CS 269I: Incentives in Computer Science
Winter 2023/Spring 2024

Eric Gao, Logan Mondal Bhamidipaty

Abstract

Lecture notes for CS 269I as taught by Professor Aviad Rubinstein in Winter 2023. Guest lectures by Okke Schrijver and Professor Balaji Prabhakar. Each section corresponds to a lecture. Computers and algorithms control economic processes:

- Online retail and advertising;
- Algorithmic trading of stocks and cryptocurrency

There is a need jointly understand economics and computation to analyze these applications. In some cases, introducing a new actor into a system will affect how others behave. While efficiency and complexity are issues, we also need to analyze how other agents will respond to new algorithms. This can't be solved solely with more powerful GPU's. All mistakes are my own.

Contents

1 One-Sided Matching

Model:

- there are n students, each with some preference over residences
- there are m dorms, each with some capacity

We want to assign students to dorms. One possible solution is to maximize the sum of utilities:

Utility Maximization

1. Create a bipartite graph with one side being students and the other side being residencies with with edges from students to dorm with edge weight equal to how much student i likes dorm j
2. Find the max-weight matching (i.e., Hungarian Algorithm)

Key question: how does the algorithm determine how much each student likes each residency? Only the students themselves knows how much they like each dorm, so we need to rely on students to tell the algorithm.

The issue: each student has an incentive to exaggerate how much they like their favorite dorm and undercut how much they like other dorms.

As such, finding the matching that maximizes happiness was the right solution concept, but failed to take incentives into account. Another possible algorithm is serial dictatorship:

Serial Dictatorship

1. Students are sorted by some fixed order (random, seniority, alphabetically, etc.)
2. Go through the list in order and each student selects their most preferred available dorm

Is this any better? There can still be students who are unhappy with their result under serial dictatorship.

Definition 1.1 (Mechanism)**.** A mechanism consists of three things: a method of collecting inputs from agents, an algorithm that acts on the inputs, and an action that is taken based on the output of the algorithm.

Remark. All three components are important: for instance, students' beliefs of how the algorithm works or how the action is taken would affect how they respond to the way inputs are collected.

1.1 Properties of Serial Dictatorship

Definition 1.2 (Strategyproof/truthful)**.** A mechanism is strategyproof if it is in every agent's best interest to act truthfully.

Definition 1.3 (Dominant Strategy Incentive Compatible)**.** A mechanism is dominant strategy incentive compatible if it is a dominant strategy for each participant to act truthfully. In particular, this means that being truthful is a best response to any possible strategy profile of other players.

Theorem 1.1 (Can't game Serial Dictatorship)**.** *The Serial Dictatorship mechanism is dominant strategy incentive compatible: it is in every student's best interest to choose their favorite available dorm in their term.*

Proof. Your room choice will not affect what rooms are available by the time your turn comes. When it gets to your turn, what you choose is what you get, so it is best to choose your favorite room among available options. □

Definition 1.4 (Pareto Improvement)**.** An outcome O' is a Pareto improvement over outcome O if all agents either strictly prefer O' to O or are indifferent between the two, with at least one strict preference.

Definition 1.5 (Pareto Optimal)**.** An outcome O is Pareto optimal if there are no Pareto improvements from O.[1]

[1] Pareto Optimality must be defined with respect to particular preferences. As such, it is difficult to conceptualize if an outcome is Pareto optimal if truthful preferences are unknown.

In other words, an outcome is Pareto optimal if you can't make anyone happier without making someone sadder.

Theorem 1.2 (Pareto Optimality of Serial Dictatorship)**.** *The outcome under serial dictatorship (with any ordering of students) is Pareto optimal.*[2]

Proof. Let the serial dictatorship outcome be O. Towards a contradiction, suppose O is not Pareto optimal, and there exists some outcome O' that is a Pareto improvement over O. Consider the first student that gets a different outcome under O and O'. Since all students before this student are assigned the same room, the set of available rooms for this student is the same under O and O'. However, the student chooses their favorite room under O, so the room they receive under O' must be worse. □

One issue with this: dictatorship can have a large affect on others. Suppose there are ten students and the first eight have chosen rooms 1 to 8. Also, suppose student nine has room 9 as their 9th favorite room and room 10 as their 10th favorite room while student ten has room 9 as their favorite room and room 10 as their least favorite room. If student nine chooses room 9 over room 10 they only gain a small improvement, but student ten is now forced to pick room 10 over room 9 which is a large jump in their preferences.

[2]This result only holds if every students' preferences over dorms is strict (no student can be indifferent between dorms).

2 Two-Sided Matching

After medical school, med students start their internship called a "residency". Each (prospective) doctor has preferences over hospitals and each hospital has preferences over doctors. How should doctors and hospitals be matched?

The biggest difference between residency matching and dorms is that we now have two sided preferences. Another difference: dorm assignments are all through Stanford's centralized system, but there is nothing stopping doctors and hospitals from matching on their own as an outside option.

Definition 2.1 (Match)**.** Consider a two-sided matching market with I being participants on one side and J being participants on the other side. Each $i \in I$ has preferences over J and each $j \in J$ has preferences over I. A match M is a function $I \cup J \to I \cup J$ where $M(i) \in J$ is i's match, $M(i) = i$ if i is unmatched; and $M(j) \in I$ is j's match, $M(j) = j$ if j is unmatched.

Definition 2.2 (Blocking Pair)**.** Given a match M, the pair (doctor i, hospital j) forms a blocking pair if they prefer each other to their assignment in M.

Definition 2.3 (Stable Matching)**.** A matching M is stable if there are no blocking pairs. Equivalently, for every unmatched pair (i, j) it must be true that either

- Doctor i prefers Hospital $M(i)$ over Hospital j or;
- Hospital j prefers Doctor $M(j)$ over Doctor i.

2.1 Deferred Acceptance

Main idea behind this algorithm: try to match each doctor to their favorite choice. If there ever is a blocking pair, switch the matching. In practice, each hospital accepts more than one doctor. For now, we assume that each hospital only has one spot (but the algorithm and results can be extended into the multiple spots case).

Deferred Acceptance

```
While there is an unmatched doctor $i$:
    1. Try to match doctor $i$ with the next-favorite hospital $j$ in their list;
    2. If hospital $j$ does not yet have a doctor then hospital $j$ and doctor $i$ (tentatively) match and
       both are happy with this match;
    3. Else-if hospital $j$ is already matched with doctor $i'$ and prefers their current match $i'$ to doctor
       $i$, then doctor $i$ remains unmatched;
    4. Else-if hospital $j$ is already matched with doctor $i'$ and prefers doctor $i$ to their current match
       $i'$, then doctor $i$ and hospital $j$ match making them both happier off while doctor $i'$ is now
       unmatched.
```

For an example, consider applying deferred acceptance to matching students to universities. Suppose there were three universities, Stanford (S), the University of Michigan (M), and Tsinghua (T). Also, suppose there are three students, Alex (A), Bill (B), and Carl (C). Suppose students have preferences

$$A : S \succ M \succ T$$
$$B : S \succ T \succ M$$
$$C : M \succ T \succ S$$

and universities have preferences

$$S : B \succ C \succ A$$
$$M : A \succ C \succ B$$
$$T : A \succ B \succ C$$

In the first round, Alex and Bill both apply to Stanford. Stanford prefers Bill (to Carl) to Alex so Bill's application is tentatively accepted. Alex is rejected, and removes Stanford from his preference list. Only Carl applies to Michigan, so Carl's application is also tentatively accepted. In the second round, only Alex needs to apply again, and Alex applies to Michigan. Now, Michigan compares Alex with Carl (from round one) and likes Alex more, so Alex is now tentatively accepted and Carl is rejected. As such, Carl strikes Michigan from his list of acceptable universities. In the third round, Carl applies to Tsinghua. This is the only application Tsinghua has received, so it is tentatively accepted. In the next round, there are no rejected students so no new applications are made. As such, the algorithm finishes. This process can be summarized in the following table, where green denotes tentative acceptance and red denotes rejection.

	1	2	3
A	S	M	M
B	S	S	S
C	M	M	T

Theorem 2.1 (Runtime of Deferred Acceptance)**.** *Deferred Acceptance runs in $O()$ time.*

Proof. Each iteration of the while loop takes $O(1)$ time. At every iteration, each doctor proposes to a new hospital. As no doctor and hospital ever try to match more than one, there are n^2 possible (doctor, hospital) pairs so there are at most n^2 iterations. Thus, overall runtime is $O(n^2)$. □

Theorem 2.2 (DA is stable)**.** *The deferred acceptance algorithm outputs a complete stable matching.*

Proof. The Theorem follows from the following three claims:

1. At every iteration, the current match is stable with respect to non-free doctors and hospitals.
2. Once a hospital is matched, it remains matched (but possibly to a different doctor) until the end of the algorithm
3. At the end of the algorithm, every doctor and hospital is matched.

Claim one. Towards a contradiction, suppose (d, h) is a blocking pair. Thus, d is currently matched to a hospital worse than h. This implies that d already tried to match with h. There are two cases: either h refused d, or h initially matched with d but left them later. In either case, h is currently matched with a doctor better than d so (d, h) cannot be a blocking pair.

Claim two. Clear from the algorithm: only way for a matched hospital to change is if there is a better doctor that proposes to them, in which case they will match with the better doctor.

Claim three. Towards a contradiction, suppose (d, h) is free. At the end of the algorithm d must have proposed to h already, so h must either be matched with d or a doctor better than d. In either case, (d, h) thus cannot be free. □

Theorem 2.3 (Efficiency of Stable Matchings)**.** *Every stable matching is Pareto-optimal.*

Proof. Suppose A is stable and let B be any other matching. Consider a doctor-hospital pair matched in B but not matched in A. By stability of A, either the doctor or hospital (or both) prefers their match under A to their match under B. □

There are many possible stable matchings that are possible, but which is best?

Theorem 2.4 (Proposing Optimality)**.** *The matching returned by doctor-proposing Deferred Acceptance is doctor-optimal.*

As a corollary of this result, we have:

Theorem 2.5 (Proposing Strategyproofness)**.** *Doctor-proposing Deferred Acceptance is strategyproof for doctors.*

On the other hand, hospitals are not quite as happy with doctor-proposing deferred acceptance.

Theorem 2.6 (Receiving Non-Optimality)**.** *The matching returned by doctor-proposing Deferred Acceptance is the worst stable matching for hospitals.*

Theorem 2.7 (Receiving Non-Strategyproofness)**.** *It is not always best for hospitals to truthfully reveal their preferences.*

Remark. At this point in the lecture, Aviad dramatically pulled out another bunch of bananas.

2.2 Deferred Acceptance in Practice

Historically, residency matches proceeded as follows:

- In the 1950's, National Resident Match Program used a deferred-acceptance like algorithm. At this time, there were very few female or openly gay doctors.
- In the 1960's, there were more couples that want to be near each other. As such, doctors no longer have ranked preferences over hospitals: individual preferences now also depends on what your partner's outcome is.
- In the 1980's, negative results were developed:
  1. A stable matching may not even exist;
  2. This matching problem is a NP-complete computational problem.
- In the 1990's, an extension of the Deferred Acceptance algorithm that resolved the couples issue was put into practice.

How do hospitals rank doctors?

- Interviews: but this process is costly, as doctors still need to strategize over which doctors to interview.
- Standardized tests: maybe not anymore, since the main test (USMLE) changed to pass-fail.
- Safety choices: hospitals might list some "safety" doctors, but using safety choices is never safer for hospitals (this is provably true).

