Algorithmic Game Theory and Applications

Lecture 1: What is game theory?

Kousha Etessami

Basic course information

- **Lecturer**: Kousha Etessami;
  Email: `kousha@inf.ed.ac.uk`
- **Lecture times: Monday&Thursday**, 11:10-12:00; All lectures are LIVE and ONLINE on Blackboard Collaborate (access the lectures via the LEARN page for the course).
  Tutorials: weekly; Starts in week 2, ONLINE on BB, a weekly "universal tutorial and Q&A with Kousha", on Mondays from 15:10 – 16:00, which will be recorded. Hopefully also separate in person tutorial sections for more individual Q&A.
- *Course web page* (with lecture notes/reading list):
  `http://www.inf.ed.ac.uk/teaching/courses/agta/`

- **No required textbook.** Course based on lecture notes + assigned readings. *Some reference texts:*

M. Maschler, E. Solan, & S. Zamir, *Game Theory*, 2013.

M. Osborne and A. Rubinstein, *A Course in Game Theory*, 1994.

R. Myerson, *Game Theory: Analysis of conflict*, 1991.

A. Mas-Colell, M. Whinston, and J. Green, Microeconomic Theory, 1995.

N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani (editors), *Algorithmic Game Theory*, 2007. *(Available for free online.)*

T. Roughgarden *Twenty Lectures on Algorithmic Game Theory*, Cambridge U. Press, 2016. (Available online from Library.)

Y. Shoham & K. Leyton-Brown, *Essentials of Game theory*, 2008. and *Multiagent systems: algorithmic, game-theoretic, and logical foundations*, 2009. (Both available from Library online.)

V. Chvátal, *Linear Programming*, 1980.

What is Game Theory?

A general and vague definition:

> *"Game Theory is the formal study of interaction between 'goal-oriented' 'agents' (or 'players'), and the strategic scenarios that arise in such settings."*

What is Game Theory?

A general and vague definition:

> *"Game Theory is the formal study of interaction between 'goal-oriented' 'agents' (or 'players'), and the strategic scenarios that arise in such settings."*

What is *Algorithmic Game Theory*?

> *"Concerned with the computational questions that arise in game theory, and that enlighten game theory. In particular, questions about finding efficient algorithms to 'solve' games."*

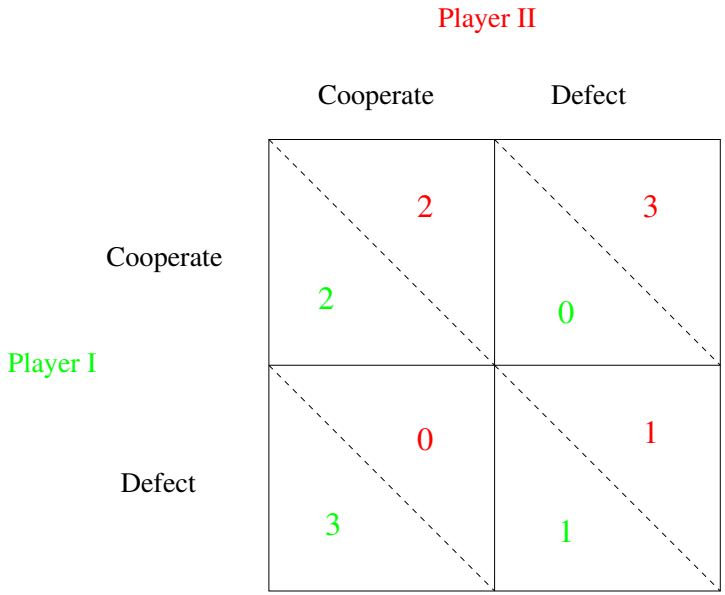
These vague sentences are best illustrated by looking at examples.

A simple 2-person game: Rock-Paper-Scissors

Player I \ Player II	Rock	Paper	Scissors
Rock	0, 0	1, −1	−1, 1
Paper	−1, 1	0, 0	1, −1
Scissors	1, −1	−1, 1	0, 0

- This is a "zero-sum" game: whatever Player I wins, Player II loses, and vice versa.
- What is an "optimal strategy" in this game?
- How do we compute such "optimal strategies" for 2-person zero-sum games?

A non-zero-sum 2-person game: Prisoner's Dilemma



- For both players Defection is a *"Dominant Strategy"* (regardless what the other player does, you're better off Defecting).
- But if they both Cooperate, they would both be better off.
- Game theorists/Economists worry about this kind of situation as a real problem for society.
- Often, there are no "dominant strategies". What does it mean to "solve" such games?

Nash Equilibria

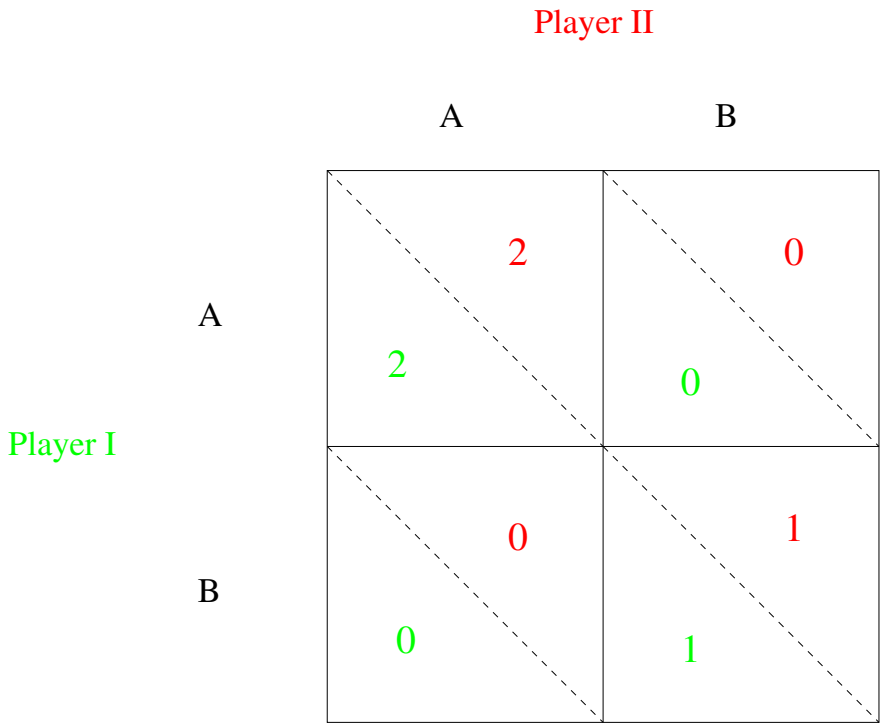
- A Nash Equilibrium (NE) is a pair (n-tuple) of strategies for the 2 players (n players) such that no player can benefit by unilaterally deviating from its strategy.

Nash Equilibria

- A Nash Equilibrium (NE) is a pair (n-tuple) of strategies for the 2 players (n players) such that no player can benefit by unilaterally deviating from its strategy.
- **Nash's Theorem:** Every (finite) game has a *mixed* (i.e., randomized) Nash equilibrium.
- **Example 1:** The pair of dominant strategies (Defect, Defect) is a pure NE in the Prisoner's Dilemma game. (In fact, it is the only NE.)
  In general, there may be many NE, none of which are pure.
- **Example 2:** In Rock-Paper-Scissors, the pair of *mixed* strategies: ((R=1/3, P=1/3,S=1/3), (R=1/3, P=1/3,S=1/3)) is a Nash Equilibrium. (And, we will learn, it is also a *minimax* solution to this zero-sum game. The *"minimax value"* is 0, as it must be because the game is *"symmetric"*.)
- **Question:** How do we compute a Nash Equilibrium for a given game?

Multiple equilibria

- Many games have > 1 NE. **Example:** A "Coordination Game":



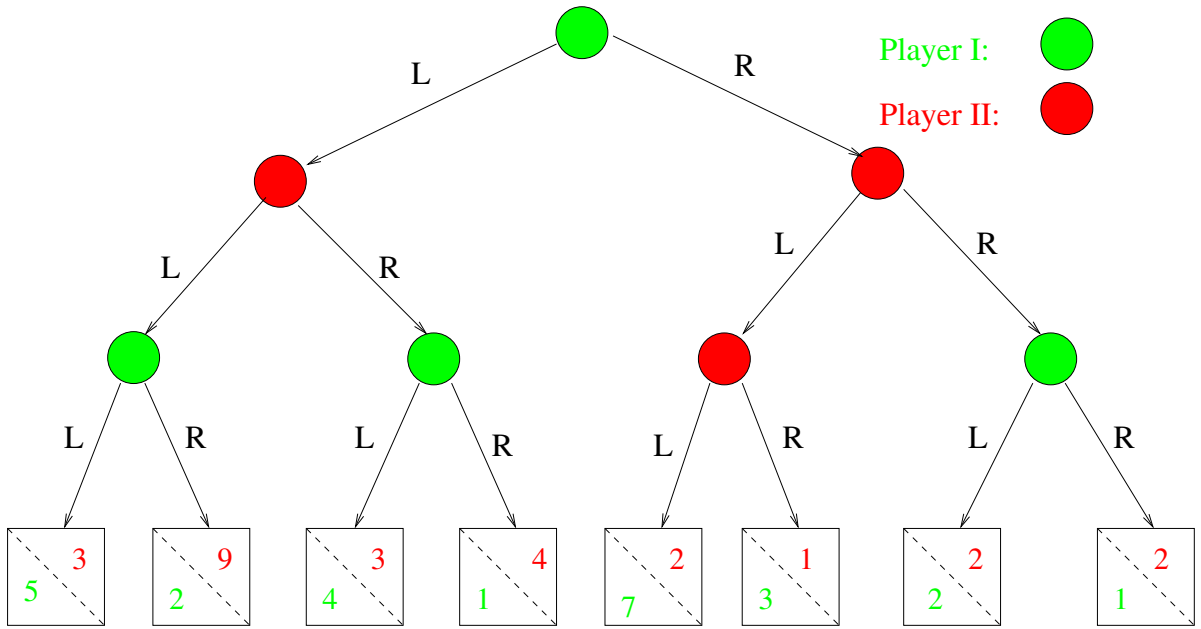
- There are two **pure** Nash Equilibria: (A, A) and (B, B). Are there any other NEs?
  Yes, there's one other *mixed* (randomized) NE.

Games in "Extensive Form"

So far, we have only seen games in "strategic form" (also called "normal form"), where all players choose their strategy simultaneously (independently).

What if, as is often the case, the game is played by a sequence of moves over time? (Think, e.g., Chess.)

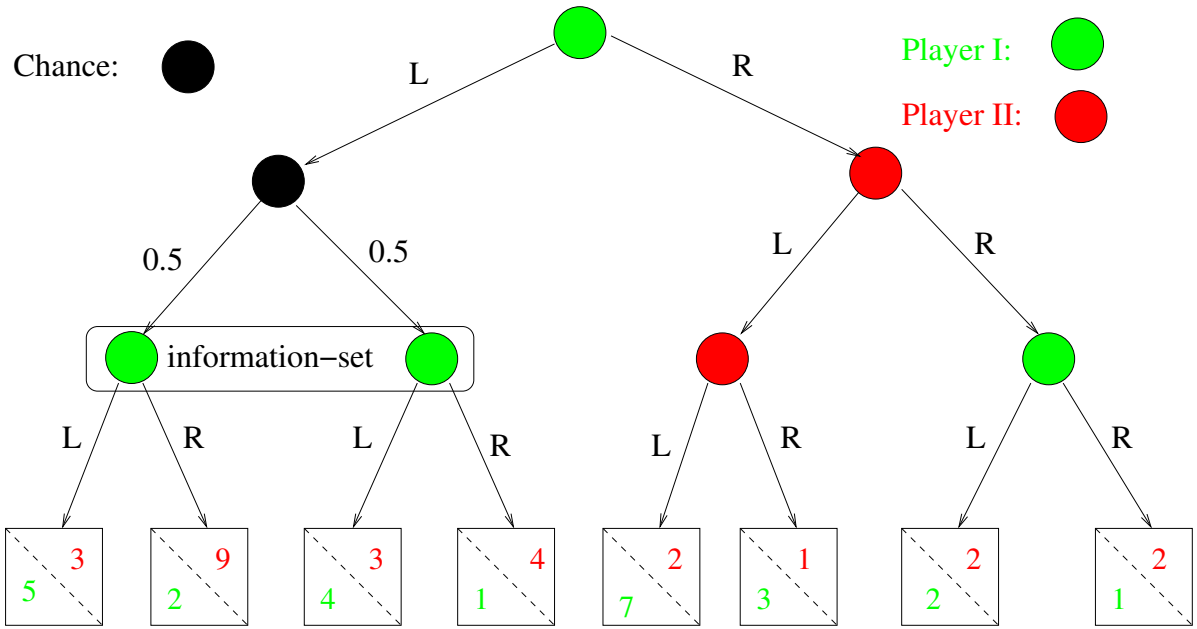
Consider the following 2-person game tree:



- How do we analyze and compute “solutions” to such *extensive form* games?
- What is their relationship to strategic form games?

chance, and information

Some tree nodes may be *chance* (probabilistic) nodes, controlled by neither player. (Poker, Backgammon.)
Also, a player may not be able to distinguish between several of its "positions" or "nodes", because not all *information* is available to it. (Think Poker, with opponent's cards hidden.) Whatever move a player employs at a node must be employed at all nodes in the same "information set".

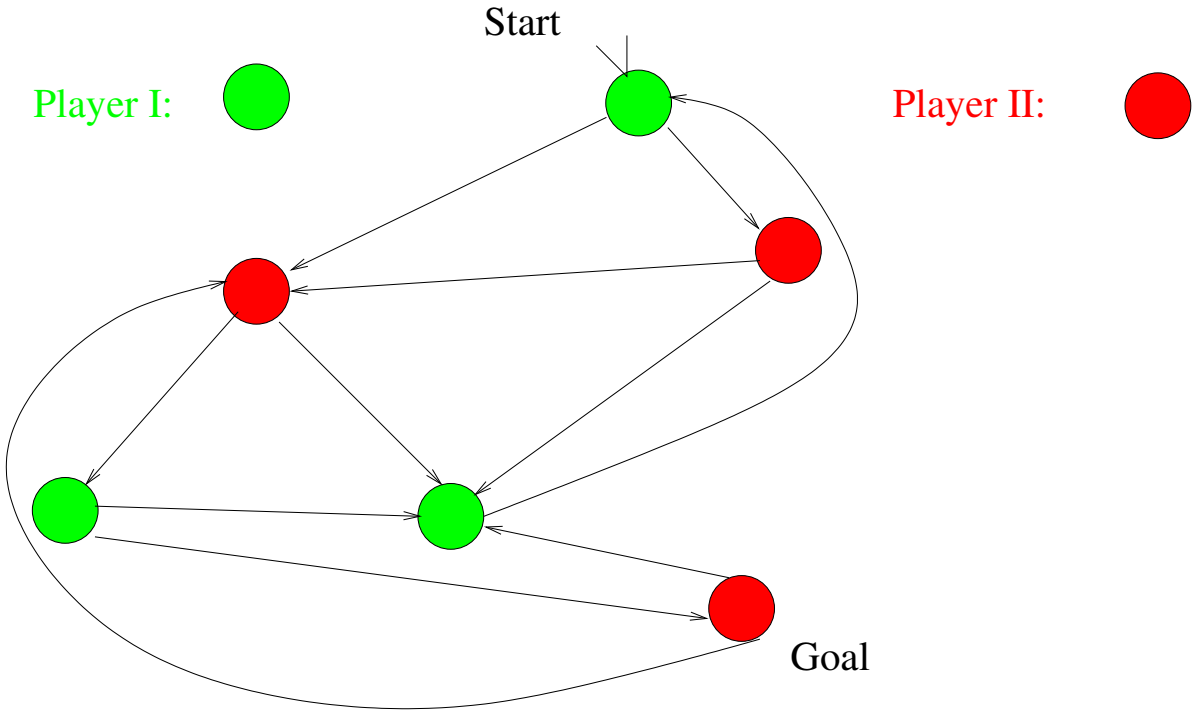


A game where every information set has only 1 node is called a game of *perfect information*.

Theorem Any finite n-person extensive game of perfect information has an "equilibrium in pure strategies".

Again, how do we compute equilibrium solutions for such games?

Extensive games on Graphs



- Does Player I have a strategy to “force” the play to reach the “Goal”?
- Such games have lots of applications.
- Again, how do we compute winning strategies in such games?
- What if some nodes are chance nodes?

Mechanism Design

Suppose you are the game designer. How would you design the game so that the "solutions" will satisfy some "objectives"?

- **Example:** Auctions: (think EBay, or Google Ads) Think of an auction as a multiplayer game between several bidders. If you are the auctioneer, how could you design the auction rules so that, for every bidder, bidding the maximum that an item is worth to them will be a "dominant strategy"?
  A answer: second price, sealed bid *Vickrey auctions*.
- How would you design protocols (such as network protocols), to encourage "cooperation" (e.g., diminish congestion)?
- Many computational questions arise in the study of "good" mechanisms for various goals.
- This is an extremely active area of research (we will only get to scratch its surface).

But why study this stuff?

GT is a core foundation of mathematical economics.
But what does it have to do with Computer Science? More than you might think: GT ideas have played an important role in CS:

- Games in AI: modeling "rational agents" and their interactions. (Similar to Econ. view.)
- Games in Modeling and analysis of reactive systems: computer-aided verification: formulations of model checking via games, program inputs viewed "adversarially", etc.
- Games in Algorithms: several GT problems have a very interesting algorithmic status (e.g., in NP, but not known to be NP-complete, etc).
- Games in Logic in CS: GT characterizations of logics, including modal and temporal logics (Ehrenfeucht-Fraisse games and bisimulation).

- Games in Computational Complexity: Many computational complexity classes are definable in terms of games: Alternation, Arthur-Merlin games, the Polynomial Hierarchy, etc.. Boolean circuits, a core model of computation, can be viewed as games (between AND and OR).

More recently:

- Games, the Internet, and E-commerce: An extremely active research area at the intersection of CS and Economics. Basic idea: "The internet is a HUGE experiment in interaction between agents (both human and automated)". How do we set up the rules of this game to harness "socially optimal" results?

I hope you are convinced: knowledge of the principles and algorithms of game theory will be useful to you for carrying on future work in many CS diciplines.

Ok, let's get started

Definition A *strategic form game* Γ, with n players, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. For each $i \in N$, a set S_i of *(pure) strategies*.
   Let $S = S_1 \times S_2 \times \ldots \times S_n$ be the set of possible combinations of (pure) strategies.
3. For each $i \in N$, a *payoff (utility) function*
   $u_i : S \mapsto \mathbb{R}$, describes the payoff $u_i(s_1, \ldots, s_n)$ to player i under each combination of strategies.

(Each player prefers to maximize its own payoff.)

Definition A zero-sum game Γ, is one in which for all $s = (s_1, \ldots, s_n) \in S$,

$$u_1(s) + u_2(s) + \ldots + u_n(s) = 0.$$

Some “food for thought”

Food for Thought (**the “guess half the average game”**): Consider a strategic-form game Γ with n-players. Each player has to guess a whole number from 1 to 1000. The player who guesses a number that is closest to half of the average guess of all players wins a payoff of 1. All other players get a payoff of 0. (If there are ties for who is closest, those who are closest share the payoff of 1 equally amongst themselves; alternatively, all who are closest get payoff 1.)

Question: What would your strategy be in such a game?

Question: What is a “Nash Equilibrium” of such a game?

Algorithmic Game Theory and Applications

Lecture 2: Mixed Strategies, Expected Payoffs, and Nash Equilibrium

Kousha Etessami

U. of Edinburgh

Finite Strategic Form Games

Recall the "strategic game" definition, now "finite":

Definition A finite *strategic form game* Γ, with n-players, consists of:

1. A set $N = \{1, \ldots, n\}$ of Players.
2. For each $i \in N$, a finite set $S_i = \{1, \ldots, m_i\}$ of *(pure) strategies*.
   Let $S = S_1 \times S_2 \times \ldots \times S_n$ be the set of possible combinations of (pure) strategies.
3. For each $i \in N$, a *payoff (utility) function*:
   $u_i : S \mapsto \mathbb{R}$, describes the payoff $u_i(s_1, \ldots, s_n)$ to player i under each combination of strategies.

(Each player wants to maximize its own payoff.)

Mixed (Randomized) Strategies

We define "mixed" strategies for general finite games.

Definition A **mixed** (i.e., **randomized**) **strategy** x_i for Player i, with $S_i = \{1, \ldots, m_i\}$, is a probability distribution over S_i. In other words, it is a vector $x_i = (x_i(1), \ldots, x_i(m_i))$, such that $x_i(j) \geq 0$ for $1 \leq j \leq m_i$, and

$$x_i(1) + x_i(2) + \ldots + x_i(m_i) = 1$$

Intuition: Player i uses randomness to decide which strategy to play, based on the probabilities in x_i.
Let X_i be the set of mixed strategies for Player i.
For an n-player game, let

$$X = X_1 \times \ldots \times X_n$$

denote the set of all possible combinations, or "**profiles**", of mixed strategies.

Expected Payoffs

Let $x = (x_1, \ldots, x_n) \in X$ be a profile of mixed strategies.
For $s = (s_1, \ldots, s_n) \in S$ a combination of pure strategies, let

$$x(s) := \prod_{j=1}^{n} x_j(s_j)$$

be the probability of combination s under mixed profile x. (We're assuming players make their random choices independently.)
Definition: The **expected payoff** of Player i under a mixed strategy profile $x = (x_1, \ldots, x_n) \in X$, is:

$$U_i(x) := \sum_{s \in S} x(s) * u_i(s)$$

I.e., the "weighted average" Player i's payoff under each pure combination s, weighted by the probability of that combination.
Key Assumption: Every player's goal is to maximize its own expected payoff. (This can somtimes be a dubious assumption.)

some notation

We call a mixed strategy $x_i \in X_i$ *pure* if $x_i(j) = 1$ for some $j \in S_i$, and $x_i(j') = 0$ for $j' \neq j$. We denote such a pure strategy by $\pi_{i,j}$. I.e., the "mixed" strategy $\pi_{i,j}$ does not randomize at all: it picks (with probability 1) exactly one strategy, j, from the set of pure strategies for player i.

Given a profile of mixed strategies $x = (x_1, \ldots, x_n) \in X$, let

$$x_{-i} = (x_1, x_2, \ldots, x_{i-1}, \texttt{empty}, x_{i+1}, \ldots, x_n)$$

I.e., x_{-i} denotes everybody's strategy except that of player i.

For a mixed strategy $y_i \in X_i$, let $(x_{-i}; y_i)$ denote the new profile:

$$(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$$

In other words, $(x_{-i}; y_i)$ is the new profile where everybody's stategy remains the same as in x, except for player i, who switches from mixed strategy x_i, to mixed strategy y_i.

Best Responses

Definition: A (mixed) strategy $z_i \in X_i$ is a **best response** for Player i to x_{-i} if for all $y_i \in X_i$,

$$U_i(x_{-i}; z_i) \geq U_i(x_{-i}; y_i)$$

Clearly, if any player were given the opportunity to "cheat" and look at what other players have done, it would want to switch its strategy to a best response.
Of course, players in a strategic form game can't do that: players pick their strategies simultaneously/independently.

But suppose, somehow, the players "arrive" at a profile where everybody's strategy is a best response to everybody else's.
Then no one has any incentive to change the situation.
We will be in a "stable" situation: an "*Equilibrium*".
That's what a "Nash Equilibrium" is.

Nash Equilibrium

Definition: For a strategic game Γ, a strategy profile $x = (x_1, \ldots, x_n) \in X$ is a mixed **Nash Equilibrium** if for every player, i , x_i is a best response to x_{-i}.
In other words, for every Player $i = 1, \ldots, n$, and for every mixed strategy $y_i \in X_i$,

$$U_i(x_{-i}; x_i) \geq U_i(x_{-i}; y_i)$$

In other words, *no player can improve its own payoff by unilaterally deviating from the mixed strategy profile* $x = (x_1, \ldots, x_n)$.

x is called a **pure Nash Equilibrium** if in addition every x_i is a pure strategy $\pi_{i,j}$, for some $j \in S_i$.

Nash's Theorem

This can, agruably, be called
"The Fundamental Theorem of Game Theory"

Theorem(Nash 1950) Every finite *n*-person strategic game has a mixed Nash Equilibrium.

We will prove this theorem next time.
To prove it, we will "cheat" and use a fundamental result from topology: the Brouwer Fixed Point Theorem.

The crumpled sheet experiment

Let's all please conduct the following experiment:

1. Take two identical rectangular sheets of paper.
2. Make sure neither sheet has any holes in it, and that the sides are straight (not dimpled).
3. "Name" each point on both sheets by its "(x, y)-coordinates".
4. Crumple one of the two sheets any way you like, *but make sure you don't rip it in the process*.
5. Place the crumpled sheet completely on top of the other flat sheet.

Fact! There must be a point named (a, b) on the crumpled sheet that is directly above the same point (a, b) on the flat sheet. (Yes, really!)

As crazy as it sounds, this fact, in its more formal and general form, will be the key to why every game has a mixed Nash Equilibrium.

Algorithmic Game Theory and Applications

Lecture 3: Nash's Theorem

Kousha Etessami

U. of Edinburgh

The Brouwer Fixed Point Theorem

We will use the following to prove Nash's Theorem.

Theorem(Brouwer, 1909) Every continuous function $f : D \to D$ mapping a compact and convex, nonempty subset $D \subseteq \mathbb{R}^m$ to itself has a "fixed point", i.e., there is $x^* \in D$ such that $f(x^*) = x^*$.

Explanation:

- A "continuous" function is intuitively one whose graph has no "jumps".
- For our current purposes, we don't need to know exactly what "compact and convex" means.

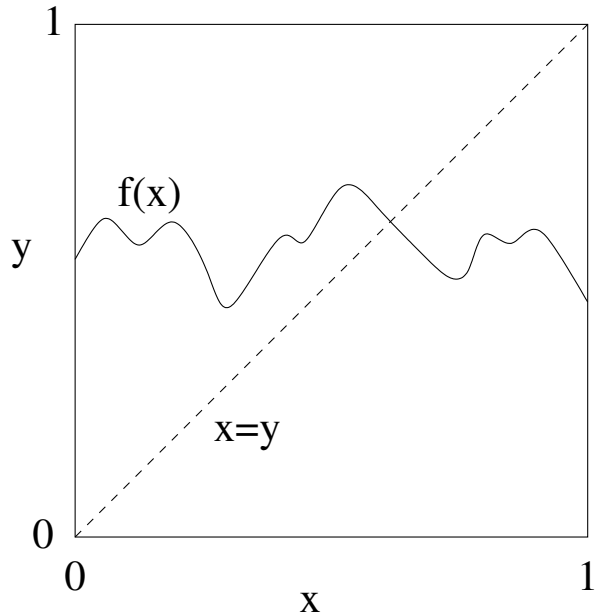
(See the appendix of this lecture for definitions.)

We only state the following fact:

Fact The set of profiles $X = X_1 \times \ldots \times X_n$ is a compact and convex subset of R^m, where $m = \Sigma_{i=1}^{n} m_i$, with $m_i = |S_i|$.

Simple cases of Brouwer's Theorem

To see a simple example of what Brouwer's theorem says, consider the interval $[0,1] = \{x \mid 0 \leq x \leq 1\}$.
$[0,1]$ is compact and convex. ($[0,1]^n$ is also compact & convex.)
For a continuous $f : [0,1] \to [0,1]$, you can "visualize" why the theorem is true. Here's the "visual proof" in the 1-dimensional case:



For $f : [0,1]^2 \to [0,1]^2$, the theorem is already far less obvious: "the crumpled sheet experiment".

brief remarks

- Brouwer's Theorem is a deep and important result in topology.
- It is not very easy to prove, and we won't prove it.
- If you are desperate to see a proof, there are many. See, e.g., any of these:
  - [Milnor'66] (Differential Topology). (uses, e.g., Sard's Theorem).
  - [Scarf'67 & '73, Kuhn'68, Border'89], uses **Sperner's Lemma**.
  - [Rotman'88] (Algebraic Topology). (uses homology, etc.)
  - [D. Gale'79], possibly my favorite proof: uses the fact that the game of (n-dimensional) HEX is a finite "win-lose" game.

proof of Nash's theorem

Proof: (Nash's 1951 proof)
We will define a continuous function $f : X \to X$, where $X = X_1 \times \ldots \times X_n$, and we will show that if $f(x^*) = x^*$ then $x^* = (x_1^*, \ldots, x_n^*)$ must be a Nash Equilibrium.

By Brouwer's Theorem, we will be done.

(In fact, it will turn out that x^* is a Nash Equilibrium if and only if $f(x^*) = x^*$.) fix point = nash equilibrium

We start with a claim.

Claim: A profile $x^* = (x_1^*, \ldots, x_n^*) \in X$ is a Nash Equilibrium if and only if, for every player i, and every pure strategy $\pi_{i,j}$, $j \in S_i$:

$$U_i(x^*) \geq U_i(x_{-i}^*; \pi_{i,j}).$$

Proof of claim: If x^* is a NE then, it is obvious by definition that $U_i(x^*) \geq U_i(x_{-i}^*, \pi_{i,j})$.
For the other direction: by calculation it is easy to see that for any mixed strategy $x_i \in X_i$,

$$U_i(x_{-i}^*; x_i) = \sum_{j=1}^{m_i} x_i(j) * U_i(x_{-i}^*; \pi_{i,j})$$

By assumption, $U_i(x^*) \geq U_i(x_{-i}^*; \pi_{i,j})$, for all j.
So, clearly $U_i(x^*) \geq U_i(x_{-i}^*; x_i)$, for any $x_i \in X_i$, because $U_i(x_{-i}^*; x_i) = \sum_{j=1}^{m_i} x_i(j) * U_i(x_{-i}^*; \pi_{i,j}) \leq \sum_{j=1}^{m_i} x_i(j) * U_i(x^*) = U_i(x^*)$.

Hence, each x_i^* is a best response strategy to x_{-i}^*. In other words, x^* is a Nash Equilibrium.

So, rephrasing our goal, we want to find $x^* = (x_1^*, \ldots, x_n^*)$ such that

$$U_i(x_{-i}^*; \pi_{i,j}) \leq U_i(x^*)$$

i.e., such that

$$U_i(x_{-i}^*; \pi_{i,j}) - U_i(x^*) \leq 0$$

for all players $i \in N$, and all $j = 1, \ldots, m_i$.
For a mixed profile $x = (x_1, x_2, \ldots, x_n) \in X$: let

$$\varphi_{i,j}(x) = \max\{0, U_i(x_{-i}; \pi_{i,j}) - U_i(x)\}$$

Intuitively, $\varphi_{i,j}(x)$ measures "how much better off" player i would be if he/she picked $\pi_{i,j}$ instead of x_i (and everyone else remained unchanged).

Define $f : X \to X$ as follows: For $x = (x_1, x_2, \ldots, x_n) \in X$, let

$$f(x) = (x_1', x_2', \ldots, x_n')$$

where for all i, and $j = 1, \ldots, m_i$,

$$x_i'(j) = \frac{x_i(j) + \varphi_{i,j}(x)}{1 + \sum_{k=1}^{m_i} \varphi_{i,k}(x)}$$

numerator
denominator

Facts:

1. If $x \in X$, then $f(x) = (x_1', \ldots, x_n') \in X$.
2. $f : X \to X$ is continuous.

(These facts are not hard to check.)

Thus, by Brouwer, there exists $x^* = (x_1^*, x_2^*, \ldots, x_n^*) \in X$ such that $f(x^*) = x^*$.

Now we have to show x^* is a NE.

For each i, and for $j = 1, \ldots, m_i$,

$$x_i^*(j) = \frac{x_i^*(j) + \varphi_{i,j}(x^*)}{1 + \sum_{k=1}^{m_i} \varphi_{i,k}(x^*)}$$

thus,

$$x_i^*(j)(1 + \sum_{k=1}^{m_i} \varphi_{i,k}(x^*)) = x_i^*(j) + \varphi_{i,j}(x^*)$$

hence,

$$x_i^*(j) \sum_{k=1}^{m_1} \varphi_{i,k}(x^*) = \varphi_{i,j}(x^*)$$

We will show that in fact this implies $\varphi_{i,j}(x^*)$ must be equal to 0 for all j.

Claim: For any mixed profile x, for each player i, there is some j such that $x_i(j) > 0$ and $\varphi_{i,j}(x) = 0$.
Proof of claim: For any $x \in X$,

$$\varphi_{i,j}(x) = \max\{0, U_i(x_{-i}; \pi_{i,j}) - U_i(x)\}$$

Since $U_i(x)$ is the "weighted average" of $U_i(x_{-i}; \pi_{i,j})$'s, based on the "weights" in x_i, there must be some j used in x_i, i.e., with $x_i(j) > 0$, such that $U_i(x_{-i}; \pi_{i,j})$ is no more than the weighted average. I.e.,

$$U_i(x_{-i}; \pi_{i,j}) \leq U_i(x)$$

I.e.,

$$U_i(x_{-i}; \pi_{i,j}) - U_i(x) \leq 0$$

Therefore,

$$\varphi_{i,j}(x) = \max\{0, U_i(x_{-i}; \pi_{i,j}) - U_i(x)\} = 0$$

Thus, for such a j, $x_i^*(j) > 0$ and

$$x_i^*(j) \sum_{k=1}^{m_1} \varphi_{i,k}(x^*) = 0 = \varphi_{i,j}(x^*)$$

But, since $\varphi_{i,k}(x^*)$'s are all ≥ 0, this means $\varphi_{i,k}(x^*) = 0$ for all $k = 1, \ldots, m_i$. Thus, for all players i, and for $j = 1, \ldots, m_i$,

$$U_i(x^*) \geq U_i(x_{-i}^*; \pi_{i,j})$$

Q.E.D. (Nash's Theorem)

In fact, since $U_i(x^*) = \sum_{j=1}^{m_i} x_i^*(j) \cdot U_i(x_{-i}^*; \pi_{i,j})$ is the "weighted average" of $U_i(x_{-i}^*, \pi_{i,j})$'s, we see that:

Useful Corollary for Nash Equilibria:

$U_i(x^*) = U_i(x_{-i}^*, \pi_{i,j})$, whenever $x_i^*(j) > 0$.

Rephrased: In a Nash Equilibrium x^*, if $x_i^*(j) > 0$ then $U_i(x_{-i}^*; \pi_{i,j}) = U_i(x^*)$; i.e., each such $\pi_{i,j}$ is itself a "best response" to x_{-i}^*.

This is a subtle but very important point. It will be useful later when we want to compute NE's.

Remarks

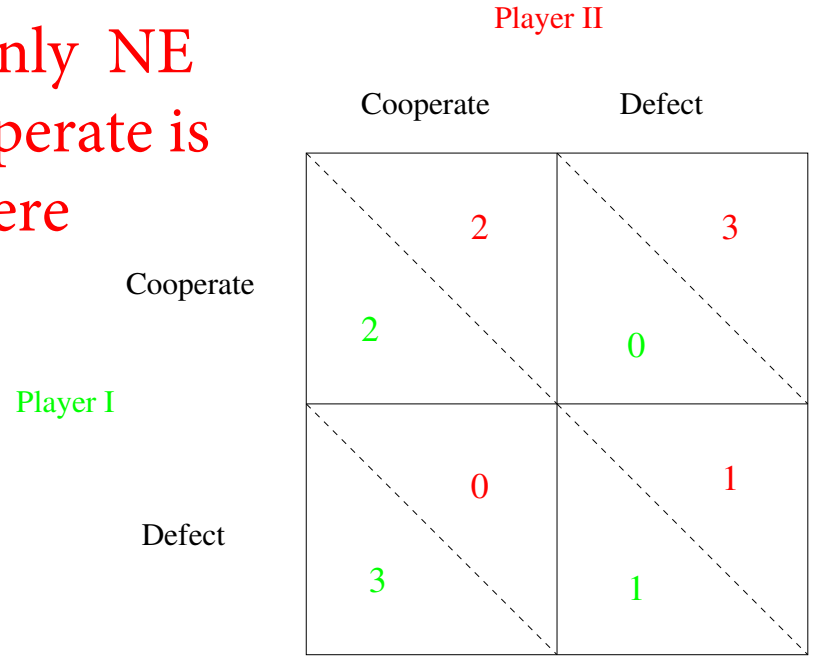
- The proof using Brouwer gives ostensibly no clue how to compute a Nash Equilibrium. It just says it exists!
- We will come back to the question of computing Nash Equilibria in general games later in the course.
- We start next time with a special case: 2-player zero-sum games (e.g., of the Rock-Paper-Scissor's variety). These have an elegant theory (von Neumann 1928), predating Nash.
- To compute solutions for 2p-zero-sum games, Linear Programming will come into play.
  Linear Programming is a very important tool in algorithms and optimization. Its uses go FAR beyond solving zero-sum games. So it will be a good opportunity to learn about LP.

