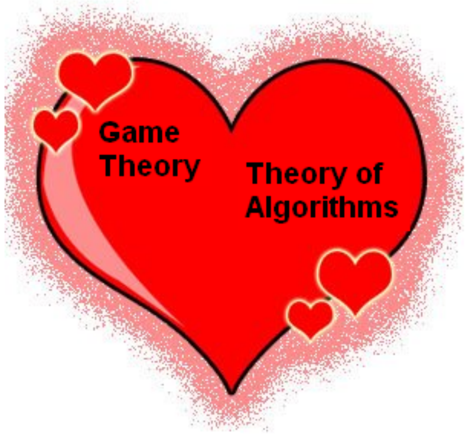
Day 1:



Matrix games (a.k.a. finite two-player zero-sum games in strategic form)

Player 1:
Max, the
Maximizer

Player 2 :
Min, the
Minimizer

- Game given by m x n real matrix A = (a_{ij})
- Strategy space for Max: 1,2,..,*m.*
- Strategy space for Min: 1,2,..,*n.*
- Max and Min each chooses a strategy without information about the choice of the other player.
- If Max plays i and Min plays j, Max earns a_{ij} and Min earns $-a_{ij}$.

Matrix games (a.k.a. finite two-player zero-sum games in strategic form)

Player 1:
Max, the
Maximizer

Player 2 :
Min, the
Minimizer

$$\underline{v} = \max_{x\in\Delta_m} \min_{y\in\Delta_n} x^{\mathrm{T}}\mathbf{A}y$$ Maxmin value

$$x^* = \arg\max_{x\in\Delta_m} \min_{y\in\Delta_n} x^{\mathrm{T}}\mathbf{A}y$$ Max' optimal strategy

$$\bar{v} = \min_{y\in\Delta_m} \max_{x\in\Delta_n} x^{\mathrm{T}}\mathbf{A}y$$ Minmax value

$$y^* = \arg\min_{y\in\Delta_m} \max_{x\in\Delta_n} x^{\mathrm{T}}\mathbf{A}y$$ Min's optimal strategy

Von Neumann's Minmax theorem

- $\underline{v} = \overline{v} \leftarrow$ value .

- Vectors (x*,y*) are the Nash equilibria of the game.

Matching Pennies

Max hides a penny. If Min can guess if it is heads up or heads up, he gets the penny.

	Guess head up	Guess tails up	
Hide heads up	-1	0	1/2
Hide tails up	0	-1	1/2
	1/2	1/2	

Optimal strategies

Value: -1/2

Solving matrix games

- Computational problem MATRIX-GAME:
  - Input: An m x n matrix A with rational entries.
  - Output: The value v of the game and one optimal strategy profile (x*, y*).

????

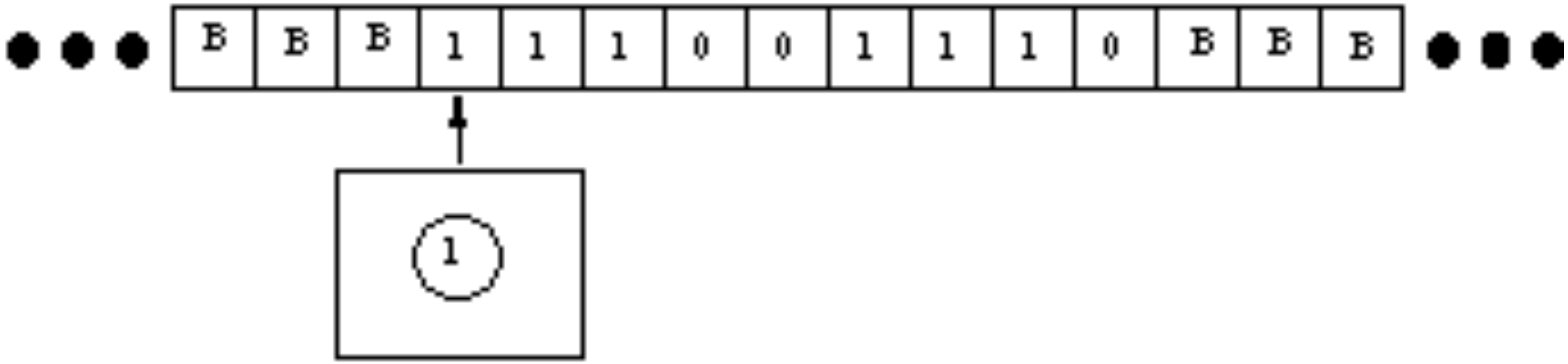
Rules for computational problems

- **Input and output should be bit strings**
  - Computer science models computation by digital computers and bit strings are all digital computers can store.
  - ***A large part of the power of the theory comes from this fact.***
- **The computational task is a specification of a relation R between inputs and outputs**.
- **There should be arbitrarily long inputs with some legal output.**
  - Otherwise the task can be trivially solved by lookup in a finite table.

Polynomial time solvable problems

- A computational task is ***polynomial time solvable*** if there is a ***Turing machine*** T that ***solves*** the task ***in polynomial time***.

Turing machine



- Turing machine = (Painfully) detailed clean formal model of digital computer.
- One computation step of Turing machine *roughly* corresponds to atomic Boolean operation (AND of two bits, OR of two bits, negation of a bit).
- Details *not* important.

.. solves the computational task …

- When a string x is placed on the input tape and there is *some* legal output y coresponding to input x, and the machine is started,
- the machine eventually halts and proeduces a bit string y so that y is a legal output for x.

… in polynomial time

- There is a polynomial p so that for any input of bit length at most L the machine halts after at most p(L) steps.

Polynomial time solvable problems

A computational task is ***polynomial time solvable*** if there is a ***Turing machine*** T that ***solves*** the task ***in polynomial time***.

Important disclaimers

A computational task is ***polynomial time solvable*** if there is a ***Turing machine*** T that ***solves*** the task ***in polynomial time***.

- The details of the model of computation does not matter for the notion of ”polynomial time solvable task.”
  - If we switch to another (reasonable) model of computation, we can replace the polynomial with a faster growing one.
- When we care about ***detailed*** running times of natural algorithms, we do ***not*** use the Turing machine model - nor do we measure running time as a function of bit length.
  - We measure some natural quantity of the algorithm, such as ”number of iterations”.
  - We measure this as a function of natural parameters of the input.

Importance of Polynomial time

- When a natural time complexity bound is not polynomially bounded, it is usually exponential.

- With 10^9 operations per second, how long does it take to perform:
- 2^{25} operations?

  0.03 seconds.
- 2^{50} operations?

  13 days.
- 2^{100} operations?

  40000 billion years…..

Real importance

- Thesis: A polynomial time algorithm often explains the ”fundamental nature” of a non-trivial computational problem.
- An exponential time algorithm often does not.

- ”Polynomial time algorithm” is a reasonable model of ”reasonable algorithm”

Solving matrix games

- Computational problem MATRIX-GAME:
  - Input: An m x n matrix A with rational entries.
  - Output: The value v of the game and one optimal strategy profile (x*, y*).

Matching Pennies

Max hides a penny. If Min can guess if it is heads up or tails up, he gets the penny.

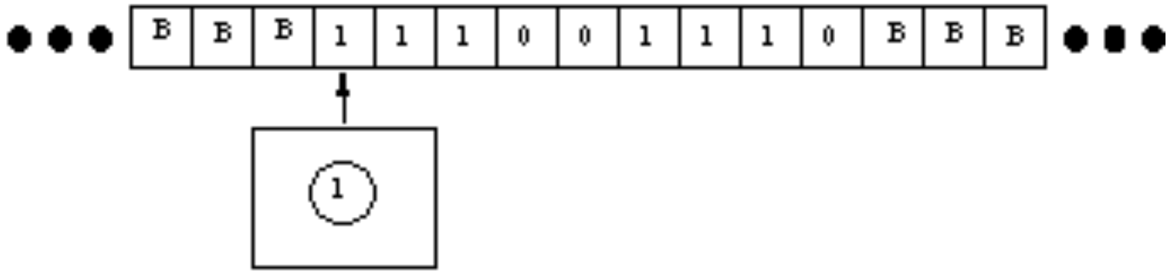
	Guess head up	Guess tails up	
Hide heads up	-1	0	1/2
Hide tails up	0	-1	1/2
	1/2	1/2	

Optimal strategies

Value: -1/2

Solving MATRIX-GAME

[[-1,0],[0,-1]]



[-1/2,[1/2,1/2],[1/2,1/2]]

Silly but correct algorithm solving matrix games

- If the input length is L, search through all strings of length at most $(10L)^{10}$.

- For each such string, check if it parses correctly into a number and two mixed strategies. Also check if these two strategies is a Nash equilibrium and that the equilibrium payoff is the number.

- The first string passing all these checks is given as output.

How to rule out the silly algorithm as an algorithm worth considering?

- **Best way I know**:
  - It is not a polynomial time algorithm, as number of computational steps in the worst case is at least $2^{(10L)^{10}}$.

How to really solve matrix games

- The value v and optimal strategy x* for matrix game **A** is given by the ***linear program:***

> Find (v,x) maximizing v s.t.
>
> $v\, \mathbf{e}_{\mathbf{n}} \cdot x^{T} \mathbf{A}$
>
> $x^{T} \mathbf{e}_{m} = 1$
>
> x ¸ 0

- where e_m, e_n are ”all-one” vectors of appropriate dimension.

Linear programming

- LINEAR-PROGRAM:
  - Input: A linear program with rational coefficients.
  - Output: An optimal solution if one exists, otherwise a report that no solution exists.

LINEAR-PROGRAM is polynomial time solvable

- Ellipsoid algorithm
  - Khachiyan (1974)
- Interior point algorithms
  - Karmakar (1984),…
- ***Not*** the simplex algorithm
  - Klee and Minty (1972)

How to solve MATRIX-GAME in polynomial time

- Since linear programming is polynomial time solvable, we can easily take a Turing machine for LINEAR-PROGRAMMING and build a Turing machine for MATRIX-GAME by doing some easy preprocessing.

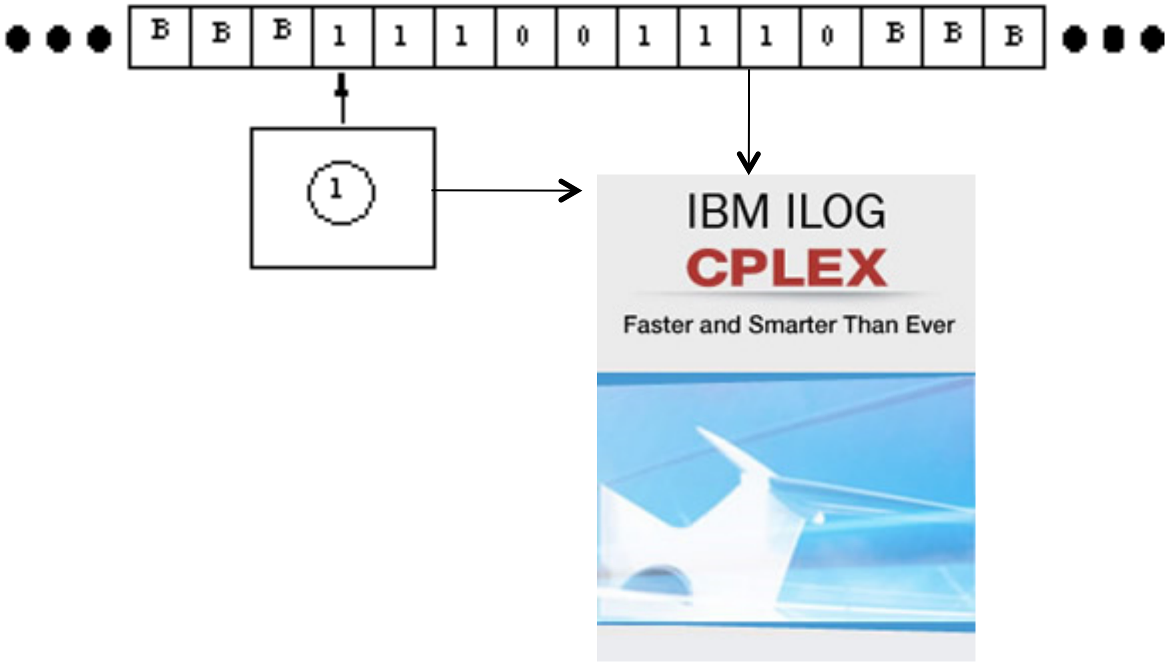
Formalization:

- MATRIX-GAME ***polynomial time reduces*** to LINEAR-PROGRAMMING

Oracle Turing Machine and polynomial time reductions

- ”Oracle Turing machine with oracle B” means:
  - Turing machine with access to ”magical box” solving computational task B ***instantaneously.***
- ”A polynomial time reduces to B” means:
  - There is an oracle Turing machine with oracle B that solves A in polynomial time.

MATRIX-GAME polynomial time reduces to LINEAR-PROGRAM



Lemma

- If
  - A polynomial time reduces to B, and
  - B is polynomial time solvable,
- then
  - A is polynomial time solvable.

CORRELATED-EQUILIBRIUM is polynomial time solvable

- CORRELATED-EQUILIBRIUM:
  - Input: finite multi-player game in strategic form (i.e. as a table of payoffs)
  - Output: A correlated equilibrium (as an explicitly given probability distribution on outcomes)

- CORRELATED-EQUILIBRIUM polynomial time reduces to LINEAR-PROGRAM

Polynomial time equivalence

- If A polynomial time reduces to B and B polynomial time reduces to A then A and B are said to be polynomial time equivalent.

- This notion induces an ***equivalence relation*** on computational tasks.

- One equivalence class is the class of polynomial time solvable tasks.

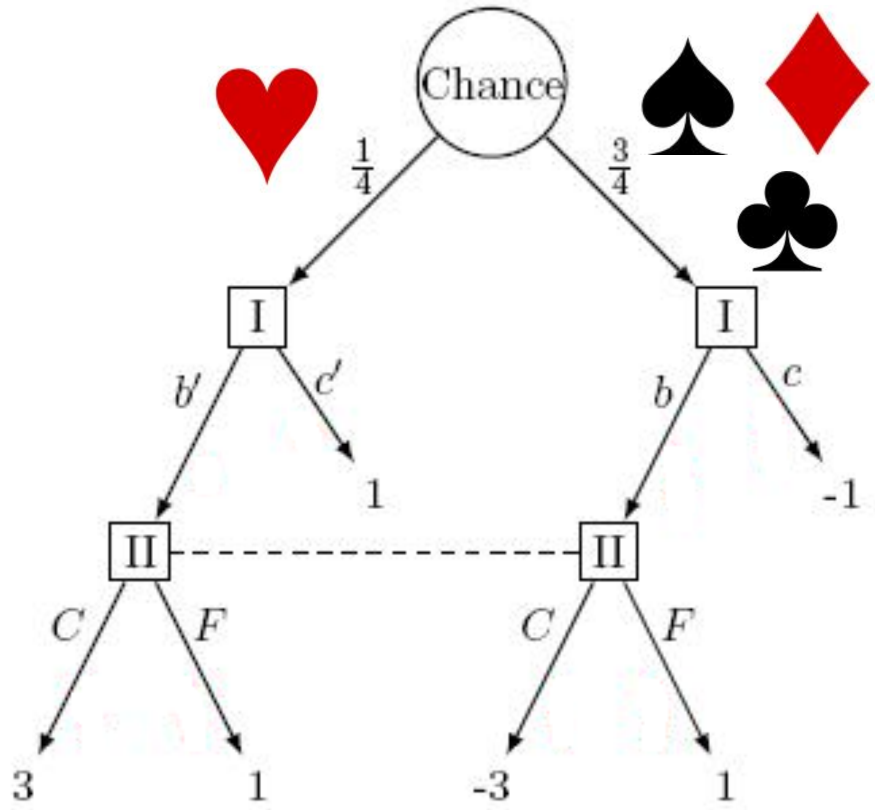
The story so far

- We defined and motivated the notions of ***polynomial time solvable computational task*** and ***polynomial time reduction***.

