_i), b_{-i}) \geq u_i(b_i, b_{-i})$ for all i, v_i, b_i, b_{-i}.

Definition 8.3 (Bayes Nash equilbrium)**.** An equilbrium of the game is that you bid in a way that maximizes your *expected utility* given the distribution of the bids of everyone else.

More formally, $s_1^*, s_2^*, \cdots, s_n^*$ is a BNE $\iff$

$$\mathbb{E}_{v_{-i}}\left[u_i\left(s_i^*(v_i), s_{-i}^*(v_{-i})\right)\right] \geq \mathbb{E}_{v_{-i}}\left[u_i\left(b_i, s_{-i}^*(v_{-i})\right)\right], \quad \forall i, v_i, b_i. \tag{32}$$

Theorem 8.4. *$s_i^*(v_i) = \frac{n-1}{n} v_i$ is a BSE of a FPSB auction for $v_i \overset{i.i.d}{\sim} U(0,1)$*

Proof. Guess and verify. Guess that the equilibrium strategy is a linear function, an ansatz of $s_1^*(v_1) = \alpha v_1, s_2^*(v_2) = \alpha v_2, 0 < \alpha < 1$. Suppose player 2 plays this. WLOG consider $b_1 \leq \alpha$. Then

$$\langle u_1 \rangle = (v_1 - b_1) P(b_2 \leq b_1) = (v_1 - b_1)\frac{b_1}{\alpha}. \tag{33}$$

We check first order and second order conditions. Take $\partial_{b_1}\langle u_1\rangle = 0$ to find optimal b_1^* and $\partial_{b_1 b_1} < 0$. If this is true, we have found the optimal bid that maximizes utility. □

Definition 8.5 (Efficiency)**.** An auction is *efficient* if it allocates an item to the person with the highest value in the BNE. The equilibrium preserves the ordering of the values.

8.1.1 Revenue (SPSB)

Definition 8.6 (Order statistics)**.** Let $z_1, \cdots, z_n$ be i.i.d random variables and $z^{(k)}$ be the kth largest variable. We call $z^{(k)}$ the *kth order statistic.*

Note. Given $z_1, \cdots, z_n \overset{\text{i.i.d}}{\sim} U(0,1)$. Then

$$E(z^{(k)}) = \frac{n - (k-1)}{n+1}. \tag{34}$$

Suppose bidder v_i wins. We want to know the expected payment of this bidder. Condition on our bid, we know that all other players have bids distributed $U(0, v_i)$. Then

$$\langle b_i \rangle = \frac{n-1}{n} v_i. \tag{35}$$

8.1.2 Revenue (FPSB)

Suppose bidder v_i wins. Then the expected payment is given by

$$\langle b_i \rangle = \frac{n-1}{n} v_i \tag{36}$$

(from above).

So the expected revenue in the SPSB and FPSB auctions are **the same**. In SPSB, we bid truthfully, but we don't pay everything we bid. In FPSB, we shave down to adjust for personal risk and we pay the same amount!

Note. All auctions — FPSB, SPSB, TPSB (third price), all-pay etc. — have the **same revenue** and are *efficient*.

Definition 8.7 (Interim allocation)**.** This is the *probability of winning* in equilibrium is given by

$$x_i^*(v_i) = \mathbb{E}_{v_{-i}} \left[x_i \left(s_i^*(v_i), s_{-i}^*(v_{-i}) \right) \right]. \tag{37}$$

Definition 8.8 (Interim payment)**.** This is an agent's *payment* in equilibrium given by

$$t_i^*(v_i) = \mathbb{E}_{v_{-i}} \left[t_i \left(s_i^*(v_i), s_{-i}^*(v_{-i}) \right) \right]. \tag{38}$$

Theorem 8.9 (Bayes Nash equilbrium characterization)**.** *In any BNE of any auction, we have*

1. *The interim allocation $x_i^*(v_i)$ is monotone non-decreasing in v_i.*
2. *The payment rule is*

$$t_i^*(v_i) = v_i x_i^*(v_i) - \int_{z=0}^{v_i} dz \; x_i^*(z) + t_0. \tag{39}$$

Some conclusions:

1. If x_i^* the same, then t_i^* the same $\implies$ if two auctions are efficient, they have the same revenue.
2. Can use this to derive a BNE of *new* auctions, which is a consequence of the expected payments being the same.

8.1.3 Deriving BNE

Step 1: Guess that the auction is efficient at s^*.

Step 2: Derive s^* such that interim payment is equal to the interim payment in the truthful equilibrium of the SPSB.

Step 3: Confirm equilibrium and confirm s^* is efficient.

Example 8.10 (FPSB BNE)**.** Suppose we are trying to derive the BNE of a $U(0,1)$ FPSB. We have

$$t_i^*(v_i) = P(\text{win}) \times s_i^*(v_i) = P(v_i \geq v_j)_{i \neq j} s_i^*(v_i) = v_i^{n-1} s_i^*(v_i) \tag{40}$$

$$= t_{i,\text{SPSB}}^*(v_i) = v_i^{n-1} \times \frac{n-1}{n} v_i \implies \boxed{s_i^*(v_i) = \frac{n-1}{n} v_i.} \tag{41}$$

Example 8.11 (All pay auction BNE)**.** We note that the interim payment is just our strategy because we always charge the amount

$$t_i^*(v_i) = s_i^*(v_i) = v_i^{n-1} \times \frac{n-1}{n} v_i. \tag{42}$$

We note that auctions can be *sealed* (SPSB, FPSB, all pay) or they can be *multi-round* (ascending clock, eBay, descending clock).

There are a whole family of auction elaborations that are *not* sealed-bid but turn out to be "equivalent" to the sealed auctions. For example, ascending clock $\cong$ SPSB and descending clock $\cong$ FPSB from equilibrium structure. This is better described/in more detailed in a the reading!

9 October 4th, 2021

9.1 Mechanism design

Imagine that you are a "social planner" or designer and we have to make a decision. Individuals have individual preferences for the decision and may *misreport* their preferences.

A mechanism design problem is the problem of mapping inputs (preferences) to outputs (a decision or allocation).

Example 9.1 (Meeting scheduling)**.** Meeting scheduling is a mechanism design problem. Consider the preference orderings

$$10 >_1 11 >_1 9, \qquad 9 >_2 11 >_2> 10, \qquad 11 >_3 9 >_3 10. \tag{43}$$

We can use the *plurality mechanism* where we take the top choices and choose the one with the most overlap. We can break ties by taking the earlier times. There is a tie between 9, 10, 11 and the mechanism will return 9.

Note that player 1 can misreport $\hat{u}_1 = 11 >_1 10 >_1 9$ and the mechanism will return 11, which is more preferable to player 1. This is a **useful misreport**.

Definition 9.2. A *strategy-proof* (SP) or *dominant-strategy incentive-compatible* (DSIC) is a mechanism *without* a useful misreport.

Example 9.3 (SP mechanisms)**.** A simple SP mechanism is a constant mechanism. A dictator mechanism is strategy proof.

Definition 9.4 (Direct mechanism)**.** Let $\hat{\boldsymbol{u}} = (\hat{u}_1, \cdots, \hat{u}_n)$ be the utility functions, A is the set of alternatives, and g an output function. A *direct mechanism* M is a function $M : \hat{\boldsymbol{u}} \to g(\hat{\boldsymbol{u}}) \in A$. Note that in direct mechanisms, the messages we send describe our utility functions.

Definition 9.5 (Strategy proof mechanisms)**.** A mechanism M is *strategy-proof* (SP) if truthful reporting is a dominant strategy equilibrium.

Definition 9.6 (Strategy)**.** A strategy is a function $s_i : u_i \to \hat{u}_i$. That is, a strategy is a function that takes utilities into misreports.

Definition 9.7 (Implementation)**.** Suppose Mhas strategies $s_1^*, \cdots, s_n^*$ in a dominant strategy equilibrium. M *implements* a function $f(u_1, \cdots, u_n) = g(s_1^*(u_1), \cdots, s_n^*(u_n))$.

The main question in mechanism design is determining what functions we can *implement* to derive decisions from true utilities.

9.1.1 Revelation principle

Suppose M' an arbitrarily complex mechanism that takes strategies and returns an alternative a. M' does not necessarily need to be direct.

The revelation principle states that we can simulate the entire system — the strategies of the agents and M'. It asks each agent for a report $\hat{u}_i$ and commits to simulating M' on the equilibrium strategies of each player to return an alternative a as an output.

That is, the new outcome function for M is

$$g(\hat{u}) = g_{M'}(s^*(\hat{u})) \tag{44}$$

Theorem 9.8. *M is strategy proof.*

Corollary 9.8.1. *Any function f that can be implemented in some mechanism can be implemented in a SP mechanism.*

Example 9.9. If we have software that bids optimally for you in the eBay iterative auction, if we can simulate the software *and* the auction, this system is strategy proof.

Note (Some intuition)**.** Let s^* the dominant strategy. If a mechanism plays s^* for each agent, each agent is incentivized to be truthful because M simulates each agent's best strategies. If there existed a useful misreport, then s^* is not the best strategy.

The revelation principle reduces the design problem entirely.

9.2 VCG mechanism

Let $v_i : A \to \mathbb{R}$ be a *valuation function.* A value is the willingness to pay (in dollars) for different alternatives.

Consider a mechanism M given by

$$M : \hat{\boldsymbol{v}} \to (x(\hat{\boldsymbol{v}}), t_i(\hat{\boldsymbol{v}})) \in A \times \mathbb{R} \tag{45}$$

that takes values and returns a *choice function* $x(\hat{\boldsymbol{v}}) \in A$ and a payment function $t_i(\hat{\boldsymbol{v}}) \in \mathbb{R}$. We can think of these functions as analogous to the allocation and payment functions in an auction.

The utility $u_i = \hat{v}_i - p$ where p the price.

Definition 9.10 (VCG mechanism)**.** We can define

$$a^* \in \text{argmax}_a \sum_i \hat{v}_i(\boldsymbol{a}) \tag{46}$$

be the alternative that maximizes the sum of values over all agents, and

$$a^{-i} \in \text{argmax}_a \sum_{j \neq i}^{i} \hat{v}_j(\boldsymbol{a}) \tag{47}$$

be the alternative that maximizes the total value over all agents except i.

The choice and payment rules are given by

$$x(\hat{\boldsymbol{v}}) = a^*, \qquad t_i(\hat{\boldsymbol{v}}) = \sum_{j \neq i} \hat{v}_j(a^{-i}) - \sum_{j \neq i} \hat{v}_j(a^*), \tag{48}$$

that is, each player pays the opportunity cost incurred by the other players.