NE need not be "Pareto optimal"

Given a profile $x \in X$ in an n-player game, the **"(purely utilitarian) social welfare"** is: $U_1(x) + U_2(x) + \ldots + U_n(x)$.
A profile $x \in X$ is **pareto efficient** (a.k.a., **pareto optimal**) if there is no other profile x' such that $U_i(x) \leq U_i(x')$ for all players i, and $U_k(x) < U_k(x')$ for some player k.

Note: The Prisoner's Dilemma game shows NE need not optimize social welfare, nor be Pareto optimal.

defect-defect is the only NE here, cooperate- cooperate is the pareto optimal here



Indeed, there is a unique NE, (Defect, Defect), and it neither optimizes social welfare nor is Pareto optimal, because (Cooperate, Cooperate) gives a higher payoff to both players.

application in biology: evolution as a game

- One way to view how we might "arrive" at a Nash equilibrium is through a process of evolution.
- John Maynard Smith (1972-3,'82) introduced game theoretic ideas into evolutionary biology with the concept of an Evolutionarily Stable Strategy.
- Your extra reading (for fun) is from Straffin(1993) which gives an amusing introduction to this.
- Intuitively, a mixed strategy can be viewed as percentages in a population that exhibit different behaviors (strategies).
- Their behaviors effect each other's survival, and thus each strategy has a certain survival value dependent on the strategy of others.
- The population is in "evolutionary equilibrium" if no "mutant" strategy could invade it and "take over".

The Hawk-Dove Game

Player I \ Player II	Hawk	Dove
Hawk	−15, −15	0, 50
Dove	50, 0	25, 25

Large population of same "species", each behaving as either "hawk" or "dove".
What proportions will behaviors eventually stabilize to (if at all)?

Definition of ESS

Definition: A 2-player game is **symmetric** if $S_1 = S_2$, and for all $s_1, s_2 \in S_1$, $u_1(s_1, s_2) = u_2(s_2, s_1)$.

Definition: In a 2p-sym-game, mixed strategy x_1^* is an **Evolutionarily Stable Strategy** (ESS), if:

1. x_1^* is a best response to itself, i.e., $x^* = (x_1^*, x_1^*)$ is a symmetric Nash Equilibrium, &
2. If $x_1' \neq x_1^*$ is another best response to x_1^*, then $U_1(x_1', x_1') < U_1(x_1^*, x_1')$.

Nash (1951, p. 289) also proves that every symmetric game has a symmetric NE, (x_1^*, x_1^*). (However, not every symmetric game has a ESS.)

A little justification of the definition of ESS

Suppose x_1^* is an ESS. Consider the "*fitness function*", $F(x_1)$, for a "mutant" strategy x_1' that "invades" (becoming a small $\epsilon > 0$ fraction of) a current ESS population, x_1^*. Then, **Claim**:

$$F(x_1') \doteq (1-\epsilon)U_1(x_1', x_1^*) + \epsilon U_1(x_1', x_1') \quad (1)$$

$$< (1-\epsilon)U_1(x_1^*, x_1^*) + \epsilon U_1(x_1^*, x_1') \doteq F(x^*) \quad (2)$$

1. is best response
2. is not best response

Proof: if x_1' is a best response to the ESS x_1^*, then $U_1(x_1', x_1^*) = U_1(x_1^*, x_1^*)$ and $U_1(x_1', x_1') < U_1(x_1^*, x_1')$, and since we assume $\epsilon > 0$, the strict inequality in (2) follows. If on the other hand x_1' is *not* a best response to x_1^*, then $U_1(x_1', x_1^*) < U_1(x_1^*, x_1^*)$, and for a *small enough* $\epsilon > 0$, we have $(1-\epsilon)(U_1(x_1^*, x_1^*) - U_1(x_1', x_1^*)) > \epsilon(U_1(x_1^*, x_1') - U_1(x_1', x_1'))$. Thus again, the strict inequality in (2) follows. □

So, an ESS x_1^* is "strictly fitter" than any other strategy, when it is already dominant in the society. This is the sense in which it is "evolutionarily stable".

Does an ESS necessarily exist?

- As mentioned, Nash (1951) already proved that every symmetric game has a symmetric NE (x^*, x^*).
- However, not every symmetric game has a ESS.
  Example: Rock-paper-scissors:

$$\begin{pmatrix} (0,0) & (1,-1) & (-1,1) \\ (-1,1) & (0,0) & (1,-1) \\ (1,-1) & (-1,1) & (0,0) \end{pmatrix}$$

  Obviously, $s = (1/3, 1/3, 1/3)$ is the only NE. But it is not an ESS: any strategy is a best reponse to s, including the pure strategy s^1 (rock). We have payoff $U(s^1, s^1) = 0 = U(s, s^1)$, so s is not an ESS.
- But many games do have an ESS. For example, in the Hawk-Dove game, $(5/8, 3/8)$ is an ESS.
- Even when a game does have an ESS, it is not at all obvious how to find one.

How hard is it to detect an ESS?

- It turns out that even deciding whether a 2-player symmetric game has an ESS is hard. It is both NP-hard and coNP-hard, and contained in Σ_2^P:
  K. Etessami & A. Lochbihler, "The computational complexity of Evolutionarily Stable Strategies", *International Journal of Game Theory*, vol. 31(1), pp. 93–113, 2008.
  (And, more recently, it has been shown Σ_2^P-complete, see: V. Conitzer, *"The exact computational complexity of Evolutionary Stable Strategies"*, in *Proceeding of Web and Internet Economics* (WINE), pages 96-108, 2013.)
- For simple 2×2 2-player symmetric games, there is a simple way to detect whether there is an ESS, and if so to compute one (described in the reading from Straffin).
- There is a huge literature on ESS and *"Evolutionary Game Theory"*. See, e.g., the book: J. Weibull, *Evolutionary Game Theory*, 1997.

Appendix: continuity, compactness, convexity

Definition For $x, y \in \mathbb{R}^n$, $\texttt{dist}(x, y) = \sqrt{\sum_{i=1}^{n}(x(i) - y(i))^2}$ denotes the Euclidean distance between points x and y.

A function $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is **continuous at a point** $x \in D$ if for all $\epsilon > 0$, there exists $\delta > 0$, such that for all $y \in D$: if $\texttt{dist}(x, y) < \delta$ then $\texttt{dist}(f(x), f(y)) < \epsilon$.

f is called **continuous** if it is continuous at every point $x \in D$.

Definition A set $K \subseteq \mathbb{R}^n$ is **convex** if for all $x, y \in K$ and all $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda)y \in K$.

Fact A set $K \subseteq \mathbb{R}^n$ is **compact** if and only if it is **closed** and **bounded**. (So, we need to define "closed" and "bounded".)

Definition A set $K \subseteq \mathbb{R}^n$ is **bounded** iff there is some non-negative integer M, such that $K \subseteq [-M, M]^n$.
(i.e., K "fits inside" a finite n-dimensional box.)

Definition A set $K \subseteq \mathbb{R}^n$ is **closed** iff for all sequences $x_0, x_1, x_2, \ldots$, where $x_i \in K$ for all i, such that $x = \lim_{i \to \infty} x_i$ for some $x \in \mathbb{R}^n$, then $x \in K$. (In other words, if a sequence of points is in K then its limit (if it exists) must also be in K.)

Algorithmic Game Theory and Applications

Lecture 4: 2-player zero-sum games, and the Minimax Theorem

Kousha Etessami

2-person zero-sum games

A finite 2-person zero-sum (2p-zs) strategic game Γ, is a strategic game where:

- For players $i \in \{1, 2\}$, the *payoff functions* $u_i : S \mapsto \mathbb{R}$ are such that for all $s = (s_1, s_2) \in S$,

$$u_1(s) + u_2(s) = 0$$

I.e., $\quad u_1(s) = -u_2(s)$.

$u_i(s_1, s_2)$ can conveniently be viewed as a $m_1 \times m_2$ payoff matrix A_i, where:

$$A_1 = \begin{bmatrix} u_1(1,1) & \ldots\ldots & u_1(1,m_2) \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ u_1(m_1,1) & \ldots\ldots & u_1(m_1,m_2) \end{bmatrix}$$

Note, $A_2 = -A_1$. Thus we may assume only one function $u(s_1, s_2)$ is given, as one matrix, $A = A_1$.

2-player zero-sum game matrix

Thus, a 2-player zero-sum game can be described by a single $m_1 \times m_2$ matrix:

$$A = \begin{bmatrix} a_{1,1} & \cdots\cdots & a_{1,m_2} \\ \vdots & \vdots & \vdots \\ \vdots & a_{i,j} & \vdots \\ \vdots & \vdots & \vdots \\ a_{m_1,1} & \cdots\cdots & a_{m_1,m_2} \end{bmatrix}$$

where $a_{i,j} = u_1(i,j)$.

Player 1 (the row player) wants to maximize $u(i,j)$, whereas Player 2 (the column player) wants to minimize it (i.e., to maximize its negative).

review of matrix and vector notations

For any $(n_1 \times n_2)$-matrix A we'll either use $a_{i,j}$ or $(A)_{i,j}$ to denote the entry in the i'th row and j'th column of A.

For $(n_1 \times n_2)$ matrices A and B, let

$$A \geq B$$

denotes that for all i, j, $a_{i,j} \geq b_{i,j}$.

Let

$$A > B$$

denotes that for all i, j, $a_{i,j} > b_{i,j}$.

For a matrix A, let $A \geq 0$ denote that every entry is ≥ 0. Likewise, let $A > 0$ mean every entry is > 0.

more review of matrices and vectors

Recall matrix multiplication: given $(n_1 \times n_2)$-matrix A and $(n_2 \times n_3)$-matrix B, the product AB is an $(n_1 \times n_3)$-matrix C, where

$$c_{i,j} = \sum_{k=1}^{n_2} a_{i,k} * b_{k,j}$$

Fact: matrix multiplication is "associative": i.e.,

$$(AB)C = A(BC)$$

(Note: for the multiplications to be defined, the dimensions of the matrices A, B, and C need to be "consistent": $(n_1 \times n_2)$, $(n_2 \times n_3)$, and $(n_3 \times n_4)$, respectively.)

Fact: For matrices A, B, C, of appropriate dimensions, if $A \geq B$, and $C \geq 0$, then

$$AC \geq BC, \text{ and likewise, } CA \geq CB.$$

more review of matrix and vector notation

For a $(n_1 \times n_2)$ matrix B, let B^T denote the $(n_2 \times n_1)$ **transpose** matrix, where $(B^T)_{i,j} := (B)_{j,i}$.

We can view a column vector, $y = \begin{bmatrix} y(1) \\ \vdots \\ \vdots \\ y(m) \end{bmatrix}$, as a $(m \times 1)$-matrix. Then, y^T would be a $(1 \times m)$-matrix, i.e., a row vector.

Typically, we think of "vectors" as column vectors. We'll call a length m vector an m-vector.

Multiplying a $(n_1 \times n_2)$-matrix A by a n_2-vector y is just a special case of matrix multiplication:Ay is a n_1-vector.

Likewise, $y^T A$ is a n_2-row vector.

For a column (row) vector y, we use $(y)_j$ to denote its i'th entry.

A matrix view of zero-sum games

Suppose we have a 2p-zs game given by a $(m_1 \times m_2)$-matrix, A.

Suppose Player 1 chooses a mixed strategy x_1, and Player 2 chooses mixed strategy x_2 (assume x_1 and x_2 are given by column vectors).

$$x_1^T A x_2 = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} (x_1(i) * x_2(j)) * a_{i,j}$$

But note that $(x_1(i) * x_2(j))$ is precisely the probability of the pure combination $s = (i, j)$. Thus, for the mixed profile $x = (x_1, x_2)$

$$x_1^T A x_2 = U_1(x) = -U_2(x)$$

where $U_1(x)$ is the expected payoff (which Player 1 is trying to maximize, and Player 2 is trying to minimize).

"minmaximizing" strategies

Suppose Player 1 chooses a mixed strategy $x_1^* \in X_1$, by trying to maximize the "worst that can happen". The worst that can happen would be for Player 2 to choose x_2 which minimizes $(x_1^*)^T A x_2$.

Definition: $x_1^* \in X_1$ is a **minmaximizer** for Player 1 if

$$\min_{x_2 \in X_2} (x_1^*)^T A x_2 = \max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2$$

Similarly, $x_2^* \in X_2$ is a **maxminimizer** for Player 2 if

$$\max_{x_1 \in X_1} (x_1)^T A x_2^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2$$

Note that

$$\min_{x_2 \in X_2} (x_1^*)^T A x_2 \leq (x_1^*)^T A x_2^* \leq \max_{x_1 \in X_1} x_1^T A x_2^*$$

Amazingly, von Neumann (1928) showed equality holds!

The Minimax Theorem

Theorem(von Neumann) Let a 2p-zs game Γ be given by an $(m_1 \times m_2)$-matrix A of real numbers. There exists a unique value $v^* \in \mathbb{R}$, such that there exists $x^* = (x_1^*, x_2^*) \in X$ such that

1. $((x_1^*)^T A)_j \geq v^*$, for $j = 1, \ldots, m_2$.
2. $(Ax_2^*)_j \leq v^*$, for $j = 1, \ldots, m_1$.
3. And (thus) $v^* = (x_1^*)^T A x_2^*$ and

$$\max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2 = v^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2$$

4. In fact, the above conditions all hold precisely when $x^* = (x_1^*, x_2^*)$ is any Nash Equilibrium.
   Equivalently, they hold precisely when x_1^* is any minmaximizer and x_2^* is any maxminimizer.

some remarks

Note:
(1.) says x_1^* guarantees Player 1 at least expected profit v^*, and

(2.) says x_2^* guarantees Player 2 at most expected "loss" v^*.

We call any such $x^* = (x_1^*, x_2^*)$ a **minimax profile.**

We call the unique v^* the **minimax value** of game Γ.

It is obvious that the maximum profit that Player 1 can guarantee for itself should be $\leq$ the minimum loss that Player 2 can guarantee for itself, i.e., that

$$\max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2 \leq \min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2$$

What is not obvious at all is why these two values should be the same!

Proof of the Minimax Theorem

The Minimax Theorem follows directly from Nash's Theorem (but historically, it predates Nash).

Proof: Let $x^* = (x_1^*, x_2^*) \in X$ be a NE of the 2-player zero-sum game Γ, with matrix A.

Let $v^* := (x_1^*)^T A x_2^* = U_1(x^*) = -U_2(x^*)$.

Since x_1^* and x_2^* are "best responses" to each other, we know that for $i \in \{1, 2\}$ $U_i(x_{-i}^*; \pi_{i,j}) \leq U_i(x^*)$.

But

1. $U_1(x_{-1}^*; \pi_{1,j}) = (Ax_2^*)_j$. Thus,
$$(Ax_2^*)_j \leq v^* = U_1(x^*)$$
for all $j = 1, \ldots, m_1$.
2. $U_2(x_{-2}^*; \pi_{2,j}) = -((x_1^*)^T A)_j$. Thus,
$$((x_1^*)^T A)_j \geq v^* = -U_2(x^*)$$
for all $j = 1, \ldots, m_2$.

3. $\max_{x_1 \in X_1} (x_1)^T A x_2^* \leq v^*$ because $(x_1)^T A x_2^*$ is a "weighted average" of $(Ax_2^*)_j$'s.
   Similarly, $v^* \leq \min_{x_2 \in X_2} (x_1^*)^T A x_2$ because $(x_1^*)^T A x_2$ is a "weighted average" of $((x_1^*)^T A)_j$'s. Thus

$$\max_{x_1 \in X_1} (x_1)^T A x_2^* \leq v^* \leq \min_{x_2 \in X_2} (x_1^*)^T A x_2$$

   We earlier noted the opposite inequalities, so,

$$\min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2 = v^* = \max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2$$

4. We didn't assume anything about the particular Nash Equilibrium we chose. So, for every NE, x^*, letting $v' = (x_1^*)^T A x_2^*$,

$$\max_{x_1 \in X_1} \min_{x_2 \in X_2} (x_1)^T A x_2 = v' = v^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} x_1^T A x_2$$

   Moreover, if $x^* = (x_1^*, x_2^*)$ satisfies conditions (1.) and (2.) for some v^*, then x^* must be a Nash Equilibrium.

Q.E.D. (Minimax Theorem)

remarks and food for thought

- Thus, for 2-player zero-sum games, Nash Equilibria and Minimax profiles are the same thing.
- Let us note here
  Useful Corollary for Minimax: In a minimax profile $x^* = (x_1^*, x_2^*)$,
  1. if $x_2^*(j) > 0$ then $((x_1^*)^T A)_j = (x_1^*)^T A x_2^* = v^*$.
  2. if $x_1^*(j) > 0$ then $(Ax_2^*)_j = (x_1^*)^T A x_2^* = v^*$.

  This is an immediate consequence of the Useful Corollary for Nash Equilibria.
- If you were playing a 2-player zero-sum game (say, as player 1) would you always play a minmaximizer strategy?
- What if you were convinced your opponent is an idiot?
- Notice, we have no clue yet how to compute the minimax value and a minimax profile.
  That is about to change.

minimax as an optimization problem

Consider the following "optimization problem":

Maximize v
Subject to constraints:
$(x_1^T A)_j \geq v$ *for* $j = 1, \ldots, m_2$,
$x_1(1) + \ldots + x_1(m_1) = 1$,
$x_1(j) \geq 0$ *for* $j = 1, \ldots, m_1$

It follows from the minimax theorem that an optimal solution (x_1^*, v^*) would give precisely the minimax value v^*, and a minmaximizer x_1^* for Player 1.
We are optimizing a "linear objective",
under "linear constraints" (or "linear inequalities").
That's what Linear Programming is.
Fortunately, we have good algorithms for it.
Next time, we start Linear Programming.

Algorithmic Game Theory and Applications

Lecture 5: Introduction to Linear Programming

Kousha Etessami

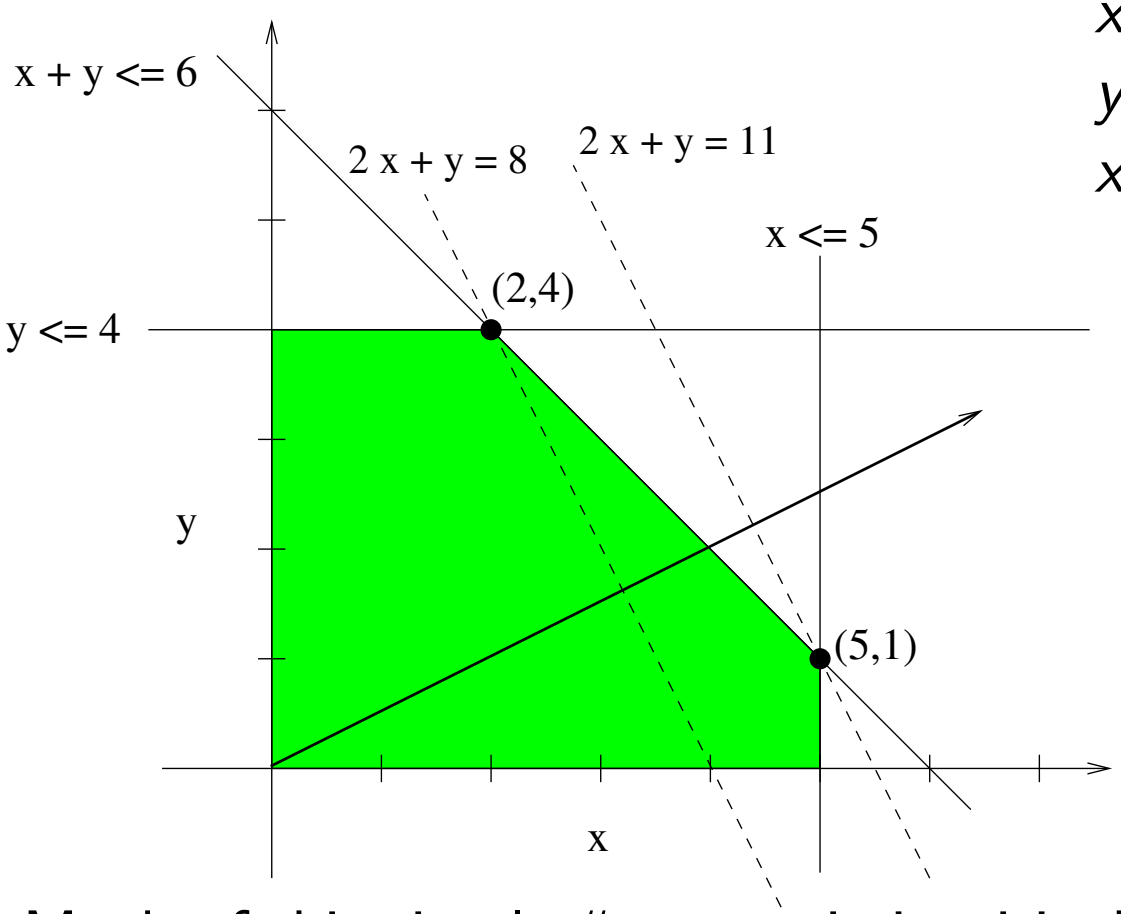
"real world example": the diet problem

- You are a fastidious eater. You want to make sure that every day you get enough of each vitamin: vitamin 1, vitamin 2,...., vitamin m.
- You are also frugal, and want to spend as little as possible.
- There are n foods available to eat: food 1, food 2, ..., food n.
- Each unit of food j has $a_{i,j}$ units of vitamin i.
- Each unit of food j costs c_j.
- Your daily need for vitamin i is b_i units.
- Assume you can buy each food in fractional amounts. (This makes your life much easier.)
- How much of each food would you eat per day in order to have all your daily needs of vitamins, while minimizing your cost?

A Linear Programming Example

Find $(x, y) \in \mathbb{R}^2$ so as to: Maximize $2x + y$

Subject to conditions ("constraints"): $x + y \leq 6$;
$x \leq 5$;
$y \leq 4$;
$x, y \geq 0$;



Much of this simple "geometric intuition" generalizes nicely to higher dimensions. (But be very careful! Things get complicated very quickly!)

The General Linear Program

Definition: A *Linear Programming* or *Linear Optimization* problem instance $(f, \texttt{Opt}, C)$, consists of:

1. A linear *objective function* $f : \mathbb{R}^n \mapsto \mathbb{R}$, given by: $f(x_1, \ldots, x_n) = c_1\ x_1 + c_2\ x_2 + \ldots + c_n\ x_n + d$ where we assume the coefficients c_i and constant d are rational numbers.
2. An optimization criterion: $\texttt{Opt} \in \{\texttt{Maximize}, \texttt{Minimize}\}$.
3. A set (or "system") $C(x_1, \ldots, x_n)$ of m *linear constraints*, or linear inequalities/equalities, $C_i(x_1, \ldots, x_n)$, $i = 1, \ldots, m$, where each $C_i(x)$ has form:

$$a_{i,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{i,n}\ x_n\ \Delta\ b_i$$

where $\Delta \in \{\leq, \geq, =\}$, and where $a_{i,j}$'s and b_i's are rational numbers.

What does it mean to solve an LP?

For a constraint $C_i(x_1, \ldots, x_n)$, we say vector $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ *satisfies* $C_i(x)$ if, plugging in v for the variables $x = (x_1, \ldots, x_n)$, the constraint $C_i(v)$ holds true.

For example, $(3, 6)$ satisfies $-x_1 + x_2 \leq 7$.

$v \in \mathbb{R}^n$ is called a *solution* to a system $C(x)$, if v satisfies every constraint $C_i \in C$. I.e., $C_1(v) \wedge \ldots \wedge C_m(v)$ is true.

Let $K(C) \subseteq \mathbb{R}^n$ denote the set of all solutions to the system $C(x)$. We say C is **feasible** if $K(C)$ is not empty.

An *optimal solution*, for $\texttt{Opt} = \texttt{Maximize}$, is some $x^* \in K(C)$ such that:

$$f(x^*) = \max_{x \in K(C)} f(x)$$

(respectively, $f(x^*) = \min_{x \in K(C)} f(x)$, for $\texttt{Opt} = \texttt{Minimize}$)).

Given an LP problem $(f, \texttt{Opt}, C)$, our goal in principle is to find an "optimal solution".

What does it mean to solve an LP?

For a constraint $C_i(x_1, \ldots, x_n)$, we say vector $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ *satisfies* $C_i(x)$ if, plugging in v for the variables $x = (x_1, \ldots, x_n)$, the constraint $C_i(v)$ holds true.

For example, $(3, 6)$ satisfies $-x_1 + x_2 \leq 7$.

$v \in \mathbb{R}^n$ is called a *solution* to a system $C(x)$, if v satisfies every constraint $C_i \in C$. I.e., $C_1(v) \wedge \ldots \wedge C_m(v)$ is true.

Let $K(C) \subseteq \mathbb{R}^n$ denote the set of all solutions to the system $C(x)$. We say C is **feasible** if $K(C)$ is not empty.
An *optimal solution*, for $\texttt{Opt} = \texttt{Maximize}$, is some $x^* \in K(C)$ such that:

$$f(x^*) = \max_{x \in K(C)} f(x)$$

(respectively, $f(x^*) = \min_{x \in K(C)} f(x)$, for $\texttt{Opt} = \texttt{Minimize}$)).

Given an LP problem $(f, \texttt{Opt}, C)$, our goal in principle is to find an "optimal solution". **Oops!!** There may not be an optimal solution!

Things that can go wrong

Two things can go wrong when looking for an optimal solution:

1. There may be no solutions at all!
   I.e., C is not feasible, i.e., $K(C)$ is empty. Consider:

   Maximize x
   Subject to: $x \leq 3$ *and* $x \geq 5$

2. $\max / \min_{x \in K(C)}$ $f(x)$ may not exist (!), because $f(x)$ is unbounded above/below in $K(C)$. Consider:

   Maximize x
   Subject to: $x \geq 5$

So, we have to revise our goals to handle these cases.

Note: If we allowed strict inequalities, e.g., $x < 5$, there would have been yet another problem:

Maximize x
Subject to: $x < 5$

The LP Problem Statement

Given an LP problem instance $(f, \texttt{Opt}, C)$ as input, output one of the following three:

1. "The problem is Infeasible."
2. "The problem is Feasible But Unbounded."
3. "An Optimal Feasible Solution (OFS) exists.
   One such optimal solution is $x^* \in \mathbb{R}^n$.
   The optimal objective value is $f(x^*) \in \mathbb{R}$."

Oops!! It seems yet another thing could go wrong: What if every optimal solution $x^* \in \mathbb{R}^n$ is irrational?
How can we "output" irrational numbers?
Likewise, what if the Opt value $f(x^*)$ is irrational?

Fact: This problem never arises. The above three answers cover all possibilities, and furthermore, as long as all our coefficients and constants are rational, if an OFS exists, a rational OFS x^* exists, and the optimal value $f(x^*)$ is also rational. (We will learn why later.)

Simplified forms for LP problems

1. In principle, we need only consider Maximization, because:
$$\min_{x \in K} f(x) = -\max_{x \in K} -f(x)$$
(either side is unbounded if and only if both are.)
2. We only need an objective function $f(x_1, \ldots, x_n) = x_i$, for some x_i, because we can:
Introduce new variable x_0. Add new constraint $f(x) = x_0$ to constraints C. Make the new objective "Optimize x_0".
3. Don't need "=" constraints: $\alpha = \beta \Leftrightarrow (\alpha \leq \beta \wedge \alpha \geq \beta)$.
4. Don't need "$\alpha \geq b$", where $b \in \mathbb{R}$: $\alpha \geq b \Leftrightarrow -\alpha \leq -b$.
5. We can constrain every variable x_i to be $x_i \geq 0$:
Introduce two variables x_i^+, x_i^- for each variable x_i.
Replace each occurence of x_i by $(x_i^+ - x_i^-)$, and add the constraints $x_i^+ \geq 0$, $x_i^- \geq 0$. Non-negative
(**N.B.** can't do both (2.) and (5.) together.)

A lovely but terribly inefficient algorithm for LP

Input: LP instance $(x_0, \texttt{Opt}, C(x_0, x_1, \ldots, x_n))$.

1. **For** $i = n$ downto 1
   a. Rewrite each constraint involving x_i as $\alpha \leq x_i$, or as $x_i \leq \beta$. (One of the two is possible.) Let these be:
   $\alpha_1 \leq x_i, \ldots, \alpha_k \leq x_i \;\; ; \;\; x_i \leq \beta_1, \ldots, x_i \leq \beta_r$
   (Retain these constraints, H_i, for later.)
   b. Remove H_i, i.e., all constraints involving x_i. Replace with constraints: $\{\alpha_j \leq \beta_l \mid j = 1, \ldots, k, \& l = 1, \ldots, r\}$.
2. Only x_0 (or no variable) remains. All constraints have the forms $a_j \leq x_0$, $x_0 \leq b_l$, or $a_j \leq b_l$, where a_j's and b_l's are constants. It's easy to check "feasibility" & "boundedness" for such a one(or zero)-variable LP, and to find an optimal x_0^* if one exists.
3. Once you have x_0^*, plug it into H_1. Solve for x_1^*. Then use x_0^*, x_1^* in H_2 to solve for x_2^*, ..., use $x_0^*, \ldots, x_{i-1}^*$ in H_i to solve for x_i^*. ... then $x^* = (x_0^*, \ldots, x_n^*)$ is an optimal feasible solution.

remarks on the lovely algorithm

- This algorithm was first discovered by Fourier (1826). Rediscovered in 1900's, by Motzkin (1936) and others.
- It is called *Fourier-Motzkin Elimination*, and can be viewed as a generalization of *Gaussian Elimination*, used for solving systems of linear equalities.
- Why is Fourier-Motzkin so inefficient? In the worst case, if every variable x_i is involved in every constraint, each iteration of the "For loop" squares the number of constraints. So, toward the end we could have roughly m^{2^n} constraints!! f(i+1)=(f(i)/2)^2
- Let's recall Gaussian Elimination (GE). It is much nicer and does not suffer from this explosion.
- In 1947, Dantzig invented the celebrated **Simplex Algorithm** for LP. It can be viewed as a much more refined generalization of GE. Next time, Simplex!

more remarks

Immediate Corollaries of Fourier-Motzkin:

Corollary 1: The three possible "answers" to an LP problem do cover all possibilities.
(In particular, unlike "Maximize x; $x < 5$", If an LP has a "Supremum" it has a "Maximum".)

Corollary 2: If an LP has an OFS, then it has a rational OFS, x^*, and $f(x^*)$ is also rational.
Proof: We used only addition, multiplication, & division by rationals to arrive at the solution. □

further remarks

Although Fourier-Motzkin is bad in the worst case, it can still be quite useful. It can be used to remove redundant variables and constraints. And its worst-case behavior may in some cases not arise in practice.

Generalizations of Fourier-Motzkin are used in some tools (e.g., [Pugh,'92]) for solving "Integer Linear Programming", where we seek an optimal solution x^* not in $\mathbb{R}^n$, but in $\mathbb{Z}^n$. ILP is a **much harder** problem! (**NP**-complete.)

For ordinary LP however, Fourier-Motzkin can't compete with Simplex.

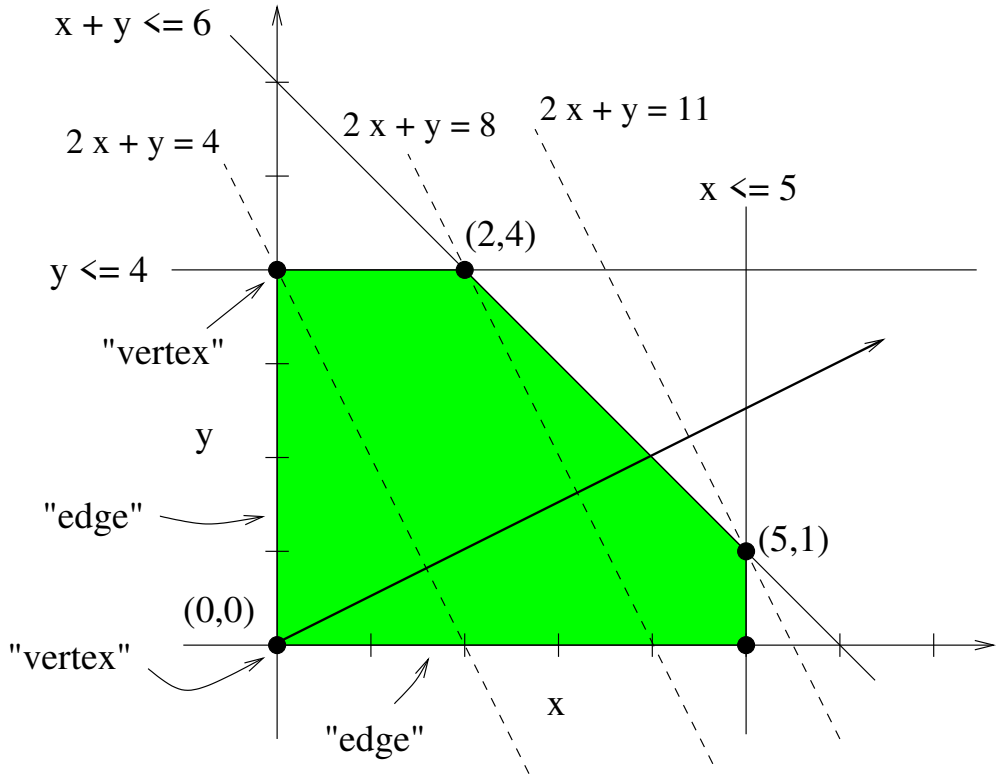
- **Food for Thought:** Think about what kinds of clever heuristics and hacks you could use during Fourier-Motzkin to keep the number of constraints as small as possible. E.g., In what order would you try to eliminate variables? (Clearly, any order is fine, as long as x_0 is last.)