- We noted that ***solving matrix games*** and ***finding correlated equilibria*** polynomial time reduces to solving linear programs and that these tasks can therefore be solved in polynomial time.

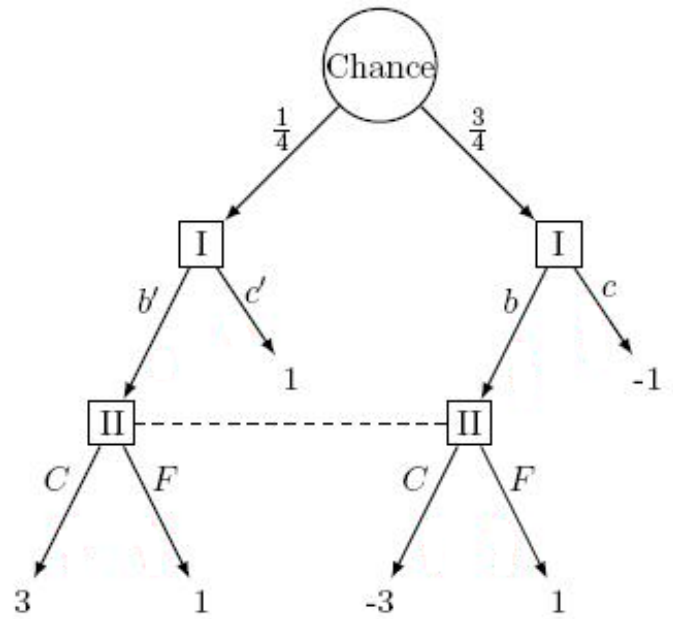
- Next: Solving two-player zero-sum ***extensive form*** games.

Extensive Form Game (2 player, 0-sum)



"Basic endgame of poker"

How to solve?



→

	C	F
b' b	-3/2	1
b' d	0	-1/2
c' b	-2	1
c' d	-1/2	-1/2

Textbook: Extensive form games can be converted into matrix games!

Why is this a silly conversion!?

Exponential blowup in size!

(100 inf.sets implies 2^{100} rows…)

Silly but correct algorithm solving matrix games

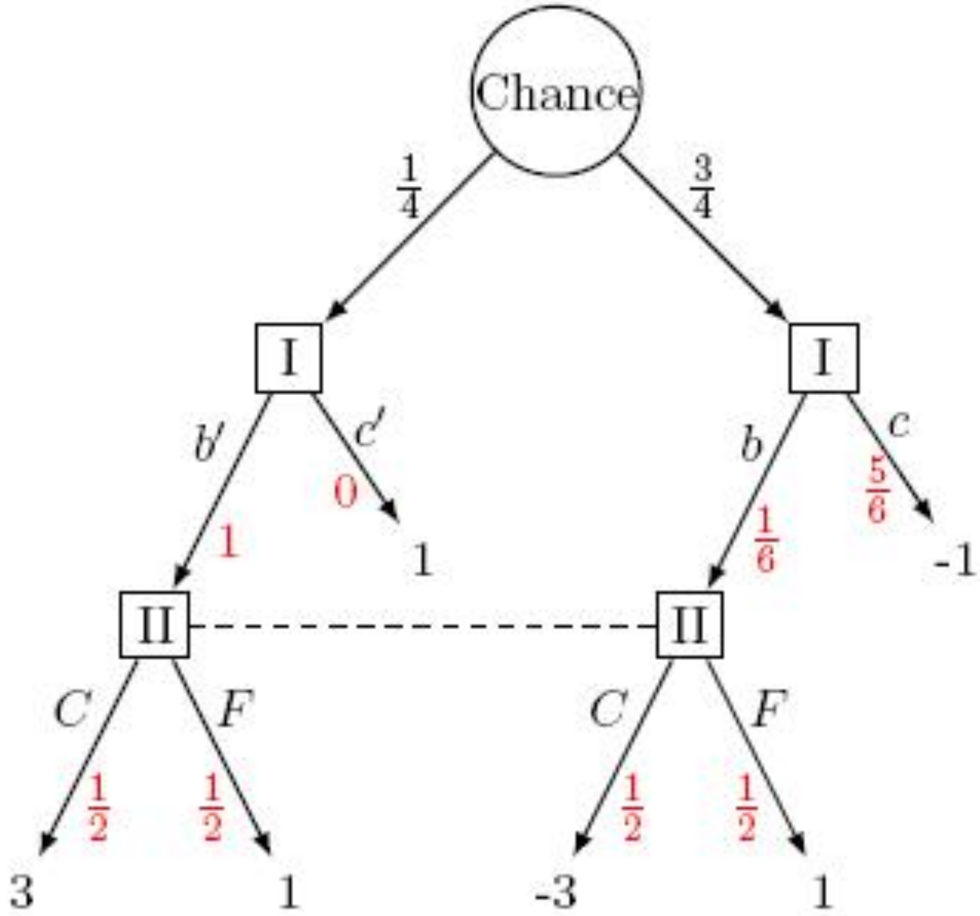
- If the input length is L, search through all strings of length at most $(10L)^{10}$.
- For each such string, check if it parses correctly into a number and two mixed strategies. Also check if these two strategies is a Nash equilibrium and that the equilibrium payoff is the number.
- The first string passing all these checks is given as output

Behavior strategies (Kuhn, 1952)

- A behavior strategy for a player is a family of probability distributions, one for each information set, the distribution being over the ***actions*** one can make there.

- For games of **perfect recall**, behavior strategies and mixed strategies are behaviorally equivalent.

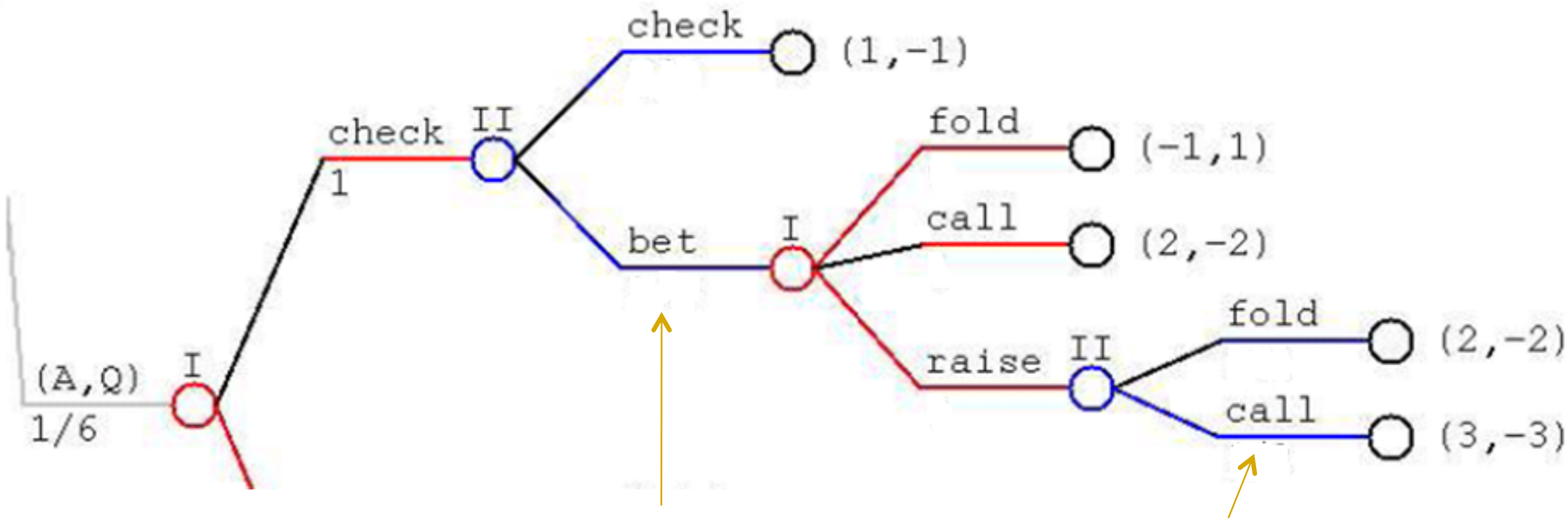
Behavior strategies (Kuhn, 1952)



Computational Task

- EXTENSIVE:
  - Input: 2-player, zero-sum extensive form game with perfect recall
  - Ouput: Its value, and optimal behavior strategies for both players.
- Can EXTENSIVE be solved in polynomial time?
- **Problem:** The optimal strategies are not described by a linear program in the behavior strategies!

Non-linearities……



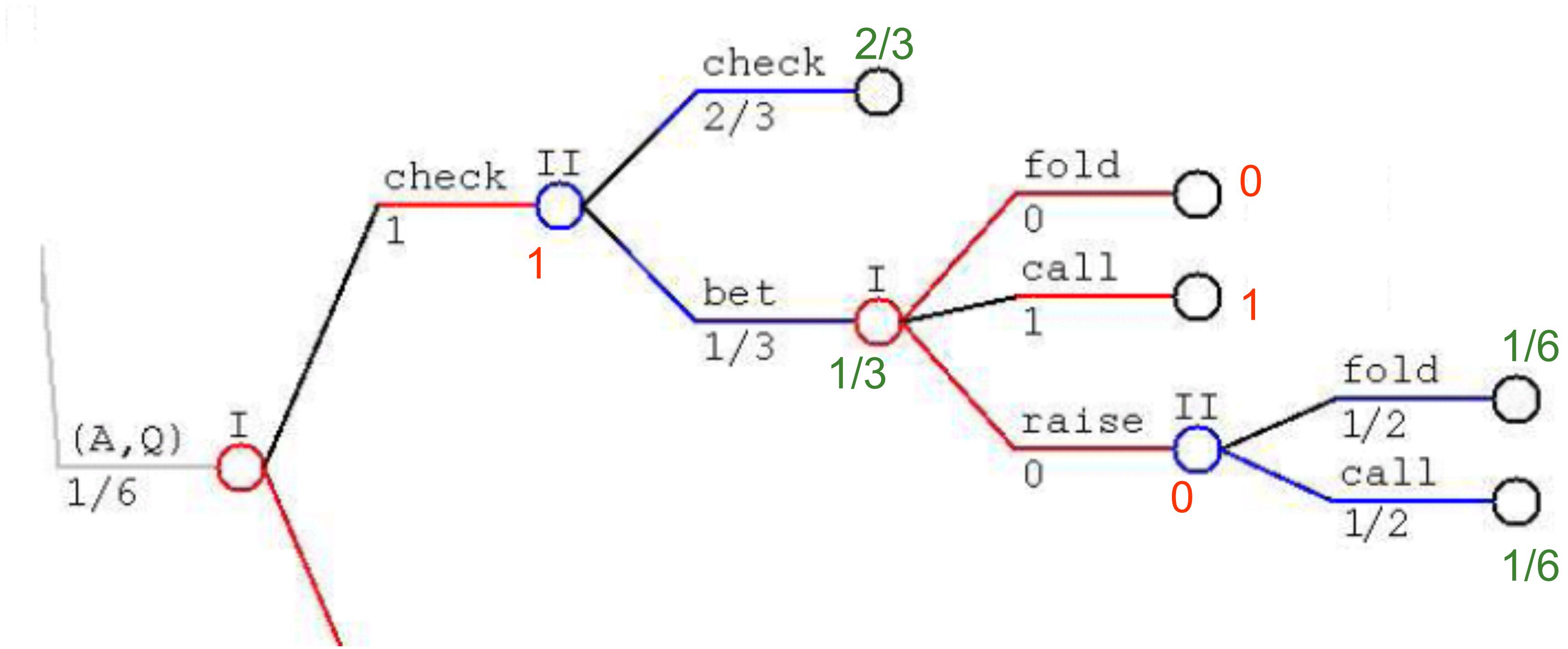
The ***product*** of the probability of betting and the probability of calling is a variable in the obvious mathematical program describing an optimal strategy

Realization plans (sequence form)
(Koller-Megiddo-von Stengel, 1994)

- Given a behavior strategy for a player, the **realization weight** of a ***sequence*** of moves is the **product** of probabilities assigned by the strategy to the moves in the sequence.

- If we have the realization weights for all sequences (a **realization plan**), we can deduce the corresponding behavior strategy (and vice versa).

Realization plans



(1,0,1,0,….) is a *realization plan* for Player I
(2/3, 1/3, 1/6, 1/6, …) is a *realization plan* for Player II

Crucial observation (Koller-Megiddo-von Stengel 1994)

- The set of valid realization plans for each of the two players (for games of perfect recall) is definable by a set of linear equations and positivity.
- The expected outcome of the game if Player 1 playing using realization plan $\mathbf{x}$ and Player 2 is playing using realization plan $\mathbf{y}$ is given by a bilinear form $\mathbf{x}^T A \mathbf{y}$.
- This implies that minimax realization plans can be found efficiently using linear programming!

Optimal response to fixed **x**.

- If Max' plan is ***fixed*** to x, the best response by Min is given by:

- Minimize $(\mathbf{x}^T A)\mathbf{y}$ so that $F\mathbf{y} = f$, $\mathbf{y} \geq 0$.

  ($F\mathbf{x} = f$, $\mathbf{y} \geq 0$ expressing that $\mathbf{y}$ is a realization plan.)

- The dual of this program is:

  Maximize $f^T \mathbf{q}$ so that $F^T \mathbf{q} \leq \mathbf{x}^T A$.

What should Max do?

- If Max plays **x** he should assume that Min plays a best reply so that he obtains the value given by
Maximize $f^T \mathbf{q}$ so that $F^T \mathbf{q}$ · $\mathbf{x}^T A$.

- Max wants to minimize this value, so his optimal strategy **y** is given by
Maximize $f^T\mathbf{q}$ so that $F^T \mathbf{q}$ · $\mathbf{x}^T A$, $E\mathbf{x} = e$, $\mathbf{x}$ ¸ 0.