Example 9.11 (Auctions as VCG mechanisms)**.** Consider three agents a_1, a_2, a_3 with values $\hat{v}_1 = 10, \hat{v}_2 = 8, \hat{v}_3 = 6$ in a single item auction. Here, $a^* = \hat{v}_1$. In this case, we can compute payments:

$$t_1(\hat{\boldsymbol{v}}) = 8 - 0 = 8, \qquad t_1(\hat{\boldsymbol{v}}) = 10 - 10 = 0, \qquad t_1(\hat{\boldsymbol{v}}) = 10 - 10 = 0. \tag{49}$$

We have found a second price auction!

Theorem 9.12. *VCG is strategy-proof.*

Proof. Fix $\hat{v}_{-1}$ and consider the utility of agent 1. The utility is given by

$$u_1 = \underbrace{v_1(a^*) + \sum_{j \neq 1} \hat{v}_j(a^*)}_{(*)} - \sum_{j \neq 1} \hat{v}_j(a^{-1}). \tag{50}$$

Note that the first two terms are affected by agent 1's report. Then consider what report we can make to maximize $(*)$. Recall that $a^* \in \text{argmax}_a \hat{v}_1(\boldsymbol{a}) + \sum_{j \neq 1} \hat{v}_j(\boldsymbol{a})$. Thus a_1 should be truthful because a^* will be chosen to maximize the agent's value and the values of everyone else. Given truthful values, the mechanism will choose the alternative that maximizes your value and everyone else's values, namely a^*. □

9.3 Taxation principle

The taxation principle is a way to think about what makes a mechanism strategy-proof.

Theorem 9.13. *(x, t) is strategy proof $\iff$ exists a family $\{p_i : A \times V_{-i} \to \mathbb{R}\}_i$ such that*

1. *Agent-independent*

$$t_i(\hat{\boldsymbol{v}}) = p_i(a, \hat{v}_i) \tag{51}$$

   for $a = x(\hat{\boldsymbol{v}})$ for all i
2. *Agent-optimizing*

$$x(\hat{\boldsymbol{v}}) \in \text{argmax}_a \left[\hat{v}_i(\boldsymbol{a}) - p_i(a, \hat{v}_{-i})\right] \tag{52}$$

   for all i.

The first property says that the amount an agent pays is dependent on the pricing function, which is dependent only on the alternative the agent picks and the values of all other players.

Example 9.14 (Single item SPSB auction)**.** Consider three players with bids $b_1 = 10, b_2 = 8, b_3 = 6$. Note

Prices	Win	No
1	8	0
2	10	0
3	10	0

Table 1: SPSB auction

that the win price is dependent only on the agent's actions and the alternative chosen.

10 October 6th, 2021

10.0.1 Review

Recall some notation. We have A the set of alternatives and valuations $v_i(\boldsymbol{a})$. Moreover, we have a choice rule $x(\hat{\boldsymbol{v}}) \in A$ and a payment rule $t(\hat{\boldsymbol{v}}) \in \mathbb{R}^n$ that maps reported valuations to an alternative and payments.

Moreover, recall the VCG mechanism, with choice and payment rules given by

$$x(\hat{\boldsymbol{v}}) \in \text{argmax}_{\boldsymbol{a}} \sum_i \hat{v}_i(\boldsymbol{a}), \qquad t_i(\hat{\boldsymbol{v}}) = \sum_{j \neq i} \hat{v}_j(\hat{\boldsymbol{a}}^{-i}) - \sum_{j \neq i} \hat{v}_j(\boldsymbol{a}^*) \tag{53}$$

where the first condition is called "efficiency" in the economic sense that we maximize the sum of all values. The payment rule states that the winner pays the negative externality it places on the system.

The VCG mechanism is **strategy-proof**.

Moreover, VCG is *individual rational* (IR) if the utility is non-negative $\implies$ players are incentivized to participate.

Note. The main problem for computer scientists is to solve $n+1$ optimization problems to find $\boldsymbol{a}^*, \boldsymbol{a}^{-i}$, of which could be NP-hard! Note there could be an exponentially large number of alternatives.

We can also consider a "VCG-based" mechanism that uses an *approximation algorithm*. We will consider different problems in the following sections.

10.1 Multi-item auctions

Consider m *identical* items with values $\{(w_i, k_i)\}_{i \in N}$ where w_i is the value and k_i is the number of items.

Note (0-1 knapsack problem)**.** This is known as the **0-1 knapsack problem**. We have a knapsack (bag) size m, each object has a value w_i and a size k_i. We want to maximize total value of items that we can fit into the bag. This problem is NP-hard.

10.1.1 Greedy algorithm

We can sort items by decreasing $\hat{w}_i/k_i$ (the "bang per buck") and add items to the bag until no other items fit.

Example 10.1 (Greedy kanpsack)**.** Consider our bag that can only hold five items. The values are given by $(1, \$6), (2, \$5), (5, \$12), (3, \$7)$. Using a greedy approach, we take the first two items and do not take the other two.

Using a VCG-based mechanism, we have payments given by

$$t_1 = 12 - 5 = 7, \qquad t_2 = 13 - 6 = 7, \qquad t_3 = 11 - 11 = 0, \qquad t_4 = 11 - 11 = 0. \tag{54}$$

Problem. Suppose 4 bids \$100. Then $\boldsymbol{a}^* = \{1, 4\}$ and $a^{-4} = \{1, 2\}$ and $t_4 = 11 - 6 = 5$. Note that agent 4 is simply making the greedy algorithm do what it was supposed to. This shows that VCG-based mechanisms are not working. We need something else.

10.2 Single parameter domains

Idea. Valuations $v_i : A \to \mathbb{R}$ can be described through a single number.

Example 10.2 (Multi-item auction)**.** $v_i = w_i$ if allocated, $v_i = 0$ otherwise. Then we only need to specify w_i to determine the valuation function.

Example 10.3 (TV advertising)**.** $v_i = w_i \times$ eyeballs.

Example 10.4 (Uber driver)**.** $v_i = -$cost per hour $\times$ trip time.

Definition 10.5 (Single parameter domains (SPDs))**.** Suppose we have a known summarization function $q_i(\boldsymbol{a}) \in \mathbb{R}_{\geq 0}$ and private values $w_i \in \mathbb{R}$, the model in a *single parameter domain* is that $v_i(\boldsymbol{a}) \equiv w_i \times q_i(\boldsymbol{a})$.

Example 10.6 (Binary domain)**.** A *binary domain* is given by $q_i(\boldsymbol{a}) = 1$ if win and 0 otherwise.

Definition 10.7. Let $x(\hat{\boldsymbol{w}})$ a choice rule. We say x is *monotone non-decreasing* $\iff$

$$\hat{w}_i' > \hat{w}_i \implies q_i\left(x(\hat{w}_i'), \hat{w}_{-i}\right) \geq q_i\left(x(\hat{w}_i), \hat{w}_{-i}\right). \tag{55}$$

Theorem 10.8. *On a single parameter domain (SPD), mechanism (x, t) is strategy-proof if*

1. *The choice rule $x(\hat{\boldsymbol{w}})$ is monotone non-decreasing*
2. *The payment identity is given by*

$$t_i(\hat{w}_i, \hat{w}_{-i}) = \hat{w}_i \times q_i(\boldsymbol{a}) - \int_{z=\underline{w}}^{\hat{w}_i} q_i\left(x(z, \hat{w}_{-i})\right). \tag{56}$$

Note. Observe that

1. Greedy is monotone non-decreasing.
2. We can use the payment identity to charge, which makes the mechanism strategy-proof!

Note (Summary)**.** VCG is great, but can involve NP-hard problems. If we plug an approximation into VCG, we can break strategy-proofness.

However, if we are in a SPD and we can design polynomial monotone algorithms, then we can use the payment identity and we obtain a strategy-proof mechanism.

10.3 C-approximations

A c-approximation if $W_{\text{opt}}/W_{\text{app}} \leq c$. We can approximate the optimal solution/values. We seek to understand some properties of approximations.

Example 10.9. Suppose there are q items, $(1, \$2)$. Then $W_{\text{opt}}/W_{\text{app}} = q/2$. Note that this diverges as $q \to \infty$.

Example 10.10 ("Smart greedy")**.** If $W_{\text{high}} > W_{\text{greedy}} \implies$ allocate highest. Otherwise allocate greedy.

Note that this algorithm is monotone increasing and is a 2-approximation.

Proof. Suppose we have m items and let us order the input $\frac{w_1}{k_1} \geq \frac{w_2}{k_2} \geq \cdots \geq \frac{w_n}{k_n}$. Suppose they don't fit. Then define $j : \sum_{i=1}^{j-1} k_i \leq m$ and $\sum_{i=1}^{j} k_i > m$. Note that

$$W_{\text{opt}} \leq \sum_{i=1}^{j} w_i \leq \sum_{i=1}^{j-1} w_i + \max_i\{w_i\} \leq W_{\text{greedy}} + W_{\text{high}} \leq 2\max\{W_{\text{greedy}}, W_{\text{high}}\} = 2W_{\text{approx}} \tag{57}$$

□

10.4 P2P tournament results and notes

Fifth place. diversifying requests for rare pieces and BitTyrant with optimistic unchoking.

Fourth place. Reference client but unchoking every ten instead of every three.

Third place. PropShare with three-round weighting with estimation with experiments.

Second place. Standard design with 7 slots.

First place. PropShare with optimized unchoking and experiments.

10.5 Midterm reminder

Midterm next next Wednesday! Materials will be sent out soon.

11 October 13th, 2021

Today, our guest lecturer is Ellie from the Embedded Ethics program. We will be shifting towards an ethics-based portion of the course to learn how to design ethical mechanisms.

11.1 Review

A *mechanism* has

1. A set of possible *outcomes*
2. A set of *self-interested participants*
3. Each participant has *privately* held preferences
4. A system or assigning those 1 to 2 based on the *disclosed* preferences

Mechanism design is different from normal game theory because agents can misreport.

11.2 Fairness in mechanism design

Today, we will focus on fairness and ethics questions in mechanism design. We can consider equality of **resources**, equality of **opportunity**, and **fair** equality of opportunity.

The central question is: how do we design a *fair* mechanism when agents are lying?

11.2.1 Equal resources

Everyone gets equal resources.

We will consider the following example for today's exercises.

Example 11.1 (Alice in Wonderland)**.** Consider Alice and her friends in Wonderland. They all fall into a lake, some up to their ankles and others up to their shoulders. They assemble at the foot of the Dodo bird to dry off and he hands **everyone a medium-sized towel**.

