

CPS 590.4: Computational Microeconomics: Game Theory, Social Choice, and Mechanism Design

Instructor: Vincent Conitzer

conitzer@cs.duke.edu

Course web page:

<http://www.cs.duke.edu/courses/spring16/compsci590.4/>

Journal, conference, ...

**ACM Transactions on
Economics and Computation
(TEAC)**



17th ACM CONFERENCE ON ECONOMICS AND COMPUTATION
JULY 24-28 2016 | MAASTRICHT, THE NETHERLANDS

History



*John von
Neumann*

computer architecture
(von Neumann
architecture)

game theory
(minimax theorem)

linear programming
(duality)

***Computer Science
& Engineering***

Economic Theory

***Mathematical
Optimization &
Operations
Research***

1900

1950

2000

→

CS-ECON@DUKE

[Home](#) [Schedule](#) [Past talks](#) [People](#)

We are a group of people interested in the intersection of computer science and economics (and the social sciences more broadly) and the impact of this interplay on decisions in information technology and digital business. This includes applying techniques from computer science and optimization to economics -- for example, using computation to design market clearing mechanisms and to implement efficient allocation and pricing in them -- as well as applying techniques from economics to computer science -- for example, designing incentives for users of networked computer systems and social networks.

Contacts

For organizational questions about the seminar series, please contact [Dima Korzhik](#). For other matters, please approach the relevant faculty contact(s): [Atila Abdulkadiroglu](#) (Econ), [Vincent Conitzer](#) (CS), [Rachel Kranton](#) (Econ), [Ben Lee](#) (ECE), [Kamesh Munagala](#) (CS), [Sasa Pekic](#) (Fuqua). [Pam Spencer](#) helps with catering and arranging the speakers' travel.

Mailing List

Please subscribe to the [cs-econ mailing list](#) if you are at Duke (or in the vicinity) and interested in the seminar series. The list will be used for talk announcements.



**[http://econ.
cs.duke.edu](http://econ.cs.duke.edu)**

- [Atila Abdulkadiroglu](#)
Department of Economics.
- [Owen Astrachan](#)
Department of Computer Science.
- [Charles Becker](#)
Department of Economics.
- [Alexandre Belloni](#)
The Fuqua School of Business.
- [David B. Brown](#)
The Fuqua School of Business.
- [Vincent Conitzer](#)
Department of Computer Science, and
- [Landon Cox](#)
Department of Computer Science.
- [Brendan Daley](#)
The Fuqua School of Business.
- [Daniel A. Graham](#)
Department of Economics.
- [Rachel E. Kranton](#)
Department of Economics.
- [R. Vijay Krishna](#)
Department of Economics, UNC.
- [Benjamin C. Lee](#)
Department of Electrical and Computer Engineering.
- [Giuseppe \(Pino\) Lopomo](#)
The Fuqua School of Business.
- [David McAdams](#)
The Fuqua School of Business, and Department of Economics.
- [Carl F. Mela](#)
The Fuqua School of Business.
- [Kamesh Munagala](#)
Department of Computer Science.
- [Andres Musalem](#)
The Fuqua School of Business.
- [Aleksandar Sasa Pekic](#)
The Fuqua School of Business.
- [Philipp Sadowski](#)
Department of Economics.
- [Peng Sun](#)
The Fuqua School of Business.
- [Curtis R. Taylor](#)
Department of Economics.
- [Kenneth C Wilbur](#)
The Fuqua School of Business.

Master's of Science in Economics and Computation (MSEC)

The MSEC degree is a joint master's program of the Departments of Economics and **Computer Science**. The joint field of economics and computation has recently emerged from two converging intellectual needs, which has created the opportunity for a truly interdisciplinary program.

The MSEC program is the outcome of exciting developments across the two fields:

- Computer science is becoming increasingly important for economists addressing complex questions on large repositories of data;
- The explosion of computer uses in all areas of life has made it necessary for computer scientists to understand the economics of computing systems; and,
- Computer scientists may now also analyze informational and financial transactions between people, businesses, governments, and electronic agents in economic terms.

"Macroeconomic problems are almost always analytically intractable and therefore require a computer to solve ... Having a background in computer science made it much easier to learn (computational) methods, as well as apply cutting-edge advances in technology to economic problems."

David Klemish

MSEM

"Tech giants such as Google, Microsoft, Yahoo, and Facebook are in many ways ahead of academia in getting computer scientists and economists to work together, and interest in these topics extends well beyond this group of companies. With this Master's program, Duke is at the forefront of catching up with industry in breaking down boundaries."

Vince Conitzer

Department of Computer Science

Duke Economics

UNIVERSITY

About

Undergraduate

Master's Program

Ph.D. Program

Research

Centers & Initiatives

Home / Master's Program / **Master's of Science in Economics and Computation**

Application Deadline

For Fall 2014: February 15, 2014

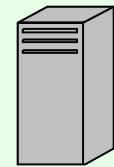
Charles Becker, MSEC Program Director

What is Economics?

- “Economics is the social science that describes the factors that determine the production, distribution and consumption of goods and services.” [Wikipedia, Jan. 2016]
- Some key concepts:
 - Economic **agents** or **players** (individuals, households, firms, ...)
 - Agents’ current **endowments** of goods, money, skills, ...
 - Possible **outcomes** ((re)allocations of resources, tasks, ...)
 - Agents’ **preferences** or **utility functions** over outcomes
 - Agents’ **beliefs** (over other agents’ utility functions, endowments, production possibilities, ...)
 - Agents’ possible **decisions/actions**
 - **Mechanism** that maps decisions/actions to outcomes

An economic picture

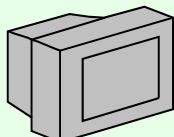
$$v(\text{server}) = 200$$



\$ 800

$$v(\text{monitor}) = 100$$

$$v(\text{laptop}) = 400$$



\$ 600

$$v(\text{laptop}) = 200$$

$$v(\text{server}, \text{monitor}) = 400$$



\$ 200



AAAClipArt.com

After trade (a more efficient outcome)

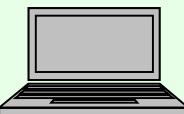
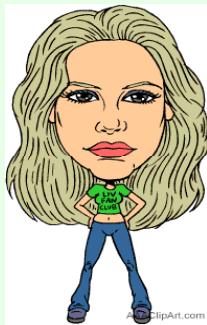
$$v(\text{server}) = 200$$



*... but how do we
get here?
Auctions?
Exchanges?
Unstructured trade?*

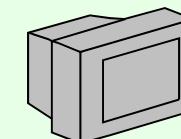
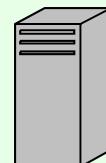
\$ 1100

$$v(\text{monitor}) = 100$$
$$v(\text{laptop}) = 400$$



\$ 400

$$v(\text{laptop}) = 200$$
$$v(\text{server}, \text{monitor}) = 400$$



\$ 100



AAAClipArt.com

Some distinctions in economics

- Descriptive vs. normative economics
 - Descriptive:
 - seeks only to describe real-world economic phenomena
 - does not care if this is in any sense the “right” outcome
 - Normative:
 - studies how people “should” behave, what the “right” or “best” outcome is
- Microeconomics vs. macroeconomics
 - Microeconomics: analyzes decisions at the level of individual agents
 - deciding which goods to produce/consume, setting prices, ...
 - “bottom-up” approach
 - Macroeconomics: analyzes “the sum” of economic activity
 - interest rates, inflation, growth, unemployment, government spending, taxation, ...
 - “big picture”

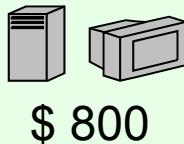
What is Computer Science?

- “Computer science is the scientific and practical approach to computation and its applications. [...] A computer scientist specializes in the theory of computation and the design of computational systems.” [Wikipedia, Jan. 2016]
- A **computational problem** is given by a function f mapping inputs to outputs
 - For integer x , let $f(x) = 0$ if x is prime, 1 otherwise
 - For an initial allocation of resources x , let $f(x)$ be the (re)allocation that maximizes the sum of utilities
- An **algorithm** is a fully specified procedure for computing f
 - E.g., sieve of Eratosthenes
 - A **correct algorithm** always returns the **right answer**
 - An **efficient algorithm** returns the answer **fast**
- Computer science is also concerned with building **larger artifacts** out of these building blocks (e.g., personal computers, spreadsheets, the Internet, the Web, search engines, artificial intelligence, ...)

Resource allocation as a computational problem

input

$$v(\text{server, monitor}) = \$400$$
$$v(\text{laptop}) = \$600$$



\$ 800

$$v(\text{server, monitor}) = \$500$$
$$v(\text{laptop}) = \$400$$

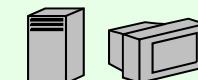


\$ 400

output



\$ 750



\$ 450

Here, gains from trade (\$300)
are divided evenly
(not essential)

Economic mechanisms

“true” input

$$v(\text{server, microwave}) = \$400$$
$$v(\text{laptop}) = \$600$$



\$ 800

$$v(\text{server, microwave}) = \$500$$
$$v(\text{laptop}) = \$400$$



\$ 400

agents’ bids

$$v(\text{server, microwave}) = \$500$$
$$v(\text{laptop}) = \$501$$



\$ 800

$$v(\text{server, microwave}) = \$451$$
$$v(\text{laptop}) = \$450$$



\$ 400

agent 1’s
bidding
algorithm

agent 2’s
bidding
algorithm

result

exchange
mechanism
(algorithm)



\$ 800



\$ 400

*Exchange mechanism designer
does not have direct access to
agents’ private information*

*Agents will selfishly respond to
incentives*

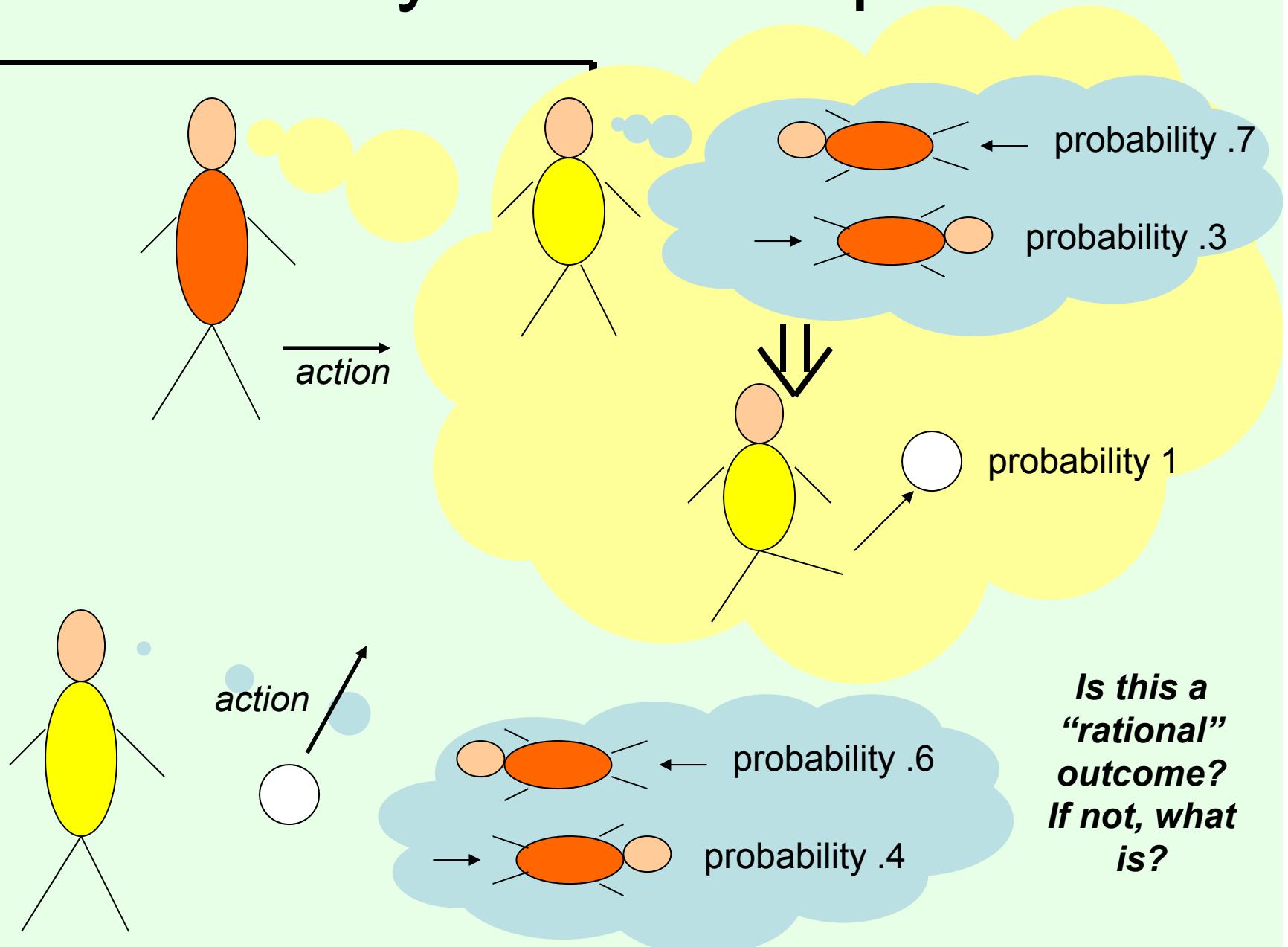
What is game theory?

- “Game theory is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. Game theory is mainly used in economics, political science, and psychology, as well as logic, computer science, biology and Poker (Texas No Limit Hold'em).”
[Wikipedia, Jan. 2016]

What is game theory...

- Game theory studies settings where multiple parties (**agents**) each have
 - different preferences (utility functions),
 - different actions that they can take
- Each agent's utility (potentially) depends on all agents' actions
 - What is optimal for one agent depends on what other agents do
 - Very circular!
- Game theory studies how agents can rationally form **beliefs** over what other agents will do, and (hence) how agents should **act**
 - Useful for acting as well as predicting behavior of others

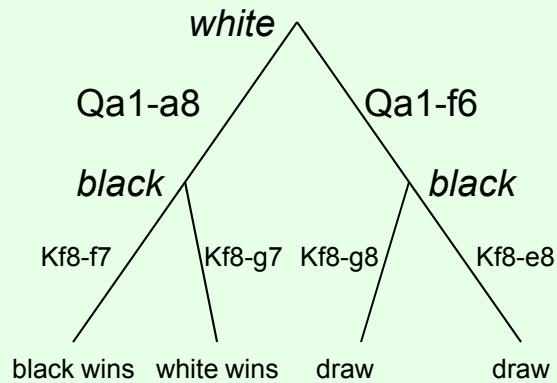
Penalty kick example



Game playing & AI

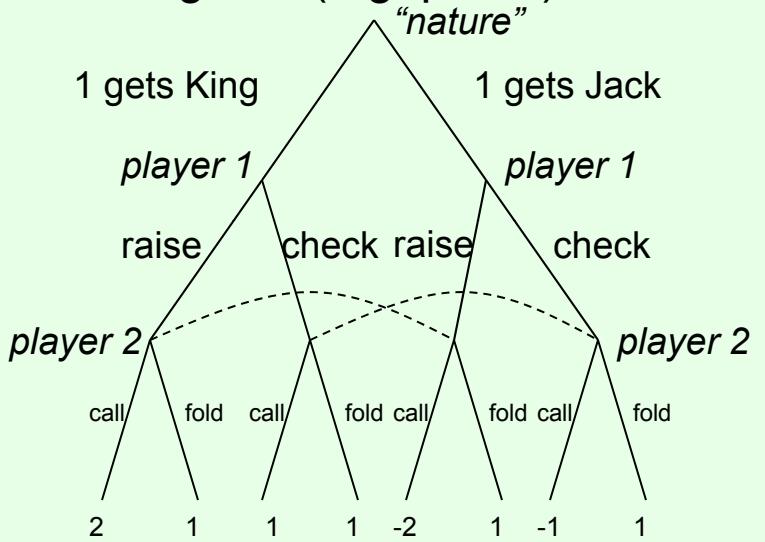
perfect information games:

no uncertainty about the state of the game (e.g. tic-tac-toe, chess, Go)



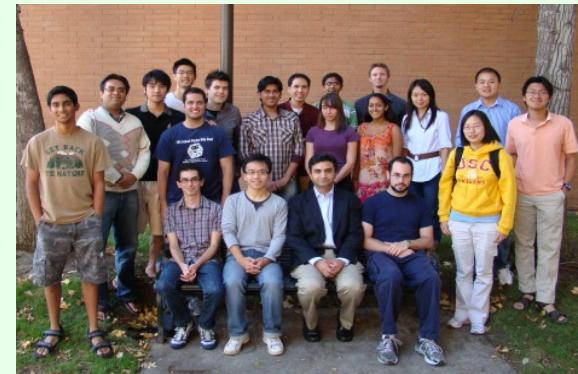
- Optimal play: value of each node = value of optimal child for current player (**backward induction**, minimax)
- For chess and Go, tree is too large
 - Use other techniques (heuristics, limited-depth search, alpha-beta, ...)
- Top computer programs (arguably) better than humans in chess, not yet in Go

imperfect information games: uncertainty about the state of the game (e.g. poker)



- Player 2 **cannot distinguish** nodes connected by dotted lines
 - Backward induction fails; need more sophisticated game-theoretic techniques for optimal play
- Small poker variants can be solved optimally
- Humans still better than top computer programs at full-scale poker (at least most versions)
- Top computer (heads-up) poker players are based on techniques for game theory

Real-world security applications



Milind Tambe's TEAMCORE group (USC)



Airport security

- Where should checkpoints, canine units, etc. be deployed?
- Deployed at LAX and another US airport, being evaluated for deployment at all US airports

Federal Air Marshals

- Which flights get a FAM?



US Coast Guard

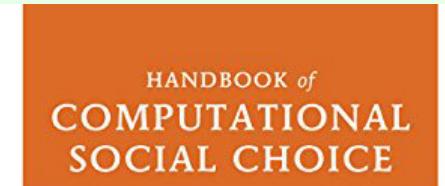
- Which patrol routes should be followed?
- Deployed in Boston Harbor

Questions and problems in (computational) game theory

- How should we **represent** games (=strategic settings)?
 - Standard game-theoretic representations not always concise enough
- What does it mean to **solve** a game?
 - Solution concepts from game theory, e.g., Nash equilibrium
- How **computationally hard** is it to solve games?
 - Can we solve them approximately?
- Is there a role for **(machine) learning** in games?
- What types of **modeling problems** do we face when addressing real-world games?
 - E.g., applications in security
- ...

What is social choice?

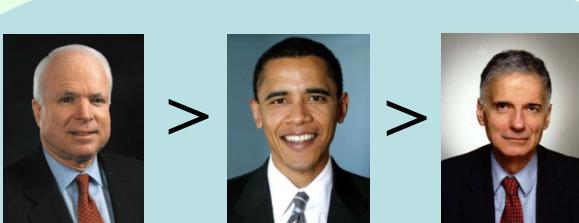
- “Social choice theory or social choice is a theoretical framework for analysis of combining individual opinions, preferences, interests, or welfares to reach a collective decision or social welfare in some sense.” [\[Wikipedia, Jan. 2016\]](#)
- I.e., making decisions based on the preferences of multiple agents
- Largely, but not exclusively, focused on **voting**



Voting over outcomes



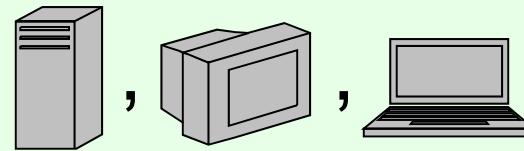
voting rule
(mechanism)
determines winner
based on votes



- Can vote over other things too
 - Where to go for dinner tonight, other joint plans, ...
- Many different rules exist for selecting the winner

Combinatorial auctions

Simultaneously for sale:



bid 1

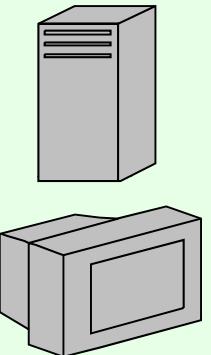
$$v(\text{server} \cup \text{monitor}) = \$500$$

bid 2

$$v(\text{laptop} \cup \text{monitor}) = \$700$$

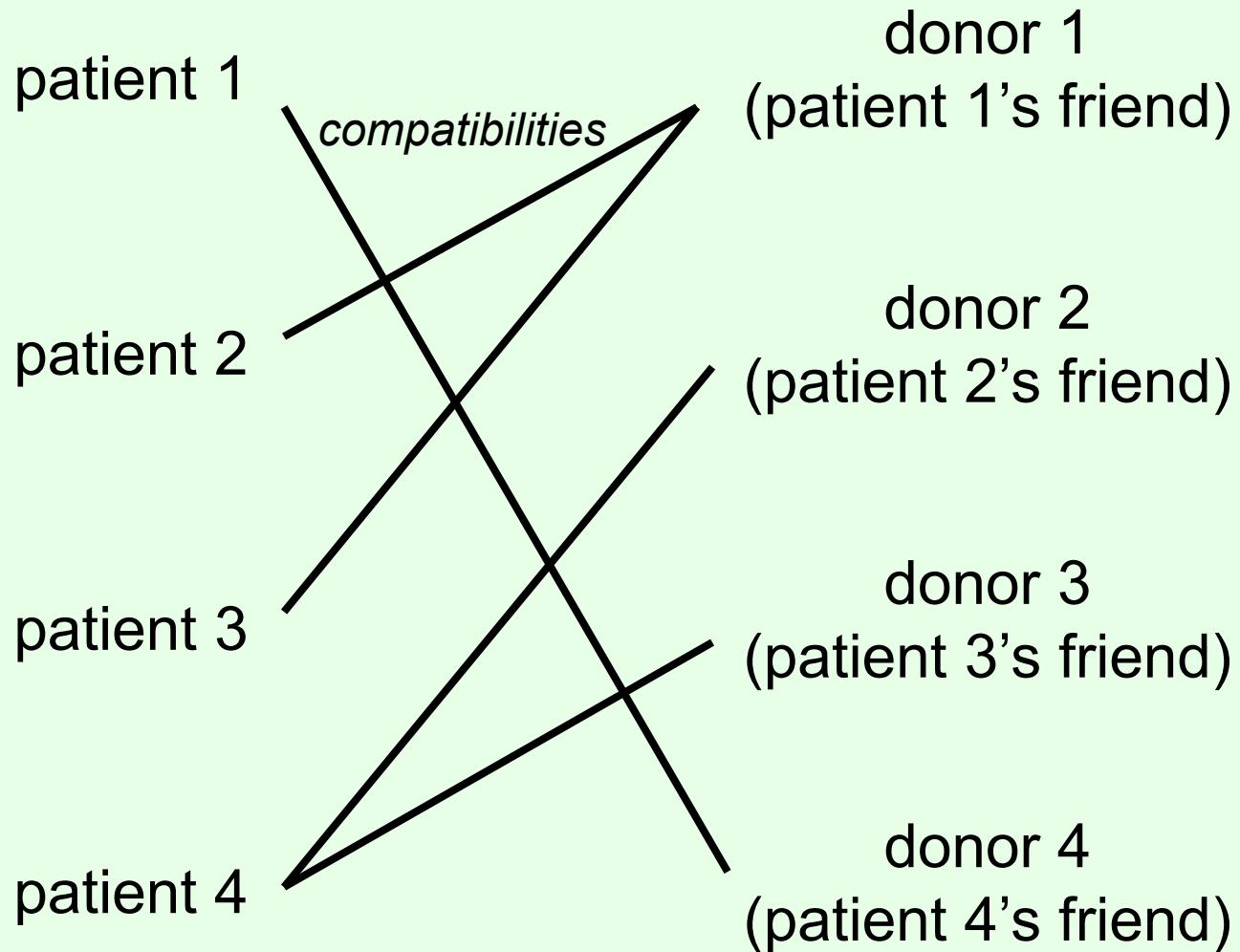
bid 3

$$v(\text{laptop}) = \$300$$



used in truckload transportation, industrial procurement, radio spectrum allocation, ...

Kidney exchange



Problems in computational social choice

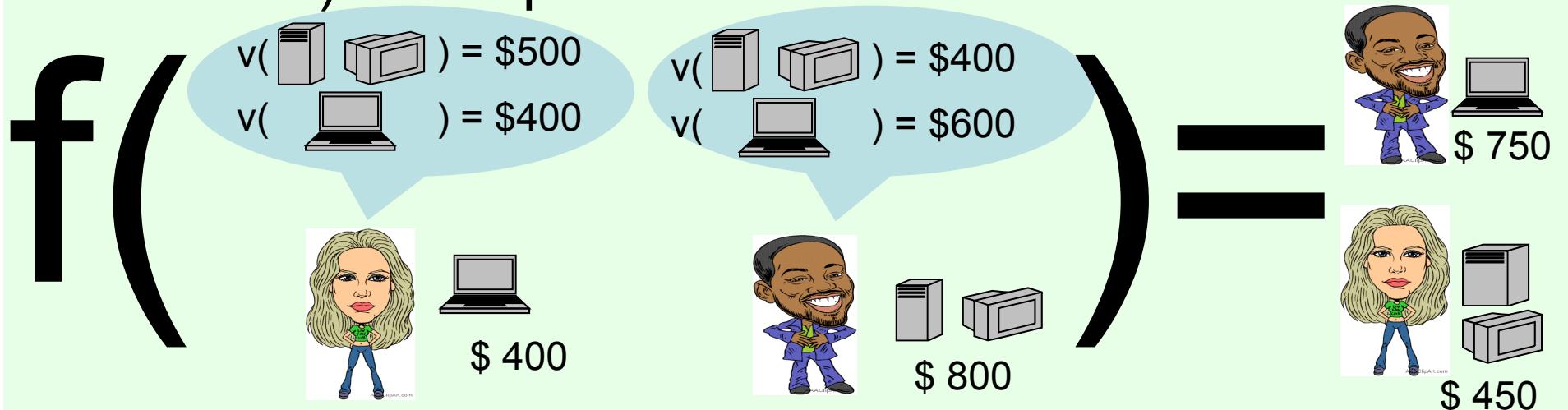
- Winner determination problem
 - For some voting rules, determining the winner is NP-hard
 - In a combinatorial auction, deciding which bids win is (in general) an NP-hard problem
- Preference elicitation (communication) problem
 - Can be impractical to communicate all of one's preferences (e.g., valuation for every bundle)
- Mechanism design problem
 - How do we get the bidders to behave so that we get good outcomes?
- These problems interact in nontrivial ways
 - E.g. limited computational or communication capacity can limit mechanism design options
 - ... but can perhaps also be used in a positive way

What is mechanism design?

- “Mechanism design is a field in economics and game theory that takes an engineering approach to designing economic mechanisms or incentives, toward desired objectives, in strategic settings, where players act rationally. [...] Two distinguishing features of [mechanism design] are:
 - that a game “designer” chooses the game structure rather than inheriting one
 - that the designer is interested in the game’s outcome
- [Wikipedia, Jan. 2016]

Mechanism design...

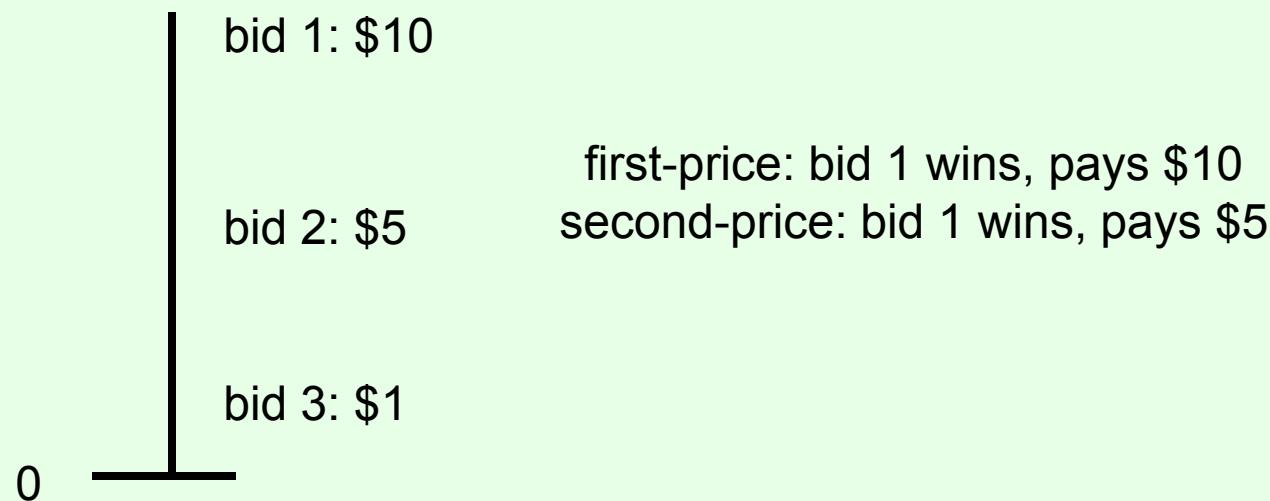
- Mechanism = rules of auction, exchange, ...
- A function that takes reported preferences (bids) as input, and produces outcome (allocation, payments to be made) as output



- The entire function f is one mechanism
- E.g., the mechanism from before: find allocation that maximizes (reported) utilities, distribute (reported) gains evenly
- Other mechanisms choose different allocations, payments

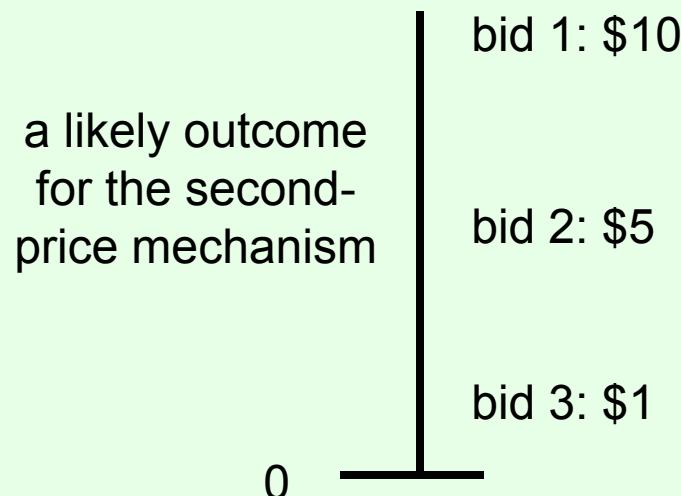
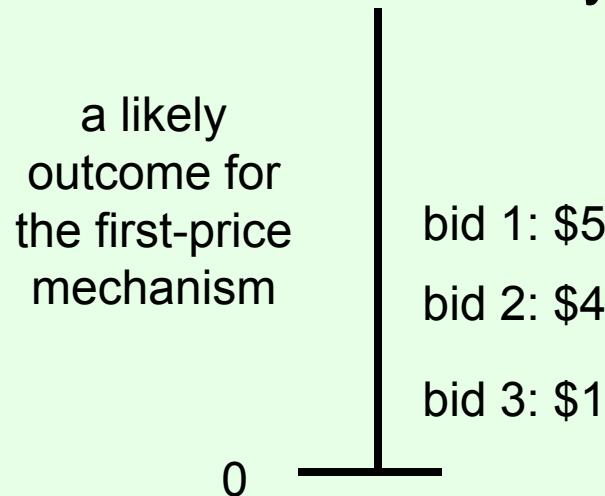
Example: (single-item) auctions

- **Sealed-bid** auction: every bidder submits bid in a sealed envelope
- **First-price** sealed-bid auction: highest bid wins, pays amount of own bid
- **Second-price** sealed-bid auction: highest bid wins, pays amount of second-highest bid



Which auction generates more revenue?

- Each bid depends on
 - bidder's **true valuation** for the item (utility = valuation - payment),
 - bidder's **beliefs** over what others will bid (\rightarrow game theory),
 - and... the **auction mechanism** used
- In a first-price auction, it does not make sense to bid your true valuation
 - Even if you win, your utility will be 0...
- In a second-price auction, (we will see later that) it always makes sense to bid your true valuation



Are there other auctions that perform better? How do we know when we have found the best one?

Mechanism design....

- Mechanism = game
- → we can use game theory to predict what will happen under a mechanism
 - if agents act strategically
- When is a mechanism “good”?
 - Should it result in outcomes that are good for the reported preferences, or for the true preferences?
 - Should agents ever end up lying about their preferences (in the game-theoretic solution)?
 - Should it always generate the best allocation?
 - Should agents ever burn money?(!?)
- Can we solve for the optimal mechanism?

Many uses of linear programming, mixed integer (linear) programming in this course

	Linear programming	Mixed integer linear programming
Game theory	Dominated strategies Minimax strategies Correlated equilibrium Optimal mixed strategies to commit to	Nash equilibrium Optimal mixed strategies to commit to in more complex settings
Social choice, expressive marketplaces	Winner determination in auctions, exchanges, ... with partially acceptable bids	Winner determination in: auctions, exchanges, ... without partially acceptable bids; Kemeny, Slater, other voting rules; kidney exchange
Mechanism design	Automatically designing optimal mechanisms that use randomization	Automatically designing optimal mechanisms that do not use randomization

Sponsored search

Web Images Videos Maps News Shopping Gmail more ▾



Web Show options...

[Scholarly articles for combinatorial auction](#)

-  [Algorithm for optimal winner determination in ...](#) - Sandholm - Cited by 755
- [Combinatorial auctions](#) - Cramton - Cited by 364
- [Taming the computational complexity of combinatorial ...](#) - Fujishima - Cited by 424

[Combinatorial auction - Wikipedia, the free encyclopedia](#)

Jun 26, 2009 ... A combinatorial auction is an auction in which bidders can place bids on combinations of items, or "packages," rather than just individual ...
en.wikipedia.org/w/index.php?title=Combinatorial_auction&oldid=3100000 - [Cached](#) - [Similar](#)

[PDF Introduction to Combinatorial Auctions](#)

File Format: PDF/Adobe Acrobat - [View](#)
combinatorial auctions can be studied in a wide range of auction In Chapter 18, Leyton-Brown and Shoham present the Combinatorial Auction Test Suite ...
www.cramton.umd.edu/.../cramton-shoham-steinberg-introduction-to-combinatorial-auctions.pdf - [Similar](#)
by P Cramton - [Cited by 17](#) - [Related articles](#) - [All 14 versions](#)

[Combinatorial Auctions](#)

A comprehensive book on combinatorial auctions—auctions in which bidders can bid on packages of items. The book consists of original material intended for ...
www.cramton.umd.edu/.../combinatorial-auctions-book-public.htm - [Similar](#)

[Show more results from www.cramton.umd.edu](#)

[PDF Combinatorial Auctions: A Survey](#)

File Format: PDF/Adobe Acrobat - [View as HTML](#)
of combinatorial auctions. Second, it uses this subject as a vehicle to the auction. This feature of combinatorial auctions is called the 'threshold ...
www.cis.upenn.edu/~mkeams/.../combinatorial-auctions-survey.pdf - [Similar](#)
by S de Vries - [Cited by 496](#) - [Related articles](#) - [All 42 versions](#)

[Amazon.com: Combinatorial Auctions \(9780262033428\): Peter Cramton ...](#)

Amazon.com: Combinatorial Auctions (9780262033428): Peter Cramton, Yoav Shoham, Richard Steinberg: Books.
www.amazon.com/Combinatorial-Auctions.../0262033429 - [Cached](#) - [Similar](#)

Sponsored Links

[Combinatorial Auctions](#)

for carbon, water, timber,
grain, ore, licenses & rights
www.tradeslot.com

[Procurement solutions](#)

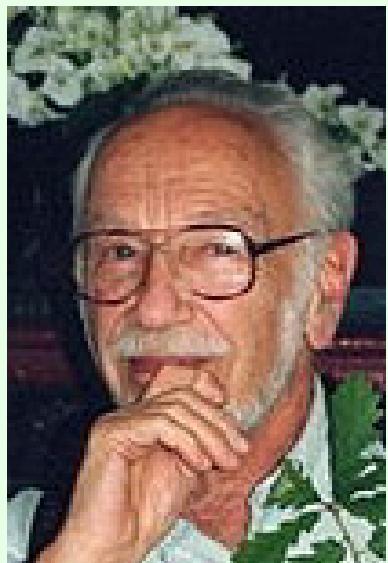
We deliver sustainable cost
reduction via systems & upskilling
www.vendigital.com

[Combinatorial Auction](#)

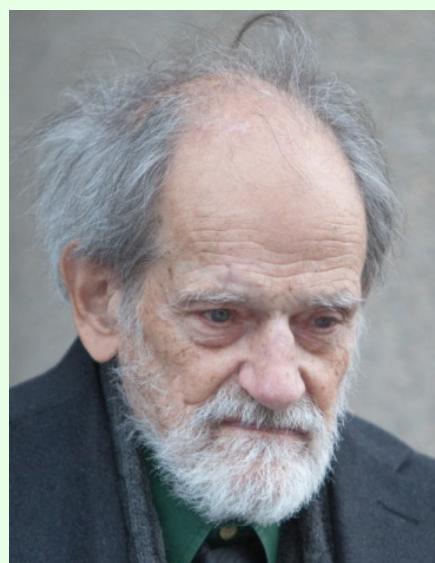
Save on Combinatorial auction
Free Shipping Available. Buy Today!
www.Amazon.com/Books

Deferred Acceptance algorithm

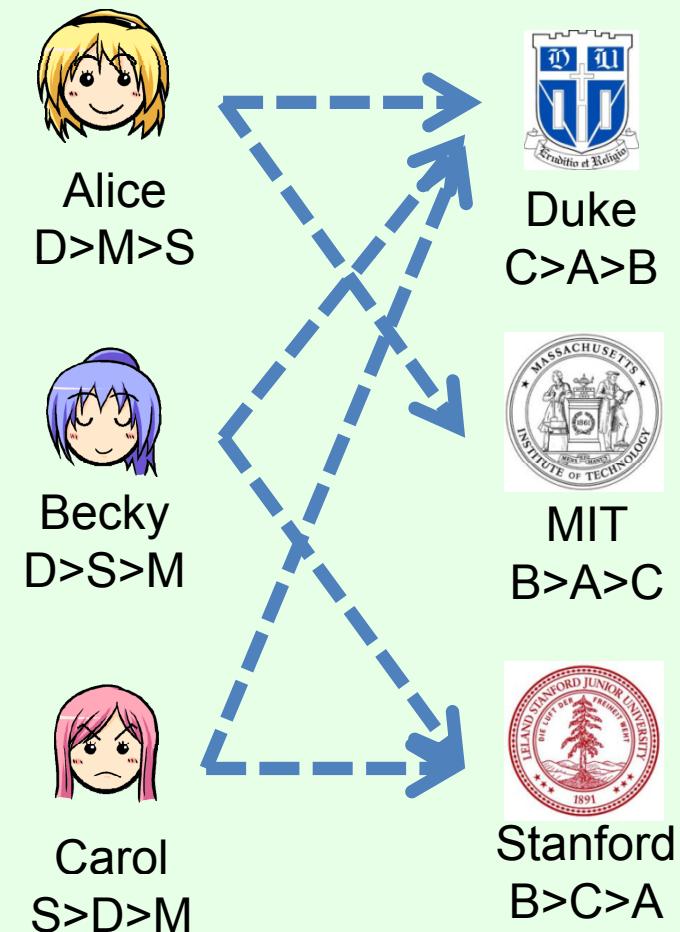
[Gale & Shapley 1962]



David Gale



Lloyd Shapley



Prediction markets

Screenshot of the PredictIt website showing a market for Hillary Clinton's 2016 presidential election win.

PredictIt | Will Hillary Clint... +/-

<https://www.predictit.org/Contract/444/Will-Hillary-Clinton-win-the-2016-US-presidential-election#>

Search ABP 4

MARKETS ANALYSIS ABOUT SIGN IN SIGN UP

Search Markets

Most Predicted Closing Soon

U.S. Elections U.S. Politics World

Will Hillary Clinton win the 2016 U.S. presidential election?

Latest Price: 44¢ ↑ 1¢



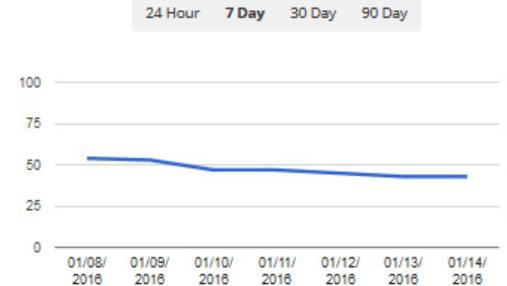
Buy Yes Click to match Offers starting at 45¢, or to make your own, lower Offer.
Buy No Click to match Offers starting at 56¢, or to make your own, lower Offer.

If this prediction comes true, PredictIt will redeem all Yes shares at \$1. Shares in No will have zero value. If this prediction does not come true, PredictIt will redeem all No shares at \$1. Shares in Yes will have zero value.

Data Rules Prices

Symbol:	CLINTON.USPREZ16
Market Type:	Linked ?
Start Date:	10/14/2014
End Date:	12/19/2016
Shares Traded:	470,751
Today's Volume:	1,370
Total Shares:	171,022
Today's Change:	+1¢ ↑

24 Hour 7 Day 30 Day 90 Day



Need Help?

Financial securities

- Tomorrow there must be one of   
- Agent 1 offers \$5 for a security that pays off \$10 if  or 
- Agent 2 offers \$8 for a security that pays off \$10 if  or 
- Agent 3 offers \$6 for a security that pays off \$10 if 
- Can we accept some of these at offers **at no risk?**

How to incentivize a weather forecaster

$$\begin{aligned}P(\text{Sun}) &= .5 \\P(\text{Rain}) &= .3 \\P(\text{Thunder}) &= .2\end{aligned}$$

$$\begin{aligned}P(\text{Sun}) &= .8 \\P(\text{Rain}) &= .1 \\P(\text{Thunder}) &= .1\end{aligned}$$



- Forecaster's bonus can depend on
 - Prediction
 - Actual weather on predicted day
- Reporting true beliefs should maximize expected bonus

Why should economists care about computer science?