(E**x** = e, **x** ¸ 0 expressing that **x** is a realization plan)

KMvS linear program

$$
\begin{aligned}
\max_{x,q} \quad & f^\top q \\
\text{s.t.} \quad & -A^\top x + F^\top q \le 0 \\
& Ex = e \\
& x \ge \mathbf{0}
\end{aligned}
$$

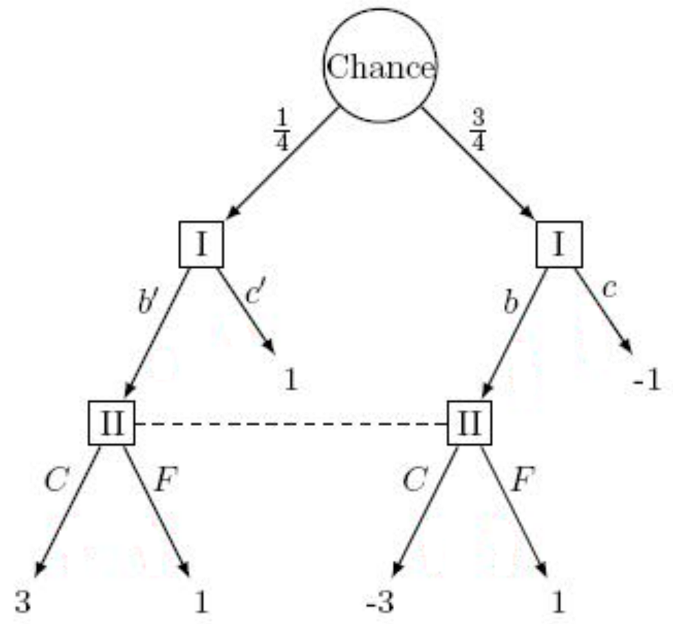
One constraint for each action (sequence) of player 2

x is valid realization plan

x – Realization plan for Player 1

q – a "value" for each information set of Player 2

Example



Variables:

$x_\epsilon, x_{b'}, x_{c'}, x_b, x_c$ (the realisation weights)
q_0 (the value)
q_h (representing the contribution to the value from plays through the information set owned by Player 2, h)

Program:

$\max q_0$

subject to

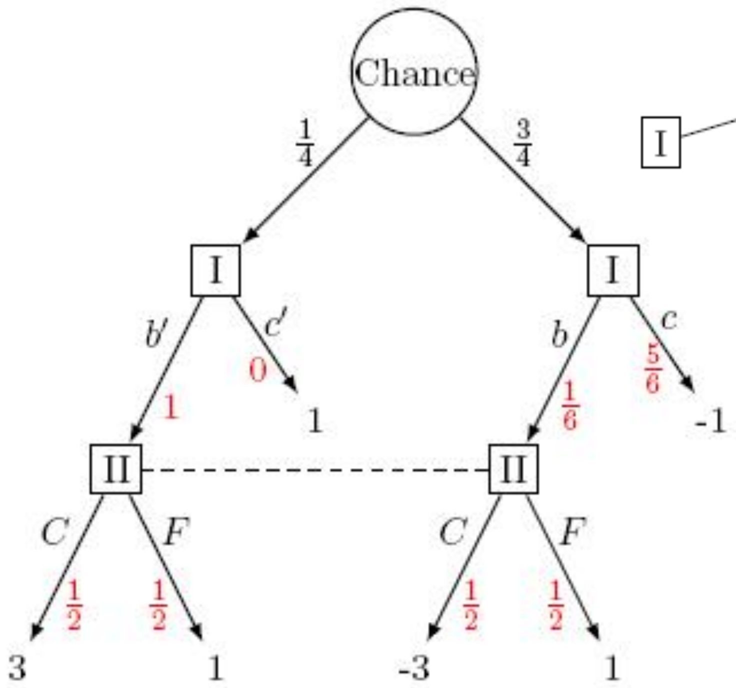
$$\begin{aligned}
\epsilon &: \quad q_0 \leq q_h + \tfrac{1}{4}x_{c'} - \tfrac{3}{4}x_c \\
C &: \quad q_h \leq \tfrac{3}{4}x_{b'} - \tfrac{9}{4}x_b \\
F &: \quad q_h \leq \tfrac{1}{4}x_{b'} + \tfrac{3}{4}x_b
\end{aligned}$$

$$\begin{aligned}
&x_\epsilon = 1 \\
&x_{b'} + x_{c'} = x_\epsilon \\
&x_b + x_c = x_\epsilon \\
&x_\epsilon, x_{b'}, x_{c'}, x_b, x_c \geq 0
\end{aligned}$$

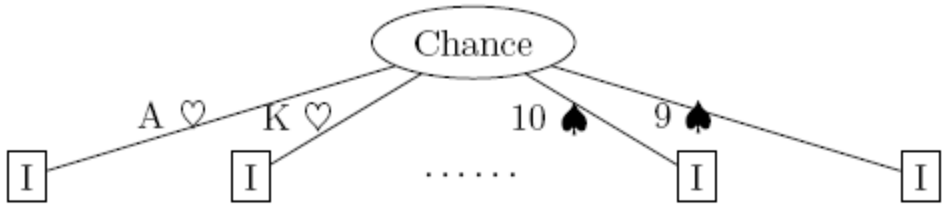
Solving extensive form games

- EXTENSIVE:
  - Input: 2-player, zero-sum, extensive form game with perfect recall
  - Ouput: Its value, and optimal behavior strategies for both players.
- EXTENSIVE can be solved in polynomial time by a reduction to linear programming.
- **Arguably, this method is a much more intuitive way of solving these games than the textbook method!**

Basic endgame of poker revisited



Unique optimal strategies



24-card deck

Several optimal strategies, some of them pure!

Finding pure optimal strategies

- PURE-EXTENSIVE:
  - Input: A 2-player extensive form game with perfect recall
  - Output: A pure optimal strategy if one exits.
- We do *not* believe that this task has a polynomial time algorithm.
- No such algorithm, ***unless P=NP***.

P and NP

- P is the class of ***decision problems*** that can be solved in polynomial time.

- Decision problem: For all inputs, the desired output is yes or no (but not both).

NP

- NP is the class of decision problems that can be solved by ***any algorithm of the following kind***:

- Let x be the input
- For each string y of length p(|x|):
  - If A(x,y) returns ”yes” then return ”yes”
- If no A(x,y) returns ”yes”, then return ”no”

Any polynomial p

Any polynomial time algorithm

NP Examples

- Let x be the input
- For each string y of length p(|x|):
  - If A(x,y) returns ”yes” then return ”yes”
- If no A(x,y) returns ”yes”, then return ”no”

- UCON: Given an undirected graph, is it connected?
- PEANO: Given a formal statement of Peano arithmetic, and a proof with ”blanks”, can the blanks be filled in to make the proof correct?

PEANO

- Input: $\neg(\exists n, a, b, c : n > 2 \wedge a^n = b^n + c^n)$

  #################.....################

- Output: ?

NP Examples

- Let x be the input
- For each string y of length $p(|x|)$:
  - If $A(x,y)$ returns "yes" then return "yes"
- If no $A(x,y)$ returns "yes", then return "no"

- UCON: Given an undirected graph, is it connected?
- PEANO: Given a formal statement of Peano arithmetic, and a proof with "blanks", can the blanks be filled in to make the proof correct?
- PURE-EXTENSIVE (modifed to just telling it a strategy exists).

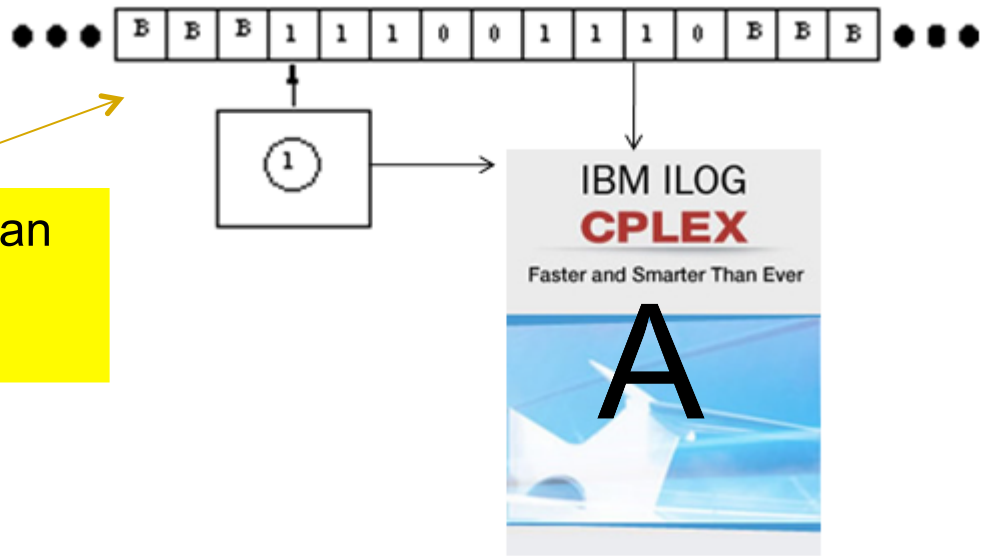
P vs. NP

- P is a subset of NP
- Is P = NP?
  - Seems very unlikely.
  - We do not know how to prove it.
- It is very useful to assume the statement **P is different from NP** as it has *great explanatory power*.
  - Like set theorists treat the continuum hypothesis or physicists treat ”laws of nature”.

NP-hard computational tasks

- A computational task A is **NP-hard** if all problems in NP polynomial time reduces to A.

Such a device can solve all of NP (e.g., PEANO)



- If an NP-hard task is polynomial time solvable, then P=NP

NP-complete computational tasks

- A decision problem A is NP-complete if
  - It is NP-hard
  - and itself in NP
- All NP-complete problems are polynomial time equivalent.
- An NP-complete problem is in P *if and only if* P=NP.
- Sounds nice, but are there any NP-complete problems?

The mother of all NP-complete problems

- NP is the class of decision problems that can be solved by any algorithm of the following kind:

Any polynomial p

- Let x be the input
- For each string y of length p(|x|):
  - If A(x,y) returns ”yes” then return ”yes”
- If no A(x,y) returns ”yes”, then return ”no”

Any polynomial time algorithm

The mother of all NP-complete problems

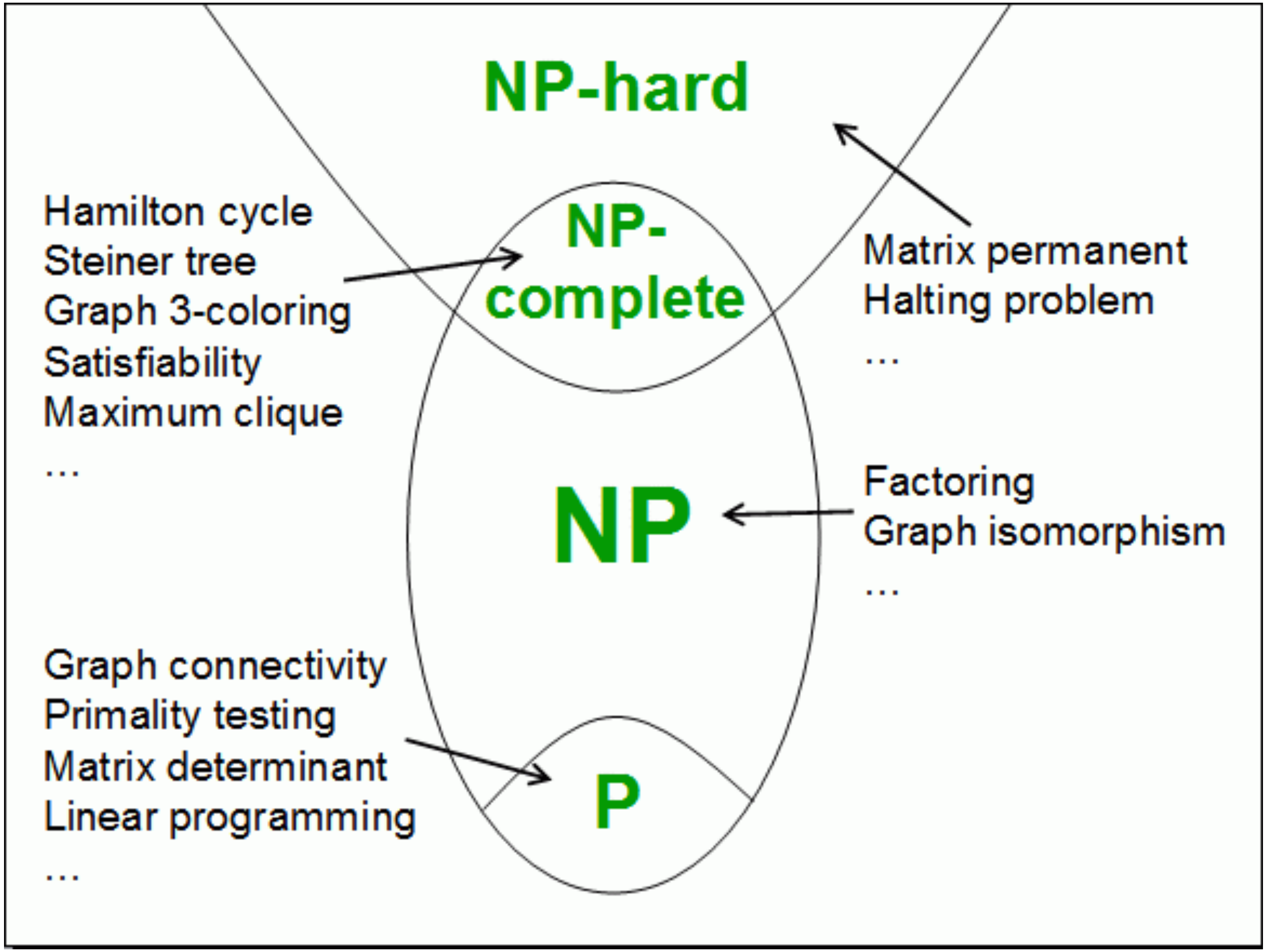
GENERIC-NP ...is NP-complete

E.g., as Turing machine, or C program..

- Let [x,z,A,q] be the input
- For each string y of length |z|:
  - If A(x,y) returns "yes" in |q| steps then return "yes"
- If no A(x,y) returns "yes", then return "no"

Cook (1972) and Karp (1973)

- Dozens of *natural* combinatorial problems are NP-complete, hence polynomial time equivalent to each other.
- Proofs use the fact that if A is NP-hard and A polynomial time reduces to B then B is NP-hard.
- Turing awards 1982, 1985.
- Since 1973, dozens have become tens of thousands…

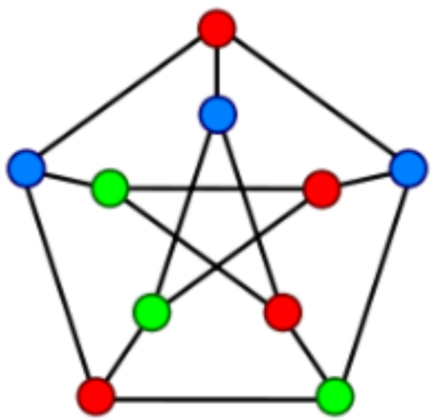

NP-hard
Hamilton cycle
Steiner tree
Graph 3-coloring
Satisfiability
Maximum clique
...
NP-
complete
Matrix permanent
Halting problem
...
NP
Factoring
Graph isomorphism
...
Graph connectivity
Primality testing
Matrix determinant
Linear programming
...
P

Examples from Cook and Karp

- PEANO (Cook)
- GRAPH-COLORING (Karp)
- INTEGER-LINEAR-PROGRAM (Karp)
- PARTITION (Karp)

GRAPH-COLORING

- Input: Finite undirected graph.
- Output: Can the vertices be colored red, blue, or green so that no adjacent vertices have the same color?

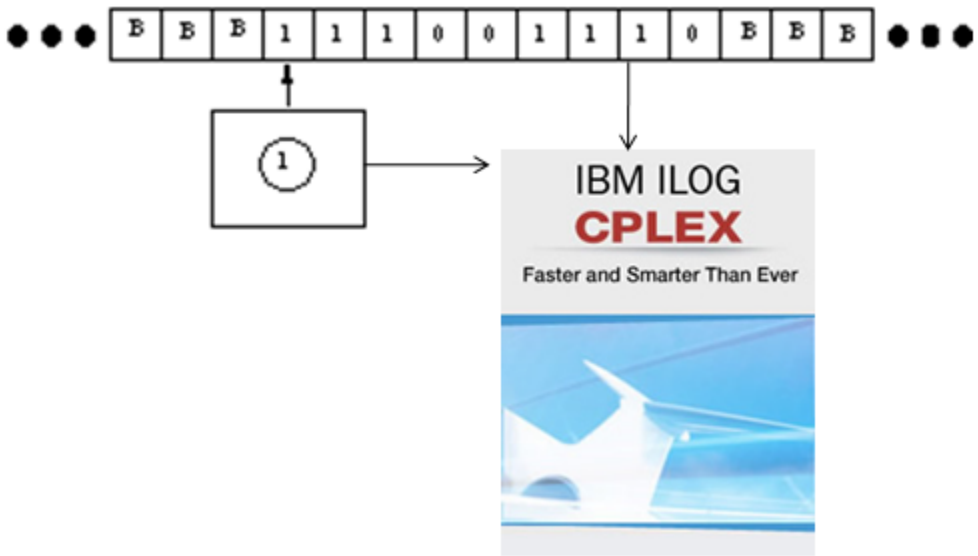


PARTITION

- Input: list of integers $a_1, a_2, .., a_m$
- Output: Can the integers be partitioned in two sets of same total sum?
- Example:
  - Input : 45, 32, 1, 19, 8, 15.
  - Output: Yes!
  - 45+15 = 32+1+19+8)

All being NP-complete, we have

- INTEGER-LINEAR-PROGRAM, PEANO, GRAPH-COLORING and PARTITION are polynomial time equivalent!