Example 11.2 (Stimulus checks)**.** Everyone getting stimulus less than $\$100,000$ received a \$1200 check, regardless of recent employment status.

Implementation of the checks was determined by income bracket.

There are concerns with **accessibility**, concerns with agents **deserving more** than others, and sometimes, it is **fair to lose to others**.

11.2.2 Equal opportunity

Everyone enters same decision procedure, which may allocated different resources to different people. Resources are not allocated based on artificial barriers or prejudices.

Example 11.3 (Alice in Wonderland)**.** Alice and her friends all fall into a lake. A Dodo bird tells them to race for a prize. To wine, they must run until they become dry. The prize is given to the person who arrives back first.

Everyone has the same chance to win, but not everyone has the same outcome.

Example 11.4 (1999 Boston mechanism)**.** Compulsory busing introduced in 1974. Boston moved to controlled choice procedures. We are looking at the 1999 Boston Mechanism.

Real preference order included sibling attendance and proximity to the school (walk zone). Families were consistently **wealthier and whiter** than families who did not game the system, exacerbating economic and racial divides in BPS.

11.2.3 Fair equal opportunity

Requires that **everyone has a fair chance to obtain** public and social offices and positions. It posits that supposing some distribution of talents, those who have the **same talent and ability and willingness to use talents, everyone should have the same prospects of success**.

12 October 18th, 2021

12.1 Internet advertising

High level view. There is search advertising (Google, Bing) and contextual advertising.

In contextual advertising, we can be selling our own inventory (Facebook news feed), or third party inventory. Third party inventory includes standing bids (AdSense) or real time bidding/programmatic (DoubleClick, AdX).

12.1.1 Early designs

The earliest design was by a company called GoTo. The search engine did not have any organic search results. It only had advertised search results.

GoTo used an auction with three features:

1. Bid per click
2. Rank by bid
3. First price

12.2 Model

We will consider advertisers with value v_i per click. We will talk about position j. Many models that we use to design auctions have the *separable assumption*.

Definition 12.1 (Separable assumption)**.** The *separable assumption* states that the click-through rate for bidder i in position j is given by

$$\text{CTR}_{ij} = Q_i \times \text{pos}_j \tag{58}$$

where $Q_i \in [0,1]$ the quality of your ad and $\text{pos}_j \in [0,1]$ the position effect. The quality is a normalized quality score for the add as it relates to the user's search. A higher quality score corresponds to a greater probability for a click. The position effect is the probability of a click in a particular position for an advertiser with $Q_i = 1$.

12.3 Generalized second price auction

In a generalized second price auction, we have bids $b_1, \cdots, b_n$. The allocation rule ranks $Q_1 b_1 > Q_2 b_2 > \cdots > Q_n b_n$. The payment rule is

$$P_{\text{GSP},i}(b) = \frac{Q_{i+1} b_{i+1}}{Q_i} \tag{59}$$

where $P_{\text{GSP},i}$ is the price per click.

Example 12.2 (GSP with quality)**.** Consider a GSP with two ads on the page

position	bids	Q	t_i
0.2	2	0.2	1.50
0.1	3	0.1	1
	1	0.1	

Table 2: GSP with quality

Notice that we can compete on the amount you bid *and* quality. Moreover, note that even if you hold the same position, your price will fall if Q_i increases. This incentivizes companies to develop high quality, well-targeted ads.

In the future, we will assume that we are talking about $Q = 1$.

Claim. GSP is **not** strategy-proof.

Example 12.3 (GSP)**.** Consider a GSP with

position	bids	values	GSP
0.2	10	10	3
0.18	3	3	2
0.1	2	1	1
	1	1	

Table 3: GSP without quality

Note that bidder 1 has a useful deviation. Consider $b_1' = 2.5$. We have expected utilities $\langle u_1 \rangle, \langle u_1' \rangle$ given by

$$\langle u_1 \rangle = 0.2(10 - 3) = 1.4 < 0.18(10 - 2) = 1.44 = \langle u_1' \rangle. \tag{60}$$

12.4 VCG position auction

We have bids $b_1, \cdots, b_n$. The allocation rule is the same as GSP, ranking $Q_1 b_1 > \cdots > Q_n b_n$. The payment rule is charging the difference between the value of others without i and the value of others with i. More formally,

$$t_i(b) = \sum_{k=i+1}^{m} \left(\text{pos}_{k-1} - \text{pos}_k\right) Q_k b_k \tag{61}$$

where m is the number of positions. To convert to a price per click, we have

$$P_{\text{VCG},i}(b) = \frac{t_i(b)}{Q_i \text{pos}_i}. \tag{62}$$

Example 12.4 (GSP)**.** Consider our example with additional payments We note that the VCG prices are

position	bids	values	GSP	VCG	per click
0.2	10	10	3	(0.2-0.18)3+(0.18-0.1)2+(0.1)1 = 0.32	0.32/0.2 = 1.6
0.18	3	3	2	(0.18-0.1)2+(0.1)1 = 0.26	0.26/0.1 = 1.4
0.1	2	1	1	(0.1)1 = 0.1	0.1/0.1 = 1
	1	1			

Table 4: GSP, VCG, and price per click for same example

less than the GSP prices (except for the final bidder). The intuition for what goes wrong with GSP is that we assume the first bidder completely blocks the second bidder. This is not the case because the second bidder still gets click-through rate. GSP does not look at the effect on other bidders. VCG does! Finally, VCG is strategy-proof, no incentive to deviate.

Question. How does VCG compare to GSP? What do Nash equilibria for GSP look like?

Let there be three bids $b_1 > b_2 > b_3$ and consider useful upwards and downwards deviations for b_2. To check no useful upwards and downwards deviation, we require

$$\text{pos}_2(v_2 - b_3) \geq \text{pos}_1(v_2 - b_1), \qquad \text{pos}_2(v_2 - b_3) \geq \text{pos}_3(v_2 - b_4) \tag{63}$$

respectively. These are the conditions to check for Nash equilibrium analysis.

Example 12.5 (Nash equilibrium for GSP)**.** Consider

position	bids	values
0.2	3	3
0.18	2.05	10
0.1	2	2
	1	1

Table 5: Nash equilbrium for GSP

We need to check the Nash inequalities for each bidder.

12.5 Envy-free outcomes

We are comparing our utility with the utility we could get if we paid the price the bidder above you is paying. To require **no** up or down envy for bidder 2, we have

$$\text{pos}_2(v_2 - b_3) \geq \text{pos}_1(v_2 - b_2), \qquad \text{pos}_2(v_2 - b_3), \geq \text{pos}_3(v_2 - b_4) \tag{64}$$

respectively. Note the first constraint is **different from** the Nash constraints and the second is the same.

If we consider Nash equilibria that are envy-free, we get GSP1 and GSP2.

Proposition 12.6 (GSP1)**.** *Envy-free* $\implies$ *value ordered.*

Proposition 12.7 (GSP2)**.** *Envy-free and Nash* $\implies$ $r_{GSP}(b^*) \geq r_{VCG}(v)$ *where* r *is revenue,* b^* *is the equilibrium, and* v *true values.*

Example 12.8 (EF Nash equilbrium)**.** Consider

position	bids	values
0.2	10	10
0.18	2.05	3
0.1	1.5	2
	1	1

Table 6: Envy-free Nash equilibrium

Checking this is left as an exercise for the reader.

12.6 Spiteful and balanced bidding

Some motivation: the thing that you are worried about when you're bidding is that the bidder above you can retaliate and bid just below the second bidder.

The answer is for bids to be *indifferent* to retaliation. Intuitively, you should bid *just high enough* so that if retaliation, the bidder will be indifferent to the different outcomes. More formally we want to satisfy

$$\text{pos}_i(v_i - b_{i+1}) \geq \text{pos}_{i-1}(v_i - b_i). \tag{65}$$

Definition 12.9 (Balanced bidding (BB))**.** For all bidders i, we have

$$\text{pos}_i(v_i - b_{i+1}) = \text{pos}_{i-1}(v_i - b_i). \tag{66}$$

Example 12.10 (Balanced bidding)**.** Consider

position	bids	values
0.2	10	10
0.18	b_2	3
0.1	b_3	2
	1	1

Table 7: Balanced bidding

We can solve

$$0.1(2-1) = 0.18(2-b_3) \iff b_3 = 1.44, \qquad 0.18(3-1.44) = 0.2(3-b_2) \iff b_2 = 1.6. \tag{67}$$

and note that we have **exactly recovered** the outcome for VCG.

Theorem 12.11. *Balanced bidding construction* $\implies$ *envy-free and Nash equilibrium play*

Balanced bidding and envy-free Nash equilibrium play $\equiv$ *VCG outcome.*

Note (VCG vs. GSP)**.** Some closing remarks about VCG and GSP auctions

1. **Pros**: VCG is flexible and strategy-proof. Strategy-proofness lets us run experiments (e.g. reserve price) and gives us useful data about values (e.g. econometrics).
2. **Cons**: VCG is susceptible to reengineering and transitioning from GSP to VCG loses revenue in the short-run.

13 October 21st, 2021

Midterm today! 9-10:20 AM EST in Jefferson 250.

14 October 25th, 2021

14.1 Combinatorial auctions

We have a set of items $G = \{A, B, C, \cdots\}$ where capital letters denote items where $|G| = m$. We also have valuations for each bidder $v_i(S) \in \mathbb{R}_{\geq 0}$ where $S \subseteq G$. v_i is a function that takes bundles S to values. We also have bids $b_i(S)$.

Example 14.1. Consider a table

	A	B	AB
1	0	0	10
2	2	5	12
3	10	8	12

Table 8: Combinatorial auction

where elements a_{ij} in the table correspond to bidder i's value for item or bundle j.

Definition 14.2 (Super-additive valuations)**.** We say a valuation is *super-additive* if

$$v_i(S \cup S') \geq v_i(S) + v_i(S') \tag{68}$$

Definition 14.3 (Sub-additive valuations)**.** We say a valuation is *sub-additive* if

$$v_i(S \cup S') \leq v_i(S) + v_i(S') \tag{69}$$

for S, S' disjoint.

Note. In the above example, bidders 1 and 2 have super-additive valuations and bidder 3 has a sub-additive valuation.

The *efficient* allocation is $a_1 \to AB, a_2 \to B, a_3 \to A$. In general, one bidder's valuation can have **both** super and sub-additive components.

14.1.1 Challenges

There are three challenges when designing combinatorial auctions

Bidding language: We don't want people to write down value explicitly for all subset of items. There are $\mathcal{P}(G) = 2^{|G|}$ subsets.

Winner determination: Determine the winner of an auction. We want an optimal algorithm to solve this NP-hard problem.