- Finding efficient allocations of resources is a (typically hard) **computational problem**
 - Sometimes beyond current computational techniques
 - If so, unlikely that **any** market mechanism will produce the efficient allocation (even without incentives issues)
 - Market mechanisms must be designed **with computational limitations in mind**
 - New algorithms allow new market mechanisms

Why should economists care about computer science...

- **Agents** also face difficult computational problems in participating in the market
 - Especially acting in a game-theoretically optimal way is often **computationally hard**
 - Game-theoretic predictions **will not come true** if they cannot be computed
 - Sometimes bad (e.g., want agents to find right bundle to trade)
 - Sometimes good (e.g., do not want agents to manipulate system)

Why should computer scientists care about economics?

- Economics provides high-value computational problems
- Interesting technical twist: no direct access to true input, must incentivize agents to reveal true input
- Conversely: Computer systems are increasingly used by multiple parties with different preferences (e.g., Internet)
- Economic techniques must be used to
 - predict what will happen in such systems,
 - design the systems so that they will work well
- Game theory is relevant for artificial intelligence
 - E.g., computer poker

CPS 590.4

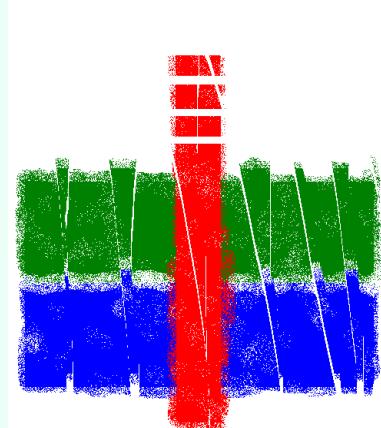
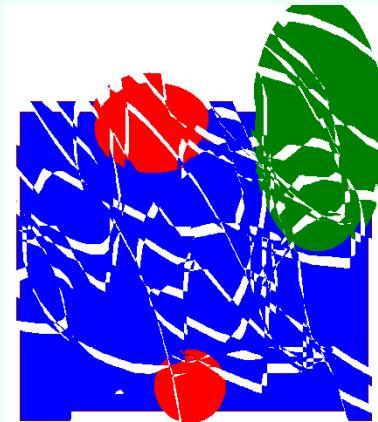
Brief introduction to linear and mixed integer programming

Vincent Conitzer

conitzer@cs.duke.edu

Linear programs: example

- We make reproductions of two paintings



$$\text{maximize } 3x + 2y$$

subject to

$$4x + 2y \leq 16$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$$x \geq 0$$

$$y \geq 0$$

- Painting 1 sells for \$30, painting 2 sells for \$20

- Painting 1 requires 4 units of blue, 1 green, 1 red

- Painting 2 requires 2 blue, 2 green, 1 red

- We have 16 units blue, 8 green, 5 red

Solving the linear program graphically

maximize $3x + 2y$

subject to

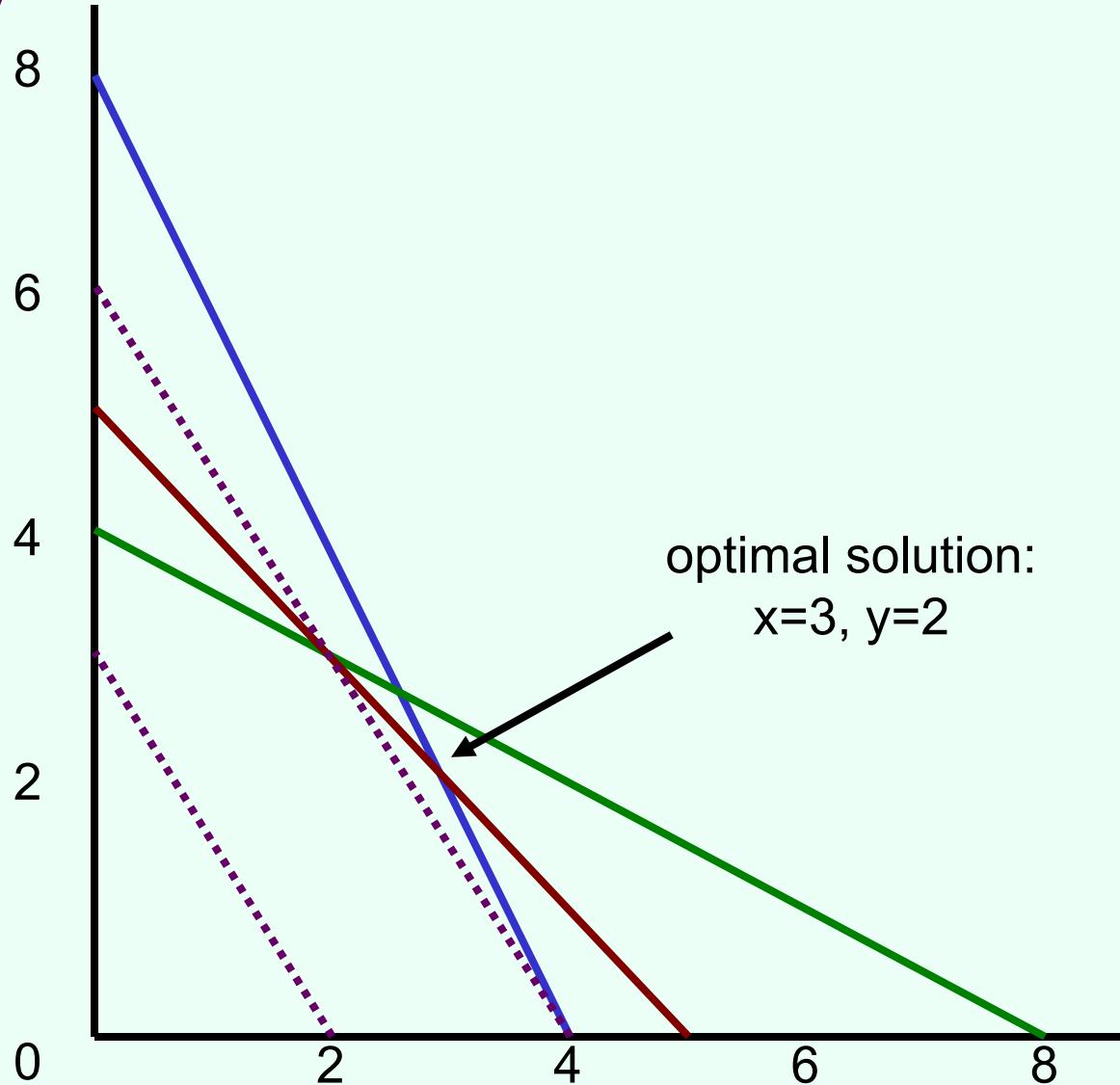
$$4x + 2y \leq 16$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$$x \geq 0$$

$$y \geq 0$$



Modified LP

maximize $3x + 2y$

subject to

$$4x + 2y \leq 15$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$$x \geq 0$$

$$y \geq 0$$

Optimal solution: $x = 2.5$,
 $y = 2.5$

Solution value = $7.5 + 5 =$
12.5

Half paintings?

Integer (linear) program

maximize $3x + 2y$

subject to

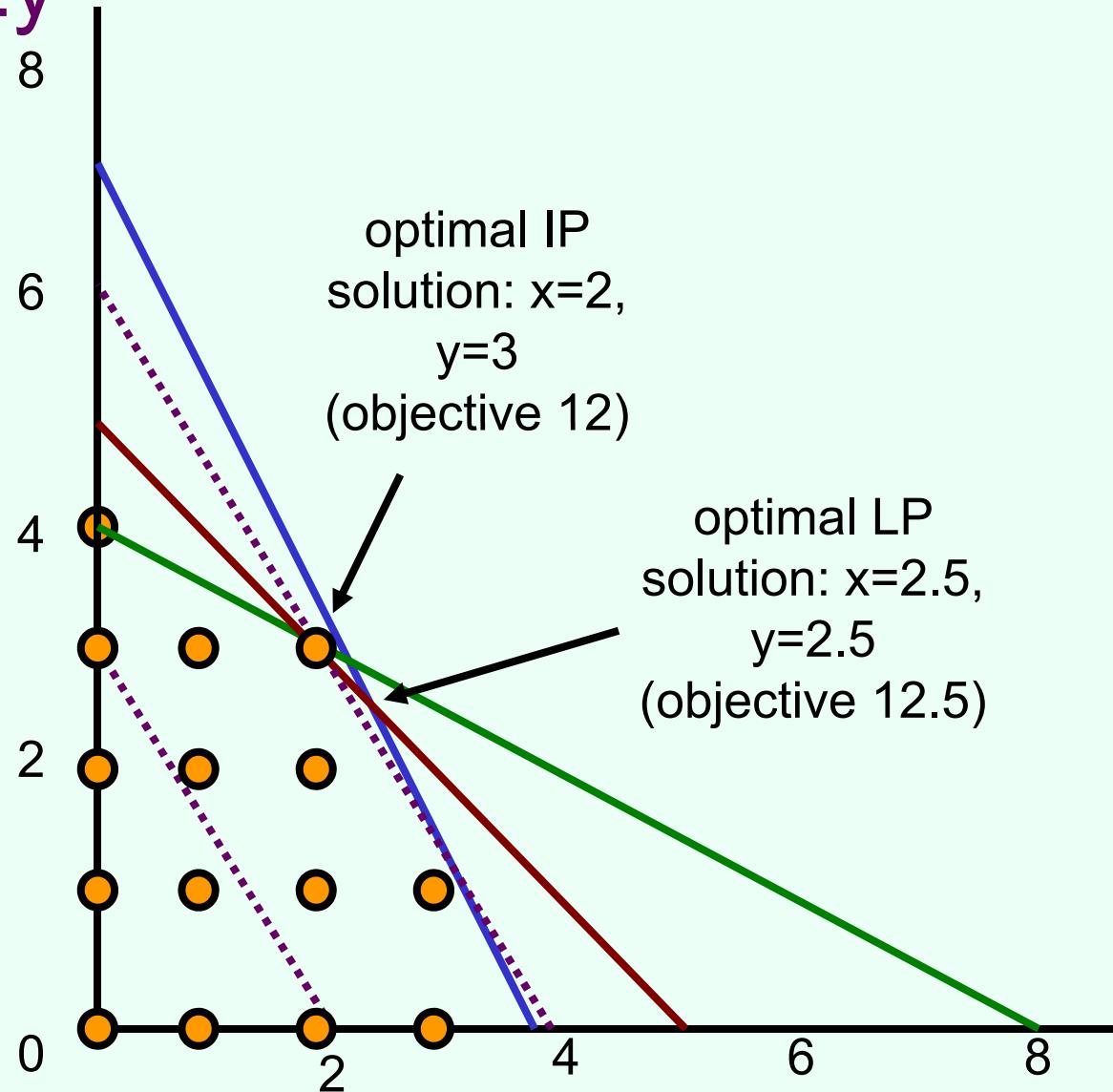
$$4x + 2y \leq 15$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$x \geq 0$, integer

$y \geq 0$, integer



Mixed integer (linear) program

maximize $3x + 2y$

subject to

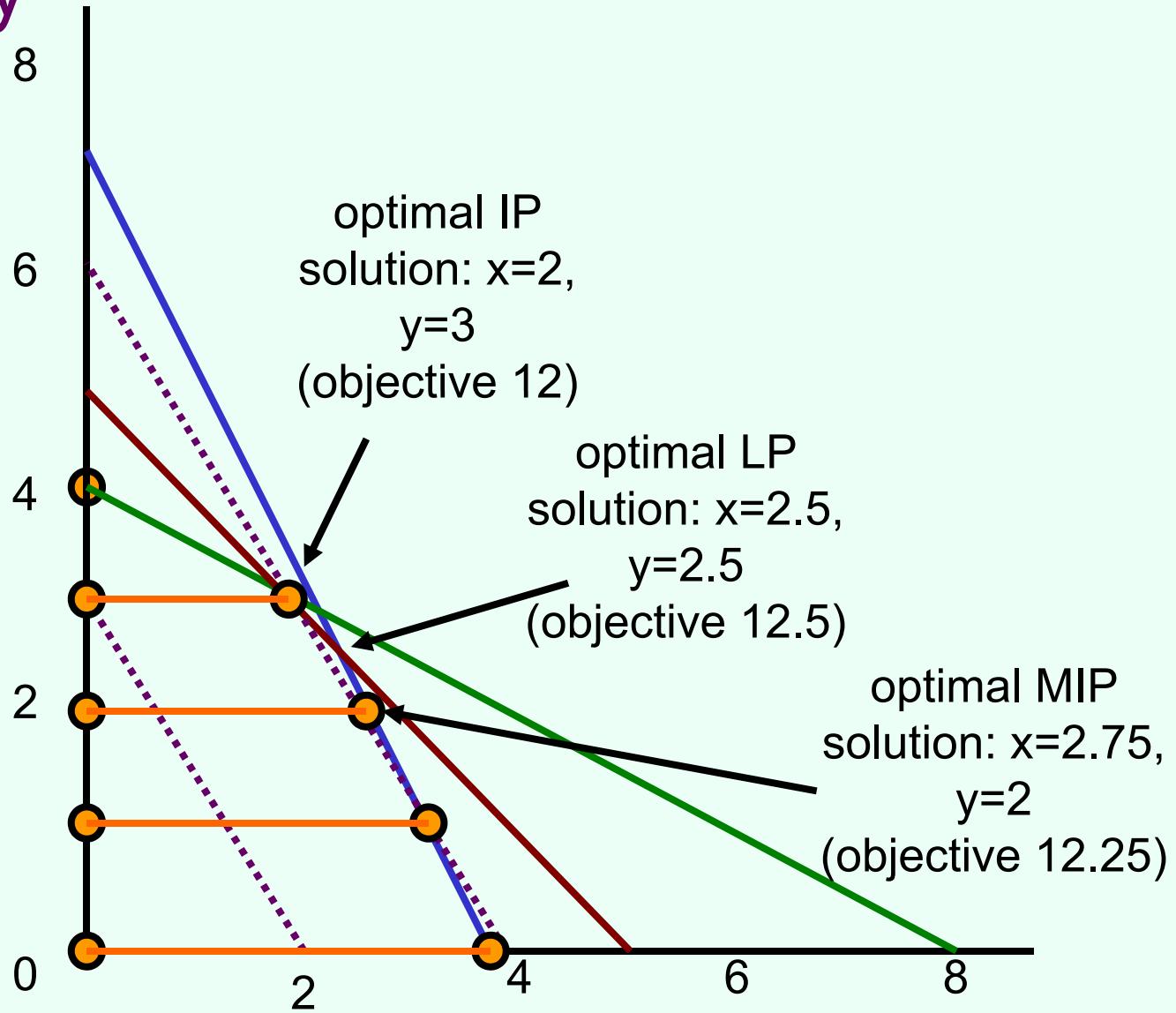
$$4x + 2y \leq 15$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$$x \geq 0$$

$$y \geq 0, \text{ integer}$$



Solving linear/integer programs

- Linear programs can be solved **efficiently**
 - Simplex, ellipsoid, interior point methods...
- (Mixed) integer programs are **NP-hard to solve**
 - Quite easy to model many standard NP-complete problems as integer programs (try it!)
 - Search type algorithms such as branch and bound
- Standard packages for solving these
 - GNU Linear Programming Kit, CPLEX, ...
- **LP relaxation** of (M)IP: remove integrality constraints
 - Gives upper bound on MIP (~**admissible heuristic**)

Exercise in modeling: knapsack-type problem

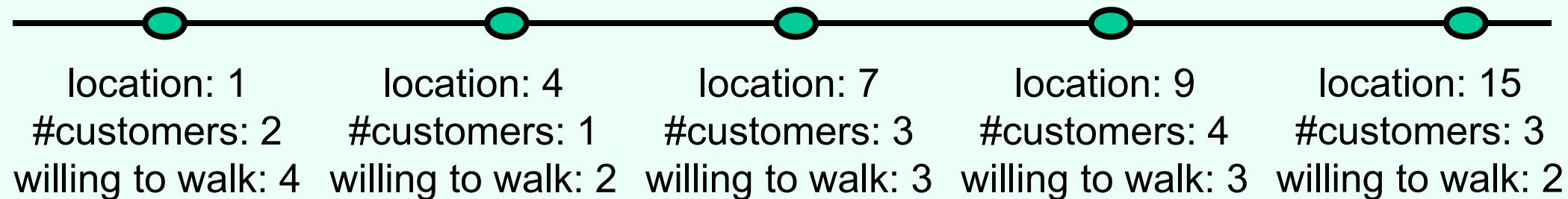
- We arrive in a room full of precious objects
- Can carry only 30kg out of the room
- Can carry only 20 liters out of the room
- Want to maximize our total value
- Unit of object A: 16kg, 3 liters, sells for \$11
 - There are 3 units available
- Unit of object B: 4kg, 4 liters, sells for \$4
 - There are 4 units available
- Unit of object C: 6kg, 3 liters, sells for \$9
 - Only 1 unit available
- What should we take?

Exercise in modeling: cell phones (set cover)

- We want to have a working phone in every continent (besides Antarctica)
- ... but we want to have as few phones as possible
- Phone A works in NA, SA, Af
- Phone B works in E, Af, As
- Phone C works in NA, Au, E
- Phone D works in SA, As, E
- Phone E works in Af, As, Au
- Phone F works in NA, E

Exercise in modeling: hot-dog stands

- We have two hot-dog stands to be placed in somewhere along the beach
- We know where the people that like hot dogs are, how far they are willing to walk
- Where do we put our stands to maximize #hot dogs sold? (price is fixed)



CPS 590.4

Computational problems, algorithms, runtime,
hardness

(a ridiculously brief introduction to theoretical computer science)

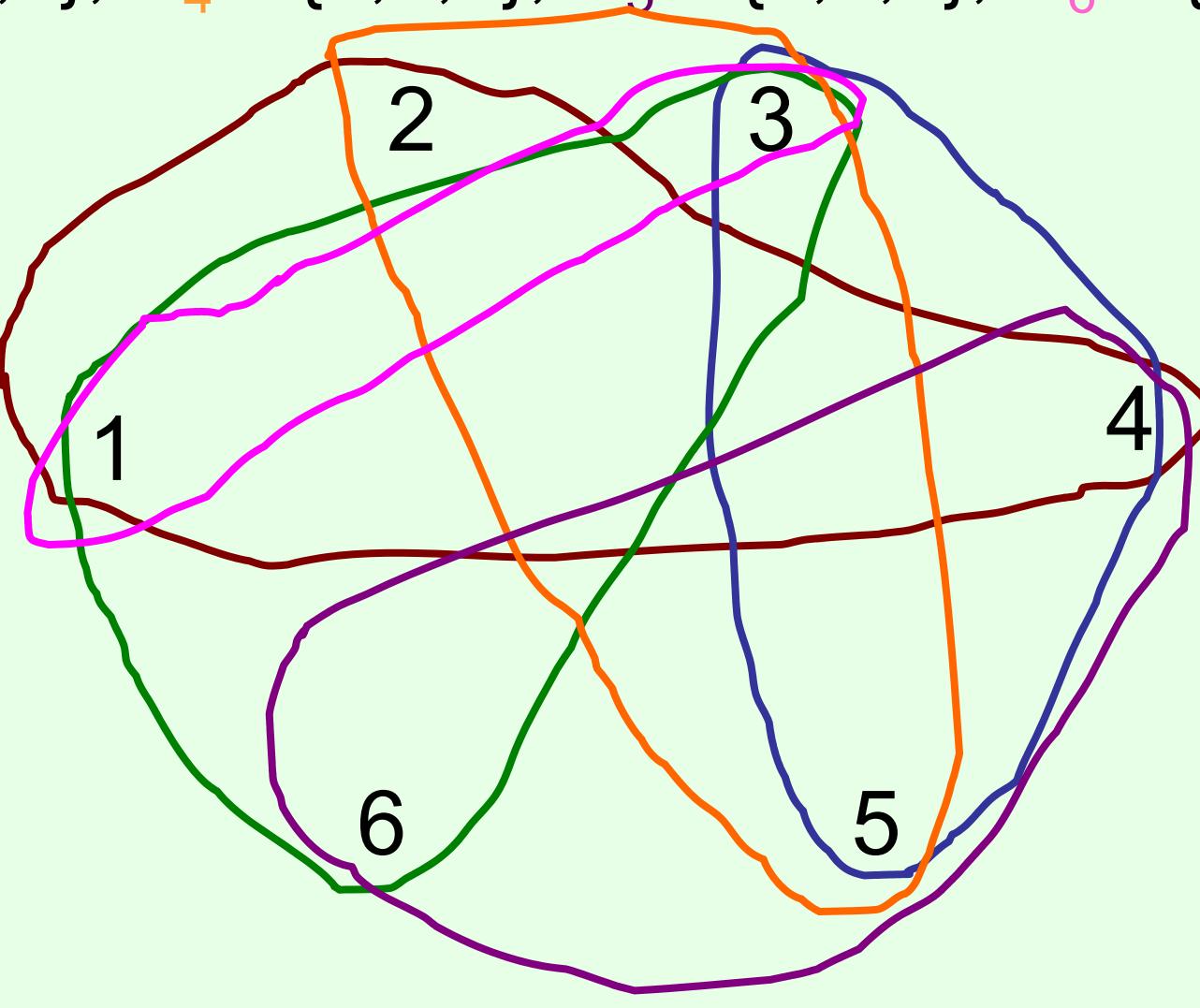
Vincent Conitzer

Set Cover (a computational problem)

- We are given:
 - A finite set $S = \{1, \dots, n\}$
 - A collection of subsets of S : S_1, S_2, \dots, S_m
- We are asked:
 - Find a subset T of $\{1, \dots, m\}$ such that $\bigcup_{j \in T} S_j = S$
 - Minimize $|T|$
- Decision variant of the problem:
 - we are additionally given a target size k , and
 - asked whether a T of size at most k will suffice
- One instance of the set cover problem:
 $S = \{1, \dots, 6\}, S_1 = \{1,2,4\}, S_2 = \{3,4,5\}, S_3 = \{1,3,6\}, S_4 = \{2,3,5\}, S_5 = \{4,5,6\}, S_6 = \{1,3\}$

Visualizing Set Cover

- $S = \{1, \dots, 6\}$, $S_1 = \{1,2,4\}$, $S_2 = \{3,4,5\}$, $S_3 = \{1,3,6\}$, $S_4 = \{2,3,5\}$, $S_5 = \{4,5,6\}$, $S_6 = \{1,3\}$



Using glpsol to solve set cover instances

- How do we model set cover as an integer program?
- See examples

Algorithms and runtime

- We saw:
 - the **runtime** of glpsol on set cover instances increases rapidly as the instances' sizes increase
 - if we drop the integrality constraint, can scale to larger instances
- Questions:
 - Using glpsol on our integer program **formulation** is but one **algorithm** – maybe other algorithms are faster?
 - different formulation; different optimization package (e.g., CPLEX); simply going through all the combinations one by one; ...
 - What is “fast enough”?
 - Do (mixed) integer programs always take more time to solve than linear programs?
 - Do set cover instances **fundamentally** take a long time to solve?

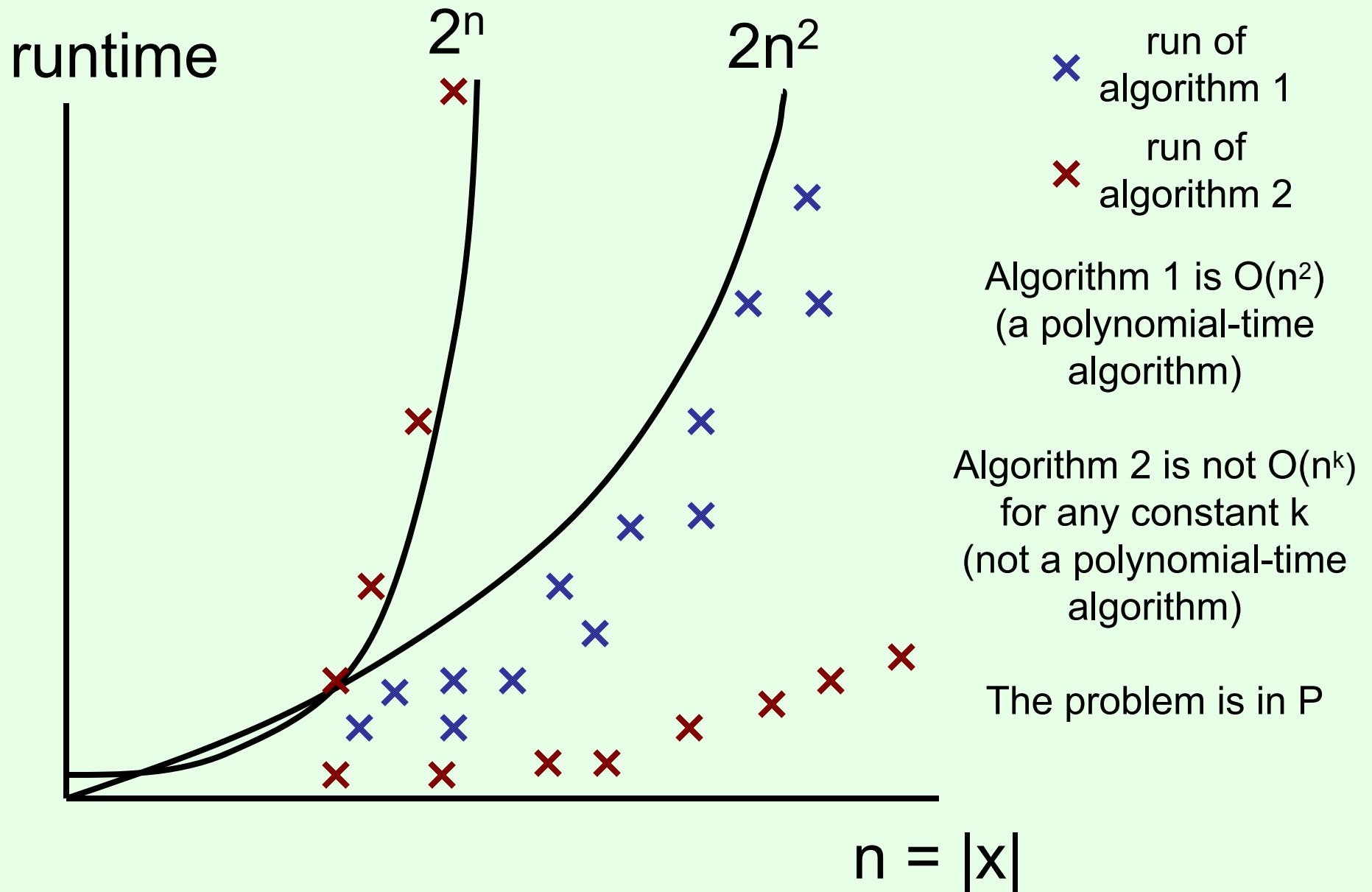
A simpler problem: sorting (see associated spreadsheet)

- Given a list of numbers, sort them
- **(Really) dumb algorithm:** Randomly perturb the numbers. See if they happen to be ordered. If not, randomly perturb the whole list again, etc.
- **Reasonably smart algorithm:** Find the smallest number. List it first. Continue on to the next number, etc.
- **Smart algorithm (MergeSort):**
 - It is easy to merge two lists of numbers, each of which is already sorted, into a single sorted list
 - So: divide the list into two equal parts, sort each part with some method, then merge the two sorted lists into a single sorted list
 - ... actually, to sort each of the parts, we can *again* use MergeSort! (The algorithm “calls itself” as a subroutine. This idea is called *recursion*.) Etc.

Polynomial time

- Let $|x|$ be the **size** of problem instance x (e.g., the size of the file in the .lp language)
- Let a be an algorithm for the problem
- Suppose that for **any** x , $\text{runtime}(a,x) < cf(|x|)$ for some constant c and function f
Then we say algorithm a 's runtime is $O(f(|x|))$
- a is a **polynomial-time algorithm** if it is $O(f(|x|))$ for some **polynomial** function f
- **P** is the class of all problems that have at least one polynomial-time algorithm
- Many people consider an algorithm **efficient** if and only if it is polynomial-time

Two algorithms for a problem



Linear programming and (mixed) integer programming

- LP and (M)IP are also computational problems
- LP is in P
 - Ironically, the most commonly used LP algorithms are not polynomial-time (but “usually” polynomial time)
- (M)IP is not known to be in P
 - Most people consider this unlikely

Reductions

- Sometimes you can reformulate problem A in terms of problem B (i.e., **reduce** A to B)
 - E.g., we have seen how to formulate several problems as linear programs or integer programs
- In this case problem A is **at most** as hard as problem B
 - Since LP is in P, all problems that we can formulate using LP are in P
 - Caveat: only true if the linear program itself can be created in polynomial time!

NP (“nondeterministic polynomial time”)

- Recall: **decision problems** require a yes or no answer
- **NP**: the class of all decision problems such that if the answer is yes, there is a simple proof of that
- E.g., “does there exist a set cover of size k ?”
- If yes, then just show which subsets to choose!
- Technically:
 - The proof must have polynomial length
 - The correctness of the proof must be verifiable in polynomial time

P vs. NP

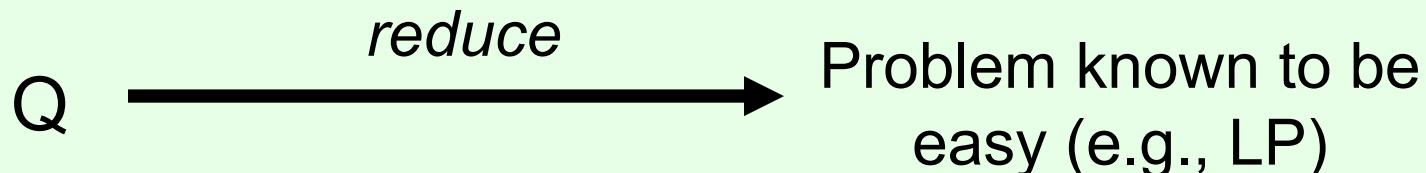
- Open problem: is it true that $P=NP$?
- The most important open problem in theoretical computer science (maybe in mathematics?)
- \$1,000,000 Clay Mathematics Institute Prize
- Most people believe P is not NP
- If P *were* equal to NP...
 - Current cryptographic techniques can be broken in polynomial time
 - Computers may be able to solve many difficult mathematical problems...
 - ... including, maybe, some other Clay Mathematics Institute Prizes!

NP-hardness

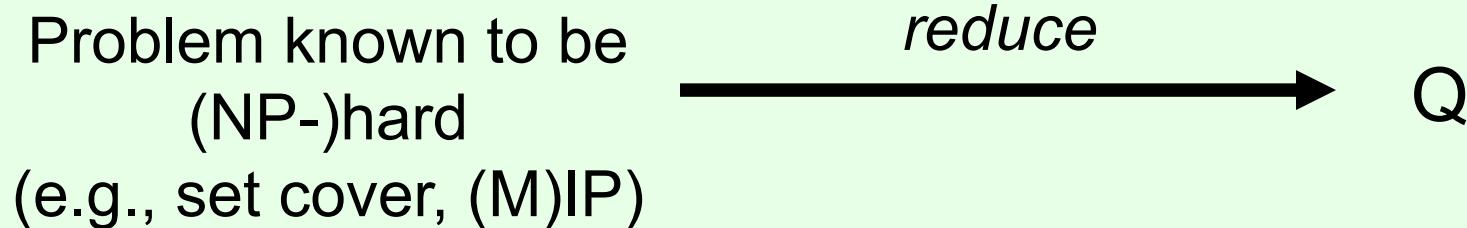
- A problem is **NP-hard** if the following is true:
 - Suppose that it is in P
 - Then $P=NP$
- So, trying to find a polynomial-time algorithm for it is like trying to prove $P=NP$
- Set cover is NP-hard
- Typical way to prove problem Q is NP-hard:
 - Take a known NP-hard problem Q'
 - Reduce it to your problem Q
 - (in polynomial time)
- E.g., (M)IP is NP-hard, because we have already reduced set cover to it
 - (M)IP is more general than set cover, so it can't be easier
- A problem is **NP-complete** if it is 1) in NP, and 2) NP-hard

Reductions:

To show problem Q is easy:



To show problem Q is (NP-)hard:

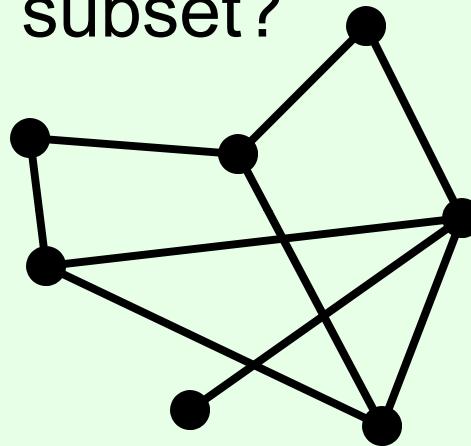


ABSOLUTELY NOT A PROOF OF NP-HARDNESS:



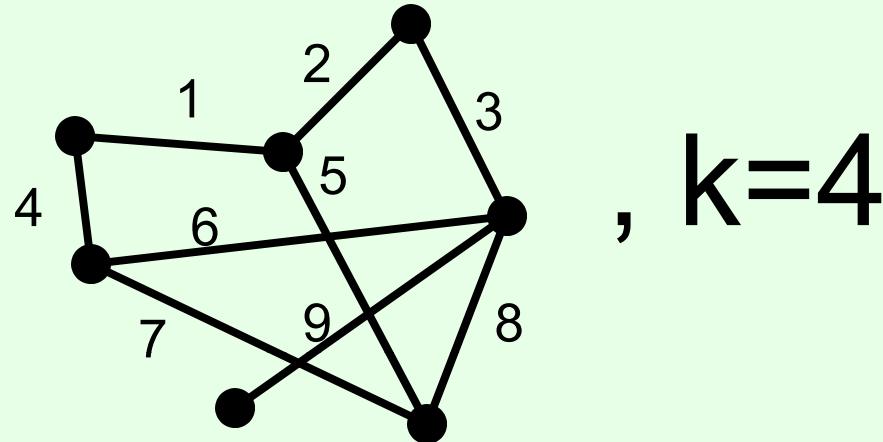
Independent Set

- In the below graph, does there exist a subset of **vertices**, of size 4, such that there is no **edge** between members of the subset?



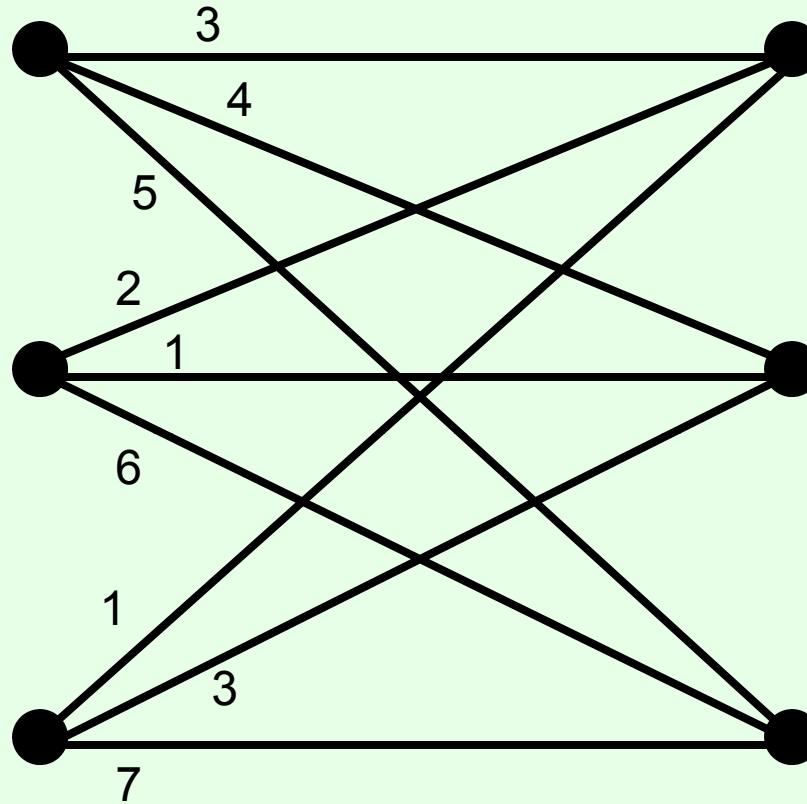
- General problem (decision variant): given a graph and a number k , are there k vertices with no edges between them?
- NP-complete

Reducing independent set to set cover



- In set cover instance (decision variant),
 - let $S = \{1,2,3,4,5,6,7,8,9\}$ (set of edges),
 - for each vertex let there be a subset with the vertex's adjacent edges: $\{1,4\}$, $\{1,2,5\}$, $\{2,3\}$, $\{4,6,7\}$, $\{3,6,8,9\}$, $\{9\}$, $\{5,7,8\}$
 - target size = #vertices - k = $7 - 4 = 3$
- Claim: answer to both instances is the same (why??)
- So which of the two problems is harder?

Weighted bipartite matching



- Match each node on the left with one node on the right (can only use each node once)
- Minimize total cost (weights on the chosen edges)

Weighted bipartite matching...