And also…

- PURE-EXTENSIVE:
  - Input: A 2-player extensive form game with perfect recall
  - Output: Does a pure optimal strategy exist?

- PURE-EXTENSIVE is NP-complete
  - Blair, Mutchler, van Lent 1996

- We show that PURE-EXTENSIVE is NP-hard by reducing a known NP-complete problem – PARTITION - to it.

PARTITION

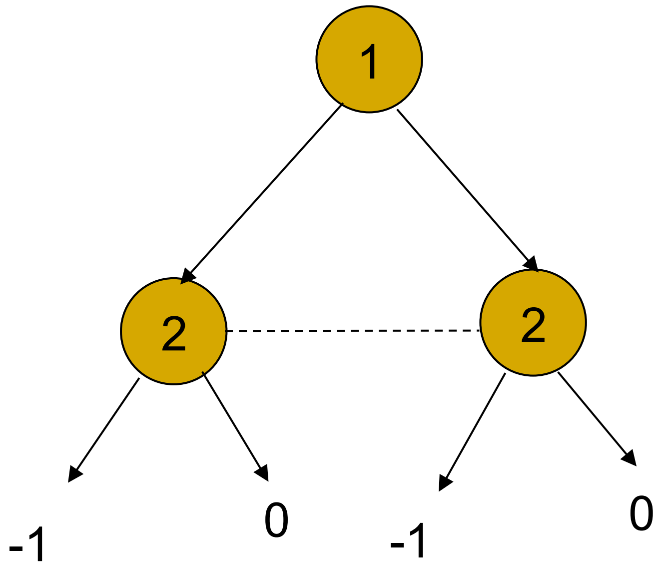
- Input: list of integers $a_1, a_2, .., a_m$
- Output: Can the integers be partitioned in two sets of same total sum?

Reducing PARTITION to PURE-EXTENSIVE

- Given 45, 32, 1, 19, 8, 15, I want to construct an extensive form game so that the maximizer has a pure optimal strategy if and only if the list can be perfectly partitioned.

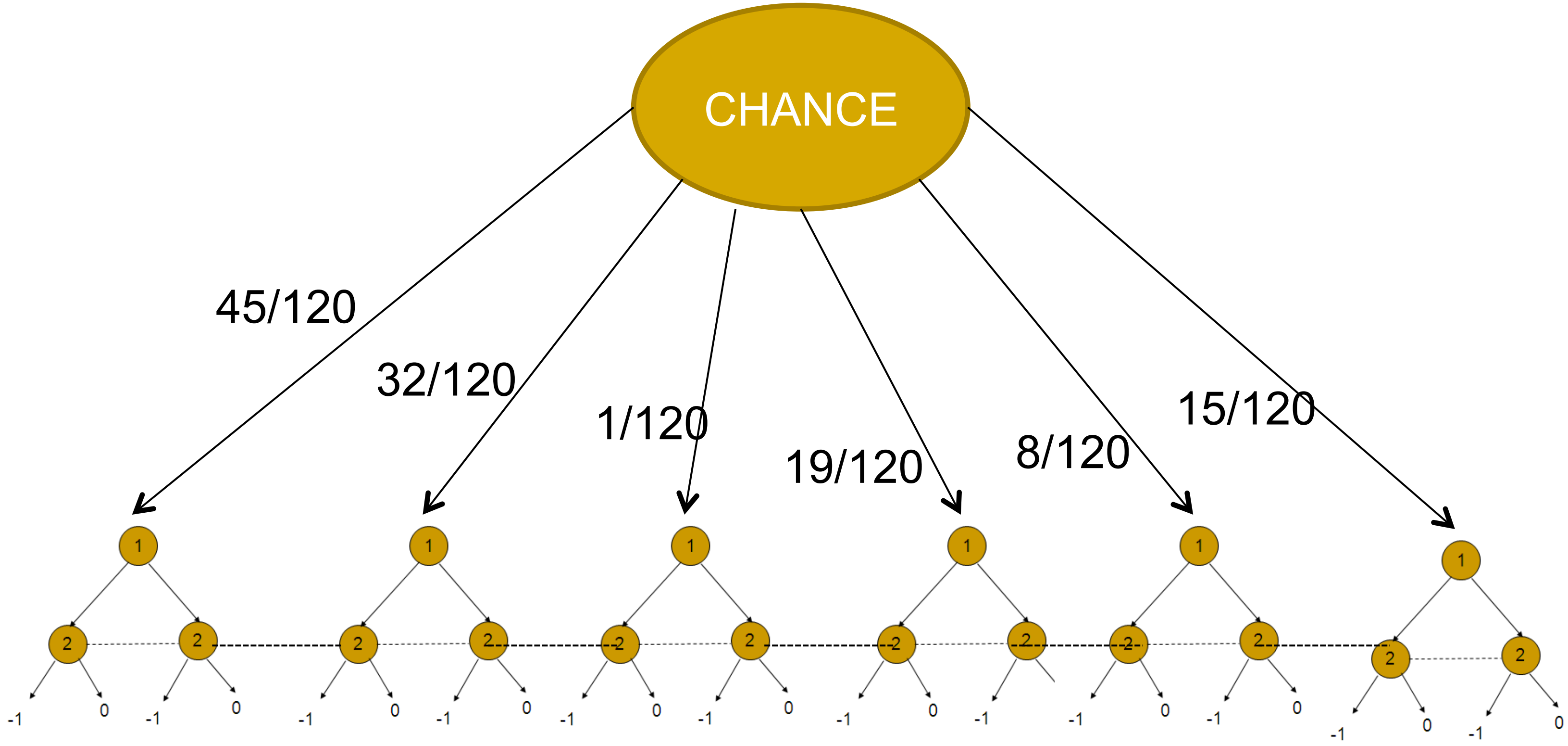
45,32,1,19,8,15

45,32,1,19,8,15



Matching Pennies

45,32,1,19,8,15



Player one has a pure optimal strategy if and only if the list has a balanced partition.

And also…

- PURE
  - I ct
  - r
-
- We show t E is NP-hard by reducing a known NP-co ete problem – PARTITION - to it.

Conceptual signficance:
Finding pure optimal
strategies in extensive form
games captures generic
exhaustive search.

The computational complexity of trembling hand perfection and other equilibrium refinements

Kristoffer Arnsfelt Hansen, Aarhus U.
Peter Bro Miltersen, Aarhus U.
Troels Bjerre Sørensen, U. Warwick

Equilibrium refinements

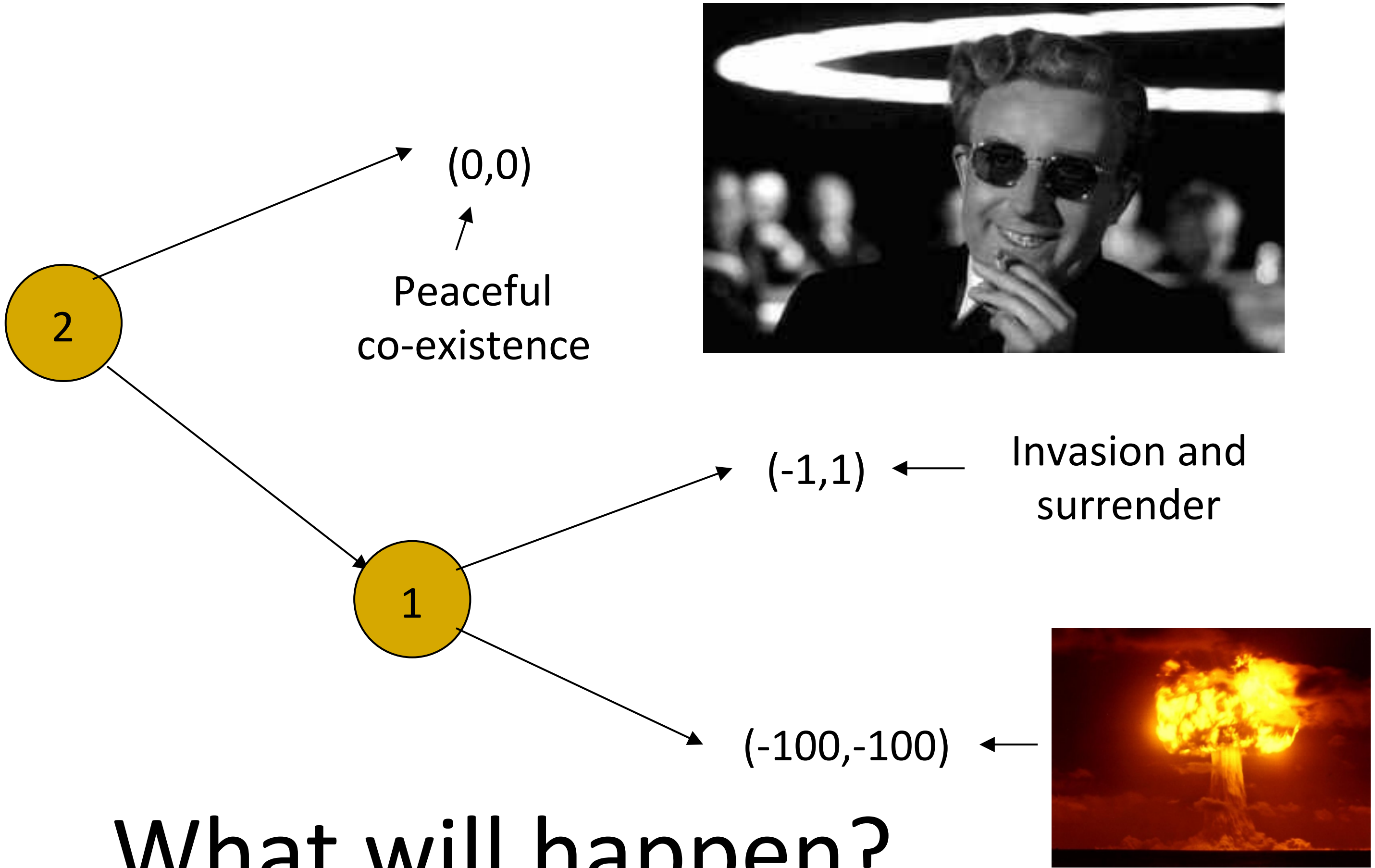
- Ideally, game theory, and the notion of Nash equilibrium can be used to make predictions about what will happen when a game is played.

- **Q**: When there is more than one Nash equilibrium in a game, how can we make predictions about what will happen when the game is played?

- **A**: We can try to rule out the more fishy ones…

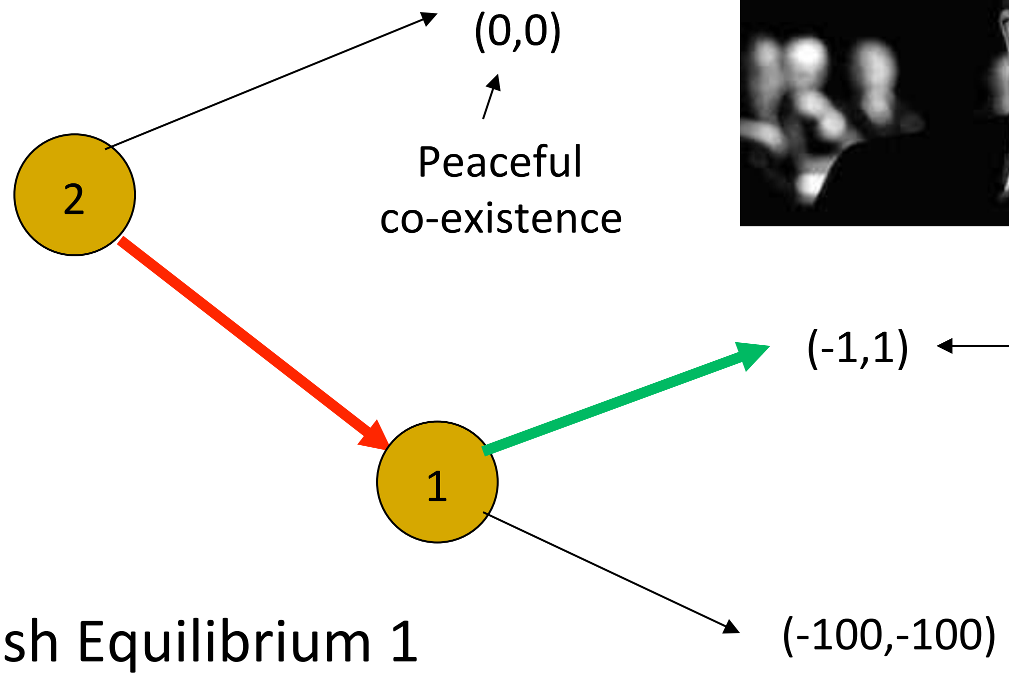
How often have I said to you that when you have eliminated the impossible, whatever remains, however improbable, must be the truth?