Pricing: Need a way to price!

14.2 Bidding languages

There are two components: the syntax and the semantics. The syntax is how we will write this down in a language.

Let $\mathcal{L}$ be a language, denoting the set of bids we can write down syntactically and $B_i \in \mathcal{L}$ be the string we write down.

We will use b_i be the semantics, or the bidding functions. B_i is the representation of b_i.

We will focus on languages that are *expressive* and *succinct*.

Example 14.4 (XOR language)**.** The semantics of this language is "at most one". Consider

$$B_i = (A, 10) \oplus (B, 8) \oplus (AB, 12) \oplus (EF, 20). \tag{70}$$

The semantics are that

$$b_i(S) \equiv \frac{\max \text{atom price}}{\text{atoms that are subset of } S}. \tag{71}$$

We note that A is an atom and 10 is the atom price for $(A, 10)$.

The XOR language is expressive because we can represent all bids.

Example 14.5 (OR language)**.** We can take "any number" of atoms. Consider

$$B_i = (A, 2) \vee (B, 5) \vee (AB, 12) \vee (EF, 20) \tag{72}$$

with semantics

$$b_i(S) \equiv \max\{\text{total atom price on set of atoms that } \textit{fit} \text{ into } S\}. \tag{73}$$

Theorem 14.6. *OR is expressive for super-additive valuations.*

Example 14.7 (XOR-of-OR)**.** We can consider languages with syntax

$$B_i = \bigoplus x_1 \vee x_2 \vee \cdots \vee x_n \tag{74}$$

Example 14.8 (OR-of-XOR)**.** We can consider languages with syntax

$$B_i = \bigvee x_1 \oplus x_2 \oplus \cdots \oplus x_n \tag{75}$$

Note. Both of the above are expressive because we have XORs!

Definition 14.9 (Small bid)**.** We say B_i is *small* if the number of atoms is polynomial in the number of items.

Definition 14.10 (Language succinctness)**.** We say a language $\mathcal{L}$ is *succinct* on the valuation domain V if every $v_i \in V$ has a small $B_i \in \mathcal{L}$.

Moreover, we say $\mathcal{L}_1$ is *as succinct* as $\mathcal{L}_2$ if for every $v_i \in V$, the size of $B_i^{(1)} \in \mathcal{L}_1$ is **at most** polynomially larger than the size of $B_i^{(2)} \in \mathcal{L}_2$.

We say $\mathcal{L}_1$ is *more succinct* than $\mathcal{L}_2$ if $\mathcal{L}_1$ is *as succinct* as $\mathcal{L}_2$ and $\exists v_i \in V : B_i^{(1)}$ is small and $B_{2i}^{(2)}$ is **not**.

Example 14.11 (Succinctness)**.** Consider the additive valuation given by

$$V_{\text{add}} : v_i(S) = |S|. \tag{76}$$

Theorem 14.12. *OR is more succinct than XOR on V_{add}.*

Proof. To express the additive function in the OR and XOR languages, we write

$$B_1^{(OR)} = (A, 1) \vee (B, 1) \vee (C, 1), \qquad B_1^{(XOR)} = (A, 1) \oplus (B, 1) \oplus (C, 1) \oplus (AB, 2) \oplus \cdots \tag{77}$$

More formally, suppose we skip some $S \subseteq G$. If the bid is correct for all $S' \subsetneq S$, then $b_1(S) < |S|$. □

Example 14.13 (OR*)**.** This is OR with dummy items, for example $(9d_1, 100) \vee (10d_1, 100) \vee (18d_2, 150) \vee (19d_2, 150)$. Note that $G = (9, 10, 18, 19)$ where d_1, d_2 are dummy items. In this language, we are allowed to bring in any amount of dummy items that give us the valuation function we want for the auction.

Note that in effect, the first two and last two clauses behave like $\oplus$ because we are only allowed to take one of d_1, d_2 for allocation. This is like an OR-of-XOR language.

Theorem 14.14. *OR* is as succinct as OR-of-XOR.*

Proof. Idea: add a dummy to each atom in each XOR clause. This is a polynomial difference. □

Theorem 14.15. *OR* is as succinct as XOR-of-OR.*

Proof. Let A_j atoms and suppose our XOR-of-OR looks like

$$(A_1 \vee A_2) \oplus A_3 \tag{78}$$

we can write an equivalent OR* statement as

$$A_1 d_1 \vee A_2 d_2 \vee A_3 d_1 d_2. \tag{79}$$

Suppose that our XOR-of-OR looks like

$$(A_1 \vee A_2) \oplus (A_3 \vee A_4) \iff A_1 d_1 d_3 \vee A_2 d_2 d_4 \vee A_3 d_1 d_2 \vee A_4 d_3 d_4 \tag{80}$$

where the second statement is our OR* statement. □

Claim. We can do this in ℓ^2 dummy items where ℓ is the number of items in the XOR-of-OR.

14.2.1 Other valuations

Definition 14.16 (Monochromatic valuations)**.** In monochromatic valuations, items are one of two types (let's say blue or red). Then we have

$$v_i(S) = \max\{\#\text{red}(S), \#\text{blue}(S)\} \tag{81}$$

Definition 14.17 (k-Budget valuation)**.** The value for a set is constrained by k

$$v_i(S) = \min\{k, |S|\} \tag{82}$$

Then consider the table

	XOR-of-OR	OR-of-XOR
Monochromatic	$(R_1 \vee R_2 \vee R_3) \oplus (B_1 \vee B_2 \vee B_3)$	Not succinct
k-Budget	Not succinct	$(A \oplus B \oplus C \oplus D) \vee (A \oplus B \oplus C \oplus D)$ for $k = 2$

Table 9:

Theorem 14.18. *OR* is expressive and more succinct than each of XOR, XOR-of-OR, and OR-of-XOR.*

Proof. XOR is additive. XOR-of-OR is **not** succinct for k-Budget. OR* is succinct because OR-of-XOR is succinct for k-Budget and we can get OR-of-XOR from OR* because OR* is as succinct as Or-of-XOR. We argue the last one similarly using the monochromatic valuation. □

Note. This means that we can have a bidder go to a user interface and have the user interface generate an OR* representation for the bid, compile it into OR* and use this for the winner determination problem and payments.

15 October 27th, 2021

15.1 Review

From last time items are denoted $G = \{A, B, C, \cdots\}$ and valuations on bundles $v_i(S), S \subseteq G$.

We also recall the OR* language and concluded (by observation) that the OR* language is a *universal back-end representation* of the bids that are placed by bidders in a combinatorial auction. We can then use the results from OR* to determine the winner and payments of the auction.

For winner determination, we can think about the entire bid as an input to the problem. We seek to find the allocation that maximizes total value.

15.2 Winner determination problem

The winner determination problem (WDP) takes an OR* bid as an input and outputs a feasible set of atoms to maximize total value.

Theorem 15.1. *The winner determination problem is NP-hard.*

Definition 15.2 (Independent set)**.** An *independent set* is a set of vertices with no shared edges.

Proof. We will reduce MAX-WEIGHTED-IND-SET to WDP. In MAX-WEIGHT-IND-SET, the input is a weighted undirected graph $G = (V, E)$ and the output is an independent set that *maximizes total weight.*

For the reduction, we turn edges into items and vertices as values for a bundle created by connected edges. Then feasible set of atoms $\iff$ independent set and WDP is NP-hard. □

Example 15.3 (Reduction)**.** In the example on the board, we have

$$(A, 2)^* \vee (ABD, 3) \vee (BC, 2)^* \vee (CD, 1) \tag{83}$$

where * indicates efficient allocation.

Because WDP is NP-hard, we need some approximation algorithms to solve this problem.

15.2.1 Integer programming

Same idea as a linear program, but our decision variables are either a 0 or 1.

Consider the example above, we associate variables $z_i \in \{0, 1\}$ with each of the atoms. We associate $z_1 = A, z_2 = ABD, z_3 = BC, z_4 = CD$ in our example. We can write

$$\max_z 2z_1 + 3z_2 + 2z_3 + 1z_4 \tag{84}$$

with constraints

$$z_1 + z_2 \leq 1, \qquad z_3 + z_4 \leq 1, \qquad z_2 + z_3 \leq 1, \qquad z_2 + z_4 \leq 1. \tag{85}$$

Note that we have *integer* decision variables and linear optimization problems and constraints.

We can solve integer programs using the Branch and Bound algorithm.

Note (Branch and Bound algorithm)**.** At the root node, we solve a linear program relaxation of the problem, where we do **not** impose $z_i \in \{0, 1\}$. We allow for fractional solutions.

If the solution is not integer, we branch on one of the variables in the solution of the root. Take for example if $z_1 = 0.5$. We then consider two subproblems where $z_1 \leq 1$ and $z_1 \leq 0$. We again solve a linear program

relaxation. We might obtain the optimal or feasible (optimal) solution from one of the subproblems. For the problems that do not give a feasible solution (or gives a greater value), we branch off the second variable and continue.

We stop if all feasible solutions are less than the optimal.

15.2.2 Tractable special cases (OR)

In general, we can solve these tractable special cases using linear programs.

All pairs: If atoms are of size 1 or 2, we can determine the winner quickly.

Desert island: If all the bundles respect a cyclic order, we can solve this problem quickly.

Tree structure: We can associate all items in a tree and define valid bundles $S : \exists$ a seed vertex and integer $r \geq 0 : S$ includes all items. This problem can be solved.

Hierarchy: We can create bundles that are hierchies where the different sub-bundles are disjoint. We can solve this problem in polynomial time using dynamic programming from the bottom up.

15.2.3 Approximation algorithms

For example, greedy bang-per-buck.

Claim (VCG for combinatorial auctions)**.** VCG does **not** do well for combinatorial auctions. We can consider

A	B	AB
0	0	10*
10	0	10

A	B	AB
0	0	10*
2	0	2
0	2	2

A	B	AB
0	0	10*
2	0	2

A	B	AB
0	0	10
10*	0	0
0	10*	10

The VCG revenue for each case, the revenue is $\boldsymbol{r} = (10, 4, 12, 0)$. For cases (a) and (d), we see that this fails revenue monotonicity. If we look at (b) and (d), we see that losers can collude and win and pay 0. If we look at (c) and (d), we see false name bidding. Moreover, (d) is low revenue.

Claim. VCG avoids these pathologies if the language is OR-of-XOR-of-singletons.

Example 15.4 (OR-of-XOR-of-singletons)**.** Consider the bid

$$[(A, 1) \oplus (B, 2)] \vee [(B, 4) \oplus (D, 10)] \cdots \tag{86}$$

Note. This is one useful way to think about what is going wrong with VCG. If we look at example (d), the VCG outcome is outside the *core*.