Theorem 2.8 (No Improvements from Safety Choices)**.** *Hospitals misreporting preferences by moving their ith preferred doctor to a higher position cannot help with matching with doctor of rank i or better.*

Proof. Until doctor i tries to match with this hospital, the manipulation has no effect. After doctor i tries to match with this hospital, they can only be replaced by someone better. As such, either this hospital would have gotten doctor i anyways, or this hospital is stuck with doctor i opposed to someone better. $\square$

Why do hospitals still use safety picks?

- Hospitals don't know this theorem.
- Reputation/ego: hospitals publish a "number needed to fill" number, the lowest rank in a hospital's list of a doctor matched to the hospital.

Another situation where two-sided matching comes up is in college admissions: students apply and colleges decide who to accept. In the US, application fees and the time it takes to write applications restricts how many applications a student can submit. In other countries, there is a single standardized test that determines admissions.

Even when there is no outside option, deferred acceptance is fast to run and is used to match packets to servers.

3 Top Trading Cycles

From before, we had two mechanisms for two different models of matching:

- Random Serial Dictatorship for one sided matching;
- Deferred Acceptance for two sided matching.

Remark. Clarification: Strategyproofness is a guarantee against a single agent deviating from a truthful strategy. It is does not say anything about collusion. A mechanism can be strategyproof but not collusion-proof.

This lecture: what happens if people are already endowed with a good? For instance, Stanford PhD students are allowed to renew housing every year. However, they can re-enter the lottery at a low priority if they choose not to renew. The issue: students might get a room worse than their current room if they enter the lottery. This mechanism is not strategyproof and leads to sub-optimal assignments.

Top Trading Cycles

Input: n students and n rooms; each student has a current room and preferences over all n rooms.

```
While there are still unmatched students and rooms:
    1. Create a graph with each unmatched room being a node and each unmatched student being a node;
    2. Draw a directed edge from each room to the student that owns the room;
    3. Draw a directed edge from each student to their most preferred available room;
    4. Remove any cycles that form and trade along the directed edges.
```

We call this problem the *housing assignment problem with* ***incumbency***.

Definition 3.1 (Directed Bipartite Graph)**.** Informally, a directed bipartite graph is a graph with directed edges (arrows) and two disjoint groups (the two sides in a market, e.g., students and houses)

Theorem 3.1 (Runtime of TTC)**.** *Top trading cycles runs in $O(n^2)$ time, where n is the number of students/rooms.*

Proof. Constructing the graph takes $O(n)$ time. Drawing edges also takes $O(n)$ time. To find cycles, we can start at any node and follow the path traced by directed edges. Since every node has an outward-facing directed edge and there are a finite number of nodes, we will eventually visit some node twice; this gives a cycle.[3] We need to pass through at most $2n + 1$ nodes so finding a cycle will take $O(n)$ time. Finally, every run of the while-loop removes at least one student so there are $O(n)$ iterations. As such, total runtime is

$$(O(n) + O(n) + O(n)) \cdot O(n) = O(n^2).$$

□

Proof. To find all cycles, we run the strongly connected components (SCC) algorithm.[4] (Lemma: Every SCC with vertex > 1 is a cycle). So each iteration runs in $O(n)$ time. Each iteration removes ≥ 1 students, so at most $O(n)$ iterations total. Total running time: $O(n^2)$. □

Remark. Since each of the n students reports a preference list of length n, input size is n^2 so runtime is linear in input size.

Definition 3.2 (Individual Rationality (IR))**.** A mechanism is individually rational if no agent is worse off after participating in the mechanism.

Theorem 3.2 (IR of TTC)**.** *The top trading cycles mechanism is individually rational.*

Proof. A student trades their room if and only if they are getting a better room. □

[3]To formalize this, we can use the pigeonhole principle: take a path of length $2n + 1$; there are a total of $2n$ nodes so at least one node must have been visited twice or more.

[4]Tarjan's algorithm runs in $O(|V| + |E|)$, but every node has degree at most 2, so $|E| \leq |2V|$ and complexity is $O(n)$ where $n = |V|$.

Theorem 3.3 (Strategyproofness)**.** *Top trading cycles is strategyproof.*

Proof. Consider any student s and room r that is better than their assignment in TTC. By contradiction: s misreports, then receives r. Let i be the round where r forms a cycle in TTC with honest reports. We try to find the round j where s received r in TTC with dishonest reports. We know that $j > i$ otherwise s would've been able to receive r honestly (intuitively,) □

Theorem 3.4 (Efficiency of TTC)**.** *The TTC allocation is Pareto-optimal.*

Proof. The TTC allocation is the same as ordering students by iteration when they form a cycle (breaking ties arbitrarily) and then running serial dictatorship in that order as every student gets their most preferred room at the time they form a cycle.

We know that serial dictatorship is Pareto-optimal. □

3.1 Top Trading Cycles and Chains

What happens when a student graduates? They give up their old room but do not enter the market for new rooms. Similarly, there can be new students that do not yet have a room. Solution: combine ideas from TTC and RSD to get "You want my house, I get your turn".

Top Trading Cycles with Chains

1. Process all students in random order.
2. In student i's turn, mark i as visited and...
   (a) If i's top available room is unoccupied, assign that room to i and free i's old room.
   (b) Else-If the top available room is occupied by an *unvisited* student j, move j before i in the order.
   (c) Else-If the top available room is occupied by a *visited* student j, we have found a cycle.

This new mechanism still has the same nice properties: it's strategyproof, Pareto-optimal, individual rational, and has good runtime.

3.2 TTC(C) in Practice

Why is TTCC not used in practice? Running it once is strategyproof, but running it over multiple years is not. Students may try to get a popular room that they do not like to use it to get an even more preferred room the year after. This increases demand for perceived "popular" rooms even more.

Another application: school choice. New Orleans Recovery School District used TTC to assign K and 9-th grade students to schools in 2011-12. In public schools, some students have priorities (they have a sibling at the school or are in the district) that they can own and use to trade for in TTC. Next year, they switched to Deferred Acceptance. Explanation given by the board meeting was that DA is simpler to explain to parents and students. DA also has good publicity (2012 Nobel prize).

College admissions are different: universities have preferences over applicants that are a better fit. As such, it doesn't make much sense to allow a Stanford admit to trade their admission for a Harvard admittance.

Kidney transplants: over 100,000 people on the waitlist for a kidney transplant. Most humans have two healthy kidneys and can donate one before they die. However, there are blood type and other biological compatibility issues so even if a patient has a willing donor, it could be the case that the donor cannot donate to the patient.

Solution: kidney exchange, if patient x's donor can donate to patient y and patient y's donor can donate dto patient x, then the two patient-donor pairs can swap donors. Some other concerns in this setting:

- Strategyproofness is less of a concern: preferences are more determined purely based on blood type, etc. than personal preferences.

- People who are closer to death could have a higher priority.
- Not a repeated game, won't re-donate a received kidney.
- Stability and incentivizing people to join the central mechanism is more important.

However, there is a logistical challenge with long cycles: an entire cycle has to be operated on simultaneously. If not, one donor could change their mind or get ill and a patient who's donor has already donated would be left without a kidney. On the other hand, long chains are fine. Some other considerations:

- Dynamic problem: donors and patients arrive and leave over time and in an uncertain manner.
- Strategic hospitals: some hospitals might want to match kidneys internally (due to publicity or financial incentives). When this happens, the mechanism is not stable.
- High failure probability: 93% of matches actually fail (more compatibility issues are tested, donor illness, etc.).
- Ethical concerns: should we prioritize any patients over another?
- Multi-Organ exchange: trade a piece of liver for a kidney.

Practice

More notes in the ECON 136 course reader. Practice exercises here.

4 Equilibria in Games

In the mechanisms so far (except hospitals in Deferred Acceptance), agents had dominant strategies (best responses regardless of others' actions).Today, we analyze what agents should do when there is no dominant strategy.

Example 4.1 (Penalty Kicks)**.** Kicker can kick left or right. Goalie can jump left or right. Probabilities of scoring are in the following table: If the kicker kicks left, the goalie wants to jump left. But then, the kicker

	Kick Left	Kick Right
Jump Left	0.5	0.8
Jump Right	0.9	0.2

would want to kick right. Repeating this reasoning, there is no pure strategy equilibrium.

Solution. Assume we have: kick left/right with probability $0.6, 0.4$ and jump left/right with probability $0.7, 0.3$. Then the expected change to get a goal kicking left and kicking right are both 0.62. Expected chance to let a goal in jumping left and jumping right are both 0.62. As such, neither the kicker nor the goalie has an incentive to deviate from their mixed strategies. □

Definition 4.2 (Pure Strategy)**.** A pure strategy fully specifies how a player will play a game. Similar to a policy in RL and sequential decision making.

Definition 4.3 (Strategy Space/Set)**.** A player's strategy set is the set of pure strategies available to a player

Definition 4.4 (Mixed Strategy)**.** A mixed strategy is a probability distribution over pure strategies.

Definition 4.5 (Nash Equilibrium)**.** A profile of (possibly randomized) strategies such that no player has any profitable deviation keeping the other players' strategies fixed. Equivalently, a pair of (possibly randomized) strategies, such that neither player has an incentive to deviate.

4.1 Equilibrium in 2-Player Zero-sum Games

Assumptions: (1) 2 players and (2) the sum of payoffs for players one and two is always equal to zero (zero-sum game).

Let S_1 denote player one's strategy space and let S_2 denote player two's strategy space. Also, let $u(s_1, s_2)$ denote player one's payoff (so player two's payoff is $-u$). Then, player one solves

$$\max_{s_1 \in S_1} \min_{s_2 \in S_2} u(s_1, s_2)$$

while player two solves

$$\min_{s_2 \in S_2} \max_{s_1 \in S_2} -u(s_1, s_2).$$

Theorem 4.1 (Min-Max Payoffs)**.** *In a two player zero-sum game, the max-min payoff for player one is equal to the Nash equilibrium payoff, which is equal to the min-max payoff for player two (von Neumann, 1928).*

Proof. Comes from linear programming and duality (similar to the equivalence between max-flow and min-cut from 161). Can also be derived from the Min-Max Theorem of VNM. □

4.2 Aside: Max-Min Strategies

For more, see presentation from KLB at UCB.

Definition 4.6 (Some Game Theory Terminology)**.** Extensive form games have the following:

- Game Node: possible current states of the game.
- Game Tree: graph representing which game nodes are reachable from which other game nodes.
- Information Set: all game nodes consistent with/indistinguishable from my current node given my information.

Even though linear programming is "fast", it's not so nice in actual play.

Example 4.7 (Poker)**.** Consider "Heads Up" two player poker. There are 10^161 states which would be impossible to compute explicitly. Due to incomplete information, it does not suffice to only compute what happens at the current state (since my beliefs about what my opponents will do depends on their beliefs of what hand I have, which includes off-path possibilities).

Insight 1: blueprint. Can use 50TB space to store a single approximation strategy (10^13 states). Use this approximation strategy to play the current hand.

Insight 2: regret minimization. Define regret to be the payoff lost from not taking some action in hindsight. The regret minimization algorithm plays the game repeatedly, updating strategy so

$$\frac{\text{regret}}{\text{number of games}} \to 0.$$

Theorem 4.2 (Convergence of Regret Minimization)**.** *In two-player zero-sum games, both players using regret minimization converges to Nash equilibrium.*

4.3 Nash Eq in (non-zero sum) Games

Suppose the goalie tries to maximize the probability of a save minus the probability of a goal, so they goalie's payoffs are now

	Kick Left	Kick Right
Jump Left	-0.1	-0.8
Jump Right	-0.9	0.4

Good news:

Theorem 4.3 (Nash, 1951)**.** *In every finite game there exists a Nash equilibrium in possibly degenerate mixed strategies.*

Bad news:

- Not equal to max-min/min-max payoffs;
- Not unique;
- Not approached by regret minimization;
- Intractable to compute, even approximately;
- Some mixed strategy equilibria don't make sense.