Doomsday Game



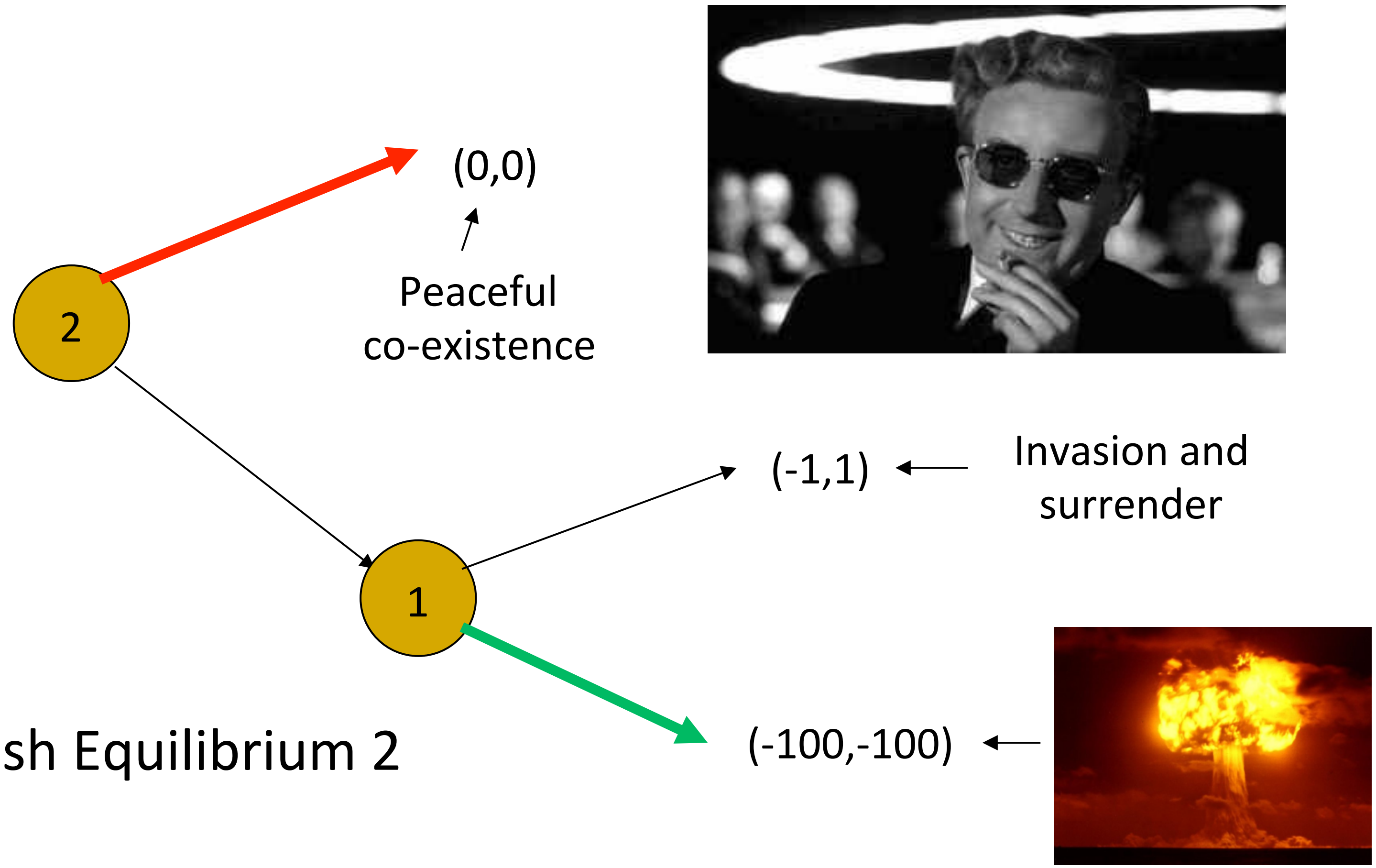
What will happen?

Doomsday Game



Nash Equilibrium 1

Doomsday Game



Nash Equilibrium 2

Doomsday Game

(-1,1)

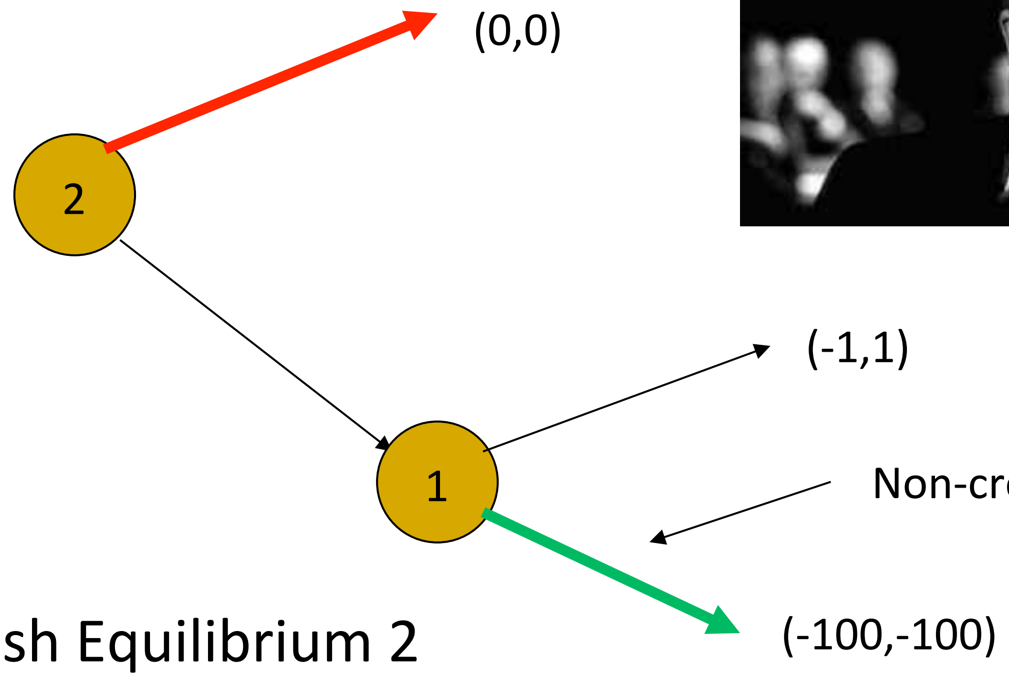
1

Non-credible threat

Nash Equilibrium 2

(-100,-100)

Doomsday Game



Nash Equilibrium 2
is not ***subgame perfect.***

Subgame perfection (Selten 1965)

- An equilibrium of an extensive form game (a.k.a. a game tree) is ***subgame perfect*** if it induces an equilibrium in all subgames.
- A *subgame* is a subtree that does not break any information sets.
- ***Nice for ruling out obviously bad behavior, but very much tied to the tree representation***

Doomsday game in normal form

	Attack	Stay at home	
Surrender-if-attacked	-1,1	0,0	0
Destroy-world-if-attacked	-100,-100	0,0	1
	0	1	

Doomsday game in normal form

	Attack	Stay at home	
Surrender-if-attacked	-1,1	0,0	0
Destroy-world-if-attacked	-100,-100	0,0	1
Tremble	ε	1-ε	

No matter how small ε is, row player must put all probability mass on Surrender-if-attacked to play a best response

Trembling hand perfection (Selten'75)

- ***Perturbed game***: For each available pure strategy i, associate a parameter $\varepsilon_i > 0$ (a tremble). Disallow probabilities smaller than this parameter for player i.

- A limit point of equilibria of perturbed games as largest pertubation parameter approaches 0 is an equilibrium of the original game and called ***trembling hand perfect.***

- **Intuition**: Think of trembles as infinitisimally small numbers.
  - formalised using non-standard analysis by Joe Halpern.
  - Formalised using formal polynomials in ε a.k.a lexicographic belief structures by Blume, Brandenburger, Dekel.

- Rules out some bad equilibrium than subgame perfection does not.

K Bogmærker Funktioner Hjælp

http://nobelprize.org/nobel_prizes/economics/laureates/1994/ | Nobel Prize Nash S

nger Dating ::. | Economics 1994 | Efficient String Matching through Wor...

bel Media | Nobel Museum | Nobel Peace Center | Nobel Web | HOME | CONTACT US | SEARCH

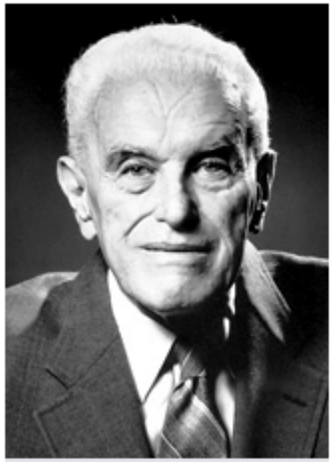
ze.org

LFRED NOBEL | PRIZE AWARDERS | NOMINATION | PRIZE ANNOUNCEMENTS | AWARD CEREMONIES | EDUCATIONAL GAMES

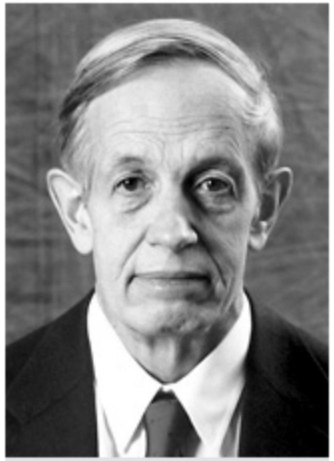
ze in Physics | Nobel Prize in Chemistry | Nobel Prize in Medicine | Nobel Prize in Literature | Nobel Peace Prize | **Prize in Economics**

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1994

"for their pioneering analysis of equilibria in the theory of non-cooperative games"



John C. Harsanyi



John F. Nash Jr.

Reinhard Selten

Printer Friendly
Comments & Questions
Tell a Friend

The 1994 Prize in:

Economics

Prev. year | Next year

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1994

Press Release
Presentation Speech

John C. Harsanyi

Autobiography
Prize Lecture
Banquet Speech

John F. Nash Jr.

Autobiography
Prize Seminar

Nobel Week 2008

Sunday, 7 December
5:30 p.m. CET

Nobel Lecture in Literature at the Swedish Academy.

Calendar

2008 Nobel Lectures - LIVE!

Try the Alfred Nobel Quiz!

Alternative explanation of why Destroy-world-if-attacked is not played

	Attack	Stay at home
Surrender-if-attacked	-1,1	0,0
Destroy-world-if-attacked	-100,-100	0,0

Destroy-world-if-attacked is ***weakly dominated*** by Surrender-if-attacked

Proposition (Selten '75)

- In a two-player game, a Nash equilibrium (s_1, s_2) is trembling hand perfect if and only if neither s_1 nor s_2 are weakly dominated (by mixtures).

- What about multi-player games?
  - No similar characterization known.
  - Game theorists tend to start from scratch constructing tremble structures to argue that equilibria in 3-player games are trembling hand perfect.

NP-hardness

It is **NP**-hard to decide if a given pure Nash equilibrium for a given 3-player game in normal form (i.e., as a table of payoffs) is trembling hand perfect.

Explains current practice in game theory (start from scratch for 3-player games) and discourages looking for a clean characterization as in the 2-player case.

NP-hardness

It is **NP**-hard to decide if a given pure Nash equilibrium for a given 3-player game in normal form (i.e., as a table of payoffs) is trembling hand perfect.

Also gives a ”computational critique” of the solution concept. Is it really reasonable that it is computationally intractable to check if a given profile satisfies the equilibrium condition?

NP-hardness

It is **NP**-hard to decide if a given pure Nash equilibrium for a given 3-player game in normal form (i.e., as a table of payoffs) is trembling hand perfect.

Proof arguably also sheds some doubts on the validity of the trembling hand concepts – even when applied to pure equilibria, some of the ”fishyness” of mixed Nash equilibria is inherited.

Proof

- GRAPH-COLORING
  - polynomial time reduces to
- THREE-PLAYER-UPPER-VALUE
  - polynomial time reduces to
- TREMBLING-HAND

Minmax of *Three*-player zero-sum games

Player 1:
Max, the
Maximizer

Players 2 and 3:
Min and Miney, the
Minimizers

"honest-but-married"

Maxmin value (lower value, security value):

$$\underline{v} = \max_{x \in \Delta(S_1)} \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} u_1(x, y, z)$$

Minmax value (upper value, threat value):

$$\bar{v} = \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} \max_{x \in \Delta(S_1)} u_1(x, y, z)$$

Uncorrelated mixed strategies.

Minmax of *Three*-player zero-sum games

Player 1:
Max, the
Maximizer

Players 2 and 3:
Min and Miney, the
Minimizers

"honest-but-married"

Maxmin value (lower value, security value):

$$\underline{v} = \max_{x \in \Delta(S_1)} \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} u_1(x, y, z)$$

Minmax value (upper value, threat value):

$$\bar{v} = \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} \max_{x \in \Delta(S_1)} u_1(x, y, z)$$

Bad news:

- **Lower value · upper value but in general not =**
- **Maxmin/Minmax not necessarily Nash**
- **Minmax value may be irrational**

Computable in **P**, given table of u_1

equality?

Maxmin value (lower value, security value):

$$\underline{v} = \max_{x \in \Delta(S_1)} \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} u_1(x,y,z)$$
$$= \max_{x \in \Delta(S_1)} \min_{(y,z) \in S_2 \times S_3} u_1(x,y,z)$$
$$= \min_{(y,z) \in \Delta(S_2 \times S_3)} \max_{x \in \Delta(S_1)} u_1(x,y,z)$$

Correlated mixed strategy (married-and-dishonest!)

Minmax value (upper value, threat value):

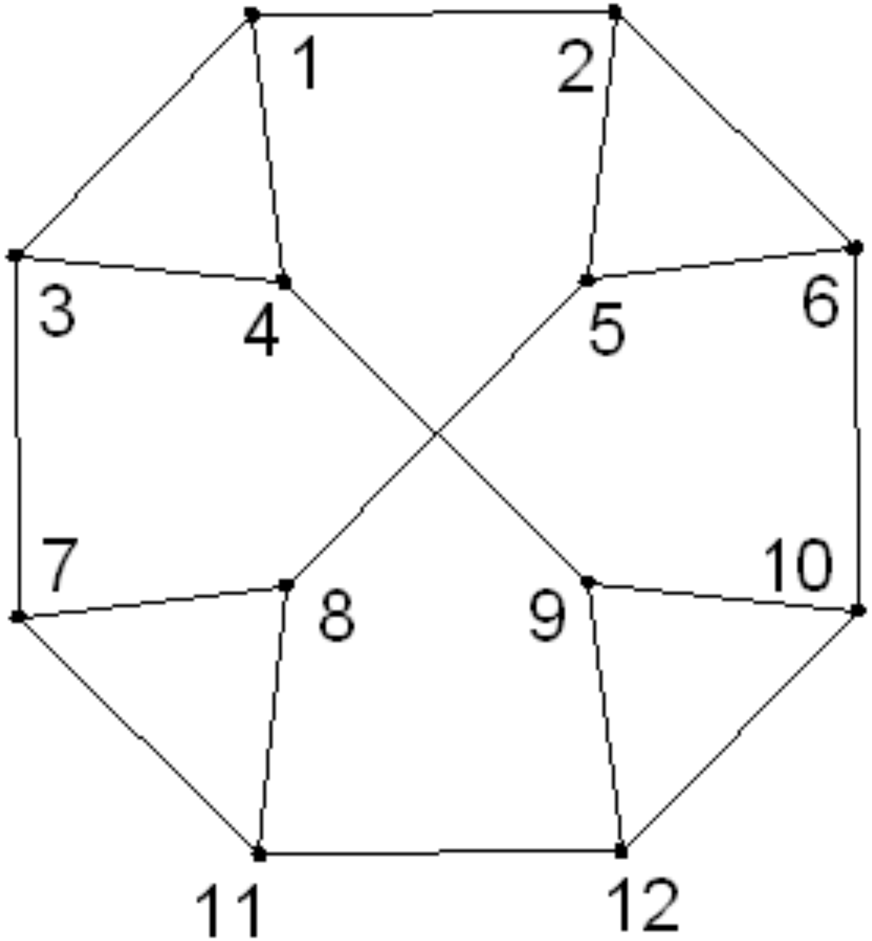
$$\bar{v} = \min_{(y,z) \in \Delta(S_2) \times \Delta(S_3)} \max_{x \in \Delta(S_1)} u_1(x,y,z)$$

Borgs *et al.*, STOC 2008:
NP-hard to approximate, given table of u_1!

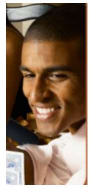
Borgs, Chayes, Immorlica, Kalai, Mirrokni, Papadimitriou, 2008

It is **NP**-hard to approximate the minmax-value of a 3-player n x n x n game with payoffs 0,1 within inverse polynomial additive error.