Definition 15.5 (Core)**.** Let $X = (X_1, \cdots, X_n)$ be allocations and $p = (p_1, \cdots, p_n)$ be payments. Let

$$V_L = \max_X \sum_{i \in L} v_i(x_i) \tag{87}$$

be the value of the max allocation considering the bidders in L. We can write profit π and revenue R functions

$$\pi_i = v_i(x_1) - p_i, \qquad R = \sum_i p_i \tag{88}$$

Then we say (X, p) is *core* $\iff$

1. $\pi_i \geq 0 \forall i$

2. $\forall$ coalitions $L \subseteq N$, we have

$$R + \sum_{i \in L} \Pi_i \geq V_L \tag{89}$$

Example 15.6 (CA example d)**.** We consider If we think about this for $L = \{1\}$, we require

A	B	AB
0	0	10
10*	0	0
0	10*	10

Table 10: Example (d)

$$R + \Pi_1 \geq V_1 \implies 0 + 0 \geq 10, \tag{90}$$

which is **not** true.

Example 15.7 (SPSB)**.** Consider values $10*, 8, 4$. If we think about $L = \{2\}$, we need

$$R + \Pi_2 \geq V_2 \implies 8 + 0 \geq 8 \ \checkmark \tag{91}$$

The intuition from these examples is that the seller is not taking enough revenue. Equivalently, the winners are not paying enough. In practice, this means that when combinatorial auctions are used at scale, we require core constraints (or we don't use VCG). Or we look at payments that are *as close to VCG as possible* and satisfy core constraints.

16 November 1st, 2021

16.1 Matching markets

The theorem in the background is the Gibbard-Satterthwaite impossitility theorem.

Definition 16.1 (Onto)**.** A mechanism is *onto* if each outcome can possibly be selected. There exists an input where each outcome will be selected.

Theorem 16.2 (Gibbard-Satterthwaite impossibility)**.** *If there are three or more outcomes and the mechanism is "onto," and all strict preference orders are possible, then strategy proof* $\implies$ *dictatorial.*

Note (VCG)**.** There are many strategy proof non-dictatorial mechanisms. For example, VCG can operate on any number of outcomes. VCG is strategy proof and non-dictatorial because *not all strict preference orders* are possible. This is because money is part of the outcome and we assume agents want to spend less money.

Moreover, note that VCG and other previous mechanisms relied on *payments* or monetary value/utility. Because of this, we were able to put them on the same scale and give strict preference orderings. Here, we have abstracted away money and utility and instead will focus on general agent-specific preference orderings.

To avoid a dictatorial mechanism, we need to consider restricting our preference domain or dropping complete strategy-proofness. We can also restrict to two outcomes, but we will more commonly consider the first two options.

Example 16.3 (Facility location)**.** Recall our example of placing a building on campus where we consider the *median* mechanism where we take the median of all student preferences. We can restrict students' preferences to 'single-peaked' preferences note that this is *strategy-proof.* This is also agent-optimizing.

16.2 Two-sided matching

We can consider a generalized matching problem that includes public school matching, medical interns to residencies, and teaching assistants to professors.

This is a problem where we have two sets of agents, where each member in one set having *strict preferences* over each member of the other. A *matching* is an assignment of each agent to *at most one* agent on the other side.

The outcome is the *entire matching.*

Claim. Not all strict preference orders are possible. We can consider this from one agent's perspective. If this agent's match is unchanged, then the agent is *indifferent* about other agents' matches. Thus we **do not** have strict preference orderings, we have many ties, i.e. any permutation of the other agents' matches.

Example 16.4 (Blues Brothers)**.** We consider a heteronormative example from a movie where boys and girls have strict preference orders over the other set.

We consider an *unstable* matching over the two sets.

Definition 16.5 (Blocking pair)**.** A matching is *unstable* if there are two agents, one on each side, that mutually prefer each other to their match. This is called a *blocking pair*

Definition 16.6 (Stable matching)**.** A matching is *stable* if there are no blocking pairs.

We can study existence, uniqueness, and algorithms to find stable matchings through the Gale-Shapley deferred acceptance algorithm.

16.2.1 Gale-Shapley deferred acceptance, 1962

Let A, B be two sets of agents on different sides. In the first round, each agent in A will be matched with their highest preference in B. Agents in B will then hold onto the proposal they receive from the agents who are in their strongest preference and reject other proposals. The agents in A who are rejected will then propose to their second-highest preference and this process continues.

Example 16.7 (Blues Brothers)**.** We consider the algorithm on the previous example, from the boy-proposing and girl-proposing DA.

Claim (Terminates)**.** In each step, someone removes a name from their list. There are only a finite number of names, so the algorithm cannot run forever.

Theorem 16.8 (Stability)**.** *Deferred acceptance terminates with a stable matching.*

Proof. AFTSOC a matching $\mu_{\text{DA}} = \{(b, g), (b', g')\}$ is blocked by (b, g'). So b prefers g' to g so by the algorithm, b proposed to g' before g. But now g' has b', so must prefer b' to $b \implies$ **not** a blocking pair. □

Note that the algorithm is intuitively in P. Moreover, stability matters because if this algorithm results in unstable matches, people may choose to leave the market over time and the market will become more and more constrained. Stable matchings are **not unique**.

Note. We can note that all stability arguments are under an assumption that *all reports are truthful.*

Definition 16.9 (Achievable outcomes)**.** g is *achievable* for b if b and g match in some stable matching. Then $\{g, g'\}$ are achievable for b.

Claim. In boy-proposing DA, each boy gets best *achievable* girl.

Proof. AFTSOC some boy rejected by his top achievable girl. Say round k is the *first* round where this happens and say boy b is rejected by top achievable girl g in favor of boy b'. g achievable for $b \implies \exists$ stable $\mu = \{(b, g), (b', g')\}$. Note that because this is the *first* round any boy is rejected by top achievable girl, then b' must prefer g to all (other) achievable girls. This is a contradiction because (b', g) blocks μ. □

Claim. In boy-proposing DA, every girl matches with her *least preferred* achievable boy.

Note (Lattic property)**.** By joining two stable matchings and asking each agent whom they prefer, we can create a new feasible stable matching that is **both** improving for girls and worse for boys and vice versa.

We note that the lattice/linearity property means that there exists a boy-optimal and a girl-optimal stable matching by this lattice structure property. Moreover, if these are the same, there is **only one** stable matching.

We also note that boys prefer any stable matching to girl-optimal and vice versa. Girls prefer any stable matching to boy-optimal.

16.3 One-sided matching

In one-sided matching, agents have a strict preference on items and items don't have preference on agents.

16.3.1 House allocation/markets

Consider a set of houses and each agent has a strict preference order over houses. We want to assign houses to agents, where each agent will get *at most* one object.

The outcome is the *complete allocation.* Note that we do **not** have all possible strict preference orders similarly to the above.

The strategy-proof Pareto optimal mechanism is *serial dictatorship*. This is a bit unfair, but to make it more fair, we can consider *randomized serial dictatorship*.

It is conjectured that random serial dictatorship is the **only** anonymous Pareto-optimal strategy-proof mechanism.

Definition 16.10 (Anonymity)**.** A mechanism is *anonymous* if the distribution over allocations is invariant over permutations of the identities of agents.

16.4 Incentives

We note that truthful reporting is the **dominant strategy** for boys in boy-proposing DA. If boys are truthful, each boy is matched to most-preferred achievable girl.

Girl proposing is truthful for girls analogously.

Claim. No matching mechanism is stable and (fully) strategy proof. We will show this on a problem set.

16.5 Applications

There are plenty of real-world matching markets.

Example 16.11 (National Residency Matching Program)**.** Adoption of student-proposing NMRP in 1998. This is easier for students. Concerns include couples with preferences on pairs of positions

Example 16.12 (Boston/New York City school choice)**.** The "Boston mechanism" for school choice was neither stable nor truthful. Boston soon adoped a student-proposing DA. This is easier, more fair, and allows for policy advice. Some concerns include priorities for schools (siblings or walk zones).

17 November 3rd, 2021

17.1 Review

Recall we are doing matching without money and in the background, we have the Gibbard-Satterthwaite impossibility theorem. We reviewed the Gale-Shapley deferred acceptance, which is stable, truthful for the proposing side, but not fully strategy-proof. We also recall the lattice structure on stable matchings and note that matchings are *not* Pareto optimal.

We also review the house allocation problem, where each agent has a strict preference on items. The strategy-proof Pareto optimal solution is serial dictatorship.

17.2 Housing market problem

Consider a set of agents, each agent with one house and a strict preference order on houses. We want to find a strategy-proof way to find a new *Pareto-dominating* allocation through trades.

17.2.1 Top-trading cycle mechanism

We consider the TTC mechanism where we follow the steps

1. Each agent points to most preferred house, allowing for self- edges.
2. Trade on cycles and agents and houses leave markets.
3. Remaining agents point to most preferred remaining house (can be their own)
4. Repeat (2), (3) until no agents left.

We can represent this using graphs.

Note. Each agent is a member of at most one cycle. This is because each agent has only one out-edge.

TTC has the following properties:

Claim (Pareto optimality)**.** TTC is pareto optimal.

Proof. Let N_k be the agents that *trade* in round k assuming truthful reports. Note that these agents are pointing at a house that is their top choice out of their remaining choices. Suppose that $x' \neq x_{\text{TTC}}$, that Pareto-dominates x_{TTC} where x is the allocation. We need

1. Any agent $i \in N_1$ to still get in x' their x_{TTC} assignment.
2. Any agent $i \in N_2$ must still get in x' their x_{TTC} allocation
3. We can continue this argument and show that $x' = x_{\text{TTC}}$, which is a contradiction.

□

Claim (Strategy-proofness)**.** TTC is strategy-proof.

Proof. Until the round that i trades, others will trade on the same cycles and update their edges in the same way. Suppose with truthful reports, we traded in round t to get item A.

If we consider a misreport where we trade in round $s \geq t$ and get some $B \neq A$. But A is best at t and so B is worse $\implies A \succ_i B$. Now consider a misreport where we trade in round $s < t$ and get some $B \neq A$. This means i is a member of some cycle where i is pointing at B. But the path $B \to \cdots \to i$ will still exist in round t. So $A \succ_i B$ because B is available at round t. □

We have some definitions to take us to our final claim about TTC.

Definition 17.1. We denote $X(L) \equiv$ the set of allocations that are feasible in an *economy* on $L \subseteq N$.