- minimize $c_{ij} x_{ij}$
- subject to
- for every i , $\sum_j x_{ij} = 1$
- for every j , $\sum_i x_{ij} = 1$
- for every i, j , $x_{ij} \geq 0$
- Theorem [Birkhoff-von Neumann]: this linear program always has an optimal solution consisting of just integers
 - and typical LP solving algorithms will return such a solution
- So weighted bipartite matching is in P

CPS 590.4

Utility theory

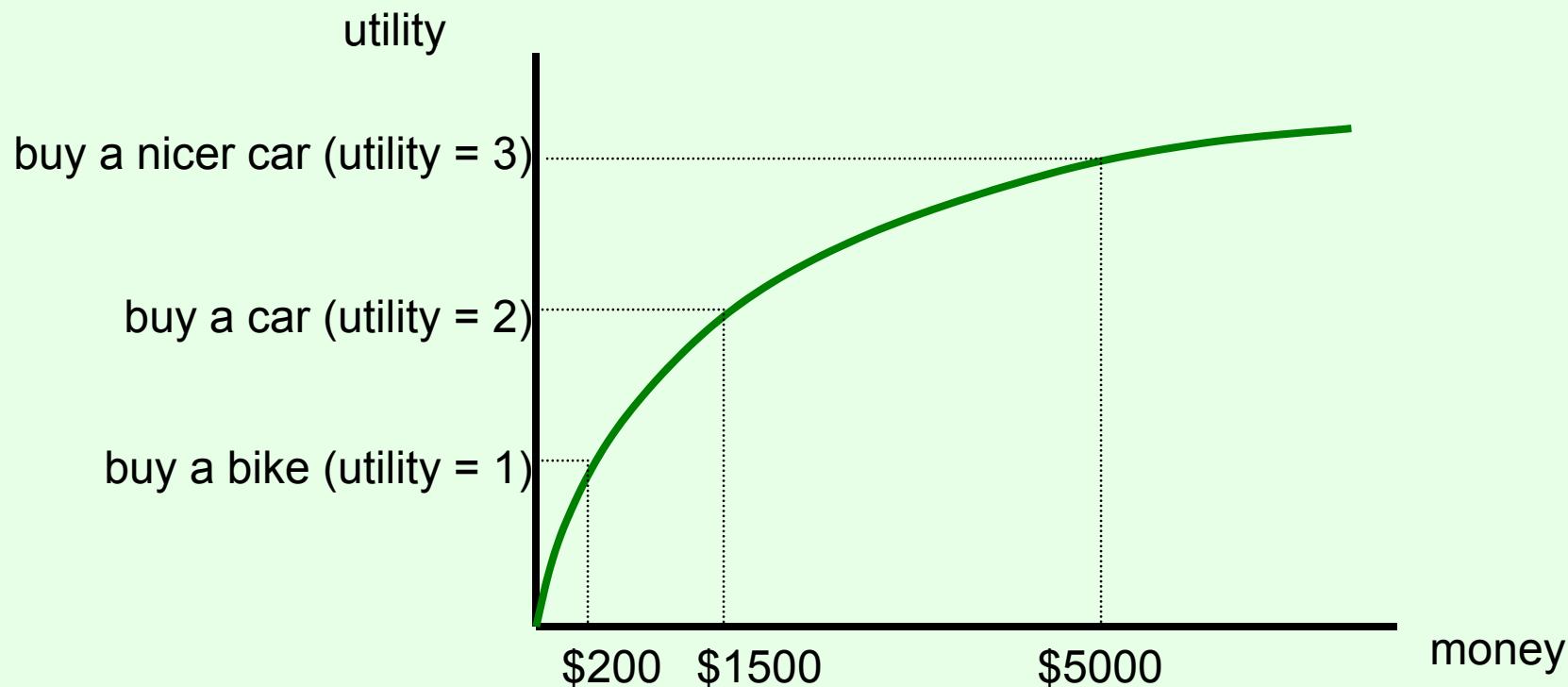
Vincent Conitzer
conitzer@cs.duke.edu

Risk attitudes

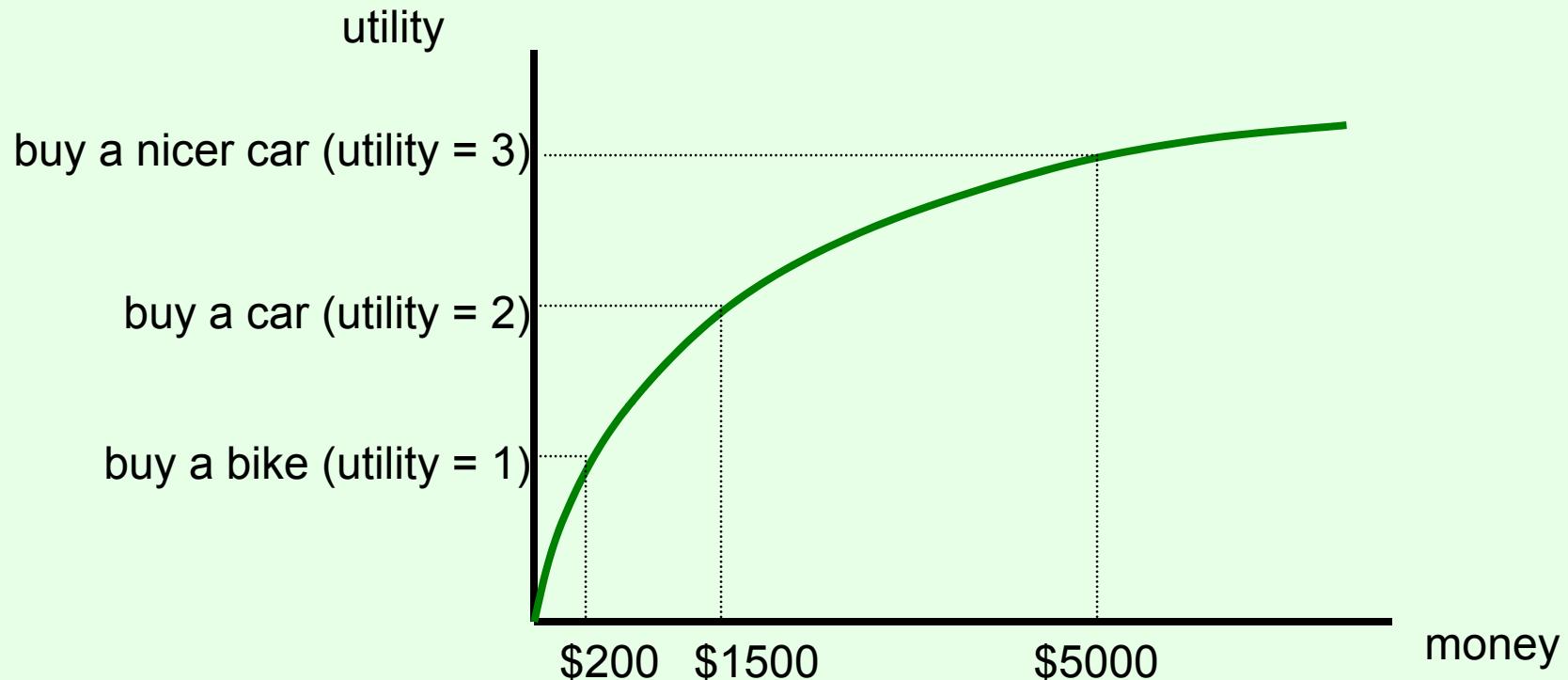
- Which would you prefer?
 - A lottery ticket that pays out \$10 with probability .5 and \$0 otherwise, or
 - A lottery ticket that pays out \$3 with probability 1
- How about:
 - A lottery ticket that pays out \$100,000,000 with probability .5 and \$0 otherwise, or
 - A lottery ticket that pays out \$30,000,000 with probability 1
- Usually, people do not simply go by expected value
- An agent is **risk-neutral** if she only cares about the expected value of the lottery ticket
- An agent is **risk-averse** if she always prefers the expected value of the lottery ticket to the lottery ticket
 - Most people are like this
- An agent is **risk-seeking** if she always prefers the lottery ticket to the expected value of the lottery ticket

Decreasing marginal utility

- Typically, at some point, having an extra dollar does not make people much happier (**decreasing marginal utility**)

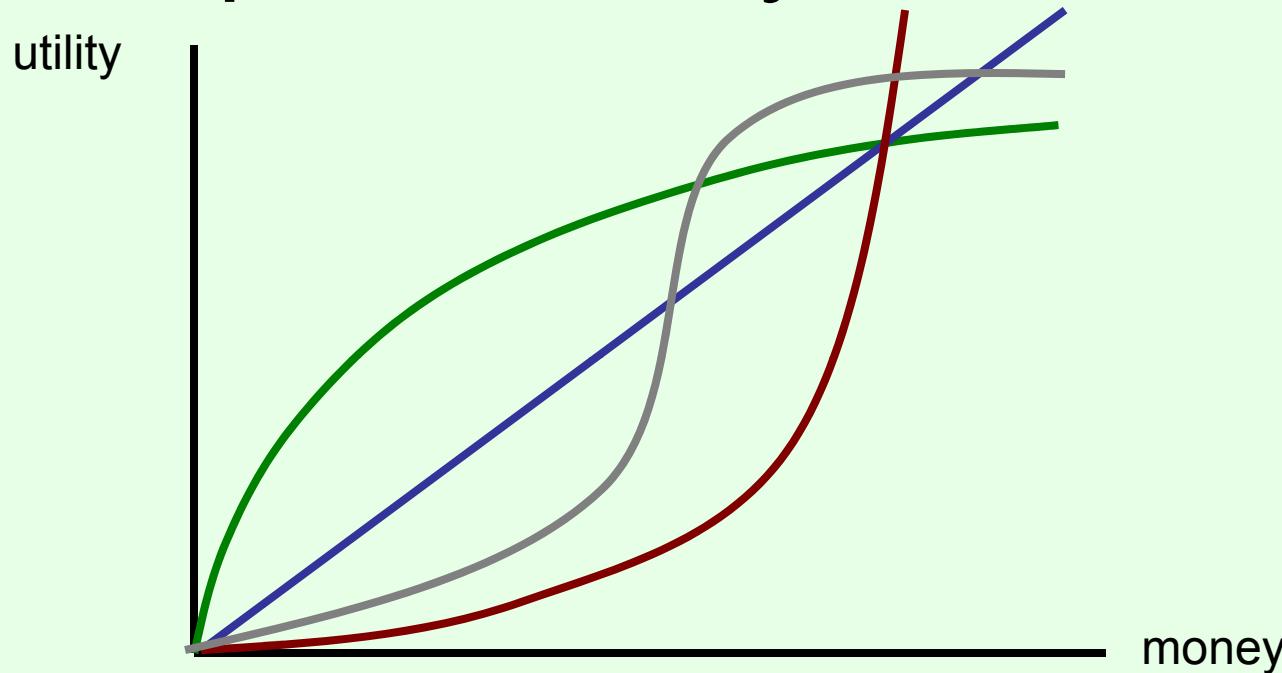


Maximizing expected utility



- Lottery 1: get \$1500 with probability 1
 - gives expected utility 2
- Lottery 2: get \$5000 with probability .4, \$200 otherwise
 - gives expected utility $.4*3 + .6*1 = 1.8$
 - (expected amount of money = $.4*\$5000 + .6*\$200 = \$2120 > \1500)
- So: maximizing expected utility is consistent with risk aversion

Different possible risk attitudes under expected utility maximization



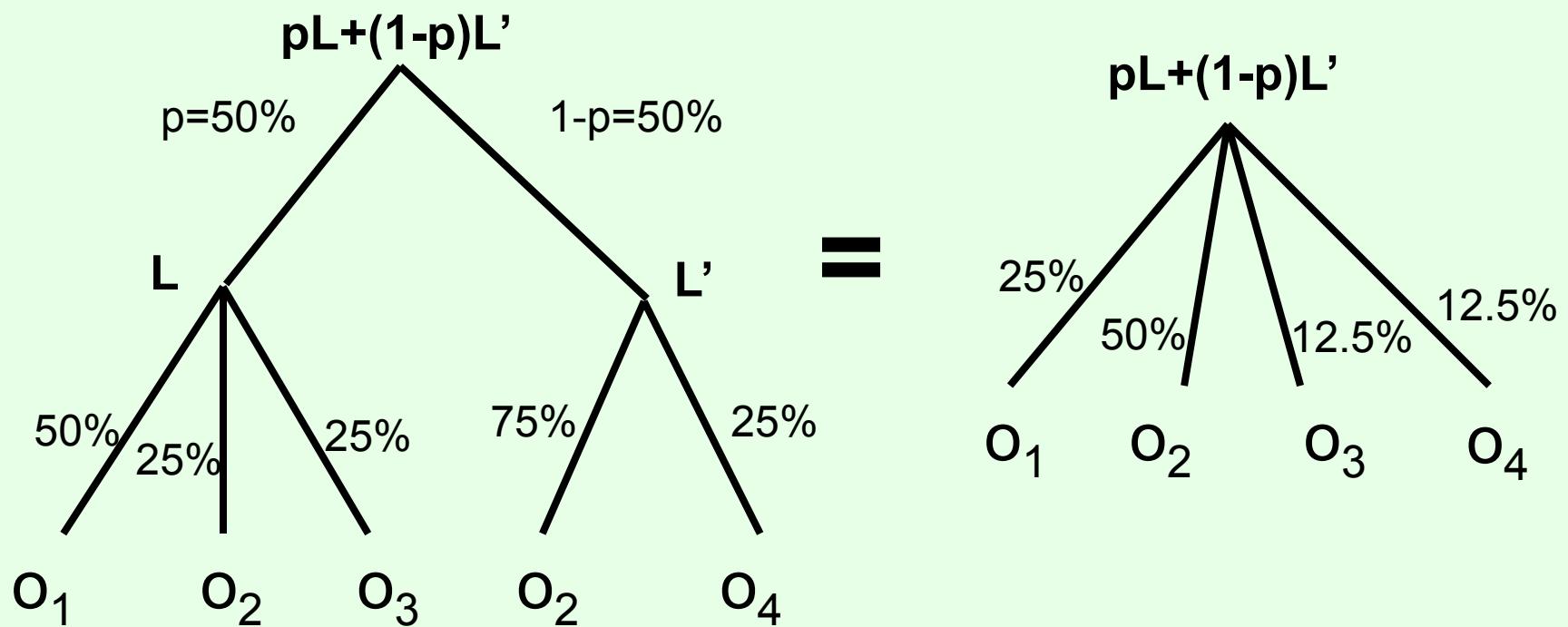
- Green has decreasing marginal utility → risk-averse
- Blue has constant marginal utility → risk-neutral
- Red has increasing marginal utility → risk-seeking
- Grey's marginal utility is sometimes increasing, sometimes decreasing → neither risk-averse (everywhere) nor risk-seeking (everywhere)

What is utility, anyway?

- Function $u: O \rightarrow \mathbb{R}$ (O is the set of “outcomes” that lotteries randomize over)
- What are its units?
 - It doesn’t really matter
 - If you replace your utility function by $u'(o) = a + bu(o)$, your behavior will be unchanged
- Why would you want to maximize expected utility?
 - This is a question about preferences over lotteries

Compound lotteries

- For two lottery tickets L and L' , let $pL + (1-p)L'$ be the “compound” lottery ticket where you get lottery ticket L with probability p , and L' with probability $1-p$



Sufficient conditions for expected utility

- $L \geq L'$ means that L is (weakly) preferred to L'
 - (\geq should be complete, transitive)
- **Expected utility theorem.** Suppose
 - (continuity axiom) for all $L, L', L'', \{p: pL + (1-p)L' \geq L''\}$ and $\{p: pL + (1-p)L' \leq L''\}$ are closed sets, and
 - (independence axiom – more controversial) for all $L, L', L'', p > 0$, we have $L \geq L'$ if and only if $pL + (1-p)L'' \geq pL' + (1-p)L''$

Then, there exists a function $u: O \rightarrow \mathbb{R}$ so that $L \geq L'$ if and only if L gives a higher expected value of u than L'

CPS 590.4

Normal-form games

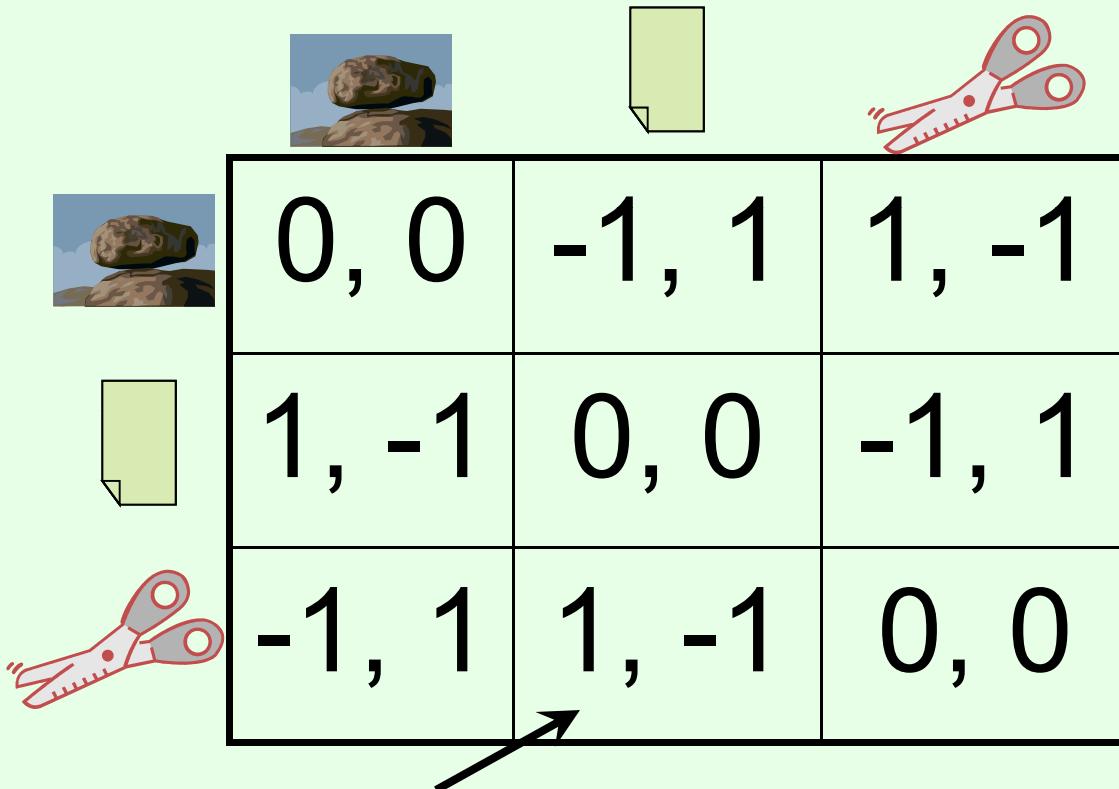
Vincent Conitzer
conitzer@cs.duke.edu

“2/3 of the average” game

- Everyone writes down a number between 0 and 100
- Person closest to 2/3 of the average wins
- Example:
 - A says 50
 - B says 10
 - C says 90
 - Average(50, 10, 90) = 50
 - 2/3 of average = 33.33
 - A is closest ($|50-33.33| = 16.67$), so A wins

Rock-paper-scissors

Column player aka.
player 2
(simultaneously)
chooses a column



A 3x3 matrix game diagram for Rock-paper-scissors. The columns represent Player 2's actions: Rock (top), Paper (middle), and Scissors (bottom). The rows represent Player 1's actions: Rock (left), Paper (middle), and Scissors (right). The payoffs are listed as (Row Player, Column Player).

			
	0, 0	-1, 1	1, -1
	1, -1	0, 0	-1, 1
	-1, 1	1, -1	0, 0

Row player
aka. player 1
chooses a row

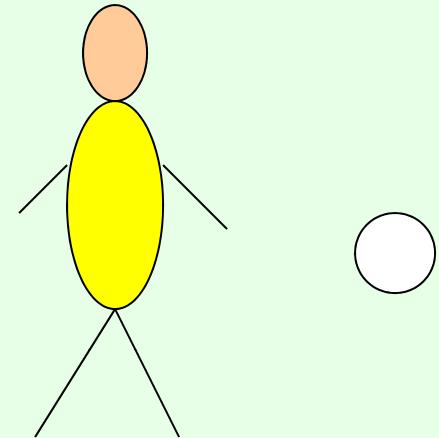
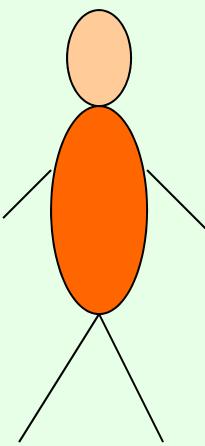
A row or column is
called an action or
(pure) strategy

Row player's utility is always listed first, column player's second

Zero-sum game: the utilities in each entry sum to 0 (or a constant)

Three-player game would be a 3D table with 3 utilities per entry, etc.

Matching pennies (~penalty kick)

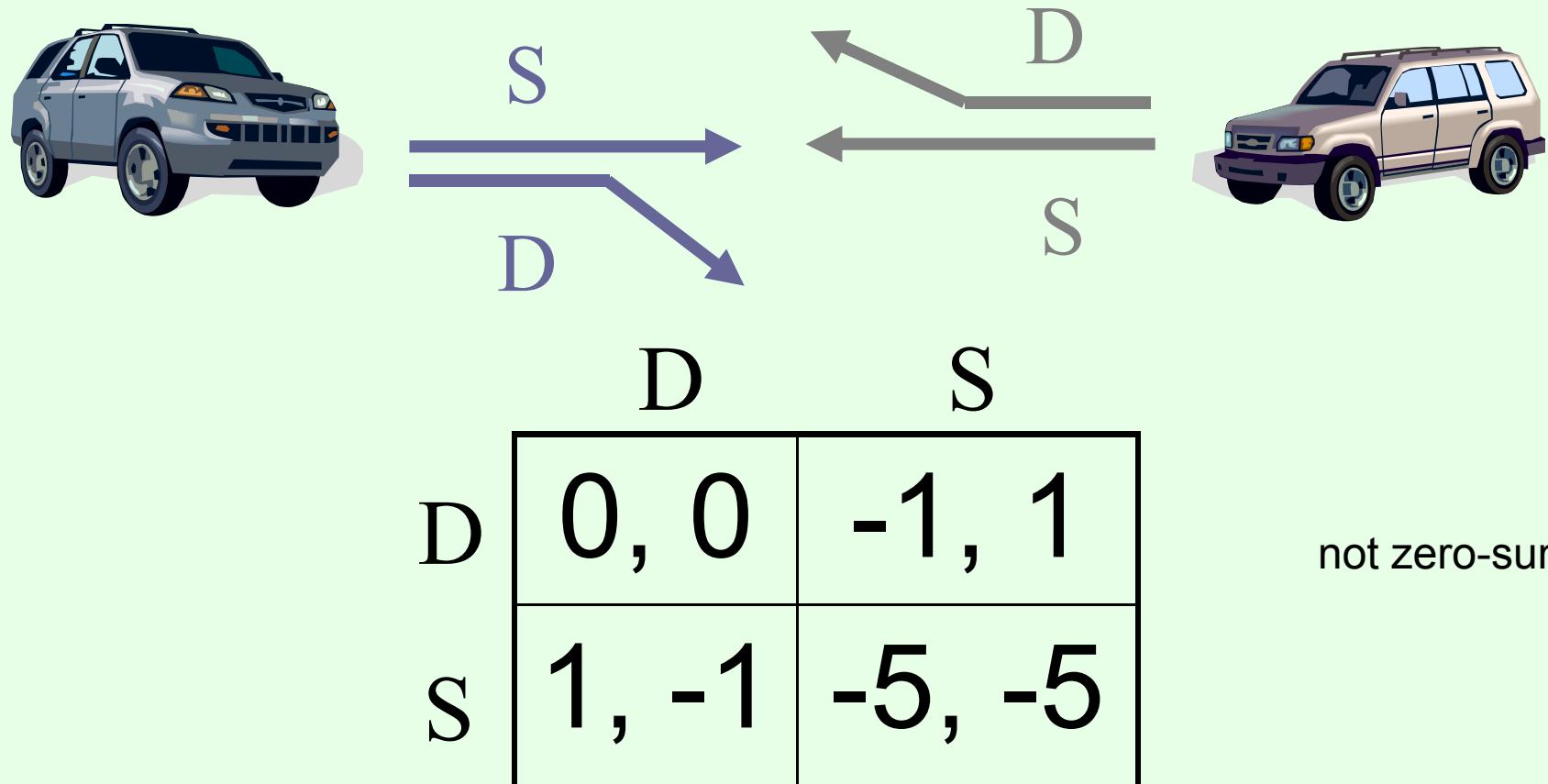


L R

L	1, -1	-1, 1
R	-1, 1	1, -1

“Chicken”

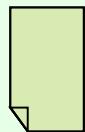
- Two players drive cars towards each other
- If one player goes straight, that player wins
- If both go straight, they both die



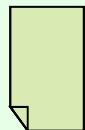
Rock-paper-scissors – Seinfeld variant



MICKEY: All right, rock beats paper!
(Mickey smacks Kramer's hand for losing)
KRAMER: I thought paper covered rock.
MICKEY: Nah, rock flies right through paper.
KRAMER: What beats rock?
MICKEY: (looks at hand) Nothing beats rock.



0, 0	1, -1	1, -1
-1, 1	0, 0	-1, 1
-1, 1	1, -1	0, 0



Dominance

- Player i 's strategy s_i **strictly dominates** s'_i if
 - for any s_{-i} , $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$
- s_i **weakly dominates** s'_i if
 - for any s_{-i} , $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$; and
 - for some s_{-i} , $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$

$-i = \text{"the player(s) other than } i\text{"}$

0, 0	1, -1	1, -1
-1, 1	0, 0	-1, 1
-1, 1	1, -1	0, 0

strict dominance

weak dominance

Prisoner's Dilemma

- Pair of criminals has been caught
- District attorney has evidence to convict them of a minor crime (1 year in jail); knows that they committed a major crime together (3 years in jail) but cannot prove it
- Offers them a deal:
 - If both confess to the major crime, they each get a 1 year reduction
 - If only one confesses, that one gets 3 years reduction

The matrix shows the payoffs for two players (Prisoner A and Prisoner B) based on their actions:

- Confess / Confess: (-2, -2)
- Confess / Don't Confess: (0, -3)
- Don't Confess / Confess: (-3, 0)
- Don't Confess / Don't Confess: (-1, -1)

Annotations:

- A blue curved arrow points from the "confess" label on the left to the "confess" label above the matrix.
- A blue curved arrow points from the "don't confess" label on the left to the "don't confess" label above the matrix.
- A blue curved arrow points from the "3 years reduction" text in the list to the value "-3" in the (Confess, Don't Confess) cell.

	confess	don't confess
confess	-2, -2	0, -3
don't confess	-3, 0	-1, -1

“Should I buy an SUV?”

purchasing cost



cost: 5



cost: 3

accident cost



cost: 5



cost: 8

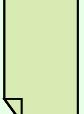


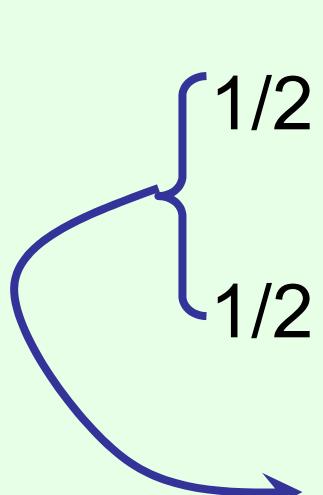
cost: 5



-10, -10	-7, -11
-11, -7	-8, -8

Mixed strategies

- Mixed strategy for player i = probability distribution over player i's (pure) strategies
- E.g., $1/3$  , $1/3$  , $1/3$ 
- Example of dominance by a mixed strategy:



	3, 0	0, 0
0, 0	3, 0	
1, 0	1, 0	

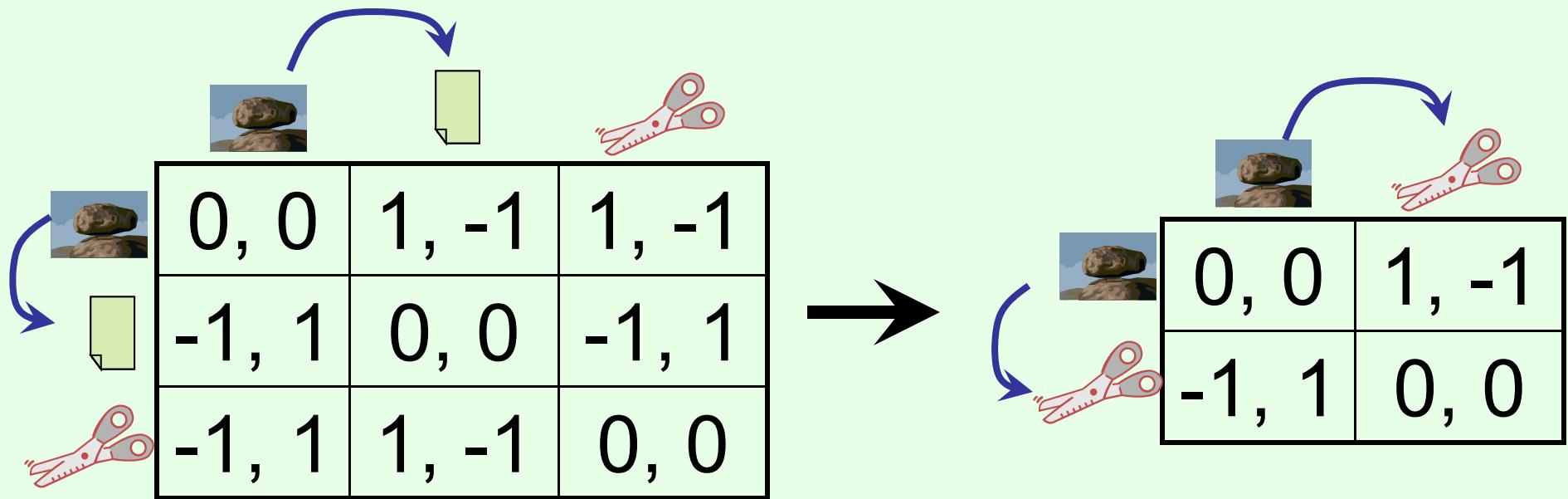
Usage:
 σ_i denotes a mixed strategy,
 s_i denotes a pure strategy

Checking for dominance by mixed strategies

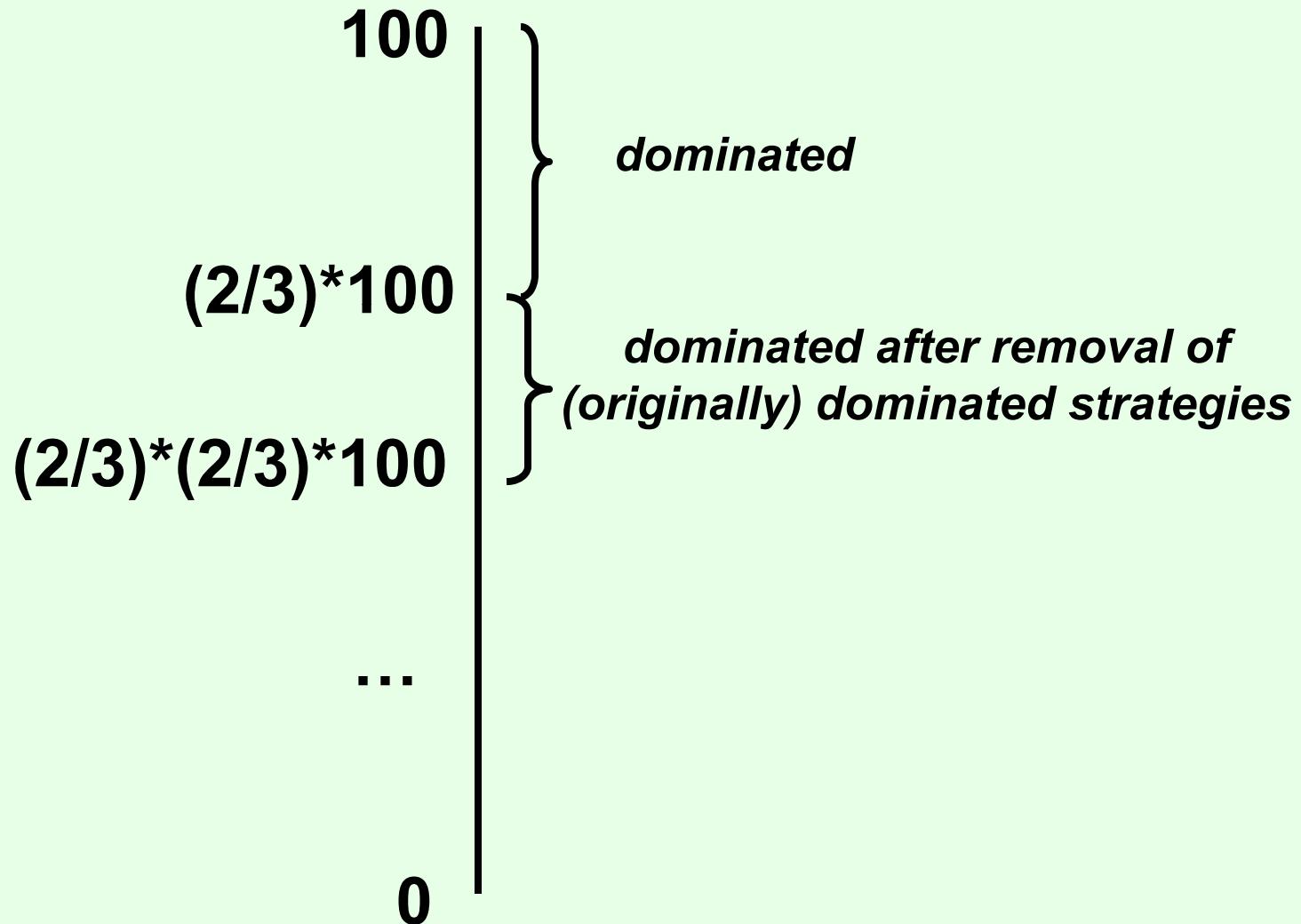
- Linear program for checking whether strategy s_i^* is **strictly** dominated by a mixed strategy:
 - maximize ε
 - such that:
 - for any s_{-i} , $\sum_{s_i} p_{s_i} u_i(s_i, s_{-i}) \geq u_i(s_i^*, s_{-i}) + \varepsilon$
 - $\sum_{s_i} p_{s_i} = 1$
- Linear program for checking whether strategy s_i^* is **weakly** dominated by a mixed strategy:
 - maximize $\sum_{s_{-i}} [(\sum_{s_i} p_{s_i} u_i(s_i, s_{-i})) - u_i(s_i^*, s_{-i})]$
 - such that:
 - for any s_{-i} , $\sum_{s_i} p_{s_i} u_i(s_i, s_{-i}) \geq u_i(s_i^*, s_{-i})$
 - $\sum_{s_i} p_{s_i} = 1$

Iterated dominance

- Iterated dominance: remove (strictly/weakly) dominated strategy, repeat
- Iterated strict dominance on Seinfeld's RPS:

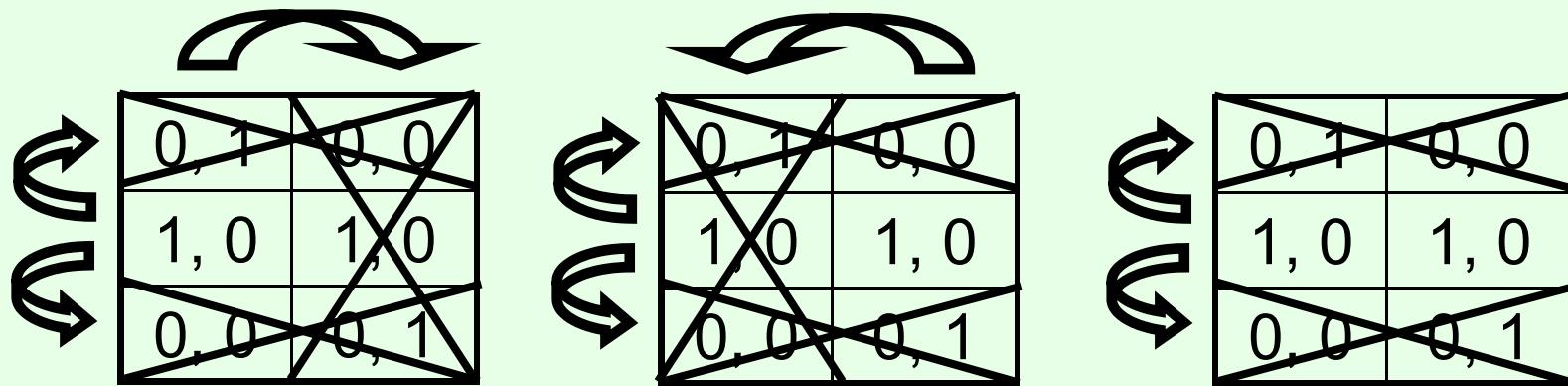


“2/3 of the average” game revisited



Iterated dominance: path (in)dependence

Iterated weak dominance is **path-dependent**:
sequence of eliminations may determine which
solution we get (if any)
(whether or not dominance by mixed strategies allowed)



Iterated strict dominance is **path-independent**: elimination
process will always terminate at the same point
(whether or not dominance by mixed strategies allowed)

Two computational questions for iterated dominance

- 1. Can a given strategy be eliminated using iterated dominance?
- 2. Is there some path of elimination by iterated dominance such that only one strategy per player remains?
- For strict dominance (with or without dominance by mixed strategies), both can be solved in polynomial time due to path-independence:
 - Check if any strategy is dominated, remove it, repeat
- For weak dominance, both questions are NP-hard (even when all utilities are 0 or 1), with or without dominance by mixed strategies [Conitzer, Sandholm 05]
 - Weaker version proved by [Gilboa, Kalai, Zemel 93]

Two-player zero-sum games revisited

- Recall: in a zero-sum game, payoffs in each entry sum to zero
 - ... or to a constant: recall that we can subtract a constant from anyone's utility function without affecting their behavior
- What one player gains, the other player loses

		
	0, 0	-1, 1
	1, -1	0, 0
	-1, 1	1, -1

Note: a general-sum k -player game can be modeled as a zero-sum $(k+1)$ -player game by adding a dummy player absorbing the remaining utility, so zero-sum games with 3 or more players have to deal with the difficulties of general-sum games; this is why we focus on 2-player zero-sum games here.

Best-response strategies

- Suppose you know your opponent's mixed strategy
 - E.g., your opponent plays rock 50% of the time and scissors 50%
- What is the best strategy for you to play?
- Rock gives $.5*0 + .5*1 = .5$
- Paper gives $.5*1 + .5*(-1) = 0$
- Scissors gives $.5*(-1) + .5*0 = -.5$
- So the best response to this opponent strategy is to (always) play rock
- There is always some **pure** strategy that is a best response
 - Suppose you have a mixed strategy that is a best response; then every one of the pure strategies that that mixed strategy places positive probability on must also be a best response

How to play matching pennies

		<i>Them</i>	
		L	R
<i>Us</i>	L	1, -1	-1, 1
	R	-1, 1	1, -1

- Assume opponent knows our **mixed** strategy
- If we play L 60%, R 40%...
- ... opponent will play R...
- ... we get $.6*(-1) + .4*(1) = -.2$
- What's optimal for us? What about rock-paper-scissors?

Matching pennies with a sensitive target

		<i>Them</i>	
		L	R
<i>Us</i>	L	1, -1	-1, 1
	R	-2, 2	1, -1

- If we play 50% L, 50% R, opponent will attack L
 - We get $.5*(1) + .5*(-2) = -.5$
- What if we play 55% L, 45% R?
- Opponent has choice between
 - L: gives them $.55*(-1) + .45*(2) = .35$
 - R: gives them $.55*(1) + .45*(-1) = .1$
- We get $.35 > -.5$

Matching pennies with a sensitive target

		<i>Them</i>	
		L	R
<i>Us</i>	L	1, -1	-1, 1
	R	-2, 2	1, -1

- What if we play 60% L, 40% R?
- Opponent has choice between
 - L: gives them $.6*(-1) + .4*(2) = .2$
 - R: gives them $.6*(1) + .4*(-1) = .2$
- We get -.2 either way
- This is the **maximin** strategy
 - Maximizes our minimum utility

Let's change roles

		<i>Them</i>	
		L	R
<i>Us</i>	L	1, -1	-1, 1
	R	-2, 2	1, -1

- Suppose **we** know **their** strategy
- If they play 50% L, 50% R,
 - We play L, we get $.5*(1)+.5*(-1) = 0$
- If they play 40% L, 60% R,
 - If we play L, we get $.4*(1)+.6*(-1) = -.2$
 - If we play R, we get $.4*(-2)+.6*(1) = -.2$
- This is the **minimax** strategy

von Neumann's minimax theorem [1928]: maximin value = minimax value (~LP duality)

Minimax theorem [von Neumann 1928]

- Maximin utility: $\max_{\sigma_i} \min_{s_{-i}} u_i(\sigma_i, s_{-i})$
 $(= - \min_{\sigma_i} \max_{s_{-i}} u_i(\sigma_i, s_{-i}))$
- Minimax utility: $\min_{\sigma_{-i}} \max_{s_i} u_i(s_i, \sigma_{-i})$
 $(= - \max_{\sigma_{-i}} \min_{s_i} u_i(s_i, \sigma_{-i}))$
- Minimax theorem:
$$\max_{\sigma_i} \min_{s_{-i}} u_i(\sigma_i, s_{-i}) = \min_{\sigma_{-i}} \max_{s_i} u_i(s_i, \sigma_{-i})$$
- Minimax theorem does not hold with pure strategies only (example?)

Practice games

20, -20	0, 0
0, 0	10, -10

20, -20	0, 0	10, -10
0, 0	10, -10	8, -8

Solving for minimax strategies using linear programming

- maximize u_i
- subject to
 - for any s_{-i} , $\sum_{s_i} p_{s_i} u_i(s_i, s_{-i}) \geq u_i$
 - $\sum_{s_i} p_{s_i} = 1$

Can also convert linear programs to two-player zero-sum games, so they are equivalent

General-sum games

- You could still play a minimax strategy in general-sum games
 - I.e., pretend that the opponent is only trying to hurt you
- But this is not rational:

0, 0	3, 1
1, 0	2, 1

- If Column was trying to hurt Row, Column would play Left, so Row should play Down
- In reality, Column will play Right (strictly dominant), so Row should play Up
- Is there a better generalization of minimax strategies in zero-sum games to general-sum games?

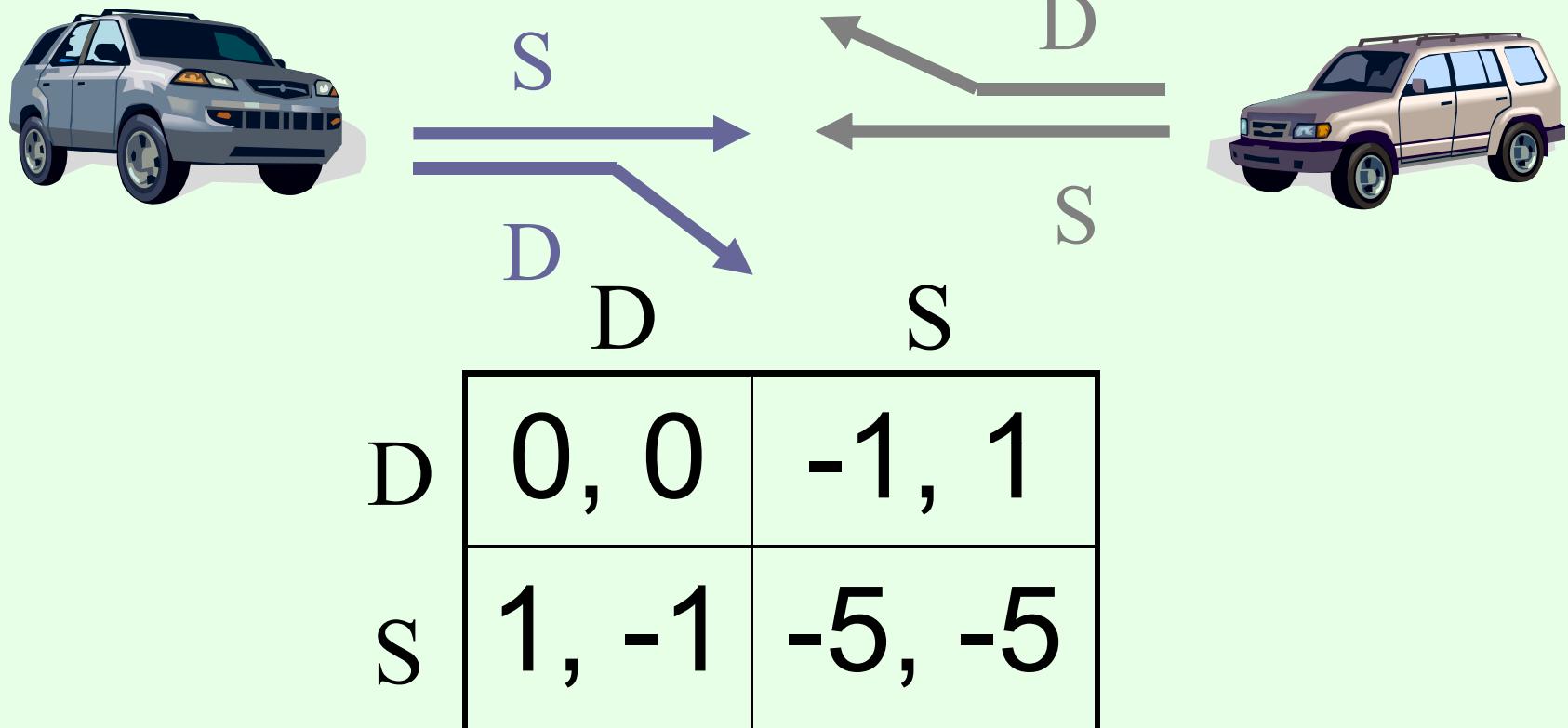
Nash equilibrium

[Nash 50]



- A vector of strategies (one for each player) is called a **strategy profile**
- A strategy profile $(\sigma_1, \sigma_2, \dots, \sigma_n)$ is a **Nash equilibrium** if each σ_i is a best response to σ_{-i}
 - That is, for any i , for any σ'_i , $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$
- Note that this does not say anything about multiple agents changing their strategies at the same time
- In any (finite) game, at least one Nash equilibrium (possibly using mixed strategies) exists [Nash 50]
- (Note - singular: equilibrium, plural: equilibria)

Nash equilibria of “chicken”



- (D, S) and (S, D) are Nash equilibria
 - They are **pure-strategy Nash equilibria**: nobody randomizes
 - They are also **strict Nash equilibria**: changing your strategy will make you strictly worse off
- No other pure-strategy Nash equilibria

Nash equilibria of “chicken”...