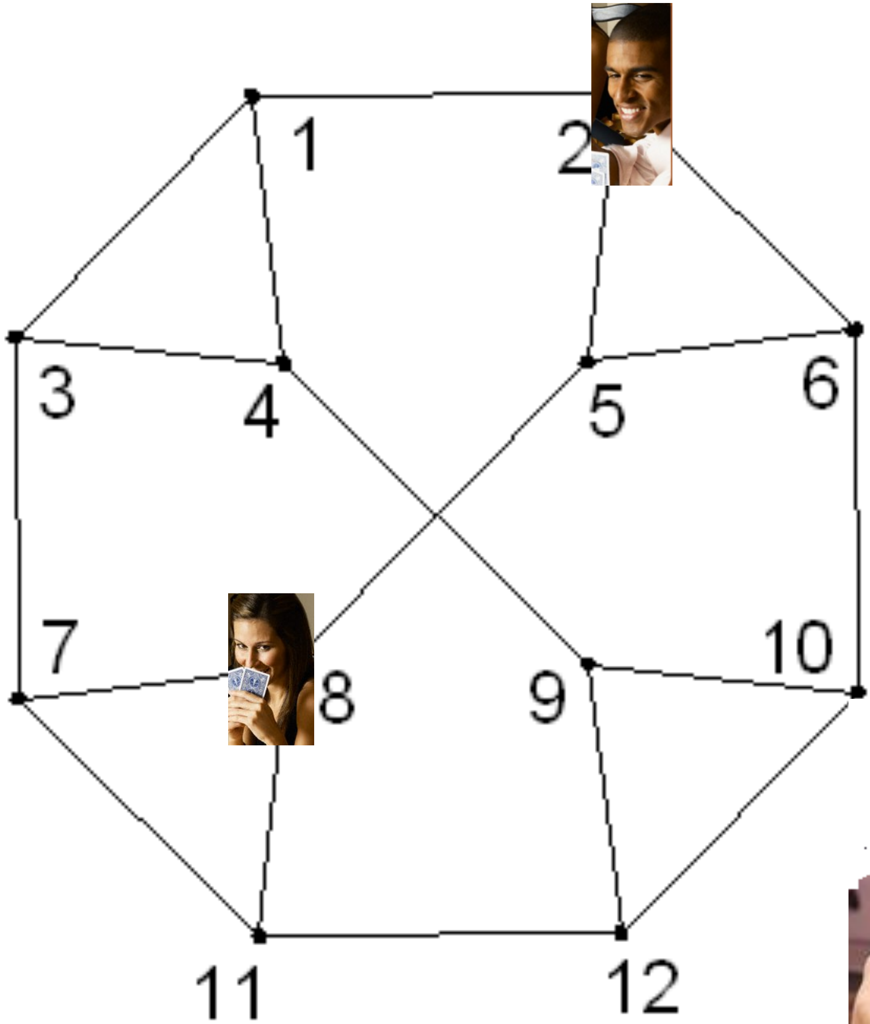
Proof – Hide and seek game



Min and Miney hide in an undirected graph.



Proof – Hide and seek game



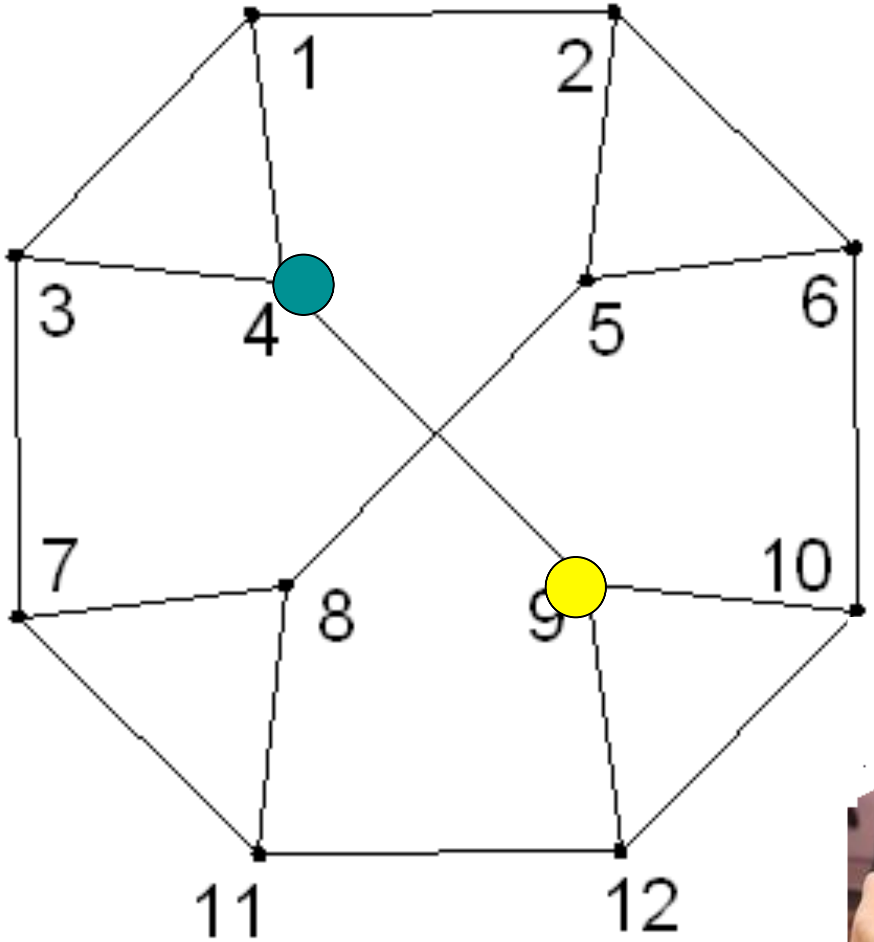
Min and Miney hide in an undirected graph.

Max, blindfolded, has to call the location of one of them.

Analysis

- Optimal strategy for Max?
  - Call arbitrary player at random vertex.
- Optimal strategy for Min and Miney?
  - Hide at random vertex
- Lower value = upper value = $1/n$.

Hide and seek game with colors

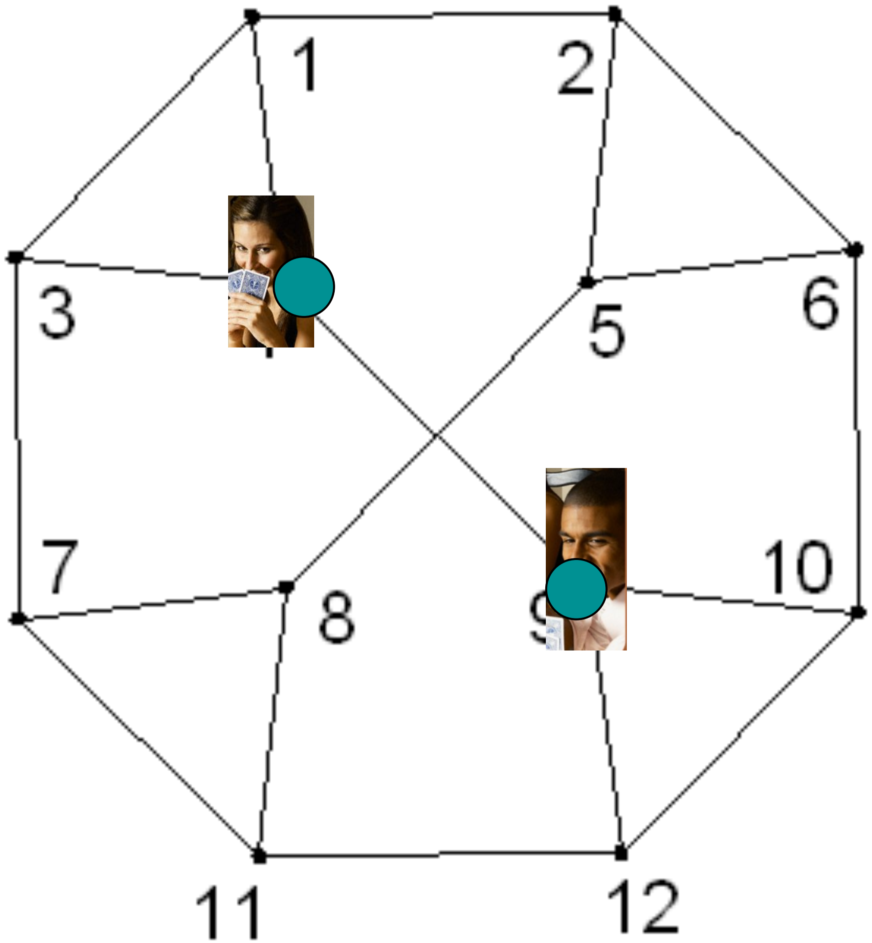


Min and Miney hide in an undirected graph.

.. and declare a color in

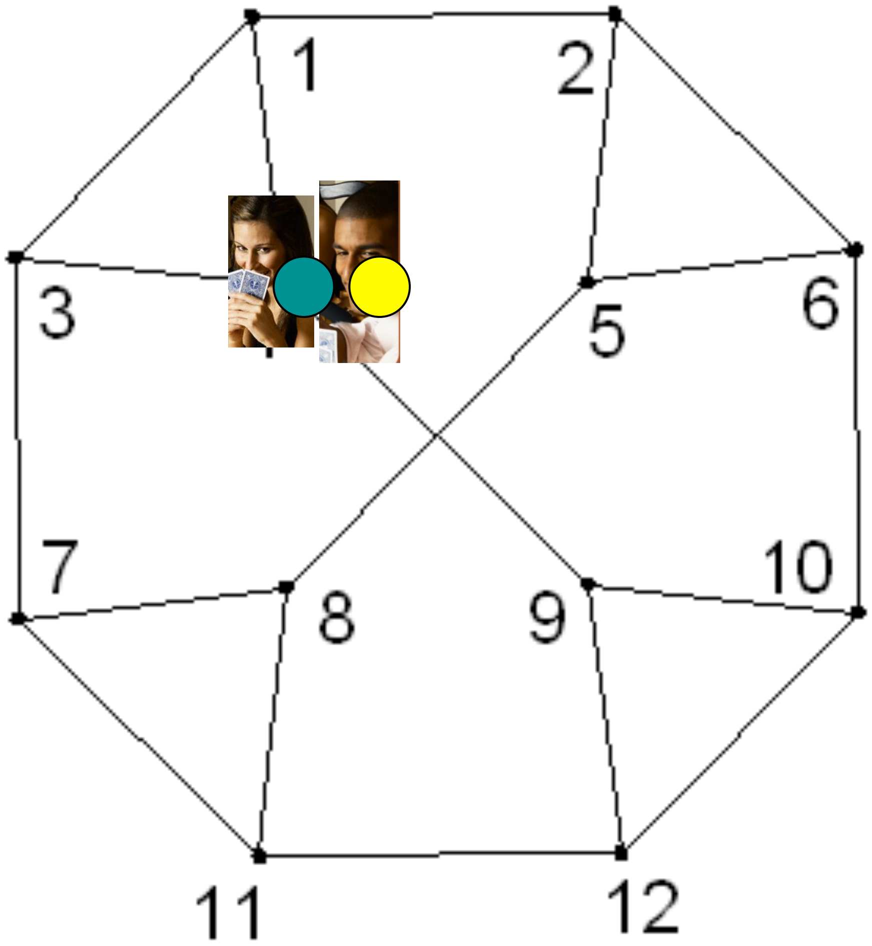
Max, blindfolded, has to call the location of one of them.

Hide and seek game with colors



Additional way in which Max may win: Min and Miney make declarations inconsistent with 3-coloring.

Hide and seek game with colors



Additional way in which Max may win: Min and Miny make declarations inconsistent with 3-coloring.

Analysis

- If graph is 3-colorable, minmax value is $1/n$: Min and Miney can play as before.

- If graph is not 3-colorable, minmax value is at least $1/n + 1/(3n^2)$.

Reduction to deciding trembling hand perfection

- Given a 3-player game G, consider the task of determining if the minmax of Player 1 value is strictly bigger than ® or strictly smaller than ® (we are promised that one of the two cases hold).
- Define G* by augmenting the strategy space of each player with a new strategy ABSTAIN.
- Payoffs of G* :
  - Players 2 and 3 get 0, no matter what is played.
  - Player 1 gets ® if at least one player plays ABSTAIN, otherwise he gets what he gets in G.
- **Claim:** ALL-ABSTAIN is trembling hand perfect in G* *if and only if* the minmax value of G is strickly smaller than ®.

Intuition

- If the minmax value of G is strictly less than ®, ALL-ABSTAIN is trembling hand perfect in G*. Why?
  - Player 2 and Player 3 are happy no matter what.
  - Player 1 may believe that when playing ALL-ABSTAIN, Players 2 and 3 may tremble and play exactly their minmax strategy.
  - He is currently playing a best response to this, since all his replies in G are strictly worse.
- If the minmax value is strictly greater than ®, ALL-ABSTAIN is **not** trembling hand perfect. Why?
  - No matter which theory about how Players 2 and 3 independently tremble that Player 1 entertains, he is not currently playing a best reply: He can achieve something better than ® by playing in G rather than playing ABSTAIN.

Extensions

- **Proper equilbrium** (Myerson'78) is NP-hard
  - Open: Is the case of two players easy?
- **Sequential equilibrium** (Kreps and Wilson '82) is NP-hard
  - Only for "strategy part" of sequential equilibrium. Open: What if an entire assessment is given?
  - Open: Is the case of two players easy?
- **Quasi-perfect equilibrium** (van Damme '84) is NP-hard
  - Is the case of two players easy?

Did we nail equivalence class of TREMBLING-HAND yet?

Is deciding trembling hand perfection in NP (and hence NP-complete)?

	Attack	Stay at home	
Surrender-if-attacked	-1,1	0,0	1
Destory-world-if-attacked	-100,-100	0,0	0
	1	0	

What kind of structure would verify that Attack-and-Surrender is trembling hand perfect?

Is deciding trembling hand perfection in NP (and hence NP-complete)?

	Attack	Stay at home	
Surrender-if-attacked	-1,1	0,0	1-ε
Destory-world-if-attacked	-100,-100	0,0	ε
	1-ε	ε	

We can use formal polynomials that describe the relative magnitude of trembles (says Blume, Brandenburger and Dekel) and verify the best-repsonse conditions

But can we keep the bitsize under control?

SQRT-SUM hardness

- The SQRT-SUM problem:

  Given $a_1, a_2, \ldots, a_n, k$, is $\sum (a_i)^{1/2} < k$?
- Not known to be in NP or even the polynomial hierarchy.
  - Which is why we do ***not*** know Euclidean TSP to be NP-complete
  - (we only know Euclidean TSP to be NP-hard).
- Pioneered as hardness notion in computational game theory by Etessami and Yannakakis '05.
  - A problem is said to be SQRT-SUM hard if the SQRT-SUM problem reduces to it.

Applying SQRT-SUM-hardness to trembling hand perfection

- Comparing the Minmax value of a 3-player game in normal form to a given rational number is SQRT-SUM hard.

- Corollary: Deciding if a given pure strategy Nash equilibrium in a 3-player game is trembling hand perfect is SQRT-SUM hard and hence not in NP unless SQRT-SUM is in NP.

Minmax values in 3-player games are SQRT-SUM hard

Picks i in {1,2,..n}

Picks j in {1,2,..n}

Picks k in {1,2,..n}

- Max loses $1/a_i$ if $i=j=k$
- The minmax value is $-1/(\sum (a_i)^{1/2})^2$

The Story so far

- Many tasks in computational game theory, solving matrix games, solving 2-player 0-sum extensive form games, or finding correlated equilibria can be solved in polynomial time by reducing them to linear programming.

- Others are NP-hard, such as checking if a 2-player 0-sum game has a pure optimal strategy or checking if a pure strategy profile is a trembling hand perfect equilibrium. The first is NP-complete, the latter may be even harder.

- Rest of today: ***What about (mixed) Nash equilibria?***

Finding Nash Equilibria

- 2P-NASH
  - Input: A finite 2-player game in strategic form with rational payoffs.
  - Output: A Nash equilibrium of the game

Finding Nash Equilibria

- ~~NASH:~~
  - ~~Input: A multi-player game in normal form with rational payoffs.~~
  - ~~Output: A Nash equilibrium of the game~~
- The unique legal output may be irrational valued!