Definition 17.2 (Core)**.** An allocation x is in the core $\iff \nexists L$ with a blocking $x' \in X(L)$ that Pareto dominates x for L.

We can think about the core as a generalization of stability for two-sided matching. More precisely, stable matchings are in the core for two-sided matching because the only coalitions we are concerned about in two-sided matching problems are subsets size two.

Claim (Core)**.** TTC is a core outcome.

Proof. AFTSOC $\exists L, x; \in X(L)$ that blocks x_{TTC}. We observe that any $i \in L \cap N_1$ must get same item as $x_{\text{TTC}} \implies$ all cycles involving $L \cap N_1$ must be included in $L \implies$ any $i \in L \cap N_2$ must get same item in x' as x_{TTC} since there is nothing better in L because $L \cap N_1$ are "taken for." We can continue this and see that $x' = x_{\text{TTC}}$ for L, a contradiction. □

17.2.2 Applications of TTC

Trading books, artwork, furniture, etc. TTC can also be used for school choice in San Francisco and has been used in New Orleans. TTC is Pareto-optimal whereas Gale-Shapley is not.

17.3 Kidney-paired donation

Kidney failure is a serious medical problem, with a preferred treatment of kidney transplants. Transplants must be blood-type and tissue-type compatible.

In kidney-paired donation, incompatible **pairs** arrive into the market. The pair is a donor and patient looking to participate in swaps or cycles, trading like-for-like. This is legal.

Note (ABO compatibility)**.** We each carry a protein in our blood where we can only receive from certain blood types. O types are universal donors and AB types are universal receptors.

We can thus have pairs like

$$\boxed{AB - O} \to \boxed{O - A} \to \boxed{A - AB} \tag{92}$$

where the convention is patient $-$ donor pairs. This is a valid 3-cycle.

Note. We cannot use a housing market model where each pair points and swap. This is because in kidney-paired donations, we assume 0/1 preferences. We also limit the cycle size because for cycles larger than three, we run into logistical issues. There are also **ethical concerns** about patients/donors changing their minds.

Thus we consider finding vertex-disjoint cycles of length $\leq k$ that cover as many vertices as possible.

17.3.1 Special case $k = 2$

In this case, we can replace bidirectional edges with an undirected edge. We will drop other edges. We can then find the *maximum cardinality matching* using Edmond's algorithm (polynomial time).

Definition 17.3 (Max cardinality matching)**.** The *maximum cardinality matching* problem is finding the maximum number of edges such that every vertex is incident on at most one edge.

17.3.2 Complexity of max vertex-disjoint cycles

$k = 2$ is in P by Edmond's algorithm. $k = \infty$ is in P via a linear program given by

$$\max_y \sum_{(i,j)\in E} y_{ij} : \qquad \sum_j y_{ij} \leq 1, \qquad \sum_j y_{ij} = \sum_i y_{ji} \ \ \forall i, \qquad y_{ij} \geq 0 \tag{93}$$

which are flow constraints. The optimal solutions are non-fractional!

For $k - 3, 4, 5, \cdots$, the problem is NP-hard. We have an integer programming formulation

$$\max_{y_C} \sum_{c\in C} \text{size}_c y_c : \qquad \sum_{c\in C: v\in c} y_c \leq 1 \ \ \forall v \in V, \qquad y_c \in \{0, 1\} \ \ \forall c \in C \tag{94}$$

17.3.3 Longer cycles

We can generate more transplants and save more lives with longer cycles. The idea is *altruistic non-simultaneous donor chains.*

Pictorally, we have

$$D^* \to \boxed{P - D} \to \boxed{P - D} \to \boxed{P - D} \to \cdots \equiv \boxed{D^* \to \boxed{P - D} \to P} - \boxed{D \to \boxed{P - D} \to P} \cdots \tag{95}$$

where bringing in an altruistic donor D^*. In the second step, we can chunk up the operations into three transactions. The dangling donor in the broken up $P - D$ pair becomes a *new altruistic donor* and we can form very long chains of donations. This assumes that donors will continue to pay it forward and donate.

There is a computational problem of matching with altruistic donors. We can solve the "cycles and chains" problem by adding zero-weight back-edges to the altruistic donors.

18 November 8th, 2021

Today, we will discuss *information elicitation*. Where we can pay agents in a way where we convince agents to report their subjective belief truthfully.

18.1 Information elicitation

We will consider a set of outcomes $O = \{o_0, \ldots, o_{n-1}\}$ and a random variable X that takes on one of the outcomes. We will have agents with beliefs $p \in \Delta O$ and reports q. Where q is the the belief is in the probability simplex over the outcomes O, p, q are probability distributions.

We seek to elicit agents' values of p by using a *scoring rule* $s(q, o_k) \in \mathbb{R}$ that is dependent on the distribution of the report and the outcome We want to design good scoring rules!

Definition 18.1 (Expected payment)**.** The *expected payment* is

$$S(q, p) = E_{X \sim p}(s(q, X)). \tag{96}$$

Definition 18.2 (Strictly proper)**.** Scoring rule s is *strictly proper* $\iff$

$$S(p, p) > S(q, p) \ \ \forall p, q \neq p. \tag{97}$$

We note that this is a strict inequality.

Definition 18.3 (Linear scoring rule)**.** The *linear scoring rule* is given by

$$s_{\text{lin}}(q, o_k) = q_k. \tag{98}$$

We note that this is not strictly proper. Consider $O = \{\text{SUN}, \text{MOON}\}$ and $p = 0.6$.x We have the expected payment

$$S(q, p) = P(X = \text{SUN})q + [1 - P(X = \text{SUN})]\,(1 - q) = pq + (1 - p)(1 - q) \tag{99}$$

and our best report is $q = 1$ to maximize expected payment.

Definition 18.4 (Log scoring rule)**.** The *log scoring rule* is given by

$$s_{\log}(q, o_k) = \ln q_k. \tag{100}$$

Note that if an agent reports 0, the agent must pay $-\infty$. Thus, this rule is typically used to prevent reporting 0.

Definition 18.5 (Quadratic scoring rule)**.** The *quadratic scoring rule* is given by

$$s_{\text{quad}}(q, o_k) = 2q_k - \sum_{\ell=0}^{m-1} q_\ell^2. \tag{101}$$

Theorem 18.6. *Both log and quadratic scoring rules are strictly proper.*

Proof. Consider a binary log scoring rule. The expected payment is

$$S(q, p) = p \ln q + (1 - p) \ln 1 - q. \tag{102}$$

If we assume $p \in [0, 1]$, note that

$$\frac{\partial S}{\partial q} = \frac{p}{q} - \frac{1 - p}{1 - q} = 0 \iff p(1 - q) - q(1 - p) = 0 \iff p = q \implies \text{ strictly proper} \tag{103}$$

as desired. □

Definition 18.7 (Expected truthful payment)**.** The *expected truthful payment* is $G(p) = S(p, p)$.

Theorem 18.8 (Convexity of expected truthful payment)**.** *A scoring rule s is strictly proper $\iff$ G is convex.*

Proof. Consider fixing a report q. Then

$$S(q, p) = p \times s(q, o_0) + (1 - p) \times s(1 - q, o_1) \tag{104}$$

which is *linear* in the belief p.

Note (Graphical intuition)**.** We can also examine this graphically. We can consider the expected payments of two reports q', q''. We can then observe that $G(p)$ can be defined graphically as the maxima over a set of linear expected payment functions because

$$G(p) = S(p, p) = \max_q S(q, p) \tag{105}$$

and the point-wise maxima over a set of linear functions is a convex function.

To get from $G(p) \to S(q, p)$ for some report q, we simply take $\partial_p G\big|_{p=q}$.

□

Example 18.9. We review the 1950 Brier rule for weather forecasting.

Note (Simplicity)**.** We call the logarithmic scoring rule *local* and the quadratic scoring rule *non-local*. This means to evaluate payments in the logarithmic rule, we need only to look at an agent's report on the outcome. The payment in the quadratic rule depends on every possible outcome.

18.2 Peer prediction mechanisms

Peer prediction mechanisms are mechanisms where peers have different beliefs (signals) X_1, X_2 about subjective questions and give reports r_1, r_2. This report may depend on what one peer thinks other peers may report. We then give payments dependent on the two reports r_1, r_2.

18.2.1 Output agreement mechanism (OA)

Simplest type of mechanism, we pay everyone \$1 if they agree. The payment rule is given by

$r_1 \backslash r_2$	0	1
0	1	0
1	0	1

Table 11: Output agreement payment rule

We can determine if truthful reporting is an equilibrium by considering the expected payoff conditional on the other agent's actions.

A crucial property for output agreement to work is *strong diagonalization* where the largest entry in each row is on the main diagonal.

Definition 18.10 (Strictly proper mechanisms)**.** A peer prediction mechanism is *strictly proper* if truthful report is a strict correlated equilibrium. That is, it satisfies

$$E_{X_2 \sim P(X_2|X_1=j)}(t_1(j, X_2)) > E_{X_2 \sim P(X_2|X_1=j)}(t_1(j', X_2)) \tag{106}$$

for signals j of agent 1 and all misreports j'. We can define the expectation for agent 2 analogously.

18.2.2 1/Prior mechanism

This uses the knowledge of marginal probabilities and works for more distributions than output agreement. The payment rule is

$r_1 \backslash r_2$	0	1
0	$\frac{1}{P(S_i=0)}$	0
1	0	$\frac{1}{P(S_i=1)}$

Table 12: 1/Prior payment rule

To be strictly proper, we require 1/Prior to be *self-predicting*, that the other peer is most likely to have a signal when you have the signal. That is,

$$P(X_2 = j|X_1 = j) > P(X_2 = j|X_1 = j') \ \ \forall j, j' \neq j. \tag{107}$$

18.2.3 Scoring-rule based mechanisms

We can use scoring rules to build peer prediction mechanisms, given that we know the joint distribution of the signals. Given report r_1, the mechanism will compute

$$q_1 = P(X_2|X_1 = r_1) \rightarrow s(q_1, r_2) \tag{108}$$

where q_1 is the belief of agent 1 on the report of agent 2. We then feed the report into a *strictly proper* scoring rule *against* r_2.

Note. We note the mechanisms use different levels of knowledge. OA assumes no knowledge. 1/Prior requires designers to know the priors. Scoring-rule based requires the full joint distribution.

19 November 10th, 2021

Today, we will talk about prediction markets and go over ad-auction results.

19.1 Prediction markets

Prediction markets are markets designed for information aggregation. You can bet on beliefs and trade contracts, with payoffs associated with an observed outcome in the future.