Example 4.8 (Grade Game)**.** Two people do a group project. Grades are assigned as follows:

1. You send $x \in \{2, \ldots, 99\}$ and your partner sends y from the same set.
2. Assign grades by:
   - If $x = y$ then your grade is x;
   - If $x < y$ then your grade is $\min\{x, y\} + 2$;
   - If $x > y$ your grade is $\min\{x, y\} - 2$.

Unique Nash equilibrium: $x = y = 2$. Game theorists call this the "Traveler's Dilemma Game."

Example 4.9 (Intersection Game)**.** Consider two cars arriving at an intersection. Payoffs are:

- Two asymmetric equilibria: (Go, Wait) and (Wait, Go).
- Symmetric equilibrium: Both go with one percent chance and wait with one percent chance.

	Go	Wait
Go	$(-99,-99)$	$(1,0)$
Wait	$(0,1)$	$(0,0)$

- Correlated Equilibrium: (Go, Wait) with fifty percent chance and (Wait, Go) with fifty percent chance.

Definition 4.10 (Correlated Equilibrium)**.** A correlated distribution of actions that every player would rather follow (e.g., a stop light)

Good news about correlated equilibria:

- Can be computed efficiently (through linear programming or regret minimization).
- If everyone runs regret minimization, play converges to the set of correlated equilibria (cannot converge to a correlated equilibrium itself without a correlating device).

Bad news:

- Just like Nash eq, sometimes it's not unique or doesn't make sense.
- Not an equilibrium without a correlating device.

What happens if there is commitment? EG intersection game except the opponent is a dog who will always play go. Then, you will always choose to wait for the dog to go first.

Definition 4.11 (Stackelberg Equilibrium)**.** Strategy profile (Leader's strategy, Follower's strategy) such that

1. Follower's strategy is optimal given Leader's strategy;
2. Payoff is optimal for leader among all pairs satisfying #1.[5]

Remark. The Leader's strategy may be a sub-optimal response to follower's strategy. However, commitment power means that the leader's Stackelberg Equilibrium is greater than their utility in any correlated equilibrium. Finally, the follower's strategy is deterministic without loss of generality (so there exists an efficient linear programming alg to solve this).

Example 4.12 (Security Games)**.** Defenders commit to defense strategy (and is a leader); attackers can plan attack after observing defenders' strategies. Some settings:

- Airport security
- Infrastructure
- Cyber-security
- Anti-Poaching

[5]Strong vs Weak Stackelberg: which way we break follower indifference against or for the leader.

5 P2P File-sharing Dilemma

Napster (1999-2001) had free file-sharing. People were happy with free music but producers were unhappy because they could not get profits. Around $40 - 60\%$ of all college dorm traffic was Napster. It was shut down for copyright infringement.

From 2000 onwards, Gnutella provided a similar service but used decentralized peer-to-peer (P2P) file-sharing. Much harder to shut down.

P2P file-sharing rely on users uploading content for sharing. A large fraction of users are free-riders: people who only download but don't upload at all. 2000: around 65% of users were free-riders, growing to 85% in the next few years. It is important to incentivizing uploads.

We can model this as a game. Two players, each a user of Gnutella. Each player can either upload or free-ride. Payoffs are -1 for uploading, $+3$ for downloading. Payoffs are as follows:

	Upload	Free-ride
Upload	$2, 2$	$-1, 3$
Free-ride	$3, -1$	$0, 0$

The outcome that maximizes the two players' payoff is for both to upload.[6] However, free-riding is a dominant strategy for each player, so the dominant strategy Nash equilibrium is for both players to free-ride leading to the socially worst payoffs of $(0, 0)$.

5.1 Repeated Games

What happens if we iterate the game n times, and both players seek to maximize total payoff across all iterations? One modeling choice: whether or not agents can or cannot commit to future behavior is important.

Definition 5.1 (SPNE)**.** A Subgame Perfect Nash Equilibrium is a Nash Equilibrium that is also a Nash Equilibrium at any subgame of the original game.

Intuitively, this definition just means that agent cannot commit to suboptimal future behavior. Alternatively, it is a Nash equilibrium that is valid by backwards induction.

Remark. In this context, it means:

- On day n, agents will play Nash equilibrium.
- On day $n-1$, agents will evaluate their actions today assuming everyone plays Nash eq tomorrow and play Nash eq today.
- So on and so forth.

What is the subgame perfect equilibrium of the repeated game?[7]

- In the nth iteration, free-riding is the dominant strategy so both players free-riding will be played.
- In the $n - 1$th iteration, both players will free-ride anyways in the future so free-riding today is still dominant.
- By induction, free-riding is dominant in every iteration.

5.2 Repeated Games with Discounting

Same setting as before, but after ever iteration there is a probability p for the game to stop.

- the network gets shut down because of a lawsuit
- you or your peer gets disconnected.

[6]But this would create an externality on content creators, which is outside the scope of this model.

[7]Formally speaking, this is the unique subgame-perfect Nash equilibrium.

This is equivalent to modeling infinite repetitions with a discounted future

- there is some interest or inflation rate that decreases the value of future payoffs
- people subjectively value the future less than how much they value today.

Grim trigger strategy: if there was ever a time where the other player free-rides, then free-ride forever. Otherwise, upload.

Then, the other player's optimal strategy is either to always upload or never upload.

- Expected utility from always uploading is $2 + 2(1-p) + 2(1-p)^2 + \cdots = \frac{2}{p}$.
- Expected utility from never uploading is $3 + 0(1-p) + 0(1-p)^2 + \cdots = 3$.

Thus, the other player always uploads whenever $p < 2/3$ and both players playing grim trigger is a subgame perfect Nash equilibrium.

Remark. Note that always uploading for both players is NOT a Nash equilibrium as either player has a profitable deviation to always free-ride. Observe that the Nash equilibrium of both players using grim-trigger is strategically different but outcome equivalent.

For any run of the game, the number of iterations will be some number n. But we proved earlier that for any n, the unique SPNE is for both players to never upload. What changed? Players no longer know what n is, so the logic of backwards induction no longer holds.

In practice, grim trigger is not very useful: what happens if there was an accident and one person's connection was down? Not a stable solution concept.

Tit for Tat: In stage one, upload. In all future stages, do whatever the opponent did in the previous stage.

If one player commits to playing tit for tat, then the best response for the other player is to always upload when $p < 2/3$ and free-ride otherwise.

5.3 BitTorrent Strategies

BitTorrent main P2P protocol for file-sharing (almost 30% of all upload traffic). Users organized into swarms sharing the same file. A decentralized tracker coordinates active uses in the swarm. Each file is broken into pieces and all pieces are needed to complete a file.

Default Strategy:

- Every 15-30 minutes, users contacts the tracker and requests a new random subset of swarm peers. Number of known peers grows over time.
- Each user attempts to download from its peers.
- Each user has s slots and allocates $1/s$-fraction of upload bandwidth to each slot.
- $s-1$ peers to receive a slot chosen using a variant of tit-for-tat: prioritize peers that sent you the most data ($s-1$ as the last slot is reserved fr pity uploads to random peers for optimistic unchoking.)
- For each peer, priority given to uploading rare pieces.

BitThief Strategy:

- Never upload anything.
- Ask tracker for peers much more frequently so number of peers grows.

BitTyrant Strategy: send data to peers who *will* send you the most data, not who *has* sent you the most data.

- For each peer j, estimate amount of upload u_j so j unchokes you.
- For each peer j¡ estimate speed of download d_j if j unchokes you
- Prioritize sending as much data as possible to peer with maximum d_j/u_j.

6 Market Equilibrium

Previously, we studied situations with no transfers:

1. If Stanford housing auctioned rooms, the distribution would look much different.
2. It is illegal in all countries (except Iran) to buy organs.
3. Hospitals and students matching with money is subject to anti-trust laws.

However, there are issues to restricting markets to operating without money.

1. Can only work with ordinal opposed to cardinal preferences.
2. Underground markets may arise (college admissions scandals, black markets for kidneys).
3. Could be issues with permissions (entering donor lists multiple times using bots to try to get a match earlier).

Definition 6.1 (Game Equilbiria vs Market Equilbria)**.** We have covered four types of game equilbiria:

- Nash
- Correlated
- Stackelberg
- Subgame Perfect Equilibrium (SPE)

Today, we begin covering *market equilibria.*

Definition 6.2 (Cardinal vs Ordinal Utilities)**.** There are two main ways to model utility:

- Cardinal: assigns some numeric value to how much someone likes something.
- Ordinal: comparison-based, only assigns some order.

Cardinal utilities tell us more (we can always extract ordinal utilities from cardinal utilities, but not the other way around) but humans think more in terms of ordinal rankings. Cardinal utilities are also easier to work with when doing expected utility calculations and can be normalized to be in terms of (usually but not always) money.

Definition 6.3 (Fungible vs Idiosyncratic)**.** There are two types of goods:

- Fungible goods have many interchangeable units for sale.
- Idiosyncratic goods are unique.

Many goods are somewhere in between: rides on ride-sharing apps are fungible in terms of how the driver doesn't matter too much, but can also be idiosyncratic in terms of caring about where you go.

Definition 6.4 (Supply and Demand Curves)**.** From economics,

- A demand curve plots the total quantity of a good consumers are willing to buy at all price levels.
- A supply curve plots the total quantity of a good firms are willing to sell at all price levels.

By tradition, price is plotted on the y-axis and quantity is plotted on the x-axis.

Remark. In general, demand is downward sloping (higher quantity demanded as price decreases) and supply is upwards sloping (lower quantity supplied as price decreases). However, this is not always the case (different market structures like a monopoly, weird classes of goods).

Definition 6.5 (Market Clearing Price)**.** The price where the supply and demand curves meet is called the market clearing price. At this price, the quantity demanded is equal to quantity demanded so all interested buyers and sellers can transact (and hence clear the market).

6.1 Unit-Demand Market Model

Setting:

1. m different (idiosyncratic) goods for sale;

2. n different buyers that will each purchase at most one good (they have unit demand);
3. Buyer i as value $v_{i,j}$ for purchasing good j;
4. If buyer i pays p_j to acquire good j, then their payoff is

$$U_{i,j} = v_{i,j} - p_j;$$

5. If buyer i doesn't purchase any good, normalize outside utility to be zero:

$$U_{i,\varnothing} = 0.$$

The solution concept we will use:

Definition 6.6 (Competitive Equilibrium)**.** A competitive equilibrium is a price vector $p = (p_1, ..., p_m)$ and a matching $M : \{1, 2, ..., n\} \to \{1, 2, ..., m\}$ of buyers to goods such that:

1. Each buyer is matched with their favorite good given prices p: for all i, j we have

$$v_{i,M(i)} - p_{M(i)} > v_{i,j} - p_j;$$

2. If no buyer is matched with good j then $p_j = 0$;
3. Buyer i is unmatched if for all goods j, we have $v_{i,j} - p_j < 0$.

Remark. Conditions (1) and (3) implies that buyer i is unmatched if *and only if* for all goods j, we have $v_{i,j} - p_j < 0$. Then, this stronger condition is equivalent to individual rationality: for all buyers i,

$$v_{i,M(i)} - p_i \geq 0.$$

In addition to equilibrium, we also care about welfare:

Definition 6.7 (Social Welfare)**.** The social welfare of an allocation M is the sum of buyers' values:

$$U(M) = \sum_i v_{i,M(i)}.$$

Remark. Prices are not included because the total cost to buyers is equal to total profit of sellers.