Finding approximate Nash equilibria

- APPROXIMATE-NASH:
  - Input: A multi-player game in normal form with rational payoffs and ε.
  - Output: An ε-Nash equilibrium of the game

- Fair substitute?
- A rather unsatisfactory notion to people who care about infinitisimals! As we do!
- More discussion later!

Solving 2-player Nash: The Lemke-Howson algorithm

- Lemke and Howson (1964).
- A reduction to a special case of ***linear complementarity programming*** (LCP).
- Linear complementarity program =
  - Linear program with non-negative variables x,y ***plus***
  - A complementarity constaint $x^T y = 0$
- In this reduction, the complementarity constraint captures that in equilibrium for each pure strategy j, ***either***
  - j is played with 0 probability, ***or***
  - j is a best reply, or equivalently, the loss from playing j instead of a best reply is 0.

The algorithm

- No general good algorithm for solving LCPs
  - In fact, the general problem is NP-hard.

- The Lemke-Howson algorithm solves the special case that arises from the Nash equillibrium problem by iterated ***pivoting*** exactly as the simplex algorithm solves linear programs.

Facts about Lemke-Howson

- Important fact for later:
  - As in the case of the simplex algorithm, the Lemke-Howson algorithm follows a piecewise linear path in Euclidean space.
  - Unlike the case of the simplex algorithm, the path can be ”locally traced backwards” – the pivoting is reversible.
- The Lemke-Howson algorithm is not polynomial time (Savani and von Stengel, 2004)
- Interesting fact just for now: We know that finite 2-player games have rational equilibria ***because of*** the Lemke-Howson algorithm.

Many-player approximate Nash: Scarf's algorithm (1967)

- Step 1: Finding approximate Nash equilibria ***polynomial time reduces*** to finding approximate Brouwer fixed points of continuous maps.
- Essentially shown by
- But you should be su

Rules for computational problems

- **Input and output should be bit strings**
  - Computer science models computation by digital computers and bit strings are all digital computers can store.
  - ***A large part of the power of the theory comes from this fact.***

How to input "a continuous map" as a bit string?

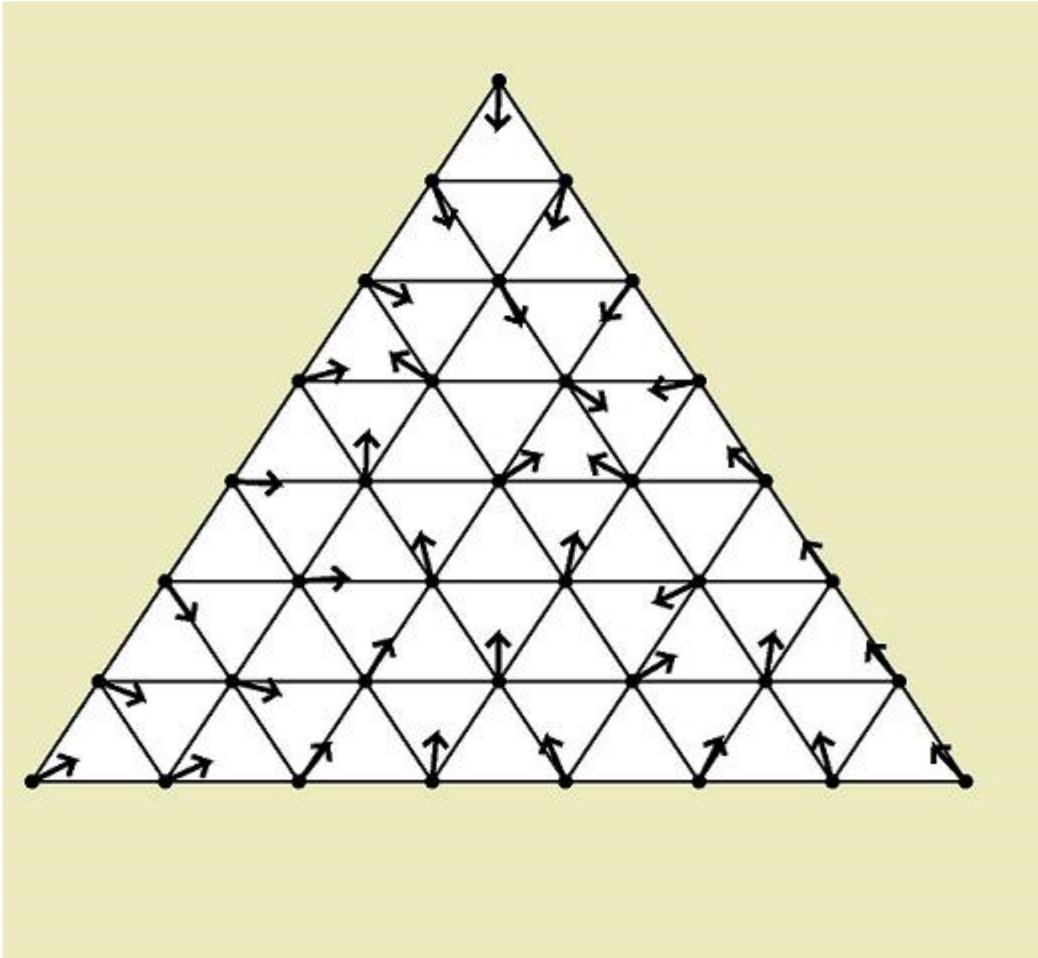
Finding approximate Brouwer fixed points

- BROUWER-TURING
  - Input: $\epsilon > 0$ and $f: [0,1]^n \rightarrow [0,1]^n$, given as a ***Turing machine*** that maps rational approximations of x to rational approximations of f(x).
  - Output: x^*, so that $|x^* - f(x^*)| \leq \epsilon$.

Finding approximate Brouwer fixed points

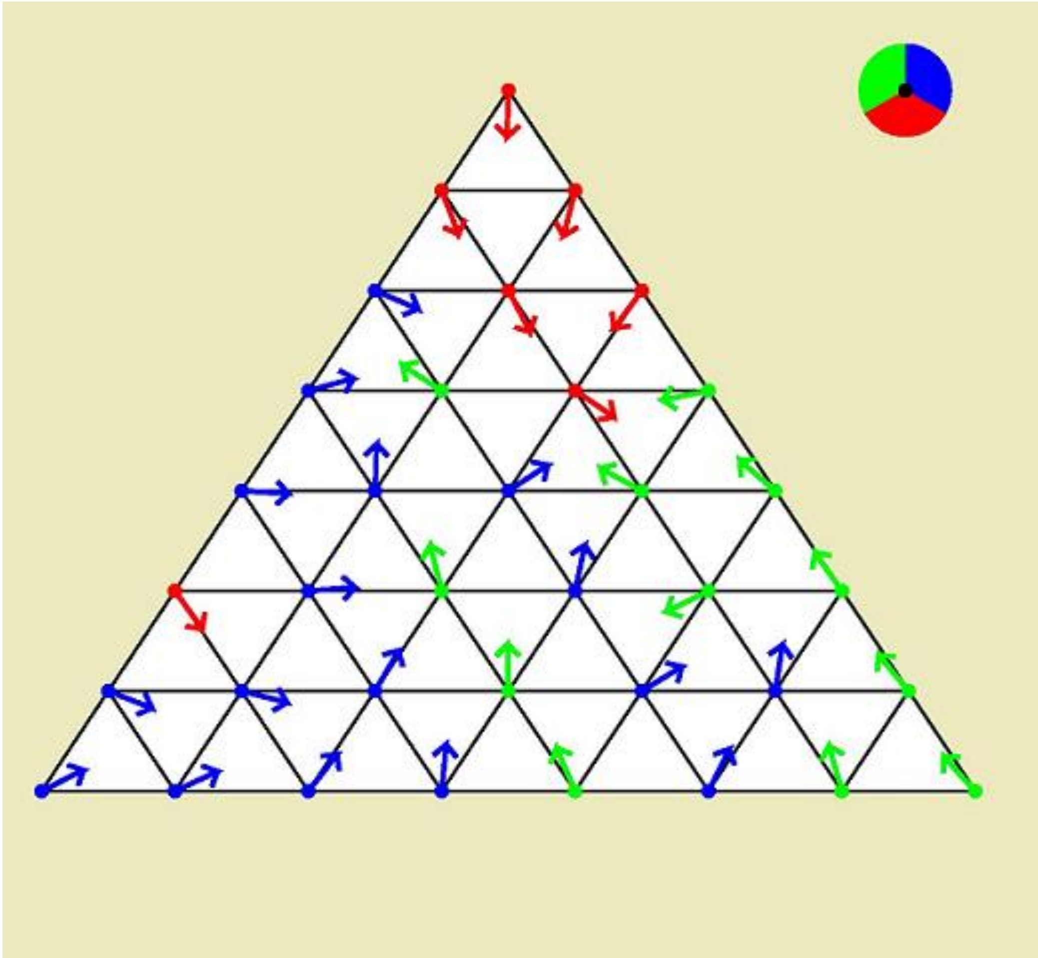
- BROUWER-FORMULA
  - Input: ² > 0 and f: $[0,1]^n$! $[0,1]^n$, given as an ***expression*** involving +,-,*,/,max,min that computes f.
  - Output: x*, so that |x* – f(x*)| · ².
- Note that BROUWER-FORMULA seems much less general.
  - Certainly, BROUWER-FORMULA polynomial time reduces to BROUWER-TURING.

Solving BROUWER



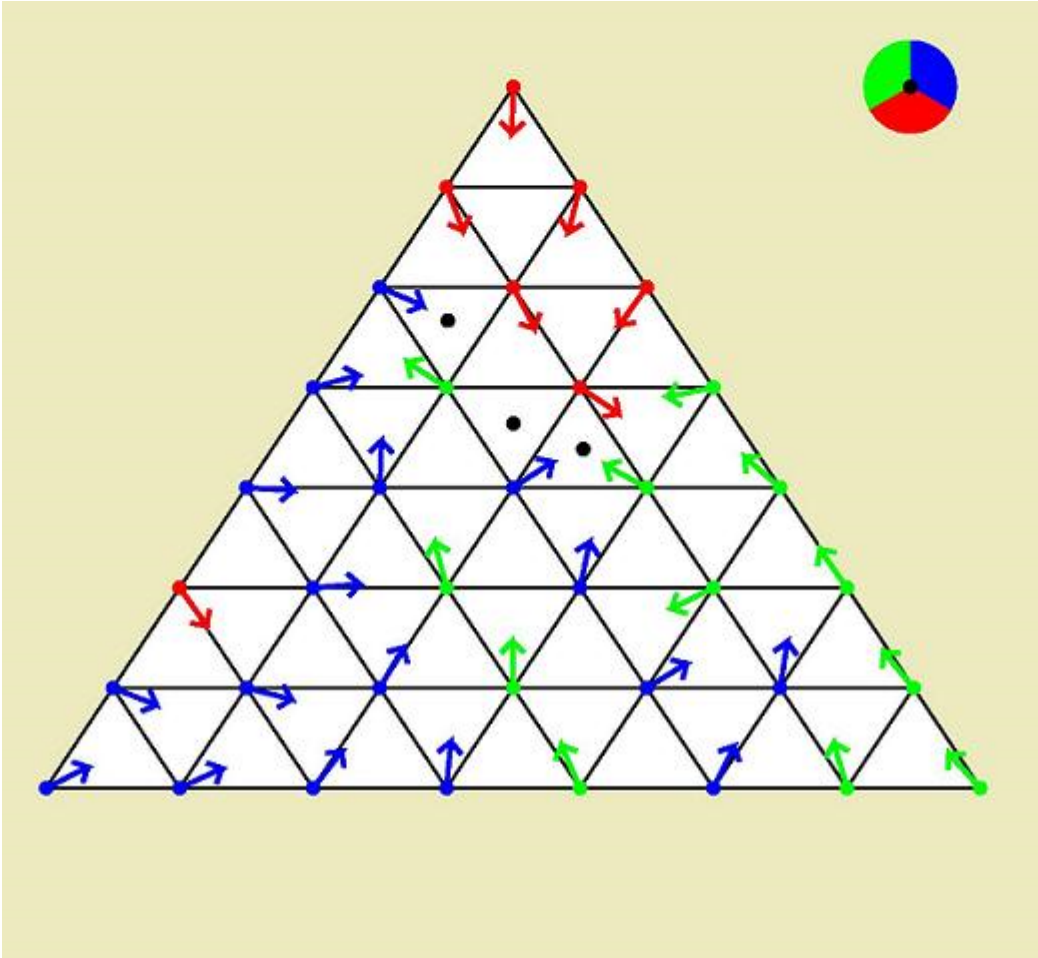
Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Solving BROUWER



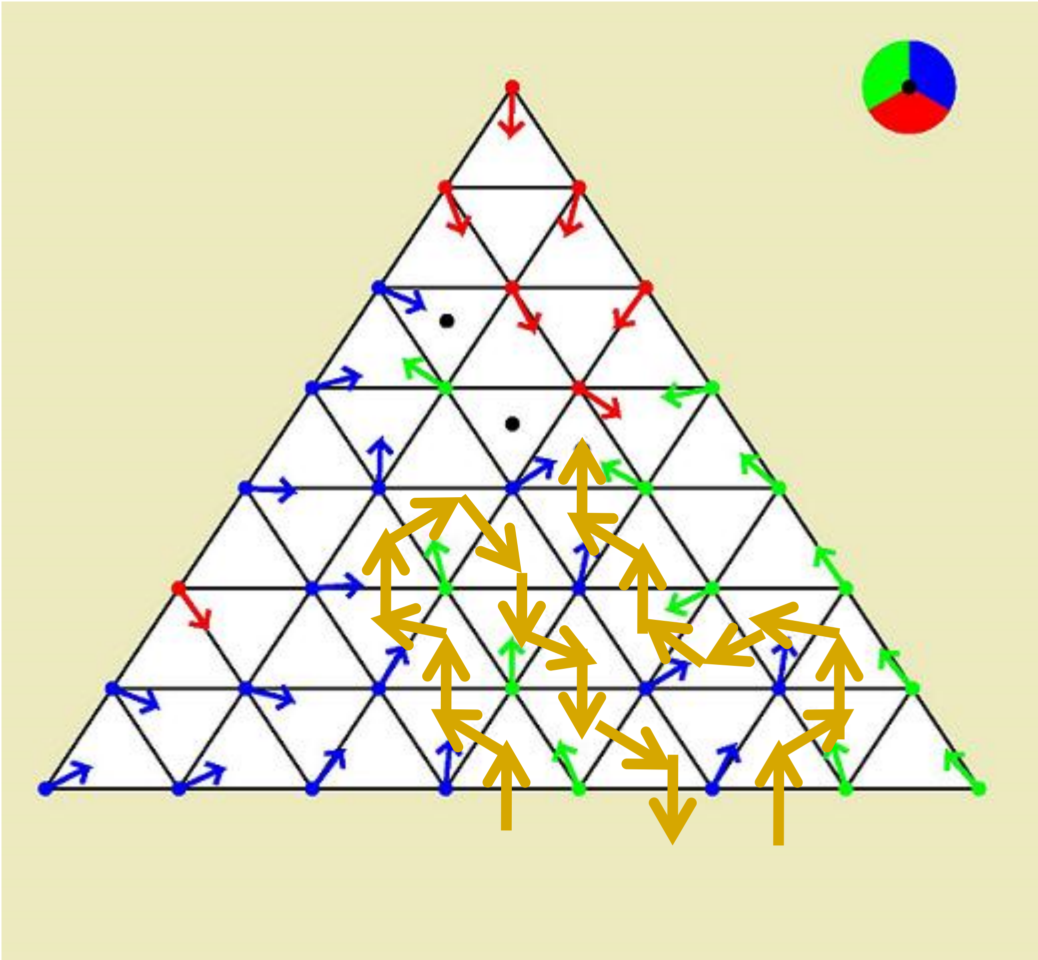
Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Solving BROUWER