The basic idea is to construct a *contract* on outcome $o \in \{0, 1\}$. If your belief $p = P(o = 1)$, the value of the contract is p. We buy a contract if price $< p$ and sell if price $> p$. The *market price* represents *aggregate belief* given dispersed information.

We can think of peer prediction relative to other mechanisms

	Elicit belief (verification)	Elicit signal (No verification)
$n = 1$	Scoring rule	
$n > 1$	Prediction market	Peer prediction

Table 13: Belief and signal elicitation

There are binary contracts that are winner-take all contracts; continuous contracts, where we pay \$1 per vote share; and multi-valued contracts where we pay based on the outcome.

19.1.1 Non-market alternatives

We can use opinion polls. Note that there is no incentive to be truthful, information is equally weighted, and it is hard to be real-time.

We can ask experts. Identifying can be hard and combining opinions can be difficult.

Prediction markets, however, have monetary incentives, information is weighted by money/value, and they are self-organizing and real-time!

Note (Properties of prediction markets)**.** We note that we can always trade at any quantity (liquidity). Information aggregation. No "round-trip arbitrage," i.e. you shouldn't be able to buy contracts and sell them and make any money without other agents also trading. Finally, prediction markets are real-time.

In the textbook, we also talk about combinatorial prediction markets, where we can bet on combinations of outcomes.

19.1.2 Call markets

We order the buy and sell offers by decreasing and increasing order, respectively. We clear when we can no longer make a trade. Consider

Buy offers	Sell offers
15	8
12	11
9	13
5	17
0	30

Table 14: Call market

Note that the first two transactions go through and we set the price $p \in [11, 12]$ to complete two trades. This is to ensure that all useful trades are made.

Note. This is not used for prediction markets because call markets are **not real time**.

We will often use continuous double auctions.

19.2 Continuous double auctions (CDA)

Double auctions have both buyers and sellers. The bids occur continuously.

In continuous double auctions, we have an order book on the buy side and the sell side. We consider the following order book

Buy offers	Sell offers
33	36
32	38
30	40

Table 15: Order books

We note that the best buy is below the best ask and no trades are possible. If a new buy offer shows up, let's say 38, they buy according to the sell book. In this case, they buy a contract for 36.

19.2.1 Short selling in a CDA

We can short sell contracts in CDA. Suppose a contract is worth \$1. If an agent believes this price is too high but does not own a contract, they can still borrow a contract to sell at sell at a lower price. They can then buy a contract later (hopefully for a lower price).

CDAs have low liquidity.

19.3 Automated market maker

A market maker quotes a price to buy or sell any quantity. The goal is to improve liquidity and thus information aggregation.

The AMM keeps track of how many total contracts are bought and sold.

Note (Desirable properties)**.** AMMs have no round-trip arbitrage. Prices are strictly positive and sum to one. Moreover it is *responsive* — if buy then price increases and if sell, then the price decreases. It is liquid (trades occur at any quantity). There are *myopic incentives*, i.e. it is optimal to buy until price is equal to belief. There is also a bounded loss of money to the market maker.

More formally, let $x \in \mathbb{R}^m_{\geq 0}$ be a market state. x_k units sold of contract k. Let $c(x) : x \to \mathbb{R}$ be the cost function. A cost function is concave and strictly increasing.

Each bidder pays their effect on the market state. Moreover, the AMM revenue is

$$c(x^T) - c(x^0) \tag{109}$$

where x^T is the final market state and we pay \$1 to the winners.

19.3.1 Logarithmic market scoring rule

The LMSR cost function is

$$c(x_0, x_1) = \beta \ln \left(e^{x_0/\beta} + e^{x_1/\beta} \right). \tag{110}$$

We know exactly how this AMM works because we know how the cost function works!

Note (Some math)**.** When we buy Q_k, we change the market state by

$$x \mapsto (x_k + Q_k, x_{-k}) \tag{111}$$

and the total payment by trader is

$$\pi_k^{\text{buy}}(x; Q_k) = c(x_k + Q_k, x_{-k}) - c(x). \tag{112}$$

Note that the price for an infinitesimal amount is

$$\pi_k(x) = \frac{\partial}{\partial x_k} c(x) \tag{113}$$

and an equivalent expression for total payment is

$$\pi_k^{\text{buy}}(x; Q_k) = \int_{z=x_k}^{x_k+Q_k} dz \ \pi_k(z, x_{-k}). \tag{114}$$

Then using the LMSR cost function, we have

$$\pi_0(x) = \frac{e^{x_0/\beta}}{e^{x_0/\beta} + e^{x_1/\beta}}, \qquad \pi_1(x) = \frac{e^{x_0/\beta}}{e^{x_1/\beta} + e^{x_1/\beta}} \tag{115}$$

and we immediately note that this is strictly positive! Sum to one! Responsive! Liquid! We need to show myopic incentives.

Recall that we want to show that it is optimal for a trader to trade until the instantaneous price on a contract equals belief p.

Definition 19.1 (Market scoring rule (MSR))**.** Consider a sequence of reports $q^{(0)}, q^{(1)}, \cdots, q^{(n)}$. We consider $q^{(0)}$ uniform. Upon realization of o_k, we pay agent i

$$t(q^{(i)}, o_k) - t(q^{(i-1)}, o_k). \tag{116}$$

We note that if t is the log scoring rule, it is rational to report truthfully in position i. If we scale the log scoring rule by β, the total cost is

$$t(q^{(n)}, o_k) - t(q^{(0)}, o_k) \leq \beta \ln 1 - \beta \ln 1/m = \beta \ln m \tag{117}$$

We note that AMMs and MSRs are equivalent!

20 November 15th, 2021

Today, we will discuss cryptoeconomics.

20.1 Background

Money is useful because we avoid *double coincidence of wants*. We don't have to find someone who is particularly interested in my available item. A double coincidence of wants makes it hard to trade in the *absence* of money.

Important properties:

- Divisible
- Storable
- Exchangeable
- Hard to fake
- Sustains its value

The first gold coins were minted around 610 BC. Paper currency became widely used in the 1800s to build trust. The *gold standard* is that the paper money is backed by gold. That by principle, demand the equivalent of an ounce of gold for the paper note.

The last currency to leave the gold standard is the Swiss franc.

Instead of a gold standard, we have *Fiat currencies*. These are currencies that are enabled by the government as the legal medium of exchange.

20.1.1 Digital currencies

We now arrive at *digital currencies*. Digital currency cannot be stopped (we can send digital currency across international boundaries), we can control the money supply, the transactions are done via code *without* intermediaries (there are no centralized banks). And with good design, we hope that there is lower cost than traditional currency and better privacy for transactions.

In a digital currency, money is equivalent to bits. The challenge is preventing double spending and money printing.

20.1.2 Cryptography

We will use the abstractions in this discussion. We leave the mathematical detail of these abstractions as an exercise to the reader.

One abstraction in cryptography is public-key infrastructure. Each agent A has a public key PK_A and a secret key SK_A pair. If X is a bit string that represents a coin, the agent will sign the coin using the secret key. Let $\text{sign}(X)_{SK_A}$ denote this operation. We require that we *need* SK_A to generate the signature. Anyone can use PK_A to verify that A signed X.

User A and give X to user B by appending PK_B to the coin and signing it:

$$\text{sign}\left([X, PK_B]_{SK_A}\right). \tag{118}$$

We will consider one-way hash functions $h(x) = y$ where $h : X \to Y$ by we cannot go from $Y \to X$.

This allow for *proof-of-work*. This is the idea that we can prove to someone that we've done something computationally difficult. The idea is that for a block B, we want to find a *nonce*: $h([B, \text{nonce}])$ is small, e.g. has 19 leading zeros in hex representation.

This is enough because the best we can do is to keep guessing numbers one by one because h is a one-way hash.

20.2 Bitcoin

Developed by Satoshi Nakamoto on Friday October 31st, 2008. The first ever bitcoin transaction occured on May 22nd.

Properties:

- Fiat currency: we cannot convert bitcoin into other currencies
- Total number of coins is bounded at 21M and the rate of new coin printing is controlled

  Note. Bitcoin is a *deflationary currency*. The value of one bitcoin will tend *increase* over time because of scarcity. In an inflationary currency, the value of one unit of the currency will fall over time.

- Coins can be copied, but the ledger system on the blockchain prevents double spending

Professor Parkes then presented the price of Gouda tulip bulbs in the Netherlands.

In Bitcoin, transactions are stored in entries in a distributed ledger. These entries are costly to make and costly to change. Once an entry is made, it stays there. This makes a currency hard to attack and anyone can check the ownership of a coin. The workers (miners) are paid coins and do the work to maintain the ledger. The workers are solving proof-of-work to generate a new *block* that contains transactions and will be added to the *blockchain*.

20.2.1 The blockchain

We will define some key terms.

Definition 20.1 (Transactions)**.** A *transaction* is a transfer of (fractional) coins that is recorded as an entry in the ledger.

Example 20.2. Consider a transaction from user $0 \to 1$. User 0 will sign the hash of PK_1 and the number of coins, e.g. $\text{sign}(h[X, PK_1]_{SK_1})$. Then if user $1 \to 2$, user 1 will sign this over to user 2 where anyone can use PK_1 to verify this signature.

Definition 20.3 (Blocks)**.** A *block* is a page in the ledger that contains a record of transactions.

new blocks are created by *miners*, or workers that are solving proof-of-work problems.

Definition 20.4 (Blockchain)**.** The *blockchain* is the entire ledger which puts an ordering on transactions. This is the genesis block that the block that Nakamoto generated.

Definition 20.5. A transaction that is not in the chain is *unconfirmed*.

We can think about the ledger as a distributed network where nodes are computers that are doing the mining. Miners "gossip" about new transactions. They are trying to solve proof-of-work and create a new block that contains transactions. Transactions may offer a fee to miners, which acts like an (unruly) first-price auction for entry into the 1MB block capacity. If a new block is valid, miner "gossips" it onto the network and the **block forms the head** of the blockchain.

A transaction is only accepted if the transaction is *valid* according to the transaction history and proof-of-work has been solved.

Definition 20.6 (Proof-of-work)**.** The *proof-of-work problem* is to find a nonce such that $h(B)$ has at least 19 leading zeros. This is a slight approximation of the actual proof-of-work problem:

$$\text{sha256}(\text{sha256}(B)) < \frac{\text{genesis-hash-target}}{\text{difficulty}}, \tag{119}$$

that is, the hash value has to be small. Over time, the difficulty has increased. The genesis-hash-target is the original hash value of the original genesis block. This reduces the size of number of successful hashes. Based on the current value of difficulty, this is approximately $\sim$19 leading zeros. The difficulty adjusts to keep the rate of generating a new block to $\sim$ 10 mins/block. This causes the chain to grow at a fixed speed.