6.2 Properties of Competitive Equilibrium

Theorem 6.1 (First Welfare Theorem)**.** *If (p, M) is a competitive equilibrium, then M is a matching that maximizes social welfare: if M' is any matching, we have*

$$\sum_i v_{i,M(i)} \geq \sum_i v_{i,M'(i)}.$$

Proof. By the definition of competitive equilibrium, we know

$$\sum_i (v_{i,M(i)} - p_{M(i)}) \geq \sum_i (v_{i,M'(i)} - p_{M'(i)})$$

as $v_{i,M(i)} - p_{M(i)} \geq v_{i,M'(i)} - p_{M'(i)}$ for all i. Then, $\sum_i p_{M(i)} = \sum_i p_{M'(i)}$ as M, M' are just permutations so we get

$$\sum_i v_{i,M(i)} \geq \sum_i v_{i,M'(i)}.$$

□

Theorem 6.2 (Existence of Competitive Equilibrium)**.** *In the unit-demand market model with finite prices and finite increments, there will always exist a competitive equilibrium.*

Remark. In more nuanced models, competitive equilibrium does not necessarily exist.

Proof. We will construct a Deferred Acceptance with Prices algorithm that gets us to competitive equilibrium. For each buyer, construct a list of all (good, price) options ordered by utility (we can truncate this list by not including anything worse than (receive nothing, pay nothing), which is always an option). At each iteration of the algorithm, every unmatched buyer whose next-favorite option is (j, p) proposes price p to good j. Then, good j tentatively accepts p (in the deferred acceptance sense) if it is higher than the prices it was previously offered. This algorithm will always terminate by the same reasoning as why deferred acceptance always terminates. The resulting match and prices when this algorithm terminates is a competitive equilibrium:

1. Every buyer is matched with their favorite (good, price) pair as any other pair must have led to the buyer being rejected at some previous stage.
2. Each unmatched good has a price of zero as a good is unmatched if and only if it was never proposed to.

□

Remark. DA with prices runs in $O(n \cdot m \cdot k)$ time where k is the number of price increments.

Proposition 6.8 (Properties of DA with Prices)**.** Deferred acceptance with prices is strategyproof and optimal for buyers.

Proof. Follows directly from the same reasoning used to analyze deferred acceptance. □

7 Market Failures

The assumption we previously made was that free markets naturally converge to an optimal outcome. However, this clearly is not always the case: market failures arise when markets fail to converge to an optimal outcome.

7.1 Externalities and Public Goods

Externalities are side effects on individuals other than the buyer or seller. **Public goods** are goods that are accessible and usable by everyone but not owned by anyone. For example:

- The environment is a public good, and activities that harm the environment (plane tickets) have negative externalities.
- WiFi bandwidth is a public good, and one person's usage imposes a negative externality of slower connections on others.

Two possible solutions.

1. Pigouvian tax: a tax on transactions proportional to the externality they generate.
2. Coasian bargaining: auction off public goods and rely on the Coarse Theorem to achieve social efficiency.

7.2 Transaction Costs

Also known as market frictions. Sometimes, transactions themselves can be costly to execute leading to mutually beneficial transactions not being made. For example:

- Agent one has a boat worth $9,900$ to them and agent two is willing to pay $10,000$.
- Agent one can sell the boat to agent two and both would be happier off.
- However, if there is a 9% sales tax on boats, this transaction would not occur.

Transaction costs do not necessarily have to be monetary:

- A hungry student is willing to buy an apple for 10 while other students have applies worth 1 to them.
- Other students can sell the apple to the hungry student and both would be better off.
- However, there are time/information costs for the hungry student to run around and ask other students if they have an apple on them.

7.3 Market Thickness

A market is thick (thin) if there are many (few) buyers and sellers. In thick markets, sellers and buyers have lots of options which makes prices close to competitive equilibrium. In thin markets, there are few buyers and if buyers come they face monopoly prices. Thick markets generate welfare-maximizing outcomes while thin markets do not.

Why are monopolies bad? Monopoly quantity sold is lower than quantity sold in competitive equilibrium; prices are higher and consumer surplus is lower.

First welfare theorem holds with reserve prices to cover costs. However, sellers setting reserve prices is not strategyproof.

Proof. If reserve prices are nonbinding, then sellers can increase reserve prices by $\epsilon > 0$ and still sell. If prices are binding, then the good is sold at the reserve price which is equal to cost and the seller makes zero profit. Assuming the necessary assumptions, sellers can increase reserve prices by ϵ and make strictly positive profit, which is a profitable deviation. □

Some ways to get around this:

- As markets become far from monopoly, it converges to a seller-strategyproof mechanism (strategyproof-in-the-large). However, this requires fungible goods in a thick market.
- If the seller doesn't know buyers' valuations, then it is difficult for sellers to set a reserve price. However, sellers collect lots of data.

How can markets be thickened?

- Encourage buyers and sellers to join and stay;
- Merge markets (larger kidney donation markets have typically led to better outcomes);
- Batch transactions: wait a bit for more buyers and sellers to join before running transactions.

7.4 Timing Issues

Market Unraveling: In certain markets, one side has incentives to make matches earlier and earlier (to get good students before competitions hire them). As a result, before introduction of the centralized matching mechanism for matching doctors and residencies even first year med students would be matched. However, whether or not a first year med student and a residency would be a good match is a bad predictor of whether or not the student and the residency is still a good match four years later.

Exploding offers: Job offers that require a very short response time. In markets where one side needs to look for other candidates if turned down, short response times can arise. Amplified when:

- if there is a limited time window for transactions due to regulatory issues;
- if one side of the market can't absorb variance in the number of matches received.

Not waiting for markets to clear: APPIC rules for psychologists in the 1970-1990's:

- All transactions must occur on selection day;
- Exploding offers are not allowed.

In theory, the result should be similar to Deferred Acceptance. In practice, employers didn't want to propose to candidates who might cancel right before the deadline leading to earlier and earlier "safe" commits.

How to resolve timing issues? Centralized matching systems will help. However, setting timing rules often fail due to incentives to violate rules being too strong ("Would you like to visit my hospital [wink wink]"?) Rules/norms that allow for accepting and then reneging from exploding offers removes the incentives to cheat and make exploding offers.

7.5 Information Asymmetries

Sellers know more about the quality of goods than buyers. As such, buyers might buy a good that they thought was worth it, but later find out that the good is worse than expected. For example, the market for lemons:

- Market for used cars, some good and some bad.
- Buyers willing to pay more for good cars than bad cars but do not know which type of car is which.
- Sellers willing to sell bad cars for less and do know which car is which.
- Let $g \in [0, 1]$ be the fraction of cars in the market that are good.
- Let $h \leq g$ be the fraction of cars that are in good condition and also for sale.

One bad equilibrium: only bad cars are sold at a low price and high cars aren't worth being sold at that low price.

One good equilibrium: if the portion of good cars is high enough, then the expected value of a car and hence price of a car is high enough to sustain good cars being sold.

Other examples where asymmetric information comes up:

- Health insurance: buyers know private information about their health status, sellers want to sell insurance to healthier individuals.
- Clickbait: content creators know if a link is clickbait or not while users do not. Creators what to maximize clicks, users want to avoid clickbait.

Adverse selection: situations where only the low quality goods/lemons stay in the market.

Possible solutions:

- Provide more information to both sides (mandatory disclosures, reputation systems);
- Disallow use of information (universal health care, anti-insider trading regulations);
- Filter out lemons (require warranties).

8 Single-Unit Auctions

Auctions are useful in many situations. For instance, if the price of a good is unknown (a market for this product, for one reason or another, might not converge to some equilibrium price) auctions can be useful for price discovery. For instance:

1. Monopolies (wireless spectrum auctions);
2. Niche products (rare used items on eBay);
3. Product with very specific attributes (ad auctions).

Another situation where auctions are useful is when buyers have the computational power and patience to bid:

1. Expensive items (wireless spectrum);
2. Automated bidders (ad auctions).

There are many interactions between auctions and CS:

1. Auctions fund a lot of computer scientists (ad auctions);
2. Fast auctions require algorithmic bidders (ad auctions);
3. Complex auctions require algorithmic auctioneers.

8.1 Case of One Buyer

Motivation: digital goods. Suppose there is a pool of users subscribed to a free product. What price should be charged to maximize revenue?

Aggregating users' willingness to pay induces a demand curve for the good. Then, revenue is equal to the price multiplied by the number of people with willingness to pay more than that price.

Roger Myerson (1981) showed that given some demand curve D, there is a formula $p(D)$ that maximizes revenue. This is the profit maximizing price.[8]

Another motivation: ad auctions with specialized advertisers so only one advertiser wants each ad spot. Sellers do not know exactly how much an advertiser is willing to pay, but have prior beliefs over the buyer's value. In this interpretation, the prior over values can be taken as a demand curve.

8.2 Case of Multiple Buyers

Model:

1. There is a set of bidders I with each bidder i having a value v_i for getting the item.
2. Each bidder knows their own value, the seller does not know bidders' values but has some prior belief over values.
3. Bidder i's payoff while paying p_i is $v_i - p_i$ if they get the item and $-p_i$ otherwise.

We will focus on sealed-bid auctions:

First-Price Auction

1. Each bidder submits a bid b_i, which is unobserved by other bidders;
2. Highest bidder $i^* = \arg\max_{i \in I}\{b_i\}$ gets the item and pays b_{i^*}.

Immediately, bidders will bid lower than their value v_i: for all $i \in I$ we have that $b_i < v_i$ in equilibrium. Intuitively, we expect bids to grow up as competition increases.

[8]In the case of n buyers and each drawing a value from D independently, the revenue-maximizing auction is a second price auction with reserve price $p(D)$.

All-Pay Auction

1. Each bidder submits a bid b_i, which is unobserved by other bidders;
2. Highest bidder $i^* = \arg\max_{i \in I}\{b_i\}$ gets the item;
3. *Every* bidder $i \in I$ pays b_i.

Not used to auctions things in practice, but used to model other situations (interest groups donating to a politician, animals hunting for food). In this model, bidders will still bid lower than their value and will generally bid lower than their first-price value.

Second-Price Auction

1. Each bidder submits a bid b_i, which is unobserved by other bidders;
2. Highest bidder $i^* = \arg\max_{i \in I}\{b_i\}$ gets the item and pays the second highest price $\max_{i \neq i^*} b_i$.

Theorem 8.1 (Equilibrium in Second Price Auction)**.** *Second-price auctions are strategyproof ao truthful bidding is a dominant strategy and hence everyone bidding truthfully is an equilibrium.*

Proof. Fix all bids of other people b_j for $j \neq i$. We will show that bidding $b_i = v_i$ is optimal for bidder i. Let

$$b^{(-i)} = \max_{j \neq i} b_j.$$

There are two cases:

1. If $v_i < b^{(-i)}$ then i prefers to not win than to pay $b^{(-i)}$ so any $b_i < b^{(-i)}$ is optimal. In particular, $b_i = v_i$ is an optimal bid.
2. If $v_i > b^{(-i)}$ then i prefers to win and pay $b^{(-i)}$ than not winning. Thus, any $v_i > b^{(-i)}$ is optimal and in particular, $b_i = v_i$ is an optimal bid.