Algorithmic Game Theory and Applications

Lecture 6: The Simplex Algorithm

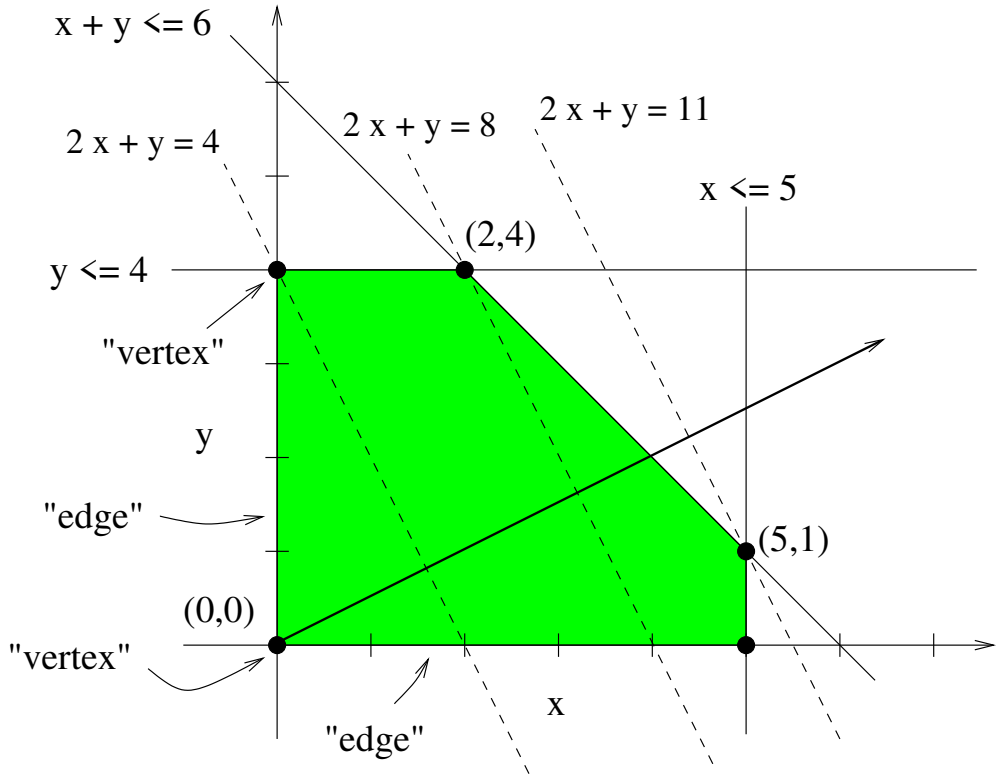
Kousha Etessami

Recall our example



Note: starting at $(0, 0)$, we can find the optimal vertex $(5, 1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

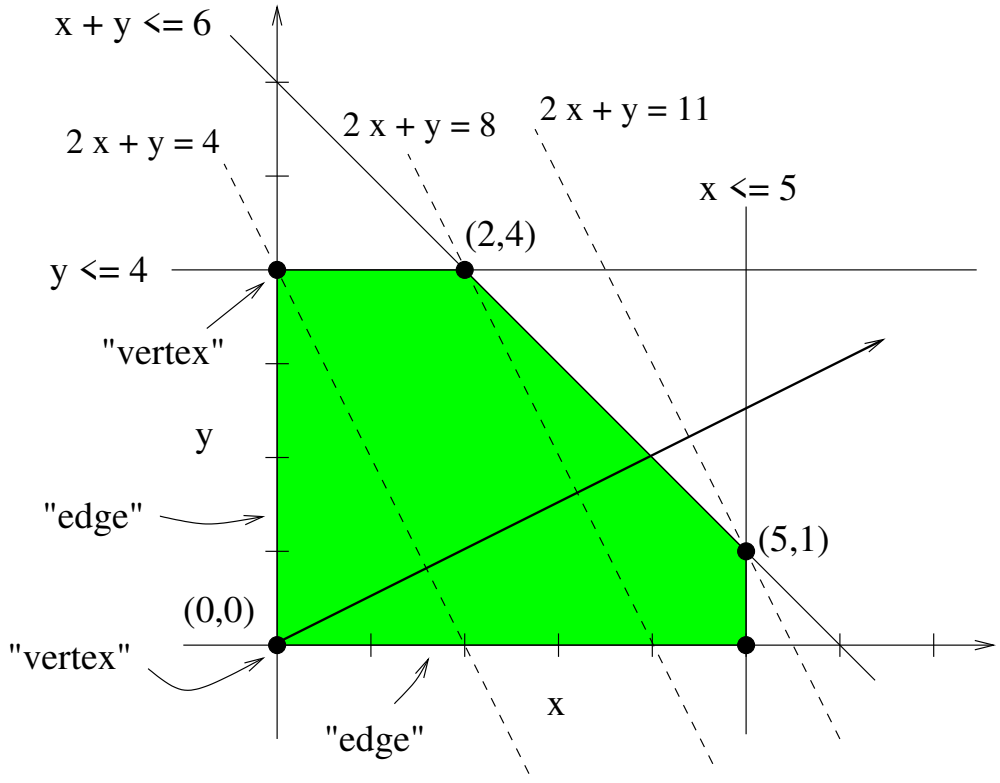
Recall our example



Note: starting at $(0, 0)$, we can find the optimal vertex $(5, 1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

We don't seem to get "stuck" in any "locally optimal" vertex.

Recall our example



Note: starting at $(0, 0)$, we can find the optimal vertex $(5, 1)$, by repeatedly moving from a vertex to a neighboring vertex (crossing an "edge") that improves the value of the objective.

We don't seem to get "stuck" in any "locally optimal" vertex. That's the geometric idea of simplex!

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

Key reason: The region $K(C)$ is convex. (Recall: K is convex iff for all $x, y \in K$, $\lambda x + (1 - \lambda)y \in K$, for all $\lambda \in [0, 1]$.)

Fact: On a convex region, a "local optimum" of a linear objective is always a "global optimum".

geometric idea of simplex

Input: Given $(f, \texttt{Opt}, C)$ and a start "vertex" $x \in K(C) \subseteq \mathbb{R}^n$.

(Never mind, for now, that we have no idea how to find any $x \in K(C)$ -or even if C is Feasible!- let alone a "vertex".)

While (x has a "neighbor vertex", x', with $f(x') > f(x)$)

- Pick such a neighbor x'. Let $x := x'$.
- (If the "neighbor" is at "infinity", Output: "Unbounded".)

Output: $x^* := x$ is optimal solution, with optimal value $f(x^*)$.

Question: Why should this work? Why don't we get "stuck" in some "local optimum"?

Key reason: The region $K(C)$ is convex. (Recall: K is convex iff for all $x, y \in K$, $\lambda x + (1-\lambda)y \in K$, for all $\lambda \in [0,1]$.)

Fact: On a convex region, a "local optimum" of a linear objective is always a "global optimum".

Ok. The geometry sounds nice and simple. But realizing it algebraically is not a trivial matter!

LP's in "Primal Form"

Using the simplification rules from the last lecture, we can convert any LP into the following form:

Maximize $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1} x_1 + a_{1,2} x_2 + \ldots + a_{1,n} x_n &\leq b_1 \\
a_{2,1} x_1 + a_{2,2} x_2 + \ldots + a_{2,n} x_n &\leq b_2 \\
\ldots &\ldots \\
\ldots &\ldots \\
a_{m,1} x_1 + a_{i,2} x_2 + \ldots + a_{m,n} x_n &\leq b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0$$

Side comment: Carrying along the constant d in the objective $f(x)$ seems silly: it doesn't affect the optimality of a solution. (It only shifts the value of a solution by d.) We keep the constant for convenience, to become apparent later.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\ x_1 + c_2\ x_2 + \ldots + c_n\ x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1}\ x_1 + a_{1,2}\ x_2 + \ldots + a_{1,n}\ x_n + y_1 &= b_1 \\
a_{2,1}\ x_1 + a_{2,2}\ x_2 + \ldots + a_{2,n}\ x_n + y_2 &= b_2 \\
\ldots & \ldots \\
a_{m,1}\ x_1 + a_{i,2}\ x_2 + \ldots + a_{m,n}\ x_n + y_m &= b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n + y_1 &= b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n + y_2 &= b_2 \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n + y_m &= b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n + y_1 &= b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n + y_2 &= b_2 \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n + y_m &= b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.
2. Picking one such variable, y_i, for each equality, we obtain a set of m variables $B = \{y_1, \ldots, y_m\}$ called a **Basis**.

slack variables

By adding a "slack" variable y_i to each inequality, we get an "equivalent" LP (explain), with only equalities (& non-neg):

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$
\begin{aligned}
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n + y_1 &= b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n + y_2 &= b_2 \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n + y_m &= b_m
\end{aligned}
$$

$$x_1, \ldots, x_n \geq 0; \quad y_1, \ldots, y_m \geq 0$$

This new LP has some particularly nice properties:

1. Every equality constraint has at least one variable with coefficient 1 that doesn't appear in any other equality.
2. Picking one such variable, y_i, for each equality, we obtain a set of m variables $B = \{y_1, \ldots, y_m\}$ called a **Basis**.
3. Objective $f(x)$ involves only non-Basis variables.

Let's call an LP in this form a "dictionary".

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + d$

Subject to:

$$\begin{aligned} x_{n+1} &= b_1 - a_{1,1} x_1 - a_{1,2} x_2 - \ldots - a_{1,n} x_n \\ x_{n+2} &= b_2 - a_{2,1} x_1 - a_{2,2} x_2 - \ldots - a_{2,n} x_n \\ \ldots & \quad \ldots \\ x_{n+m} &= b_m - a_{m,1} x_1 - a_{i,2} x_2 - \ldots - a_{m,n} x_n \end{aligned}$$

$$x_1, \ldots, x_{n+m} \geq 0$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$. The objective value is then $f(0) = d$.

a vertex related to more than one constrains

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

x_j >=0

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n + d$

Subject to:

$$\begin{aligned} x_{n+1} &= b_1 - a_{1,1}\, x_1 - a_{1,2}\, x_2 - \ldots - a_{1,n}\, x_n \\ x_{n+2} &= b_2 - a_{2,1}\, x_1 - a_{2,2}\, x_2 - \ldots - a_{2,n}\, x_n \\ \ldots & \quad \ldots \\ x_{n+m} &= b_m - a_{m,1}\, x_1 - a_{i,2}\, x_2 - \ldots - a_{m,n}\, x_n \end{aligned}$$

$$x_1, \ldots, x_{n+m} \geq 0$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$. The objective value is then $f(0) = d$.

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

Question: How do we move from one BFS with basis B to a "neighboring" BFS with basis B'?

Basic Feasible Solutions

Rewrite our dictionary (renaming "y_i", "x_{n+i}") as:

Maximize $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + d$

Subject to:

$$\begin{aligned} x_{n+1} &= b_1 - a_{1,1} x_1 - a_{1,2} x_2 - \ldots - a_{1,n} x_n \\ x_{n+2} &= b_2 - a_{2,1} x_1 - a_{2,2} x_2 - \ldots - a_{2,n} x_n \\ \ldots & \quad \ldots \\ x_{n+m} &= b_m - a_{m,1} x_1 - a_{i,2} x_2 - \ldots - a_{m,n} x_n \end{aligned}$$

$$x_1, \ldots, x_{n+m} \geq 0$$

Suppose, somehow, $b_i \geq 0$ for all $i = 1, \ldots, m$. Then we have a "*feasible dictionary*" and a feasible solution for it, namely, let $x_{n+i} := b_i$, for $i = 1, \ldots, m$, and let $x_j := 0$, for $j = 1, \ldots, n$. The objective value is then $f(0) = d$.

Call this a *basic feasible solution*(BFS),with basis B.

Geometry: A BFS corresponds to a "vertex". (But different Bases B may yield the same BFS!)

Question: How do we move from one BFS with basis B to a "neighboring" BFS with basis B'? **Answer:** Pivoting!

Pivoting

Suppose our current dictionary basis (the variables on the left) is $B = \{x_{i_1}, \ldots, x_{i_m}\}$, with x_{i_r} the variable on the left of constraint C_r.
The following pivoting procedure moves us from basis B to basis $B' := (B \setminus \{x_{i_r}\}) \cup \{x_j\}$.
Pivoting to add x_j and remove x_{i_r} from basis B:

1. Assuming C_r involves x_j, rewrite C_r as $x_j = \alpha$.
2. Substitute α for x_j in other constraints C_l, obtaining C'_l.
3. The new constraints C', have a new basis:
   $B' := (B \setminus \{x_{i_r}\}) \cup \{x_j\}$.
4. Also substitute α for x_j in $f(x)$, so that $f(x)$ again only depends on variables not in the new basis B'.

This new basis B' is a "possible neighbor" of B.
However, not every such basis B' is eligible!

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b'_i remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b'_i remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.
2. The new BFS must improve, or at least must not decrease, the value $d' = f(0)$ of the new objective function. (Recall, all non-basic variables are set to 0 in a BFS, thus $f(BFS) = f(0)$.)

sanity checks for pivoting

To check *eligibility* of a pivot, we have to make sure:

1. The new constants b'_i remain ≥ 0, so we retain a "feasible dictionary", and thus B' yields a BFS.
2. The new BFS must improve, or at least must not decrease, the value $d' = f(0)$ of the new objective function. (Recall, all non-basic variables are set to 0 in a BFS, thus $f(BFS) = f(0)$.)
3. We should also check for the following situations:
   (a) Suppose all variables in $f(x)$ have negative coefficients. Then any increase from 0 in these variables will decrease the objective. We are thus at an optimal BFS x^*. Output: Opt-BFS: x^* & $f(x^*) = f(0) = d'$.

   (b) Suppose a variable x_j in $f(x)$ has coefficient $c_j > 0$, and the coefficient of x_j in every constraint C_r is ≥ 0. Then we can increase x_j, and objective, to "infinity" without violating constraints. So, Output: "Feasible but Unbounded".

finding and choosing eligible pivots

- In principle, we could exhaustively check the sanity conditions for eligibility of all potential pairs of entering and leaving variables. There are at most $(n * m)$ candidates.
- But, there are much more efficient ways to choose pivots, by inspection of the coefficients in the dictionary.
- We can also efficiently choose pivots according to lots of additional criteria, or pivoting rules, such as, e.g., "most improvement in objective value", etc.
- There are many such "rules", and it isn't clear *a priori* what is "best".

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$
\begin{aligned}
x_4 &= 3 - \,x_1 - 3\,x_2 - \,x_3 \\
x_5 &= 4 - 2\,x_1 + \,x_2 - 2\,x_3 \\
x_6 &= 2 - 2\,x_1 - 4\,x_2 + \,x_3
\end{aligned}
$$

$x_1, \ldots, x_6 \geq 0;$

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$

$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$

$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$

$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$

$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$
$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$
$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$
$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$
$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.

x_4: x_2=1
x_5: feasible
x_6: x_2=1/2

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$
$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$
$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.
- The latter constraint is "tightest". So, once x_2 is chosen to enter the basis, the *unique* variable that can leave the basis while maintaining feasiblility is x_6.

example of a simplex pivoting step

Maximize $2x_1 + 3\,x_2 + 4x_3 + 8$

Subject to:

$$x_4 = 3 - x_1 - 3\,x_2 - x_3$$
$$x_5 = 4 - 2\,x_1 + x_2 - 2\,x_3$$
$$x_6 = 2 - 2\,x_1 - 4\,x_2 + x_3$$

goal: make objective function into d-c(x_1)..., where d is the maximum

$x_1, \ldots, x_6 \geq 0$; *the initial BFS is:* $(0, 0, 0, 3, 4, 2)$

Suppose we next choose to move x_2 into the basis.

- Constraint $x_4 = 3 - \ldots - 3x_2 - \ldots$ means we can *at most* increase x_2 to $= 1$ while maintaining a *feasible* dictionary.
- On the other hand, constraint $x_6 = 2 - \ldots - 4x_2 + \ldots$, means we can at most increase x_2 to $= 1/2$.
- The latter constraint is "tightest". So, once x_2 is chosen to enter the basis, the *unique* variable that can leave the basis while maintaining feasiblility is x_6.
- So we rewrite constraint $x_6 = 2 - 2x_1 - 4x_2 + x_3$ as: (**) $x_2 = \frac{1}{2} - \frac{1}{2}\,x_1 + \frac{1}{4}x_3 - \frac{1}{4}x_6$, and rewrite the objective and other constraints, by replacing x_2 with RHS of (**).

The Simplex Algorithm

Dantzig's Simplex algorithm can be described as follows:
Input: a feasible dictionary;

Repeat

1. *Check if we are at an optimal solution, and if so, Halt and output the solution.*
2. *Check if we have an "infinity" neighbor, and if so Halt and output "Unbounded".*
3. *Otherwise, choose an eligible pivot pair of variables, and Pivot!*

Fact If this halts the output is correct: an output solution is an optimal solution of the LP.
Oops! We could cycle back to the same basis for ever, never strictly improving by pivoting.
There are several ways to address this problem.....

how to prevent cycling

Several Solutions:

1. Carefully choose rules for variable pairs to pivot at, in a way that forces cycling to never happen.
   Fact: This can be done.
   (For example, use "*Bland's rule*": For all eligible pivot pairs (x_i, x_j), where x_i is being added the basis and x_j is being removed from it, choose the pair such that, first, i is as small as possible, and second, j is as small as possible.)
2. Choose randomly among eligible pivots.
   With probability 1, you'll eventually get out and to an optimal BFS.
3. "Perturb" the constraints slighty to make the LP "non-degenerate". (More rigorously, implement this using, e.g., the "lexicographic method".)

Perturb: move one line slightly

three line with one vertex, pivot wouldn't work

the geometry revisited

- Moving to a "neighboring" basis by pivoting roughly corresponds to moving to a neighboring "vertex". However, not literally true because several Bases can correspond to same BFS, and thus to same "vertex".

  We may not have any neighboring bases that strictly improve the objective, and yet still not be optimal, because all neighoring bases B' describe the same BFS "from a different point of view".
- pivoting rules can be designed so we never return to the same "point of view" twice.
- choosing pivots randomly guarantees that we eventually get out.
- properly "perturbing" the constraints makes sure every BFS corresponds to a unique basis (i.e., we are non-degenerate), and thus bases and "vertices" are in 1-1 correspondence.

Hold on! What about finding an initial BFS?

- So far, we have cheated: we have assumed we start with an initial "feasible dictionary", and thus have an initial BFS.
- Recall, the LP may not even be feasible!
- Luckily, it turns out, it is as easy (using Simplex) to find whether a feasible solution exists (and if so to find a BFS) as it is to find the optimal BFS given an initial BFS.....

checking feasibility via simplex

Consider the following new LP: **Maximize** $-x_0$

Subject to:

$$
\begin{aligned}
a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n - x_0 &\leq b_1 \\
a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n - x_0 &\leq b_2 \\
\ldots & \ldots \\
a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n - x_0 &\leq b_m
\end{aligned}
$$

$$x_0, x_1, \ldots, x_n \geq 0$$

- This LP is feasible: let $x_0 = -\min\{b_1, \ldots, b_m, 0\}$, $x_j = 0$, for $j = 1, \ldots, n$. We can also get a feasible dictionary, and thus initial BFS, for it by adding slack variables.
- *Key point:* the original LP is feasible if and only if in an optimal solution to the new LP, $x_0^* = 0$.
- It also turns out, it is easy to derive a BFS for the original LP from an optimal BFS for this new LP.

how efficient is simplex?

- Each pivoting iteration can be performed in $O(mn)$ arithmetic operations.
  Also, it can be shown that the coefficients never get "too large" (they stay polynomial-sized), as long as rational coefficients are kept in reduced form (e.g., removing common factors from numerator and denominator).
  So, each pivot can be done in "polynomial time".
- How many pivots are required to get to the optimal solution? Unfortunately, it can be exponentially many!
- In fact, for most "pivoting rules" known, there exist worst case examples that force exponentially many iterations. (E.g., Klee-Minty (1972).)
- Fortunately, simplex tends to be very efficient in practice: requiring $O(m)$ pivots on typical examples.

even eligible pivot rule

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10].
  Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10]. Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].
- Breakthrough: [Khachian'79] proved the LP problem does have a polynomial time algorithm, using a completely different approach: The "Ellipsoid Algorithm". Ellipsoid is theoretically very important, but not practical.

more on theoretical efficiency

- It is an open problem whether there exists a pivoting rule that achieves polynomially many pivots on all LPs.
- A randomized pivoting rule is known that requires $m^{O(\sqrt{n})}$ expected pivots [Kalai'92], [Matousek-Sharir-Welzl'92].
- Is there, in every LP, a polynomial-length "path via edges" from every vertex to every other? Without this, there can't be any polynomial pivoting rule.
  Hirsch Conjecture: "diameter $\leq m - n$". Disproved [Santos'10].
  Best known bound is $\leq m^{O(\log n)}$ [Kalai-Kleitman'92].
- Breakthrough: [Khachian'79] proved the LP problem does have a polynomial time algorithm, using a completely different approach: The "Ellipsoid Algorithm". Ellipsoid is theoretically very important, but not practical.
- Breakthrough: [Karmarkar'84] gave a completely different P-time algorithm, using "the interior-point method". It is competitive with simplex in many cases.

final remarks and food for thought

- Why is the Simplex algorithm so fast in practice? Some explanation is offered by [Borgwardt'77]'s "average case" analysis of Simplex. More convincing explanation is offered by [Spielman-Teng'2001]'s "smoothed analysis" of Simplex (not "light reading").
- Ok. Enough about Simplex. So we now have an efficient algorithm for, among other things, finding minimax solutions to 2-player zero-sum games.
- Next time, we will learn about the very important concept of Linear Programming Duality. LP Duality is closely related to the Minimax theorem, but it has far reaching consequences in many subjects.
- **Food for thought:** Suppose you have a solution to an LP in Primal Form, and you want to convince someone it is optimal. How would you do it?

Algorithmic Game Theory and Applications

Lecture 7: The LP Duality Theorem

Kousha Etessami

recall LP's in "Primal Form"

Maximize $c_1\, x_1 + c_2\, x_2 + \ldots + c_n\, x_n$

Subject to:

$$\begin{aligned} a_{1,1}\, x_1 + a_{1,2}\, x_2 + \ldots + a_{1,n}\, x_n &\leq b_1 \\ a_{2,1}\, x_1 + a_{2,2}\, x_2 + \ldots + a_{2,n}\, x_n &\leq b_2 \\ \ldots & \ldots \\ a_{m,1}\, x_1 + a_{i,2}\, x_2 + \ldots + a_{m,n}\, x_n &\leq b_m \end{aligned}$$

$$x_1, \ldots, x_n \geq 0$$

An LP can concisely be represented in matrix notation:
Define the $(m \times n)$-matrix A by: $(A)_{i,j} = a_{i,j}$. Define (column) vectors $x = [x_1, \ldots, x_n]^T$, $b = [b_1, \ldots, b_m]^T$ & $c = [c_1, \ldots, c_n]^T$. The LP is:

Maximize $c^T x$

Subject to:

$$Ax \leq b$$

$$x_1, \ldots, x_n \geq 0$$

Solving LPs against an Adversary

Suppose you want to optimize this Primal LP:

Maximize $c^T x$
Subject to:
$Ax \leq b$
$x_1, \ldots, x_n \geq 0$

Suppose an "Adversary" comes along with a m-vector $y \in \mathbb{R}^m$, $y \geq 0$, such that: $c^T \leq y^T A$.
For any solution, x, you may have for the Primal:

$$\begin{aligned} c^T x &\leq (y^T A)x \quad (\text{because } x \geq 0 \ \& \ c^T \leq y^T A) \\ &= y^T(Ax) \\ &\leq y^T b \qquad (\text{because } y^T \geq 0 \ \& \ Ax \leq b) \end{aligned}$$

So, the adversary's goal is to minimize $y^T b$, subject to $c^T \leq y^T A$, and $y \geq 0$, i.e., to optimize the **Dual LP**.

Dual LP, and "Weak Duality"

Given the **Primal LP**:

Maximize $c^T x$
Subject to: n constrains
$Ax \leq b$
$x_1, \ldots, x_n \geq 0$

The **Dual LP** is:

Minimize $b^T y$
Subject to: m constrains
$A^T y \geq c$
$y_1, \ldots, y_m \geq 0$

The LP Duality Theorem

As we already saw, any feasible value of the primal LP is no bigger than any feasible value of the dual LP, i.e.:

Proposition ("Weak Duality") If $x^* \in R^n$ and $y^* \in R^m$ are (optimal) feasible solutions to the primal and dual LPs, then:

$$c^T x^* \leq b^T y^*$$

The LP Duality Theorem

As we already saw, any feasible value of the primal LP is no bigger than any feasible value of the dual LP, i.e.:

Proposition ("Weak Duality") If $x^* \in R^n$ and $y^* \in R^m$ are (optimal) feasible solutions to the primal and dual LPs, then:

$$c^T x^* \leq b^T y^*$$

Amazingly, when x^* and y^* are optimal, equality holds!

The LP Duality Theorem

As we already saw, any feasible value of the primal LP is no bigger than any feasible value of the dual LP, i.e.:

Proposition ("Weak Duality") If $x^* \in R^n$ and $y^* \in R^m$ are (optimal) feasible solutions to the primal and dual LPs, then:

$$c^T x^* \leq b^T y^*$$

Amazingly, when x^* and y^* are optimal, equality holds!

Theorem("Strong Duality", von Neumann'47) One of the following four situations holds:

1. Both the primal and dual LPs are feasible, and for any optimal solutions x^* of the primal and y^* of the dual:

minimaximize problem

$$c^T x^* = b^T y^*$$

2. The primal is infeasible and the dual is unbounded.
3. The primal is unbounded and the dual is infeasible.
4. Both LPs are infeasible.

Simplex and the Duality Theorem

- It turns out, LP Duality follows from the "proof of correctness" of the Simplex algorithm (which we didn't provide).
- In fact, suppose Simplex on the Primal LP halts and produces an optimal solution vector x^*.
  Let $y_j^* = -\hat{c}_{n+j}$, where $\hat{c}_{n+j}$ is the coefficient of the "slack variable" x_{n+j} in the last objective function $f(x)$ associated with the final dictionary for the primal LP. This defines $y^* \in \mathbb{R}^m$, and it turns out the following holds:
  Fact $y^* \geq 0$, $A^T y^* \geq c$, and $c^T x^* = b^T y^*$.
- So, using Simplex, we can actually compute an optimal solution for the dual while we compute an optimal solution for the primal (or find out neither exists).

Complementary Slackness

Useful Corollary of LP Duality: solutions x^* and y^* to the primal and dual LPs, respectively, are both optimal if and only if both of the following hold:

- For each primal constraint, $(Ax)_i \leq b_i$, $i = 1, \ldots, m$, either $(Ax^*)_i = b_i$ or $y_i^* = 0$ (or both).
- For each dual constraint, $(A^T y)_j \geq c_j$, $j = 1, \ldots, n$, either $(A^T y^*)_j = c_j$ or $x_j^* = 0$ (or both).

Proof By weak duality

$$c^T x^* \leq ((y^*)^T A)x^* = (y^*)^T (Ax^*) \leq (y^*)^T b$$

By strong duality each inequality holds with equality precisely when x^* and y^* are optimal. So, when optimal, $(((y^*)^T A) - c^T)x^* = 0$.
Since both $((y^*)^T A) - c^T) \geq 0$ and $x^* \geq 0$, it must be that for each $j = 1, \ldots, n$, $(A^T y^*)_j = c_j$ or $x_j^* = 0$.
Likewise, when optimal, $(y^*)^T (b - Ax^*) = 0$, and thus, for each $i = 1, \ldots, m$, $(Ax^*)_i = b_i$ or $y_i^* = 0$.

general recipe for LP duals

If the "primal" is:

Maximize $c^T x$
Subject to:
$(Ax)_i \leq b_i$, $i = 1, \ldots, d$,
$(Ax)_i = b_i$, $i = d+1, \ldots, m$,
$x_1, \ldots, x_r \geq 0$

Then the "dual" is:

Minimize $b^T y$
Subject to:
$(A^T y)_j \geq c_j$, $j = 1, \ldots, r$,
$(A^T y)_j = c_j$, $j = r+1, \ldots, n$,
$y_1, \ldots, y_d \geq 0$

Strong Duality holds also in this more general setting.
Fact: The "dual" of the "dual" is the primal.

Duality and the Minimax Theorem

Recall the LP for solving Minimax for Player 1 in a zero-sum game given by matrix A:

Maximize v
Subject to:
$v - (x^T A)_j \leq 0$ *for* $j = 1, \ldots, m_2$,
$x_1 + \ldots + x_{m_1} = 1$,
$x_i \geq 0$ *for* $j = 1, \ldots, m_1$.

What is the Dual to this LP? It can be shown to be:

Minimize u
Subject to:
$u - (Ay)_i \geq 0$ *for* $i = 1, \ldots, m_1$,
$y_1 + \ldots + y_{m_2} = 1$
$y_j \geq 0$ *for* $j = 1, \ldots, m_2$.

This is exactly the LP for Player 2's optimal strategy!
Thus, the minimax theorem follows from LP Duality: the optimal value for the two LPs is the same.

From Minimax to LP Duality

In fact, LP Duality can also be "derived" from the Minimax Theorem. Consider the LP (with A a $(m \times n)$-matrix):

Maximize $c^T x$
Subject to:
$Ax \leq b$
$x_1, \ldots, x_n \geq 0$

Proof (sketch) that Minimax $\Rightarrow$ LP-duality: Consider the zero-sum payoff matrix:

$$B = \begin{bmatrix} 0 & A & -b \\ -A^T & 0 & c \\ b^T & -c^T & 0 \end{bmatrix}$$

This is a symmetric zero-sum game: $B = -B^T$. By a simple exercise, such a game must have minimax value 0, and every minmaximizer for player 1 is also a maxminimizer for player 2.

Minimax $\Rightarrow$ LP Duality, continued

Consider a symmetric minimax profile for the zero-sum game with payoff matrix B:

$$((Y^*, X^*, z), (Y^*, X^*, z))$$

Here Y^* is a (row) vector of length m, and X^* is a (row) vector of length n, and $z \in \mathbb{R}$.

Suppose $z > 0$, in the minimax strategy (Y^*, X^*, z).
Note that, by the Minimax Theorem, we have B @ (Y*,X*,z) <=0

$$AX^* - bz \leq 0 \quad \text{and} \quad cz - A^T Y^* \leq 0$$

Letting $y^* = (1/z)Y^*$, and $x^* = (1/z)X^*$, we would have $Ax^* \leq b$ and $A^T y^* \geq c$, i.e., feasible solutions for the LP and its Dual.

Minimax $\Rightarrow$ LP-duality, continued

Moreover, by Useful Corollary to Minimax, player 1 can switch to ANY pure strategy j in its "support" (i.e., where $x_1(j) > 0$) and not change its profit. Let it switch to the last row (i.e., letting $z = 1$). Then we also have: $bY^* - cX^* = 0$, and hence $by^* - cx^* = 0$. Thus,

$$by^* = cx^*$$

The only thing left to prove is:
Claim If both the LP and its Dual are feasible, the game B has some minimax profile where $z > 0$.

remarks

- This last claim can be proved (see [Dantzig'51], [Raghavan,HGT,Ch.20]) using facts related to the "geometry" of LP and Minimax: specifically the "Farkas lemmas".
- Such "Farkas Lemmas" can actually be proved very nicely using Fourier-Motzkin elimination.
  Here is a Farkas lemma:
  $Ax \leq b$ has no solutions if and only if there exists $y \geq 0$ such that $y^T A = 0$ and $y^T b < 0$.
  (HW1 asks you to prove this.)
- This proof is somewhat unsatisfactory because the Farkas lemmas are essentially "equivalent" to LP-duality. (A recent modification of this proof, by [Adler 2013], avoids the use of Farkas lemmas.)

Significance of LP Duality

- The Duality Theorem is an extremely important fact. It has many uses, e.g., in (combinatorial) optimization and mathematical economics.

Significance of LP Duality

- The Duality Theorem is an extremely important fact. It has many uses, e.g., in (combinatorial) optimization and mathematical economics.
- Often, you can "learn something" about an LP by looking at its dual.

Significance of LP Duality

- The Duality Theorem is an extremely important fact. It has many uses, e.g., in (combinatorial) optimization and mathematical economics.
- Often, you can "learn something" about an LP by looking at its dual.
- In Economics, one often sets up an LP to optimize some economic goal. Often, the dual LP can be meaningfully interpreted as a problem of optimizing some real economic "counter" goal.
  The fact that the optimal solutions of the two goals are the same is very powerful.

Significance of LP Duality

- The Duality Theorem is an extremely important fact. It has many uses, e.g., in (combinatorial) optimization and mathematical economics.
- Often, you can "learn something" about an LP by looking at its dual.
- In Economics, one often sets up an LP to optimize some economic goal. Often, the dual LP can be meaningfully interpreted as a problem of optimizing some real economic "counter" goal.
  The fact that the optimal solutions of the two goals are the same is very powerful.
- Duality has many consequences in algorithms.
  For example, duality implies that LPs can be solved in "NP $\cap$ co-NP".
  (Of course, we know from [Khachian'79] that the LP problem is in "P".)

more on algorithmic significance

(*) approximation algorithms: hard combinatorial optimization problems can often be formulated as "Integer Linear Progams" (ILPs). One can often use the "LP-relaxation" of the ILP together with its Dual to find approximate solutions and to bound the proximity of the approximate solution to the optimal. (See, e.g., [Vazirani'2001].)

(*) Many important results in combinatorics can be viewed as particular manifestations of LP Duality.

- Max-Flow Min-Cut Theorem; Hall's Theorem; Dilworth's Theorem; Konig-Egervary Theorem.....

Each result says "the maximum value of one useful quantity associated with a combinatorial object is the same as the minimum value of another useful quantity associated with it." These follow from LP-Duality: dual LPs for these "complementary" quantities can be set up (with solutions that consist necessarily of integers, due to the LPs' structure).

food for thought (and "thought for food")

- Recall the "Diet Problem". It has the form:

  Minimize $c^T x$
  Subject to:
  $Ax \geq b$
  $x_1, \ldots, x_n \geq 0$

- Construct the dual to this LP.
- What do the dual variables "mean" in the context of the diet problem?
  Try to come up with an interpretation.
  (Hint: try to assign consistent "units of measure" to the primal variables, constants, and coefficients. These will determine the "units of measure" of the dual variables, and will guide you toward an interpretation.)
- What is the dual LP trying to optimize?

Algorithmic Game Theory and Applications

Lecture 8: Computing solutions for General Strategic Games: Part 1: dominance and iterated strategy elimination

Kousha Etessami

a partial-order on strategies: dominance

Let's again consider general finite strategic games.

Definition For $x_i, x_i' \in X_i$, we say x_i **dominates** x_i', denoted $x_i \succeq x_i'$, if for all $x_{-i} \in X_{-i}$,

$$U_i(x_{-i}; x_i) \geq U_i(x_{-i}; x_i')$$

We say x_i **strictly dominates** x_i', denoted $x_i \succ x_i'$, if for all $x_{-i} \in X_{-i}$

$$U_i(x_{-i}; x_i) > U_i(x_{-i}; x_i')$$

Proposition x_i dominates x_i' if and only if for all pure "counter profiles" $\pi_{-i} \in X_{-i}$

$$\pi_{-i} = (\pi_{1,j_1}, \ldots, \texttt{empty}, \ldots, \pi_{n,j_n}),$$

$$U_i(\pi_{-i}; x_i) \geq U_i(\pi_{-i}; x_i').$$

Likewise, x_i strictly dominates x_i' iff for all π_{-i}

$$U_i(\pi_{-i}; x_i) > U_i(\pi_{-i}; x_i')$$

Proof Another easy "weighted average" argument. □

obviously good strategies: dominant strategies

Definition A mixed strategy $x_i \in X_i$ is **dominant** if for all $x_i' \in X_i$, $x_i \succeq x_i'$. x_i is **strictly dominant** if for all $x_i' \in X_i$ such that $x_i' \neq x_i$, $x_i \succ x_i'$.

Definition For a mixed strategy x_i, its **support**, $support(x_i)$, is the set of pure strategies $\pi_{i,j}$ such that $x_i(j) > 0$.

Proposition Every dominant strategy x_i is in fact a "weighted average" of pure dominant strategies. I.e., each $\pi_{i,j} \in support(x_i)$ is also dominant.

if dominant strategy exists, the pure dominant strategy also exist

Moreover, only a pure strategy can be strictly dominant.

Proof Again, easy "weighted average" argument:

$$U_i(x_{-i}; x_i) = \sum_{j=1}^{m_i} x_i(j) * U_i(x_{-i}; \pi_{i,j}).$$

If x_i is dominant, then for any x_{-i} $U_i(x_{-i}; x_i) \geq U_i(x_{-i}; \pi_{i,j})$, for all j. But then if $x_i(j) > 0$, $U_i(x_{-i}; x_i) = U_i(x_{-i}; \pi_{i,j})$.

If x_i is strictly dominant, it must clearly be equal to the unique pure strategy in its support.

So, easy algorithm to find dominant strategies

- For each player i and each pure strategy $s_j \in S_i$,
  - Check if, for all pure combinations $s \in S = S_1 \times \ldots S_n$, $u_i(s_{-i}; s_j) \geq u_i(s)$.
  - If this is so for all s, output "s_j is a dominant strategy for player i".
- If no such pure strategy found, then there are no dominant strategies.

Same easy algorithm for a strictly dominant strategy.
But there may be no dominant strategies. . ..

obviously bad: strictly dominated strategies

Definition We say a strategy $x_i \in X_i$ is **strictly dominated** if there exists another strategy x_i' such that $x_i' \succ x_i$. We say x_i is **weakly dominated** if there exists x_i' such that $x_i' \succeq x_i$ and for some $x_{-i} \in X_{-i}$, $U_i(x_{-i}; x_i') > U_i(x_{-i}; x_i)$.

Clearly, strictly dominated strategies are "*bad*": "rational" players would be stupid to play them.