Every 210k blocks ($\sim$4 years), the payment is halved, limited to 21M coins.

20.2.2 Consensus protocol

There may be accidental branches with two blocks created and shared at the same time. We need a protocol to maintain a consistent record. The protocol is that miners extend the *heaviest chain*, the one with the most cumulative difficulty.

Note. It is unlikely that two identical blocks will be mined at the exact same time, and the miner who does so will extend one of the chains. Suppose this is the top branch. Then all other miners will recognize the top branch as the heaviest one.

20.2.3 Security vulnerabilities

There have been small attacks, including denial of service attacks, overflow problems with code that checks transactions, a DoS attack on signature verification, etc.

There is a concern about miner concentration. For lower variance, miners join *mining pools* that collectively work to find blocks. When one minder solves a problem, the pool shares the rewards.

There are sometimes disagreements that cause *hard forks*. For example, disagreements about the protocol etc.

There are regulatory concerns, illegal transaction and money laundering concerns, and energy expenditure concerns due to mining.

20.3 Ethereum

Ethereum is the second largest coin. Ethereum is a virtual machine. We can do computations on it! This has enable a lot of exciting things related to smart contracts and distributed finance.

We will talk more about Ethereum next time!

21 November 17th, 2021

Today, we will discuss the price of anarchy.

21.1 Anarchy

We will think about anarchy as equilibria.

We are interested in the cost of equilibrium. Or rather, how bad things can be in equilibrium.

Example 21.1 (Routing game)**.** Consider a game Γ with two players who are trying to go from $s \to t$. The social objective is to minimize total cost. Consider a directed graph $s \to t$ with two edges with weights $2, x$.

The optimal a^{opt} is to split up with cost 3. This is a Nash equilibrium. The other Nash equilbrium a^* is for both players to use the path weight x and incur cost 4. Then the price of anarchy PoA is

$$\text{PoA}(\Gamma^*) = \max_{a^* \in \text{NE}(\Gamma)} \left[\frac{C(a^*)}{C(a^{\text{opt}})} \right] = \frac{4}{3} \tag{120}$$

21.1.1 Routing games

Routing games are games played on directed graphs. Each edge e has cost $c_e \in \mathbb{R}$. There are N players each moving from source $s_i \to t_i$. The action of a player a_i is the path of a player over the directed graph, and the cost is $c_i(a_1, \cdots, a_n)$ and define $C(a) = \sum_{i \in N} c_i(a)$ the total cost.

It is useful to consider the *flow on edge e given action profile a* denoted $f_e(a)$. Then we can redefine the cost function

$$c_i(a) = \sum_{e \in a_i} c_e(f_e(a)). \tag{121}$$

We define $a^{\text{opt}} \in \min_a C(a)$ and $a^* \in \text{NE}(\Gamma)$.

We are interested in understanding the price of anarchy on games like this. We are *particularly* interested in determining the lower or upper bound over a family of games (if these bounds exist). More formally, we are interested in the quantity

$$\max_{\Gamma} \text{PoA}(\Gamma^*) = \max_{\Gamma} \left\{ \max_{a^* \in \text{NE}(\Gamma)} \left[\frac{C(a^*)}{C(a^{\text{opt}})} \right] \right\} \tag{122}$$

where Γ is a game in some family of games. We can exhibit a lower bound by exhibiting a game with a price of anarchy of the lower bound. We seek to find the maximum.

Example 21.2. Consider vertices s, v, t with directed edges and weights

$$(s \to v, x), \qquad (v \to s, 0), \qquad (s \to t, x), \qquad (t \to s, 0), \qquad (v \to t, x), \qquad (t \to v, x). \tag{123}$$

We can consider four players with objectives

$$1 : s \to v, \qquad 2 : s \to t, \qquad 3 : t \to v, \qquad 4 : v \to t. \tag{124}$$

We note that $a^{\text{opt}} = $ 1-hop solution where each player takes their most direct path where $C(a) = 10$. We note that a^* is the inverse of the 1-hop. Players take the other route. This solution has cost 10 $\implies$ $\text{PoA}(\Gamma) = \frac{5}{2}$.

We will prove this remarkable theorem.

Theorem 21.3 (Price of anarchy for affine routing games)**.** *PoA $\leq \frac{5}{2}$ for all affine routing games. An affine routing game is one where the cost functions are linear on the edges.*

Definition 21.4 (Affine cost functions)**.** A cost function is *affine* if $C_e(a) = \alpha x + \beta$ for $\alpha, \beta \in \mathbb{Z}_{\geq 0} \forall e$.

Definition 21.5 ((λ, μ)-smoothness)**.** A cost minimization game Γ is (λ, μ)-*smooth* for $\lambda, \mu \in \mathbb{Z}_{\geq 0}$ if

$$\sum_{i \in N} c_i(a_i^{\text{opt}}, a_{-i}) \leq \lambda C(a^{\text{opt}}) + \mu C(a) \tag{125}$$

for all a.

The sum is an entanglement of the play of player i in the optimal profile and the play of others in another profile. We then take the sum of this over all players. We say that the cost is bounded by a linear combination of the optimal play and the actual play.

Theorem 21.6. *In a (λ, μ)-smooth game Γ with $0 \leq \mu < 1, PoA(\Gamma) \leq \frac{\lambda}{1-\mu}$.*

Proof. We have

$$C(a^*) = \sum_{i \in N} c_i(a^*) \leq \sum_{i \in N} c_i(a_i^{\text{opt}}, a_{-i}^*) \leq \lambda C(a^{\text{opt}}) + \mu C(a^*) \implies \frac{C(a^*)}{C(a^{\text{opt}})} \leq \frac{\lambda}{1-\mu}. \tag{126}$$

where we have the first inequality due to the Nash property and the second comes from the definition of (λ, μ)-smoothness. □

Thus if we can prove that our game Γ is smooth for some λ, μ, we can simply plug in our value for λ, μ into the above theorem and we obtain the price of anarchy. Then we simply need to show that affine routing games are $(\frac{5}{3}, \frac{1}{3})$-smooth.

Claim (A humorous fact)**.** David Parkes then gave us this useful humorous fact. For any $y, z \in \mathbb{Z}_{\geq 0}$, we have

$$(z+1)y \leq \frac{5}{3}y^2 + \frac{1}{3}z^2. \tag{127}$$

We remark that this is a bit of a magic fact that no one would think of to prove.

We recall that $f_e(a)$ is the number of agents on edge e and we can always write down the total cost on an action profile $c(a) = \sum_e c_e(f_e(a)) \times f_e(a)$.

Then we proceed to prove the price of anarchy for affine routing games.

Theorem 21.7 (PoA of affine routing games)**.** *Affine routing games are $(\frac{5}{3}, \frac{1}{3})$-smooth.*

Proof. We have

$$\sum_i c_i(a_i^{\text{opt}}, a_{-i}) \leq \sum_i \sum_{e \in a_i} c_e(f_e(a) + 1). \tag{128}$$

The left-hand side is the cost to player i when they play according to the optimal and others play according to another action profile. The right-hand side is the cost on every edge we are using when $f_e(a) + 1$ players use the edge. We are relying on $1 + f_e(a) \geq 1 + f_e(a_{-i})$. Then noting that

$$\begin{aligned}
\sum_i c_i(a_i^{\text{opt}}, a_{-i}) &\leq \sum_i \sum_{e \in a_i} c_e(f_e(a) + 1) = \sum_e c_e(f_e(a) + 1) f_e(a^{\text{opt}}) \\
&= \sum_e \alpha_e(f_e(a) + 1) f_e(a^{\text{opt}}) + \sum_e \beta_e f_e(a^{\text{opt}}) \leq \sum_e \alpha_e \left[\frac{5}{3} \left(f_e(a^{\text{opt}}) \right)^2 + \frac{1}{3} \left(f_e(a) \right)^2 \right] + \sum_e \beta_e f_e(a^{\text{opt}}) \\
&\leq \frac{5}{3} C(a^{\text{opt}}) + \frac{1}{3} C(a)
\end{aligned} \tag{129}$$

so we have what we want! □

There is more we can do with this! The punchline is that the game above is the worst-case possible game for any routing game with affine cost functions.

We recall that

$$\text{PNE} \subseteq \text{MNE} \subseteq \text{CE} \subseteq \text{CCE} \tag{130}$$

where CCE is the set of coarse-correlated equilibrium. This is a strict nexting. CCE is a relaxation of correlated equilibria. Algorithms will find play in CCE! One example of these is a *no regret algorithm*. Algorithms like these will converge to coarse equilibrium play.

We note here that PNP denotes pure-Nash, MNE denotes mixed-Nash, and CE denotes correlated Nash equilibria.

Definition 21.8 (Robust price of anarchy)**.** The *robust* price of anarchy on a game Γ is

$$\text{robustPoA} = \max_{p^* \in \text{CCE}}(\Gamma) \left[\frac{E_{a \sim p^*}(C(a))}{C(a^{\text{opt}}} \right]. \tag{131}$$

We are choosing to bound things in a bigger class of games. When we get to coarse-correlated equilibrium, we can learn things quickly.

Professor Tim Roughgarden at Columbia proved the following extension theorem. This theorem extends the proof for price of anarchy in Nash equilibria to a proof of price of anarchy in this robust sense.

Theorem 21.9 (Extension theorem)**.** *In (λ, μ)-smooth cost minimization games for $0 \leq \mu < 1$, we have*

$$\textit{robustPoA}(\Gamma) \leq \frac{\lambda}{1-\mu}, \tag{132}$$

which is exactly the same bound.

Following the extension theorem, we get a bound of $5/2$ with learning dynamics.

Definition 21.10 (CCE)**.** A *coarse-corrrelated equilibrium* says that

$$E_{a \sim p^*}(u_i(a)) \geq E_{a \sim p^*} u_i(a_i', a_{-i}) \tag{133}$$

for all a_i'. This is a weaker property than correlated equilibrium which is why this is a superset of CE.

Note (No-regret algorithms)**.** No-regret algorithms are a family of algorithms that have the property that if we play them for long enough, your regret for following the algorithm goes to zero relative to playing one fixed action all the time. You do not regret the choice of following an algorithm rather than playing the same action. We look at the history of play and calculate the opportunity cost.

One example of this is multiplicative weight algorithms. This converges to no-regret play.

There is a correspondence that can be shown between the convergence point of no-regret algorithms and the distribution p^* of a CCE. Then because we have the extension theorem, then we have the same claim for a set of no-regret learners.

22 November 22nd, 2021

Midterm today! 9-10:20 AM EST in Geological Labs 100.