□

Theorem 8.2 (Payoffs of Second Price Auctions)**.** *The second price auction is individual rational in equilibrium: if $b_i = v_i$ for all i, then $v_i - p_i \geq 0$ for all i.*

Proof. If i doesn't win, then their payoff is 0. If they do win, then the price they pay is upper bounded by their valuation, so $v_i - p_i \geq v_i - v_i = 0$. □

Theorem 8.3 (Optimality)**.** *In equilibrium, the second price auction allocates the good to the bidder with the highest value.*

Proof. In equilibrium, $v_i = b_i$ for all i so $\arg\max_i b_i = \arg\max_i v_i$. □

8.3 Revenue Maximization

Example 8.1 (Full Information)**.** Suppose A values the good at 1 and B values the good at 2. In a second price auction, both bid truthfully and B wins at a price of 1. In a first price auction, the unique equilibrium is A bids 1 and B bids $1 + \epsilon$ and revenue is $1 + \epsilon$ which is basically equivalent to 1.

To analyze the case with uncertainty and Bayesian agents, we need to define a new notion of equilibrium.

Definition 8.2 (Bayesian Nash Equilibrium)**.** A Bayesian Nash Equilibrium is a strategy profile such that each player is maximizing their own payoff with respect to the posterior belief generated by others' strategies.

Example 8.3 (Bayesian Agents)**.** Suppose A and B both have values drawn uniformly from $[0, 1]$. In a second price auction, expected payoff is $\mathbb{E}\left[\min\{v_A, v_B\} | v_A, v_B \sim Uni[0, 1]\right] = 1/3$. In a first price auction, the equilibrium is $b_A = v_A/2, b_B = v_B/2$ and in expectation, revenue is

$$\mathbb{E}\left[\max\{b_A, b_B\}\right] = \mathbb{E}\left[\max\{v_A, v_B\}/2\right] = 1/3.$$

It turns out that these auctions generate the same revenue:

Theorem 8.4 (Revenue Equivalence)**.** *At equilibrium, expected payments are fully determined by the auction's allocation rule.*

Corollary 8.4.1. *Taking the auction's allocation rule to be the one that gives the item to the bidder with the highest bid, first, second, and all-pay auctions generate the same revenue in Bayes-Nash Equilibria.*

It is not always the case that auctions always give the good to the max-value bidder:

1. There can be overbidding in second price auctions (A bids $v_B + 1$, B bids 0 is an equilibrium but doesn't give the good to the max-value bidder if $v_A < v_B$);
2. In an all-pay auction, there can be equilibria where the highest value bidder does not get the item;
3. In auctions with reserve prices, it could be the case that no bidder gets the item.

9 Sponsored Search Auctions

Used to auction ad slots on websites. Model:

1. There are N slots with slot $j \in N$ having associated click-through rate α_j which is assumed to only depend on j (so for instance, quality of the ad itself does not matter);
2. There are M advertisers with advertiser $i \in M$ having a value of v_i per click;
3. An allocation is a function $A : N \to M$ that matches the ith slot to the $A(i)$th advertiser;
4. Define social welfare to be
$$W(A) = \sum_j v_{A(j)} \alpha_j.$$

What is the optimal allocation A to maximize W?

General Second Price Auction

1. Ask each advertiser for a bid b_i;
2. Assign highest bid to first slot, second highest bid to second slot, etc;
3. For each slot j, advertiser $A(j)$ pays $b_{A(j+1)}$.

Remark. With the special case of one slot $j = 1$, GSP is equivalent to the second price auction.

The allocation rule in step two maximizes
$$\sum_j b_{A(j)} \alpha_j.$$

However, GSP is not strategyproof: some advertisers may prefer to win a lower slot at an even lower price than a higher slot at a high price.

9.1 Vickrey-Clarke-Groves Mechanism

Definition 9.1 (Externality). The externality of agent i is the difference in utility for all other agents when i is present versus when i is not.

Example 9.2. Suppose $v_1 > v_2 > v_3$ and there are two slots. In a welfare-optimal outcome, 1 gets $v_1\alpha_1$ for the top slot, 2 gets $v_2\alpha_2$ for the top slot, and 3 gets nothing. If 1 were not present, 2 gets $v_2\alpha_1$ and 3 gets $v_3\alpha_2$. Thus, 1's externality is
$$(v_2\alpha_2) - (v_2\alpha_1 + v_2\alpha_1).$$

VCG Mechanism

1. Ask each bidder for their valuation;
2. Find the welfare-maximizing allocation with respect to solicited bids in step 1;
3. Allocate slots via the welfare-maximizing allocation;
4. For each bidder i:
   (a) Find the allocation that maximizes welfare for all agents other than i;
   (b) Set i's payment equal to the difference between how satisfied everyone else is when i is present versus when i is not.

Theorem 9.1 (Strategyproofness). *The VCG auction is dominant strategy incentive compatible.*

Proof. Bidder i's payoff is
$$v_i(X) - p_i(X) = v_i(X) + \sum_{k\neq i} b_k(X) - \sum_{k \neq i} b_k(X^{-i}).$$

However, $\sum_{k\neq i} b_k(X^{-i})$ does not depend on the bid i submits so i's maximization problem is equivalent to maximizing $v_i(X) + \sum_{k\neq i} b_k(X)$. Since the VCG mechanism already chooses the socially optimal outcome, it is a dominant strategy for all individuals to truthfully report. □

In the context of sponsored search auctions, the assignment rule is still the same (highest bidder gets first slot, second highest bidder gets second slot, etc.) but payments are different. For slot α_j, advertiser $A(j)$ pays

$$\sum_{k>j} b_{A(k)}(\alpha_{k-1} - \alpha_k).$$

Another nice property of VCG auctions is that it is envy-free:

Definition 9.3 (Envy-Free)**.** An assignment (A, p) consisting of an allocation rule and prices is envy-free if for all advertisers i, j, advertiser i does not envy advertiser j: i's utility is (weakly) greater than i's utility if they got j's slot and paid j's price.

In the context of sponsored search auctions, this means that

$$\alpha_{A^{-1}(i)}(v_i - p_i) \geq \alpha_{A^{-1}(j)}(v_i - p_j)$$

for all $i, j \in M$.

9.2 Unnatural Equilibria

Example 9.4. Suppose there is a single item and there are two bidders with $v_A = 10, v_B = 9$. One equilibrium is $b_A = 7, b_B = 100$. Then, B wins the auction and pays \$7. This is an equilibrium (exercise for the reader to check this).

While no bidder has a profitable deviation, this outcome is not envy free: A would rather get B's outcome than their current outcome. Adding the envy free requirement gets the following:

Theorem 9.2 (Refinement of Eq)**.** *There is a correspondence between the following:*

- *Envy-free equilibria of the GSP action;*
- *Competitive market equilibria;*
- *Stable matchings between buyers and (price,good) pairs.*

As such, we can get two immediate corollaries:

- We can efficiently find equilibria using deferred acceptance with prices.
- The buyer-optimal (seller-worst) equilibrium corresponds to VCG payments.

9.3 GSP vs VCG in Practice

History:

- In late 1990's: Overture runs first price auctions;
- Early 2000's: Google, Yahoo, Bing start running GSP auctions;
- Late 2000's: Facebook runs auctions using VCG;
- Late 2010's: Google switches back to first price auctions.

Why the switch back to first price auctions? Many advertisers participate in different auctions using the same bids, and given fixed bids a first price auction makes more money than GSP, which makes more money than VCG.

Another trend: auctions are sensitive to bidder collusion.

- 2005: Bidding software must be authorized by search engine (easy to prevent collusion);
- Early 2010's: most ad bidding is through a small number of agencies (bad for competition and revenue);
- Late 2010's: ML based auto-bidders on Google, Bing, etc.

What about now? General move away from explicit auction rules:

- Auction details are highly optimized and hard to understand (a lot of things are ML);
- Big tech companies often know vales better than bidders (so they provide in-house bidders);
- Advertisers exploit the fact that different companies have to compete (less incentive to explain rules).

10 Pacing Equilibrium in First-Price Auction Markets (Guest Lecture)

Setting: bidding for ad slots. However, ad slots are dynamically sold over and over as users arrive over time. Furthermore, companies have budgets over all occurrences of the auction. What's the impact of extending our model to 1. multiple auctions and 2. budgets?

Example 10.1. Suppose there are two auctions and bidder i has value $v_{i,1} = 2$ in the first auction and $v_{i,2} = 2$ in the second auction. Bidder i has a budget of 3.

If $v_{-i,1} = 1.6; v_{-i,2} = 1.7$, bidder i cannot win both auctions as then they would need to pay $3.3 > 3$. As such, i would want to win the first auction and receive a payoff of 0.4 but not the second auction, which would only give a payoff of 0.3. Thus, i would not want to bid truthfully in the second auction.

Similarly, bidder i might want to conserve their budget for the second auction and shade their bid in the first auction (take $v_{-i,1} = 1.9, v_{-i,2} = 1.7$).

As such, repeated second price auctions with budgets may no longer be strategyproof. In the presence of budgets, what are the ecosystem properties of first price auctions?

10.1 Model

Input:

1. Set of buyers $N = \{1, 2, ..., n\}$ with generic element i;
2. Set of divisible goods $M = \{1, 2, ..., m\}$ with generic element j;
3. Each buyer has a valuation for each good $v_{i,j} \geq 0$ and a budget $B_i > 0$.

Find:

1. Pacing multipliers $\alpha_i \in [0, 1]$;
2. Fractional allocation $x_{i,j} \in [0, 1]$

such that bidding $\alpha_i \cdot v_{i,j}$ in auction j does not exceed the budget over the entire campaign.

Then, buyer i bids $\alpha_i \cdot v_{i,j}$ for good j which is then sold in first-price auction.

Remark. This is a static problem in the sense that all inputs are known immediately. All valuations and budgets are known and there is no strategic reporting.

Definition 10.2 (Budget-Feasible Pacing Multipliers (BFPM))**.** A set of BFPM's is a set of pacing multipliers α and an allocation x such that:

1. Unit price $p_j = \max_{i \in N} \alpha_i v_{i,j}$;
2. Goods go to highest bidders: $x_{i,j} > 0$ implies $\alpha_i v_{i,j} = \max_{i' \in N} \alpha_i v_{i,j}$;
3. Demanded goods are sold completely: $p_j > 0$ implies $\sum_{i \in N} x_{i,j} = 1$;
4. No overselling: $\sum_{i \in N} x_{i,j} \leq 0$ for all j;
5. Budget feasible: $\sum_{j \in M} x_{i,j} p_j \leq B_i$ for all i.

Remark. One (trivial) solution will always be $\alpha_i = 0$ for all i and any allocation that satisfies no overselling.

Definition 10.3 (First-Price Pacing Equilibrium (FPPE))**.** A first-price pacing equilibrium is a set of multipliers α and an allocation x such that

1. (α, x) is a BFPM;
2. No unnecessary pacing: $\sum_{j \in M} x_{i,j} \cdot p_j < B_i$ implies $\alpha_i = 1$.

10.2 Properties of FPPE

Proposition 10.4 (Existence)**.** There exists a Pareto-dominant BFPM. This BFPM does not have any unnecessarily paced bidders. Thus, it is an FPPE.

Proposition 10.5 (Uniqueness)**.** Any BFPM that is dominated by some other BFPM must have at least one unnecessarily paced bidder, so FPPE are essentially unique.

Proposition 10.6 (Revenue Dominance)**.** If $(\alpha^{(1)}, x^{(1)})$ and $(\alpha^{(2)}, x^{(2)})$ are both BFPMs with $\alpha^{(1)} \geq \alpha^{(2)}$, then we must have that the revenue in $(\alpha^{(1)}, x^{(1)})$ is weakly greater than the revenue in $(\alpha^{(2)}, x^{(2)})$.