Weakly dominated strategies aren't necessarily as "bad". It depends on what you think others will play. In particular, there can be Nash Equilibria where everybody is playing a weakly dominated strategy:

$$\begin{bmatrix} (0,0) & (0,0) \\ (0,0) & (1,1) \end{bmatrix}$$

Question How can we compute whether a strategy is (strictly) dominated?

Example Consider the following table, showing only Player 1's payoffs: Is the last row strictly dominated?

yes, if u play rows 1-3 with probability 1/3?

$$\begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \\ 5 & 5 & 5 \end{bmatrix}$$

pure strategy (0,0,0,1) is strictly dominated by (1/3,1/3,1/3,0)

finding strictly dominated strategies via LP

Goal: Determine if $x_i \in X_i$ is (strictly) dominated.

To do this, we can use an LP with strict inequalities.
For each pure "counter profile" π_{-i}, we add a constraint $C_{\pi_{-i}}(x_i'(1), \ldots, x_i'(n))$, given by:

$$U_i(\pi_{-i}; x_i') > U_i(\pi_{-i}; x_i)$$

Note that this is a linear constraint: the right hand side is a constant we can compute, and the left hand side is linear in the variables $x_i'(1), \ldots, x_i'(n)$.
We also add the constraints $x_i'(1) + \ldots + x_i'(n) = 1$, and $x_i'(j) \geq 0$, for $j = 1, \ldots, n$.
x_i is strictly dominated iff this "strict LP" is feasible.

Question: But how do we cope with strict inequalities?

Coping with strict inequalities when checking feasibility of LP constraints

- Introduce a new variable $y \geq 0$, to be `Maximized`, and change constraints to:

$$U_i(\pi_{-i}; x_i') \geq U_i(\pi_i; x_i) + y$$

- Then x_i is strictly dominated if and only if the new LP (with objective "`Maximize` y") is feasible and the optimal value for y is > 0 (or unbounded, but in this particular example that can't happen).
- Observe: Any optimal solution x_i' to this revised LP is itself not strictly dominated.
- **Note:** This provides a general recipe for converting the problem of checking *feasibility* of any set of linear constraints *including strict inequalities*, to a new LP optimization problem, *without* strict inequalities.

common knowledge and strategy elimination

- Recall the games "Guess Half the Average", and "Give a (matched) dollar to the other player".
- How do we reason about such games? Suppose I "know" that all players are "*rational*" (i.e., aim to optimize their own expected payoff). Then I might conclude: "Jane will never play a strictly dominated (SD) strategy. So I can eliminate her SD strategies from consideration." But by eliminating her SDSs, some of my strategies may become SD'ed! Deepening the reasoning, suppose

  "I know that she knows that I know that . . . . . ".

- **Definition (somewhat informal):** A fact P is "**common knowledge**" among all n players if:
  - For every player i, "Player i knows P": call this fact $P_{\langle i \rangle}$.
  - And, inductively, for $k \geq 1$, for all players i, and all sequences $s = i_1 \dots i_k \in \{1, \dots, n\}^k$, "Player i knows $P_{\langle s \rangle}$": call this fact $P_{\langle i\, s \rangle}$.

  (To be more formal, we would have to delve deeper into "*logics of knowledge*". Outside the scope of this class. See, e.g., the book [Fagin-Halpern-Moses-Vardi'95].)

- **RKN hypothesis:** every player's "rationality" is common knowledge among all players.

iterated SDS elimination algorithm

Assuming the RKN hypothesis, we can safely conduct the following strategy elimination algorithm:

- **While** (some pure strategy $\pi_{i,j}$ is SD'ed)
  eliminate $\pi_{i,j}$ from the game,
  obtaining a new residual game;

Sometimes, a player's strategy may be uniquely determined by the end of elimination, but certainly not always:

$$\begin{bmatrix} (0,7) & (2,5) & (7,0) & (0,1) \\ (5,2) & (3,3) & (5,2) & (0,1) \\ (7,0) & (2,5) & (0,7) & (0,1) \\ (0,0) & (0,-2) & (0,0) & (10,-1) \end{bmatrix}$$

Note: we iteratively eliminate only pure SDSs. There may in fact remain mixed SDSs. Before playing any mixed strategy in the residual game we should make sure it is not SD'ed (by, e.g., checking it is an optimal solution of appropriate LP).

remarks

- There is a more general notion of **rationalizability** ([Bernheim'84,Pierce84]), which says:
  - A rational player i should never play a strategy x_i which is "never a best response" to any counter strategy x_{-i} (see below).
  - Assuming rationality is common knowledge, we should also iteratively eliminate all strategies that are "never a best response".
- It turns out, for 2-player games, this elimination yields exactly the same residual game as iterated SDS elimination. So the same algorithm applies.
- For > 2 players, things get more complicated: this equivalence doesn't hold unless we adopt a different notion of "never a best response" (with respect to any "*belief*" of player i about other players' strategies, ... we will not consider this further.)

weakly vs. strictly dominated strategies

- Note: We did not eliminate weakly dominated strategies.
- In fact, the residual game obtained from iterated WDS elimination depends on the order of elimination:

$$\begin{bmatrix} (5,1) & (4,0) \\ (6,0) & (3,1) \\ (6,4) & (4,4) \end{bmatrix}$$

- This problem does not arise for strictly dominated strategies:
  Proposition Iterated elimination of strictly dominated strategies produces the same final residual game regardless of the order in which strategies are eliminated.
  Proof If a pure strategy is strictly dominated, it will remain strictly dominated even after another strictly dominated pure strategy is removed. □

Computing Nash Equilibria: a first clue

Recall "Useful corollary for NEs", from Lecture 3:

> *If x^* is an NE and $x_i^*(j) > 0$ then*
> $U_i(x_{-i}^*; \pi_{i,j}) = U_i(x^*)$.

Using this, we can fully characterize Nash Equilibria:

Proposition In an n-player game, a profile x^* is a Nash Equilibrium if and only if there exist $w_1, \ldots, w_n \in \mathbb{R}$, such that the following hold:

1. For all players i, and every $\pi_{i,j} \in \mathit{support}(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) = w_i$, and
2. For all players i, and every $\pi_{i,j} \notin \mathit{support}(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) \leq w_i$.

Note: such w_i's necessarily satisfy $w_i = U_i(x^*)$.

Proof Easy from what we already know. □

Food for thought: Can you use this to find a NE?

Algorithmic Game Theory and Applications

Lecture 9: Computing solutions for General Strategic Games: Part II: Nash Equilibria

Kousha Etessami

recall: Computing Nash Equilibria: a first clue

Recall "Useful corollary for NEs", from Lecture 3:

If x^* is an NE, then if $x_i^*(j) > 0$ then $U_i(x_{-i}^*; \pi_{i,j}) = U_i(x^*)$.

Using this, we can fully characterize NEs:

Proposition 1 In an n-player game, profile x^* is a NE if and only if there exist $w_1, \ldots, w_n \in \mathbb{R}$, such that:

1. $\forall$ players i, & $\forall\, \pi_{i,j} \in support(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) = w_i$, &
2. $\forall$ players i, & $\forall\, \pi_{i,j} \notin support(x_i^*)$, $U_i(x_{-i}^*; \pi_{i,j}) \leq w_i$.

Note: Any such w_i's necessarily satisfy $w_i = U_i(x^*)$.

Proof Follows easily from what we already know, particularly 1st claim in the proof of Nash's theorem. □

using our first clue

- Suppose we somehow know support sets, $support_1 \subseteq S_1, \ldots, support_n \subseteq S_n$, for some Nash Equilibrium $x^* = (x_1^*, \ldots, x_n^*)$.
- Then, using Proposition 1, to find a NE we only need to solve the following system of constraints:
  1. $\forall$ players i, & $\forall\ \pi_{i,j} \in support_i$, $U_i(x_{-i}; \pi_{i,j}) = w_i$,
  2. $\forall$ players i, & $\forall\ \pi_{i,j} \notin support_i$, $U_i(x_{-i}; \pi_{i,j}) \leq w_i$.
  3. $\forall$ players $i = 1, \ldots, n$, $x_i(1) + \ldots + x_i(m_i) = 1$.
  4. $\forall players i = 1,\ldots,n, \& for j \in support_i$, $x_i(j) \geq 0$.
  5. for $i = 1, \ldots, n$, & for $j \notin support_i$, $x_i(j) = 0$.
- This system has $\sum_{i=1}^{n} m_i + n$ variables, $x_1(1), \ldots, x_1(m_1), \ldots, x_n(1), \ldots, x_n(m_n), w_1, \ldots, w_n$.
- Unfortunately, for $n > 2$ players, this is a non-linear system of constraints.

Let's come back to the case $n > 2$ players later.

two-player case

- In the 2-player case, the system of constraints is an LP!! But:

  Question: How do we find $support_1$ & $support_2$?

two-player case

- In the 2-player case, the system of constraints is an LP!! But:

  Question: How do we find $support_1$ & $support_2$?

  Answer: Just guess!!

First algorithm to find NE's in 2-player games

Input: A 2-player strategic game Γ, given by rational values $u_1(s, s')$ & $u_2(s, s')$, for all $s \in S_1$ & $s' \in S_2$. (I.e., the input is $(2 \cdot m_1 \cdot m_2)$ rational numbers.)

Algorithm:

- For all possible $support_1 \subseteq S_1$ & $support_2 \subseteq S_2$:
  - Check if the corresponding LP has a feasible solution $x^*, w_1, \ldots, w_n$. (using, e.g., Simplex).
  - If so, STOP: the feasible solution x^* is a Nash Equilibrium (and $w_i = U_i(x^*)$).

Question: How many possible subsets $support_1$ and $support_2$ are there to try?

First algorithm to find NE's in 2-player games

Input: A 2-player strategic game Γ, given by rational values $u_1(s, s')$ & $u_2(s, s')$, for all $s \in S_1$ & $s' \in S_2$. (I.e., the input is $(2 \cdot m_1 \cdot m_2)$ rational numbers.)

Algorithm:

- For all possible $support_1 \subseteq S_1$ & $support_2 \subseteq S_2$:
  - Check if the corresponding LP has a feasible solution $x^*, w_1, \ldots, w_n$. (using, e.g., Simplex).
  - If so, STOP: the feasible solution x^* is a Nash Equilibrium (and $w_i = U_i(x^*)$).

Question: How many possible subsets $support_1$ and $support_2$ are there to try?

Answer: $2^{(m_1+m_2)}$

So, unfortunately, the algorithm requires worst-case exponential time.

But, at least we have our first algorithm.

remarks on algorithm 1

- The algorithm immediately yields:
  Proposition Every finite 2-player game has a rational NE. (Furthermore, the rational numbers are not "too big", i.e., are polynomial sized.)
- The algorithm can easily be adapted to find not just any NE, but a "good" one. For example:
  Finding a NE that maximizes "(util.) social welfare":
  - For each support sets, simply solve the LP constraints while maximizing the objective
    $$f(x, w) = w_1 + w_2 + \ldots + w_n$$
  - Keep track of best NE encountered, & output optimal NE after checking all support sets.

- The same algorithm works for any notion of "good" NE that can be expressed via a linear objective and (additional) linear contraints: (e.g.: maximize Jane's payoff, minimize John's, etc.)
- Note: This algorithm shows that finding a NE for 2-player games is in "**NP**".

Towards another algorithm for 2-players

Let A be the $(m_1 \times m_2)$ payoff matrix for player 1,
B be the $(m_2 \times m_1)$ payoff matrix for player 2,
$\mathbf{w_1}$ be the m_1-vector, all entries $= w_1$,
$\mathbf{w_2}$ be the m_2-vector, all entries $= w_2$.

Note: We can safely assume $A > 0$ and $B > 0$: by adding a large enough constant, d, to every entry we "shift" each matrix > 0. Nothing essential about the game changes: payoffs just increase by d.

We can get another, related, characterization of NE's by using "slack variables" as follows:

Lemma $x^* = (x_1^*, x_2^*)$ is a NE if and only if:

1. There exists a m_1-vector $y \geq 0$, and $w_1 \in \mathbb{R}$, such that

$$Ax_2^* + y = \mathbf{w_1}$$

& for all $j = 1, \ldots, m_1$, $x_1^*(j) = 0$ or $(y)_j = 0$.

2. There exists a m_2-vector $z \geq 0$, and $w_2 \in \mathbb{R}$, such that

$$Bx_1^* + z = \mathbf{w_2}$$

& for all $j = 1, \ldots, m_2$, $x_2^*(j) = 0$ or $(z)_j = 0$.

Proof Again follows by the Useful Corollary to Nash: in a NE x^* whenever, e.g., $x_1^*(j) > 0$, $U(x_{-1}^*; \pi_{1,j}) = U(x^*)$. Let $(y)_j = U(x^*) - U(x_{-1}^*; \pi_{1,j})$. □

rephrasing the problem

The Lemma gives us some "constraints" that characterize NE's:

1. $Ax_2 + y = \mathbf{w_1}$ and $Bx_1 + z = \mathbf{w_2}$
2. $x_1, x_2, y, z \geq 0$.
3. x_1 and x_2 must be probability distributions, i.e., $\sum_{j=1}^{m_1} x_1(j) = 1$ and $\sum_{j=1}^{m_2} x_2(j) = 1$.
4. Additionally, x_1 and y, as well as x_2 and z, need to be "**complementary**":
   for $j = 1, \ldots, m_1$, either $x_1(j) = 0$ or $(y)_j = 0$,
   for $j = 1, \ldots, m_2$, either $x_2(j) = 0$ or $(z)_j = 0$.

   Since everything is ≥ 0, we can write this as

$$y^T x_1 = 0 \quad \text{and} \quad z^T x_2 = 0$$

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let $\mathbf{1}$ denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = \mathbf{1} \quad \text{and} \quad Bx_1' + z' = \mathbf{1}$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let $\mathbf{1}$ denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = \mathbf{1} \quad \text{and} \quad Bx_1' + z' = \mathbf{1}$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

If, in addition, $x_1' \neq 0$ or $x_2' \neq 0$, then, by complementarity both $x_1' \neq 0$ and $x_2' \neq 0$.

continuing the reformulation

Note that, because $A > 0$ and $B > 0$, we know that $w_1 > 0$ and $w_2 > 0$ in any solution.

Using this, we can "eliminate" w_1 and w_2 from the constraints as follows: Let $x_2' = (1/w_1)x_2$, $y' = (1/w_1)y$, $x_1' = (1/w_2)x_1$, and $z' = (1/w_2)z$.

Let $\mathbf{1}$ denote an all 1 vector (of appropriate dimension).

Suppose we find a solution to

$$Ax_2' + y' = \mathbf{1} \quad \text{and} \quad Bx_1' + z' = \mathbf{1}$$

$x_1', x_2', y', z' \geq 0$, $(y')^T x_1' = 0$, and $(z')^T x_2' = 0$.

If, in addition, $x_1' \neq 0$ or $x_2' \neq 0$, then, by complementarity both $x_1' \neq 0$ and $x_2' \neq 0$.

In this case we can "recover" a solution x_1, x_2, y, z, and w_1 and w_2 to the original constraints, by multiplying x_1' and x_2' by "normalizing" constants w_1 and w_2, so that each of $x_1 = w_1 x_1'$ and $x_2 = w_2 x_2'$ define probability distributions. These normalizing constants define w_1 and w_2 in our solution.

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$. rephrasing from x@y=0

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

Our Actual Goal: is to find a solution where $u \neq 0$.

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = \mathbf{1}$" look familiar??

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = \mathbf{1}$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = \mathbf{1}$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

Question: How do we move from this "*complementary basis*" to one where $u \neq 0$?

2-player NE's as Linear Complementarity Problem

Let $$M = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad u = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} \quad v = \begin{bmatrix} y' \\ z' \end{bmatrix}$$

"**Our Goal:**" Find a solution u, v, to

$$Mu + v = \mathbf{1}$$

such that $u, v \geq 0$, and $u^T v = 0$.

This is an intance of a Linear Complementarity Problem, a classic problem in mathematical programming (see, e.g., the book [Cottle-Pang-Stone'92]).

But, we already know one solution: $u = 0$, $v = \mathbf{1}$.

Our Actual Goal: is to find a solution where $u \neq 0$.

Wait! Doesn't "$Mu + v = \mathbf{1}$" look familiar??

Sure! It's just a "Feasible Dictionary" (from lect. 6 on Simplex), with "Basis" the variables in vector v.

Question: How do we move from this "*complementary basis*" to one where $u \neq 0$?

Answer: Pivoting!! (in a very selective way)

sketch of the Lemke-Howson Algorithm

1) Start at the "extra" "complementary Basis" $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = \mathbf{1}$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).

sketch of the Lemke-Howson Algorithm

1) Start at the "extra" "complementary Basis" $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = \mathbf{1}$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).

2) For some i, move via pivoting to a "neighboring" "i-almost complementary" basis β'. β' is **i-almost complementary** if for $k \in \{1, \ldots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.

sketch of the Lemke-Howson Algorithm

1) Start at the "extra" "complementary Basis" $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = \mathbf{1}$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).

2) For some i, move via pivoting to a "neighboring" "i-almost complementary" basis β'. β' is **i-almost complementary** if for $k \in \{1, \ldots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.

3) While (new basis isn't actually complementary)

- There's a unique j, such that both $(u)_j$ & $(v)_j$ are not in the new basis: one was just kicked out of the basis.
- If $(u)_j$ was just kicked out, move $(v)_j$ into the basis by pivoting. If $(v)_j$ was just kicked, move $(u)_j$ in. (Selective pivot rules ensure only one possible entering/leaving pair.)
- Newest basis is also i-almost complementary.

sketch of the Lemke-Howson Algorithm

1) Start at the "extra" "complementary Basis" $\beta = \{(v)_1, \ldots, (v)_m\}$, where $m = m_1 + m_2$ (with BFS $u = 0, v = \mathbf{1}$). β is **complementary** if for $k \in \{1, \ldots, m\}$, either $(u)_k \notin \beta$ or $(v)_k \notin \beta$ (but not both, since $|\beta| = m$).

2) For some i, move via pivoting to a "neighboring" "i-almost complementary" basis β'. β' is **i-almost complementary** if for $k \in \{1, \ldots, m\} \setminus \{i\}$, $(u)_k \notin \beta'$ or $(v)_k \notin \beta'$.

3) While (new basis isn't actually complementary)

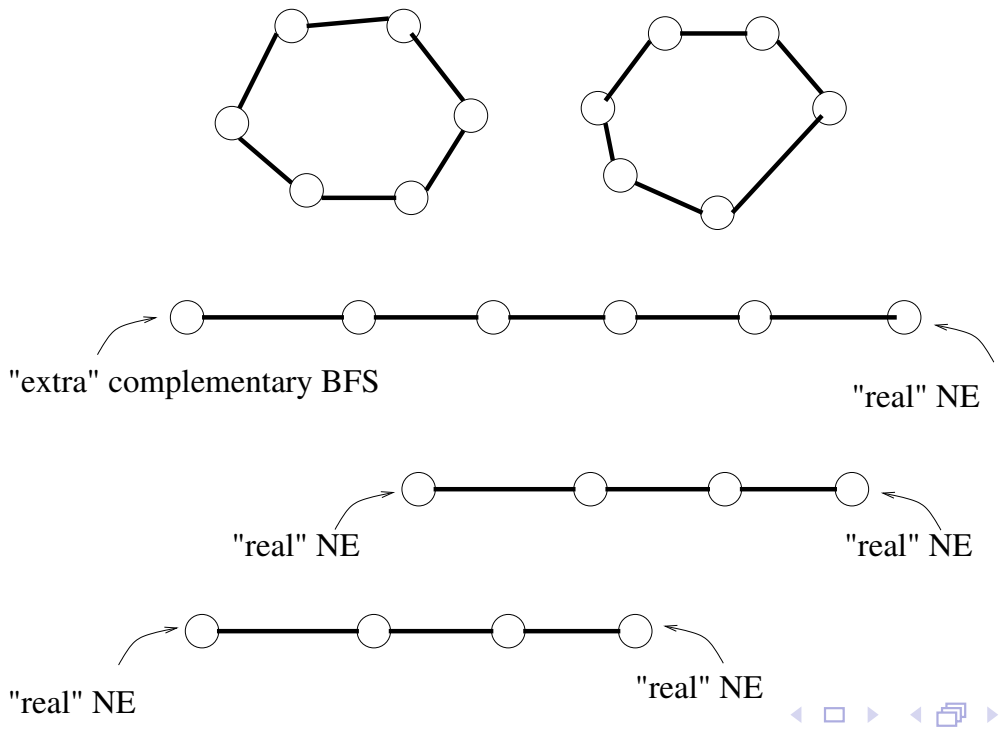
- There's a unique j, such that both $(u)_j$ & $(v)_j$ are not in the new basis: one was just kicked out of the basis.
- If $(u)_j$ was just kicked out, move $(v)_j$ into the basis by pivoting. If $(v)_j$ was just kicked, move $(u)_j$ in. (Selective pivot rules ensure only one possible entering/leaving pair.)
- Newest basis is also i-almost complementary.

4) STOP: we've reached a different complementary basis & BFS. A NE is obtained by "normalizing" $u = [x_1' \; x_2']^T$.

We are skipping lots of details related to "degeneracy", etc. (similar to complications that arose in Simplex pivoting).

Question Why should this work?

A key reason: With appropriately selective pivoting rules, each i-almost complementary Basis ("vertex") has 2 neighboring "vertices" unless it is actually a complementary Basis, in which case it has 1. This assures that starting at the "extra" complementary BFS, we will end up at "the other end of the line". Let's see it in pictures:



remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

▷ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots,
for any permissible pivots (see [Savani-von Stengel'03]).

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

▷ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots, for any permissible pivots (see [Savani-von Stengel'03]).

▷ Is there a polynomial time algorithm to find a NE in 2-player games? This is an *open problem*!

remarks

▷ The Lemke-Howson (1964) algorithm has a "geometric" interpretation. (See, [von Stengel, Chapter 3, in Nisan et. al. AGT book, 2007]. Our treatment is closer to [McKelvey-McLennan'96], see course web page.)

▷ The algorithm's correctness gives another proof of Nash's theorem *for 2-player games only*, just like Simplex's gives another proof of Minimax (via LP-duality).

▷ How fast is the LH-algorithm? Unfortunately, examples exist requiring exponentially many pivots, for any permissible pivots (see [Savani-von Stengel'03]).

▷ Is there a polynomial time algorithm to find a NE in 2-player games? This is an *open problem*!

▷ However, finding "good" NE's that, e.g., maximize "social welfare" is NP-hard. Even knowing whether there is > 1 NE is NP-hard. ([Gilboa-Zemel'89], [Conitzer-Sandholm'03]).

games with > 2 players

▷ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is irrational.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

irrational numbers, ranther than irrational players

games with > 2 players

$\triangleright$ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is irrational.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

$\triangleright$ We can try to **approximate** NEs. But there are different notions of approximate NE:
Definition 1: A mixed strategy profile x is called a ϵ-**Nash Equilibrium**, for some $\epsilon > 0$, if $\forall\ i$, and all mixed strategies y_i: $U_i(x) \geq U_i(x_{-i}; y_i) - \epsilon$.
I.e.: *No player can increase its own payoff by more than ϵ by unilaterally switching its strategy.*

games with > 2 players

▷ Nash himself (1951, page 294) gives a 3 player "poker" game where the only NE is irrational.
So, it isn't so sensible to speak of computing an "exact" NE when the number of players is > 2.

▷ We can try to **approximate** NEs. But there are different notions of approximate NE:
Definition 1: A mixed strategy profile x is called a ϵ-**Nash Equilibrium**, for some $\epsilon > 0$, if $\forall\ i$, and all mixed strategies y_i: $U_i(x) \geq U_i(x_{-i}; y_i) - \epsilon$.
I.e.: *No player can increase its own payoff by more than ϵ by unilaterally switching its strategy.*
Definition 2: A mixed strategy profile x is ϵ-**close** to an actual **NE**, for some $\epsilon > 0$, if there is an actual NE x^*, such that $\|x - x^*\|_\infty \leq \epsilon$, i.e., $|x_i^*(j) - x_i(j)| < \epsilon$ for all i, j.

▷ It turns out these different notions of approximation of an NE have very different computation complexity implications.

What is the complexity of computing an ϵ-NE?

$\triangleright$ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1*."*.

What is the complexity of computing an ϵ-NE?

▷ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1*."*.

▷ [Papadimitriou 1992], defined a complexity class called **PPAD** to capture such problems, where ANOTHER-LINE-ENDPOINT is PPAD-complete.
He took inspiration from ideas in Lemke-Howson algorithm and an algorithm by [Scarf'67] for computing *almost* fixed points.

What is the complexity of computing an ϵ-NE?

▷ It turns out that:
(A) computing an NE for 2-player games, and
(B) computing an ϵ-NE for $>$ 2-player games
are reducible to each other.
Both are at least as hard as ANOTHER-LINE-ENDPOINT:
"Find another end-point of a succinctly given (directed) line graph, with indegree and outdegree ≤ 1*."*.

▷ [Papadimitriou 1992], defined a complexity class called **PPAD** to capture such problems, where ANOTHER-LINE-ENDPOINT is PPAD-complete.
He took inspiration from ideas in Lemke-Howson algorithm and an algorithm by [Scarf'67] for computing *almost* fixed points.

▷ [Chen-Deng'06] & [Daskalakis-Goldberg-Papadimitriou,'06], showed that computing an NE in 2-player games, & computing a ϵ-NE in $>$ 2-player games, respectively, are PPAD-complete.

▷ What about ϵ-close approximating an actual NE??

The complexity of computing an actual NE in games with > 2 players

$\triangleright$ For games with > 2 players, approximating an actual NE, i.e., computing a profile ϵ-close to an actual NE, even for any $\epsilon < 1/2$, is MUCH harder. Not even known to be in **NP**. The best complexity upper bound we know is **PSPACE** (using deep but impractical algorithms for solving nonlinear systems of equations [Gregoriev-Vorobjov'88,Canny'88,Renegar'92]).

$\triangleright$ [Etessami-Yannakakis'07] showed that if we can approximate an actual NE even in NP, that would resolve major open problems in the complexity of numerical analysis (seems unlikely at present). They showed computing or approximating an actual NE is **FIXP**-complete, where FIXP consists of all problems reducible to computing a fixed point for algebraic Brouwer functions defined by operators $\{+, *, -, /, \max, \min\}$ and rational constants.

▷ Such fixed point computation problems have many other important applications, in particular, for computation of **market equilibria**.

▷ In turns out that PPAD is exactly the *"piecewise linear"* fragment of FIXP, consisting of problems reducible to Brouwer fixed point problems defined by algebraic functions using operators $\{+, -, \max, \min\}$.

▷ These results are beyond the scope of this course.
If you are interested to learn more, see:

K. Etessami and M. Yannakakis, "On the Complexity of Nash Equilibria and other Fixed Points", *SIAM Journal on Computing*, 39(2), pp. 2531-2597, 2010.

and the references therein.

non-examinate

Algorithmic Game Theory and Applications

Lecture 10: Games in Extensive Form

Kousha Etessami

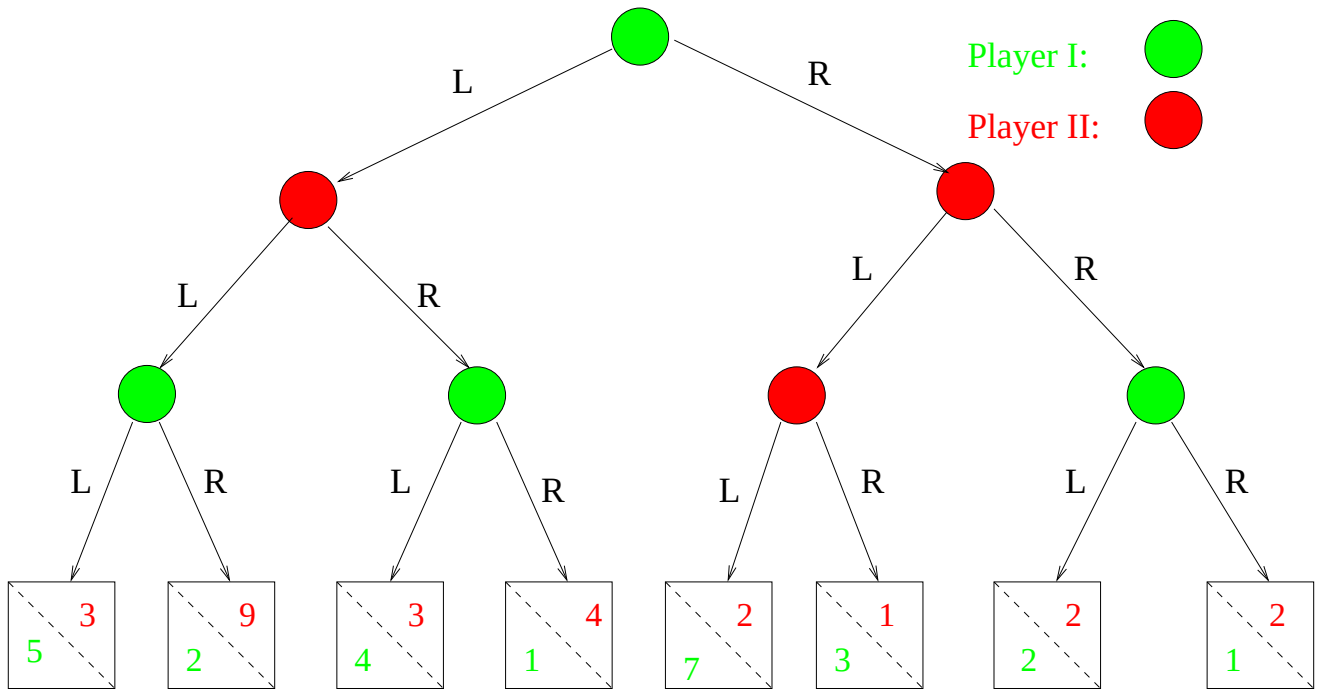
the setting and motivation

▷ Most games in "real life" are not in "strategic form": players don't pick their entire strategies independently ("simultaneously"). Instead, the game transpires over time, with players making "moves" to which other players react with "moves", etc. Examples: chess, poker, bargaining, dating, ...

the setting and motivation

▷ Most games in "real life" are not in "strategic form": players don't pick their entire strategies independently ("simultaneously"). Instead, the game transpires over time, with players making "moves" to which other players react with "moves", etc. Examples: chess, poker, bargaining, dating, ...

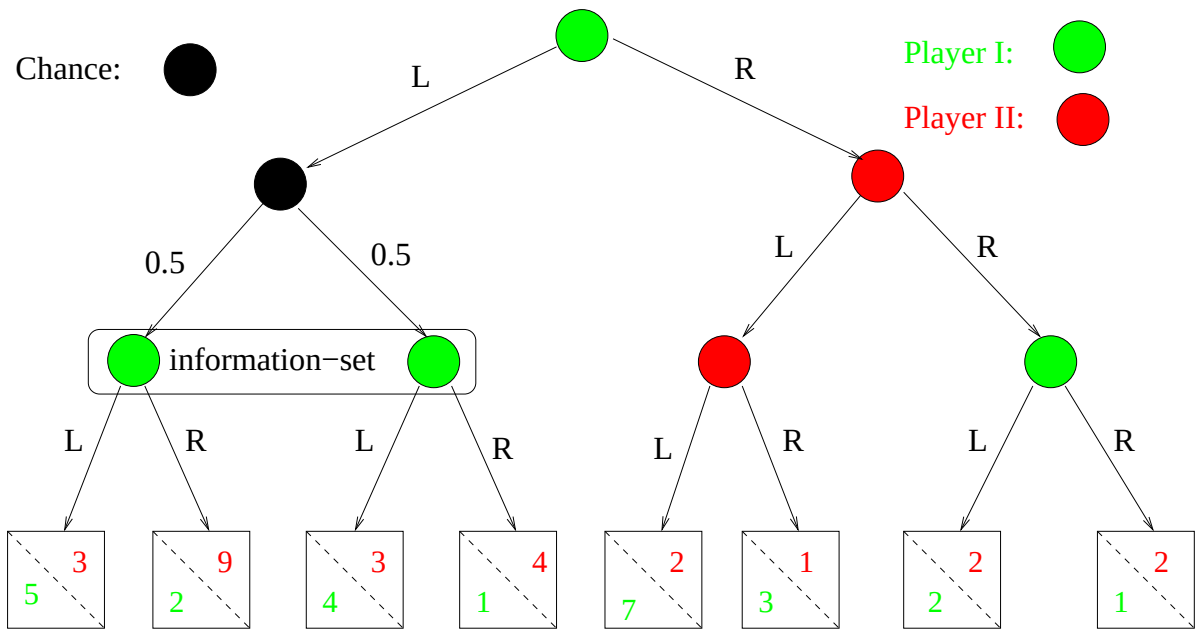
▷ A "game tree" looks something like this:



▷ But we may also need some other "features".

chance, information, etc.

Some tree nodes may be *chance* (probabilistic) nodes, controlled by no player (by "nature"). (Poker, Backgammon.)
Also, a player may not be able to distinguish between several of its "positions" or "nodes", because not all *information* is available to it. (Think Poker, with opponent's cards hidden.)
Whatever move a player employs at a node must be employed at all nodes in the same "information set".



To define extensive form games, we have to formalize all these.

Trees: a formal definition

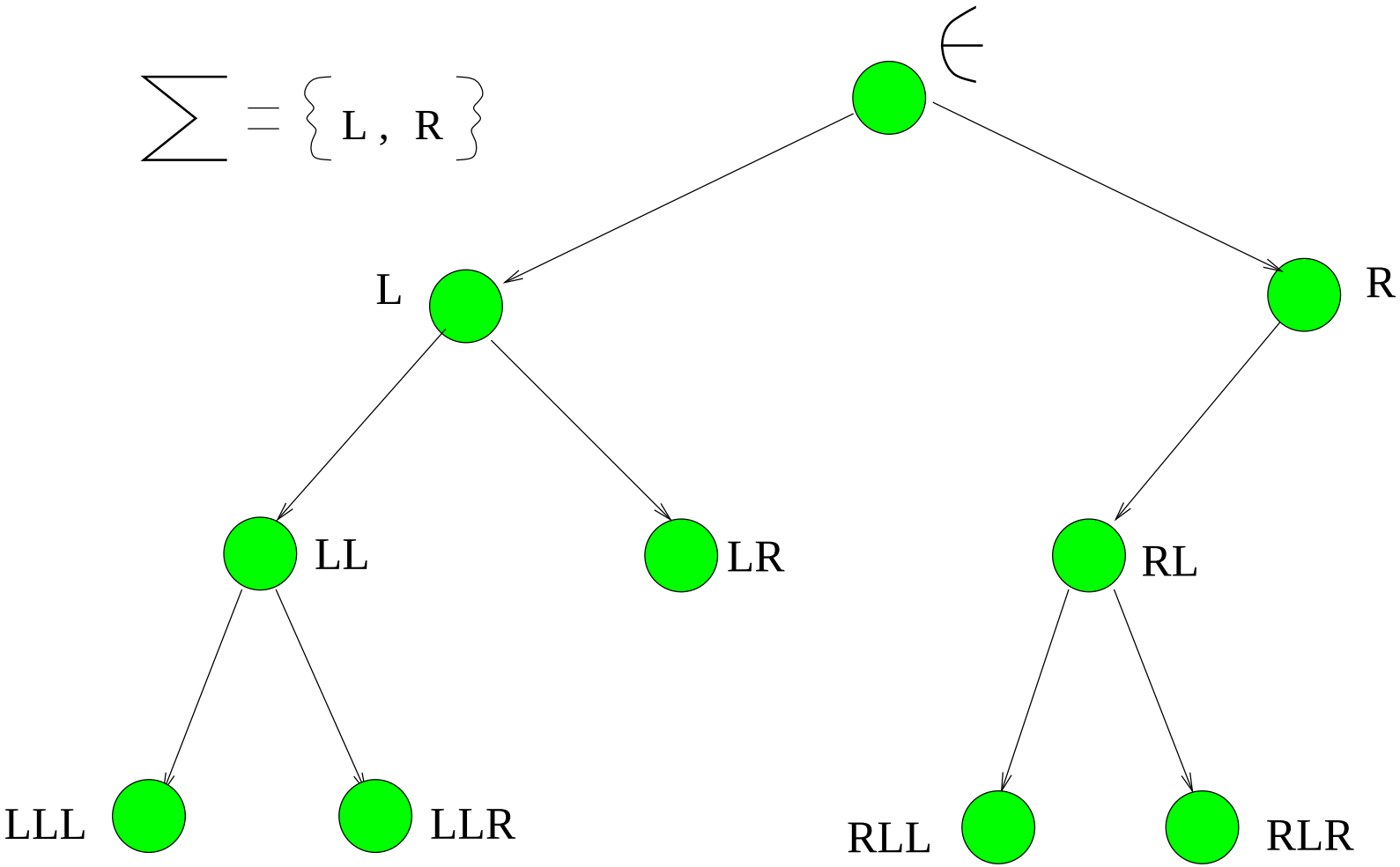
▷ Let $\Sigma = \{a_1, a_2, \ldots, a_k\}$ be an alphabet. A tree over Σ is a set $T \subseteq \Sigma^*$, of nodes $w \in \Sigma^*$ such that: if $w = w'a \in T$, then $w' \in T$. (I.e., it is a prefix-closed subset of Σ^*.)