	D	S
D	0, 0	-1, 1
S	1, -1	-5, -5

- Is there a Nash equilibrium that uses mixed strategies? Say, where player 1 uses a mixed strategy?
- Recall: if a mixed strategy is a best response, then all of the pure strategies that it randomizes over must also be best responses
- So we need to make player 1 **indifferent** between D and S
- Player 1's utility for playing D = $-p^c_S$
- Player 1's utility for playing S = $p^c_D - 5p^c_S = 1 - 6p^c_S$
- So we need $-p^c_S = 1 - 6p^c_S$ which means $p^c_S = 1/5$
- Then, player 2 needs to be indifferent as well
- Mixed-strategy Nash equilibrium: $((4/5 D, 1/5 S), (4/5 D, 1/5 S))$
 - People may die! Expected utility $-1/5$ for each player

The presentation game

		Presenter	
		<i>Put effort into presentation (E)</i>	<i>Do not put effort into presentation (NE)</i>
Audience	<i>Pay attention (A)</i>	4, 4	-16, -14
	<i>Do not pay attention (NA)</i>	0, -2	0, 0

- Pure-strategy Nash equilibria: (A, E), (NA, NE)
- Mixed-strategy Nash equilibrium:
((1/10 A, 9/10 NA), (4/5 E, 1/5 NE))
 - Utility 0 for audience, -14/10 for presenter
 - Can see that some equilibria are strictly better for **both** players than other equilibria, i.e., some equilibria **Pareto-dominate** other equilibria

The “equilibrium selection problem”

- You are about to play a game that you have never played before with a person that you have never met
- According to which equilibrium should you play?
- Possible answers:
 - Equilibrium that maximizes the sum of utilities (**social welfare**)
 - Or, at least not a Pareto-dominated equilibrium
 - So-called **focal** equilibria
 - “Meet in Paris” game - you and a friend were supposed to meet in Paris at noon on Sunday, but you forgot to discuss where and you cannot communicate. All you care about is meeting your friend. Where will you go?
 - Equilibrium that is the convergence point of some learning process
 - An equilibrium that is easy to compute
 - ...
- Equilibrium selection is a difficult problem

Some properties of Nash equilibria

- If you can eliminate a strategy using strict dominance or even iterated strict dominance, it will not occur in any (i.e., it will be played with probability 0 in every) Nash equilibrium
 - Weakly dominated strategies may still be played in some Nash equilibrium
- In 2-player zero-sum games, a profile is a Nash equilibrium if and only if both players play minimax strategies
 - Hence, in such games, if (σ_1, σ_2) and (σ'_1, σ'_2) are Nash equilibria, then so are (σ_1, σ'_2) and (σ'_1, σ_2)
 - No equilibrium selection problem here!

How hard is it to compute *one* (any) Nash equilibrium?

- Complexity was open for a long time
 - [Papadimitriou STOC01]: “together with factoring [...] the most important concrete open question on the boundary of P today”
- Recent sequence of papers shows that computing one (any) Nash equilibrium is PPAD-complete (even in 2-player games) [Daskalakis, Goldberg, Papadimitriou 2006; Chen, Deng 2006]
- All known algorithms require exponential time (in the worst case)

What if we want to compute a Nash equilibrium with a specific property?

- For example:
 - An equilibrium that is not Pareto-dominated
 - An equilibrium that maximizes the expected social welfare (= the sum of the agents' utilities)
 - An equilibrium that maximizes the expected utility of a given player
 - An equilibrium that maximizes the expected utility of the worst-off player
 - An equilibrium in which a given pure strategy is played with positive probability
 - An equilibrium in which a given pure strategy is played with zero probability
 - ...
- All of these are NP-hard (and the optimization questions are inapproximable assuming $P \neq NP$), even in 2-player games
[Gilboa, Zemel 89; Conitzer & Sandholm IJCAI-03/GEB-08]

Search-based approaches (for 2 players)

- Suppose we know the **support** X_i of each player i's mixed strategy in equilibrium
 - That is, which pure strategies receive positive probability
- Then, we have a linear feasibility problem:
 - for both i , for any $s_i \in S_i - X_i$, $p_i(s_i) = 0$
 - for both i , for any $s_i \in X_i$, $\sum p_{-i}(s_{-i}) u_i(s_i, s_{-i}) = u_i$
 - for both i , for any $s_i \in S_i - X_i$, $\sum p_{-i}(s_{-i}) u_i(s_i, s_{-i}) \leq u_i$
- Thus, we can search over possible supports
 - This is the basic idea underlying methods in [Dickhaut & Kaplan 91; Porter, Nudelman, Shoham AAAI04/GEB08]
- Dominated strategies can be eliminated

Solving for a Nash equilibrium using MIP (2 players)

[Sandholm, Gilpin, Conitzer AAAI05]

- maximize *whatever you like* (e.g., *social welfare*)
- subject to
 - for both i , for any s_i , $\sum_{s_{-i}} p_{s_{-i}} u_i(s_i, s_{-i}) = u_{s_i}$
 - for both i , for any s_i , $u_i \geq u_{s_i}$
 - for both i , for any s_i , $p_{s_i} \leq b_{s_i}$
 - for both i , for any s_i , $u_i - u_{s_i} \leq M(1 - b_{s_i})$
 - for both i , $\sum_{s_i} p_{s_i} = 1$
- b_{s_i} is a binary variable indicating whether s_i is in the support, M is a large number

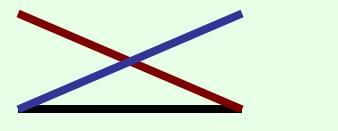
Lemke-Howson algorithm (1-slide sketch!)

	GREEN	ORANGE
RED	1, 0	0, 1
BLUE	0, 2	1, 0

player 2's utility as
function of 1's mixed
strategy

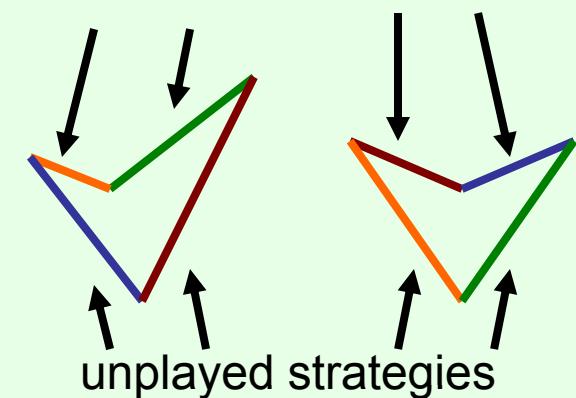


player 1's utility as
function of 2's mixed
strategy



redraw both

best-response strategies



- Strategy profile = pair of points
- Profile is an equilibrium iff every pure strategy is either a best response or unplayed
- I.e. equilibrium = pair of points that includes all the colors
 - ... except, pair of bottom points doesn't count (the "artificial equilibrium")
- Walk in some direction from the artificial equilibrium; at each step, throw out the color used twice

Correlated equilibrium [Aumann 74]

- Suppose there is a trustworthy **mediator** who has offered to help out the players in the game
- The mediator chooses a profile of pure strategies, perhaps randomly, then tells each player what her strategy is in the profile (but not what the other players' strategies are)
- A **correlated equilibrium** is a distribution over pure-strategy profiles so that every player wants to follow the recommendation of the mediator (if she assumes that the others do so as well)
- Every Nash equilibrium is also a correlated equilibrium
 - Corresponds to mediator choosing players' recommendations independently
- ... but not vice versa
- *(Note: there are more general definitions of correlated equilibrium, but it can be shown that they do not allow you to do anything more than this definition.)*

A correlated equilibrium for “chicken”

	D	S
D	0, 0 20%	-1, 1 40%
S	1, -1 40%	-5, -5 0%

- Why is this a correlated equilibrium?
- Suppose the mediator tells the row player to Dodge
- From Row's perspective, the conditional probability that Column was told to Dodge is $20\% / (20\% + 40\%) = 1/3$
- So the expected utility of Dodging is $(2/3)*(-1) = -2/3$
- But the expected utility of Straight is $(1/3)*1 + (2/3)*(-5) = -3$
- So Row wants to follow the recommendation
- If Row is told to go Straight, he knows that Column was told to Dodge, so again Row wants to follow the recommendation
- Similar for Column

A nonzero-sum variant of rock-paper-scissors (Shapley's game [Shapley 64])

			
	0, 0 0	0, 1 1/6	1, 0 1/6
	1, 0 1/6	0, 0 0	0, 1 1/6
	0, 1 1/6	1, 0 1/6	0, 0 0

- If both choose the same pure strategy, both lose
- These probabilities give a correlated equilibrium:
- E.g. suppose Row is told to play Rock
- Row knows Column is playing either paper or scissors (50-50)
 - Playing Rock will give $\frac{1}{2}$; playing Paper will give 0; playing Scissors will give $\frac{1}{2}$
- So Rock is optimal (not uniquely)

Solving for a correlated equilibrium using linear programming (n players!)

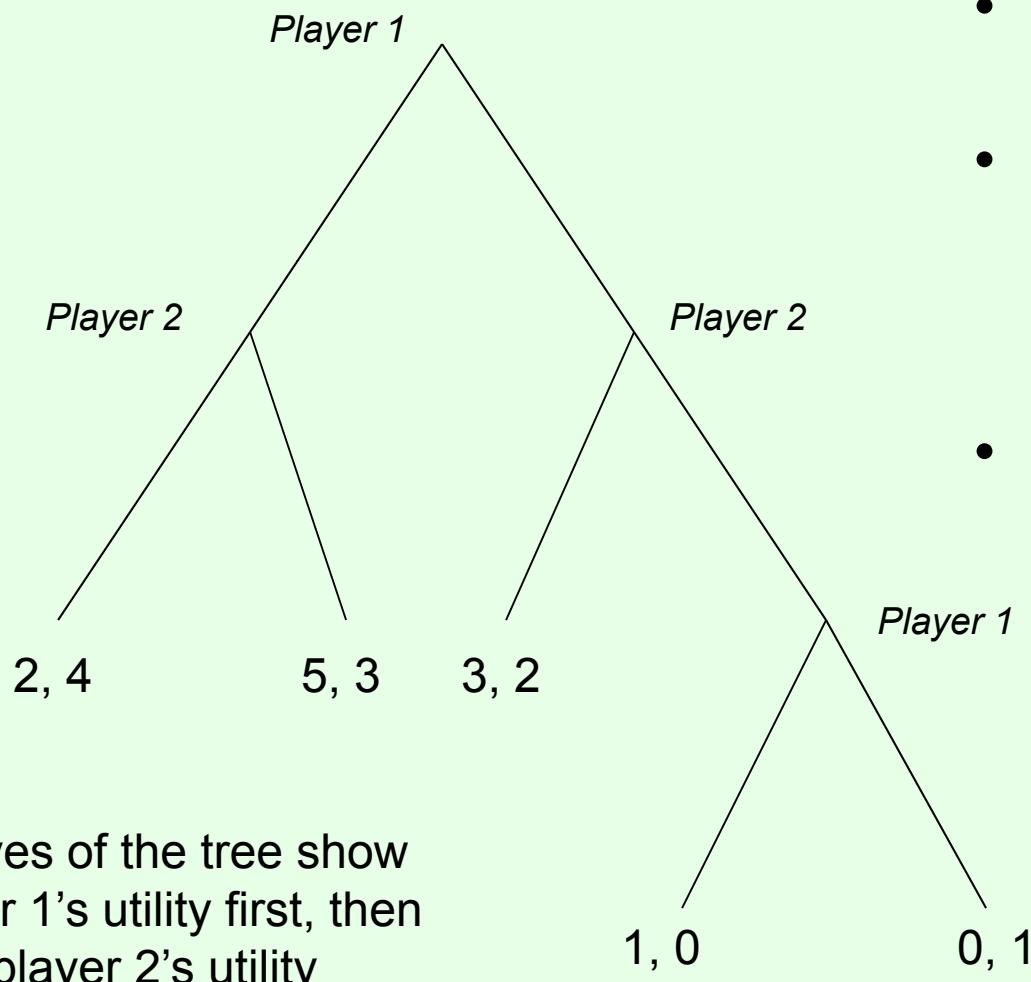
- Variables are now p_s where s is a profile of pure strategies
- maximize *whatever you like* (e.g., *social welfare*)
- subject to
 - for any $i, s_i, s'_i, \sum_{s_{-i}} p_{(s_i, s_{-i})} u_i(s_i, s_{-i}) \geq \sum_{s_{-i}} p_{(s'_i, s_{-i})} u_i(s'_i, s_{-i})$
 - $\sum_s p_s = 1$

CPS 590.4

Extensive-form games

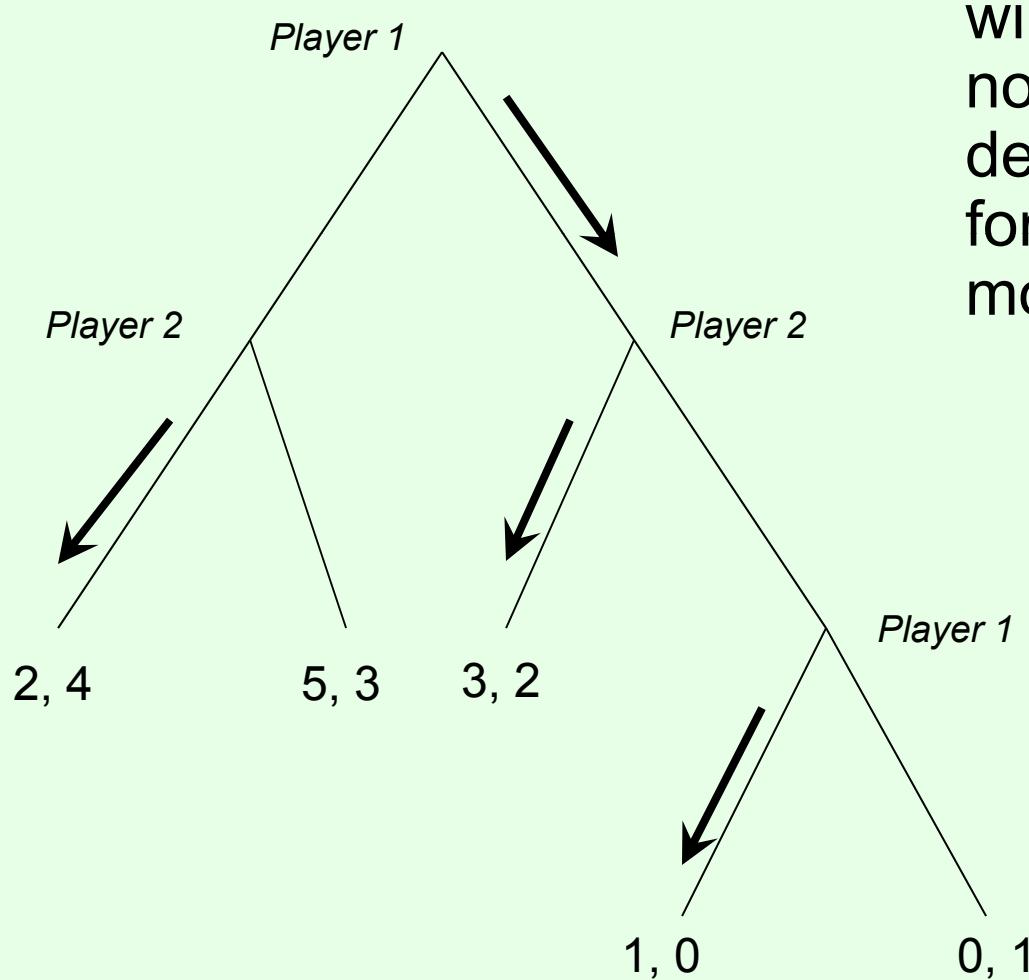
Vincent Conitzer
conitzer@cs.duke.edu

Extensive-form games with perfect information



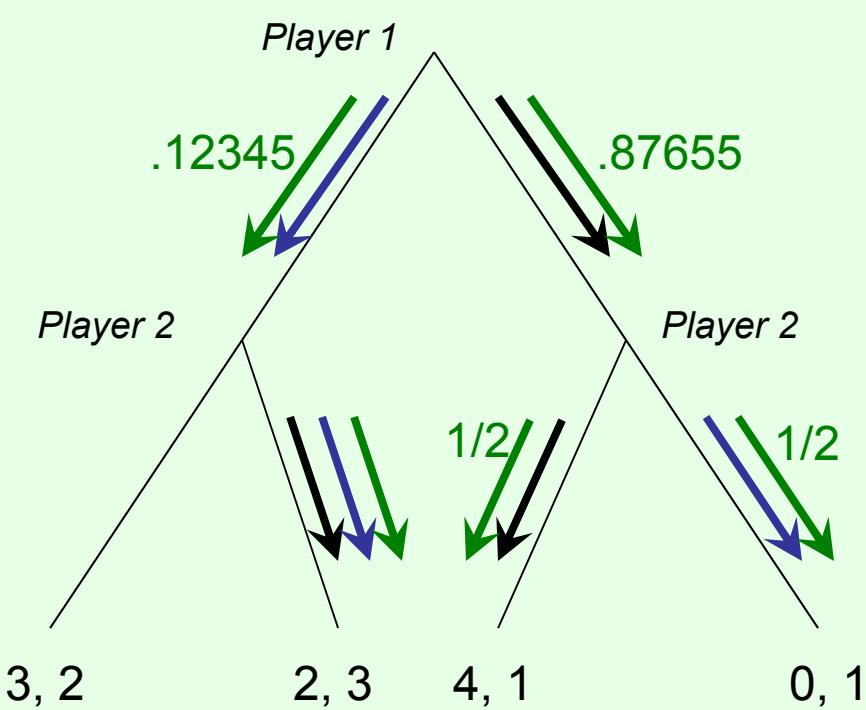
- Players do not move simultaneously
- When moving, each player is aware of all the previous moves (**perfect information**)
- A (**pure**) strategy for player i is a mapping from player i 's nodes to actions

Backward induction



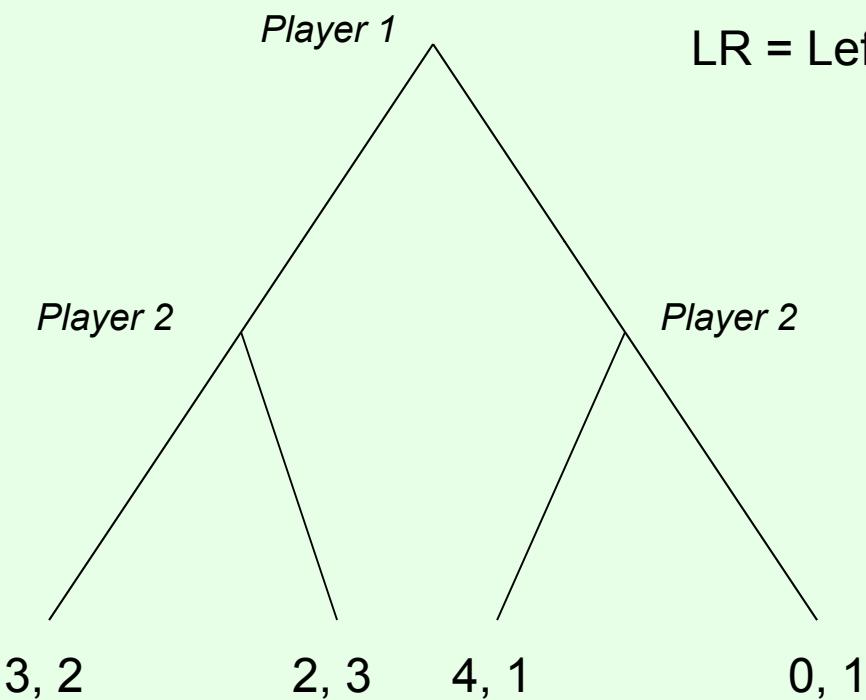
- When we know what will happen at each of a node's children, we can decide the best action for the player who is moving at that node

A limitation of backward induction



- If there are ties, then how they are broken affects what happens higher up in the tree
- Multiple **equilibria**...

Conversion from extensive to normal form

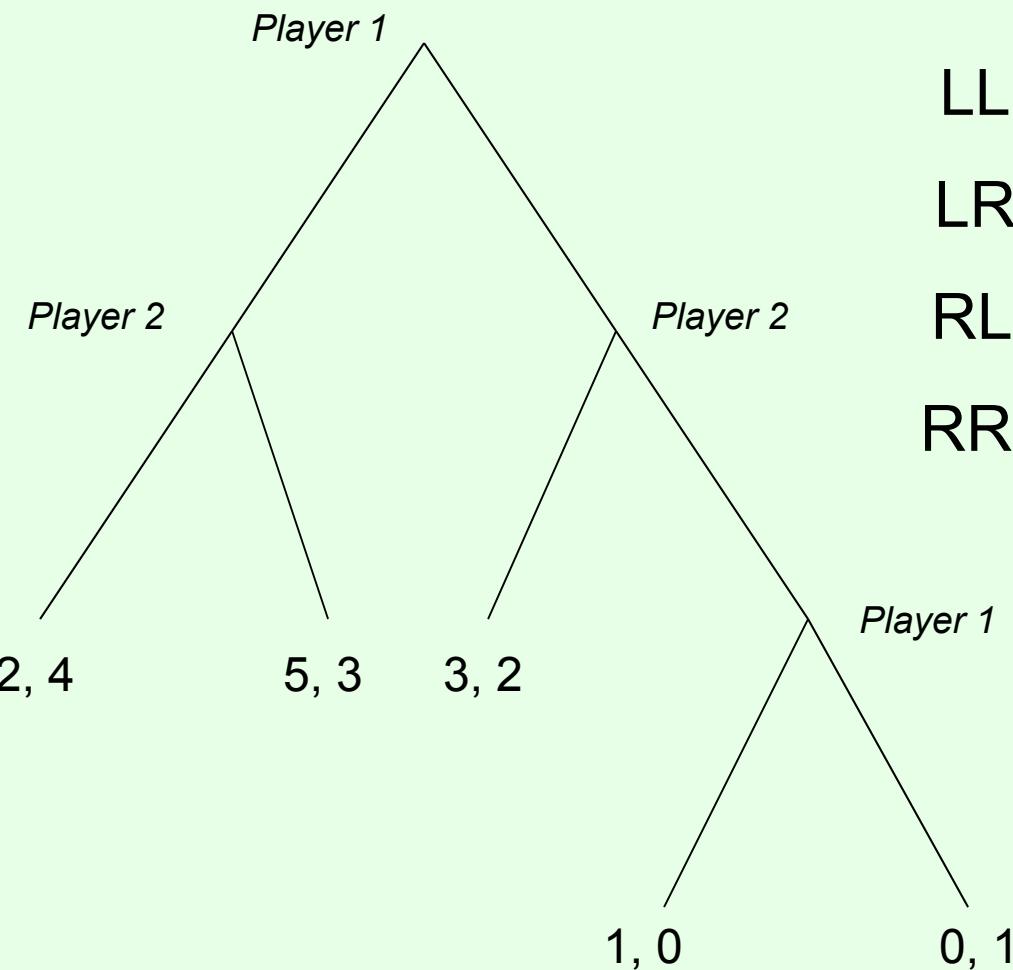


LR = Left if 1 moves Left, Right if 1 moves Right; etc.

		LL	LR	RL	RR	
		L	3, 2	3, 2	2, 3	2, 3
		R	4, 1	0, 1	4, 1	0, 1

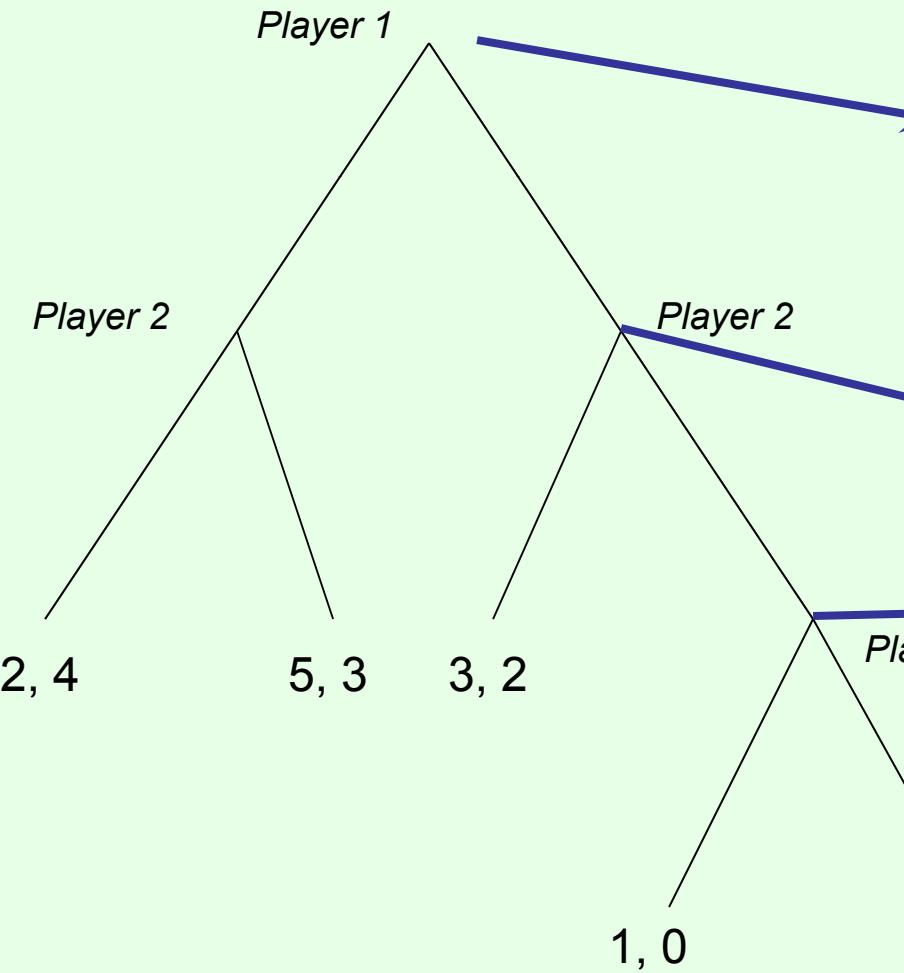
- Nash equilibria of this normal-form game include (R, LL), (R, RL), (L, RR) + infinitely many mixed-strategy equilibria
- In general, normal form can have exponentially many strategies

Converting the first game to normal form



Subgame perfect equilibrium

- Each node in a (perfect-information) game tree, together with the remainder of the game after that node is reached, is called a **subgame**
- A strategy profile is a **subgame perfect equilibrium** if it is an equilibrium for **every** subgame



	LL	LR	RL	RR
LL	2, 4	2, 4	5, 3	5, 3
LR	2, 4	2, 4	5, 3	5, 3
RL	3, 2	1, 0	3, 2	1, 0
RR	3, 2	0, 1	3, 2	0, 1

*L	*R
3, 2	1, 0
3, 2	0, 1

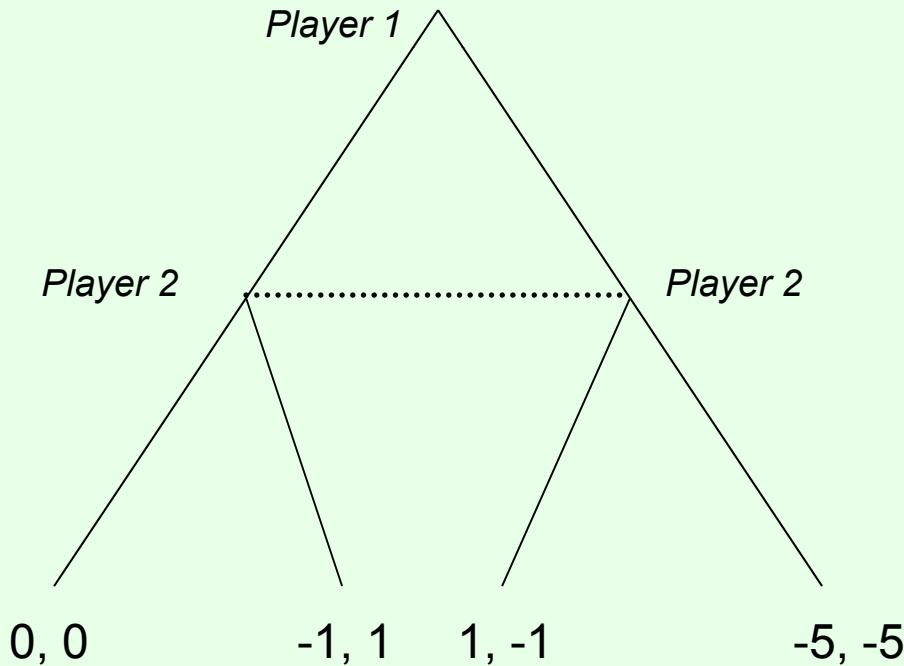
**
*L
1, 0

*R
0, 1

- (RR, LL) and (LR, LR) are not subgame perfect equilibria because (*R, **) is not an equilibrium
- (LL, LR) is not subgame perfect because (*L, *R) is not an equilibrium
- *R is not a **credible threat**

Imperfect information

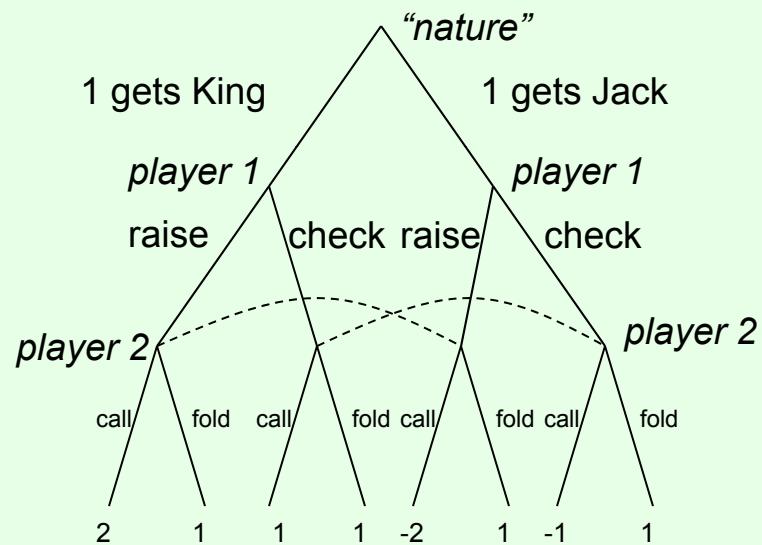
- Dotted lines indicate that a player cannot distinguish between two (or more) states
 - A set of states that are connected by dotted lines is called an **information set**
- Reflected in the normal-form representation



	L	R
L	0, 0	-1, 1
R	1, -1	-5, -5

- Any normal-form game can be transformed into an imperfect-information extensive-form game this way

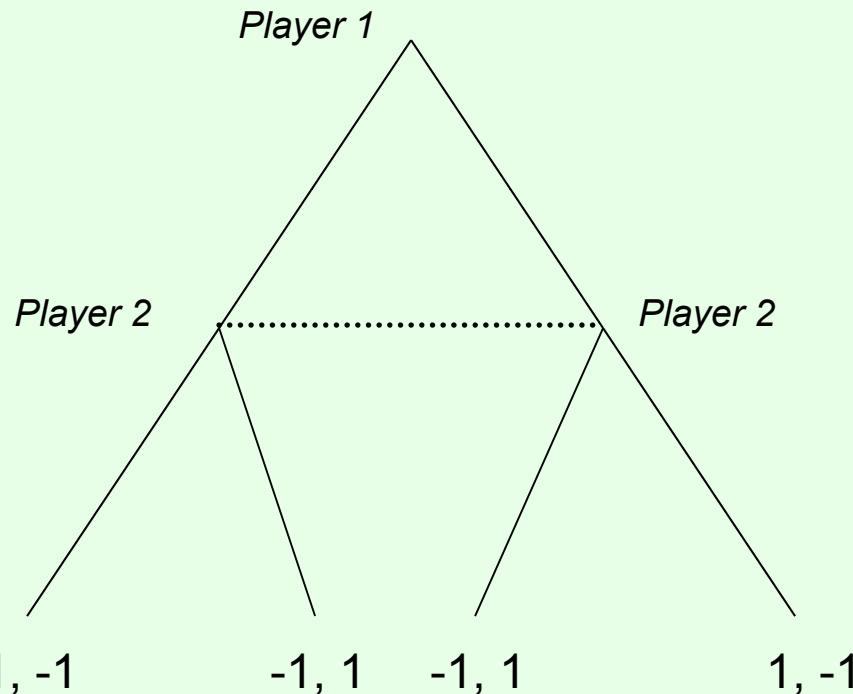
A poker-like game



		$\frac{2}{3}$ cc	$\frac{1}{3}$ cf	$\frac{1}{3}$ fc	$\frac{1}{3}$ ff	
		rr	0, 0	0, 0	1, -1	1, -1
		rc	.5, -.5	1.5, -1.5	0, 0	1, -1
		cr	$-.5, .5$	$-1.5, -1.5$	$1, -1$	$1, -1$
		cc	0, 0	1, -1	0, 0	1, -1

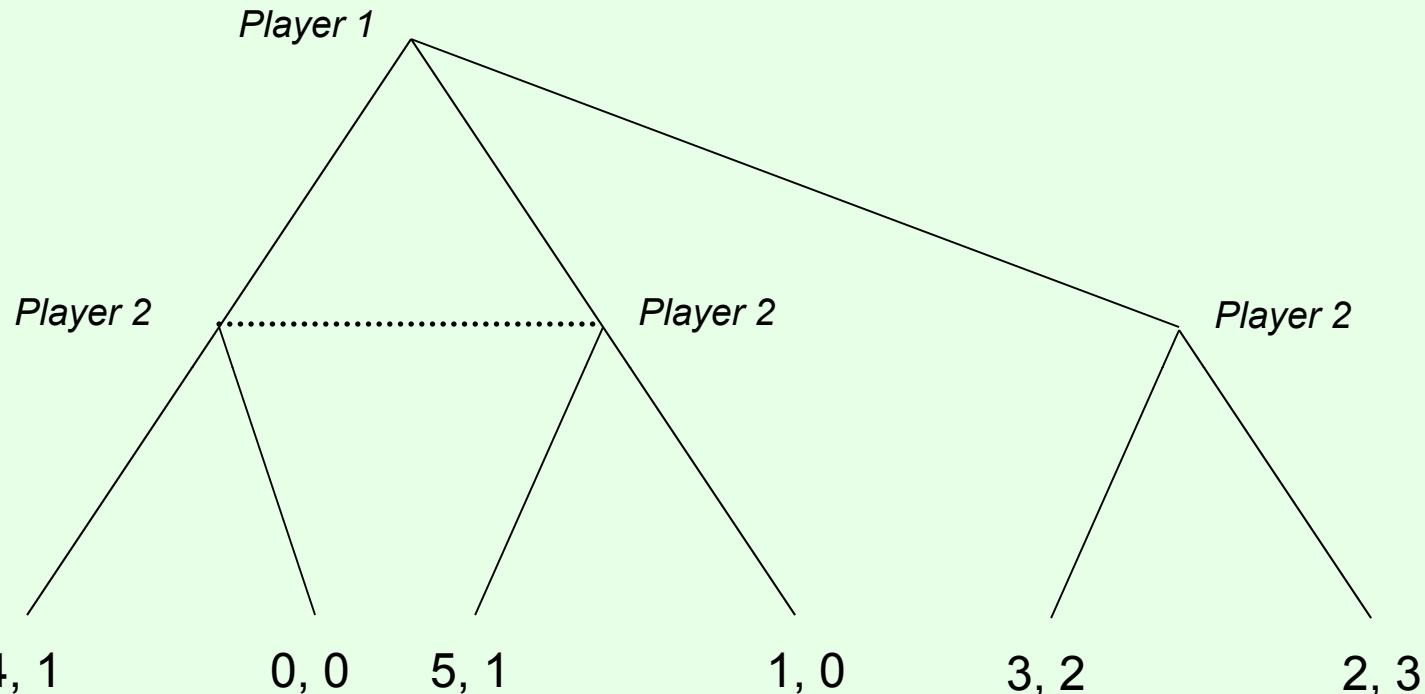
Subgame perfection and imperfect information

- How should we extend the notion of subgame perfection to games of imperfect information?



- We cannot expect Player 2 to play Right after Player 1 plays Left, and Left after Player 1 plays Right, because of the information set
- Let us say that a subtree is a subgame only if there are no information sets that connect the subtree to parts outside the subtree

Subgame perfection and imperfect information...



- One of the Nash equilibria is: (R, RR)
- Also subgame perfect (the only subgames are the whole game, and the subgame after Player 1 moves Right)
- But it is not reasonable to believe that Player 2 will move Right after Player 1 moves Left/Middle (not a credible threat)
- There exist more sophisticated refinements of Nash equilibrium that rule out such behavior

Computing equilibria in the extensive form

- Can just use normal-form representation
 - Misses issues of subgame perfection, etc.
- Another problem: there are exponentially many pure strategies, so normal form is exponentially larger
 - Even given polynomial-time algorithms for normal form, time would still be exponential in the size of the extensive form
- There are other techniques that reason directly over the extensive form and scale much better
 - E.g., using the **sequence form** of the game

Commitment

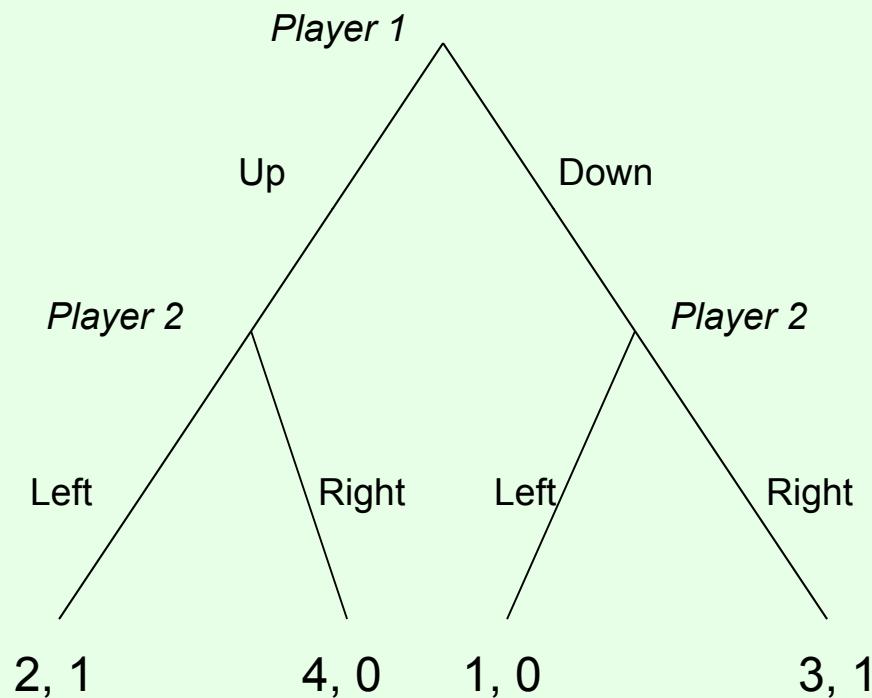
- Consider the following (normal-form) game:

2, 1	4, 0
1, 0	3, 1

- How should this game be played?
- Now suppose the game is played as follows:
 - Player 1 **commits** to playing one of the rows,
 - Player 2 observes the commitment and then chooses a column
- What is the optimal strategy for player 1?
- What if 1 can commit to a **mixed** strategy?