Proof. We will show that prices nor quantity sold can decrease. First, prices cannot decrease as $p_j = \max_{i \in N} \alpha_i v_{i,j}$, weakly increasing α cannot decrease prices. Next, since both $(\alpha^{(1)}, x^{(1)})$ and $(\alpha^{(2)}, x^{(2)})$ are BFPMs, all demanded goods are sold completely. Since weakly increasing α cannot turn change a good from being demanded to not being demanded, quantity sold cannot decrease. □

Proposition 10.7 (Adding a Good)**.** In FPPE, adding a good weakly increases revenue.

Proof. Let (α, x) be the FPPE for N, M. Let (α', x') be the FPPE for $N, M \cup \{j\}$ for some $j \notin M$. Define $T = \{i : \alpha'_i \geq \alpha_i\}$ and $N \setminus T = \{i : \alpha'_i < \alpha_i\}$. Then,

1. Collectively, T spends at least as much in (α', x') as (α, x): If someone from T won a good before, it must still be someone from T winning the same good. Clearly, prices of goods won by T cannot decrease.
2. Collectively, $N \setminus T$ spends at least as much in (α', x') as (α, x): for any buyer in $N \setminus T$, their pacing multiplier α'_i is strictly less than one, which means they are spending their entire budgets in the new FPPE.

□

11 Prediction Markets and Information Cascades

Example 11.1 (Guessing Game)**.** How much does an Arctic Musk Ox weigh? At a country fair, the average of all guesses turned out to be correct. This example motivates using "the wisdom of the crowd". Why might asking a crowd be good?

1. Different people might have different perspectives or biases that balance out in a large crowd;
   - Elections
2. Information may be fundamentally distributed
   - State of a group project

Even in these instances, it is important to consider incentives: people might have an incentive to lie on polls due to social desirability, etc.

Example 11.2 (Iowa Electronic Market)**.** Prediction market hosted by the University of Iowa. For \$1, a better can buy a bundle of a D-contract and an R-contract where a D-contract pays \$1 if a democrat wins and an R-contract pays \$1 if a republican wins (so trades are zero-sum). Then, betters can trade via a continuous limit order book.

Definition 11.3 (Continuous Limit Order Book)**.** Let *Bid* denote the highest buy order and *Ask* denote the lowest sell order. Then, buy and sell orders arrive at any time and trade whenever a new buy order is greater than *Ask* or if a new sell order is less than *Bid.* The house (market) keeps any surplus from trade. Define the *Spread* at any point in time to be $Ask - Bid$. Any order that induces a trade upon arrival is called a Marketable order, while any order that does not induce a trade upon arrival is called a Resting order.

We can interpret the price of a D or R contract as the probability of a D or R win to aggregate the market's beliefs. At equilibrium prices, the number of D contracts is equal to the number of R contracts and the total budget of predictors who think D is more likely to win over p_D is equal to the total budget of predictors who think R is more likely to win over p_R

11.1 Liquidity in Prediction Markets

One challenge in some prediction markets is large bid-ask spreads. This leads to poor information aggregation. Why might spreads be large?

Definition 11.4 (Liquidity Providers)**.** People who buy low now and sell high later without caring about the final realization. Liquidity providers leave bid/ask resting orders on the order book that other investors can buy/sell with at any time. When liquidity providers buy and sell, they earn their spread.

In general, competition between liquidity providers drives spread down. Providing liquidity seems like a good deal, but prices can change.

Theorem 11.1 (No-Trade Theorem)**.** *If the market has already aggregated all available public information, there is no point in trading.*

It turns out this result still holds with private information:

Theorem 11.2 (No-Trade Theorem)**.** *If the market has already aggregated all available public information, there are no further trades even if some individuals have private information.*

Proof. Suppose someone has private information that leads to them wanting a trade. Then, no one would want to be the other person in the trade since trades are zero-sum so if the person with private information expects to make a profit, anyone that trades with them must be making an expected loss. □

11.2 Information Cascades

Information aggregation is a dynamic process. To model this:

1. There are n ordered agents $1, 2, ..., n$;
2. Going in sequence, agent i:
   (a) Observes a private signal;
   (b) Observes others' actions but not others' signals;
   (c) Chooses some action.

Example 11.5 (Urn Experiment)**.** Suppose an urn has red and blue balls, Either 2/3 balls are red and 1/3 are blue, or 2/3 are blue and 1/3 are red. Participants draw a ball and observe its color (with replacement to avoid probability issues), then sequentially guess if there are more blue or red balls. If the first two players draw a blue ball and guess that there are more blue balls, then even if someone in the future draws a red ball, they still will guess blue. This leads to an information cascade where players past the third ignore their information and blindly trust players one and two.

Suppose in truth, there are 2/3 red balls and 1/3 blue balls. Three cases of what happens in the first two rounds:

1. Correct cascade: $\mathbb{P}[Red, Red] = \frac{2}{3} \cdot \frac{2}{3} = \frac{4}{9}$;
2. Wrong cascade: $\mathbb{P}[Blue, Blue] = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$;
3. No cascade: $\mathbb{P}[Red, Blue \text{ or } Blue, Red] = \frac{2}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9}$.

Then, the probability of a correct cascade is four times likelier than a wrong cascade, so even with infinitely many samples there's a 20% chance of the crowd guessing completely wrong.

Example 11.6 (Stories of Information Cascades)**.**
1. Two pricing algorithms for the same book led to a spiral of misinforming one another, leading to textbooks priced at 20 million.
2. On 5/6/2010, stock markets lost around one trillion of value but mostly recovered in 36 minutes. Official theory: one trader allegedly caused the crash by trading from their parents' garage.

11.3 Legal Issues of Prediction Markets

In general, trading in prediction markets is essentially gambling which is mostly illegal in the United States (outside of Las Vegas). The IEM was granted a special exception by the Commodity Futures Trading Commission subject to some constraints (limited to 1000 trades, \$500 per trader). Other prediction markets have been forced to shut down.

However, "naked shots" can be legally bought for billions on Wall Street.

12 Fixing Congestion with Dollars and Data (Guest Lecture)

Networks: usually refers to computer networks (internet algorithms, cloud computing) but there can also be social networks. Another example of networks are travel networks (trains, roads, etc.). Incentives can be used to influence behavior of agents acting in networks.

12.1 Urban Congestion

Traffic not only is an annoyance for commuters, but have macro-level costs. Time and fuel wasted by traffic was:

1. \$115 billion in 2007;
2. \$121 billion in 2011;
3. \$160 billion in 2014;

while emissions due to traffic comprised a significant portion of overall emissions in major countries.

Congestion occurs when demand is greater than supply. Demand can be reduced by changing incentives (instead of penalties), while supply can be increased by using big data and mobile apps. On example of incentivizing good behavior instead of penalizing bad behavior is speed gun lotteries: if drivers were found to be *under* the speed limit, they are entered into a lottery for some prize.

Transportation architecture in the 20th century was operated by agencies (Department of Transportation, etc.) and used by commuters. However, there was no flow of information between the users and suppliers of transportation.

In the 21st century, there is more communication: commuters can use apps to get real-time information about capacity, while suppliers have access to large quantities of data to analyze transportation usage. Similar changes have happened in other industries, such as retail.

12.2 Large-scale Nudge Engines

Example 12.1 (Two Envelopes Game)**.** You are offered two envelopes. One has \$1000 with certainty, and the other has \$110,000 with 1% chance and \$0 with 99% chance. Most people would choose the first envelope.

Now, suppose one envelope has \$1 with certainty, and the other has \$110 with 1% chance and \$0 with 99% chance. In this care, most people would choose the second envelope.

In games with low stakes, players tend to be more risk-seeking. As such, lotteries or raffles should be used to pay rewards. Similarly, changing a lottery draw into a "game" where the player draws numbers to increase user engagement.

Friends also play a role in incentive programs: help with recruitment and can influence behavior. For instance: Insinc (Incentives for Singapore Commuters) had 3,500 participants at launch. A month later, the Recommend-a-friend system launched and sign-ups increased right away.

1. Jan: 3,500 users;
2. Feb: Recommend-a-friend launched;
3. Mar: 8,084 users, 20% were from friends;
4. Apr: 11,851 users, 30% were from friends;
5. Jun: 22,000 users, 50% were from friends;

Friends also influence how likely an individual sticks to a program: participants with no friends in the program stay around 50% of the time, but participants with 9+ friends in the program stay around 90% of the time.

12.3 Singapore MRT

Goal of Insinc was to incentivize offpeak travel. In 2016, project had 400,000+ participants; program acquired by Google and ended. Similar initiative started but under different name.

Commuters tap in at different MRT stations, which generates a commute history for every participant. Points are given for any commute, with three times the points awarded for commutes outside of peak hours. Rewards are given based on points.

Data from this project corroborates with previous results: people with friends who joined did better and people who played the reward game instead of getting a fixed payoff were more engaged.

Is the program sustainable? It costs the program to provide rewards but the program also leads to additional usage of public transport. On average, a user takes one more weekday trip on public transport. This makes up the cost of awarding users.

13 Proof-of-Work Blockchains

Definition 13.1 (Digital Signatures)**.** A digital signature consists of the following:

1. a public key and a secret key;
2. a signature function sign;
3. a verification function verify.

To show agreement with statement x, set $y \leftarrow sign(x, secret_key)$. Anyone with the public key can then $verify(x, y, public_key)$ to make sure the person with the secret key agreed with x.

Definition 13.2 (Cryptographic Hash Function)**.** A hash function H such that given some input a, it is easy to compute $H(a)$ but given some output b, it is difficult to compute $H^{-1}(b)$.

In general, the fastest way to "invert" a cryptographic hash function is to randomly try inputs. To get a such that $H(a)$ agrees with b for the first k bits requires 2^k tries.

Consider Alice transferring $1 to Bob. If done through a bank or other centralized institution, the intermediary can verify that Alice does indeed have $1 to give to Bob. However, can this transfer be done without a bank?

Bob may require Alice to verify that the $1 exists and is transferred. One way Alice may do this is to provide a history of where the $ has been for Bob to track Alice's history of payments and make sure Alice has not already spent the dollar.

Definition 13.3 (Ledger)**.** A ledger is an ordered history of payments. A decentralized ledger is not controlled by any one entity and can continue to work even if a bank is hacked (Robinhood, etc.). Consensus occurs when all entities agree that a transaction happened.

Ledgers are used to verify ownership of capital. In the early 1980's, Byzantine protocols (BFT's) can reach consensus if a certain fraction (around 1/2 to 2/3) of actors are honest. However, who can join?

Definition 13.4 (Permissionless)**.** A network is permissionless if anyone can join or exit at any time.

While this is a nice property to have, verification relying on a majority of users is useless as anyone can simply create more copies of themselves to obtain an artificial majority. Alternatively, votes can be tied to more concrete things:

- Proof of Work: 1 cpu equals one vote;
- Proof of Stake: 1 coin equals one vote;
- Proof of Space: 1 bit equals one vote.

13.1 (Simplified) Bitcoin Protocol

Ledger is constructed as a chain of blocks. Each block contains:

1. Some account of payments or transfers;
2. A cryptographic hash of the previous block;
3. A *nonce*.

What is a nonce? A nonce for a block is valid if the hash for a new block (which includes information about payments, hash of previous block, and nonce) has k leading zeros. In general,

$$\mathbb{P}\left[\text{nonce is valid}\right] = \frac{\text{number of hashes tried}}{2^k}.$$

A nonce is one way to establish Proof of Work: it proves that a lot of hashes were tried (and some luck was involved). Miners compete to find the first valid nonce and create ("mine") new blocks. Winners are rewarded in the form of a transaction in the new block. The k leading zeros that need to match is adjusted dynamically so each block takes around 10 minutes to mine in expectation.