▷ For a node $w \in T$, the children of w are $ch(w) = \{w' \in T \mid w' = wa \text{ , for some } a \in \Sigma\}$. For $w \in T$, let $Act(w) = \{a \in \Sigma \mid wa \in T\}$ be "actions" available at w.

▷ A leaf (or terminal) node $w \in T$ is one where $ch(w) = \emptyset$. Let $L_T = \{w \in T \mid w \text{ a leaf}\}$.

▷ A (finite or infinite) path π in T is a sequence $\pi = w_0, w_1, w_2, \ldots$ of nodes $w_i \in T$, where if $w_{i+1} \in T$ then $w_{i+1} = w_i a$, for some $a \in \Sigma$. It is a complete path (or play) if $w_0 = \epsilon$ and every non-leaf node in π has a child in π. Let Ψ_T denote the set of plays of T.

Tree example



games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.

games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. A tree T, called the game tree, over some Σ.

games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. A tree T, called the game tree, over some Σ.
3. A partition of the tree nodes T into disjoint sets $Pl_0, Pl_1, \ldots, Pl_n$, such that $T = \bigcup_{i=0}^{n} Pl_i$. Where Pl_i, $i \geq 1$, denotes the nodes of player i, where it is player i's turn to move. (And Pl_0 denotes the "chance"/"nature" nodes.)

games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. A tree T, called the game tree, over some Σ.
3. A partition of the tree nodes T into disjoint sets $Pl_0, Pl_1, \ldots, Pl_n$, such that $T = \bigcup_{i=0}^{n} Pl_i$. Where Pl_i, $i \geq 1$, denotes the nodes of player i, where it is player i's turn to move. (And Pl_0 denotes the "chance"/"nature" nodes.)
4. For each "nature" node, $w \in Pl_0$, a probability distribution $q_w : Act(w) \mapsto \mathbb{R}$ over its actions: $q_w(a) \geq 0$, & $\sum_{a \in Act(w)} q_w(a) = 1$.

games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. A tree T, called the game tree, over some Σ.
3. A partition of the tree nodes T into disjoint sets $Pl_0, Pl_1, \ldots, Pl_n$, such that $T = \bigcup_{i=0}^{n} Pl_i$. Where Pl_i, $i \geq 1$, denotes the nodes of player i, where it is player i's turn to move. (And Pl_0 denotes the "chance"/"nature" nodes.)
4. For each "nature" node, $w \in Pl_0$, a probability distribution $q_w : Act(w) \mapsto \mathbb{R}$ over its actions: $q_w(a) \geq 0$, & $\sum_{a \in Act(w)} q_w(a) = 1$.
5. For each player $i \geq 1$, a partition of Pl_i into disjoint non-empty information sets $Info_{i,1}, \ldots, Info_{i,r_i}$, such that $Pl_i = \bigcup_{j=0}^{r_i} Info_{i,j}$. Moreover, for any i, j, & all nodes $w, w' \in Info_{i,j}$, $Act(w) = Act(w')$. (I.e., the set of possible "actions" from all nodes in one information set is the same.)

games in extensive form

A **Game in Extensive Form**, $\mathcal{G}$, consists of:

1. A set $N = \{1, \ldots, n\}$ of players.
2. A tree T, called the game tree, over some Σ.
3. A partition of the tree nodes T into disjoint sets $Pl_0, Pl_1, \ldots, Pl_n$, such that $T = \bigcup_{i=0}^{n} Pl_i$. Where Pl_i, $i \geq 1$, denotes the nodes of player i, where it is player i's turn to move. (And Pl_0 denotes the "chance" /"nature" nodes.)
4. For each "nature" node, $w \in Pl_0$, a probability distribution $q_w : Act(w) \mapsto \mathbb{R}$ over its actions: $q_w(a) \geq 0$, & $\sum_{a \in Act(w)} q_w(a) = 1$.
5. For each player $i \geq 1$, a partition of Pl_i into disjoint non-empty information sets $Info_{i,1}, \ldots, Info_{i,r_i}$, such that $Pl_i = \bigcup_{j=0}^{r_i} Info_{i,j}$. Moreover, for any i, j, & all nodes $w, w' \in Info_{i,j}$, $Act(w) = Act(w')$. (I.e., the set of possible "actions" from all nodes in one information set is the same.)
6. For each player i, a function $u_i : \Psi_T \mapsto \mathbb{R}$, from (complete) plays to the payoff for player i.

explanation and comments

▷ Question: Why associate payoffs to "plays" rather than to leaves at the "end" of play?

explanation and comments

▷ Question: Why associate payoffs to "plays" rather than to leaves at the "end" of play?
Answer: We in general allow infinite trees. We will later consider "infinite horizon" games in which play can go on for ever. Payoffs are then determined by the entire history of play. For "finite horizon" games, where tree T is finite, it suffices to associate payoffs to the leaves, i.e., $u_i : L_T \mapsto \mathbb{R}$.

explanation and comments

▷ Question: Why associate payoffs to "plays" rather than to leaves at the "end" of play?
Answer: We in general allow infinite trees. We will later consider "infinite horizon" games in which play can go on for ever. Payoffs are then determined by the entire history of play. For "finite horizon" games, where tree T is finite, it suffices to associate payoffs to the leaves, i.e., $u_i : L_T \mapsto \mathbb{R}$.

▷ We defined our alphabet of possible actions Σ to be finite, which is generally sufficient for our purposes. In other words, the tree is finitely branching. In more general settings, even the set of possible actions at a given node can be infinite.

explanation and comments

▷ Question: Why associate payoffs to "plays" rather than to leaves at the "end" of play?
Answer: We in general allow infinite trees. We will later consider "infinite horizon" games in which play can go on for ever. Payoffs are then determined by the entire history of play. For "finite horizon" games, where tree T is finite, it suffices to associate payoffs to the leaves, i.e., $u_i : L_T \mapsto \mathbb{R}$.

▷ We defined our alphabet of possible actions Σ to be finite, which is generally sufficient for our purposes. In other words, the tree is finitely branching. In more general settings, even the set of possible actions at a given node can be infinite.

▷ Later, we will focus on the following class of games:
Definition An extensive form game $\mathcal{G}$ is called a game of **perfect information**, if every information set $Info_{i,j}$ contains only 1 node.

pure strategies

▷ A pure strategy s_i for player i in an extensive game $\mathcal{G}$ is a function $s_i : Pl_i \mapsto \Sigma$ that assigns actions to each of player i's nodes, such that $s_i(w) \in Act(w)$, & such that if $w, w' \in Info_{i,j}$, then $s_i(w) = s_i(w')$.

Let S_i be the set of pure strategies for player i.

▷ Given pure profile $s = (s_1, \ldots, s_n) \in S_1 \times \ldots \times S_n$, if there are no chance nodes (i.e., $Pl_0 = \emptyset$) then s uniquely determines a play π_s of the game: players move according their strategies:

- Initialize $j := 0$, and $w_0 := \epsilon$;
- While (w_j is not at a terminal node)
  If $w_j \in Pl_i$, then $w_{j+1} := w_j\ s_i(w_j)$;
  $j := j + 1$;
- $\pi_s = w_0, w_1, \ldots$

▷ What if there are chance nodes?

pure strategies and chance

If there are chance nodes, then $s \in S$ determines a probability distribution over plays π of the game.

For finite extensive games, where T is finite, we can calculate the probability $p_s(\pi)$ of play π, using probabilities $q_w(a)$:

Suppose $\pi = w_0, \ldots, w_m$, is a play of T. Suppose further that for each $j < m$, if $w_j \in Pl_i$, then $w_{j+1} = w_j\ s_i(w_j)$. Otherwise, let $p_s(\pi) = 0$.

Let $w_{j_1}, \ldots, w_{j_r}$ be the chance nodes in π, and suppose, for each $k = 1, \ldots, r$, $w_{j_k+1} = w_{j_k} a_{j_k}$, i.e., the required action to get from node w_{j_k} to node w_{j_k+1} is a_{j_k}. Then

$$p_s(\pi) := \prod_{k=1}^{r} q_{w_{j_k}}(a_{j_k})$$

For infinite extensive games, defining these distributions in general requires much more elaborate definitions (proper "measure theoretic" probability). We will avoid the heavy stuff.

chance and expected payoffs

For a finite extensive game, given pure profile $s = (s_1, \ldots, s_n) \in S_1 \times \ldots \times S_n$, we can, define the "expected payoff" for player i under s as:

$$h_i(s) := \sum_{\pi \in \Psi_t} p_s(\pi) * u_i(\pi)$$

Again, for infinite games, much more elaborate definitions of "expected payoffs" would be required.

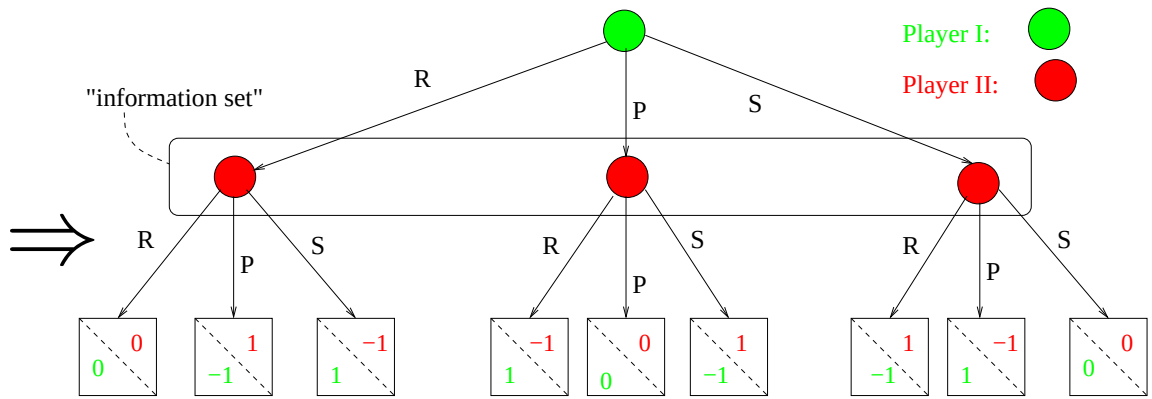
Note: This "expected payoff" does not arise because any player is mixing its strategies. It arises because the game itself contains randomness.

We can also combine both: players may also randomize amongst their strategies, and we could then define the overall expected payoff.

from strategic to extensive games

Every finite strategic game Γ can be encoded easily and concisely as an extensive game $\mathcal{G}_\Gamma$. We illustrate this via the Rock-Paper-Scissor 2-player game (the n-player case is an easy generalization):

Player I \ Player II	Rock	Paper	Scissors
Rock	0, 0	1, −1	−1, 1
Paper	−1, 1	0, 0	1, −1
Scissors	1, −1	−1, 1	0, 0



from extensive to strategic games

Every extensive game $\mathcal{G}$ can be viewed as a strategic game $\Gamma_{\mathcal{G}}$:
$\triangleright$ In $\Gamma_{\mathcal{G}}$, the strategies for player i are the mappings $s_i \in S_i$.
$\triangleright$ In $\Gamma_{\mathcal{G}}$, we define payoff $u_i(s) := h_i(s)$, for all pure profiles s.
If the extensive game $\mathcal{G}$ is finite, i.e., tree T is finite, then the strategic game $\Gamma_{\mathcal{G}}$ is also finite.
Thus, all the theory we developed for finite strategic games also applies to finite extensive games.
Unfortunately, the strategic game $\Gamma_{\mathcal{G}}$ is generally exponentially bigger than $\mathcal{G}$. Note that the number of pure strategies for a player i with $|Pl_i| = m$ nodes in the tree, is in the worst case $|\Sigma|^m$.
So it is often unwise to naively translate a game from extensive to strategic form in order to "solve" it.
If we can find a way to avoid this blow-up, we should.

imperfect information & "perfect recall"

▷ An extensive form game (EFG) is a game of **imperfect information** if it has non-trivial (size > 1) information sets. Players don't have full knowledge of the current "state" (current node of the game tree).

▷ Informally, an imperfect information EFG has **perfect recall** if each player i never "forgets" its own sequence of prior actions and information sets. I.e., a EFG has perfect recall if whenever $w, w' \in \mathit{Info}_{i,j}$ belong to the same information set, then the "*visible history*" for player i (sequence of information sets and actions of player i during the play) prior to hitting node w and w' must be exactly the same.

▷ [Kuhn'53]: with perfect recall it suffices to restrict players' strategies to **behavior strategies**: strategies that only randomize (independently) on actions at each information set.

▷ Perfect recall is often assumed as a "sanity condition" for EFGs (most games we encounter do have perfect recall).

subgames and (subgame) perfection

▷ A **subgame** of an extensive form game is any subtree of the game tree which has *self-contained information sets*. (I.e., every node in that subtree must be contained in an information set that is itself entirely contained in that subtree.)

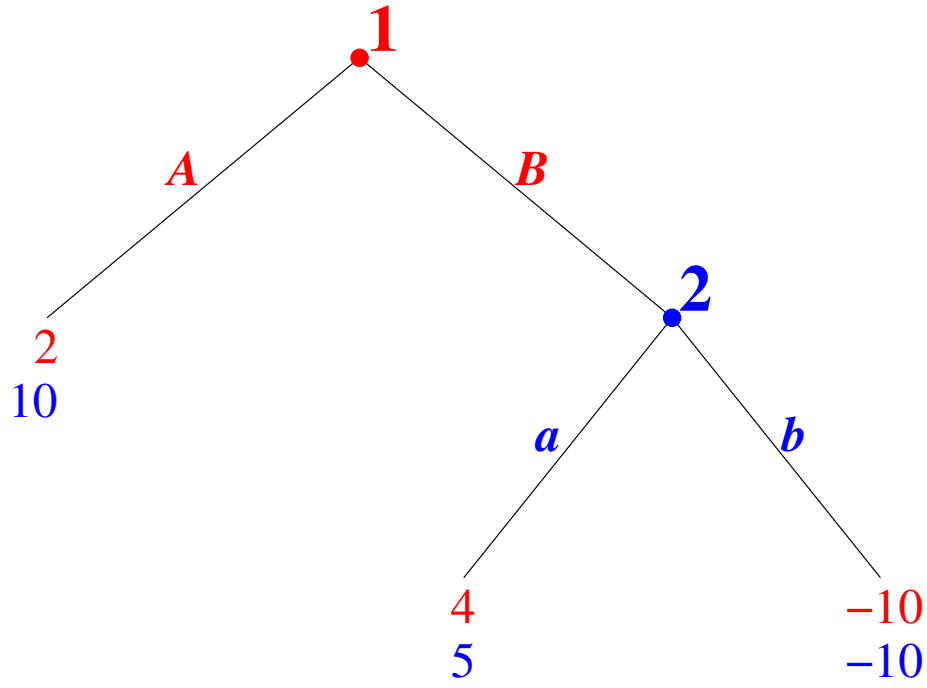
▷ For an extensive form game G, a profile of behavior strategies $b = (b_1, \ldots, b_n)$ for the players is called a **subgame perfect equilibrium** (SGPE) if it defines a Nash equilibrium for *every* subgame of G.

▷ [Selten'75]: Nash equilibrium (NE) (and even SPGE) is inadequately refined as a solution concept for extensive form games. In particular, such equilibria can involve **"Non-credible threats"**:

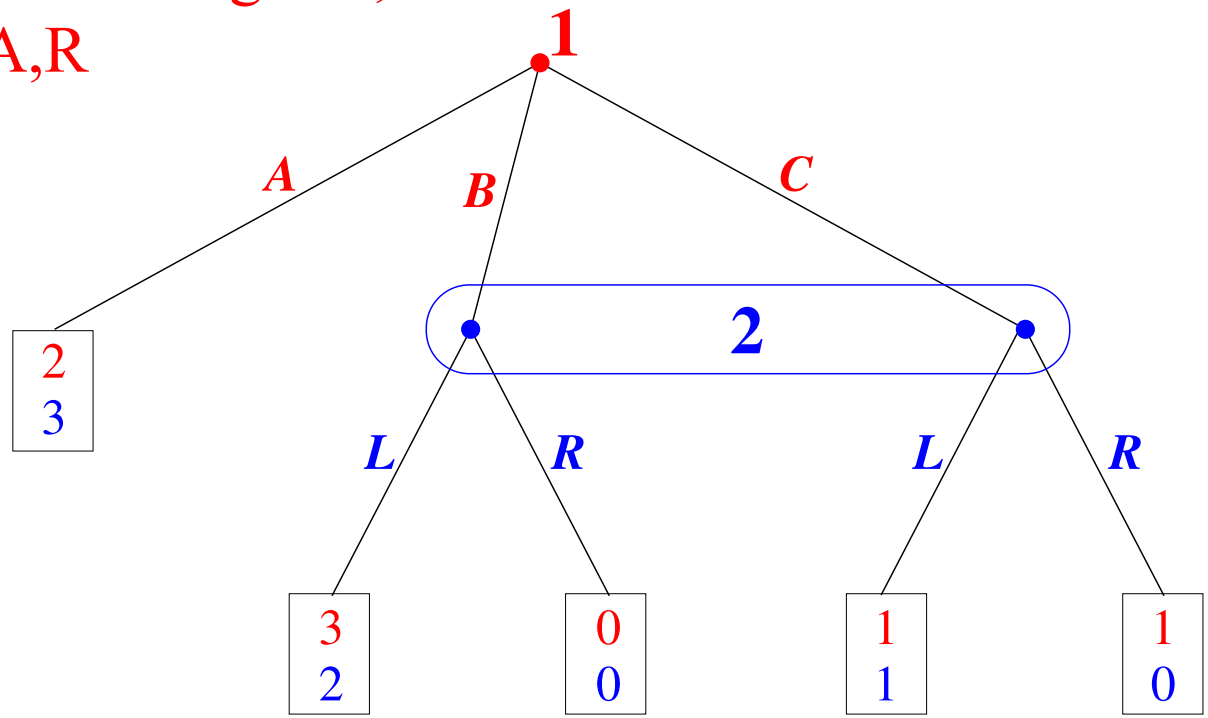
NE: A,b & B,a

Two subgames: after node 2 and the whole graph

B,a is the only subgame perfect equilbrium



the blue node is not a subgame,
there is no trivial subgame, so all NE is SGPE
NE: B,L & A,R



Addressing this general inadequecy of NE and SGPE requires a more refined notion of equilibrium called **trembling-hand perfect equilibrium** [Selten'73].

solving games of imperfect info.

For EFGs *with perfect recall* there are ways to avoid the exponential blow-up of converting to normal form. We only briefly mention algorithms for imp-inf games.
(See, e.g., [Koller-Megiddo-von Stengel'94].)

▷ In strategic form 2-player zero-sum games we can find minimax solutions efficiently (P-time) via LP. For 2-player zero-sum extensive imp-info games (without perfect recall), finding a minimax solution is **NP-hard**. NE's of 2-player EFGs can be found in exponential time.

▷ The situation is better with perfect recall: 2-player zero-sum imp-info games of perfect recall can be solved in P-time, via LP, and 2-player NE's for arbitrary perfect recall games can be found in exponential time using a Lemke-type algorithm.

▷ [Etessami'2014]: For EFGs with ≥ 3 players with perfect recall, computing refinements of Nash equilibrium (including "trembling-hand perfect" and "quasi-perfect") can be reduced to computing a NE for a 3-player normal form game.

Our main focus will be games of perfect information. There the situation is much easier.

games of perfect information

A game of perfect information has only 1 node per information set. So, for these we can forget about information sets.
Examples: Chess, Backgammon, ...
counter-Examples: Poker, Bridge, ...

Theorem([Kuhn'53]) *Every finite extensive game of perfect information, $\mathcal{G}$, has a NE (in fact a SGPE) in pure strategies.*

In other words, there is a pure profile $(s_1, \ldots, s_n) \in S$ that is a Nash Equilibrium (and a subgame perfect equilibrium).
Our proof provides an efficient algorithm to compute such a pure profile, given $\mathcal{G}$, using "backward induction".
A special case of this theorem says the following:

Proposition([Zermelo'1912]) *In Chess, either:*

1. *White has a "winning strategy", or*
2. *Black has a "winning strategy", or*
3. *Both players have strategies to force a draw.*

Next time, perfect information games.

Algorithmic Game Theory and Applications

Lecture 11: Games of Perfect Information

Kousha Etessami

finite games of perfect information

A perfect information (PI) game: 1 node per information set.

Theorem([Kuhn'53]) Every finite n-person extensive PI-game, $\mathcal{G}$, has a NE, in fact, a subgame-perfect NE (SPNE), in pure strategies.

I.e., some pure profile, $s^* = (s_1^*, \ldots, s_n^*)$, is a SPNE.

To prove this, we use some definitions. For a game $\mathcal{G}$ with game tree T, and for $w \in T$, define the **subtree** $T_w \subseteq T$, by: $T_w = \{w' \in T \mid w' = ww'' \text{ for } w'' \in \Sigma^*\}$.

Since tree is finite, we can just associate payoffs to the leaves.

Thus, the subtee T_w, in an obvious way, defines a "**subgame**", $\mathcal{G}_w$, which is also a PI-game.

The **depth** of a node w in T is its length $|w|$ as a string. The depth of tree T is the maximum depth of any node in T. The depth of a game $\mathcal{G}$ is the depth of its game tree.

proof of Kuhn's theorem (backward induction)

Proof We prove by induction on the depth of a subgame $\mathcal{G}_w$ that it has a pure SPNE, $s^w = (s_1^w, \ldots, s_n^w)$. Then $s^* := s^\epsilon$.

Base case, depth 0: In this case we are at a leaf w. there is nothing to show: each player i gets payoff $u_i(w)$, and the strategies in the SPNE s^* are "empty" (it doesn't matter which player's node w is, since there are no actions to take.)

Inductive step: Suppose depth of $\mathcal{G}_w$ is $k+1$. Let $Act(w) = \{a'_1, \ldots, a'_r\}$ be the set of actions available at the root of $\mathcal{G}_w$. The subtrees $T_{wa'_j}$, for $j = 1, \ldots, r$, each define a PI-subgame $\mathcal{G}_{wa'_j}$, of depth $\leq k$.
Thus, by induction, each game $\mathcal{G}_{wa'_j}$ has a pure strategy SPNE, $s^{wa'_j} = (s_1^{wa'_j}, \ldots, s_n^{wa'_j})$.
To define $s^w = (s_1^w, \ldots, s_n^w)$, there are two cases to consider

.....

two cases

1. $w \in Pl_0$, i.e., the root node, w, of T_w is a chance node (belongs to "nature").
Let the strategy s_i^w for player i be just the obvious "union" $\bigcup_{a' \in Act(w)} s_i^{wa'}$, of its pure strategies in each of the subgames. (Explanation of "union" of disjoint strategy functions.)
Claim: $s^w = (s_1^w, \ldots, s_n^w)$ is a pure SPNE of $\mathcal{G}_w$. Suppose not. Then some player i could improve its expected payoff by switching to a different pure strategy in one of the subgames. But that violates the inductive hypothesis on that subgame.

2. $w \in Pl_i$, $i > 0$: the root, w, of T_w belongs to player i. For $a \in Act(w)$, let $h_i^{wa}(s^{wa})$ be the expected payoff to player i in the subgame $\mathcal{G}_{wa}$. Let $a' = \arg\max_{a \in Act(w)} h_i^{wa}(s^{wa})$. For players $i' \neq i$, define $s_{i'}^w = \bigcup_{a \in Act(w)} s_{i'}^{wa}$.
For i, define $s_i^w = (\bigcup_{a \in Act(w)} s_i^{wa}) \cup \{w \mapsto a'\}$.
Claim: $s^w = (s_1^w, \ldots, s_n^w)$ is a pure SPNE of $\mathcal{G}_w$. □

algorithm for computing a SPNE in finite PI-games

The proof yields an EASY "bottom up" algorithm for computing a pure SPNE in a finite PI-game:
We inductively "attach" to the root of every subtree T_w, a SPNE s^w for the game $\mathcal{G}_w$, together with the expected payoff vector $h^w := (h_1^w(s^w), \ldots, h_n^w(s^w))$.

1. Initially: Attach to each leaf w the empty profile $s^w = (\emptyset, \ldots, \emptyset)$, & payoff vector $h^w := (u_1(w), \ldots, u_n(w))$.
2. **While** ($\exists$ unattached node w whose children are attached)
   - if $(w \in Pl_0)$ then
     $s^w := (s_1^w, \ldots, s_n^w)$, where $s_i^w := \bigcup_{a \in Act(w)} s_i^{wa}$;
     hence h^w is: $h_i^w(s^w) := \sum_{a \in Act(w)} q_w(a) * h_i^{wa}(s^{wa})$;

   else if $(w \in Pl_i \ \& \ i > 0)$ then
     Let $s^w := (s_1^w, \ldots, s_n^w)$, & $h^w := h^{wa'}$, where
     $a' := \arg\max_{a \in Act(w)} h_i^{wa}(s^{wa})$,
     $s_{i'}^w := \bigcup_{a \in Act(w)} s_{i'}^{wa}$, for $i' \neq i$, and
     $s_i^w := (\bigcup_{a \in Act(w)} s_i^{wa}) \bigcup \{w \mapsto a'\}$;

consequences for zero-sum finite PI-games

Recall that, by the Minimax Theorem, for every finite zero-sum game Γ, there is a value v^* such that for any NE (x_1^*, x_2^*) of Γ, $v^* = U(x_1^*, x_2^*)$, and

$$\max_{x_1 \in X_1} \min_{x_2 \in X_2} U(x_1, x_2) = v^* = \min_{x_2 \in X_2} \max_{x_1 \in X_1} U(x_1, x_2)$$

But it follows from Kuhn's theorem that for extensive PI-games $\mathcal{G}$ there is in fact a pure NE (in fact, SPNE) (s_1^*, s_2^*) such that $v^* = u(s_1^*, s_2^*) := h(s_1^*, s_2^*)$, and thus that in fact

$$\max_{s_1 \in S_1} \min_{s_2 \in S_2} u(s_1, s_2) = v^* = \min_{s_2 \in S_2} \max_{s_1 \in S_1} u(s_1, s_2)$$

Definition A finite zero-sum game Γ is **determined**, if

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

It thus follows from Kuhn's theorem that:

Proposition ([Zermelo'1912]) Every finite zero-sum PI-game, $\mathcal{G}$, is determined. Moreover, the value & a pure minimax profile can be computed "efficiently" from $\mathcal{G}$.

chess

Chess is a finite PI-game (after 50 moves with no piece taken, it ends in a draw). In fact, it's a **win-lose-draw** PI-game: no chance nodes possible payoffs are 1, -1, and 0.

Proposition([Zermelo'1912]) In Chess, either:

1. White has a "winning strategy", or
2. Black has a "winning strategy", or
3. Both players have strategies to force a draw.

A "**winning strategy**", e.g., for White (Player 1) is a pure strategy s_1^* that guarantees value $u(s_1^*, s_2) = 1$, for all s_2.

chess

Chess is a finite PI-game (after 50 moves with no piece taken, it ends in a draw). In fact, it's a **win-lose-draw** PI-game: no chance nodes possible payoffs are 1, -1, and 0.

Proposition([Zermelo'1912]) In Chess, either:

1. White has a "winning strategy", or
2. Black has a "winning strategy", or
3. Both players have strategies to force a draw.

A "**winning strategy**", e.g., for White (Player 1) is a pure strategy s_1^* that guarantees value $u(s_1^*, s_2) = 1$, for all s_2.

Question: Which one is the right answer??

chess

Chess is a finite PI-game (after 50 moves with no piece taken, it ends in a draw). In fact, it's a **win-lose-draw** PI-game: no chance nodes possible payoffs are 1, -1, and 0.

Proposition([Zermelo'1912]) In Chess, either:

1. White has a "winning strategy", or
2. Black has a "winning strategy", or
3. Both players have strategies to force a draw.

A "**winning strategy**", e.g., for White (Player 1) is a pure strategy s_1^* that guarantees value $u(s_1^*, s_2) = 1$, for all s_2.

Question: Which one is the right answer??

Problem: The tree is far too big!!

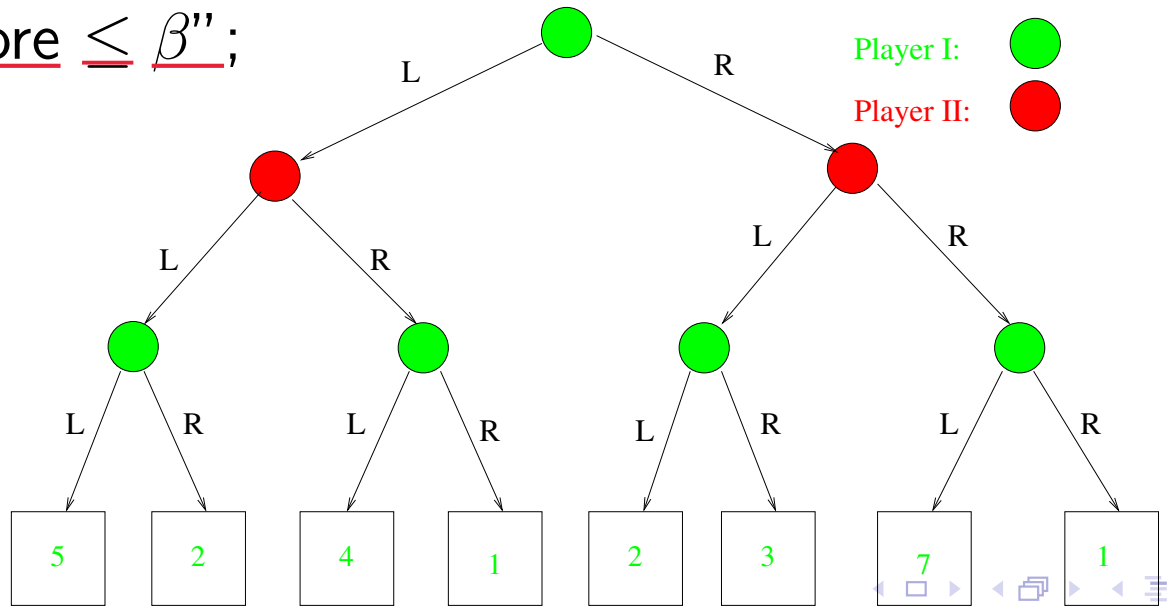
Even with ~ 200 depth & ~ 5 moves per node:

$$5^{200} \text{ nodes!}$$

Despite having an "efficient" algorithm to compute the value v^* given the tree, we can't even look at the whole tree! We need algorithms that don't look at the whole tree.

50 years of game-tree search

There's $>$ 50 years of research on chess & other game playing programs, (Shannon, Turing, ...). Heuristic game-tree search is now very refined. See any AI text (e.g., [Russel-Norvig]). If we have a function *Eval*(w) that heuristically "evaluates" a node's "goodness" score, we can use *Eval*(w) to stop the search at, e.g., desired depth. While searching "top-down", we can "prune out" irrelevant subtrees using α-β-**pruning**. Idea: while searching minmax tree, maintain two values: α- "maximizer can assure score $\geq \alpha$"; & β- "minimizer can assure score $\leq \beta$";



minmax search with α-β-pruning

Assume, for simplicity, that players alternate moves, root belongs to Player 1 (maximizer), and $-1 \leq \mathit{Eval}(w) \leq +1$. Score -1 ($+1$) means player 1 definitely loses (wins). Start the search by calling: **MaxVal**$(\epsilon, -1, +1)$;

MaxVal(w, α, β)

If $\mathit{depth}(w) \geq \mathit{MaxDepth}$ then **return** $\mathit{Eval}(w)$.

Else, for each $a \in \mathit{Act}(w)$

$\alpha := \max\{\alpha,$ **MinVal**$(wa, \alpha, \beta)\}$;

if $\alpha \geq \beta$, then **return** β

return α

MinVal(w, α, β)

If $\mathit{depth}(w) \geq \mathit{MaxDepth}$, then **return** $\mathit{Eval}(w)$.

Else, for each $a \in \mathit{Act}(w)$

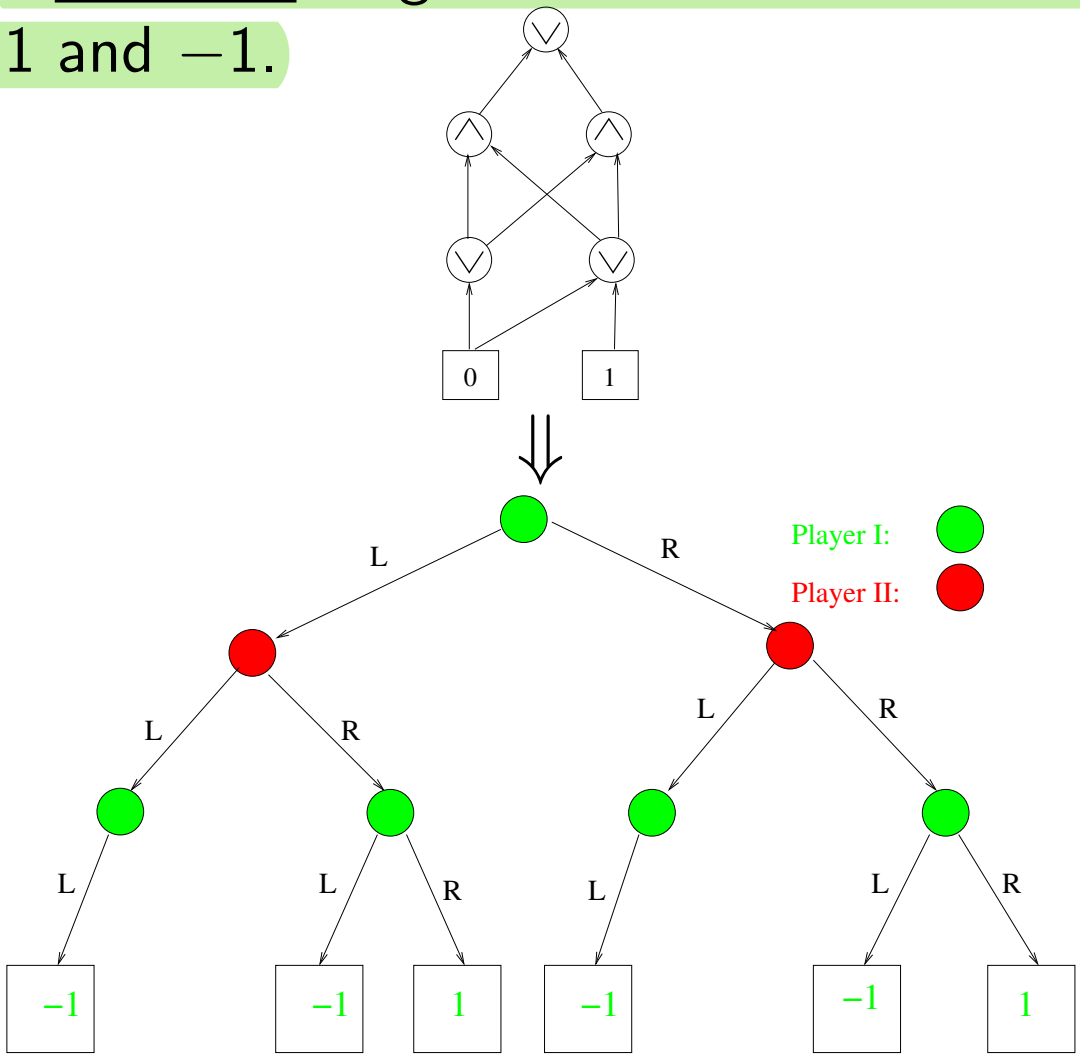
$\beta := \min\{\beta,$ **MaxVal**$(wa, \alpha, \beta)\}$;

if $\beta \leq \alpha$, then **return** α

return β

boolean circuits as finite PI-games

Boolean circuits can be viewed as a zero-sum PI-game, between AND and OR: OR the maximizer, AND the minimizer: a **win-lose** PI-game: no chance nodes & only payoffs are 1 and -1.