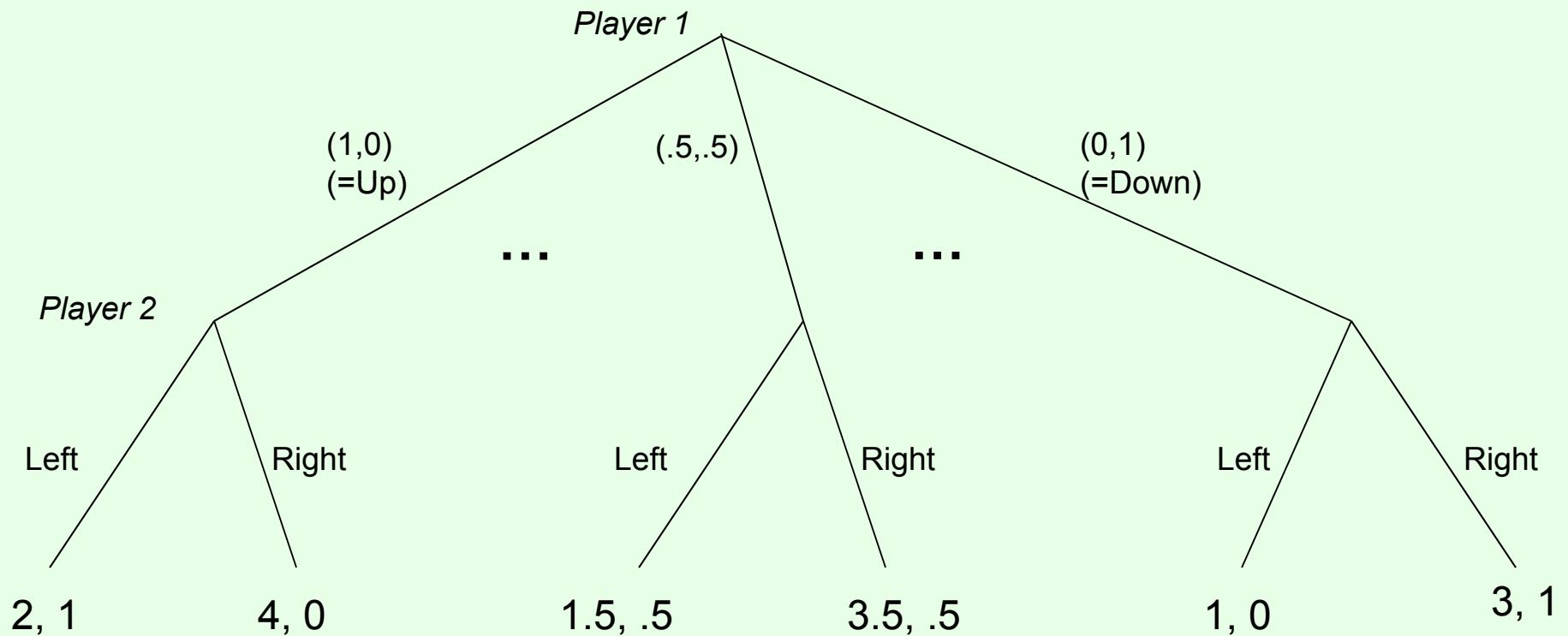
Commitment as an extensive-form game

- For the case of committing to a pure strategy:



Commitment as an extensive-form game

- For the case of committing to a mixed strategy:



- Infinite-size game; computationally impractical to reason with the extensive form here

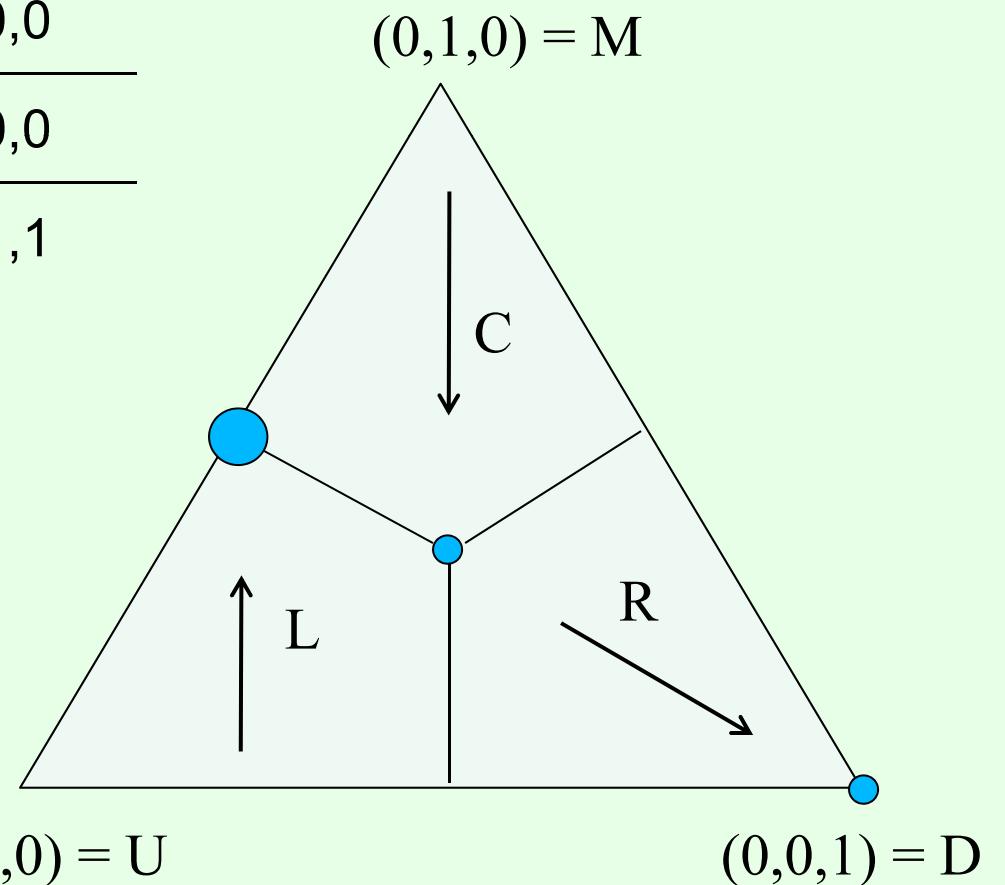
Solving for the optimal mixed strategy to commit to

[Conitzer & Sandholm 2006, von Stengel & Zamir 2010]

- For every column t separately, we will solve separately for the best mixed row strategy (defined by p_s) that induces player 2 to play t
- maximize $\sum_s p_s u_1(s, t)$
- subject to
 - for any t' , $\sum_s p_s u_2(s, t) \geq \sum_s p_s u_2(s, t')$
 - $\sum_s p_s = 1$
- (May be infeasible, e.g., if t is strictly dominated)
- Pick the t that is best for player 1

Visualization

	L	C	R
U	0,1	1,0	0,0
M	4,0	0,1	0,0
D	0,0	1,0	1,1

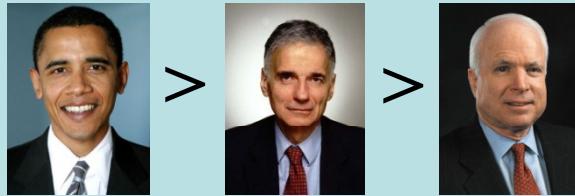


CPS 590.4

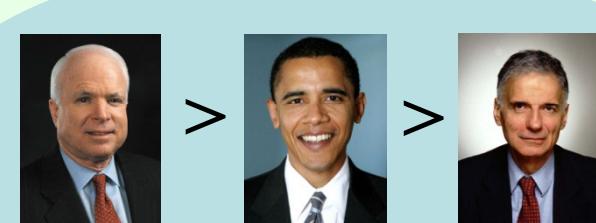
Voting and social choice

Vincent Conitzer
conitzer@cs.duke.edu

Voting over alternatives



voting rule
(mechanism)
determines winner
based on votes



- Can vote over other things too
 - Where to go for dinner tonight, other joint plans, ...

Voting (rank aggregation)

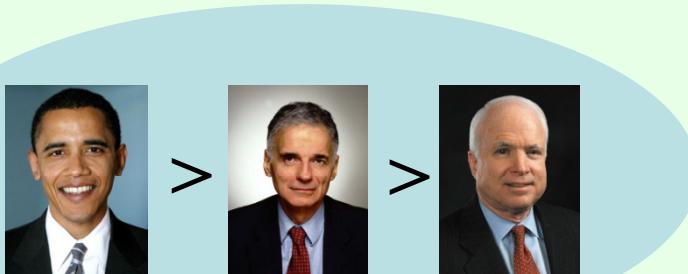
- Set of m candidates (aka. alternatives, outcomes)
- n voters; each voter ranks all the candidates
 - E.g., for set of candidates $\{a, b, c, d\}$, one possible vote is $b > a > d > c$
 - Submitted ranking is called a vote
- A voting rule takes as input a vector of votes (submitted by the voters), and as output produces either:
 - the winning candidate, or
 - an aggregate ranking of all candidates
- Can vote over just about anything
 - political representatives, award nominees, where to go for dinner tonight, joint plans, allocations of tasks/resources, ...
 - Also can consider other applications: e.g., aggregating search engines' rankings into a single ranking

Example voting rules

- Scoring rules are defined by a vector (a_1, a_2, \dots, a_m) ; being ranked i^{th} in a vote gives the candidate a_i points
 - Plurality is defined by $(1, 0, 0, \dots, 0)$ (winner is candidate that is ranked first most often)
 - Veto (or anti-plurality) is defined by $(1, 1, \dots, 1, 0)$ (winner is candidate that is ranked last the least often)
 - Borda is defined by $(m-1, m-2, \dots, 0)$
- Plurality with (2-candidate) runoff: top two candidates in terms of plurality score proceed to runoff; whichever is ranked higher than the other by more voters, wins
- Single Transferable Vote (STV, aka. Instant Runoff): candidate with lowest plurality score drops out; if you voted for that candidate, your vote transfers to the next (live) candidate on your list; repeat until one candidate remains
- Similar runoffs can be defined for rules other than plurality

Pairwise elections

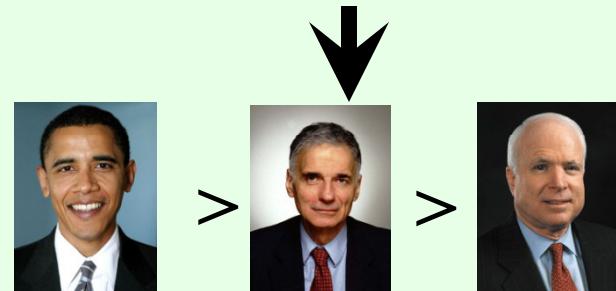
two votes prefer Obama to McCain



two votes prefer Obama to Nader

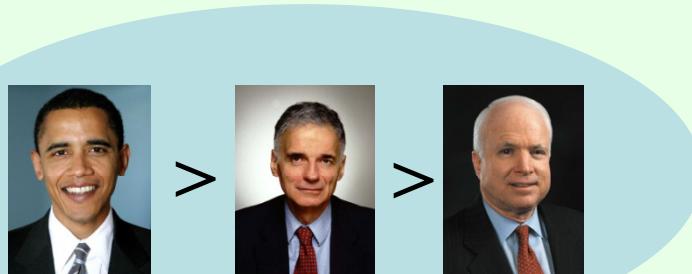


two votes prefer Nader to McCain

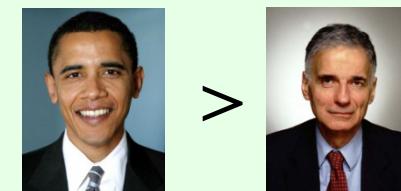
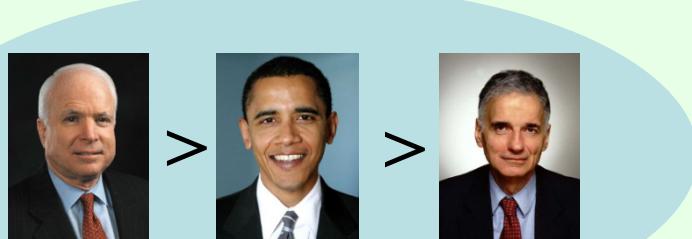


Condorcet cycles

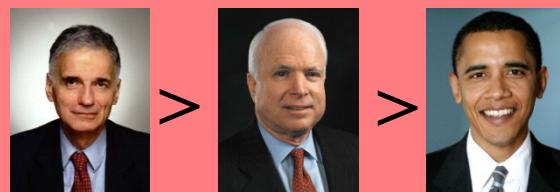
two votes prefer McCain to Obama



two votes prefer Obama to Nader



two votes prefer Nader to McCain



“weird” preferences



Voting rules based on pairwise elections

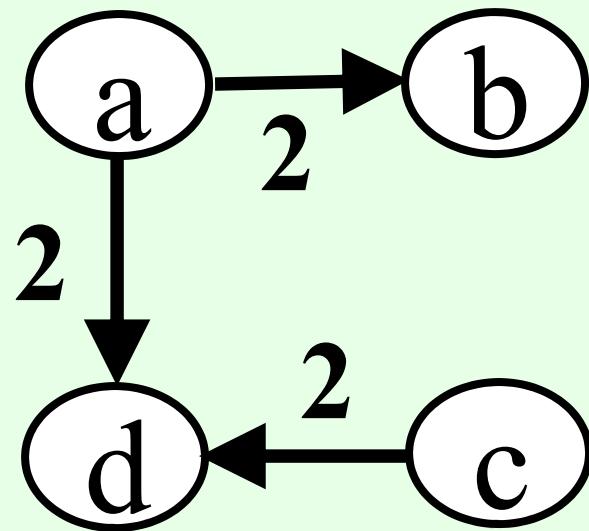
- **Copeland**: candidate gets two points for each pairwise election it wins, one point for each pairwise election it ties
- **Maximin** (aka. **Simpson**): candidate whose worst pairwise result is the best wins
- **Slater**: create an overall ranking of the candidates that is inconsistent with as few pairwise elections as possible
 - NP-hard!
- **Cup/pairwise elimination**: pair candidates, losers of pairwise elections drop out, repeat

Even more voting rules...

- **Kemeny**: create an overall ranking of the candidates that has as few *disagreements* as possible (where a disagreement is with a vote on a pair of candidates)
 - NP-hard!
- **Bucklin**: start with $k=1$ and increase k gradually until some candidate is among the top k candidates in more than half the votes; that candidate wins
- **Approval** (not a ranking-based rule): every voter labels each candidate as approved or disapproved, candidate with the most approvals wins

Pairwise election graphs

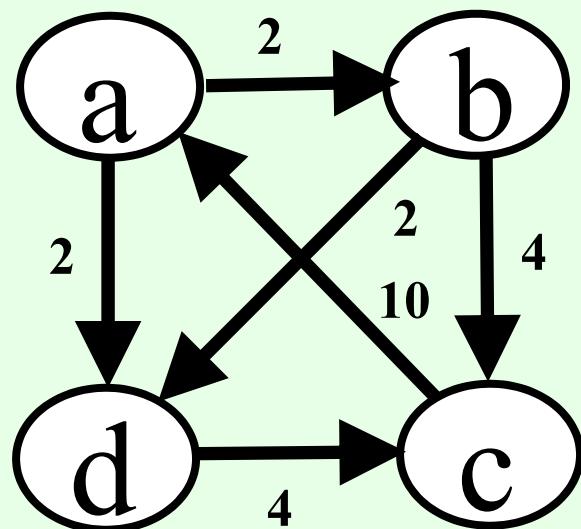
- **Pairwise election** between a and b : compare how often a is ranked above b vs. how often b is ranked above a
- Graph representation: edge from winner to loser (no edge if tie), weight = margin of victory
- E.g., for votes $a > b > c > d$, $c > a > d > b$ this gives



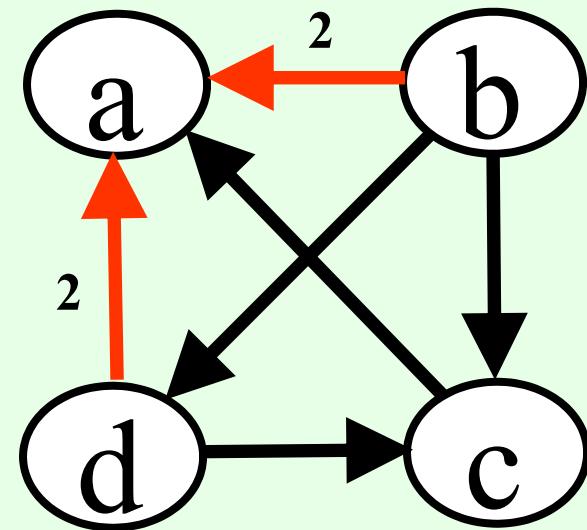
Kemeny on pairwise election graphs

- Final ranking = acyclic tournament graph
 - Edge (a, b) means a ranked above b
 - Acyclic = no cycles, tournament = edge between every pair
- Kemeny ranking seeks to minimize the total weight of the inverted edges

pairwise election graph



Kemeny ranking

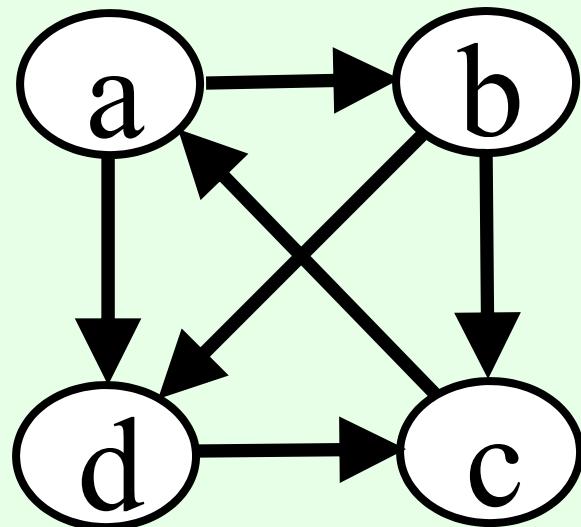


$$(b > d > c > a)$$

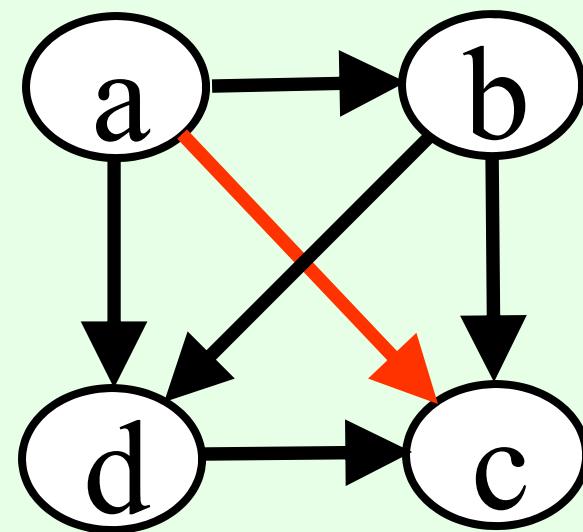
Slater on pairwise election graphs

- Final ranking = acyclic tournament graph
- Slater ranking seeks to minimize the number of inverted edges

pairwise election graph



Slater ranking



$$(a > b > d > c)$$

An integer program for computing Kemeny/Slater rankings

$y_{(a, b)}$ is 1 if a is ranked below b, 0 otherwise

$w_{(a, b)}$ is the weight on edge (a, b) (if it exists)

in the case of Slater, weights are always 1

minimize: $\sum_{e \in E} w_e y_e$

subject to:

for all $a, b \in V$, $y_{(a, b)} + y_{(b, a)} = 1$

for all $a, b, c \in V$, $y_{(a, b)} + y_{(b, c)} + y_{(c, a)} \geq 1$

Choosing a rule

- How do we choose a rule from all of these rules?
- How do we know that there does not exist another, “perfect” rule?
- Let us look at some **criteria** that we would like our voting rule to satisfy

Condorcet criterion

- A candidate is the Condorcet winner if it wins all of its pairwise elections
- Does not always exist...
- ... but the Condorcet criterion says that if it does exist, it should win

- Many rules do not satisfy this
- E.g. for plurality:
 - $b > a > c > d$
 - $c > a > b > d$
 - $d > a > b > c$
- a is the Condorcet winner, but it does not win under plurality

Majority criterion

- If a candidate is ranked first by most votes, that candidate should win
 - Relationship to Condorcet criterion?
- Some rules do not even satisfy this
- E.g. Borda:
 - $a > b > c > d > e$
 - $a > b > c > d > e$
 - $c > b > d > e > a$
- a is the majority winner, but it does not win under Borda

Monotonicity criteria

- Informally, monotonicity means that “ranking a candidate higher should help that candidate,” but there are multiple nonequivalent definitions
- A **weak** monotonicity requirement: if
 - candidate w wins for the current votes,
 - we then improve the position of w in some of the votes and leave everything else the same,then w should still win.
- E.g., STV does not satisfy this:
 - 7 votes $b > c > a$
 - 7 votes $a > b > c$
 - 6 votes $c > a > b$
- c drops out first, its votes transfer to a, a wins
- But if 2 votes $b > c > a$ change to $a > b > c$, b drops out first, its 5 votes transfer to c, and c wins

Monotonicity criteria...

- A **strong** monotonicity requirement: if
 - candidate w wins for the current votes,
 - we then change the votes in such a way that for each vote, if a candidate c was ranked below w originally, c is still ranked below w in the new votethen w should still win.
- Note the other candidates can jump around in the vote, as long as they don't jump ahead of w
- None of our rules satisfy this

Independence of irrelevant alternatives

- Independence of irrelevant alternatives criterion: if
 - the rule ranks a above b for the current votes,
 - we then change the votes but do not change which is ahead between a and b in each votethen a should still be ranked ahead of b.
- None of our rules satisfy this

Arrow's impossibility theorem [1951]

- Suppose there are at least 3 candidates
- Then there exists no rule that is simultaneously:
 - Pareto efficient (if all votes rank a above b, then the rule ranks a above b),
 - nondictatorial (there does not exist a voter such that the rule simply always copies that voter's ranking), and
 - independent of irrelevant alternatives

Muller-Satterthwaite impossibility theorem

[1977]

- Suppose there are at least 3 candidates
- Then there exists no rule that simultaneously:
 - satisfies **unanimity** (if all votes rank a first, then a should win),
 - is **nondictatorial** (there does not exist a voter such that the rule simply always selects that voter's first candidate as the winner), and
 - is **monotone** (in the strong sense).

Manipulability

- Sometimes, a voter is better off revealing her preferences insincerely, aka. **manipulating**
- E.g. plurality
 - Suppose a voter prefers $a > b > c$
 - Also suppose she knows that the other votes are
 - 2 times $b > c > a$
 - 2 times $c > a > b$
 - Voting truthfully will lead to a tie between b and c
 - She would be better off voting e.g. $b > a > c$, guaranteeing b wins
- All our rules are (sometimes) manipulable

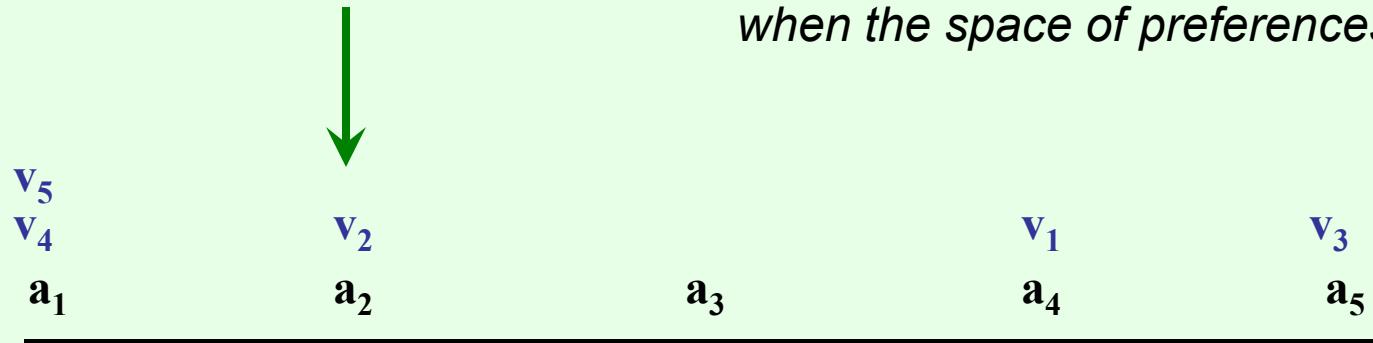
Gibbard-Satterthwaite impossibility theorem

- Suppose there are at least 3 candidates
- There exists no rule that is simultaneously:
 - **onto** (for every candidate, there are some votes that would make that candidate win),
 - **nondictatorial** (there does not exist a voter such that the rule simply always selects that voter's first candidate as the winner), and
 - **nonmanipulable**

Single-peaked preferences

- Suppose candidates are ordered on a line
- Every voter prefers candidates that are closer to her most preferred candidate
- Let every voter report only her most preferred candidate (“peak”)
- Choose the **median voter’s** peak as the winner
 - This will also be the Condorcet winner
- Nonmanipulable!

Impossibility results do not necessarily hold when the space of preferences is restricted



Some computational issues in social choice

- Sometimes computing the winner/aggregate ranking is hard
 - E.g. for Kemeny and Slater rules this is NP-hard
- For some rules (e.g., STV), computing a successful manipulation is NP-hard
 - Manipulation being hard is a **good** thing (circumventing Gibbard-Satterthwaite?)... But would like something stronger than NP-hardness
 - Also: work on the complexity of controlling the outcome of an election by influencing the list of candidates/schedule of the Cup rule/etc.
- Preference elicitation:
 - We may not want to force each voter to rank **all** candidates;
 - Rather, we can selectively query voters for parts of their ranking, according to some algorithm, to obtain a good aggregate outcome
- Combinatorial alternative spaces:
 - Suppose there are multiple interrelated issues that each need a decision
 - Exponentially sized alternative spaces
- Different models such as ranking webpages (pages “vote” on each other by linking)

CPS 590.4 Auctions & Combinatorial Auctions

Vincent Conitzer
conitzer@cs.duke.edu

A few different 1-item auction mechanisms

- English auction:

- Each bid must be higher than previous bid
- Last bidder wins, pays last bid

- Japanese auction:

- Price rises, bidders drop out when price is too high
- Last bidder wins at price of last dropout

- Dutch auction:

- Price drops until someone takes the item at that price

- Sealed-bid auctions (direct-revelation mechanisms):

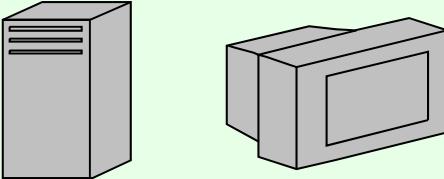
- Each bidder submits a bid in an envelope

- Auctioneer opens the envelopes, highest bid wins

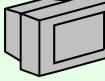
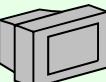
- First-price sealed-bid auction: winner pays own bid

- Second-price sealed bid (or Vickrey) auction: winner pays second-highest bid

Complementarity and substitutability

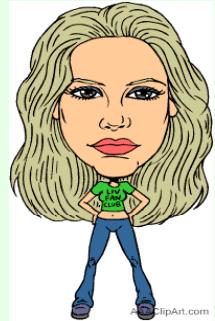
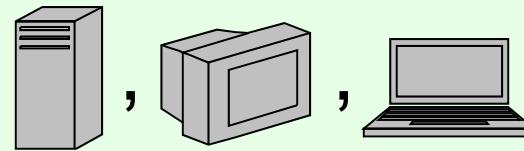
- How valuable one item is to a bidder may depend on whether the bidder possesses another item
- Items a and b are **complementary** if $v(\{a, b\}) > v(\{a\}) + v(\{b\})$
- E.g.  Items a and b are complementary if the bidder values both together more than the sum of their individual values.
- Items a and b are **substitutes** if $v(\{a, b\}) < v(\{a\}) + v(\{b\})$
- E.g.  Items a and b are substitutes if the bidder values either item more than the sum of their individual values.

Inefficiency of sequential auctions

- Suppose your valuation function is $v(\text{ }\text{ }) = \$200$, $v(\text{ }\text{ }) = \$100$, $v(\text{ }\text{ }\text{ }) = \500
- Now suppose that there are two (say, Vickrey) auctions, the first one for  and the second one for 
- What should you bid in the first auction (for )?
- If you bid \$200, you may lose to a bidder who bids \$250, only to find out that you could have won  for \$200
- If you bid anything higher, you may pay more than \$200, only to find out that  sells for \$1000
- Sequential (and parallel) auctions are inefficient

Combinatorial auctions

Simultaneously for sale:



bid 1

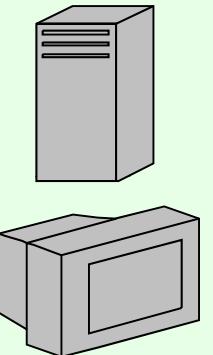
$$v(\text{server} \cup \text{monitor}) = \$500$$

bid 2

$$v(\text{laptop} \cup \text{monitor}) = \$700$$

bid 3

$$v(\text{laptop}) = \$300$$



used in truckload transportation, industrial procurement, radio spectrum allocation, ...

The winner determination problem (WDP)

- Choose a subset A (the accepted bids) of the bids B,
- to maximize $\sum_{b \in A} v_b$,
- under the constraint that every item occurs at most once in A
 - This is assuming **free disposal**, i.e., not everything needs to be allocated

WDP example

- Items A, B, C, D, E
- Bids:
 - ($\{A, C, D\}$, 7)
 - ($\{B, E\}$, 7)
 - ($\{C\}$, 3)
 - ($\{A, B, C, E\}$, 9)
 - ($\{D\}$, 4)
 - ($\{A, B, C\}$, 5)
 - ($\{B, D\}$, 5)
- *What's an optimal solution?*
- *How can we prove it is optimal?*

Price-based argument for optimality

- Items A, B, C, D, E
- Bids:
 - ($\{A, C, D\}$, 7)
 - ($\{B, E\}$, 7)
 - ($\{C\}$, 3)
 - ($\{A, B, C, E\}$, 9)
 - ($\{D\}$, 4)
 - ($\{A, B, C\}$, 5)
 - ($\{B, D\}$, 5)
- Suppose we create the following “prices” for the items:
 - $p(A) = 0, p(B) = 7, p(C) = 3, p(D) = 4, p(E) = 0$
 - Every bid bids at most the sum of the prices of its items, so we can’t expect to get more than 14.

Price-based argument does not always give matching upper bound

- Clearly can get at most 2
- Items A, B, C
- Bids:
- $(\{A, B\}, 2)$
- $(\{B, C\}, 2)$
- $(\{A, C\}, 2)$
- If we want to set prices that sum to 2, there must exist two items whose prices sum to < 2
- But then there is a bid on those two items of value 2
 - (Can set prices that sum to 3, so that's an upper bound)

Should not be surprising, since it's an NP-hard problem and we don't expect short proofs for negative answers to NP-hard problems (we don't expect $NP = coNP$)

An integer program formulation

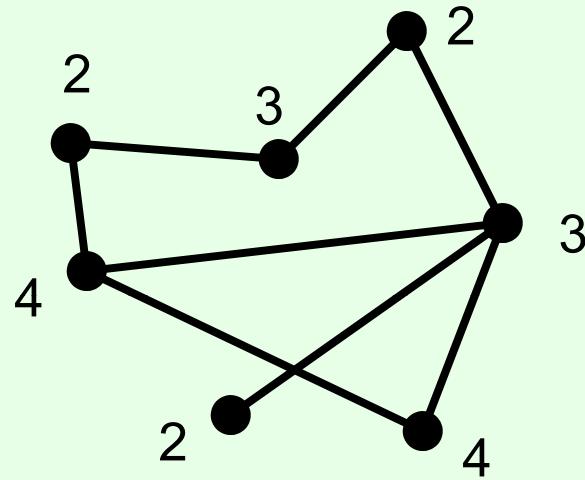
- x_b equals 1 if bid b is accepted, 0 if it is not
- maximize $\sum_b v_b x_b$
- subject to
 - for each item j , $\sum_{b: j \text{ in } b} x_b \leq 1$
- If each x_b can take any value in $[0, 1]$, we say that bids can be **partially accepted**
- In this case, this is a **linear** program that can be solved in polynomial time
- This requires that
 - each item can be divided into fractions
 - if a bidder gets a fraction f of **each** of the items in his bundle, then this is worth the same fraction f of his value v_b for the bundle

Price-based argument **does** always work for partially acceptable bids

- Items A, B, C
- Bids:
- $(\{A, B\}, 2)$
- $(\{B, C\}, 2)$
- $(\{A, C\}, 2)$
- Now can get 3, by accepting half of each bid
- Put a price of 1 on each item

General proof that with partially acceptable bids, prices always exist to give a matching upper bound is based on linear programming duality

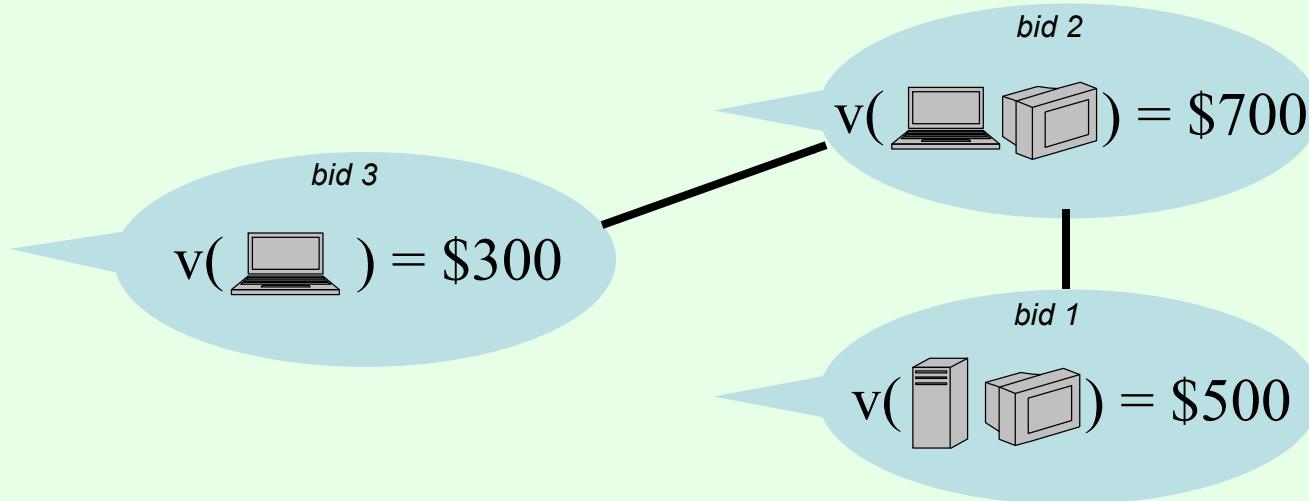
Weighted independent set



- Choose subset of the vertices with maximum total weight,
- Constraint: no two vertices can have an edge between them
- NP-hard (generalizes regular independent set)

The winner determination problem as a weighted independent set problem

- Each bid is a vertex
- Draw an edge between two vertices if they share an item



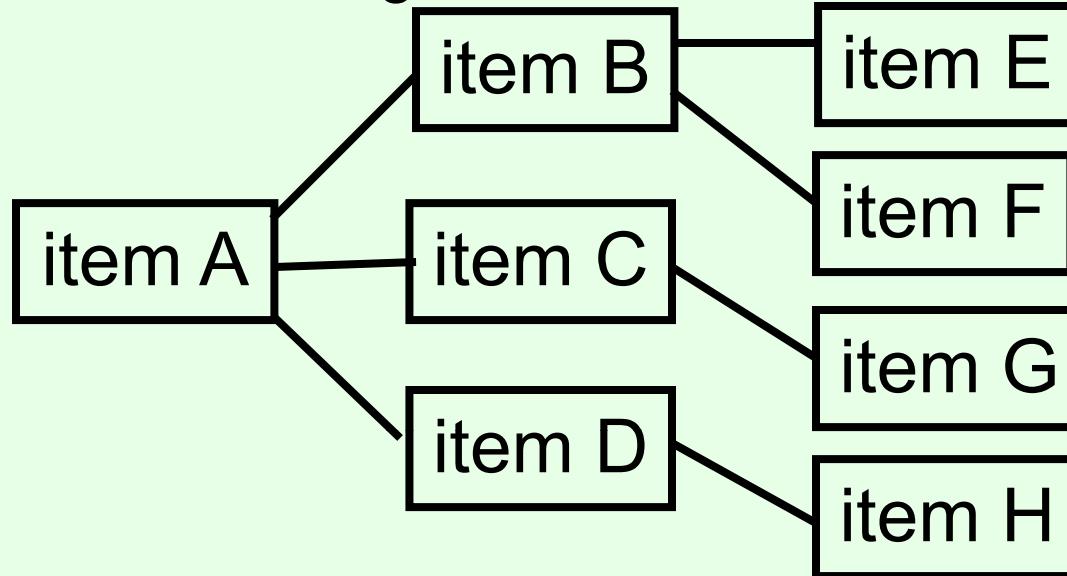
- Optimal allocation = maximum weight independent set
- Can model any weighted independent set instance as a CA winner determination problem (1 item per edge (or clique))
- Weighted independent set is NP-hard, even to solve approximately [Håstad 96] - hence, so is WDP
 - [Sandholm 02] noted that this inapproximability applies to the WDP

Dynamic programming approach to WDP [Rothkopf et al. 98]

- For every subset S of I , compute $w(S)$ = the maximum total value that can be obtained when allocating only items in S
- Then, $w(S) = \max \{ \max_i v_i(S), \max_{S' : S' \text{ is a subset of } S, \text{ and there exists a bid on } S'} w(S') + w(S \setminus S') \}$
- Requires exponential time

Bids on connected sets of items in a tree

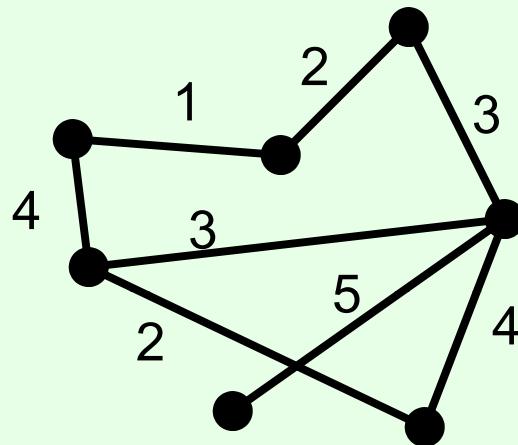
- Suppose items are organized in a tree



- Suppose each bid is on a connected set of items
 - E.g. {A, B, C, G}, but not {A, B, G}
- Then the WDP can be solved in polynomial time (using dynamic programming) [Sandholm & Suri 03]
- Tree does not need to be given: can be constructed from the bids in polynomial time if it exists [Conitzer, Derryberry, Sandholm 04]
- More generally, WDP can also be solved in polynomial time for graphs of bounded treewidth [Conitzer, Derryberry, Sandholm 04]
 - Even further generalization given by [Gottlob, Greco 07]

Maximum weighted matching

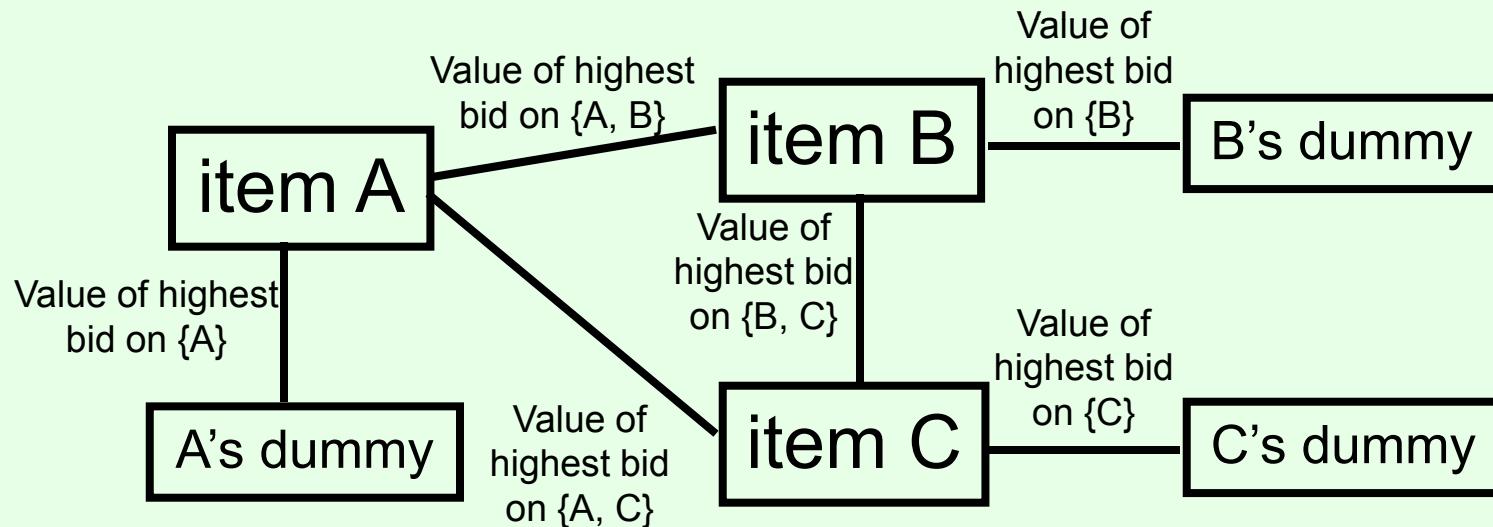
(not necessarily on bipartite graphs)



- Choose subset of the edges with maximum total weight,
- Constraint: no two edges can share a vertex
- Still solvable in polynomial time

Bids with few items [Rothkopf et al. 98]

- If each bid is on a bundle of at most **two** items,
- then the winner determination problem can be solved in polynomial time as a maximum weighted matching problem
 - 3-item example:



- If each bid is on a bundle of **three** items, then the winner determination problem is NP-hard again

Variants [Sandholm et al. 2002]: combinatorial reverse auction

- In a **combinatorial reverse auction (CRA)**, the auctioneer seeks to **buy** a set of items, and bidders have values for the different bundles that they may **sell** the auctioneer
 - minimize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_{b: j \in b} x_b \geq 1$

WDP example (as CRA)

- Items A, B, C, D, E
- Bids:
 - ($\{A, C, D\}$, 7)
 - ($\{B, E\}$, 7)
 - ($\{C\}$, 3)
 - ($\{A, B, C, E\}$, 9)
 - ($\{D\}$, 4)
 - ($\{A, B, C\}$, 5)
 - ($\{B, D\}$, 5)

Variants: multi-unit CAs/CRAs

- Multi-unit variants of CAs and CRAs: multiple units of the same item are for sale/to be bought, bidders can bid for multiple units
- Let q_{bj} be number of units of item j in bid b , q_j total number of units of j available/demanded
 - maximize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_b q_{bj} x_b \leq q_j$
- minimize $\sum_b v_b x_b$
 - subject to
 - for each item j , $\sum_b q_{bj} x_b \geq q_j$

Multi-unit WDP example (as CA/CRA)

- Items: 3A, 2B, 4C, 1D, 3E
- Bids:
 - ($\{1A, 1C, 1D\}$, 7)
 - ($\{2B, 1E\}$, 7)
 - ($\{2C\}$, 3)
 - ($\{2A, 1B, 2C, 2E\}$, 9)
 - ($\{2D\}$, 4)
 - ($\{3A, 1B, 2C\}$, 5)
 - ($\{2B, 2D\}$, 5)

Variants: (multi-unit) combinatorial exchanges

- Combinatorial exchange (CE): bidders can simultaneously be buyers and sellers
 - Example bid: “If I receive 3 units of A and -5 units of B (i.e., I have to give up 5 units of B), that is worth \$100 to me.”
- maximize $\sum_b v_b x_b$
- subject to
 - for each item j, $\sum_b q_{b,j} x_b \leq 0$

CE WDP example

- Bids:
- $(\{-1A, -1C, -1D\}, -7)$
- $(\{2B, 1E\}, 7)$
- $(\{2C\}, 3)$
- $(\{-2A, 1B, 2C, -2E\}, 9)$
- $(\{-2D\}, -4)$
- $(\{3A, -1B, -2C\}, 5)$
- $(\{-2B, 2D\}, 0)$

Variants: no free disposal

- Change all inequalities to equalities

(back to 1-unit CAs) Expressing valuation functions using bundle bids

- A bidder is **single-minded** if she only wants to win one particular bundle
 - Usually not the case
- But: one bidder may submit multiple bundle bids
- Consider again valuation function $v(\text{ }) = \$200$, $v(\text{ }) = \$100$, $v(\text{ } \text{ } \text{ }) = \500
- What bundle bids should one place?
- What about: $v(\text{ }) = \$300$, $v(\text{ }) = \$200$, $v(\text{ } \text{ } \text{ }) = \400 ?