To verify a single payment from Alice to Bob in block b_t:

1. Verify that Alice has signed the payment using Alice's public key;
2. Verify that Alice has received this \$1 in some previous block;
3. Verify that Alice has not spent the \$1 in some previous block.

To verify an entire block:

1. Verify all payments individually;
2. Verify that b_t correctly hashes b_{t-1};
3. Verify the nonce.

For Alice to send money to Bob, a signed transaction is sent to all miners. Then, Alice and Bob wait for the next block to be mined.

What happens if two miners find different nonces for a new block at the same time? Two new blocks could be made and there is no longer consensus. In this case, the standard protocol is to extend the longest chain available: if block b_t and $\hat{b_t}$ both extend b_{t-1} but b_t has a chain to b_{t+n} while $\hat{b_t}$ has a chain that goes to $\hat{b_{t+m}}$ for $m < n$¡ then extend b_t over $\hat{b_t}$.

13.2 Incentive Issues with Big Miners

Original idea (2000's): computing power would be distributed about equally over all people so each individual miner has a small fraction of total power. However, emergent trends in the 2010's disincentivized this:

1. Rise of corporations taking over major computing power (the cloud);
2. Crypto-specific hardware that is much more efficient (and expensive);
3. Cheaper energy that miners might have access to;
4. Creation of mining pools where professional miners organize to reduce variance in gains from mining.

Recall the longest chain rule: if one miner has access to more than half of computing resources, they can take advantage of this to "overwrite" blocks they don't like by forcing a fork in the chain. For instance, Alice can pay Bob and encode that in a block that is overwritten as a result of a fork, causing the payment to not actually go through.

Even with $\alpha < 1/2$ of computing power, Alice can try to carry out the same attack and rely on luck: if Alice can mine two blocks before any other miner extends the blockchain to the block where Bob gets paid, Alice successfully avoids needing to pay Bob. Probability this succeeds is α^2. As such, best practice is to wait $d > 1$ blocks before accepting payment.

13.3 Selfish Mining

Suppose there is a selfish miner who seeks to break the system. Furthermore, suppose their connection is so fast that all longest-chain ties are broken in their favor. What can they do? Consider the following strategy:

- Mine blocks.
- If block is mined, do not reveal the new block, but mine follow-ups to that block.
- If another miner mines the next block, reveal the block you have already mined.

Nothing is lost by waiting before revealing blocks: since ties are broken in favor of the selfish miner, their block is still the block that goes through. However, there is some benefit to them: the rest of the network wastes time on mining a block that eventually becomes orphaned. If the selfish miner has a fraction α of computing power, their *effective* relative hashrate is $\frac{\alpha}{1-\alpha} > \alpha$ as for every block they mine, other honest miners waste a block.

What if ties were always broken against the selfish miner? In this case, the selfish miner risks losing their additional (hidden) progress in case of a tie. However, what if the selfish miner can mine two blocks in a row to ensure no ties occur? Selfish mining can be profitable in this case if $\alpha > 1/3$ opposed to the previous $\alpha > 1/2$.

This effect becomes more pronounced if the honest miners then respond to the selfish miner's actions: if honest miners know the selfish miner is always going to “win”, they might leave the blockchain (to save energy or mine other blockchains), leaving the selfish miner with an even larger portion of hashrate.

13.4 Big Miners in Practice

In 2014: GHash.io controlled more than 51% of computing power but did not execute attacks. Instead, they encouraged miners to leave to prevent the value of bitcoin from dropping. For smaller coins with lower total hash rate, the issue is larger since less computational power is needed to achieve majority control. Furthermore, attackers may care less about the value od smaller coins. The 51% attack happened multiple times to “Ethereum Classic”.

14 Proof of Stake

Continuing the analysis of big miners, proof-of-work or longest chain protocols leads to fragile incentive structures. Luckily, the benefit is that pools or miners with high portion of computing power want the value of a coin to go up. However, this argument falls apart if mining hardware can be repurposed or if the blockchain is small.

In proof of work (PoW), hash power is used as a proxy of how much a miner cares about a blockchain. But why use a proxy?

At each iteration of a vanilla proof of stake (PoS) protocol,

1. A random coin is sampled;
2. owner of that coin is asked to create the next block.

Intuitively, people with a lot of coins care about continued success of the blockchain, while people with few coins are unlikely to be asked to create the next block. PoS has more robust incentives and reduces competition through computing power (which is good for the environment).

Mining PoS is computationally cheap, but still requires servers with a robust connection. In practice, over half of servers are hosted by Amazon (which means it's not quite decentralized, but if Amazon goes down the world is half over anyways).

14.1 Cryptographic Preliminaries

Definition 14.1 (Pseudo-random Function)**.** Deterministic function that you can share code for, but looks like a random function for computationally bounded algorithms.

Remark. For the purposes of this class, we will treat pseudo-random functions as "cryptographic black magic".

Definition 14.2 (Verifiable Delay Function)**.** A function f that is similar to a Hash function (given a, it is easy to verify that $f^{-1}(a) = b$) such that parallelization does not help speed up how long it takes to find a.

Definition 14.3 (Cryptographic commitment scheme)**.** Prevents people from changing responses ex-post. Inputs: a secret x and a random string r.

Protocol:

1. In this period, one agent sends $c = \mathit{commit}(x, r)$;
2. In the next period, they send x, r.

Guarantees:

1. Any other agent can verify that $c = \mathit{commit}(x, r)$;
2. But it is computationally intractable to ind x', r' such that $c = \mathit{commit}(x', r')$.

How to introduce randomness in a decentralized, permissionless, strategyproof consensus protocol? Some possible approaches are:

1. Ask the tth miner to choose the $(t+1)$th miner at random;
   - Not strategy proof: the tth miner could choose themselves.
2. Pseudo-Random function of entire tth block;
   - Still not strategy-proof.
3. External event like stock prices or the weather;
   - Not decentralized (have to agree on what external event) and not strategyproof (stock prices are manipulable).
4. Some decentralized coin-flipping protocol;
   - Not permissionless.

Some other issues with PoS protocols:

1. Nothing-at-Stake: in PoW, there theoretically should only be an incentive to extend the longest change. However, in PoS miners can try (for free) to extend every block;
   - Doesn't work for PoW since tie-breaking is based on computational power, not randomness.
2. Predictability Attacks: one possible way to generate pseudo-randomness is to use a pseudo-random function of the first block and the current block. However, miners might be able to predict when it's their turn to create blocks allowing for double-spending attacks.
   - Very difficult to predict who mines the next block in PoW protocols.
3. Long range attack: attacker buys almost all coins at time t and then sell all coins at time $t + 1$. By some time $t + k$, buyers are satisfied and pay attackers off chain so attackers are no longer staked. Attacker can fork a chain from b_t in which they still own all the coins and can mine much faster.
   - Information about who gets to mine the next block is a function of the block in PoS, but who gets to mine the next block is independent of block in PoW. Thus, rewriting the history doesn't change how much computing resources you have.

14.2 Approaches to Implementing Proof-of-Stake

1. Slashing: fining coins from miners that misbehave.
   - Mitigates the noting-at-stake problem by slashing miners that produce blocks pointing to conflicting blocks;
   - However, this needs other miners to vote on who is misbehaving;
   - Network issues might look like misbehavior (potentially is fine since network issues also cause loss of blocks in PoW);
   - Miners need to place slashable coins in escrow so they can't spend it before it gets slashed (capital inefficiency).
2. RANDAO: multi-agent randomness. Agents select a secret random bit and send each other commitments. Then, actual bits are sent and verified, and output is a function of all agents' bits. Then, only one truly random agent is sufficient for the output to be random.
3. Verifiable Delay Functions: randomness is a VDF of a random event, like stock prices. Then, even if a miner wants to manipulate stock prices, the VDF means that they do not know how to manipulate prices.
4. Byzantine coin-flipping Protocols: select random coin instead of a random attack time. Byzantine fault-tolerant (BFT) protocols can reach consensus on random bits. Technically not permissionless, but even more technically PoS protocols are also not permissionless (coins are permission). These protocols may also be challenging to scale.
   - Can randomly select a committee of active Byzantine generals; then each committee selects the next committee using a Byzantine protocol.
   - Requires participants to be online (can slash participants that are offline for too long).
5. Finality/Checkpoints: in pure permissionless longest chain, agents are never 100% sure that the chain they are currently seeing is really the longest chain out there. For some transactions, waiting to transact with bitcoins is prudent, but maybe not feasible for other transactions. If 2/3 of holders agree (using BFT) on a block, consider it final. Treat final blocks as new genesis blocks. Mitigates long-range attacks if attackers selling their stake takes longer than the gap between checkpoints.
6. Staking: network participants lock a stake which corresponds to how likely they are chosen to create the next block. Receive fees for each time they create a block.
   - Aligns incentives of miners with cryptocurrency;
   - Prevents sybil attack: can't change the amount of money clones have;
   - Stakes can be slashed;
   - Raises money for blockchain startups.

   Why run own miner if you can lend someone your coins and have them mine for you? Still receive rewards proportional to the number of coins owned, convenient, and funds are still liquid. However, the delegee might have warped incentives: if they get slashed, it's still your coins that are being lost.

15 Security Markets

15.1 High Frequency Trading Arms Race

In 2010, Spread Networks invested \$300 million to create a new straight-line optic fiber cable from Chicago to New York, reducing latency from 16ms to 13ms. High frequency traders race to make trades first, which is why they value the decrease in latency. However in 2011, microwaves brought latency down to 10ms.

Is this a market failure? From Spread Network's perspective, the cable might have made back its investment (and then some profit), but even then the opportunity cost of the investment might have been high as well. Furthermore, does high frequency trading in and of itself even benefit society?

15.2 Stock Markets

We build on the definitions introduced in Section 11 (Prediction Markets and Information Cascades).

Definition 15.1 (Naive Investors)**.** Individuals who want to buy/sell stock as a means to store or liquidize assets. Naive investors do not have a primary goal of profiting from private information.

With just naive investors, there can be market frictions: there may not be stocks to buy or sell when a naive investor wants to buy or sell. Furthermore, the no-trade theorem does not apply as these investors derive value from the act of buying or selling itself instead of purely caring about the monetary value of their assets.

As before, liquidity providers leave resting orders and buy low, sell high. They provide an intermediary for naive investors to trade with, but naive investors lose the spread that liquidity providers gain. As such spread is part of naive investors' transaction costs.

A stock's "true" value represents aggregate beliefs about the stock's returns. In equilibrium, a stock's true value is between the stock's bid and ask prices (otherwise, some agents have a profitable deviation to buy or sell the stock). True value may vary depending on external events. For example:

1. Correlated Stocks/Securities: If someone buys gold in Chicago, the price of gold in New York increases. If there is demand for security X increases and security X is correlated with security Y, then the price of Y will go up.
2. Fed announcements: Fed made announcement on Sept 18, 2013. Markets reacted faster than the speed of light.
3. Twitter: Trump tweets (about tariffs, the Fed, covfefe) changed stock values.
4. Reddit: wallstreetbets and Gamestop stock.

Definition 15.2 (Snipers)**.** Agents that wait for changes in stock values and rush to trade with liquidity providers' standing orders before those standing orders can be canceled. As such, being faster than competitors can be profitable.