Let's generalize to infinite zero-sum PI-games

For a (possibly infinite) zero-sum 2-player PI-game, we would like to similarly define the game to be "determined" if

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

But, for infinite games max & min may not exist! Instead, we call an (infinite) zero-sum game **determined** if:

$$\sup_{s_1 \in S_1} \inf_{s_2 \in S_2} u(s_1, s_2) = \inf_{s_2 \in S_2} \sup_{s_1 \in S_1} u(s_1, s_2)$$

In the simple setting of infinite win-lose PI-games (2 players, zero-sum, no chance nodes, and only payoffs are 1 and -1), this definition says a game is determined precisely when one player or the other has a **winning strategy**: a strategy $s_1^* \in S_1$ such that for any $s_2 \in S_2$, $u(s_1^*, s_2) = 1$ (and vice versa for player 2).

Question: Is every win-lose PI-game determined?

Let's generalize to infinite zero-sum PI-games

For a (possibly infinite) zero-sum 2-player PI-game, we would like to similarly define the game to be "determined" if

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

But, for infinite games max & min may not exist! Instead, we call an (infinite) zero-sum game **determined** if:

$$\sup_{s_1 \in S_1} \inf_{s_2 \in S_2} u(s_1, s_2) = \inf_{s_2 \in S_2} \sup_{s_1 \in S_1} u(s_1, s_2)$$

In the simple setting of infinite win-lose PI-games (2 players, zero-sum, no chance nodes, and only payoffs are 1 and -1), this definition says a game is determined precisely when one player or the other has a **winning strategy**: a strategy $s_1^* \in S_1$ such that for any $s_2 \in S_2$, $u(s_1^*, s_2) = 1$ (and vice versa for player 2).

Question: Is every win-lose PI-game determined?

Answer: No .....

determinacy and its boundaries

For win-lose PI-games, we can define the payoff function by providing the set $Y = u_1^{-1}(1) \subseteq \Psi_T$, of complete plays on which player 1 wins (player 2 necessarily wins on all other plays).

If, additionally, we assume that players alternate moves, we can specify such a game as $\mathcal{G} = \langle T, Y \rangle$.

Fact For tree $T = \{L, R\}^*$, there are sets $Y \subseteq \Psi_T$, such that the win-lose PI-game $\mathcal{G} = \langle T, Y \rangle$ is **not** determined.

(Proof uses the "axiom of choice". See, e.g., [Mycielski, Ch. 3 of Handbook of GT,1992].)

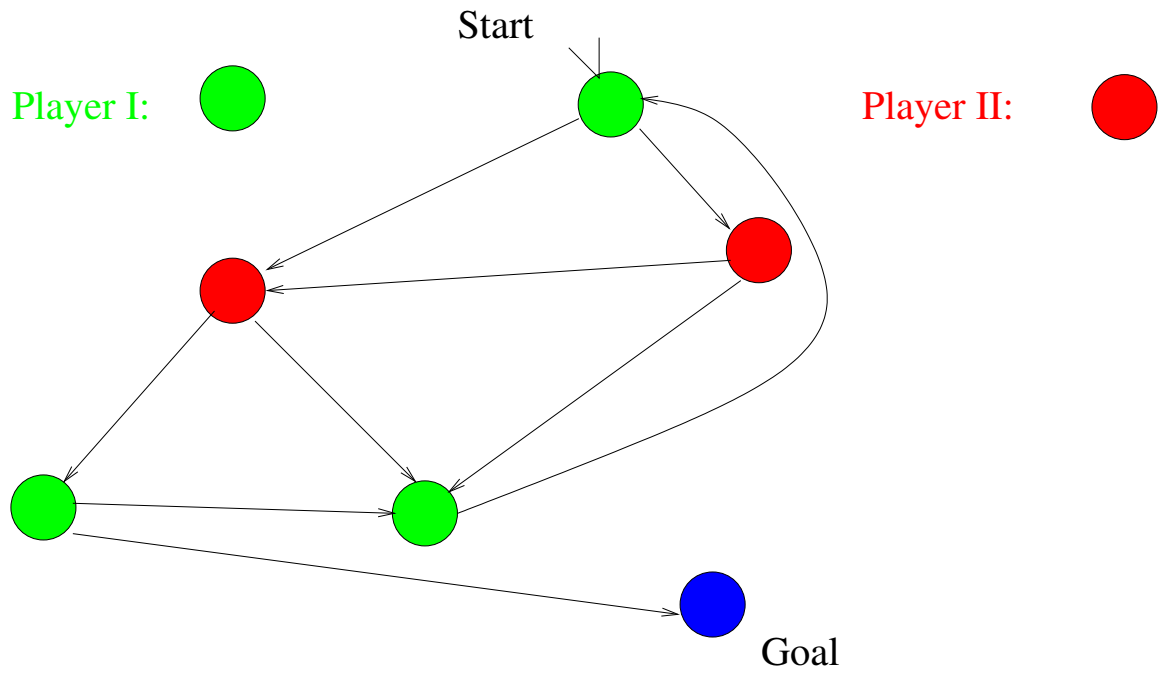
Fortunately, large classes of win-lose PI-games are determined:

Theorem([D. A. Martin'75]) Whenever Y is a so called "*Borel set*", the game $\langle \Sigma^*, Y \rangle$ is determined.

(A deep theorem, with connections to logic and set theory. Theorem holds even when the action alphabet Σ is infinite.)

food for thought: win-lose games on finite graphs

Instead of a tree, we have a finite directed graph:



▷ Starting at "Start", does Player I have a strategy to "force" the play to reach the "Goal"?

▷ Note: this is a (possibly infinite) win-lose PI-game.

▷ Is this game determined for all finite graphs?

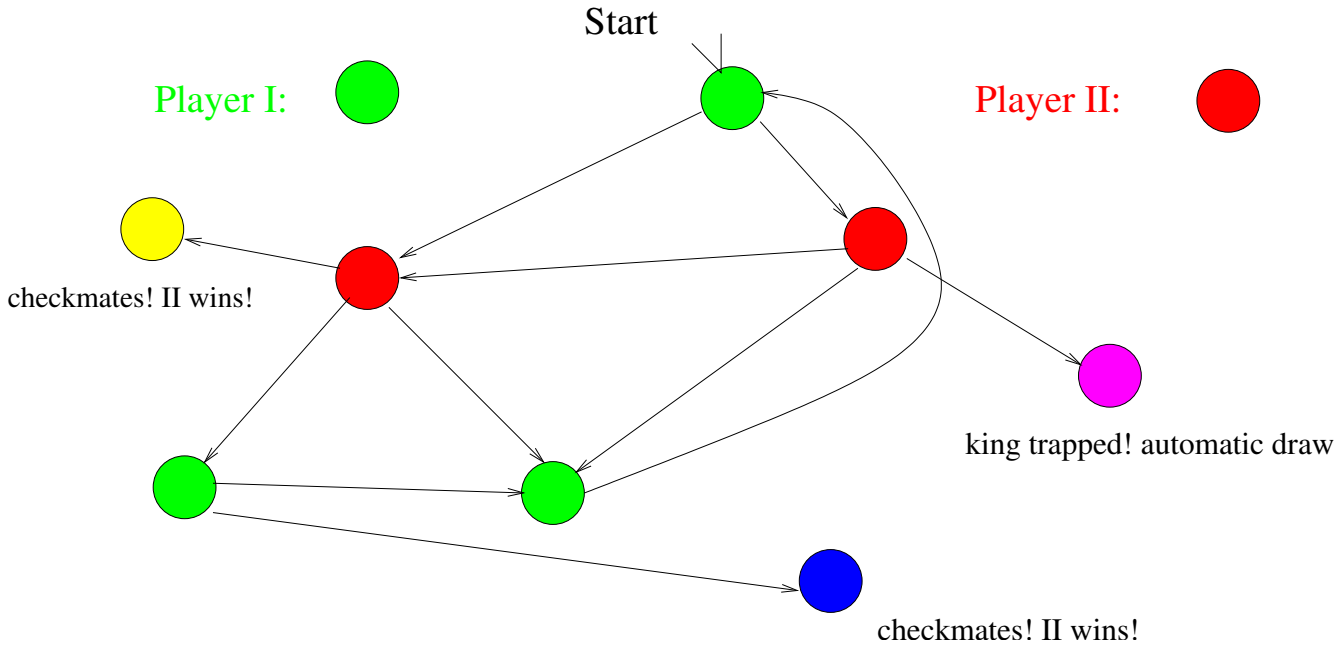
▷ If so, how would you compute a winning strategy for Player 1?

Algorithmic Game Theory and Applications

Lecture 12: Games on Graphs

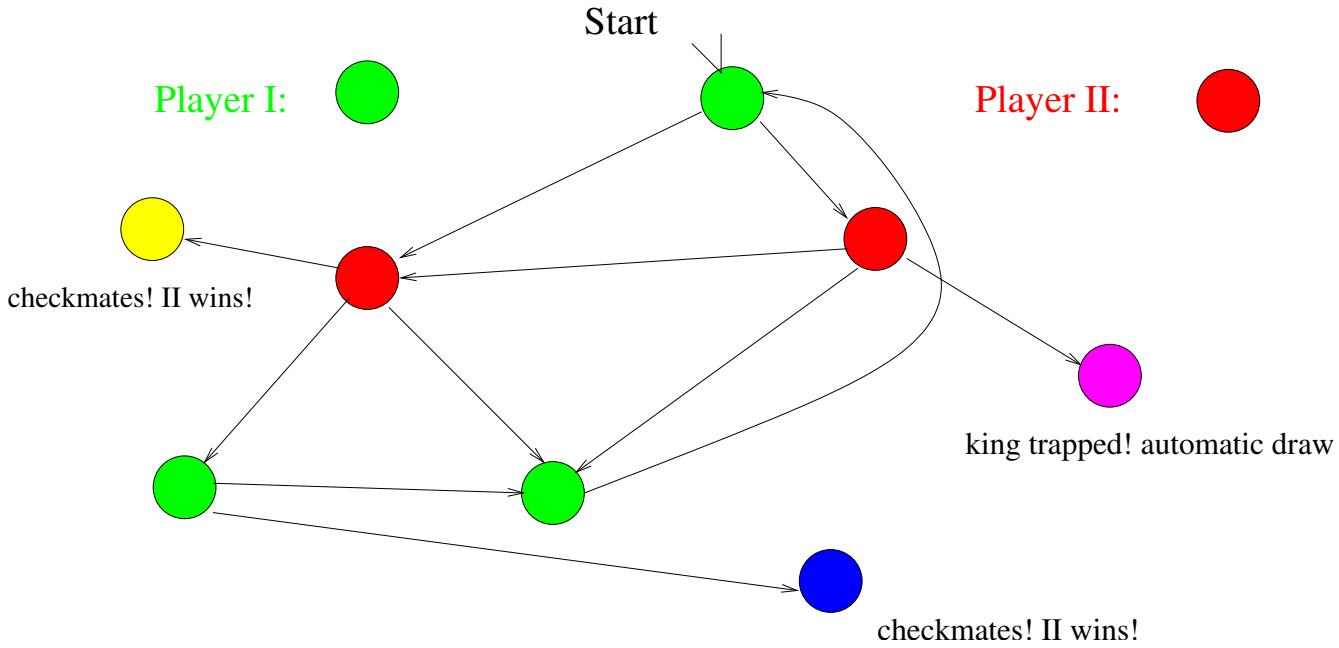
Kousha Etessami

unbounded chess



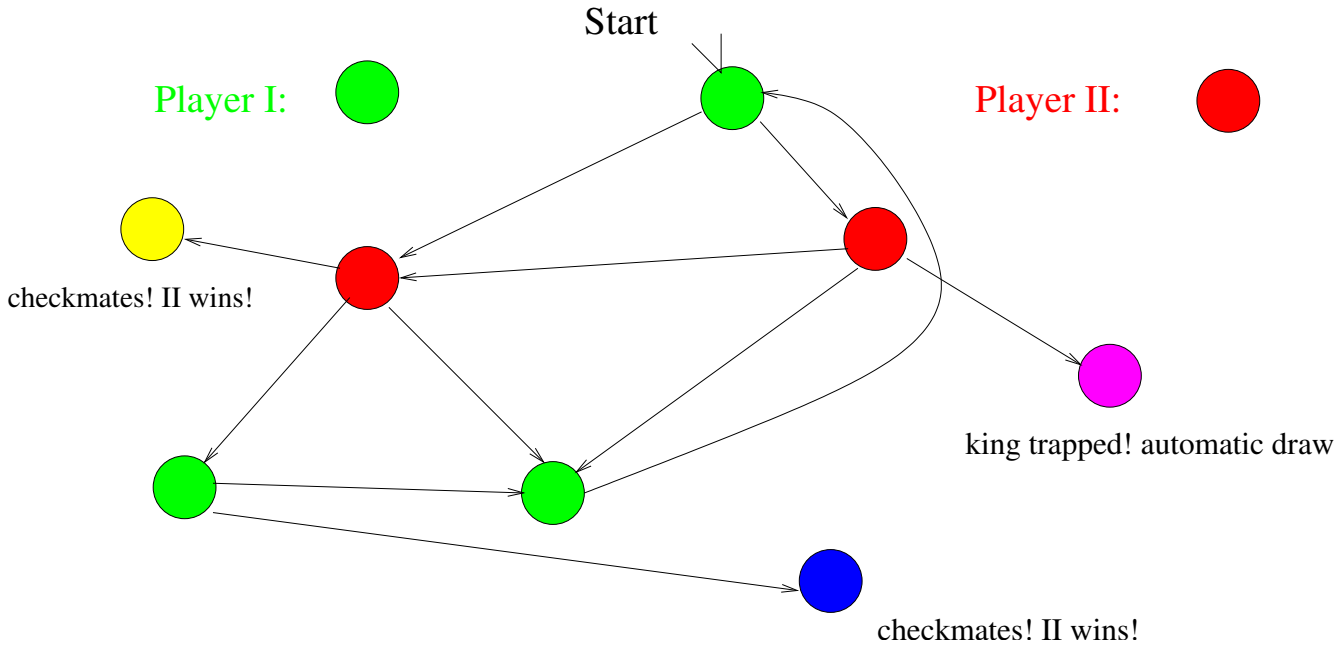
▷ Chess: the same "position/configuration" might recur in the game, but the (infinite) "game tree" does not reflect this.

unbounded chess



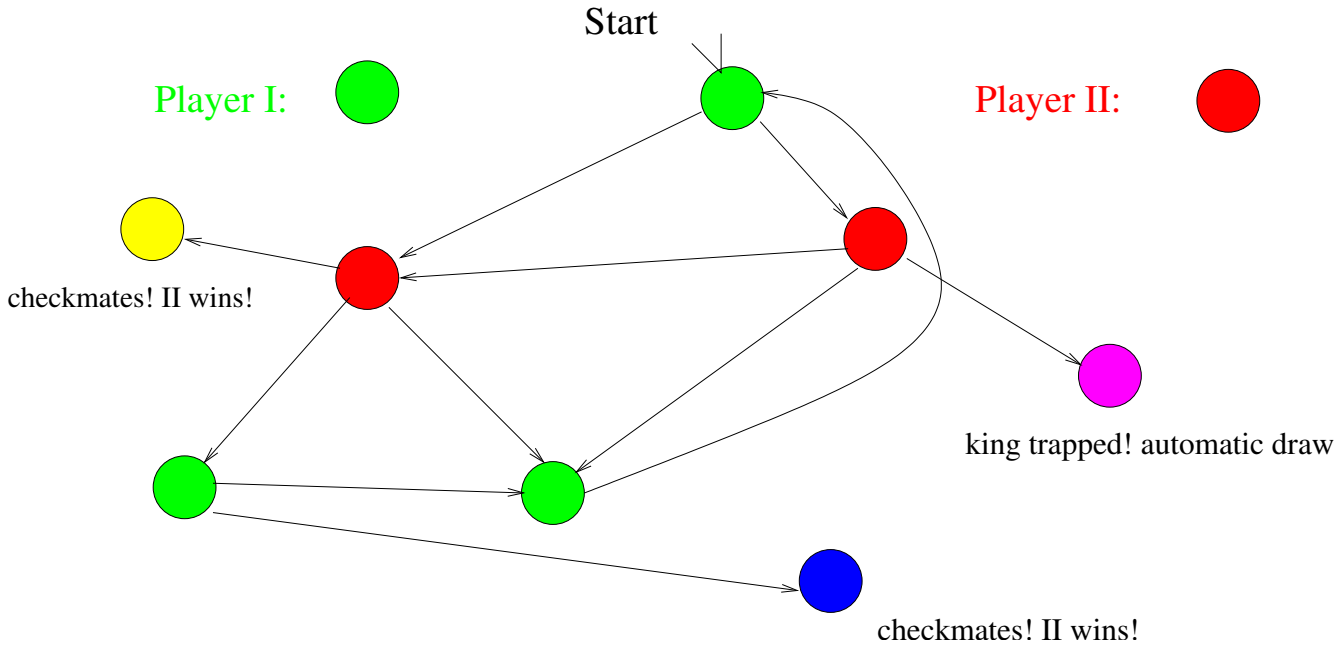
▷ Chess: the same "position/configuration" might recur in the game, but the (infinite) "game tree" does not reflect this.
▷ There are finitely many positions ($\leq 64^{32}$). After some depth, every "play" contains recurrences of positions.

unbounded chess



▷ Chess: the same "position/configuration" might recur in the game, but the (infinite) "game tree" does not reflect this.
▷ There are finitely many positions ($\leq 64^{32}$). After some depth, every "play" contains recurrences of positions.
▷ Consider "unbounded chess" without artificial stopping conditions: an infinite play is by definition a draw.

unbounded chess



▷ Chess: the same "position/configuration" might recur in the game, but the (infinite) "game tree" does not reflect this.
▷ There are finitely many positions ($\leq 64^{32}$). After some depth, every "play" contains recurrences of positions.
▷ Consider "unbounded chess" without artificial stopping conditions: an infinite play is by definition a draw.

Is this win-lose-draw game determined? I.e., does Zermelo's theorem still hold?

more serious motivation

▷ We can often model the dynamics of a system (e.g., a running program) as a state transition system.

▷ If the system interacts with an environment, transitions out of some states can be viewed as "controlled by the environment". Can the environment force the system, with some sequence of inputs, into a "bad state"?

▷ Even for state machines without environments, certain temporal queries about the behavior of the system over time can be formulated as a game on a graph.

▷ Such queries, and much more, can be formalized in certain "temporal logics": formal languages for describing relationships between the occurrence of events over time. Efficiently checking such queries against a system model (e.g., a state transition system) is the task of "model checking". Some key model checking tasks are intimately related to efficiently solving certain games on graphs.

game graphs and their trees

A 2-player **game graph**, $G = (V, E, pl)$ consists of:

$\triangleright$ A (finite) set V of vertices.

$\triangleright$ A set $E \subseteq V \times V$ of edges.

$\triangleright$ A partition (V_1, V_2) of the vertices $V = V_1 \cup V_2$ into two disjoint sets belonging to players 1 and 2, respectively.

game graphs and their trees

A 2-player **game graph**, $G = (V, E, pl)$ consists of:
▷ A (finite) set V of vertices.
▷ A set $E \subseteq V \times V$ of edges.
▷ A partition (V_1, V_2) of the vertices $V = V_1 \cup V_2$ into two disjoint sets belonging to players 1 and 2, respectively.

A game graph G together with a start vertex $v_0 \in V$, defines a game tree T_{v_0} given by:
▷ Action alphabet $\Sigma = V$. Thus $T_{v_0} \subseteq V^*$.
▷ $\epsilon \in T_{v_0}$, and $wv'' \in T_{v_0}$, for $v'' \in V$, if and only if

- $w = \epsilon$ and $(v_0, v'') \in E$, or
- $w = w'v'$, for some $v' \in V$, and $(v', v'') \in E$.

We extend the partition (V_1, V_2) to a partition (T_1', T_2') of the tree nodes of T_{v_0} as follows: if $v_0 \in V_i$, then $\epsilon \in T_i'$, and if $v' \in V_i$, then any tree node $wv' \in T_i$.
T_{v_0} is thus a game tree, where $Act(wv') = \{v'' \mid (v', v'') \in E\}$, whose plays are all paths in the graph G starting from v_0.

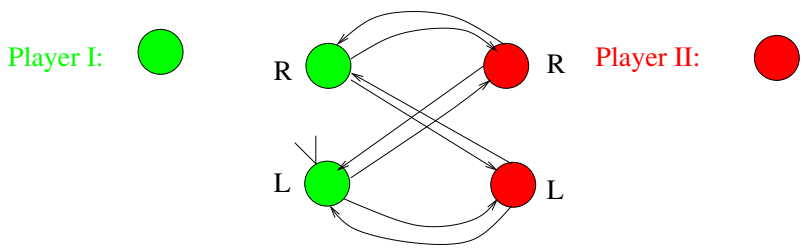
games on graphs

A **game on a graph**, $\mathcal{G}_{v_0}$, is given by:

> *A finite game graph G, vertex $v_0 \in V$, and payoff function $u : \Psi_{T_{v_0}} \mapsto \mathbb{R}$.*

These together define a 2-player zero-sum PI-game with game tree T_{v_0}.

Note: We already know that even for win-lose payoff functions u, games on finite graphs are not in general determined, because the infinite binary tree $\{L, R\}^*$ is the game tree for the following game graph:



and we already know (lecture 11) that there are sets Y of plays such that the win-lose game $\langle \{L, R\}^*, Y \rangle$ is not determined. So, let's restrict the possible payoff functions.

"history oblivious" payoffs

▷ Suppose $\exists$ vertex v' of graph G that is a "dead end". E.g., in chess this could be "checkmate for Player I".

▷ There may be many ways to get to v', but the winner is the same for any finite play $wv' \in V^*$. I.e., $u(wv') = u(w'v')$, for all $wv', w'v' \in V^*$. So, the payoff is "history oblivious".

▷ What about for infinite plays π? We can think of π as an infinite sequence $v_0 v_1 v_2 v_3 v_4 v_5 \ldots\ldots$, where each $v_i \in V$. We use the notation $\pi \in V^\omega$.

▷ For $\pi = v_0 v_1 \ldots$, let

$$\inf(\pi) = \{v \in V \mid \text{for } \infty\text{-many } i \in \mathbb{N},\ v_i = v\}$$

▷ Let's call payoff function $u()$ history oblivious (**h.o.**), if for all infinite plays π & π', if $\inf(\pi) = \inf(\pi')$, then $u(\pi) = u(\pi')$, and for all finite complete plays wv and $w'v$,

$$u(wv) = u(w'v).$$

Call a graph game h.o. if its payoffs are h.o.

We will only consider h.o. games (and often less).

"finitistic" payoffs

▷ Note that in chess, if the play π is infinite, then the play is always a draw, i.e., $u(\pi) = 0$.

▷ Let's call an h.o. payoff function finitistic if for all infinite plays π and π', $u(\pi) = u(\pi')$. Let's call a game on a graph $\mathcal{G}_{v_0}$ finitistic if its payoff function is.

So, in win-lose-draw finitistic games, infinite plays are either all wins, all losses, or all draws, for player 1.

Question: Are all finitistic games on graphs determined?

Answer: Yes.....

In fact, more it true: for finitistic games there is always a memoryless strategy for each player that achieves the value of the game, and we can efficiently compute these strategies.

memoryless strategies and determinacy

Definition For a game $\mathcal{G}_{v_0}$, a strategy s_i for player i is a **memoryless strategy** if for all $wv, w'v \in Pl'_i$, $s_i(wv) = s_i(w'v)$, and if $wv_0 \in Pl'_i$ then $s_i(wv_0) = s_i(\epsilon)$.

I.e., the strategy always makes the same move from a vertex, regardless of the history of how it got there.

Let MLS_i denote the set of memoryless strategies for player i. MLS_i is a finite set, even if S_i is not. In particular, if $m = |Pl_i|$ is the number of vertices belonging to player i, then $|MLS_i| \leq |\Sigma|^m$.

Definition $\mathcal{G}_{v_0}$ is **memorylessly determined** if both players have memoryless strategies that achieve "the value". I.e.,

$$\max_{s_1 \in MLS_1} \inf_{s_2 \in S_2} u(s_1, s_2) = \min_{s_2 \in MLS_2} \sup_{s_1 \in S_1} u(s_1, s_2)$$

Theorem A Finitistic games on finite graphs are memorylessly determined. Moreover, there is an efficient (P-time) algorithm to compute memoryless value-achieving strategies in such games.

the win-lose case: easy "fixed point" algorithm

We first prove the theorem for finitistic win-lose games via an easy "bottom up" fixed point algorithm.

Input: Game graph $G = (V, E, pl, v_0)$.

Assume w.l.o.g. all infinite plays are win for player 2 (other case is symmetric). "Dead end": vertex with no outgoing edge. $Good := \{v \in V \mid v \text{ a dead end that wins for player 1}\}$. $Bad := \{v \in V \mid v \text{ a dead end that wins for player 2}\}$.

1. Initialize: $Win_1 := Good$; $St_1 := \emptyset$;
2. **Repeat**
   Foreach $v \notin Win_1$: pl(v) =1, node v belongs to player 1
   If $(pl(v) = 1 \;\&\; \exists\, (v, v') \in E : \; v' \in Win_1)$
   $Win_1 := Win_1 \cup \{v\}$; $St_1 := St_1 \cup \{v \mapsto v'\}$;
   If $(pl(v) = 2 \;\&\; \forall\, (v, v') \in E : \; v' \in Win_1)$
   $Win_1 := Win_1 \cup \{v\}$;

   Until The set Win_1 does not change;

Player 1 has a Win.-Strategy iff $v_0 \in Win_1$. If so, St_1 is a memoryless winning strategy for player 1.

why does this work?

Proof of Theorem A: (for the win-lose case)

$\triangleright$ First, we claim that for each $v \in Win_1$, St_1 is a winning strategy for player 1 in the game $\mathcal{G}_v$ (i.e., the game that starts at node v).

Suppose $v \in Win_1$. It must have entered Win_1 after, say, m iterations of the repeat loop. By induction on m, if player 1 plays according to (partial) strategy St_1, then it is guaranteed a win in the game $\mathcal{G}_v$ within m moves. Note that St_1 may be partial: it may only tell us how to move from some vertices. This won't matter.

Base case: $m = 0$, $v \in Good$.

Inductively: either v is player 1's vertex or 2's.

If it is player 1's, then $St_1(v) = v'$, where $(v, v') \in E$ and $v' \in Win_1$, and furthermore v' entered Win_1 by $m - 1$ iterations. By induction St_1 wins for player 1 from v' in $m - 1$ moves.

If v is player 2's, then we know that for all $(v, v') \in E$, $v' \in Win_1$, and furthermore v' entered Win_1 by $\leq m-1$ iterations. Thus, no matter what move player 2 makes, in 1 move, by induction, we will be at a vertex $v' \in Win_1$ where player wins with St_1 within $m-1$ moves.

pl^{-1}(2): all nodes belongs to player 2

▷ Now consider $v \notin Win_1$ when algorithm halts. For each $v' \in pl^{-1}(2)$, if $\exists\,(v', v'') \in E$, with $v'' \notin Win_1$, then pick one such v'', and let $St_2 := St_2 \cup \{v' \mapsto v''\}$. St_2 may also be partial.

We claim St_2 is a memoryless winning strategy for player 2 in every game $\mathcal{G}_v$, where $v \notin Win_1$. Suppose St_2 is not a winning strategy for some $v \notin Win_1$. Then player 1 must be able to win by reaching a *Good* vertex within say, m moves from v against St_2. Let's show this is a contradiction.

Base case: m$=0$, but then $v \in Good$. $\Rightarrow\Leftarrow$.

Inductively: either v is player 1's or player 2's.

If player 1's, then $\forall (v, v') \in E$, $v' \notin Win_1$, because otherwise by the algorithm $v \in Win_1$. Suppose player 1's winning strategy is to play $(v, v') \in E$. It must have a win within $m-1$ moves from $v' \notin Win_1$ against St_2. $\Rightarrow\Leftarrow$.

If it is player 2's move, then one possibility is $v \in Bad$, ($\Rightarrow\Leftarrow$). Otherwise, $St_2(v) = v'$ must be defined: since $v \notin Win_1$, there must exist $(v, v') \in E$ with $v' \notin Win_1$. Otherwise, by the algorithm, $v \in Win_1$.

By induction, player 1 must have a $(m-1)$-winning strategy from $v' \notin Win_1$.$\Rightarrow\Leftarrow$.

generalizing to finitistic zero-sum

The generalization is not hard:
In a finitistic game, there can only be a bounded number, $r \leq |V| + 1$, of distinct payoffs $u(\pi)$,

$$j_1 < j_2 < j_3 < \ldots < j_r$$

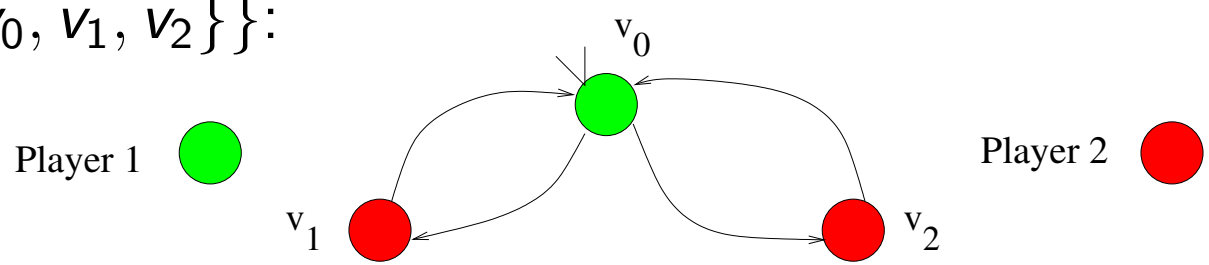
and one of these, say j_k, is the payoff $u(\pi)$ for all infinite plays π. Suppose, w.l.o.g., that $k < r$. (If instead $1 < k$, then we work symmetrically with respect to player 2. If $1 = k = r$, then all payoffs are equal and there is nothing to do.)
Consider a new win-lose game where player 1 wins if it attains payoff j_r, and loses if its payoff is any less. Use the fixed point algorithm on this game to find a memoryless (partial) strategy for player 1 that is winning from vertices in Win_1 where payoff j_r can be obtained. We can then eliminate Win_1 vertices and the payoff j_r. We get a new finitistic zero-sum game, with payoffs $j_1 < \ldots < j_{r-1}$. Repeat!! (Homework asks you to solve some of these.)

non-finitistic win-lose h.o. games: Muller games

▷ We will only be interested in win-lose h.o. games.
By attaching a "self-loop" to every dead-end vertex, every play becomes infinite, and we can define the "payoffs" via a set $\mathcal{F} \subseteq 2^V$, where

$$\mathcal{F} = \{F \subseteq V \mid \text{player 1 wins if } \inf(\pi) = F\}$$

We call $\mathcal{F}$ the (Muller) winning condition. Let's call such win-lose h.o. games Muller games.
▷ **Question:** Are all Muller games determined? **Answer:** Yes.
▷ **Question:** Are all Muller games memorylessly determined?
Answer: No! Consider the following Muller game, $\mathcal{F} = \{\{v_0, v_1, v_2\}\}$:



Does Player 1 have a winning strategy?
Does it have a memoryless winning strategy?

remarks

▷ Muller games and restricted variants of them are important in applications to model checking. We can't do them full justice here.

▷ Every Muller game can be converted to an "equivalent" (but potentially exponentially larger) game with a limited kind of Muller winning condition called a parity condition. These so called parity games *are* memorylessly determined.

▷ Can we find winning strategies in parity games efficiently (in P-time)? This is a tantalizing open problem.

It follows from memoryless determinacy that finding winning strategies for them is in **NP** ∩ **co-NP**: we can guess a memoryless strategy for either player and efficiently verify that it is a winning strategy.

▷ An older survey text on all this is:
["*Automata, Logics, and Infinite Games*",
edited by E. Grädel, W. Thomas, T. Wilke, 2002].

food for thought: back to LP

Consider the following LP, for solving a finitistic win-lose game with game graph G. (Suppose w.l.o.g., player 1 loses if the play is infinite.) Let $V = \{v_1, \ldots, v_n\}$ be vertices of G. We will have one LP variable x_i for each vertex $v_i \in V$.

Minimize x_m

Subject to:

$0 \leq x_i \leq 1$, *for* $i = 1, \ldots, n$;

$x_i = 1$, *for* v_i *a winning dead end for player 1.*

$x_i = 0$, *for* v_i *a losing dead end for player 1.*

For each x_i *where* $\mathrm{pl}(v_i) = 1$,

$x_i \geq x_j$, *for each* $(v_i, v_j) \in E$.

For each x_i *where* $\mathrm{pl}(v_i) = 2$,

and $\{v_{j_1}, \ldots, v_{j_r}\} = \{v' \mid (v_i, v') \in E\}$,

$x_i \leq x_{j_k}$, *for* $k = 1, \ldots, r$, *and*

$x_i \geq x_{j_1} + \ldots + x_{j_r} - (r - 1)$

▷ The optimal value of the given LP is 1 iff player 1 has a winning strategy in $\mathcal{G}_{v_m}$.

▷ Now, what if instead of 2 players, player 1 was playing "alone against nature/chance"? Could you formulate an LP for 1's optimal payoff? This would be a simple instance of a "Markov Decision Process".

Algorithmic Game Theory and Applications

Lecture 15: a brief taster of Markov Decision Processes and Stochastic Games

Kousha Etessami

warning

▷ The subjects we will touch on today are vast: one can easily spend an entire course on them alone.

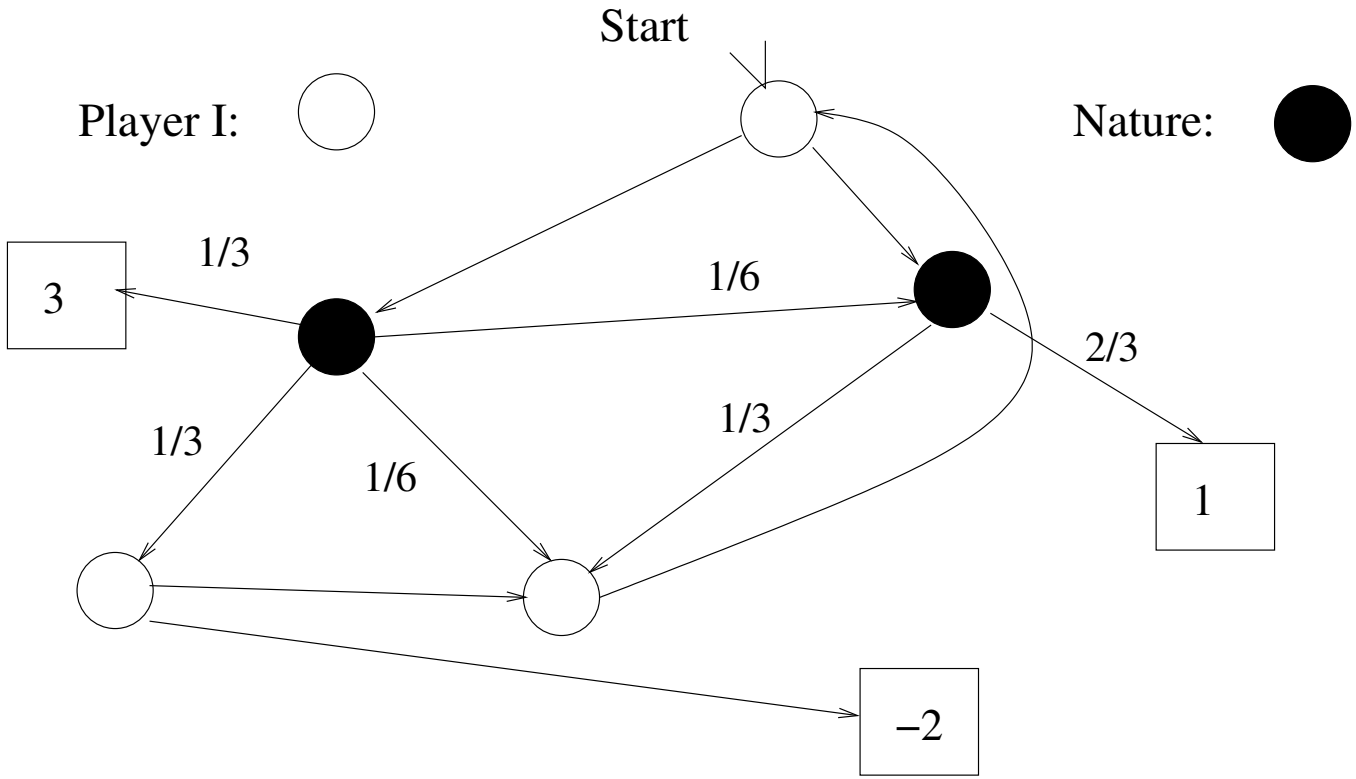
▷ So, what we discuss today is only a brief "taster". Please do explore further if the subject interests you.