Alternative approach: report entire valuation function

- I.e., every bidder i reports $v_i(S)$ for every subset S of I (the items)
- Winner determination problem:
- Allocate a subset S_i of I to each bidder i to maximize $\sum_i v_i(S_i)$ (under the constraint that for $i \neq j$, $S_i \cap S_j = \emptyset$)
 - This is assuming free disposal, i.e., not everything needs to be allocated

Exponentially many bundles

- In general, in a combinatorial auction with set of items I ($|I| = m$) for sale, a bidder could have a different valuation for every subset S of I
 - Implicit assumption: **no externalities** (bidder does not care what the other bidders win)
- Must a bidder communicate 2^m values?
 - Impractical
 - Also difficult for the bidder to evaluate **every** bundle
- Could require $v_i(\emptyset) = 0$
 - Does not help much
- Could require: if S is a superset of S' , $v(S) \geq v(S')$ (**free disposal**)
 - Does not help in terms of number of values

Bidding languages

- Bidding language = a language for expressing valuation functions
- A good bidding language allows bidders to concisely express natural valuation functions
- Example: the OR bidding language [Rothkopf et al. 98, DeMartini et al. 99]
- Bundle-value pairs are ORed together, auctioneer may accept any number of these pairs (assuming no overlap in items)
- E.g. $(\{a\}, 3) \text{ OR } (\{b, c\}, 4) \text{ OR } (\{c, d\}, 4)$ implies
 - A value of 3 for $\{a\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 7 for $\{a, b, c\}$
- Can we express the valuation function $v(\{a, b\}) = v(\{a\}) = v(\{b\}) = 1$ using the OR bidding language?
- OR language is good for expressing complementarity, bad for expressing substitutability

XORs

- If we use XOR instead of OR, that means that only **one** of the bundle-value pairs can be accepted
- Can express **any** valuation function (simply XOR together all bundles)
- E.g. $(\{a\}, 3) \text{ XOR } (\{b, c\}, 4) \text{ XOR } (\{c, d\}, 4)$ implies
 - A value of 3 for $\{a\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 4 for $\{a, b, c\}$
- Sometimes not very concise
- E.g. suppose that for any S , $v(S) = \sum_{s \in S} v(\{s\})$
 - How can this be expressed in the OR language?
 - What about the XOR language?
- Can also combine ORs and XORs to get benefits of both [Nisan 00, Sandholm 02]
- E.g. $((\{a\}, 3) \text{ XOR } (\{b, c\}, 4)) \text{ OR } (\{c, d\}, 4)$ implies
 - A value of 4 for $\{a, b, c\}$
 - A value of 4 for $\{b, c, d\}$
 - A value of 7 for $\{a, c, d\}$

WDP and bidding languages

- Single-minded bidders bid on only one bundle
 - Valuation is v for any subset including that bundle, 0 otherwise
- If we can solve the WDP for single-minded bidders, we can also solve it for the OR language
 - Simply pretend that each bundle-value pair comes from a different bidder
- We can even use the same algorithm when XORs are added, using the following trick:
 - For bundle-value pairs that are XORed together, add a dummy item to them [Fujishima et al 99, Nisan 00]
 - E.g. $(\{a\}, 3) \text{ XOR } (\{b, c\}, 4)$ becomes $(\{a, \text{dummy}_1\}, 3) \text{ OR } (\{b, c, \text{dummy}_1\}, 4)$
- So, we can focus on single-minded bids

CPS 590.4

Bayesian games and their use in auctions

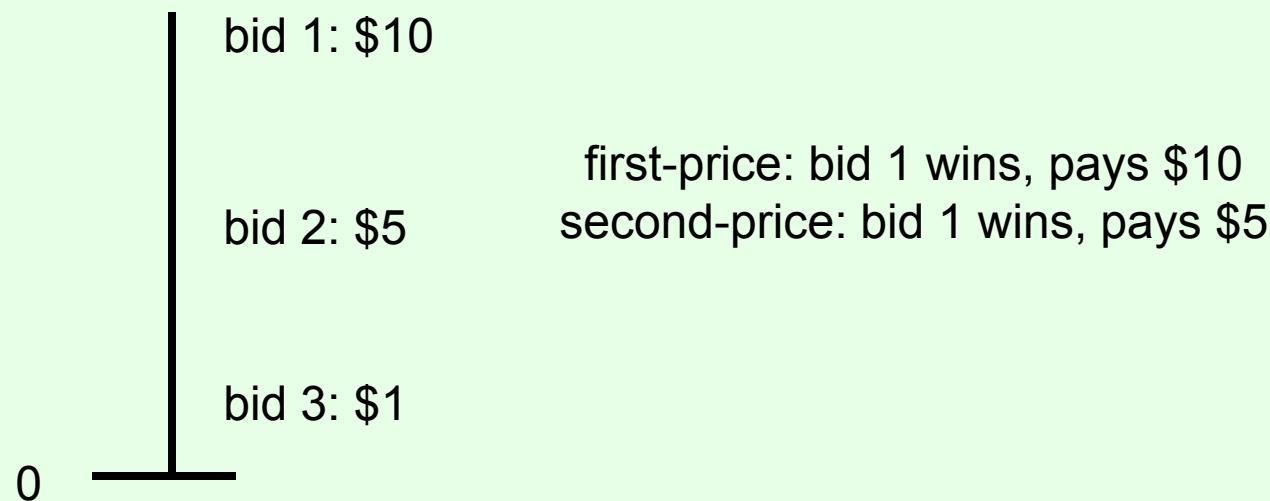
Vincent Conitzer
conitzer@cs.duke.edu

What is mechanism design?

- In mechanism design, we get to **design** the game (or mechanism)
 - e.g. the rules of the auction, marketplace, election, ...
- Goal is to obtain good outcomes when agents behave **strategically** (game-theoretically)
- Mechanism design often considered part of game theory
- 2007 Nobel Prize in Economics!
 - 2012 Prize also related
- Before we get to mechanism design, first we need to know how to **evaluate** mechanisms

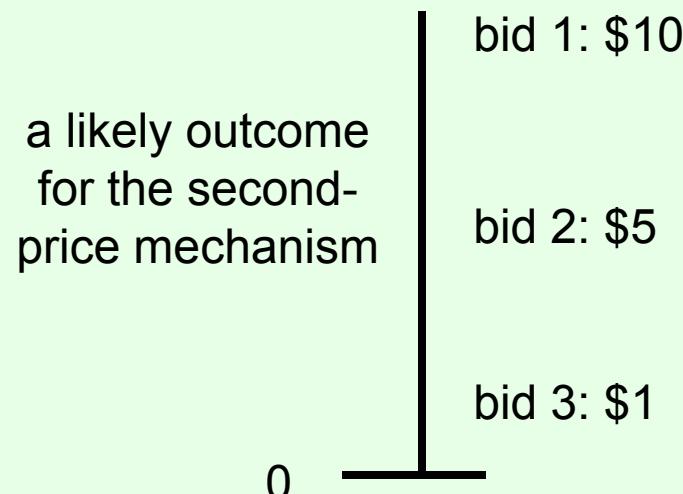
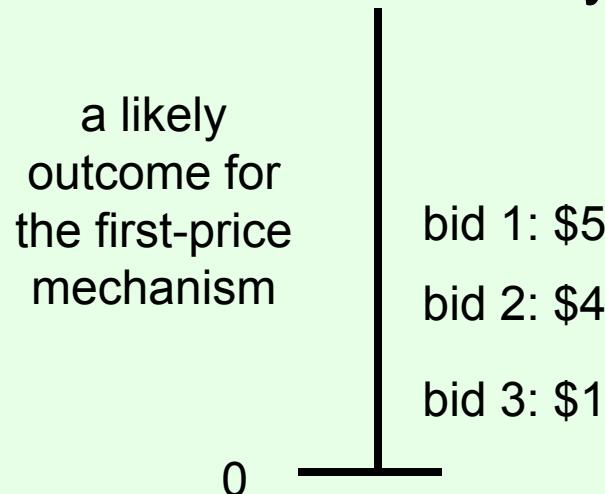
Example: (single-item) auctions

- **Sealed-bid** auction: every bidder submits bid in a sealed envelope
- **First-price** sealed-bid auction: highest bid wins, pays amount of own bid
- **Second-price** sealed-bid auction: highest bid wins, pays amount of second-highest bid



Which auction generates more revenue?

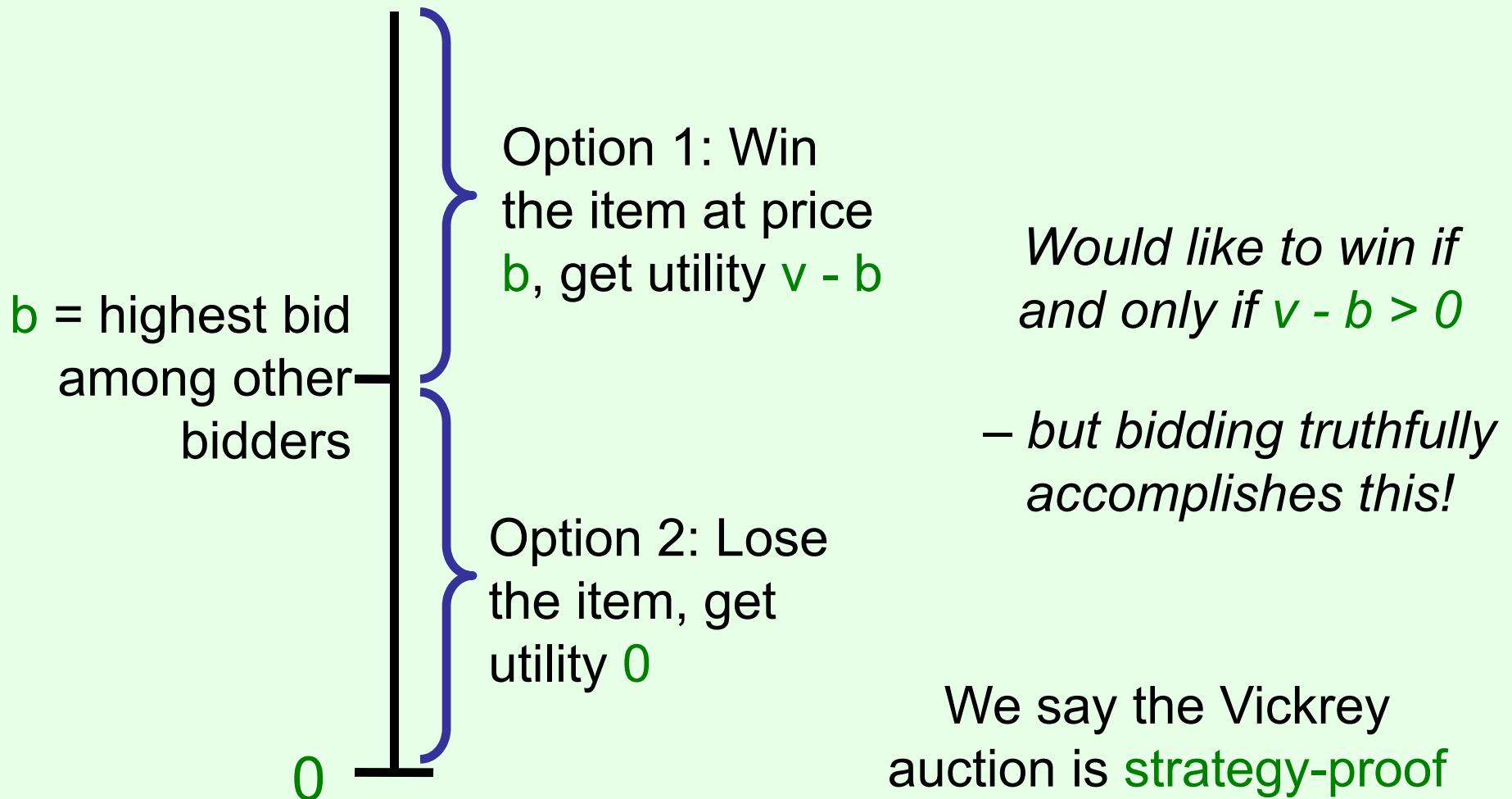
- Each bid depends on
 - bidder's **true valuation** for the item (utility = valuation - payment),
 - bidder's **beliefs** over what others will bid (\rightarrow game theory),
 - and... the **auction mechanism** used
- In a first-price auction, it does not make sense to bid your true valuation
 - Even if you win, your utility will be 0...
- In a second-price auction, (we will see next that) it always makes sense to bid your true valuation



Are there other auctions that perform better? How do we know when we have found the best one?

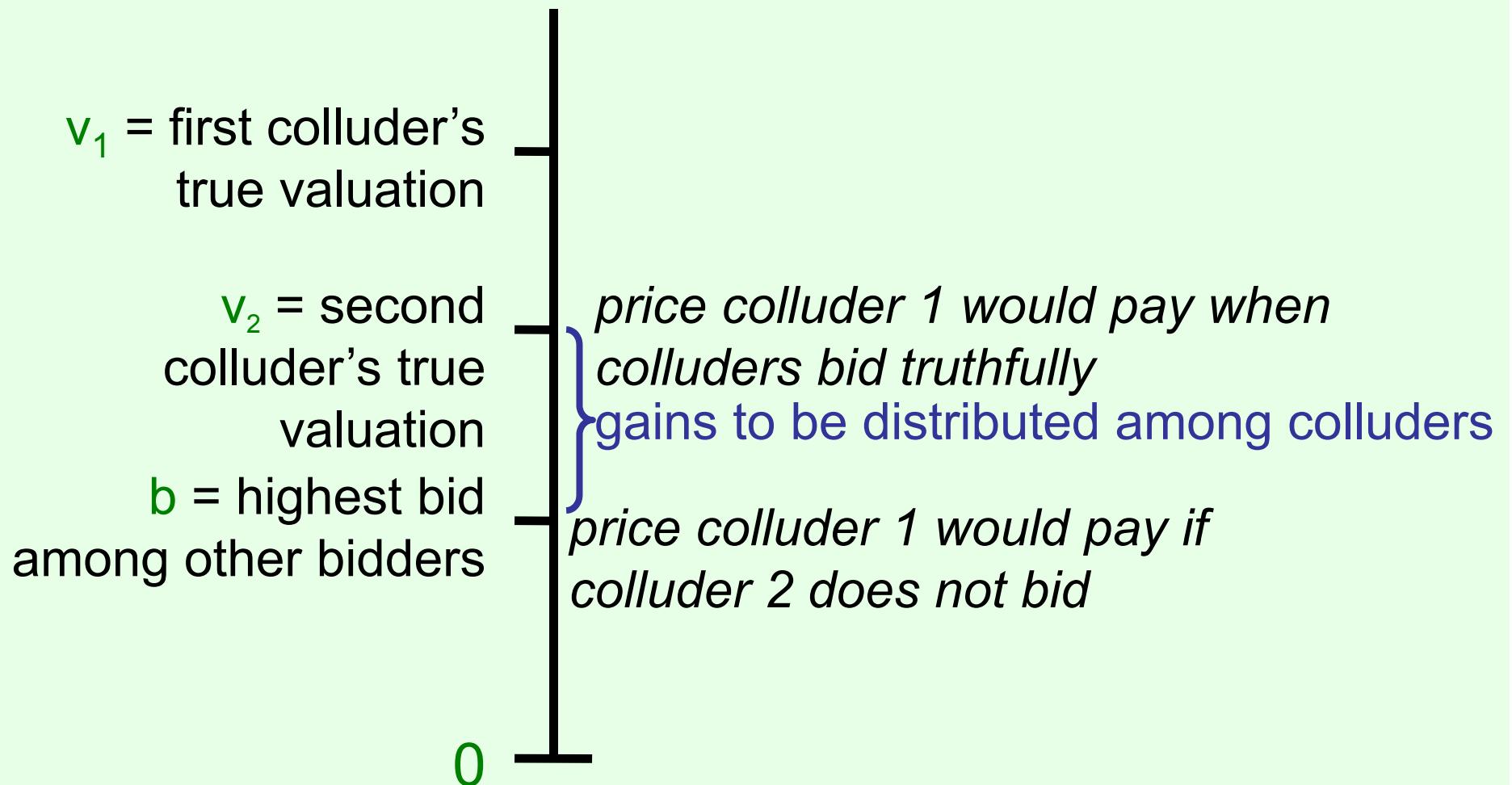
Bidding truthfully is optimal in the Vickrey auction!

- What should a bidder with value v bid?



Collusion in the Vickrey auction

- Example: two colluding bidders



Attempt #1 at using game theory to predict auction outcome

- First-price sealed-bid (or Dutch) auction
- Bidder 1 has valuation 4, bidder 2 has val. 2
- Discretized version, random tie-breaking

	0	1	2	3	4
0	2, 1	0, 1	0, 0	0, -1	0, -2
1	3, 0	1.5, .5	0, 0	0, -1	0, -2
2	2, 0	2, 0	1, 0	0, -1	0, -2
3	1, 0	1, 0	1, 0	.5, -.5	0, -2
4	0, 0	0, 0	0, 0	0, 0	0, -1

- What aspect(s) of auctions is this missing?

Bayesian games

- In a Bayesian game a player's utility depends on that player's type as well as the actions taken in the game
 - Notation: θ_i is player i's type, drawn according to some distribution from set of types Θ_i
 - Each player knows/learns its own type, not those of the others, before choosing action
 - Pure strategy s_i is a mapping from Θ_i to A_i (where A_i is i's set of actions)
 - In general players can also receive signals about other players' utilities; we will not go into this

		L	R
row player	U	4	6
type 1 (prob. 0.5)	D	2	4

		L	R
row player	U	2	4
type 2 (prob. 0.5)	D	4	2

		L	R
column player	U	4	6
type 1 (prob. 0.5)	D	4	6

		L	R
column player	U	2	2
type 2 (prob. 0.5)	D	4	2

Converting Bayesian games to normal form

		L	R
row player	U	4	6
	D	2	4

		L	R
row player	U	2	4
	D	4	2

		L	R
column player	U	4	6
	D	4	6

		L	R
column player	U	2	2
	D	4	2

		type 1: L	type 1: L	type 1: R	type 1: R
		type 2: L	type 2: R	type 2: L	type 2: R
type 1: U	U	3, 3	4, 3	4, 4	5, 4
type 2: U	U	4, 3.5	4, 3	4, 4.5	4, 4
type 1: U	D	2, 3.5	3, 3	3, 4.5	4, 4
type 2: D	D	3, 4	3, 3	3, 5	3, 4
type 1: D	U	3, 3	4, 4	4, 4	5, 5
type 2: D	U	4, 4	3, 3	3, 5	4, 4

exponential
blowup in size

Bayes-Nash equilibrium

- A profile of strategies is a **Bayes-Nash equilibrium** if it is a Nash equilibrium for the normal form of the game
 - Minor caveat: each type should have >0 probability
- Alternative definition: for every i , for every type θ_i , for every alternative action a_i , we must have:

$$\sum_{\theta_{-i}} P(\theta_{-i}) u_i(\theta_i, \sigma_i(\theta_i), \sigma_{-i}(\theta_{-i})) \geq \sum_{\theta_{-i}} P(\theta_{-i}) u_i(\theta_i, a_i, \sigma_{-i}(\theta_{-i}))$$

First-price sealed-bid auction BNE

- Suppose every bidder (independently) draws a valuation from $[0, 1]$
- What is a **Bayes-Nash equilibrium** for this?
- Say a bidder with value v_i bids $v_i(n-1)/n$
- Claim: this is an equilibrium!
- Proof: suppose all others use this strategy
- For a bid $b < (n-1)/n$, the probability of winning is $(bn/(n-1))^{n-1}$, so the expected value is $(v_i - b)(bn/(n-1))^{n-1}$
- Derivative w.r.t. b is $- (bn/(n-1))^{n-1} + (v_i - b)(n-1)b^{n-2}(n/(n-1))^{n-1}$ which should equal zero
- Implies $-b + (v_i - b)(n-1) = 0$, which solves to $b = v_i(n-1)/n$

Analyzing the expected revenue of the first-price and second-price (Vickrey) auctions

- **First-price auction:** probability of there not being a bid higher than b is $(bn/(n-1))^n$ (for $b < (n-1)/n$)
 - This is the cumulative density function of the highest bid
- Probability density function is the derivative, that is, it is $nb^{n-1}(n/(n-1))^n$
- Expected value of highest bid is
 $n(n/(n-1))^{n-1} \int_0^{(n-1)/n} b^n db = (n-1)/(n+1)$
- **Second-price auction:** probability of there not being two bids higher than b is $b^n + nb^{n-1}(1-b)$
 - This is the cumulative density function of the second-highest bid
- Probability density function is the derivative, that is, it is $nb^{n-1} + n(n-1)b^{n-2}(1-b) - nb^{n-1} = n(n-1)(b^{n-2} - b^{n-1})$
- Expected value is $(n-1) - n(n-1)/(n+1) = (n-1)/(n+1)$

Revenue equivalence theorem

- Suppose valuations for the single item are drawn i.i.d. from a continuous distribution over $[L, H]$ (with no “gaps”), and agents are risk-neutral
- Then, any two auction mechanisms that
 - in equilibrium always allocate the item to the bidder with the highest valuation, and
 - give an agent with valuation L an expected utility of 0, will lead to the same expected revenue for the auctioneer

(As an aside) what if bidders are not risk-neutral?

- Behavior in second-price/English/Japanese does not change, but behavior in first-price/Dutch does
- Risk averse: first price/Dutch will get higher expected revenue than second price/Japanese/English
- Risk seeking: second price/Japanese/English will get higher expected revenue than first price/Dutch

(As an aside) interdependent valuations

- E.g. bidding on drilling rights for an oil field
- Each bidder i has its own geologists who do tests, based on which the bidder assesses an expected value v_i of the field
- If you win, it is probably because the other bidders' geologists' tests turned out worse, and the oil field is not actually worth as much as you thought
 - The so-called **winner's curse**
- Hence, bidding v_i is no longer a dominant strategy in the second-price auction
- In English and Japanese auctions, you can update your valuation based on other agents' bids, so no longer equivalent to second-price
- In these settings, English (or Japanese) > second-price > first-price/Dutch in terms of revenue

Expected-revenue maximizing

(“optimal”) auctions [Myerson 81]

- Vickrey auction does not maximize expected revenue
 - E.g. with only one bidder, better off making a **take-it-or-leave-it offer** (or equivalently setting a **reserve price**)
- Suppose agent i draws valuation from probability density function f_i (cumulative density F_i)
- Bidder's **virtual valuation** $\psi(v_i) = v_i - (1 - F_i(v_i))/f_i(v_i)$
 - Under certain conditions, this is increasing; assume this
- The bidder with the highest virtual valuation (according to his reported valuation) wins (unless all virtual valuations are below 0, in which case nobody wins)
- Winner pays value of **lowest bid that would have made him win**
- E.g. if all bidders draw uniformly from $[0, 1]$, Myerson auction = second-price auction with reserve price $\frac{1}{2}$

Vickrey auction without a seller



$$v(\text{dog painting}) = 2$$

$$v(\text{dog painting}) = 4$$

$$v(\text{dog painting}) = 3$$



**pays 3
(money wasted!)**



Can we redistribute the payment?

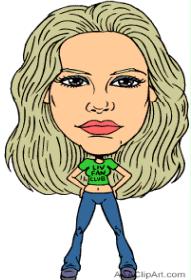
Idea: give everyone $1/n$ of the payment



$$v(\text{Panther}) = 2$$

$$v(\text{Panther}) = 4$$

$$v(\text{Panther}) = 3$$



receives 1



pays 3

receives 1



receives 1

not strategy-proof

Bidding higher can increase your redistribution payment

Incentive compatible redistribution

[Bailey 97, Porter et al. 04, Cavallo 06]

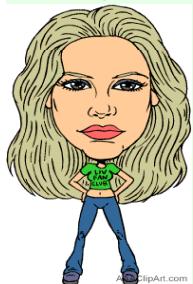
Idea: give everyone $1/n$ of second-highest **other** bid



$$v(\text{framed painting}) = 2$$

$$v(\text{framed painting}) = 4$$

$$v(\text{framed painting}) = 3$$



receives 1



pays 3
receives 2/3



receives 2/3

2/3 wasted (22%)

Strategy-proof

*Your redistribution does not depend on your bid;
incentives are the same as in Vickrey*

Bailey-Cavallo mechanism...

- Bids: $V_1 \geq V_2 \geq V_3 \geq \dots \geq V_n \geq 0$
- First run Vickrey auction
- Payment is V_2
- First two bidders receive V_3/n
- Remaining bidders receive V_2/n
- Total redistributed: $2V_3/n + (n-2)V_2/n$

$$R_1 = V_3/n$$

$$R_2 = V_3/n$$

$$R_3 = V_2/n$$

$$R_4 = V_2/n$$

...

$$R_{n-1} = V_2/n$$

$$R_n = V_2/n$$

Is this the best possible?

Another redistribution mechanism

- Bids: $V_1 \geq V_2 \geq V_3 \geq V_4 \geq \dots \geq V_n \geq 0$
- First run Vickrey
- Redistribution:
Receive $1/(n-2) * \text{second-highest other bid}$,
 $- 2/[(n-2)(n-3)] \text{ third-highest other bid}$
- Total redistributed:
 $V_2 - 6V_4 / [(n-2)(n-3)]$

$$R_1 = V_3 / (n-2) - 2 / [(n-2)(n-3)] V_4$$

$$R_2 = V_3 / (n-2) - 2 / [(n-2)(n-3)] V_4$$

$$R_3 = V_2 / (n-2) - 2 / [(n-2)(n-3)] V_4$$

$$R_4 = V_2 / (n-2) - 2 / [(n-2)(n-3)] V_3$$

...

$$R_{n-1} = V_2 / (n-2) - 2 / [(n-2)(n-3)] V_3$$

$$R_n = V_2 / (n-2) - 2 / [(n-2)(n-3)] V_3$$

Idea pursued further in Guo & Conitzer 07 / Moulin 07

CPS 590.4

Mechanism design

Vincent Conitzer
conitzer@cs.duke.edu

Mechanism design: setting

- The **center** has a set of outcomes O that she can choose from
 - Allocations of tasks/resources, joint plans, ...
- Each agent i draws a **type** θ_i from Θ_i
 - usually, but not necessarily, according to some probability distribution
- Each agent has a (commonly known) **valuation function** $v_i: \Theta_i \times O \rightarrow \mathbb{R}$
 - Note: depends on θ_i , which is **not** commonly known
- The center has some **objective function** $g: \Theta \times O \rightarrow \mathbb{R}$
 - $\Theta = \Theta_1 \times \dots \times \Theta_n$
 - E.g., efficiency ($\sum_i v_i(\theta_i, o)$)
 - May also depend on payments (more on those later)
 - The center does **not** know the types

What should the center do?

- She would like to know the agents' types to make the best decision
- Why not just ask them for their types?
- Problem: agents might **lie**
- E.g., an agent that slightly prefers outcome 1 may say that outcome 1 will give him a value of 1,000,000 and everything else will give him a value of 0, to force the decision in his favor
- But maybe, if the center is clever about choosing outcomes and/or requires the agents to make some **payments** depending on the types they report, the incentive to lie disappears...

Quasilinear utility functions

- For the purposes of mechanism design, we will assume that an agent's utility for
 - his type being θ_i ,
 - outcome o being chosen,
 - and having to pay π_i ,can be written as $v_i(\theta_i, o) - \pi_i$
- Such utility functions are called **quasilinear**
- Some of the results that we will see can be generalized beyond such utility functions, but we will not do so

Definition of a (direct-revelation) mechanism

- A **deterministic mechanism without payments** is a mapping $o: \Theta \rightarrow O$
- A **randomized mechanism without payments** is a mapping $o: \Theta \rightarrow \Delta(O)$
 - $\Delta(O)$ is the set of all probability distributions over O
- Mechanisms **with payments** additionally specify, for each agent i , a payment function $\pi_i: \Theta \rightarrow \mathbb{R}$ (specifying the payment that that agent must make)
- Each mechanism specifies a **Bayesian game** for the agents, where i 's set of actions $A_i = \Theta_i$
 - We would like agents to use the truth-telling strategy defined by $s(\theta_i) = \theta_i$

The Clarke (aka. VCG) mechanism [Clarke 71]

- The Clarke mechanism chooses some outcome o that maximizes $\sum_i v_i(\theta'_i, o)$
 - θ'_i = the type that i reports
- To determine the payment that agent j must make:
 - Pretend j does not exist, and choose o_{-j} that maximizes $\sum_{i \neq j} v_i(\theta'_i, o_{-j})$
 - j pays $\sum_{i \neq j} v_i(\theta'_i, o_{-j}) - \sum_{i \neq j} v_i(\theta'_i, o) = \sum_{i \neq j} (v_i(\theta'_i, o_{-j}) - v_i(\theta'_i, o))$
- We say that each agent pays the **externality** that she imposes on the other agents
- (VCG = Vickrey, Clarke, Groves)

Incentive compatibility

- Incentive compatibility (aka. truthfulness) = there is never an incentive to lie about one's type
- A mechanism is dominant-strategies incentive compatible (aka. strategy-proof) if for any i , for any type vector $\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n$, and for any alternative type θ'_i , we have

$$v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n) \geq$$

$$v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta'_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta'_i, \dots, \theta_n)$$

- A mechanism is Bayes-Nash equilibrium (BNE) incentive compatible if telling the truth is a BNE, that is, for any i , for any types θ_i, θ'_i ,

$$\sum_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)] \geq$$

$$\sum_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta'_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta'_i, \dots, \theta_n)]$$

The Clarke mechanism is strategy-proof

- Total utility for agent j is
$$v_j(\theta_j, o) - \sum_{i \neq j} (v_i(\theta'_i, o_{-j}) - v_i(\theta_i, o)) = \\ v_j(\theta_j, o) + \sum_{i \neq j} v_i(\theta'_i, o) - \sum_{i \neq j} v_i(\theta_i, o_{-j})$$
- But agent j cannot affect the choice of o_{-j}
- Hence, j can focus on maximizing $v_j(\theta_j, o) + \sum_{i \neq j} v_i(\theta'_i, o)$
- But mechanism chooses o to maximize $\sum_i v_i(\theta'_i, o)$
- Hence, if $\theta'_j = \theta_j$, j 's utility will be maximized!
- Extension of idea: add any term to agent j 's payment that does not depend on j 's reported type
- This is the family of Groves mechanisms [Groves 73]

Individual rationality

- A selfish center: “All agents must give me all their money.” – but the agents would simply not participate
 - If an agent would not participate, we say that the mechanism is not **individually rational**
- A mechanism is **ex-post** individually rational if for any i , for any type vector $\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n$, we have
$$v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n) \geq 0$$
- A mechanism is **ex-interim** individually rational if for any i , for any type θ_i ,
$$\sum_{\theta_{-i}} P(\theta_{-i}) [v_i(\theta_i, o(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)) - \pi_i(\theta_1, \theta_2, \dots, \theta_i, \dots, \theta_n)] \geq 0$$
 - i.e., an agent will want to participate given that he is uncertain about others’ types (not used as often)

Additional nice properties of the Clarke mechanism

- Ex-post individually rational (never hurts to participate), assuming:
 - An agent's presence never makes it impossible to choose an outcome that could have been chosen if the agent had not been present, and
 - No agent ever has a negative value for an outcome that would be selected if that agent were not present
- **Weakly budget balanced** - that is, the sum of the payments is always nonnegative - assuming:
 - If an agent leaves, this never makes the combined welfare of the other agents (not considering payments) smaller

Generalized Vickrey Auction (GVA)

(= VCG applied to combinatorial auctions)

- Example:
 - Bidder 1 bids $(\{A, B\}, 5)$
 - Bidder 2 bids $(\{B, C\}, 7)$
 - Bidder 3 bids $(\{C\}, 3)$
- Bidders 1 and 3 win, total value is 8
- Without bidder 1, bidder 2 would have won
 - Bidder 1 pays $7 - 3 = 4$
- Without bidder 3, bidder 2 would have won
 - Bidder 3 pays $7 - 5 = 2$
- Strategy-proof, ex-post IR, weakly budget balanced
- Vulnerable to **collusion** (more so than 1-item Vickrey auction)
 - E.g., add two bidders $(\{B\}, 100), (\{A, C\}, 100)$
 - What happens?
 - More on collusion in GVA in [\[Ausubel & Milgrom 06, Conitzer & Sandholm 06\]](#)

Clarke mechanism is not perfect

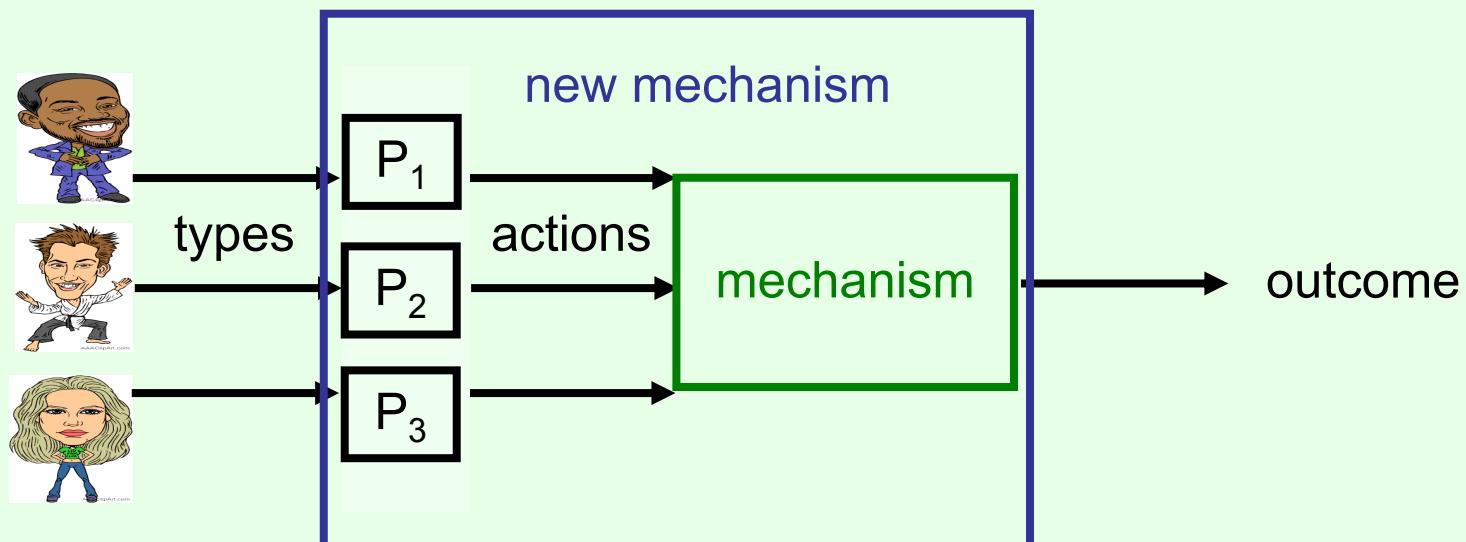
- Requires payments + quasilinear utility functions
- In general money needs to flow away from the system
 - Strong budget balance = payments sum to 0
 - In general, this is impossible to obtain in addition to the other nice properties [Green & Laffont 77]
- Vulnerable to collusion
 - E.g., suppose two agents both declare a ridiculously large value (say, \$1,000,000) for some outcome, and 0 for everything else. What will happen?
- Maximizes sum of agents' utilities (if we do not count payments), but sometimes the center is not interested in this
 - E.g., sometimes the center wants to maximize revenue

Why restrict attention to truthful direct-revelation mechanisms?

- Bob has an incredibly complicated mechanism in which agents do not report types, but do all sorts of other strange things
- E.g.: Bob: “In my mechanism, first agents 1 and 2 play a round of rock-paper-scissors. If agent 1 wins, she gets to choose the outcome. Otherwise, agents 2, 3 and 4 vote over the other outcomes using the Borda rule. If there is a tie, everyone pays \$100, and...”
- Bob: “The *equilibria* of my mechanism produce better results than any truthful direct revelation mechanism.”
- Could Bob be right?

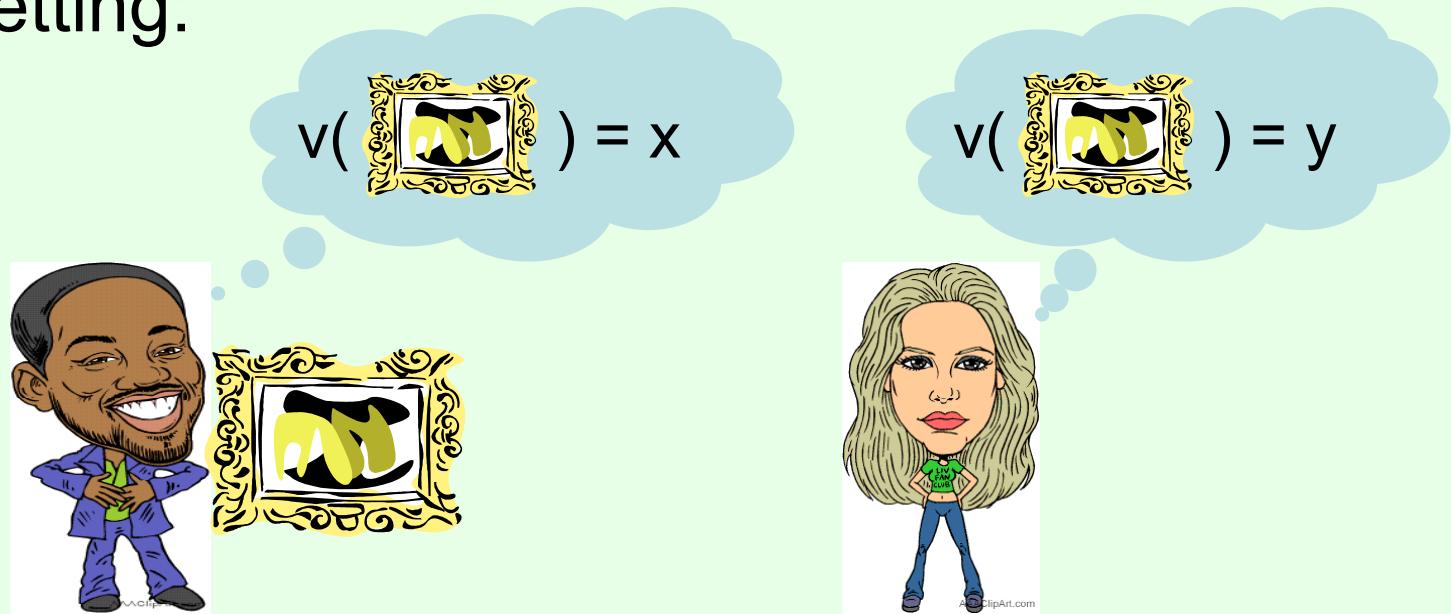
The revelation principle

- For any (complex, strange) mechanism that produces certain outcomes under strategic behavior (dominant strategies, BNE)...
- ... there exists a (dominant-strategies, BNE) incentive compatible direct revelation mechanism that produces the same outcomes!



Myerson-Satterthwaite impossibility [1983]

- Simple setting:



- We would like a mechanism that:
 - is efficient (trade if and only if $y > x$),
 - is budget-balanced (seller receives what buyer pays),
 - is BNE incentive compatible, and
 - is ex-interim individually rational
- This is impossible!

A few computational issues in mechanism design

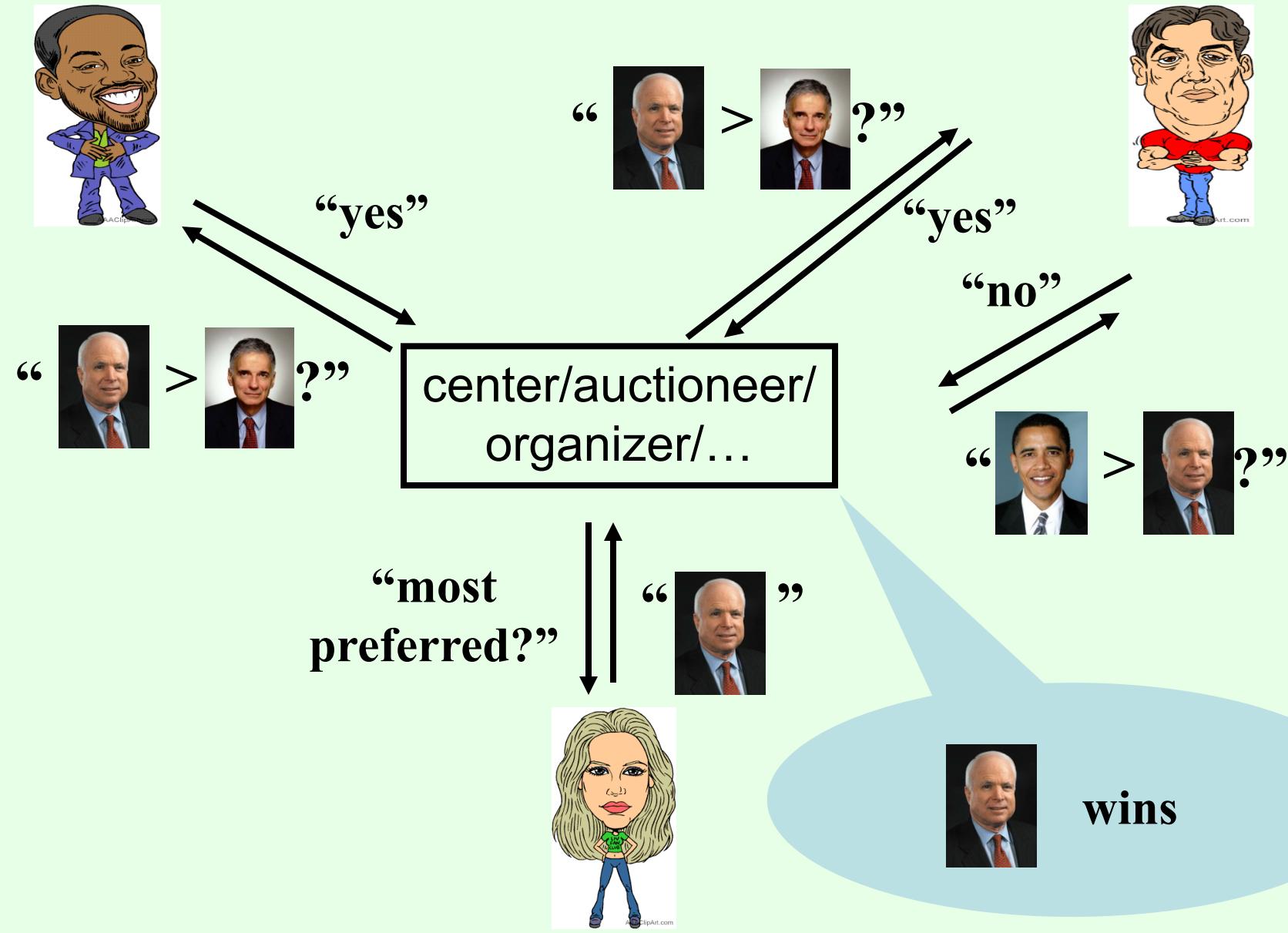
- **Algorithmic** mechanism design
 - Sometimes standard mechanisms are too hard to execute computationally (e.g., Clarke requires computing optimal outcome)
 - Try to find mechanisms that are easy to execute computationally (and nice in other ways), together with algorithms for executing them
- **Automated** mechanism design
 - Given the specific setting (agents, outcomes, types, priors over types, ...) and the objective, have a **computer** solve for the best mechanism for this particular setting
- When agents have **computational limitations**, they will not necessarily play in a game-theoretically optimal way
 - Revelation principle can collapse; need to look at nontruthful mechanisms
- Many other things (computing the outcomes in a **distributed** manner; what if the agents come in over time (**online** setting); ...)