In response to snipers, liquidity providers might increase spread to decrease sniping opportunities. However, this hurts naive agents by increasing their effective transaction costs.

15.3 Market Failure Fixes

How can we resolve these market failures?

1. **Symmetric Speed Bumps:** all orders get delayed by the same amount.
   - Doesn't solve the speed arms race, nor does it remove sniping risk for liquidity providers.
2. **Random Speed Bumps:** each order is delayed by some random amount.
   - Mostly solves the speed arms race (as luck in random difference is more important than speed differences), but makes the sniping risk worse: if a single sniper gets lucky, then the sniping succeeds.

3. **Sniper-Only Speed Bumps:** only delay snipers.
    - If implemented, this would resolve the speed arms race and sniping risk, but it is difficult to determine who is a sniper and who is not.
4. **Frequent Batch Auctions:** batch orders for some short interval and find the market-clearing price at each batch.
    - Everyone now has time to react to value-changing events as orders are prioritized by price instead of arrival (as long as the order arrives in the same batch).

15.4 Blockchain Flashboys

Definition 15.3 (Smart Contract)**.** Agreement between two agents for one transaction (borrowing) in the current period and a second transaction in some future period (paying back, perhaps with interest). If the second transaction does not happen, some collateral promised by the first agent is given to the second agent.

The collateral needs to be sent simultaneously with the first transaction (otherwise one agent can abort after getting money).

Definition 15.4 (Atomic Transaction)**.** A transaction such that either all steps are executed successfully or all steps are canceled.

One application of smart contracts is in decentralized finance (DeFi): cryptocurrencies, stocks, and contracts are traded. Limit order books and automatic market makers can be automated via smart contracts.

Definition 15.5 (Arbitrage)**.** Simultaneously buying low in one market while selling high in another.

Arbitrage is especially attractive with decentralized finance:

1. Risk-Free Atomic Arbitrage:
    - In centralized finance, the arbitrageur takes takes risk (if they buy low first, then the high price they wanted to sell at may drop before they get to selling).
    - However, bundling buy and sell actions into a single atomic transaction resolves this risk.
2. Cryptocurrencies are highly volatile, leading to more sniping opportunities;
3. Information (code) is made public so full code and state of smart contract automated market makers are available. As such, it is easy to exploit bugs.

Definition 15.6 (Front-Running)**.** Transacting right before another agent to take advantage of the distortionary effects of the latter's transaction.

In general, this is illegal in centralized finance. In decentralized finance however, all transactions are made publicly known before blocks are mined. Now, suppose miners are also traders and an arbitrageur finds an opportunity. If the arbitrageur broadcasts this transaction, then miners can execute the same trade on their own account. Miner includes the arbitrageur's transaction on their block but after the miner's own transaction.

15.5 Miners' Extractable Value (MEV)

In DeFi, arbitrageurs compete to be at the top of the next block via transaction fees. Then, a miner's MEV is how much they can charge an arbitrageur. Miners receive rewards for each block they mine and additional transaction fees. Originally, the value for mining the block itself is greater than the MEV, but in recent years block rewards have went down (bitcoin rewards halve every 4ish years) while MEVs have gone up (arbitrageurs increasing demand for transactions). When MEVs come into play, not all blocks are created equal.

As such, miners can access a new type of fee-sniping attack: if block $\hat{b}$ has a larger MEV than block b, then miners would prefer to mine $\hat{b}$ over b. Then, a miner with fraction α of computing power profits from a fee-sniping attack if

$$\alpha^2 \cdot \text{higher MEV} > \alpha \cdot \text{lower MEV} \iff \alpha > \frac{\text{lower MEV}}{\text{higher MEV}}.$$

Success probability is even higher if other miners also try to attack instead of extending b_t.

With selfish tiebreaking, miners want to extend the block with higher MEV opposed to the block that was seen first. Furthermore, a miner can incentivize other miners to help extend their block by leaving some MEV leftover in the next block (known as undercutting). Some possible ways to undercut these issues:

1. Transaction fees distributed among miners that mine the next k blocks;
2. Capped transaction fees;
3. "Burn" transaction fees instead of giving them to miners.

However, arbitrageurs and miners can always make side payments.

16 Envy and Fairness

Consider cake cutting: Alice and Bob want to split a cake, and the cake has different toppings (strawberries, chocolate, etc.) over different regions. One canonical mechanism is I-cut-you-choose:

1. One agent cuts the cake;
2. The other picks which slice they want;
3. First agent gets the remaining cake.

Proposition 16.1. I-cut-you-choose is not dominant-strategy incentive compatible for the cutting agent but is for the choosing agent.

Proof. The cutter's optimal strategy depends on the chooser's preferences. The chooser directly chooses their outcome. □

Why, has this mechanism been used? There is some notion of fairness in this:

Definition 16.2 (Envy-Free)**.** An allocation is envy-free if no agent envies another agent's allocation.

Proposition 16.3. The I-cut-you-choose mechanism can always lead to an envy-free allocation for both players.

Proof. The cutter can always cut the cake into two pieces that are equal in their perspective, so no matter which slice they get the allocation is envy-free for them. Then, the choose chooses their favorite slice, so it clearly is envy-free for them. □

16.1 Formalizing Fair Allocations

Goal: from a set of items A, choose some allocation for each of n agents. Each agent i has a valuation function $v_i : \mathcal{P}(A) \to \mathbb{R}$. Then, there are several measures of fairness (defined in this setting):

- Envy-Free: for all i, j we have $v_i(A_i) \geq v_i(A_j)$;
- Proportional: for all $i, v_i(A_i) \geq v_i(A)/n$;
- Equitable: for all i, j we have $v_i(A_i) = v_j(A_j)$;
- Perfect: for all $i, v_i(A_i) = v_i(A)/n$.

Another possibility: competitive equilibrium from equal incomes, which is similar to competitive equilibrium except give everyone a "fake dollar" to spend.

In cases with indivisible goods, often none of them can be satisfied. In particular, if there is a single good (and both individuals gain strictly positive utility from it), none of these can be satisfied. A compromise:

Maximin Share (MMS)

1. Each agent i proposes an allocation A^1 to all agents;
2. The MMS condition is
$$v_i(A_i) \geq \max_{A^i} \min_j v_i(A^i_j)$$
.

Another compromise can be to approximately satisfy these conditions: let $\overline{a}$ be the best item and $\underline{a}$ be the worst item. Then,

1. Envy free up to one item: $v_i(A_i) \geq v_i(A_j \setminus \overline{a})$;
   - Can always be satisfied.
2. Envy free up to any item: $v_i(A_i) \geq v_i(A_j \setminus \underline{a})$
   - Open problem for whether or not this can always be satisfied.
3. α-Maximin Share: $v_i(A_i) \geq \alpha \max_{A^i} \min_j v_i(A^i_j)$
   - Can always be satisfied for $\alpha \leq 3/4$.

4. Approximate competitive equilibrium from equal incomes: only require approximate equilibrium.
    - Can always be satisfied but computationally difficult to compute

Finally, we can directly seek to maximize some "fair" measure of overall utility. Some examples are:

1. Utilitarian/Social Welfare: $\max_{A_1,\dots,A_n} \sum_i v_i(A_i)$;
2. Nash social welfare: $\max_{A_1,\dots,A_n} \prod_i v_i(A_i)$;
    - Middle ground between Utilitarian and Egalitarian.
    - Competitive equilibrium from equal incomes maximizes Nash social welfare;
3. Egalitarian/maximin: $\max_{A_1,\dots,A_n} \min_i v_i(A_i)$.

16.2 Dominant Resource Fairness

Suppose there are m resources with fixed capacities and n users that want to run as many identical tasks as possible. Each task demands some vector of resources to run. For now, suppose resources are infinitesimally small. How can resources be allocated fairly and efficiently without transfers?

For any allocation, each user has some fraction of total capacity of each resource (given by point-wise division of the user's usage vector by the capacity vector). Then, a user's dominant resource is the resource they have the highest fraction of usage of. Dominant resource fairness requires equal shares of dominant resources.

Algorithm to get DRF Allocation

1. Every agent submits their ratio of demands;
2. Algorithm computes the maximal allocation subject to DRF.

Proposition 16.4. Every user prefers the DRF allocation to just getting $1/n$ of every resource.

Proof. As the allocation is maximal, some resource j^* is exhausted so some agent i^* gets at least $1/n$ of j^*. Then, each agent has at least $1/n$ of their dominant resource. □

Proposition 16.5. The DRF mechanism is envy-free.

Proof. If i envies j, then j must have more of every resource so in particular, j must have more of i's dominant resource than i does, a contradiction. □

Proposition 16.6. The DRF mechanism is Pareto-optimal and strategyproof.

Proof. Pareto-Optimal: As the allocation is maximal, some resource ie exhausted and no agent can be made happier without making some other agent sadder.

Strategyproof: for any misreport, Pareto-optimality of the allocation reached by truthful reporting means that less of the dominant resource is received. □

16.3 DRF in Practice

Some of the assumptions made during the setup might break:

1. Tasks are not identical, arrive over time, and depart;
    - While resources are not exhausted, select user i with minimal dominant share and add their next feasible task. Not strategyproof (can clump tasks together) but works well in practice.
2. Tasks might have flexibility in resource use;
3. Tasks might have more specific criteria;
4. Tasks not infinitesimally small;
5. Users might have different priority levels;
    - Weighted DRF.

6. Some tasks have zero demand for some resources.
    - Find the maximal DRF allocation and then recurse on remaining feasible tasks.

17 Course Selection

Many discussion, project, or sports based classes have enrollment caps. Some of those classes are extremely important and possibly change entire career trajectories. Other classes (like CS classes) are essentially uncapped. How should limited seats be allocated?

At Stanford, class registration is essentially (random) serial dictatorship depending on how fast internet is for the registrant. Other institutions have serial dictatorship based on other factors such as seniority. Serial dictatorship is strategyproof and Pareto optimal, but not very fair: dictators get all their favorite classes, while people later on might get none of their favorite classes.

Another possibility is an auction-based mechanism: use fake money (100 points) to bid on classes in an all-pay auction. Why all-pay? Since the money is fake, it is lost at the end of the auction anyways. However, this makes the auction not strategyproof. Furthermore, it is difficult to express complex preferences, such as when classes are substitutes for one another: "either this class or that class".

Finally, competitive equilibrium from equal incomes does fairly well: smaller courses will be more expensive reflecting the size constraint, while larger courses will be relatively cheaper. While there are some distortionary effects of each individual's demands, this mechanism is strategyproof in the large: approximately strategyproof for large markets. In a large market, each student has a small effect on prices so prices are functionally fixed. As such, students all independently optimize to get their best outcome. Furthermore, initial incomes are equal so the mechanism is envy-free as well.

One potential problem is that a competitive equilibrium might not exist. However, if we settle for approximate equilibrium, one will always exist. The mechanism that implements approximate competitive equilibrium from equal incomes is approximately strategyproof (strategyproof in the large), approximately Pareto optimal, approximately fair (envy-free up to one good), and always exists.

The final problem: it may be computationally difficult to find. In general, it's intractable and about as hard to compute as Nash equilibrium (which is hard). In practice, real instances tend to be easier, and inputs are not too large. Even then, taking a few days isn't too bad if it only needs to happen once ever quarter or semester. Recently, there's been a faster heuristic algorithm. Using this and looking at incentives, we would need more students than atoms in the universe to be large enough for strategyproofness in the large to hold. It is possible to manipulate the algorithm, and using these insights the mechanism can be re-designed to be "more strategyproof".