▷ Here are two standard textbooks that you can look up if you are interested in learning more:

- M. Puterman, *Markov Decision Processes*, Wiley, 1994.
- J. Filar and K. Vrieze, *Competitive Markov Decision Processes*, Springer, 1997. (For 2-player zero-sum stochastic games.)

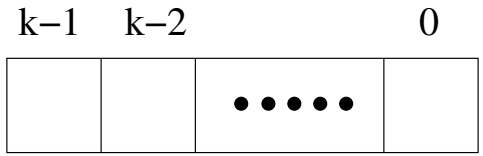
Games against Nature

Consider a game graph, where some nodes belong to player 1 but others are chance nodes of "Nature":



Question: What is Player 1's "optimal strategy" and "optimal expected payoff" in this game?

a simple finite game: "make a big number"



▷ Your goal: create as large a k-digit number as possible, using digits from $D = \{0, 1, 2, \ldots, 9\}$, which "nature/chance" will give you, one by one.

▷ The game proceeds in k rounds.

▷ In each round, "nature" chooses $d \in D$ "uniformly at random", i.e., each digit has probability $1/10$.

▷ You then choose which "unfilled" position in the k-digit number should be "filled" with digit d. (Initially, all k positions are "unfilled".)

▷ The game ends when all k positions are "filled".

Your goal: maximize final number's expected value.

Question: What should your strategy be?

▷ This is a "finite horizon" "Markov Decision Process".

▷ Note that this is a finite PI-game and, *in principle*, we can solve it using the "bottom-up" algorithm.

▷ But we wouldn't want to look at the entire tree if we can avoid it!

vast applications

Beginning in the 1950's with the work of Bellman, Wald, and others, these kinds of "Games Against Nature", a.k.a., "Markov Decision Processes", a.k.a. "Stochastic Dynamic Programs", have been applied to a huge range of subjects.
Examples where MDPs have actually been applied:

- highway repair scheduling.
- bus engine replacement scheduling.
- waste management.
- call center scheduling.
- ...

(See [Puterman'94,Ross'83,Derman'72,Howard'70,Bellman'57,..])

▷ "Reinforcement Learning" (RL), see e.g., [Sutton-Barto'98], is founded on the underlying model of MDPs.
▷ However, in RL, the MDP itself is often not fully visible, and one has to "discover" it by a process of exploration/exploitation.

- The richness of applications shouldn't surprise you.
- We live in an uncertain world, where we constantly have to make decisions in the face of uncertainty about future events.
- But we may have some information, or "belief", about the "likelihood" of future events.
- "I know I may get hit by a car if I walk out of my apartment in the morning." But somehow I still muster the courage to get out.
- I don't however walk into a random pub in Glasgow and yell "I LOVE CELTIC FOOTBALL CLUB", because I know my chances of survival are far lower.

Markov Decision Processes

Definition A **Markov Decision Process** is given by a game graph $G_{v_0} = (V, E, pl, q, v_0, u)$, where:

- V is a (finite) set of vertices.
- $pl : V \mapsto \{0, 1\}$, maps each vertex either to player 0 ("Nature") or to player 1.
- Let $V_0 = pl^{-1}(0)$, and $V_1 = pl^{-1}(1)$.
- $E : V \mapsto 2^V$ maps each vertex v to a set E(v) of "successors" (or "actions" at v).
- For each "nature" vertex, $v \in V_0$, a probability distribution $q_v : E(v) \mapsto [0, 1]$, over the set of "actions" at v, such that $\sum_{v' \in E(v)} q_v(v') = 1$.
- A start vertex $v_0 \in V$.
- A **payoff function**: $u : \Psi_{T_{v_0}} \mapsto \mathbb{R}$, from plays to payoffs for player 1.

Player 1 want to *maximize its expected payoff*.

Many different payoff functions

Many different payoff functions have been studied in the MDP and stochastic game literature. Examples:

1. **Mean payoff**: for every state $v \in V$, associate a payoff $u(v) \in \mathbb{R}$ to that state. For a play $\pi = v_0 v_1 v_2 v_3 \ldots$, the goal is to maximize the expected *mean payoff*:

$$\mathbb{E}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} u(v_i)}{n})$$

Many different payoff functions

Many different payoff functions have been studied in the MDP and stochastic game literature. Examples:

1. **Mean payoff**: for every state $v \in V$, associate a payoff $u(v) \in \mathbb{R}$ to that state. For a play $\pi = v_0 v_1 v_2 v_3 \ldots$, the goal is to maximize the expected *mean payoff*:

$$\mathbb{E}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} u(v_i)}{n})$$

2. **Discounted total payoff:** For a given **discount** factor $0 < \beta < 1$, the goal is to maximize:

$$\mathbb{E}(\lim_{n\to\infty} \sum_{i=0}^{n} \beta^i u(v_i))$$

Many different payoff functions

Many different payoff functions have been studied in the MDP and stochastic game literature. Examples:

1. **Mean payoff**: for every state $v \in V$, associate a payoff $u(v) \in \mathbb{R}$ to that state. For a play $\pi = v_0 v_1 v_2 v_3 \ldots$, the goal is to maximize the expected *mean payoff*:

$$\mathbb{E}(\liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1} u(v_i)}{n})$$

2. **Discounted total payoff:** For a given **discount** factor $0 < \beta < 1$, the goal is to maximize:

$$\mathbb{E}(\lim_{n\to\infty} \sum_{i=0}^{n} \beta^i u(v_i))$$

3. **Probability of reaching target:** Given target $v_T \in V$, goal: maximize (or minimize) probability of reaching v_T. Can be rephrased: for a play $\pi = v_0 v_1 v_2 \ldots$, let $\chi(\pi) := 1$ if $v_i = v_T$ for some $i \geq 0$. Otherwise, $\chi(\pi) := 0$. Goal: maximize/minimize $\mathbb{E}(\chi(\pi))$.

Expected payoffs

▷ Intuitively, we want to define expected payoffs as the sum of payoffs of each play times its probability.
▷ However, of course, this is not possible in general because after fixing a strategy it may be the case that all plays are infinite, and every infinite play has probability 0!
▷ In general, to define the expected payoff for a fixed strategy requires a proper *measure theoretic* treatment of the probability space of infinite plays involved, etc.
▷ This is the same thing we have to do in the theory of *Markov chains* (where there is no player).
▷ We will avoid all the heavy probability theory.
(You have to take it on faith that the intuitive notions can be formally defined appropriately, or consult the cited textbooks.)

memoryless optimal strategies

A **strategy** is again any function that maps each history of the game (ending in a node controlled by player 1), to an *action* (or a probability distribution over actions) at that node.

Theorem *(memoryless optimal strategies)* For every finite-state MDP, with any of the the following objectives:

▷ *Mean payoff*,

▷ *Discounted total payoff*, or

▷ *Probability of reaching target*,

player 1 has a pure memoryless optimal strategy.

In other words, player 1 has an optimal strategy where it just picks one edge from $E(v)$ for each vertex $v \in V_1$.

(For a proof see, e.g., [Puterman'94].)

Bellman Optimality Equations

For the objective of **maximizing probability to reach target vertex** v_T, consider the following system of equations. Let $V = \{v_1, \ldots, v_n\}$ be the set of vertices of the MDP, G. Consider the following system of equations, with one variable x_i for every vertex v_i.

$$\begin{aligned} x_T &= 1 \\ x_i &= \max\{x_j \mid v_j \in E(v_i)\} \quad \text{for } v_i \in V_1 \\ x_i &= \sum_{v_j \in E(v_i)} q_{v_i}(v_j) \cdot x_j \quad \text{for } v_i \in V_0 \end{aligned}$$

Theorem *These max-linear* **Bellman equations** *for the MDP, have a (unique) least non-negative solution vector* $x^* = (x_1^*, \ldots, x_n^*) \in [0,1]^n$, *in which* x_i^* *is the optimal probability for player 1 to reach the target* v_T *in the MDP* G_{v_i} *starting at* v_i. We won't prove this (but it is not difficult).

computing optimal values

One way to compute the solution x^* for the Bellman equations $\mathbf{x} = L(\mathbf{x})$ is **value iteration**: consider the sequence $L(\mathbf{0})$, $L(L(\mathbf{0}))$, ..., $L^m(\mathbf{0})$. **Fact**: $\lim_{m\to\infty} L^m(\mathbf{0}) = x^*$.
Unfortunately, value iteration can be very slow in the worst case (requiring exponentially many iterations). Instead, we can use LP. Let $V = \{v_1, \ldots, v_n\}$ be the vertices of MDP, G. We have one LP variable x_i for each vertex $v_i \in V$.

Minimize $\sum_{i=1}^{n} x_i$

Subject to:

$x_T = 1$;

$x_i \geq x_j$, *for each* $v_i \in V_1$, *and* $v_j \in E(v_i)$;

$x_i = \sum_{v_j \in E(v_i)} q_{v_i}(v_j) \cdot x_j$, *for each* $v_i \in V_0$;

$x_i \geq 0$ *for* $i = 1, \ldots, n$.

Theorem *For* $(x_1^*, \ldots, x_n^*) \in \mathbb{R}^n$ *an optimal solution to this LP (which must exist), each* x_i^* *is the optimal value for player 1 in the game* G_{v_i}. (This follows from Bellman equations.)

extracting the optimal strategy

Suppose you computed the optimal values x^* for each vertex. How do you find an optimal (memoryless) strategy?
One way to find an optimal strategy for player 1 in this MDPs is to solve the **dual LP**.
First, remove all vertices v_i such that the maximum probability of reaching the target from v_T is 0. This is easy to do, by just doing reachability analysis on the underlying graph of the MDP, and ignoring probabilities.
Once this is done, it turns out that an optimal solution to the dual LP encodes an optimal strategy of player 1 in the MDP associated with the primal LP. And, furthermore, if you use Simplex, the optimal basic feasible solution to the dual will yield a pure strategy. (Too bad we don't have time to prove this.)

Stochastic Games

What if we introduce a second player in the game against nature?
In 1953 L. Shapley, one of the major figures in game theory, introduced "stochastic games", a general class of zero-sum, not-necessarily perfect info, two-player games which generalize MDPs. This was about the same time that Bellman and others were studying MDPs.
In Shapley's stochastic games, at each state, both players simultaneously and independently choose an action. Their joint actions yield both a 1-step reward, and a probability distribution on the next state.
We will confine ourselves to a restricted perfect information stochastic games where the objective is the probability of reaching a target.
These are callled "simple stochastic games" by [Condon'92].

simple stochastic games

Definition A zero-sum **simple stochastic game** is given by a game graph $G_{v_0} = (V, E, pl, q, v_0, u)$, where:

▷ V is a (finite) set of vertices.

▷ $pl : V \mapsto \{0, 1, 2\}$, maps each vertex to one of player 0 ("Nature"), player 1, or player 2.

▷ Let $V_0 = pl^{-1}(0)$, $V_1 = pl^{-1}(1)$, & $V_2 = pl^{-1}(2)$.

▷ $E : V \mapsto 2^V$ maps each vertex v to a set E(v) of "successors" (or "actions" at v).

▷ Let $V_{dead} = \{v \in V \mid E(v) = \emptyset\}$.

▷ For each "nature" vertex, $v \in V_0$, a probability distribution $q_v : E(v) \mapsto [0, 1]$, over the set of "actions" at v, such that $\sum_{v' \in E(v)} q_v(v') = 1$.

▷ A start vertex $v_0 \in V$.

▷ A target vertex $v_T \in V$.

memoryless determinacy

▷ The goal of player 1 is to *maximize* the probability of hitting the target state v_T.
▷ The goal of of player 2 is to *minimize* this probability. (So, the game is a zero-sum 2-player game.)
▷ We call the game *memorylessly determined* if both players have (deterministic) memoryless optimal strategies.

Theorem([Condon'92]) Every simple stochastic game is memorylessly determined.

computing optimal strategies

▷ Memoryless determinacy immediately gives us one algorithm for computing optimal strategies:

- "Guess" the strategy for one of the two players.
- The "residual game" is a MDP; solve corresponding LP.

▷ This gives a **NP** ∩ **co-NP** procedure for solving simple stochastic games.

▷ [Hoffman-Karp'66] studied a "strategy improvement algorithm" for stochastic games based on LP, which can be adapted to simple stochastic games ([Condon'92]).
Strategy improvement works well in practice, but recent results show that this algorithm requires exponential time for both MDPs and stochastic games, with SOME objective functions.

▷ Is there a P-time algorithm for solving simple stochastic games? This is an open problem.

▷ Solving parity games and mean payoff games (Lecture 13) can be reduced to solving SSGs ([Zwick-Paterson'96]).

food for thought

What is the relationship between computing Nash Equilibria in finite (two-person, n-person) strategic games and computing solutions to (simple) stochastic games?

In other words:

What does Nash have to do with Shapley?

To put it more concretely: is either computational problem efficiently reducible to the other?
ANSWER: we now know that both computing the value of Simple Stochatic Games, and approximating the (irrational) value of Shapley's Stochastic Games are reducible to computing a NE in 2-player strategic games.
(See [Etessami-Yannakakis,'07,SICOMP'10]).

Algorithmic Game Theory and Applications

Lecture 16: Selfish Network Routing, Congestion Games, and the Price of Anarchy

Kousha Etessami

games and the internet

- Basic idea: "The internet is a huge experiment in interaction between agents (both human and automated)".
- Such interactions can profitably be viewed from a game theoretic viewpoint: agents trying to maximize their own payoffs, etc.
- What are the implications of selfish behavior?
- How do we set up the rules of these games to harness "socially optimal" results?

(Selfish) Network Routing as a Game

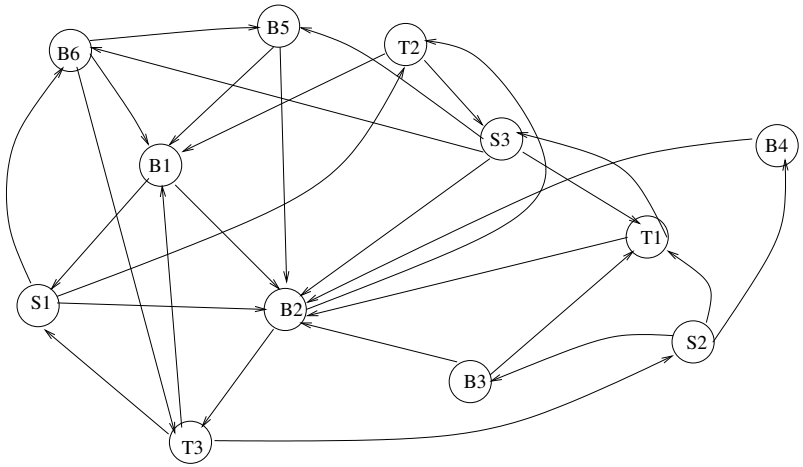


Figure: "The Internet"

▷ Selfish agent $i = 1, 2, 3$, wants to route packets from Si to Ti. So agent i must choose a **directed path** from Si to Ti.

▷ The **delay** on each edge of the path is governed by the **congestion** of that edge, i.e., the total number of agents using that edge in their path.

▷ Agents can change their choice to decrease their delay.

▷ What is a **Nash Equilibrium** in this game? What are the **welfare properties** of such an NE? (Is it socially optimal? If not, how bad can it be?)

Congestion Games ([Rosenthal,1973])

A **Congestion Game**, $G = (N, R, (Z_i)_{i \in N}, (d_r)_{r \in R})$ has:

▷ A finite set $N = \{1, \ldots, n\}$ of **players**.

▷ A finite set of $R = \{1, \ldots, m\}$ of **resources**.

▷ For each player, i, a set $Z_i \subseteq 2^R$, of admissible **strategies** for player i. So a pure strategy, $s_i \in Z_i$ is a set of resources.

▷ Each resource $r \in R$ has a **cost function**: $d_r : \mathbb{N} \to \mathbb{Z}$. Intuitively, $d_r(j)$ is the cost of using resource r if there are j agents simultaneously using r.

how mang player are using this resource

▷ For a pure strategy profile $s = (s_1, \ldots, s_n) \in Z_1 \times \ldots Z_n$, the **congestion** on resource r is: $n_r(s) \doteq |\{i \mid r \in s_i\}|$.

▷ Under strategy profile $s = (s_1, \ldots, s_n)$, the **total cost** to player i is:

$$C_i(s) \doteq \sum_{r \in s_i} d_r(n_r(s))$$

▷ Every player, i, wants to *minimize* its own (expected) cost.

Best response dynamics, and pure Nash Equilibria

In a congestion game G, for any pure strategy profile $s = (s_1, \ldots, s_n)$, suppose that some player i has a better alternative strategy, $s_i' \in Z_i$, such that $C_i(s_{-i}; s_i') < C_i(s)$.

Player i can switch (unilaterally) from s_i to s_i'. This takes us from profile s to profile (s_{-i}, s_i').

We call this a single (strict) improvement step.

Starting at an arbitrary pure strategy profile s, what happens if the players perform a sequence of such improvement steps?

Theorem: ([Rosenthal'73]) *In any congestion game, every sequence of strict improvement steps is necessarily finite, and terminates in a pure Nash Equilibrium.*
Thus, in particular, every congestion game has a pure strategy Nash Equilibrium.

Proof: Potential functions

Proof: Consider the following **potential function**:

$$\varphi(s) \doteq \sum_{r \in R} \sum_{i=1}^{n_r(s)} d_r(i) \tag{1}$$

What happens to the value of $\varphi(s)$ if player i switches unilaterally from s_i to s_i', taking profile s to $s' := (s_{-i}; s_i')$?

Claim: $\varphi(s) - \varphi(s') = C_i(s) - C_i(s')$.

Proof: Re-order the players in any arbitrary way, and index them as players $1, 2, \ldots, n$. (In particular, any player formerly indexed i could be re-indexed as n.) For $i' \in \{1, \ldots, n\}$, define

$$n_r^{(i')}(s) = |\{i \mid r \in s_i \wedge i \in \{1, \ldots, i'\}\}|$$

By exchanging the order of summation in equation (1) for $\varphi(s)$, it can be seen that (check this yourself):

make player i to the last player n

$$\varphi(s) = \sum_{i=1}^{n} \sum_{r \in s_i} d_r(n_r^{(i)}(s))$$

proof of claim, continued

Now note that $n_r^{(n)}(s) = n_r(s)$. Thus

$$\sum_{r \in s_n} d_r(n_r^{(n)}(s)) = \sum_{r \in s_n} d_r(n_r(s)) = C_n(s)$$

So, if player n switches from strategy s_n to s'_n, leading us from profile s to $s' = (s_{-n}; s'_n)$, then:
$\varphi(s) - \varphi(s') = C_n(s) - C_n(s')$.
But note that when re-ordering we could have chosen player n to be any player we want! So this holds for every player i. □

Proof of Theorem, continued

To complete the proof of Rosenthal's Theorem: Observe that every strict improvement step must decreases the value of the potential function $\varphi(s)$ by at least 1 (the costs $d_r(s)$ are all integers). Furthermore, there are only finitely many pure strategies s, so there are finite integers:
$a = \min_s \varphi(s)$ and $b = \max_s \varphi(s)$. Thus, every improvement sequence is finite.

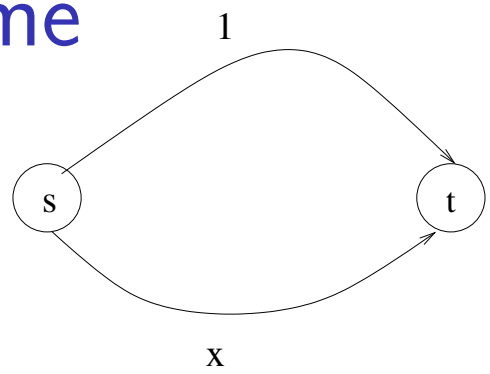
Finally, note that the last profile s in any improvement sequence which can not be further improved is, by definition, a pure Nash equilibrium.$\square$

Complexity of pure NE in network conges. games

Consider a **network congestion game** where we are given a network with source-sink node pairs (S_i, T_i), for each player i, and each player must to choose a route (path) from S_i to T_i. Suppose the cost (**delay**) of an edge, e, under profile s, is defined to be some *linear* function: $d_e(n_e(s)) = \alpha_e n_e(s) + \beta_e$.

One obvious way to compute a pure NE is to perform an arbitrary improvement sequence. However, this may conceivably require many improvement steps.
Is there a better algorithm?
It turns out that it is as hard as any **polynomial local search** problem to compute a pure NE for network congestion games:
Theorem: [Fabrikant et.al.'04, Ackermann et.al.'06].
Computing a pure NE for a network congestion game is **PLS-complete**, even when all edge delay functions, d_e, are *linear*.
So, unfortunately, a P-time algorithm is unlikely.

A flow network game



(from [Roughgarden-Tardos'00])

▷ n customers in network: each wants to go from s to t.

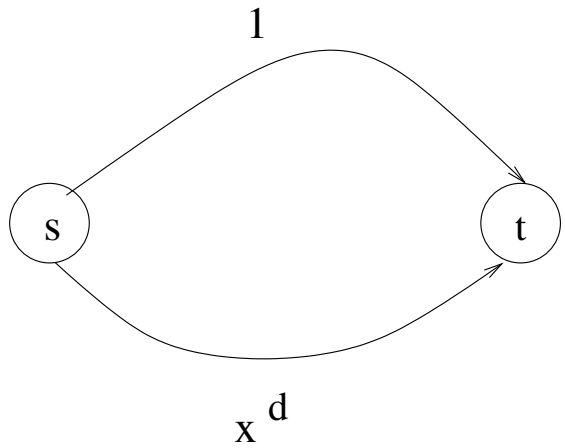
▷ Each can either take the edge with "latency" 1 (delay of crossing the edge), or edge with latency x. Here x represents the "congestion": the ratio of the number of customers that are using that edge divided by the total n.

▷ Assume n is **very large**, (basically, $n \to \infty$). What is the delay in Nash Equilibrium? (NEs in such a setting yield essentially a unique average delay [Beckmann, et. al. '56].) all go down

▷ What is a "globally optimal" customer routing strategy profile that minimizes average delay?

What is the globally optimal average delay? 1/2 ;1/2; 3/4

a modified game

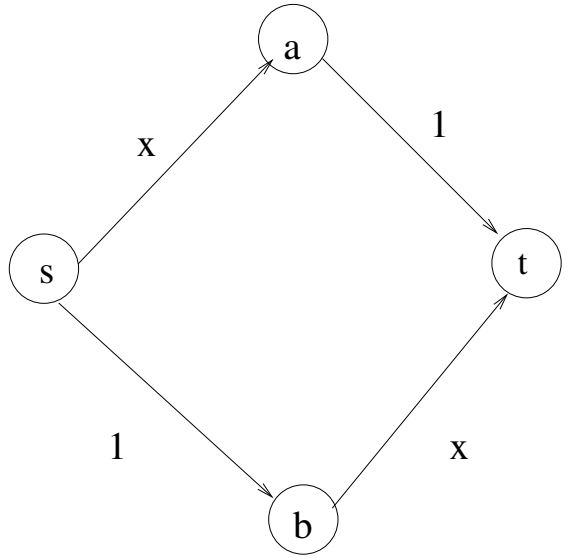


all go down;1

- What is the NE, and what is the average delay it induces?
- What is the globally optimal average delay?

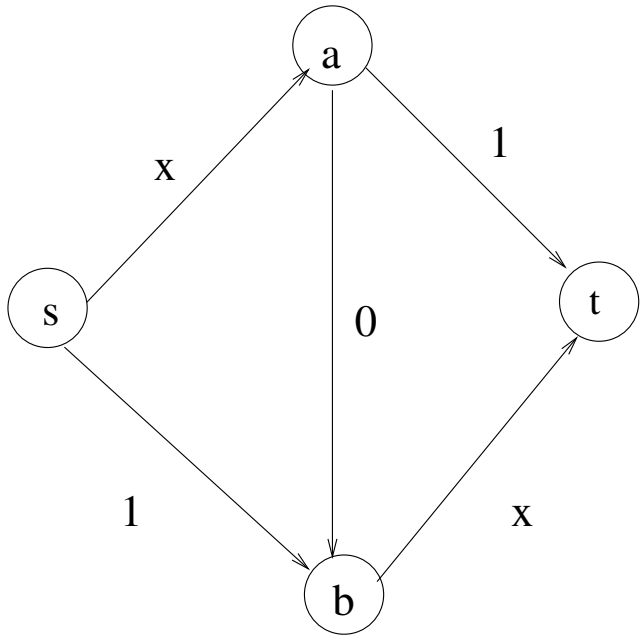
e; (1-e);e+(1-e)^d

a different network



- What is the NE, and what is its average delay? 1/2;1/2;3/2
- What is a globally optimal strategy profile and optimal average delay?
- What if an ambitious "network service provider" wanted to build an additional "high capacity superfast broadband" line?

Braess's paradox



all players: S->a->b->t
total cost: 2

- What is the NE and its average delay?
- What is the globally optimal average delay?

social welfare and the price of anarchy

Recall that in a strategic game Γ, "utilitarian social welfare", $welfare(x)$, under a particular profile of mixed strategies $x \in X$, is defined as $welfare(x) := \sum_{i=1}^{n} U_i(x)$. For a game Γ, let $NE(\Gamma)$ be the set of NE's of Γ.

For our next definition suppose $welfare(x) > 0$ for all $x \in X$. (In many games, we can enforce this by, e.g., adding a fixed value to all payoffs.)

A version of "the price of anarchy" can be defined as: ([Koutsoupias-Papadimitriou'98])

$$PA(\Gamma) := \frac{\max_{x \in X} welfare(x)}{\min_{x \in \mathrm{NE}(\Gamma)} welfare(x)}$$

Thus, the "price of anarchy" is the ratio of best "global" outcome to the the worst NE outcome. Note: this ratio is ≥ 1 and larger means "worse".

It would be comforting to establish that in various situations the "price of anarchy" isn't too high.

Pure price of anarchy

In some settings, such as congestion games, where we know that a pure equilibrium exists, it is sometimes more sensible to compare the best overall outcome to the worst pure-NE outcome.

Let pure-$NE(\Gamma)$ denote the set of pure NEs in the game Γ. For settings (such as congestion games) where we know pure-$NE(\Gamma)$ is non-empty, we define "the pure price of anarchy" as:

$$\text{pure-}PA(\Gamma) := \frac{\max_{s \in S} welfare(s)}{\min_{s \in \text{pure-NE}(\Gamma)} welfare(s)}$$

Thus, the "pure price of anarchy" is the ratio of best (pure) "global" outcome to the the worst pure NE outcome.

price of anarchy in the flow network game

- For flow f let $\mathit{welfare}(f) := 1/(\text{average s-t-delay})$.
- In Braess's paradox, the price of anarchy is $4/3$: by playing the NE the average delay is 2, but playing half-and-half on the upper and lower route, the average delay is $3/2$ (and that's optimal).
- We have seen that the price of anarchy in network games can be arbitrarily high, when x^d is an edge label.
- Remarkably, [Roughgarden-Tardos'00] showed that in a more general flow network setting (where there can be multiple source-destination pairs (s_j, t_j)), as long as "congestions" labeling edges are linear functions of x, the worst-case price of anarchy is $4/3$.
- In other words, for linear latencies, the Braess's paradox example yields the worst-case scenario.

Back to atomic network congestion games

By an "atomic" network congestion game, we simply mean a standard network congestion game with a *finite* number of players, where each aims to minimize its own cost. (Whereas in non-atomic network flow games the average cost in equilibrium is uniquely determined, this is not the case with atomic network congestion games.)
What is the (pure) price of anarchy in atomic network congestion games?

Theorem: [Christodoulou-Koutsoupias'2005]. *The pure price of anarchy for a pure NE in atomic network congestion games with linear utilities is*

$$5/2$$

(And this is tight, just like 4/3 for non-atomic network congestion games.)

Algorithmic Game Theory and Applications

Lecture 17:
A first look at Auctions and Mechanism Design:
Auctions as Games, Bayesian Games, Vickrey auctions

Kousha Etessami

Food for thought: sponsored search auctions

Question

How should advertisement slots on your Google search page be auctioned?

You may have already heard that Google uses a so-called "Generalized Second Price Auction" mechanism to do this.

But why do they do so? Is there any "better" way?
What does "better" actually mean? What is the goal of the auctioneer?
How should other (electronic) auctions be conducted?

More generally, how should we design games when our goal is to compel selfish players to behave in a desired way (e.g., a "socially optimal" way, or "truthfully"). This is the topic of Mechanism Design.

Auction theory and Mechanism Design are rich and vast subdisciplines of game/economic theory. We can not do them adequate justice. We briefly explore them in remaining lectures, focusing on algorithmic aspects.
Some reference reading: Chapters 9, 11, 12, 13, & 28 of AGT book.

Auctions as games

Consider one formulation of a single-item, sealed-bid, auction as a game:

- Each of n bidders is a player.
- Each player i has a valuation, $v_i \in \mathbb{R}$, for the item being auctioned.
- if the outcome is: player i wins the item and pays price pr, then the payoff to player i, is

$$u_i(outcome) := v_i - pr$$

  and all other players $j \neq i$ get payoff 0: $u_j(outcome) := 0$.
- Let us require that the auctioneer must set up the rules of the auction so that they satisfy the following reasonable constraints: given (sealed) bids $(b_1, \ldots, b_n)$, one of the highest bidders must win, and must pay a price pr such that $0 \leq pr \leq \max_i b_i$.
- Question: What rule should the auctioneer employ, so that for each player i, bidding their "true valuation" v_i (i.e., letting $b_i := v_i$) is a dominant strategy?

Vickrey auctions

- In a Vickrey auction, a.k.a., *second-price, sealed bid* auction, a highest bidder, j, whose bid is $b_j = \max_i b_i$, gets the item, but pays the second highest bid price: $pr = \max_{i \neq j} b_i$.

Claim

Bidding their true valuation, v_i, i.e., letting $b_i := v_i$, is a (weakly) dominant strategy in this game for all players i.

Let us prove this on the board. (We actually prove something stronger.)

- Note: there is *something very fishy/unsatisfactory* about our formulation so far of an auction as a *complete information* game: player i normally does not know the valuation v_j of other players $j \neq i$. But if viewed as a *complete information* game, then every player knows every one else's valuation. This is totally unrealistic!
- We need a better game-theoretic model for settings like auctions.

Bayesian Games (Games of Incomplete Information) [Harsanyi,'67,'68]

A **Bayesian Game**, $G = (N, (A_i)_{i \in N}, (T_i)_{i \in N}, (u_i)_{i \in N}, p)$, has:

- A (finite) set $N = \{1, \ldots, n\}$ of **players**.
- A (finite[1]) set A_i of **actions** for each player $i \in N$.
- A (finite[1]) set of **possible types**, T_i, for each player $i \in N$.
- A **payoff (utility) function**, for each player $i \in N$:

$$u_i : A_1 \times \ldots \times A_n \times T_1 \times \ldots \times T_n \to \mathbb{R}$$

private value of the item

- A **(joint) probability distribution over types**:

$$p : T_1 \times \ldots \times T_n \to [0, 1]$$

where, letting $T = T_1 \times \ldots \times T_n$, we must have:

$$\sum_{(t_1, \ldots, t_n) \in T} p(t_1, \ldots, t_n) = 1$$

p is sometimes called a **common prior**.

everyone know the common prior

[1] We can and do often remove the finiteness assumption on the type/action spaces, e.g., letting T_i be a closed interval $[a, b] \subseteq \mathbb{R}$.

strategies and expected payoffs in Bayesian games

- A pure strategy for player i is a function $s_i : T_i \to A_i$.
  I.e., player i knows its own type, t_i, and chooses action $s_i(t_i) \in A_i$.
  (In a mixed strategy, x_i, $x_i(t_i)$ is a probability distribution over A_i.)
- Players' types are chosen randomly according to (joint) distribution p.
- Player i knows $t_i \in T_i$, but doesn't know the type t_j of players $j \neq i$.
- But every player "knows" the joint distribution p (this is often too strong an assumption, dubiously so, but it is useful), so each player i can compute the conditional probabilities, $p(t_{-i} \mid t_i)$, on other player's types, given its own type t_i.
- The **expected payoff** to player i, under the pure profile $s = (s_1, \ldots, s_n)$, when player i has type t_i (which it knows) is:

$$U_i(s, t_i) = \sum_{t_{-i}} p(t_{-i} \mid t_i) u_i(s_1(t_1), \ldots, s_n(t_n), t_i, t_{-i})$$

Bayesian Nash Equilibrium

Definition

A strategy profile $s = (s_1, \ldots, s_n)$ is a (pure) Bayesian Nash equilibrium (BNE) if for all players i and all types $t_i \in T_i$, and all strategies s_i' for player i, we have: $U_i(s, t_i) \geq U_i((s_i'; s_{-i}), t_i)$.
*(A **mixed** BNE is defined similarly, by allowing players to use mixed strategies, x_i, and defining expected payoffs $U_i(x, t_i)$ accordingly.)*

Proposition

Every finite Bayesian Game has a mixed strategy BNE.

Proof

Follows from Nash's Theorem. Every finite Bayesian Games can be encoded as a finite extensive form game of imperfect information: the game tree first randomly choose a subtree labeled $(t_1, \ldots, t_n)$, with probability $p(t_1, \ldots, t_n)$. All nodes belonging to player i in different subtrees labeled by the same type $t_i \in T_i$ are in the same information set.

Back to Vickrey auctions

Now suppose we model a sealed-bid single-item auction using a Bayesian game, with some arbitrary prior probability distribution $p(v_1, \ldots, v_n)$ over valuations (suppose every v_i is in some finite nonnegative range $[0, v_{max}]$). The private information of each player i is $t_i := v_i$.

Proposition

In the Vickrey (second-price, sealed-bid) auction game, with any prior p, the truth revealing profile of bids $v = (v_1, \ldots, v_n)$, is a weakly dominant strategy profile.
(In fact, under suitable conditions on p, v is the unique BNE of the game.)

Proof

The exact same proof as for the complete information version of the vickrey game works to show that v is a weakly dominant strategy profile. Indeed, we didn't use the fact that the game has complete information. We didn't even assume the players have a common prior for this!

What about other auctions?

- In first-price, sealed-bid auctions, (maximum bidder gets the item and pays $pr = \max_i b_i$), bidding truthfully may indeed not be a dominant strategy. E.g., if player i knows (with high probability) that its valuation $v_i >> v_j$ for all $j \neq i$, then i will not "waste money" and will instead bid $b_i << v_i$, since it will still get the item.
- What implications does this have for the expected revenue of the auction? Surprisingly, it has less than you might think:

Proposition

If prior p is a product of i.i.d. uniform distributions over some interval $[0, v_{max}]$, then the expected revenue of the second-price and first-price sealed-bid auctions are both the same in their (unique) symmetric BNEs.

This is actually a special case of a much more general result in Mechanism Design called the Revenue Equivalence Principle. (But anyway, note that one-shot revenue maximization is not always a wise goal for an auctioneer.)