CPS 590.4

Preference elicitation/ iterative mechanisms

Vincent Conitzer
conitzer@cs.duke.edu

Preference elicitation (elections)



Preference elicitation (auction)



“30”
“v({A})?”

center/auctioneer/
organizer/...

“v({A,B,C})
< 70?”

“yes” “40”
“v({B, C})?”



“What would you buy
if the price for A is 30,
the price for B is 20,
the price for C is 20?”



“nothing”



gets {A},
pays 30



gets {B,C},
pays 40

Unnecessary communication

- We have seen that mechanisms often force agents to communicate large amounts of information
 - E.g., in combinatorial auctions, should in principle communicate a value for every single bundle!
- Much of this information will be **irrelevant**, e.g.:
 - Suppose each item has already received a bid $> \$1$
 - Bidder 1 values the **grand bundle** of all items at $v_1(I) = \$1$
 - To find the optimal allocation, we need not know anything more about 1's valuation function (assuming free disposal)
 - We may still need more detail on 1's valuation function to compute Clarke payments...
 - ... but not if each item has received **two** bids $> \$1$
- Can we spare bidder 1 the burden of communicating (and figuring out) her whole valuation function?

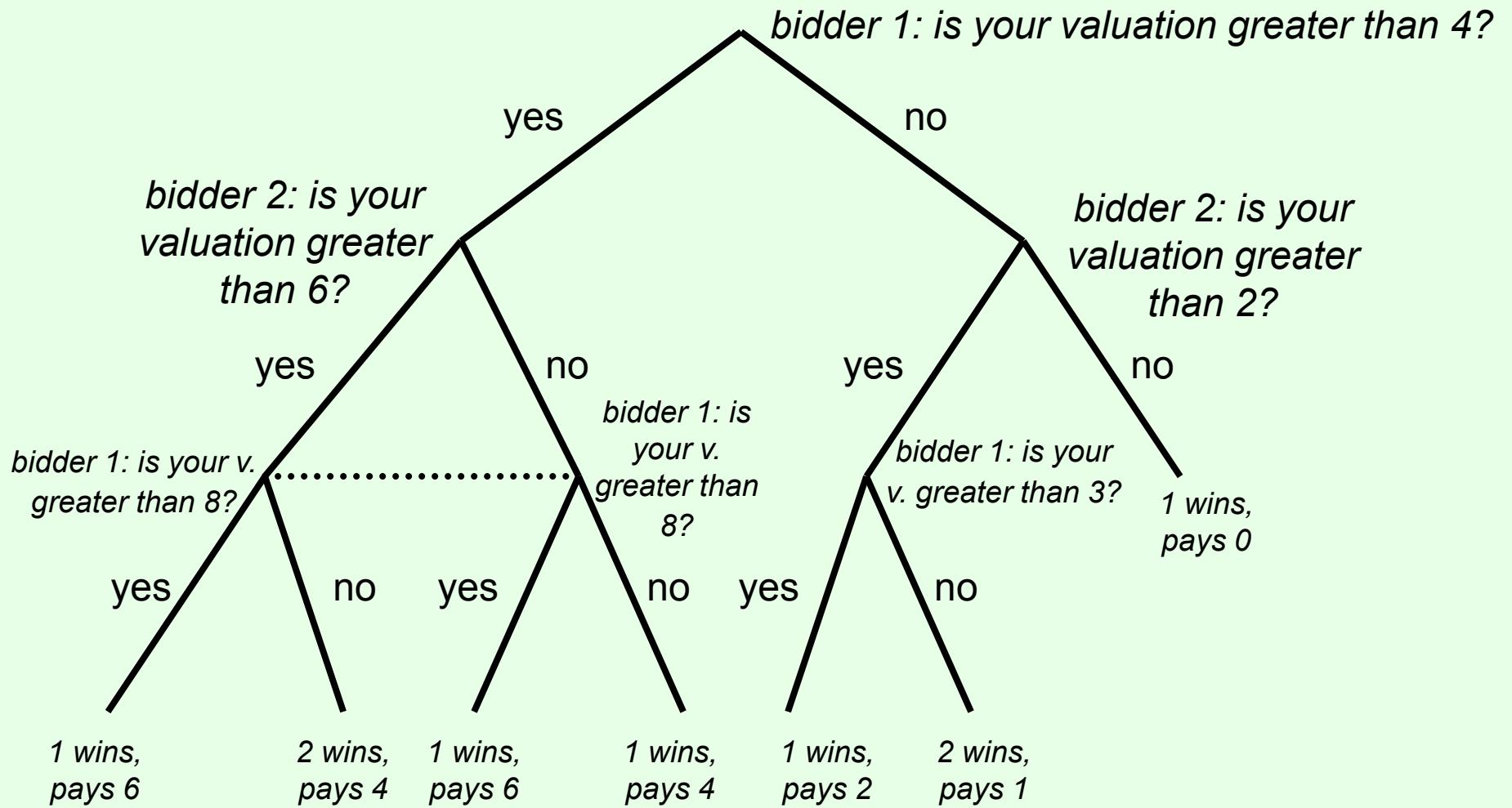
Single-stage mechanisms

- If all agents must report their valuations (types) at the same time (e.g., sealed-bid), then almost no communication can be saved
 - E.g., if we do not know that other bidders have already placed high bids on items, we may need to know more about bidder 1's valuation function
 - Can only save communication of information that is irrelevant **regardless** of what other agents report
 - E.g. if a bidder's valuation is below the reserve price, it does not matter exactly where below the reserve price it is
 - E.g. a voter's second-highest candidate under plurality rule
- Could still try to **design the mechanism** so that most information is (unconditionally) irrelevant
 - E.g. [Hyafil & Boutilier IJCAI 07]

Multistage mechanisms

- In a multistage (or iterative) mechanism,
 - bidders communicate something,
 - then find out something about what others communicated,
 - then communicate again, etc.
- After enough information has been communicated, the mechanism declares an outcome
- What multistage mechanisms have we seen already?

A (strange) example multistage auction



- Can choose to hide information from agents, but only insofar as it is not implied by queries we ask of them

Converting single-stage to multistage

- One possibility: start with a single-stage mechanism (mapping σ from $\Theta_1 \times \Theta_2 \times \dots \times \Theta_n$ to O)
- Center asks the agents **queries** about their types
 - E.g., “Is your valuation greater than v ?”
 - May or may not (explicitly) reveal results of queries to others
- Until center knows enough about $\theta_1, \theta_2, \dots, \theta_n$ to determine $\sigma(\theta_1, \theta_2, \dots, \theta_n)$
- The center’s strategy for asking queries is an **elicitation algorithm** for computing σ
- E.g., Japanese auction is an elicitation algorithm for the second-price auction

Elicitation algorithms

- Suppose agents always answer truthfully
- Design elicitation algorithm to minimize queries for given rule
- What is a good elicitation algorithm for STV?
- What about Bucklin?

An elicitation algorithm for the Bucklin voting rule based on binary search

[Conitzer & Sandholm 05]

- Alternatives: A B C D E F G H



- Top 4? $\{A \ B \ C \ D\}$ $\{A \ B \ F \ G\}$ $\{A \ C \ E \ H\}$
- Top 2? $\{A \ D\}$ $\{B \ F\}$ $\{C \ H\}$
- Top 3? $\{A \ C \ D\}$ $\{B \ F \ G\}$ $\{C \ E \ H\}$

Total communication is $nm + nm/2 + nm/4 + \dots \leq 2nm$ bits
(n number of voters, m number of candidates)

Funky strategic phenomena in multistage mechanisms

- Suppose we sell two items A and B in parallel English auctions to bidders 1 and 2
 - Minimum bid increment of 1
- No complementarity/substitutability
- $v_1(A) = 30$, $v_1(B) = 20$, $v_2(A) = 20$, $v_2(B) = 30$, all of this is **common knowledge**
- 1's strategy: "I will bid 1 on B and 0 on A, unless 2 starts bidding on B, in which case I will bid up to my true valuations for both."
- 2's strategy: "I will bid 1 on A and 0 on B, unless 1 starts bidding on A, in which case I will bid up to my true valuations for both."
- This is an equilibrium!
 - Inefficient allocation
 - Self-enforcing collusion
 - Bidding truthfully (up to true valuation) is **not** a dominant strategy

Ex-post equilibrium

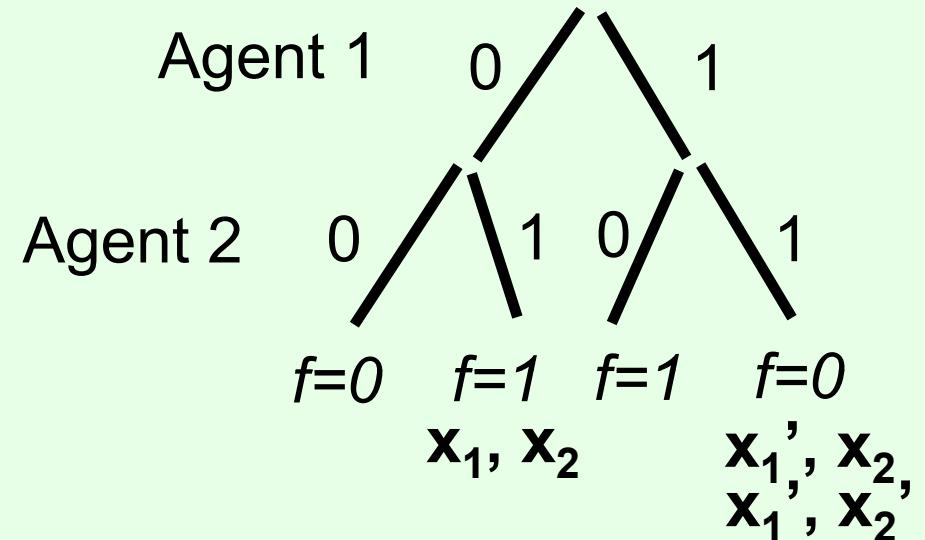
- In a Bayesian game, a profile of strategies is an **ex-post equilibrium** if for each agent, following the strategy is optimal for **every** vector of types (given the others' strategies)
 - That is, even if you are told what everyone's type was after the fact, you never regret what you did
 - Stronger than Bayes-Nash equilibrium
 - Weaker than dominant-strategies equilibrium
 - Although, single-stage mechanisms are ex-post incentive compatible if and only if they are dominant-strategies incentive compatible
- If a single-stage mechanism is dominant-strategies incentive-compatible, then **any** elicitation protocol for it (any corresponding multistage mechanism) will be ex-post incentive compatible
- E.g., if we elicit enough information to determine the Clarke payments, telling the truth will be an ex-post equilibrium (but not dominant strategies)

Lower bounds on communication

- Communication complexity theory can be used to show lower bounds
 - “Any elicitation algorithm for rule r requires communication of at least N bits (in the worst case)”
- Voting [Conitzer & Sandholm 05]
 - Bucklin requires at least on the order of nm bits
 - STV requires at least on the order of $n \log m$ bits
 - Natural algorithm uses on the order of $n(\log m)^2$ bits
- Combinatorial auction winner determination requires exponentially many bits [Nisan & Segal 06]
 - ... unless only a limited set of valuation functions is allowed

How do we know that we have found the best elicitation protocol for a mechanism?

- Communication complexity theory: agent i holds input x_i , agents must communicate enough information to compute some $f(x_1, x_2, \dots, x_n)$
- Consider the tree of all possible communications:
- Every input vector goes to some leaf
- If x_1, \dots, x_n goes to same leaf as x'_1, \dots, x'_n then so must any mix of them (e.g., $x_1, x'_2, x_3, \dots, x'_n$)
- Only possible if f is same in all 2^n cases
- Suppose we have a fooling set of t input vectors that all give the same function value f_0 , but for any two of them, there is a mix that gives a different value
- Then all vectors must go to different leaves \Rightarrow tree depth must be $\geq \log(t)$
- Also lower bound on nondeterministic communication complexity
 - With false positives or negatives allowed, depending on f_0



Example on board: finding which valuation is higher (or tie)

Combinatorial auction WDP requires exponential communication [Nisan & Segal JET 06]

- ... even with two bidders!
- Let us construct a fooling set
- Consider valuation functions with
 - $v(S) = 0$ for $|S| < m/2$
 - $v(S) = 1$ for $|S| > m/2$
 - $v(S) = 0$ or 1 for $|S| = m/2$
- If m is even, there are $2^{m \choose m/2}$ such valuation functions (doubly exponential)
- In the fooling set, bidder 1 will have one such valuation function, and bidder 2 will have the **dual** such valuation function, that is, $v_2(S) = 1 - v_1(I \setminus S)$
- Best allocation gives total value of 1
- However, now suppose we take distinct $(v_1, v_2), (v'_1, v'_2)$
- WLOG there must be some set S such that $v_1(S) = 1$ and $v'_1(S) = 0$ (hence $v'_2(I \setminus S) = 1$)
- So on (v_1, v'_2) we can get a total allocation value of 2!

iBundle: an ascending CA [Parkes & Ungar 00]

- Each round, each bidder i faces separate price $p_i(S)$ for each bundle S
 - Note: different bidders may face different prices for the **same** bundle
 - Prices start at 0
- A bidder (is assumed to) bid $p_i(S)$ on the bundle(s) S that maximize(s) her utility given the current prices, i.e., that maximize(s) $v_i(S) - p_i(S)$ (**straightforward bidding**)
 - Bidder drops out if all bundles would give negative utility
- Winner determination problem is solved with these bids
- If some (active) bidder i did not win anything, that bidder's prices are increased by ϵ on each of the bundles that she bid on (and supersets thereof), and we go to the next round
- Otherwise, we terminate with this allocation & these prices

Restricted valuations

- For (e.g.) combinatorial auctions, if we know that agents' valuation functions lie in a **restricted class** of functions, then they may be easy to elicit
- E.g. if we **know** that an agent's valuation function is an OR of bundles of size at most 2, then all we need to ask a bidder for is his value of each bundle of size at most 2, to know the **entire** function
 - $O(m^2)$ queries
 - So-called **value queries**
- Which **classes of valuations** can we elicit using only polynomially many queries?
 - ... and what types of queries do we need?
- Closely related to **query learning** in machine learning

Restricted valuations...

- Various restricted classes can be elicited using polynomially many value queries
 - Read-once & toolbox valuations [Zinkevich, Blum, Sandholm EC 03]
 - Valuations with limited item interdependency [Conitzer, Sandholm, Santi AAAI 05]
- Other classes inherently require other types of query
- E.g., demand query: “Which bundle would you buy given prices $p(S)$ on bundles?”
 - Could also just have prices on items
 - Compare iBundle ascending CA
- A value query can be simulated using polynomially many demand queries (even just with item prices), but not vice versa [Blumrosen & Nisan EC 05]
- Using (bundle-price) demand queries, XOR valuations can be elicited using $O(m^2 \# \text{terms})$ queries [Lahaie & Parkes EC 04]
- ... but if only item-price demand queries (and value queries) are allowed, exponentially many queries are required [Blum et al. JMLR 04]

CPS 590.4

Repeated games

Vincent Conitzer
conitzer@cs.duke.edu

Repeated games

- In a (typical) repeated game,
 - players play a normal-form game (aka. the **stage game**),
 - then they see what happened (and get the utilities),
 - then they play again,
 - etc.
- Can be repeated finitely or infinitely many times
- Really, an extensive form game
 - Would like to find subgame-perfect equilibria
- One subgame-perfect equilibrium: keep repeating some Nash equilibrium of the stage game
- But are there other equilibria?

Finitely repeated Prisoner's Dilemma

- Two players play the Prisoner's Dilemma k times

	cooperate	defect
cooperate	2, 2	0, 3
defect	3, 0	1, 1

- In the last round, it is dominant to defect
- Hence, in the second-to-last round, there is no way to influence what will happen
- So, it is optimal to defect in this round as well
- Etc.
- So the only equilibrium is to always defect

Modified Prisoner's Dilemma

- Suppose the following game is played twice

		cooperate	defect ₁	defect ₂	
		cooperate	5, 5	0, 6	0, 6
		defect ₁	6, 0	4, 4	1, 1
		defect ₂	6, 0	1, 1	2, 2

- Consider the following strategy:
 - In the first round, cooperate;
 - In the second round, if someone defected in the first round, play defect₂; otherwise, play defect₁,
- If both players play this, is that a subgame perfect equilibrium?

Another modified Prisoner's Dilemma

- Suppose the following game is played twice

		cooperate	defect	crazy	
		cooperate	5, 5	0, 6	1, 0
		defect	6, 0	4, 4	1, 0
		crazy	0, 1	0, 1	0, 0

- What are the subgame perfect equilibria?
- Consider the following strategy:
 - In the first round, cooperate;
 - In the second round, if someone played defect or crazy in the first round, play crazy; otherwise, play defect
- Is this a Nash equilibrium (**not** subgame perfect)?

Infinitely repeated games

- First problem: are we just going to add up the utilities over infinitely many rounds?
 - Everyone gets infinity!
- (Limit of) **average** payoff: $\lim_{n \rightarrow \infty} \sum_{1 \leq t \leq n} u(t)/n$
 - Limit may not exist...
- **Discounted** payoff: $\sum_t \delta^t u(t)$ for some $\delta < 1$

Infinitely repeated Prisoner's Dilemma

	cooperate	defect
cooperate	2, 2	0, 3
defect	3, 0	1, 1

- **Tit-for-tat** strategy:
 - Cooperate the first round,
 - In every later round, do the same thing as the other player did in the **previous** round
- Is both players playing this a Nash/subgame-perfect equilibrium? Does it depend on δ ?
- **Trigger** strategy:
 - Cooperate as long as everyone cooperates
 - Once a player defects, defect **forever**
- Is both players playing this a subgame-perfect equilibrium?
- What about one player playing tit-for-tat and the other playing trigger?

Folk theorem(s)

- Can we somehow characterize the equilibria of infinitely repeated games?
 - Subgame perfect or not?
 - Averaged utilities or discounted?
- Easiest case: averaged utilities, no subgame perfection
- We will characterize what (averaged) utilities (u_1, u_2, \dots, u_n) the agents can get in equilibrium
- The utilities must be **feasible**: there must be outcomes of the game such that the agents, on average, get these utilities
- They must also be **enforceable**: deviation should lead to punishment that outweighs the benefits of deviation
- **Folk theorem**: a utility vector can be realized by some Nash equilibrium if and only if it is both feasible and enforceable

Feasibility

2, 2	0, 3
3, 0	1, 1

- The utility vector (2, 2) is feasible because it is one of the outcomes of the game
- The utility vector (1, 2.5) is also feasible, because the agents could **alternate** between (2, 2) and (0, 3)
- What about (.5, 2.75)?
- What about (3, 0.1)?
- In general, **convex combinations** of the outcomes of the game are feasible
 - $p_1a_1 + p_2a_2 + \dots + p_na_n$ is a convex combination of the a_i if the p_i sum to 1 and are nonnegative

Enforceability

2, 2	0, 3
3, 0	1, 1

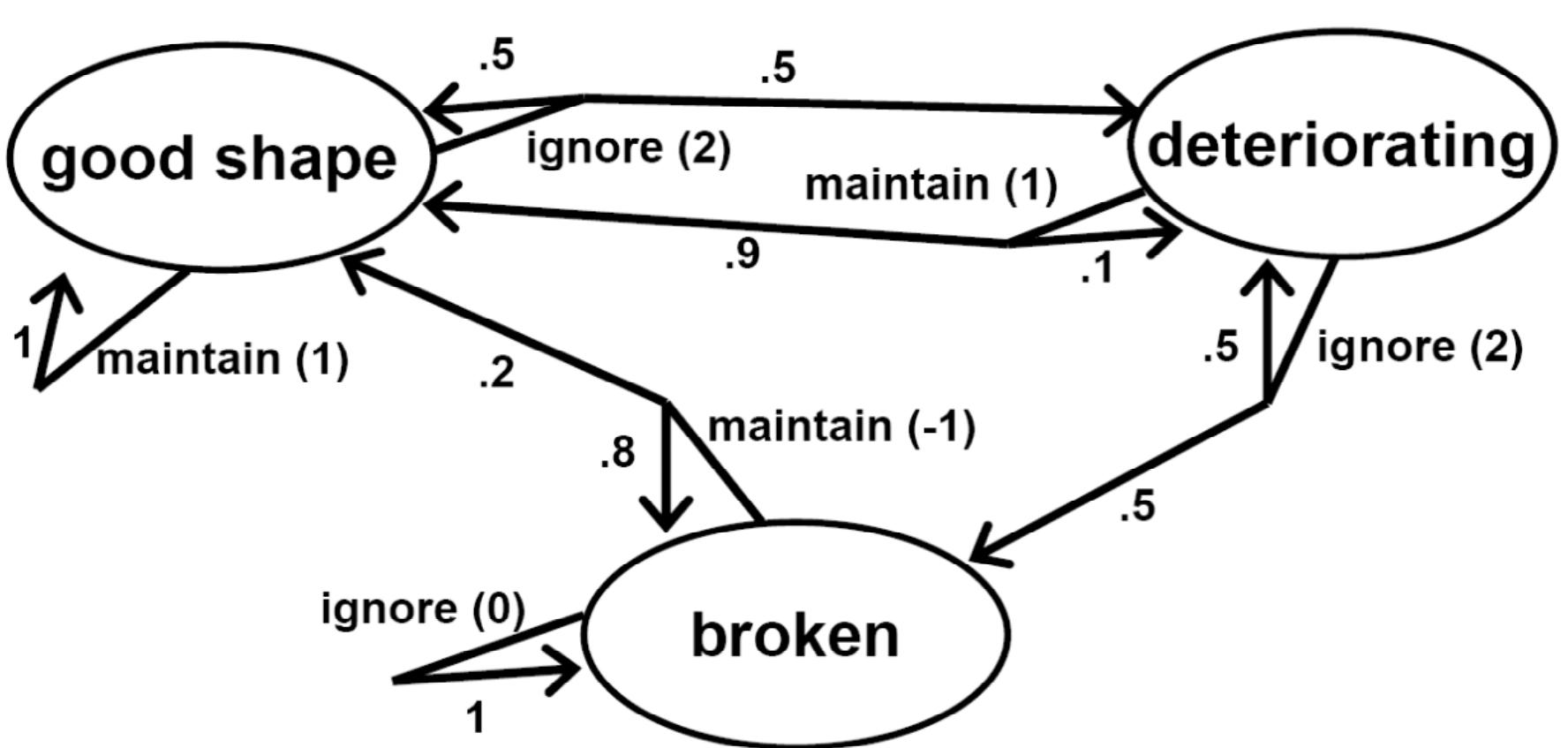
- A utility for an agent is **not enforceable** if the agent can guarantee herself a higher utility
- E.g. a utility of .5 for player 1 is not enforceable, because she can guarantee herself a utility of 1 by defecting
- A utility of 1.2 for player 1 is enforceable, because player 2 can guarantee player 1 a utility of at most 1 by defecting
- What is the relationship to minimax strategies & values?

Computing a Nash equilibrium in a 2-player repeated game using folk theorem

- Average payoff, no subgame perfection
- Can be done in polynomial time:
 - Compute minimum enforceable utility for each agent
 - I.e., compute maxmin values & strategies
 - Find a feasible point where both players receive at least this utility
 - E.g., both players playing their maxmin strategies
 - Players play feasible point (by rotating through the outcomes), unless the other deviates, in which case they punish the other player by playing minmax strategy forever
 - Minmax strategy easy to compute
- A more complicated (and earlier) algorithm by [Littman & Stone \[04\]](#) computes a “nicer” and subgame-perfect equilibrium

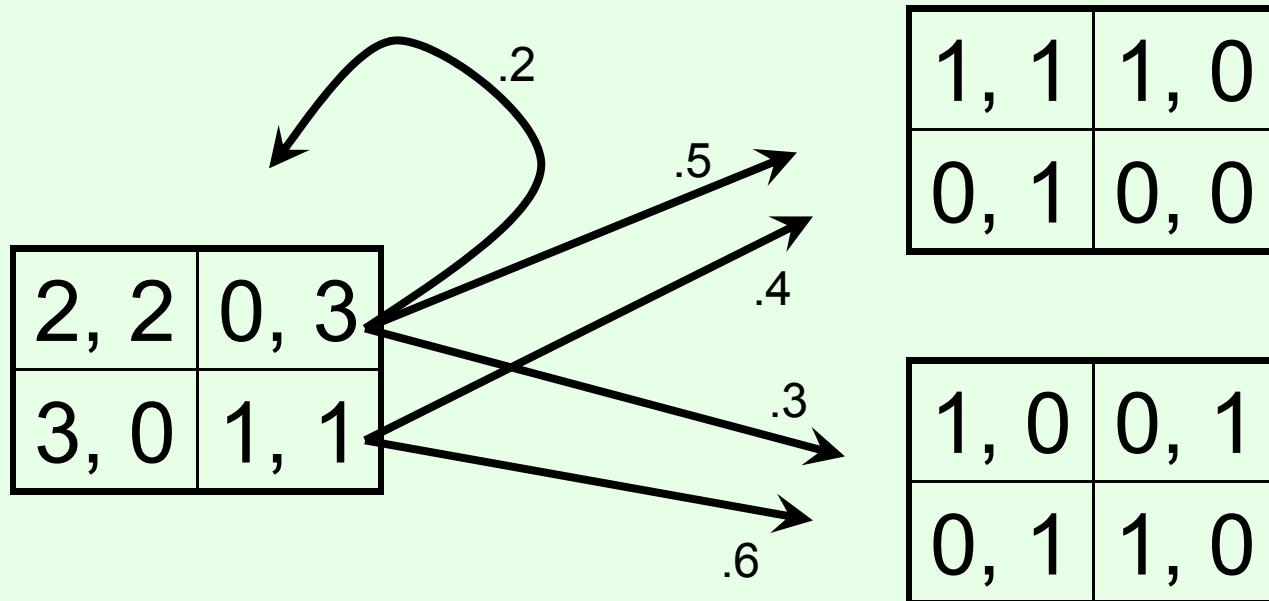
Example Markov Decision Process (MDP)

- Machine can be in one of three states: good, deteriorating, broken
- Can take two actions: maintain, ignore



Stochastic games

- A stochastic game has multiple **states** that it can be in
- Each state corresponds to a normal-form game
- After a round, the game randomly **transitions** to another state
- Transition probabilities depend on state and actions taken
- Typically utilities are discounted over time



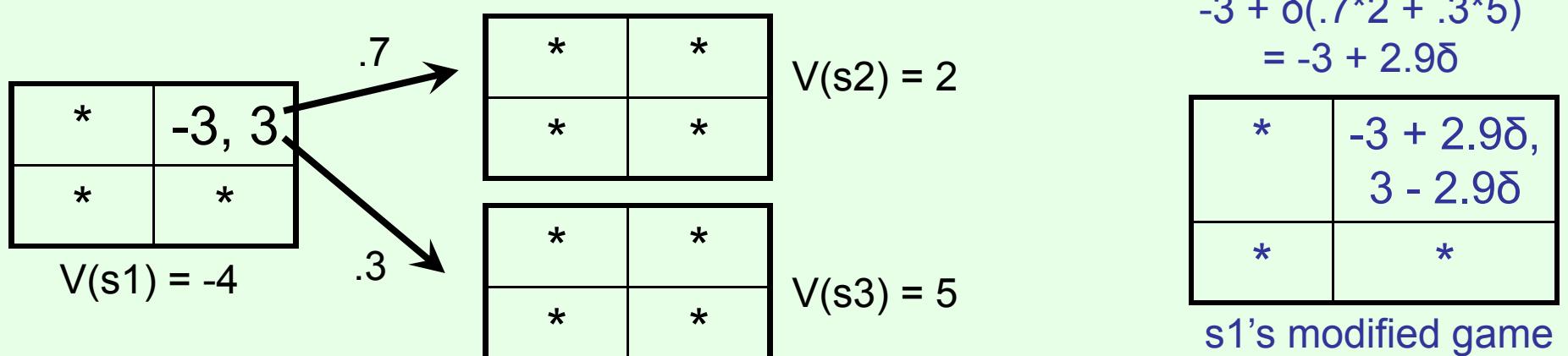
- 1-state stochastic game = (infinitely) repeated game
- 1-agent stochastic game = Markov Decision Process (MDP)

Stationary strategies

- A **stationary strategy** specifies a mixed strategy for each state
 - Strategy does **not** depend on history
 - E.g., in a repeated game, stationary strategy = always playing the same mixed strategy
- An equilibrium in stationary strategies always exists [Fink 64]
- Each player will have a **value** for being in each state

Shapley's [1953] algorithm for 2-player zero-sum stochastic games (~value iteration)

- Each state s is arbitrarily given a value $V(s)$
 - Player 1's utility for being in state s
- Now, for each state, compute a normal-form game that takes these (discounted) values into account



- Solve for the value of the modified game (using LP)
- Make this the new value of s_1
- Do this for all states, repeat until convergence
- Similarly, analogs of policy iteration [Pollatschek & Avi-Itzhak] and Q-Learning [Littman 94, Hu & Wellman 98] exist

CPS 590.4

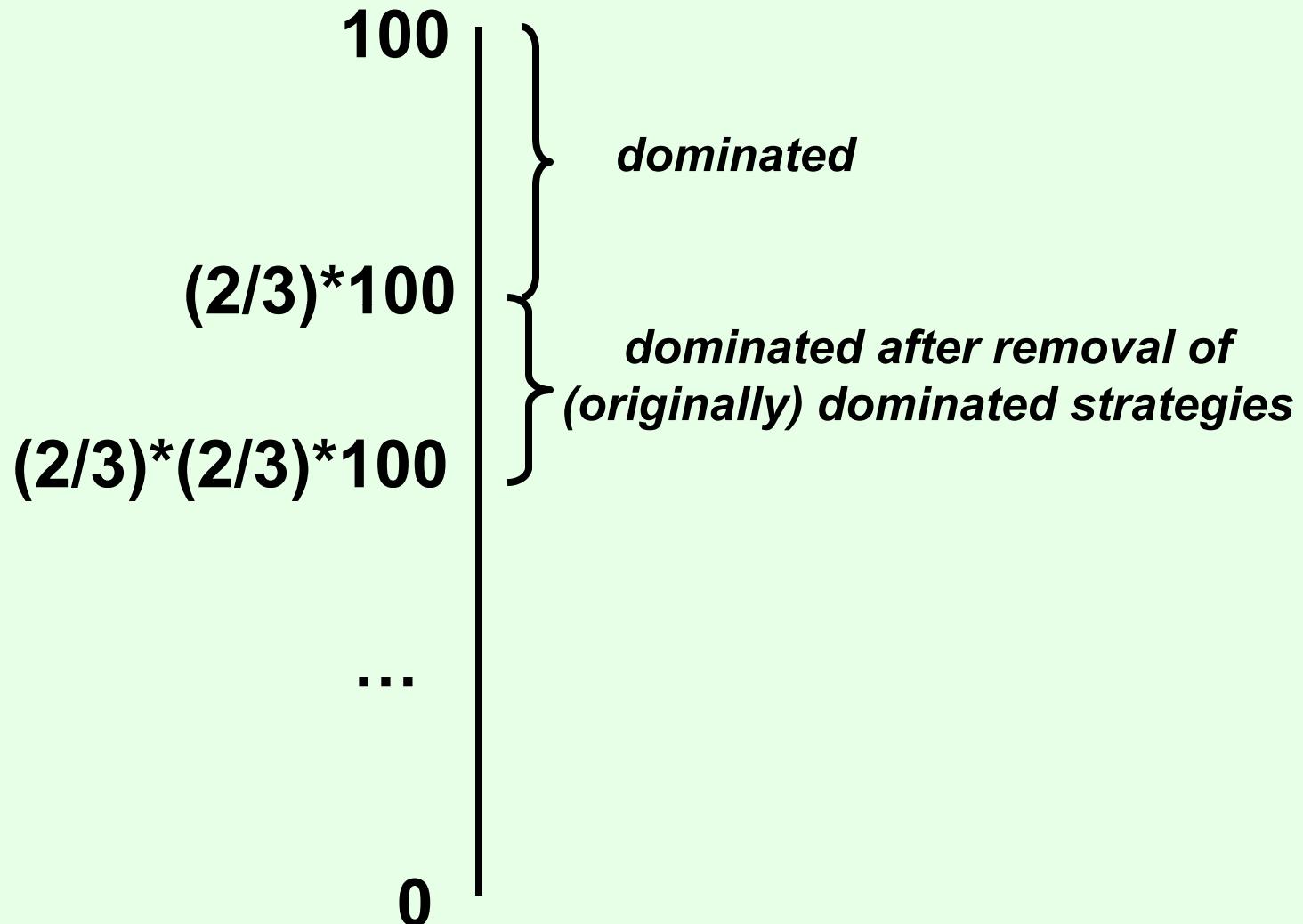
Learning in games

Vincent Conitzer
conitzer@cs.duke.edu

“2/3 of the average” game

- Everyone writes down a number between 0 and 100
- Person closest to 2/3 of the average wins
- Example:
 - A says 50
 - B says 10
 - C says 90
 - Average(50, 10, 90) = 50
 - 2/3 of average = 33.33
 - A is closest ($|50-33.33| = 16.67$), so A wins

“2/3 of the average” game revisited



Learning in (normal-form) games

- Approach we have taken so far when playing a game: just compute an optimal/equilibrium strategy
- Another approach: **learn** how to play a game by
 - playing it many times, and
 - updating your strategy based on experience
- Why?
 - Some of the game's utilities (especially the other players') may be **unknown** to you
 - The other players may **not be** playing an equilibrium strategy
 - Computing an optimal strategy can be **hard**
 - Learning is what **humans** typically do
 - ...
- Learning strategies ~ strategies for the repeated game
- Does learning converge to equilibrium?

Iterated best response

- In the first round, play something arbitrary
- In each following round, play a best response against what the other players played in the previous round
- If all players play this, it can converge (i.e., we reach an equilibrium) or cycle

0, 0	-1, 1	1, -1
1, -1	0, 0	-1, 1
-1, 1	1, -1	0, 0

rock-paper-scissors

-1, -1	0, 0
0, 0	-1, -1

a simple congestion game

- **Alternating best response:** players alternatingly change strategies: one player best-responds each odd round, the other best-responds each even round

Fictitious play [Brown 1951]

- In the first round, play something arbitrary
- In each following round, play a best response against the **empirical distribution** of the other players' play
 - I.e., as if other player randomly selects from his past actions
- Again, if this converges, we have a Nash equilibrium
- Can still fail to converge...

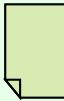
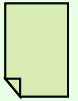
0, 0	-1, 1	1, -1
1, -1	0, 0	-1, 1
-1, 1	1, -1	0, 0

rock-paper-scissors

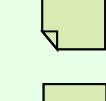
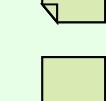
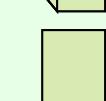
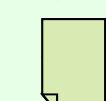
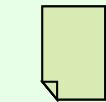
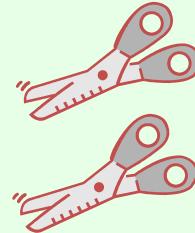
-1, -1	0, 0
0, 0	-1, -1

a simple congestion game

Fictitious play on rock-paper-scissors

			
	0, 0	-1, 1	1, -1
	1, -1	0, 0	-1, 1
	-1, 1	1, -1	0, 0

Row



Column



30% R, 50% P, 20% S

30% R, 20% P, 50% S

Does the empirical distribution of play converge to equilibrium?

- ... for iterated best response?
- ... for fictitious play?

3, 0	1, 2
1, 2	2, 1

Fictitious play is guaranteed to converge in...

- Two-player zero-sum games [Robinson 1951]
- Generic 2x2 games [Miyasawa 1961]
- Games solvable by iterated strict dominance [Nachbar 1990]
- Weighted potential games [Monderer & Shapley 1996]
- **Not** in general [Shapley 1964]
- But, fictitious play always converges to the set of $\frac{1}{2}$ -approximate equilibria [Conitzer 2009; more detailed analysis by Goldberg, Savani, Sørensen, Ventre 2011]

Shapley's game on which fictitious play does not converge

- starting with (U, M) :

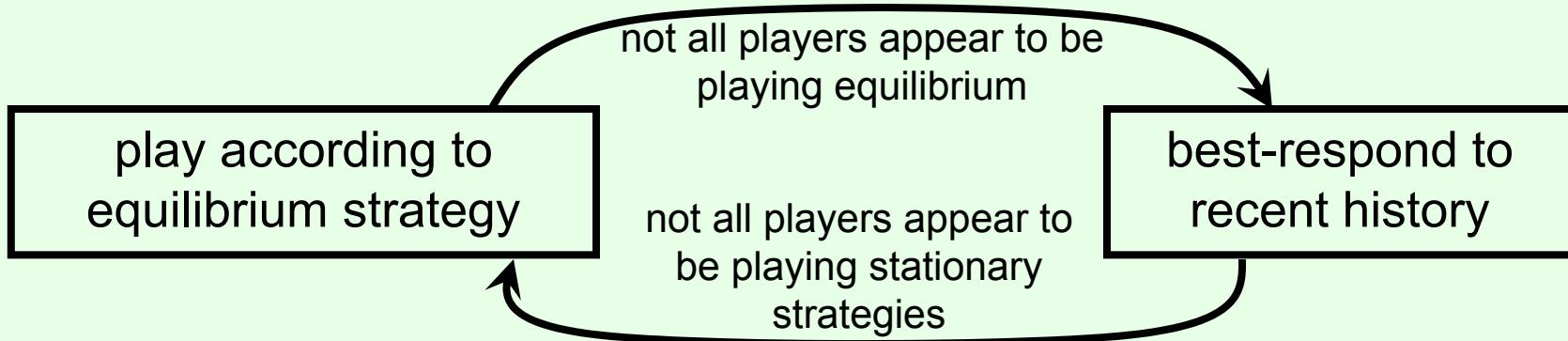
0, 0	0, 1	1, 0
1, 0	0, 0	0, 1
0, 1	1, 0	0, 0

Regret

- For each player i , action a_i and time t , define the **regret** $r_i(a_i, t)$ as $(\sum_{1 \leq t' \leq t-1} u_i(a_i, a_{-i, t'}) - u_i(a_{i,t'}, a_{-i,t'})) / (t-1)$
- An algorithm has **zero regret** if for each a_i , the regret for a_i becomes nonpositive as t goes to infinity (almost surely) against **any** opponents
- **Regret matching** [Hart & Mas-Colell 00]: at time t , play an action that has positive regret $r_i(a_i, t)$ with probability proportional to $r_i(a_i, t)$
 - If none of the actions have positive regret, play uniformly at random
- Regret matching has zero regret
- If all players use regret matching, then play converges to the set of **weak correlated equilibria**
 - Weak correlated equilibrium: playing according to joint distribution is at least as good as any strategy that does not depend on the signal
- Variants of this converge to the set of correlated equilibria
- **Smooth fictitious play** [Fudenberg & Levine 95] also gives no regret
 - Instead of just best-responding to history, assign some small value to having a more “mixed” distribution

Targeted learning

- Assume that there is a **limited** set of possible opponents
- Try to do well against these
- Example: is there a learning algorithm that
 - learns to best-respond against any stationary opponent (one that always plays the same mixed strategy), and
 - converges to a Nash equilibrium (in actual strategies, not historical distribution) when playing against a copy of itself (so-called **self-play**)?
- [Bowling and Veloso AIJ02]: yes, if it is a 2-player 2x2 game and mixed strategies are observable
- [Conitzer and Sandholm ML06]: yes (without those assumptions)
 - AWESOME algorithm (Adapt When Everybody is Stationary, Otherwise Move to Equilibrium): (very) rough sketch:



“Teaching”

- Suppose you are playing against a player that uses a strategy that eventually learns to best-respond
- Also suppose you are very patient, i.e., you only care about what happens in the long run
- How will you (the row player) play in the following repeated games?