The bigger picture

- This was our first look at Auctions (with hints of Mechanism design), in the simple setting of a single-item, sealed-bid, auction.
- To delve deeper into (algorithmic aspects of) Mechanism Design and auctions, we need a deeper understanding of the "bigger picture":
- Starting next time, we will step back to glimpse at where all this fits in the broader scope of Economic Theory.
- In particular, we will briefly discuss some social choice theory, and also Market equilibria, before returning to the important VCG mechanism, which vastly generalizes the Vickrey single-item auction.
- Some of the topics we will discuss are beyond the scope of this AGT course, and can be properly learned, e.g., in a good Microeconomic Theory course/text (e.g., [Mas-Colell-Whinston-Green'95]).
- However, I feel I would be "cheating you" if I didn't at least provide some (necessarily impressionistic) sense of the "bigger picture".

Algorithmic Game Theory and Applications

Lecture 18: Auctions and Mechanism Design II: a little social choice theory, the VCG Mechanism, and Market Equilibria

Kousha Etessami

Reminder: Food for Thought: sponsored search auctions

Question

How should advertisement slots on your Google search page be auctioned?

You may have already heard that Google uses a so-called "Generalized Second Price Auction" mechanism to do this.

But why do they do so? Is there any "better" way?

Before we can even begin to address such questions (which are in fact largely outside the scope of this course), it is useful to get a better understanding of the broader picture in economic theory within which auction theory and mechanism design operate.
Again, reference reading for today's lecture includes: Chapters 9, 11, 12, 13, & 28 of AGT book.
Also very useful are the revelant chapters of the excellent textbook:
A. Mas-Colell, M. Whinston, & J. Green, *Microeconomic Theory*, 1995.

Walrasian Market Equilibrium

An Exchange Economy

- n agents, and m divisible goods (commodities, serices, etc.).
- Each agent i starts with an initial endowment of goods, $w_i = (w_{i,1}, \ldots, w_{i,m}) \in \mathbb{R}^m_{\geq 0}$.
- Each agent i has a utility function, $u_i(x)$, that defines how much it "likes" a given bundle $x \in \mathbb{R}^m_{\geq 0}$ of goods.
  Assume utility functions satisfy certain "reasonable" conditions:
  - (1) Continuity; (2) Quasi-concavity; (3) Non-satiation
- Assume agent endowments satisfy certain "reasonable conditions".
  E.g., each agent has a positive endowment of every good. (Much less suffices: e.g., each agent has something another agent "wants", and the graph of this is strongly-connected.)
- Given a price vector, $p \in \mathbb{R}^m_{\geq 0}$, each agent i has an optimal demand set, $D_i(p) \subseteq \mathbb{R}^m_{\geq 0}$. This is the set of bundles of goods, x_i, that maximize agent i's utility, and which i can afford to buy at prices p by selling its endowment w_i at prices p.

Background: existence of market equilibrium

Market (price) equilibrium for an exchange economy

A non-zero price vector, $p^* \geq 0$, together with bundles, x_i^*, of goods for each agent i, constitutes a market (price) equilibrium for an exchange economy, if at prices p^*, for every agent i, x_i^* is an optimal demand bundle, and moreover with these demands the market clears (i.e., basically supply = demand). In other words:

- For all agents i, $x_i^* \in D_i(p^*)$. (All agents optimize utility at prices p^*)
- $\sum_i x_i^* \leq \sum_i w_i$. (Demands do not exceed supply for any commodity.)
- Furthermore, for all goods j, if $(\sum_i x_i^*)_j < (\sum_i w_i)_j$, then $p_j^* = 0$.

  (In other words, only free goods can have excess supply.)

Background: existence of market equilibrium

Market (price) equilibrium for an exchange economy

A non-zero price vector, $p^* \geq 0$, together with bundles, x_i^*, of goods for each agent i, constitutes a market (price) equilibrium for an exchange economy, if at prices p^*, for every agent i, x_i^* is an optimal demand bundle, and moreover with these demands the market clears (i.e., basically supply = demand). In other words:

- For all agents i, $x_i^* \in D_i(p^*)$. (All agents optimize utility at prices p^*)
- $\sum_i x_i^* \leq \sum_i w_i$. (Demands do not exceed supply for any commodity.)
- Furthermore, for all goods j, if $(\sum_i x_i^*)_j < (\sum_i w_i)_j$, then $p_j^* = 0$.

  (In other words, only free goods can have excess supply.)

Theorem ([Arrow-Debreu, 1954])

Every exchange economy has a market (price) equilibrium.

Background: existence of market equilibrium

Market (price) equilibrium for an exchange economy

A non-zero price vector, $p^* \geq 0$, together with bundles, x_i^*, of goods for each agent i, constitutes a market (price) equilibrium for an exchange economy, if at prices p^*, for every agent i, x_i^* is an optimal demand bundle, and moreover with these demands the market clears (i.e., basically supply = demand). In other words:

- For all agents i, $x_i^* \in D_i(p^*)$. (All agents optimize utility at prices p^*)
- $\sum_i x_i^* \leq \sum_i w_i$. (Demands do not exceed supply for any commodity.)
- Furthermore, for all goods j, if $(\sum_i x_i^*)_j < (\sum_i w_i)_j$, then $p_j^* = 0$.

  (In other words, only free goods can have excess supply.)

Theorem ([Arrow-Debreu, 1954])

Every exchange economy has a market (price) equilibrium.

Their proof is non-algorithmic. It crucially uses a fixed point theorem.

two quotes

Kamal Jain (MSR), 2006: "If your laptop can't find it, then neither can the market."

Mas-Colell,Whinston, & Green, 1995 (standard graduate text in Microeconomic Theory): "A characteristic feature [of] economics is that for us the equations of equilibrium constitute the center of our discipline. By contrast, other sciences put more emphasis on the dynamic laws of change. The reason... is that economists are good at recognizing a state of equilibrium, but are poor at predicting precisely how an economy in disequilibrium will evolve..."

Or, to paraphrase: "Modern economic theory is largely non-algorithmic."
By contrast, computer science is very good at "dynamics" (algorithmics).

A sizable portion of Algorithmic Game Theory research, broadly speaking, aims to remedy this difficiency of modern economic theory.

A side question: Do we really need money?

A side question: Do we really need money?

- Note that money is itself extraneous to the goals of agents in an exchange economy. Why do we need it?
- Couldn't we achieve the same "goals" of the market without money?
- In other words, couldn't we organize a "barter economy" to conduct a simultaneous exchange of goods between all agents, in a way that achieves Pareto optimal bundles of goods for every agent?

Recall: Pareto optimal means that there is no other reallocation of everyone's goods in which everybody's utility is at least as high, and somebody's utility is strictly higher.

Note: the The First Fundamental Theorem of Welfare Economics says that allocations of goods in any market equilibrium are Pareto optimal.

A side question: Do we really need money?

- Note that money is itself extraneous to the goals of agents in an exchange economy. Why do we need it?
- Couldn't we achieve the same "goals" of the market without money?
- In other words, couldn't we organize a "barter economy" to conduct a simultaneous exchange of goods between all agents, in a way that achieves Pareto optimal bundles of goods for every agent?

Recall: Pareto optimal means that there is no other reallocation of everyone's goods in which everybody's utility is at least as high, and somebody's utility is strictly higher.

Note: the The First Fundamental Theorem of Welfare Economics says that allocations of goods in any market equilibrium are Pareto optimal.

It is intuitively clear that such an exchange is difficult to do without money.

We shall see that some things become impossible to do without money.

Background: Impossibility theorems in social choice theory

- Let C be a set of outcomes, and let L be the set of total orderings on C. A social choice function is a function $f : L^n \to C$.
- f can be strategically manipulated by player (voter) i, if for some $\prec_1, \ldots, \prec_n \in L$, we have $c \prec_i c'$, and $c = f(\prec_1, \ldots, \prec_n)$, and there exists some other ordering $\prec_i'$ such that $c' = f(\prec_1, \ldots, \prec_i', \ldots, \prec_n)$.
- f is called incentive compatible (strategy-proof) if it can not be strategically manipulated by anyone.
- f is a dictatorship if there is some player i such that for all $\prec_1, \ldots, \prec_n$, $f(\prec_1, \ldots, \prec_n) = c_i$, where c_i is the maximum outcome for player i, i.e., $\forall b \neq c_i,\ b \prec_i c_i$.

Theorem [Gibbard-Satterthwaite'73,'75]

If $|C| \geq 3$, and $f : L^n \mapsto C$ is an incentive compatible social choice function which is onto C (i.e., all outcomes are possible), then f is a dictatorship.

Arrow's Theorem

The Gibbard-Satterthwaite Theorem is closely related to the following famous theorem:

Arrow's Impossibility Theorem

There is no "really good" way to aggregate people's preference orders (over ≥ 3 alternatives) into a societal preference order.

"Really good" here means it should satisfy the following three criteria:

- *Unanimity*: if everybody has the same preference order, then that should become the societal preference order.
- *Independence of irrelevant alternatives:* The societal preference between any pair of outcomes a and b should depend only on the preference between a and b of all individuals, and not on their preferences for other alternatives $c \neq a, b$.
- *Non-dictatorship:* no individual's preferences should dictate the societies preferences.

Let's bring money back into the picture

- Let V be the set of "voters".
- Suppose that instead of a "preference ordering", each voter $i \in V$ has a (non-negative) "monetary value function": $v_i : C \to \mathbb{R}_{\geq 0}$, where $v_i(c)$ says how much money a win by candidate c is worth for voter i.
- Suppose we want to choose a candidate c^* that maximizes the total value for the entire voter population, i.e., we want to choose:

$$f(v_1, \ldots, v_n) = c^* \in \arg\max_c \sum_{i \in V} v_i(c)$$

- But voters could be dishonest and lie about their true value function. (Voter i can declare a different non-negative valuation function v_i'.)
- *Question:* Can we incentivize the voters to tell the truth?

Let's bring money back into the picture

- Let V be the set of "voters".
- Suppose that instead of a "preference ordering", each voter $i \in V$ has a (non-negative) "monetary value function": $v_i : C \to \mathbb{R}_{\geq 0}$, where $v_i(c)$ says how much money a win by candidate c is worth for voter i.
- Suppose we want to choose a candidate c^* that maximizes the total value for the entire voter population, i.e., we want to choose:

$$f(v_1, \ldots, v_n) = c^* \in \arg\max_c \sum_{i \in V} v_i(c)$$

- But voters could be dishonest and lie about their true value function. (Voter i can declare a different non-negative valuation function v_i'.)
- *Question:* Can we incentivize the voters to tell the truth?
- Yes, if we are allowed to ask them to pay money!

The Vickrey-Clarke-Groves (VCG) Mechanism

- Each voter $i \in V$ is asked to submit their nonnegative valuation function, v_i'. (v_i' may or may not be i's actual valuation function v_i.)
- The mechanism then computes an optimal candidate, $f(v_1', \ldots, v_n') := c^*$, that maximizes total value:

$$f(v_1', \ldots, v_n') := c^* \in \arg\max_c \sum_{k \in V} v_k'(c)$$

  This candidate c^* is chosen as the winner of the election.
- Moreover, each voter i has to pay an amount $p_i(c^*)$ to the mechanism, which is independent of v_i', defined by:

$$p_i(c^*) := (\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v_j'(c')) - \sum_{j \in V \setminus \{i\}} v_j'(c^*)$$

Key Point: These payments align the incentives of all voters: each voter i, even knowing the valuation functions v_j' declared by voters $j \neq i$, will want to declare a function v_i' that yields a winner $c^* = f(v_1', \ldots, v_n')$ which maximizes $v_i(c^*) + \sum_{k \in V \setminus \{i\}} v_k'(c^*)$. So, it should declare $v_i' := v_i$.

Basic properties of VCG

Proposition

Let $f(v_1', \ldots, v_n') = c^*$ be the outcome chosen as the winner by the VCG mechanism, based on their declared (non-negative) valuation functions $v_1', \ldots, v_n'$. Then for every voter i both its payment $p_i(c^*)$ and its purported utility $u_i'(c^*) = v_i'(c^*) - p_i(c^*)$ are non-negative, i.e.:

$$1.) \quad p_i(c^*) \geq 0 \quad \text{and} \quad 2.) \quad u_i'(c^*) := v_i'(c^*) - p_i(c^*) \geq 0$$

Proof: 1.) $p_i(c^*) = (\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v_j'(c')) - \sum_{j \in V \setminus \{i\}} v_j'(c^*) \geq 0.$

2.) $$\begin{aligned} v_i'(c^*) - p_i(c^*) &= (\sum_{j \in V} v_j'(c^*)) - (\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v_j'(c')) \\ &= (max_{c \in C} \sum_{i \in V} v_i'(c)) - (\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v_j'(c')) \geq 0 \end{aligned}$$

(The last ≥ 0 holds because we require $v_j'(c) \geq 0$ for all j and all $c \in C$.)

VCG is incentive-compatible (i.e., strategy-proof)

Theorem ([Vickrey, 1961],[Clarke, 1971], [Groves,1973])

The VCG mechanism is incentive compatible (i.e., strategy proof). In other words, declaring their true valuation function v_i is a (weakly) dominant strategy for all players i.

Proof: Suppose players declare valuations $v'_1, \ldots, v'_n$. Suppose player i's true valuation is v_i. Let $c^* = f(v_i, v'_{-i})$, and let $c'' = f(v'_i, v'_{-i})$. Recall $c^* \in \arg\max_c v_i(c) + \sum_{k \in V \setminus \{i\}} v'_k(c)$. In other words, for all $c \in C$, $v_i(c^*) + \sum_{k \in V \setminus \{i\}} v'_k(c^*) \geq v_i(c) + \sum_{k \in V \setminus \{i\}} v'_k(c)$. Thus, $u_i(c^*) =$

$$
\begin{aligned}
v_i(c^*) - p_i(c^*) &= v_i(c^*) + \Big(\sum_{k \in V \setminus \{i\}} v'_k(c^*)\Big) - \Big(\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v'_j(c')\Big) \\
&\geq v_i(c'') + \Big(\sum_{k \in V \setminus \{i\}} v'_k(c'')\Big) - \Big(\max_{c' \in C} \sum_{j \in V \setminus \{i\}} v'_j(c')\Big) \\
&= v_i(c'') - p_i(c'') = u_i(c''). \quad \square
\end{aligned}
$$

Example: Vickrey's second-price sealed-bid auction

- Recall: There is a single item being auctioned.
- Each bidder, i, declares its own monetary value (price), b_i, for getting the item.
- The highest bidder, say i^*, gets the item, and pays the second highest bid price, say $p^* = b_j$.
- The overall utility for bidder i^* who got the item is $v_{i^*} - p^*$.
  Both the valuation and payment (and thus utility) for all others who don't get it is 0.
- Recall: **Theorem:** the Vickrey auction is incentive compatible: every bidder i's dominant strategy is to declare its true valuation v_i.
- In fact, this is precisely the VCG mechanism, applied in the setting of a single-item auction, where the set of "candidates", now "outcomes", is given by "who gets the item?".
  The machanism makes sure that the item goes to the person who values it most, because that maximizes $\sum_i v_i(outcome)$.

An ascending price auction closely related to Vickrey's

- In practice, Vickrey's sealed-bid auction are often not used.
- There are a variety of (good) reasons for this. (For example, bidder valuation may not be independent. This leads, e.g., to Winner's curse. Thus information about other people's valuation matters a lot.)
- Instead, often an Ascending price (English) auction is used:
  - The price starts at a low reserve price, and is repeatedly incremented by a small amount, $\epsilon > 0$, until all but one of the bidders drops out saying "I don't want it at that high a price".
  - At that point, the one remaining bidder gets the item, at the last price.
- It is intuitively clear that if we model these auctions as (Bayesian) games with private valuations then the outcome of the English auction should be "essentially the same" as that of the Vickrey auction.
- Things change a lot if we model them with interdependent valuations. (A model we haven't discussed: players don't know exactly their own valuation. Their valuation depends on everybody's private signal.). That is one reason why an English auction is often preferred.

Why not use VCG for multi-item auctions?

- Suppose there is a set J of items, with $|J| = k > 1$, up for auction.
- Suppose each bidder i is "single-minded" and values only one subset $J_i \subseteq J$ of the items, so he has a valuation $v_i(J') = v_i^* > 0$ for any $J' \supseteq J_i$, and $v_i(J') = 0$ for all $J' \not\supseteq J_j$. (So, we can specify valuation v_i by just giving (v_i^*, J_i).)
- Note: if $J_i \cap J_{i'} \neq \emptyset$, then auction can't satisfy both bidders i and i'.
- Suppose, we want the auction outcome to maximize total value for all bidders.
- Why not use the VCG mechanism?
- A problem: for VCG we have to compute an outcome, c^*, that maximizes the total value $\sum_i v_i(c^*)$.
- This is NP-hard in general, even when we always have $v_i^* = 1$: easily encodes maximum independent set.

Proof of NP-hardness

Here is the simple reduction from the maximum independent set problem:

Let $G = (V, E)$ be any graph, with $V = \{s_1, \ldots, s_n\}$.

We associate with each edge $e \in E$ an item in the auction.

We associate a player (bidder), i, with every vertex $s_i \in V$.

We let J_i be the set of edges adjacent to vertex v_i.

We let $v_i^* = 1$ for all i (i.e., every player has value 1 for its desired set).

Then finding an outcome c^* of the auction (i.e., a collection of disjoint sets J_i, indicating those bidders who get their desired set) such that this outcome maximizes $\sum_i v_i(c^*)$ is equivalent to finding a maximum independent set in the graph G.

Algorithmic Mechanism Design

- So, achieving incentive compatibility by means of VCG may involve solving NP-hard problems.
- Why not try the standard approximation approach used for coping with NP-hardness?
  Problem: if the mechanism outputs only an approximate solution, and yet tries to compute VCG-like prices for it, then incentive compability totally breaks down.
- In some auction settings (like single-minded bidders), it is possible to design non-VCG mechanisms which are: (1) incentive compatible, (2) where prices and allocations are polynomial-time computable, and (3) such that the outcome yields a total valuation within the best possible approximation ratio of optimal (any better approximation ratio would be NP-hard).
- These are the kinds of problems addressed by *Algorithmic Mechanism Design*.

next time

- There are other multi-item auction settings where the (exact) VCG mechanism remains useful and can be efficiently computed.

- One important and beautiful example of this is Matching markets (multi-item unit-demand auctions).

  The setting of matching markets also gets us much closer to the setting of sponsored search auctions (also known as position auctions). These can be viewed as special subcases of matching markets and unit-demand auctions.

- We will discuss these next time.

- We will then end our lectures by presenting a general formal framework for Mechanism Design.

Algorithmic Game Theory and Applications

Lecture 19: Auctions and Mechanism Design III: Matching Markets, unit-demand auctions, and VCG; Myerson's revenue-optimal auctions; and a Formal Framework of Mechanism Design

Kousha Etessami

Matching markets: multi-item unit-demand auctions

(Now start thinking of, e.g., Google's Sponsored Search Auctions.)

- A set B of n bidders (Advertisers);
  A set Q, of k items (Advertisement slots on your Google page).
- Each bidder wants to buy at most one item (at most one ad on a web page),
  i.e., each bidder has unit demand.
- Each bidder $i \in B$ has a valuation, $v_{i,j} \geq 0$, for each item j.
- Each item j has a reserve price, $r_j \geq 0$, below which it won't be sold.
- A (feasible) price vector is any vector $p \geq r$.
- At prices p, the demand set of bidder i is
  $D_i(p) := \{j \in Q \mid v_{i,j} - p_j = \max_{j' \in Q}(v_{i,j'} - p_{j'}) \geq 0\}$.
- The payoff (utility), u_i, to a bidder i who gets item j and pays p_j is $u_i = v_{i,j} - p_j$. (And $u_i = 0$ if bidder i gets nothing.)

Market equilibrium in matching markets

- A partial matching, μ, is a partial one-to-one function from B to Q.
- A price vector p, together with a partial matching μ, is called a Market price equilibrium, (p, μ), if:
  - For all matched $i \in B$, $\mu(i) \in D_i(p)$.
  - If $i \in B$ is unmatched in μ, then either $D_i(p) = \emptyset$ or for all $j \in D_i(p)$, $v_{i,j} - p_j = 0$.
  - For any unmatched item $j \notin \mu(B)$, $p_j = r_j$.

Theorem ([Shapley-Shubik,1971])

1. *A market equilibrium always exists.*
2. *The set P of market equilibrium price vectors p form a lattice in $\mathbb{R}^k_{\geq 0}$.*

The least element of this price lattice has very special properties.
(For maximizing revenue, sellers would prefer the greatest element, but...)

Computing market equilibrium: an ascending price auction

[Demange-Gale-Sotomayor,1986]

1. Assume (only for simplicity) that valuations $v_{i,j}$ are all integers.
2. Set the initial price vector, p^0, to the reserve prices, $p^0 := r$.
3. At round t, when current prices are p^t, each bidder i declares their demand set, $D_i(p^t)$.
4. Does there exist an overdemanded set of items, $S \subseteq Q$, i.e., where $0 < |S| < |\{i \in B \mid \emptyset \neq D_i(p^t) \subseteq S\}|$? If not, STOP: By Hall's theorem, a market equilibrium at prices p^t exists (and a corresponding matching can be found by maximum matching).
5. If so, find a minimal over-demanded set S; $\forall\, j \in S$ let $p_j^{t+1} := p_j^t + 1$; $\forall\, k \in Q \setminus S$, let $p_k^{t+1} := p_k^t$; let $t := t + 1$; go to step 3.

Theorem ([DGS'86])

This auction algorithm always finds a market equilibrium. Moreover, the equilibrium it finds is the least price equilibrium.

Some algorithmic considerations

- Can we compute this auction outcome efficiently? Yes.
- We can compute (minimal) over-demanded sets in P-time.
- We don't need to increment prices by 1 in each iteration. (We can raise all prices in the over-demanded set until the demand set of one of the respective bidders enlarges beyond S.)
- It turns out that this is essentially isomorphic to the Primal-Dual Hungarian Algorithm for finding a maximum-weight matching in a weighted bipartite graph [Kuhn'55], which runs in P-time.

More lovely facts about this lovely auction

- As we said, the DGS auction always computes the minimum price equilibrium, p^*.
- In fact, it turns out that the payments, p^*, are precisely the VCG payments!
- Therefore, the mechanism is incentive compatible, and for every bidder it is a dominant strategy to specify its true valuations.
- Moreover, this mechanism is even group strategy-proof, meaning that no strict subset of bidders who can collude have an incentive to misrepresent their true valuations.

Generalized Second Price (GSP) auctions

The GSP auction looks somewhat similar to Vickrey's second-price auction.

1. Assume there are k advertisement slots on the web page.
   We assume there is a preference total ordering on the advertisement slots $1, \ldots, k$: all bidders prefer slot 1 to 2, slot 2 to 3, etc.
   (It can be assumed (for simplicity) that associated with each slot j is a clickthrough rate, α_j, and that for $j < j'$, we have $\alpha_j \geq \alpha_{j'}$.)
2. Each advertiser i, chooses a single bid value, $b_i \in \mathbb{R}_{\geq 0}$. (This is a bid associated with a click on their ad.)
3. The advertisers are ordered (from highest to smallest) based on their bid b_i. Let $b_{(j)}$ denote the bid that is ranked j'th.
4. If advertiser i is ranked $j < k$, then it gets its ad in position j, and pays, per click, $b_{(j+1)}$, the bid by the advertiser in position $j + 1$.
5. If there are fewer bidders than slots, then the advertiser in the last position pays a reserve price r (per click).

What are the incentive properties of the GSP auction?

- It is not quite VCG (but it is related).
  Given equivalent bids by all bidders, GSP payments are at least as high (or higher) than VCG.
- Truth telling is not a dominant strategy under GSP.
- But ([Edelman-Ostrovsky-Schwarz'07],[Varian'07]) if bidding is modeled as a repeated game, then it is argued bids will eventually reach an locally envy-free equilibrium. Such equilibria of the GSP auction map directly to market equilibria of a matching market.
- Thus, we again have a lattice structure to price equilibria, and there is a minimum equilibrium in which payoffs correspond precisely to VCG. (But it is not a dominant strategy of GSP, just one equilibrium.)
- If we end up at any other equilibrium, the revenue of the seller is higher than VCG. (Not bad for Google.)

Complications: budget limits

- In actual GSP ad auctions, bidders are asked to submit not just a value bid saying how much they value a clickthrough, but also a budget limit, which specifies the maximum they are willing to actually pay (which may be strictly less).
- This creates a discontinuity in the player's utility function, and in fact (in degenerate cases), market equilibria need not exist.
- However, when bids and budgets satisfy certain basic non-degeneracy conditions: (1) market equilibria do exist, (2) the DGS auction algorithm works, and finds incentive compatible prices and allocations.

What if our objective was revenue maximization?

Suppose we are interested in devising a **single-item auction** to maximize (expected) revenue. Suppose each bidder i's valuation $v_i \geq 0$ is independently distributed according to a distribution D_i. Suppose each bidder i submits a bid $b_i \geq 0$ for the item. We want the auction to satisfy:

- **non-negative payments:** the price p_i that each bidder i is asked to pay is non-negative: $p_i \geq 0$.
- **individual rationality:** $b_i - p_i \geq 0$, for all bidders i.
  (A (sub-)truthful agent gets non-negative utility.)
- **Incentive compatibility:** For every bidder, i, it is a dominant strategy to bid their true valuation, $b_i := v_i$.

Vickrey's auction of course satifies these criteria. Suppose our goal is to find an auction satisfying these criteria that maximizes the **expected revenue**: the expected value of the selling price.

Question: Can we do this?

What if our objective was revenue maximization?

Suppose we are interested in devising a **single-item auction** to maximize (expected) revenue. Suppose each bidder i's valuation $v_i \geq 0$ is independently distributed according to a distribution D_i. Suppose each bidder i submits a bid $b_i \geq 0$ for the item. We want the auction to satisfy:

- **non-negative payments:** the price p_i that each bidder i is asked to pay is non-negative: $p_i \geq 0$.
- **individual rationality:** $b_i - p_i \geq 0$, for all bidders i.
  (A (sub-)truthful agent gets non-negative utility.)
- **Incentive compatibility:** For every bidder, i, it is a dominant strategy to bid their true valuation, $b_i := v_i$.

Vickrey's auction of course satifies these criteria. Suppose our goal is to find an auction satisfying these criteria that maximizes the **expected revenue**: the expected value of the selling price.

Question: Can we do this? Yes! Under suitable assumptions on distributions D_i ([Myerson'81]).

A very limited special case of Myerson's Theorem

Theorem (a very limited special case of [Myerson,1981])

Consider a single-item auction with $n \geq 1$ bidders, where all bidders have independent valuations, each uniformly distributed in the interval $[0, v_{\max}]$, and each bidder i submits a bid $b_i \geq 0$.
Among all individually-rational auctions with non-negative payments that are incentive compatible, the following auction is the unique one that **maximizes expected revenue**:

- *Set the* **reserve price** *to $r := \frac{v_{\max}}{2}$.*
- *Sell the item to the highest bidder if the highest bid price is $\geq r$. Otherwise, don't sell the item.*
- *If sold, charge that highest bidder the following price:*

$$\max\{\text{the second highest bid price}, r\}.$$

For **multi-item auctions**, revenue maximization is a **much** more difficult challenge (including from a computational complexity perspective: NP-hard even in very very simple settings).

A formal framework for mechanism design

Formally, the input to a mechanism design problem can be specified by giving three things:

- An **environment**, $\mathcal{E}$, in which the game designer operates.
- A **choice function**, f, which describes the designer's preferred outcomes in the given environment.
- A **solution concept**, *Sol*, which describes the kinds of strategy profiles of games (such as their (Bayes) Nash Equilibria) which will be considered "solutions" in which the desired choice function should be implemented.

We describe each of these three inputs separately, using single-item auctions as a running example to illustrate each concept.

Once this is done, we will be able to state the mechanism design problem more formally.

environments

An **environment**, $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$, consists of:

- A set $N = \{1, \ldots, n\}$ of players.
- A set C of "outcomes".
  auction example: $C = \{(i, \mathrm{pr}) \in N \times \mathbb{R}_{\geq 0}\}$, where outcome (i, pr) means player i wins and pays pr.
- A cartesian product $\Theta = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_n$, of types where each Θ_i is a set of possible types for player i.
- We assume that the type $\theta_i \in \Theta_i$ of player i determines its "utility function" over outcomes. So, we have $u_i : \Theta_i \times C \to \mathbb{R}$.
  auction example:
  $$u_i(v_i, (i', \mathrm{pr})) = \begin{cases} 0 & \text{if } i \neq i' \\ v_i - \mathrm{pr} & \text{if } i = i' \end{cases}$$
  Note: the "type", v_i, of player i determines its utility function, u_i.
- $p : \Theta \to [0, 1]$ is a common prior joint distribution on types of players.
- $\mathcal{M}$ is a set of "Mechanisms"... Question: What is a Mechanism?

What is a Mechanism?

- Each Mechanism, $M \in \mathcal{M}$, has the form $M = (A_1, \ldots, A_n, g)$, where A_i is a set of actions for player i, and $g : A_1 \times \ldots \times A_n \to C$, is a outcome function from strategy profiles to outcomes.
  auction example: the action sets A_i are the possible bids that each player i could bid, and g gives a function that maps a profile of bids to an "outcome" of who wins the item and at what price. Such outcome functions can be constrained by the definition of the set $\mathcal{M}$. For example, we may only allow top bidders to get the item, and we may ask for at most the maximum bid price to be paid by whoever gets the item.

Note that a mechanism $M = (A_1, \ldots, A_n, g) \in \mathcal{M}$ together with an environment $\mathcal{E}$, gives a full description of a (Bayesian) game, $\Gamma_{M,\mathcal{E}} = (N, A, \tilde{u}, \Theta, p)$, where the payoff function $\tilde{u}_i : A \times \Theta_i \to \mathbb{R}$ for player i is now given by: $\tilde{u}_i(a_1, \ldots, a_n, \theta_i) := u_i(g(a_1, \ldots, a_n), \theta_i)$.
Note: A (pure) strategy for player i in this Bayesian setting is a function $s_i : \Theta_i \to A_i$, i.e., a function from its types to its actions.

choice functions

Given an environment $\mathcal{E}$, a **choice function**

$$f : \Theta \mapsto 2^C$$

specifies what set of outcomes the designer would prefer if it knew exactly what type (i.e., what utility (payoff) function) each player has.
single-item auction example: the choice function of the designer could be, e.g., to give the item to a person who values it most:

$$f(v_1, \ldots, v_n) = \{(i, pr) \mid i \in \arg\max_{i'} v_{i'}\}$$

solution concepts

We lastly need the notion of a solution concept, which specifies what kind of solutions of a game we are trying to implement the choice function in. Such solutions might be, e.g., (Bayes) Nash Equilibria, or more strong conditions such as a Dominant Strategies Equilibrium.

Given an environment $\mathcal{E}$, a **solution concept** is given by a mapping Sol, whose domain is $\mathcal{M} \times \Theta$, and such that for each mechanism $M = (A_1, \ldots, A_n, g)$ and each tuple $\theta = (\theta_1, \ldots, \theta_n)$ of types, $Sol((A_1, \ldots, A_n, g), \theta) \in 2^S$, where $S = S_1 \times \ldots \times S_n$, and where $S_i = \{s_i : \Theta_i \to A_i\}$ is the strategy set of player i.

auction example: in the auction example we may wish to implement the choice in dominant strategies. In this setting, we ask that $Sol(M, \theta)$ be the set of strategy combinations $(s_1, \ldots, s_n)$ such that each $s_i : \Theta_i \to A_i$ is a dominant strategy for player i in the (Bayesian) game $\Gamma_{M,\mathcal{E}}$.

finally: the mechanism design problem statement

The mechanism design problem can now be stated as follows:

> *Given environment* $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$, *a choice function* f, *and a solution concept* Sol, *find a mechanism* $M = (A_1, \ldots, A_n, g) \in \mathcal{M}$ *such that for all tuples* $\theta \in \Theta$ *of types,* $\text{Sol}(M, \theta)$ *is a non-empty set, and for all* $s \in \text{Sol}(M, \theta)$, $g(s(\theta)) \in f(\theta)$.

If such a mechanism M exists, we say M *Sol*-**implements** choice function f in environment $\mathcal{E}$. We then also say f is *Sol*-**implementable** in $\mathcal{E}$.
of mechanism design. See books
[Mas-Colell-Whinston-Green'95,Osborne-Rubinstein'94].
Note: As already indicated by the *Gibbard-Satterthwaite Theorem*, in some environments, some social choice functions, such as truth revelation, are not (dominant-strategy)-implementable.
A result that is at first surprising, called the "Revelation Principle", says that everything that is dominant-strategy implementable can in fact be implemented as a truth revelation dominant strategy.

The Revelation Principle

Let *Dom* be the solution concept where $s = (s_1, \ldots, s_n) \in Dom(M, \theta)$ if and only if for every player i, $s_i : \theta_i \to A_i$ is a dominant strategy in $\Gamma_{M,\mathcal{E}}$. Such a profile s is called a dominant strategy equilibrium (DSE) of $\Gamma_{M,\mathcal{E}}$. Similarly, define *BNE* to be the solution concept where $s \in BNE(M, \theta)$ iff s is a Bayesian Nash Equilibrium of $\Gamma_{M,\mathcal{E}}$.
In environment $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$, a mechanism M is called a **direct revelation mechanism** if $M = (\Theta_1, \ldots, \Theta_n, g)$. I.e., players' strategies $s_i : \Theta_i \to \Theta_i$ amount to "announcing" their type. Players can lie, but.....

Theorem (*Revelation Principle*)

Suppose f is *Dom*-implemented (*BNE*-implemented, respectively) by $M = (A_1, ..., A_n, g)$ in environment $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$.
Then in environment $(N, C, \Theta, u, p, \mathcal{M}')$, where $\mathcal{M}'$ consists of all direct revelation mechanisms, there is some $M' = (\Theta_1, \ldots, \Theta_n, g') \in \mathcal{M}'$ such that for all $\theta \in \Theta$, $s^* \in Dom(M', \theta)$ (respectively $s^* \in BNE(M', \theta)$), where $s_i^*(\theta_i) = \theta_i$ for all i. And, $\exists s \in Dom(M, \theta)$, ($\exists s \in BNE(M, \theta)$, respectively), such that $\forall \theta \in \Theta$, $g'(s^*(\theta)) = g(s(\theta)) \in f(\theta)$.

interpreting the revelation principle

The revelation principle (RP), for now restricted to the dominant-strategy case, says that if a mechanism M *Dom*-implements a choice function f then there is another mechanism M' which *Dom*-implements f, where each player revealing their true type is a DSE in $\Gamma_{M',\mathcal{E}}$. We then say M' **truthfully** ***Dom*-implements** f.

(An analogous version can be stated for implementation in BNE.)

Note the contrast to the Gibbard-Satterthwaite Theorem, which says that there is no non-dictatorial social choice function for player's preferences among 3 or more alternatives for which "truth revealing" is implementable by dominant strategies.

proof of the revelation principle

Proof

Let's prove it in the case of dominant strategy implementation.
Suppose that some indirect mechanism $M = (A_1, \ldots, A_n, g)$, *Dom*-implements f.
Suppose $s' = (s'_1, \ldots, s'_n)$ is a dominant strategy profile in the game $\Gamma_{M,\mathcal{E}}$, where $s'_i : \Theta_i \to A_i$.

In other words, each player i will prefer to play $s'_i(\theta_i)$ if his type is θ_i, regardless of what the other strategies s_{-i} of other players are.

Now we create a direct revelation mechanism $M' = (\Theta_1, \ldots, \Theta_n, g')$ by using a mediator.
The mediator says to each player i: "If you tell me your type, and if you say your type is θ_i then I will play strategy $s'_i(\theta_i)$ for you."
Clearly, since s'_i was dominant for M, telling the truth, θ_i, will be dominant for the new direct revelation mechanism M', and it will implement the same choice function f. □

In the Algorithmic Mechanism Design problem, we will additionally want to insist that functions like the choice function, f, and the outcome function $g : A_1 \times \ldots \times A_n \to C$ of the mechanism $M = (A_1, \ldots, A_n, g)$ that implements f, should be efficiently computable; but importantly, it is often necessary to adjust the choice function f to allow more outcomes (e.g., not necessarily optimal outcomes, but perhaps only approximately optimal ones), in order to allow efficient computability (and maintain implementability).

THE END

(hope you enjoyed the course)