4, 4	3, 5
5, 3	0, 0

1, 0	3, 1
2, 1	4, 0

- Note relationship to optimal strategies to commit to
- There is some work on learning strategies that are in **equilibrium** with each other [Brafman & Tennenholtz AIJ04]

Evolutionary game theory

- Given: a symmetric game

	dove	hawk
dove	1, 1	0, 2
hawk	2, 0	-1, -1

Nash equilibria: (d, h),
(h, d), ((.5, .5), (.5, .5))

- A large population of players plays this game, players are randomly matched to play with each other
- Each player plays a pure strategy
 - Fraction of players playing strategy $s = p_s$
 - p is vector of all fractions p_s (the **state**)
- Utility for playing s is $u(s, p) = \sum_{s'} p_{s'} u(s, s')$
- Players **reproduce** at a rate that is proportional to their utility, their offspring play the same strategy
 - Replicator dynamic**
- $\frac{dp_s(t)}{dt} = p_s(t)(u(s, p(t)) - \sum_{s'} p_{s'} u(s', p(t)))$
- What are the **steady states** of this?

Stability

	dove	hawk
dove	1, 1	0, 2
hawk	2, 0	-1, -1

- A steady state is **stable** if slightly perturbing the state will not cause us to move far away from the state
- E.g. everyone playing dove is not stable, because if a few hawks are added their percentage will grow
- What about the mixed steady state?
- Proposition: every stable steady state is a Nash equilibrium of the symmetric game
- Slightly stronger criterion: a state is **asymptotically stable** if it is stable, and after slightly perturbing this state, we will (in the limit) return to this state

Evolutionarily stable strategies

- Now suppose players play **mixed** strategies
- A (single) mixed strategy σ is **evolutionarily stable** if the following is true:
 - Suppose all players play σ
 - Then, whenever a very small number of **invaders** enters that play a different strategy σ' ,
 - the players playing σ must get strictly **higher** utility than those playing σ' (i.e., σ must be able to **repel invaders**)
- σ will be evolutionarily stable if and only if for all σ'
 - $u(\sigma, \sigma) > u(\sigma', \sigma)$, or:
 - $u(\sigma, \sigma) = u(\sigma', \sigma)$ and $u(\sigma, \sigma') > u(\sigma', \sigma')$
- Proposition: every evolutionarily stable strategy is asymptotically stable under the replicator dynamic

Invasion (1/2)

	Dove	Hawk
Dove	1, 1	0, 2
Hawk	2, 0	-1, -1

- Given: population P_1 that plays $\sigma = 40\%$ Dove, 60% Hawk
- Tiny population P_2 that plays $\sigma' = 70\%$ Dove, 30% Hawk **invades**
- $u(\sigma, \sigma) = .16*1 + .24*2 + .36*(-1) = .28$ but
 $u(\sigma', \sigma) = .28*1 + .12*2 + .18*(-1) = .34$
- σ' (initially) grows in the population; invasion is **successful**

Invasion (2/2)

	Dove	Hawk
Dove	1, 1	0, 2
Hawk	2, 0	-1, -1

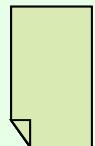
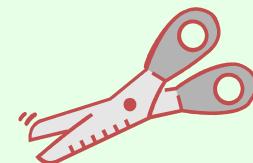
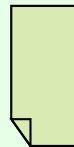
- Now P_1 plays $\sigma = 50\%$ Dove, 50% Hawk
- Tiny population P_2 that plays $\sigma' = 70\%$ Dove, 30% Hawk **invades**
- $u(\sigma, \sigma) = u(\sigma', \sigma) = .5$, so second-order effect:
- $u(\sigma, \sigma') = .35*1 + .35*2 + .15*(-1) = .9$ but
 $u(\sigma', \sigma') = .49*1 + .21*2 + .09*(-1) = .82$
- σ' shrinks in the population; invasion is **repelled**

Evolutionarily stable strategies

[Price and Smith, 1973]

- A strategy σ is **evolutionarily stable** if the following two conditions both hold:
 - (1) For all σ' , we have $u(\sigma, \sigma) \geq u(\sigma', \sigma)$ (i.e., symmetric Nash equilibrium)
 - (2) For all $\sigma' (\neq \sigma)$ with $u(\sigma, \sigma) = u(\sigma', \sigma)$, we have $u(\sigma, \sigma') > u(\sigma', \sigma')$

Rock-Paper-Scissors

		
		
0, 0	-1, 1	1, -1
1, -1	0, 0	-1, 1
-1, 1	1, -1	0, 0

- Only one Nash equilibrium (Uniform)
- $u(\text{Uniform}, \text{Rock}) = u(\text{Rock}, \text{Rock})$
- No ESS

The standard Σ_2^P -complete problem

Input: Boolean formula f over variables X_1 and X_2

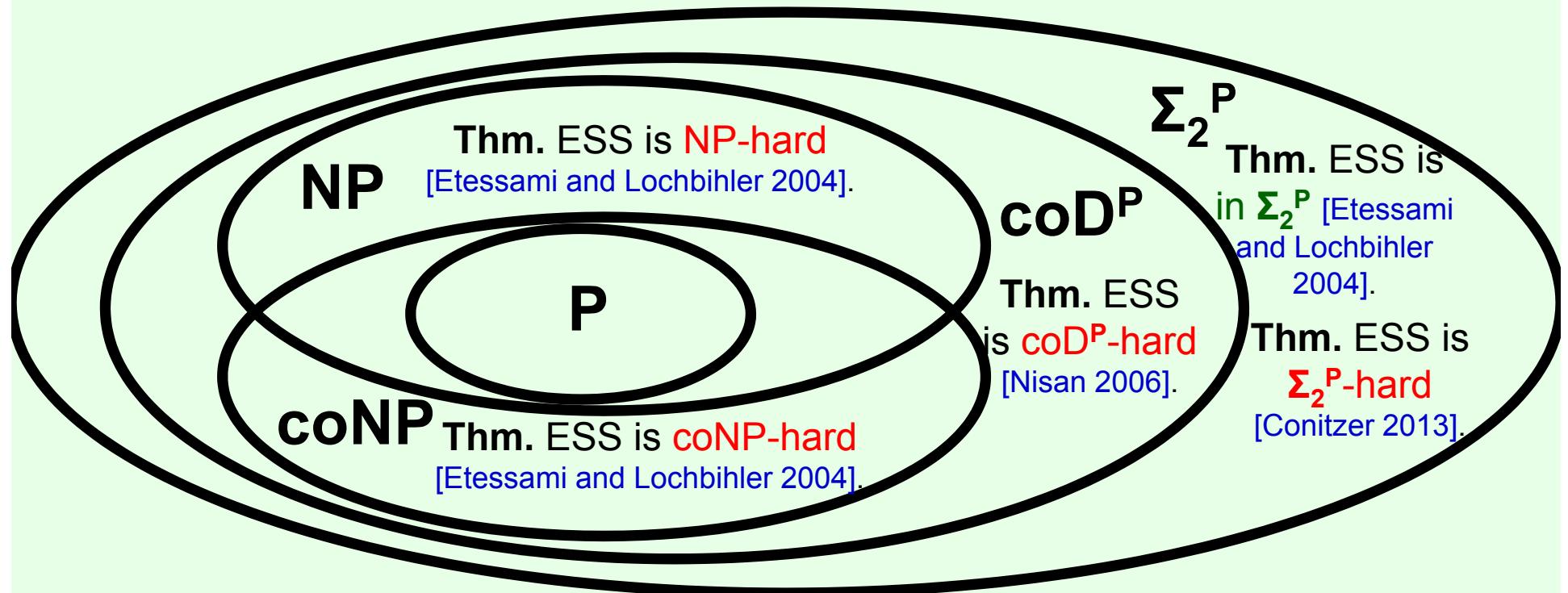
Q: Does there exist an assignment of values to X_1 such that for every assignment of values to X_2 , f is true?

The ESS problem

Input: symmetric 2-player normal-form game.

Q: Does it have an evolutionarily stable strategy?

(Hawk-Dove: yes. Rock-Paper-Scissors: no. Safe-Left-Right: no.)



Automated mechanism design

Vincent Conitzer
conitzer@cs.duke.edu

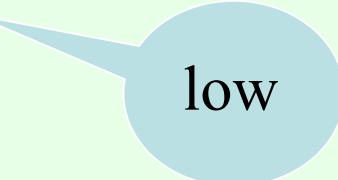
General vs. specific mechanisms

- Mechanisms such as Clarke (VCG) mechanism are very **general**...
- ... but will instantiate to something **specific** in any specific setting
 - This is what we care about

Example: Divorce arbitration

- Outcomes:
- Each agent is of *high* type w.p. .2 and *low* type w.p. .8
 - Preferences of *high* type:
 - $u(\text{get the painting}) = 11,000$
 - $u(\text{museum}) = 6,000$
 - $u(\text{other gets the painting}) = 1,000$
 - $u(\text{burn}) = 0$
 - Preferences of *low* type:
 - $u(\text{get the painting}) = 1,200$
 - $u(\text{museum}) = 1,100$
 - $u(\text{other gets the painting}) = 1,000$
 - $u(\text{burn}) = 0$

Clarke (VCG) mechanism

		
		
	 Both pay 5,000	 Husband pays 200
	 Wife pays 200	 Both pay 100

Expected sum of divorcees' utilities = 5,136

“Manual” mechanism design has yielded

- some **positive results**:
 - “Mechanism x achieves properties P in any setting that belongs to class C”
- some **impossibility results**:
 - “There is no mechanism that achieves properties P for all settings in class C”

Difficulties with manual mechanism design

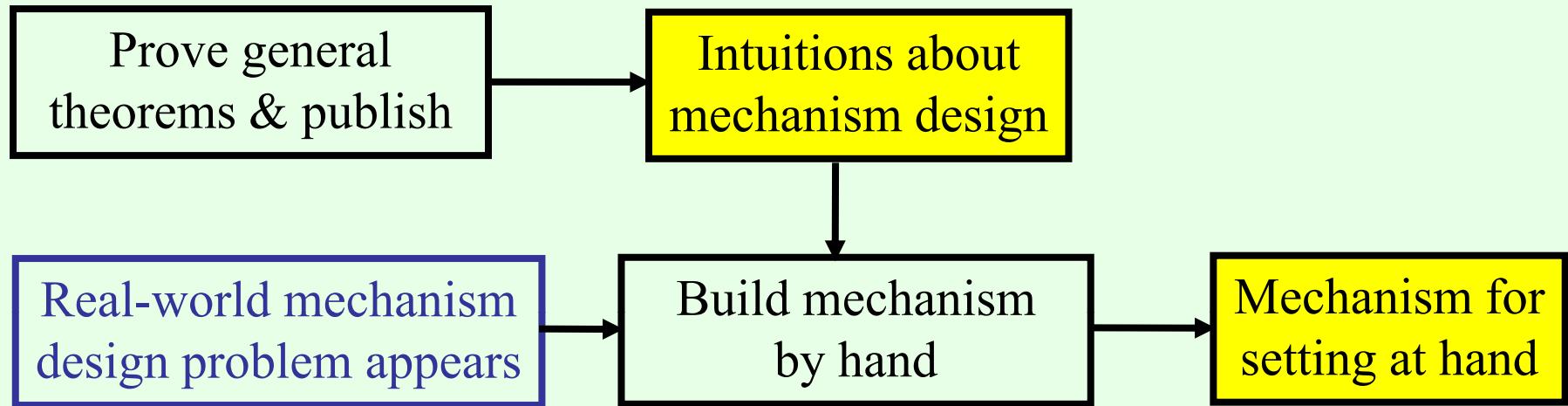
- Design problem instance comes along
 - Set of outcomes, agents, set of possible types for each agent, prior over types, ...
- What if **no** canonical mechanism covers this instance?
 - Unusual objective, or payments not possible, or ...
 - Impossibility results may exist for the general class of settings
 - But instance may have additional structure (restricted preferences or prior) so good mechanisms exist (but unknown)
- What if a canonical mechanism **does** cover the setting?
 - Can we use instance's structure to get higher objective value?
 - Can we get stronger nonmanipulability/participation properties?
- Manual design for every instance is prohibitively slow

Automated mechanism design (AMD)

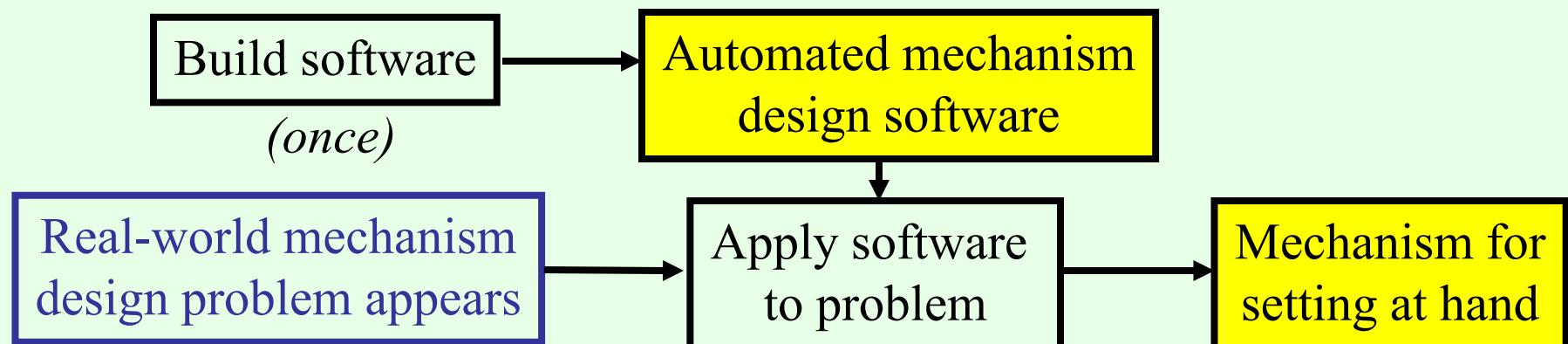
- Idea: Solve mechanism design **as optimization problem** automatically
- Create a mechanism **for the specific setting at hand** rather than a class of settings
- Advantages:
 - Can lead to greater value of designer's objective than known mechanisms
 - Sometimes circumvents economic impossibility results & always minimizes the pain implied by them
 - Can be used in new settings & for unusual objectives
 - Can yield stronger incentive compatibility & participation properties
 - Shifts the burden of design from human to machine

Classical vs. automated mechanism design

Classical



Automated



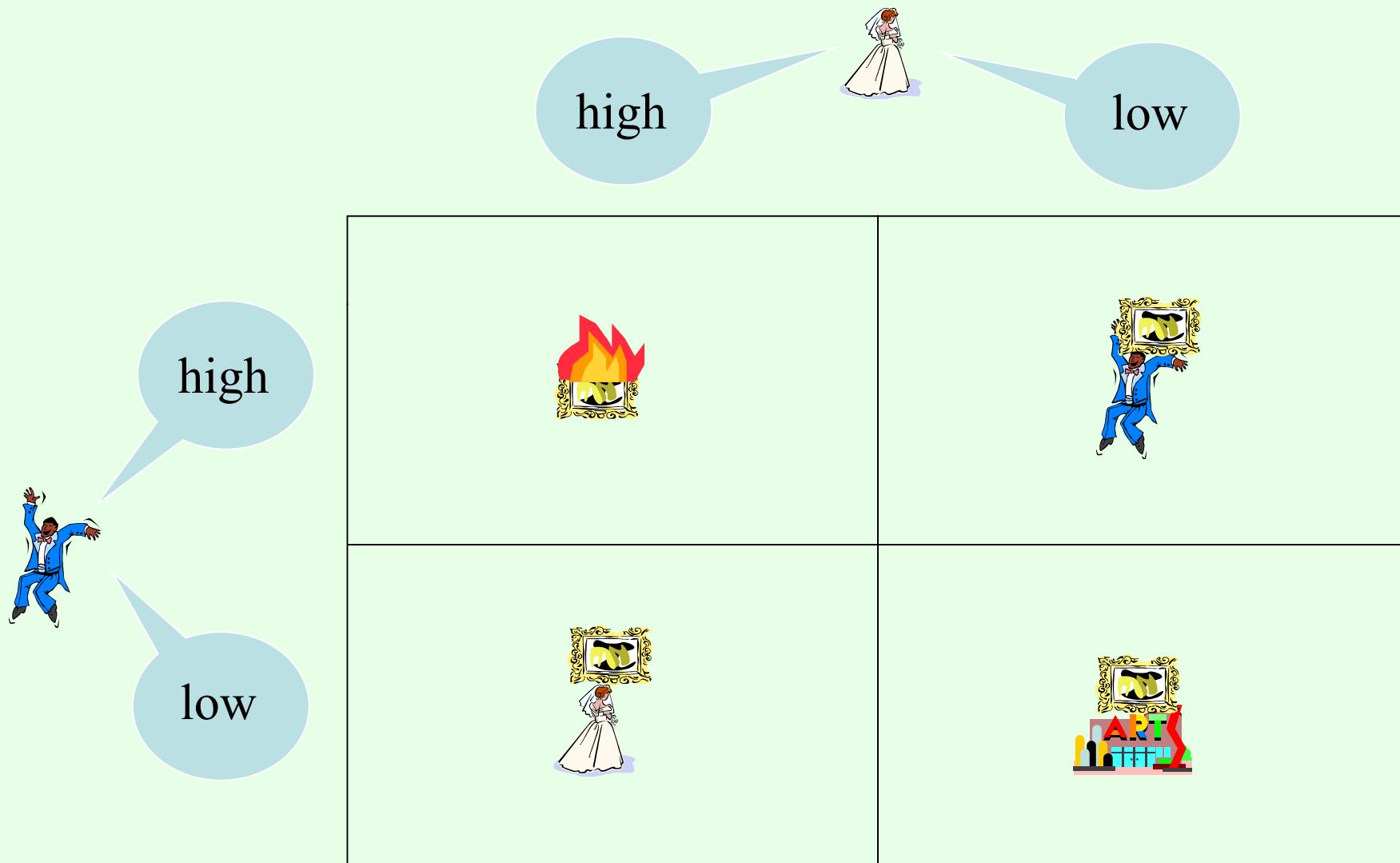
Input

- Instance is given by
 - Set of possible *outcomes*
 - Set of *agents*
 - For each agent
 - set of possible *types*
 - *probability distribution* over these types
 - *Objective function*
 - Gives a value for each outcome for each combination of agents' types
 - E.g. social welfare, payment maximization
 - Restrictions on the mechanism
 - Are payments allowed?
 - Is randomization over outcomes allowed?
 - What versions of incentive compatibility (IC) & individual rationality (IR) are used?

Output

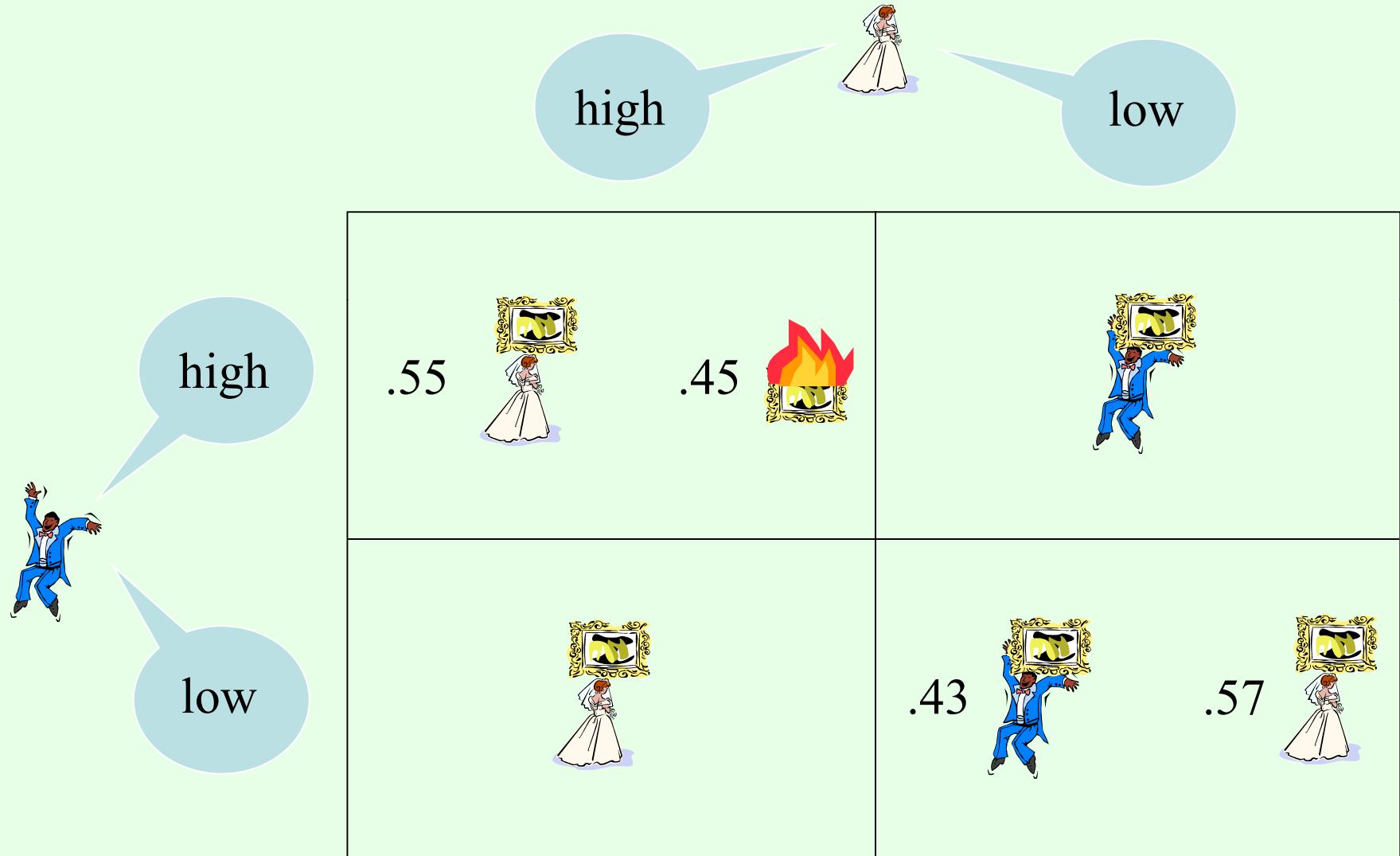
- *Mechanism*
 - A mechanism maps combinations of agents' revealed types to outcomes
 - *Randomized mechanism* maps to probability distributions over outcomes
 - Also specifies payments by agents (if payments allowed)
- ... which
 - satisfies the IR and IC constraints
 - maximizes the expectation of the objective function

Optimal BNE incentive compatible deterministic mechanism without payments for maximizing sum of divorcees' utilities



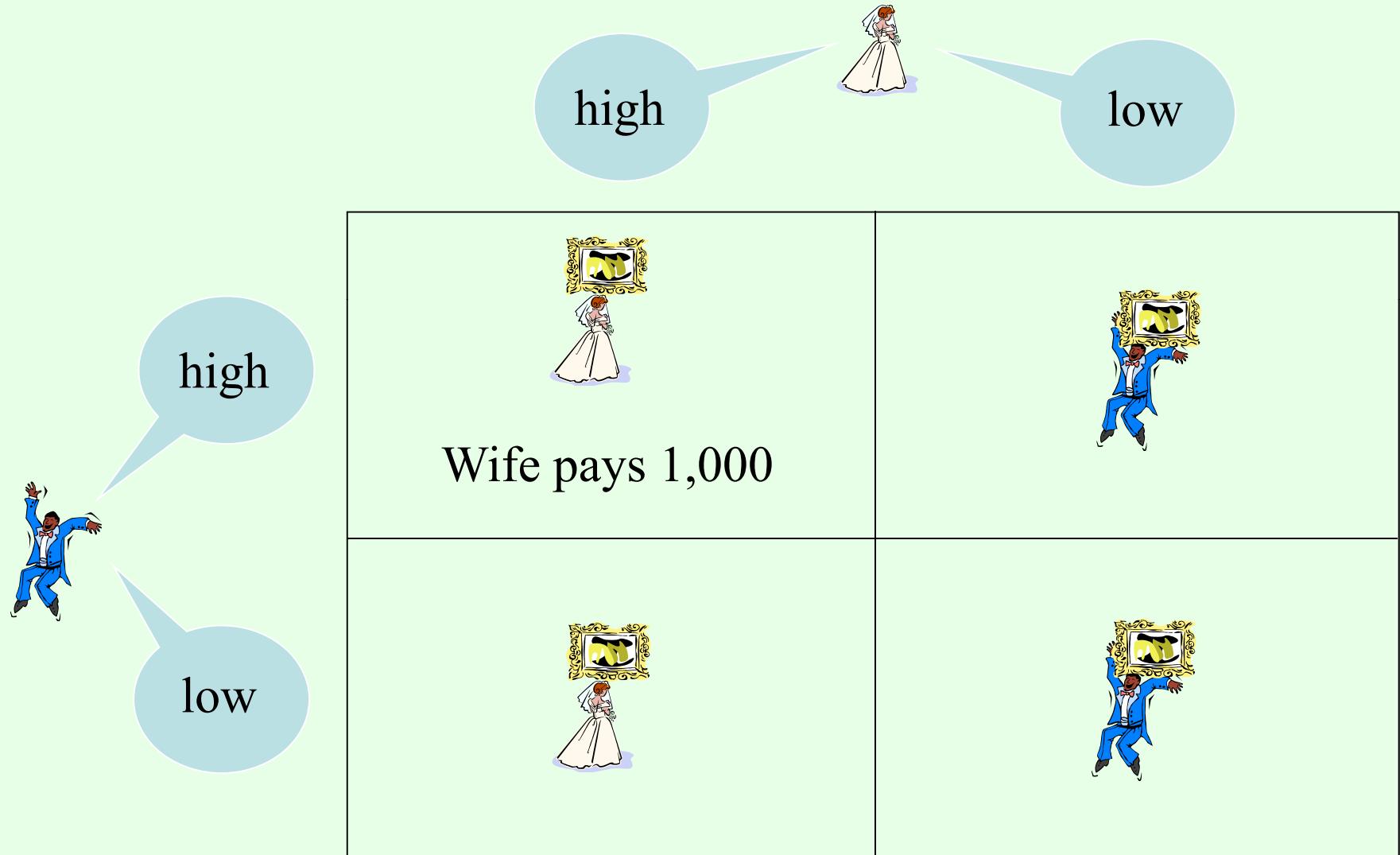
Expected sum of divorcees' utilities = 5,248

Optimal BNE incentive compatible *randomized* mechanism without payments for maximizing sum of divorcees' utilities



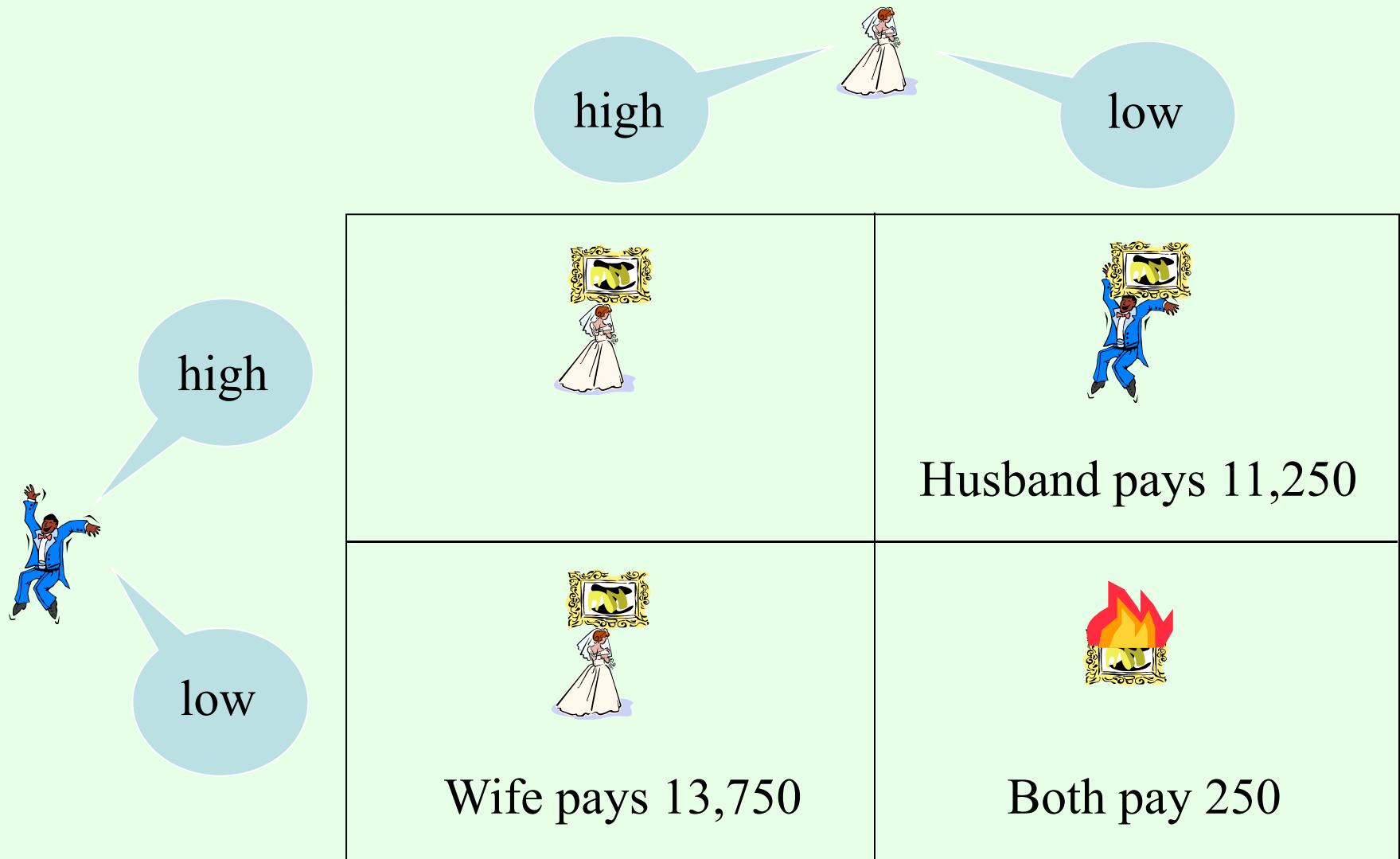
Expected sum of divorcees' utilities = 5,510

Optimal BNE incentive compatible randomized mechanism *with payments* for maximizing sum of divorcees' utilities



Expected sum of divorcees' utilities = 5,688

Optimal BNE incentive compatible randomized mechanism with payments for *maximizing arbitrator's revenue*



Expected sum of divorcees' utilities = 0

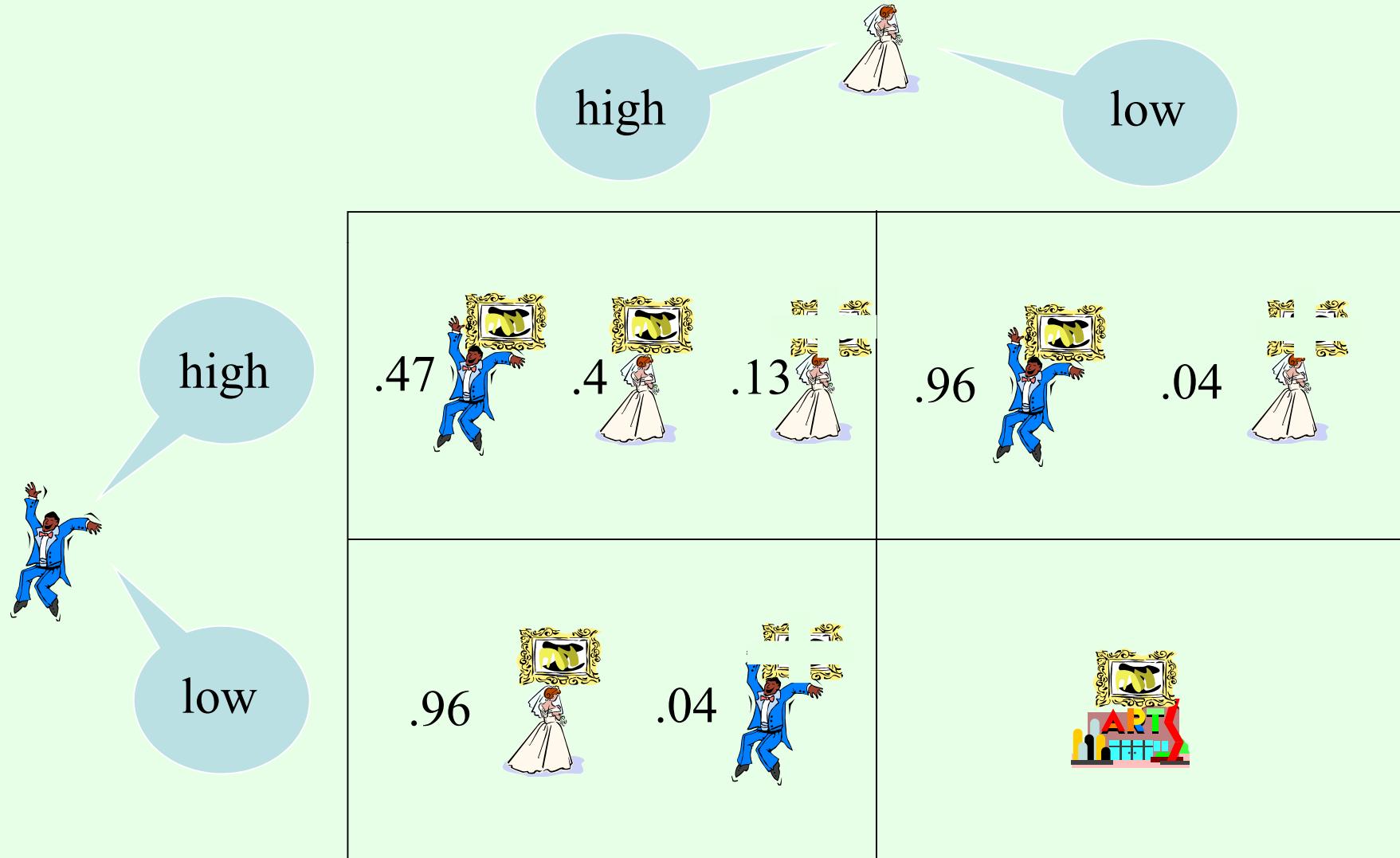
Arbitrator expects 4,320

Modified divorce arbitration example



- Outcomes:
- Each agent is of *high* type with probability 0.2 and of *low* type with probability 0.8
 - Preferences of *high* type:
 - $u(\text{get the painting}) = 100$
 - $u(\text{other gets the painting}) = 0$
 - $u(\text{museum}) = 40$
 - $u(\text{get the pieces}) = -9$
 - $u(\text{other gets the pieces}) = -10$
 - Preferences of *low* type:
 - $u(\text{get the painting}) = 2$
 - $u(\text{other gets the painting}) = 0$
 - $u(\text{museum}) = 1.5$
 - $u(\text{get the pieces}) = -9$
 - $u(\text{other gets the pieces}) = -10$

Optimal *dominant-strategies* incentive compatible randomized mechanism for maximizing expected sum of utilities



How do we set up the optimization?

- Use linear programming
- Variables:
 - $p(o | \theta_1, \dots, \theta_n)$ = probability that outcome o is chosen given types $\theta_1, \dots, \theta_n$
 - (maybe) $\pi_i(\theta_1, \dots, \theta_n)$ = i 's payment given types $\theta_1, \dots, \theta_n$
- Strategy-proofness constraints: for all $i, \theta_1, \dots, \theta_n, \theta'_i$:
$$\sum_o p(o | \theta_1, \dots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \dots, \theta_n) \geq \sum_o p(o | \theta_1, \dots, \theta'_i, \dots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \dots, \theta'_i, \dots, \theta_n)$$
- Individual-rationality constraints: for all $i, \theta_1, \dots, \theta_n$:
$$\sum_o p(o | \theta_1, \dots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \dots, \theta_n) \geq 0$$
- Objective (e.g. sum of utilities)
$$\sum_{\theta_1, \dots, \theta_n} p(\theta_1, \dots, \theta_n) \sum_i (\sum_o p(o | \theta_1, \dots, \theta_n) u_i(\theta_i, o) + \pi_i(\theta_1, \dots, \theta_n))$$
- Also works for BNE incentive compatibility, ex-interim individual rationality notions, other objectives, etc.
- For deterministic mechanisms, use mixed integer programming (probabilities in $\{0, 1\}$)
 - Typically designing the optimal deterministic mechanism is NP-hard

Computational complexity of automatically designing deterministic mechanisms

- Many different variants
 - Objective to maximize: Social welfare/revenue/designer's agenda for outcome
 - Payments allowed/not allowed
 - IR constraint: ex interim IR/ex post IR/no IR
 - IC constraint: Dominant strategies/Bayes-Nash equilibrium
- The above already gives $3 * 2 * 3 * 2 = 36$ variants
- Approach: Prove hardness for the case of only 1 type-reporting agent
 - results imply hardness in more general settings

DSE & BNE incentive compatibility constraints coincide when there is only 1 (reporting) agent

Dominant strategies:

Reporting truthfully is optimal
for any types the others
report

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$$u_1(t_{11}, o_5) \geq u_1(t_{11}, o_3)$$

AND

$$u_1(t_{11}, o_9) \geq u_1(t_{11}, o_2)$$

With only 1
reporting agent,
the constraints are
the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

Bayes-Nash equilibrium:

Reporting truthfully is optimal
in expectation over the other
agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$$P(t_{21})u_1(t_{11}, o_5) + P(t_{22})u_1(t_{11}, o_9) \geq P(t_{21})u_1(t_{11}, o_3) + P(t_{22})u_1(t_{11}, o_2)$$

$$u_1(t_{11}, o_5) \geq u_1(t_{11}, o_3)$$

is equivalent to

$$P(t_{21})u_1(t_{11}, o_5) \geq P(t_{21})u_1(t_{11}, o_3)$$

Ex post and *ex interim* individual rationality constraints coincide when there is only 1 (reporting) agent

Ex post:

Participating never hurts (for any types of the other agents)

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$$u_1(t_{11}, o_5) \geq 0 \text{ AND } u_1(t_{11}, o_9) \geq 0$$

Ex interim:

Participating does not hurt *in expectation* over the other agents' (true) types

	t_{21}	t_{22}
t_{11}	o_5	o_9
t_{12}	o_3	o_2

$$P(t_{21})u_1(t_{11}, o_5) + P(t_{22})u_1(t_{11}, o_9) \geq 0$$

With only 1 reporting agent, the constraints are the same

	t_{21}
t_{11}	o_5
t_{11}	o_3

$$u_1(t_{11}, o_5) \geq 0 \text{ is equivalent to } P(t_{21})u_1(t_{11}, o_5) \geq 0$$

How hard is designing an optimal *deterministic* mechanism?

NP-complete (even with 1 reporting agent):	Solvable in polynomial time (for any constant number of agents):
<ol style="list-style-type: none">1. Maximizing social welfare (no payments)2. Designer's own utility over outcomes (no payments)3. General (linear) objective that doesn't regard payments4. Expected revenue	<ol style="list-style-type: none">1. Maximizing social welfare (not regarding the payments) (VCG)

1 and 3 hold even with no IR constraints

AMD can create optimal (expected-revenue maximizing) combinatorial auctions

- Instance 1
 - 2 items, 2 bidders, 4 types each (LL, LH, HL, HH)
 - H=utility 2 for that item, L=utility 1
 - But: utility 6 for getting both items if type HH (complementarity)
 - Uniform prior over types
 - Optimal *ex-interim* IR, BNE mechanism (0 = item is burned):
 - Payment rule not shown
 - Expected revenue: 3.94 (VCG: 2.69)
- Instance 2
 - 2 items, 3 bidders
 - Complementarity and substitutability
 - Took 5.9 seconds
 - Uses randomization

	LL	LH	HL	HH
LL	0,0	0,2	2,0	2,2
LH	0,1	1,2	2,1	2,2
HL	1,0	1,2	2,1	2,2
HH	1,1	1,1	1,1	1,1

Optimal mechanisms for a public good

- AMD can design optimal mechanisms for public goods, taking money burning into account as a loss
- Bridge building instance
 - Agent 1: High type (prob .6) values bridge at 10. Low: values at 1
 - Agent 2: High type (prob .4) values bridge at 11. Low: values at 2
 - Bridge costs 6 to build
- Optimal mechanism (*ex-post* IR, BNE):

<i>Outcome rule</i>	Low	High	<i>Payment rule</i>	Low	High
Low	Don't build	Build	Low	0, 0	0, 6
High	Build	Build	High	4, 2	.67, 5.33

- There is no general mechanism that achieves budget balance, *ex-post* efficiency, and *ex-post* IR [Myerson-Satterthwaite 83]
- However, for this instance, AMD found such a mechanism

Combinatorial public goods problems

- AMD for interrelated public goods
- Example: building a bridge and/or a boat
 - 2 agents each uniform from types: {None, Bridge, Boat, Either}
 - Type indicates which of the two would be useful to the agent
 - If something is built that is useful to you, you get 2, otherwise 0
 - Boat costs 1 to build, bridge 3
- Optimal mechanism (*ex-post* IR, dominant strategies):

Outcome rule
 $(P(\text{none}), P(\text{boat}), P(\text{bridge}), P(\text{both}))$

	None	Boat	Bridge	Either
None	(1,0,0,0)	(0,1,0,0)	(1,0,0,0)	(0,1,0,0)
Boat	(.5,.5,0,0)	(0,1,0,0)	(0,.5,0,.5)	(0,1,0,0)
Bridge	(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,1,0)
Either	(.5,.5,0,0)	(0,1,0,0)	(0,0,1,0)	(0,1,0,0)

- Again, no money burning, but outcome not always efficient
 - E.g., sometimes nothing is built while boat should have been

Additional & future directions

- Scalability is a major concern
 - Can sometimes create more concise LP formulations
 - Sometimes, some constraints are implied by others
 - In restricted domains faster algorithms sometimes exist
 - Can sometimes make use of partial characterizations of the optimal mechanism
- Automatically generated mechanisms can be complex/hard to understand
 - Can we make automatically designed mechanisms more intuitive?
- Using AMD to create conjectures about general mechanisms