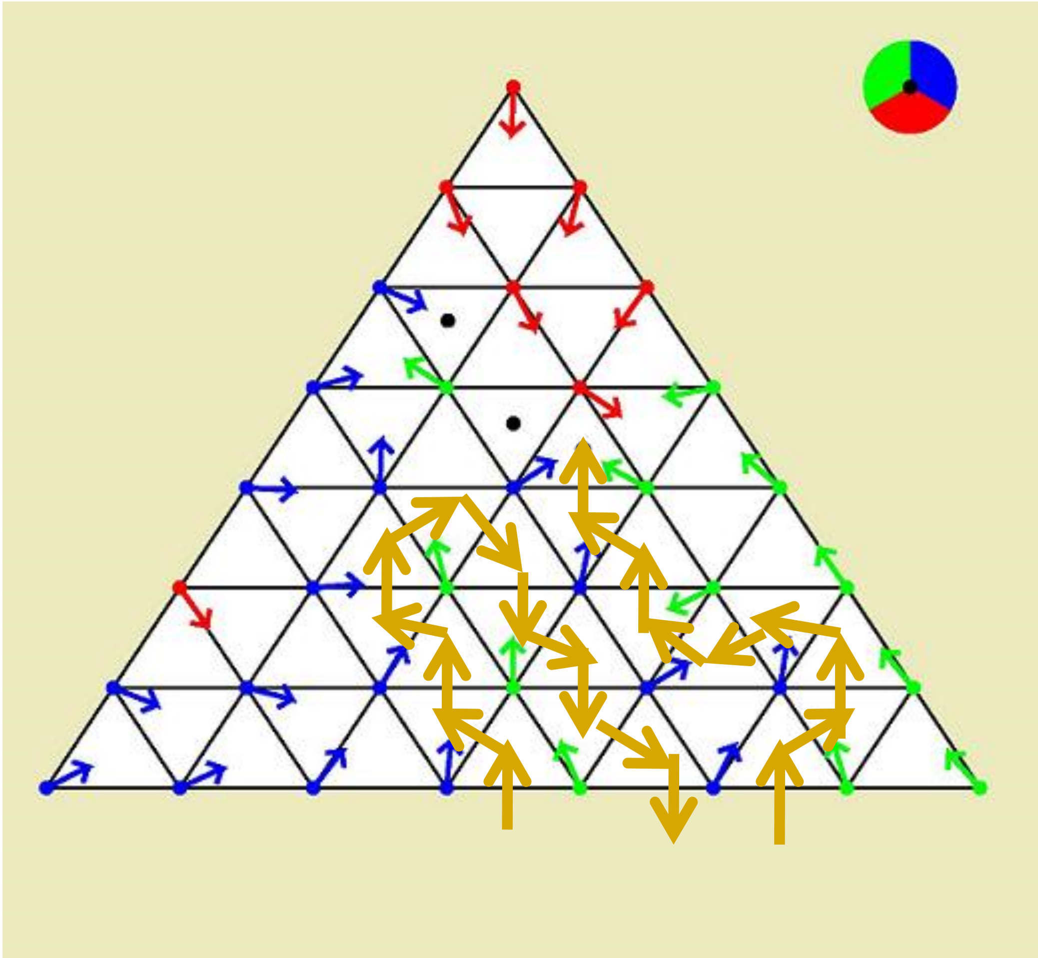
Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Solving BROUWER



Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Solving BROUWER

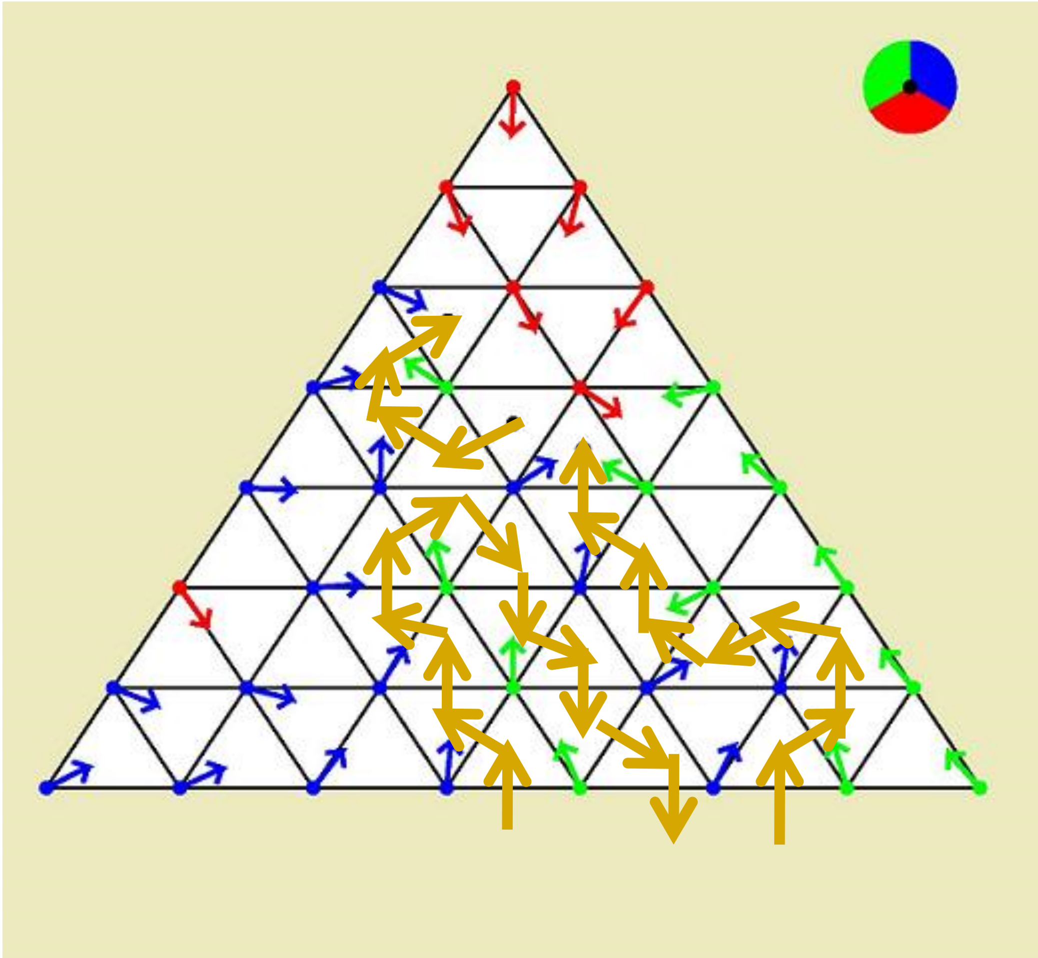


Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Sperner's lemma

- If any Sperner colouring of the n-dimensional complex has an odd number of panchromatic simplices ***then*** any Sperner colouring of the (n+1)-dimensional complex has at least one panchromatic simplex.

Solving BROUWER



Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

Sperner's lemma

- If any Sperner colouring of the n-dimensional complex has an odd number of panchromatic simplices ***then*** any Sperner colouring of the (n +1)-dimensional complex has an odd number of panchromatic simplices.

- By induction, any Sperner colouring of the n-dimensional complex has an odd number of panchomatic simplices.

Scarf's algorithm

- Scarf's algorithm: Do the Sperner walk!
- A trick makes all the paths of the induction (including failed paths) into a ***single*** path.
- The path is "locally reversible".
- Unfortunately, the path r
  the algorithm is not poly
- Deja-vu?

Facts about Lemke-Howson

- Important fact for later:
  - As in the case of the simplex algorithm, the Lemke-Howson algorithm follows a piecewise linear path in Euclidean space.
  - Unlike the case of the simplex algorithm, the path can be "locally traced backwards" – the pivoting is reversible.
- The Lemke-Howson algorithm is not polynomial time (Savani and von Stengel, 2004)

Finding exact or approximate Nash

- No polynomial time algorithm is known.
- Could the problems be NP-hard?
- We don’t think so!
- Computing Nash equilibrium is not NP-hard unless NP=coNP (Megiddo, 1988)

NP

- NP is the class of decision problems that can be solved by any algorithm of the following kind (p polynomial, A polynomial time)

- Let x be the input
- For each string y of length p(|x|):
  - If A(x,y) returns ”**yes**” then return ”**yes”**
- If no A(x,y) returns ”**yes**”, then return ”**no**”

coNP

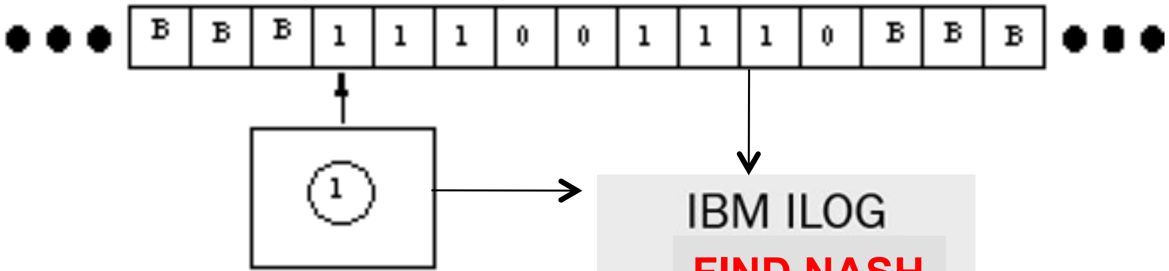
- coNP is the class of decision problems that can be solved by any algorithm of the following kind (p polynomial, A poly. time)

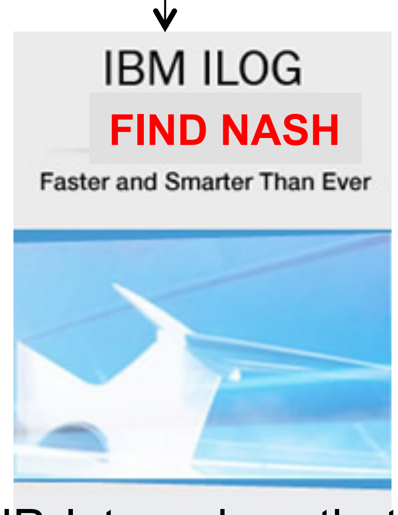
- Let x be the input
- For each string y of length p(|x|):
  - If A(x,y) returns ”**no**” then return ”**no**”
- If no A(x,y) returns ”**no**”, then return ”**yes**”

NP vs. coNP

- A decision problem is in coNP is and only if its negation is in NP.
- We don’t know how to turn an NP-type program into a co-NP type program so we do not known that NP=coNP.
- If P=NP then NP=coNP, so it is a stonger assumption to assume a separation of NP and coNP.
- While there is no philosophical evidence that the classes are different, it is still regarded a safe assumption.

Suppose Finding Nash in NP-hard





Any problem in NP can
be solved by such an device

Then, so can any problem in coNP

Now, given such a problem A in coNP, let us show that it is in NP

NP-type algorithm solving A

- Go through all pos… ***races*** of the oracl…
- Trace … Turi… …e, and…
- For … orac…
- If so… …tput "yes"
- Otherwise,

Same proof works for any task involving ***finding*** something that is guaranteed to exist ifyou can ***tell*** that you have found once you have found it.

Finding exact or approximate Nash

- No polynomial time algorithm is known.
- If P=NP, then the problem is polynomial time solvable (exercise).
- Could the problems be NP-hard?
- We don’t think so!

- Computing Nash equilibrium is not NP-hard unless NP=coNP (Megiddo, 1988)

The story so far:

Solvable (not in polynomial time) by eerily similar reversible path following algorithms.

So far, no contributions from computer science..

BROUWER-TURING

Special-LCP

BROUWER-FORMULA

2P-NASH

APPROXIMATE-NASH

Papadimitriou 1988

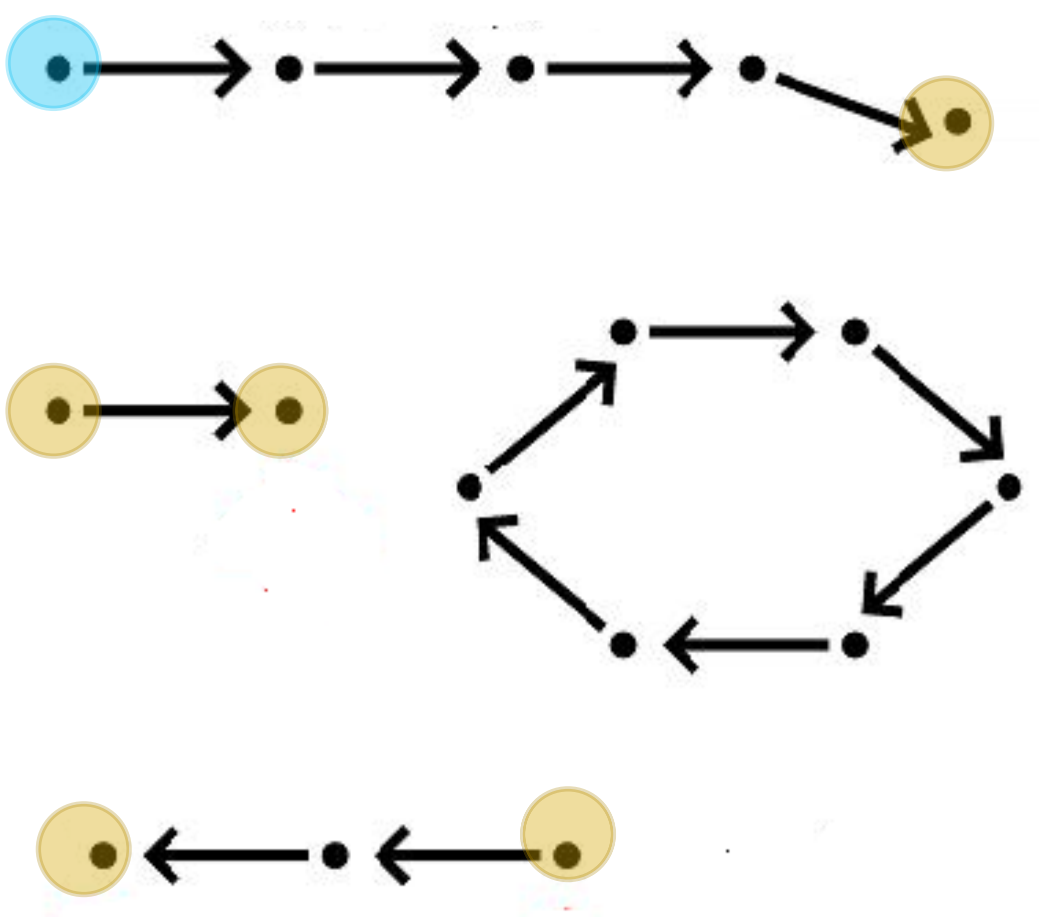
$$\begin{tikzcd}
 & \text{PPAD} & \\
\text{Special-LCP} \arrow[ur] & & \text{BROUWER-TURING} \arrow[ul] \\
 & & \text{BROUWER-FORMULA} \arrow[u] \\
\text{2P-NASH} \arrow[uu] & & \text{APPROXIMATE-NASH} \arrow[u]
\end{tikzcd}$$

PPAD (some intuition)

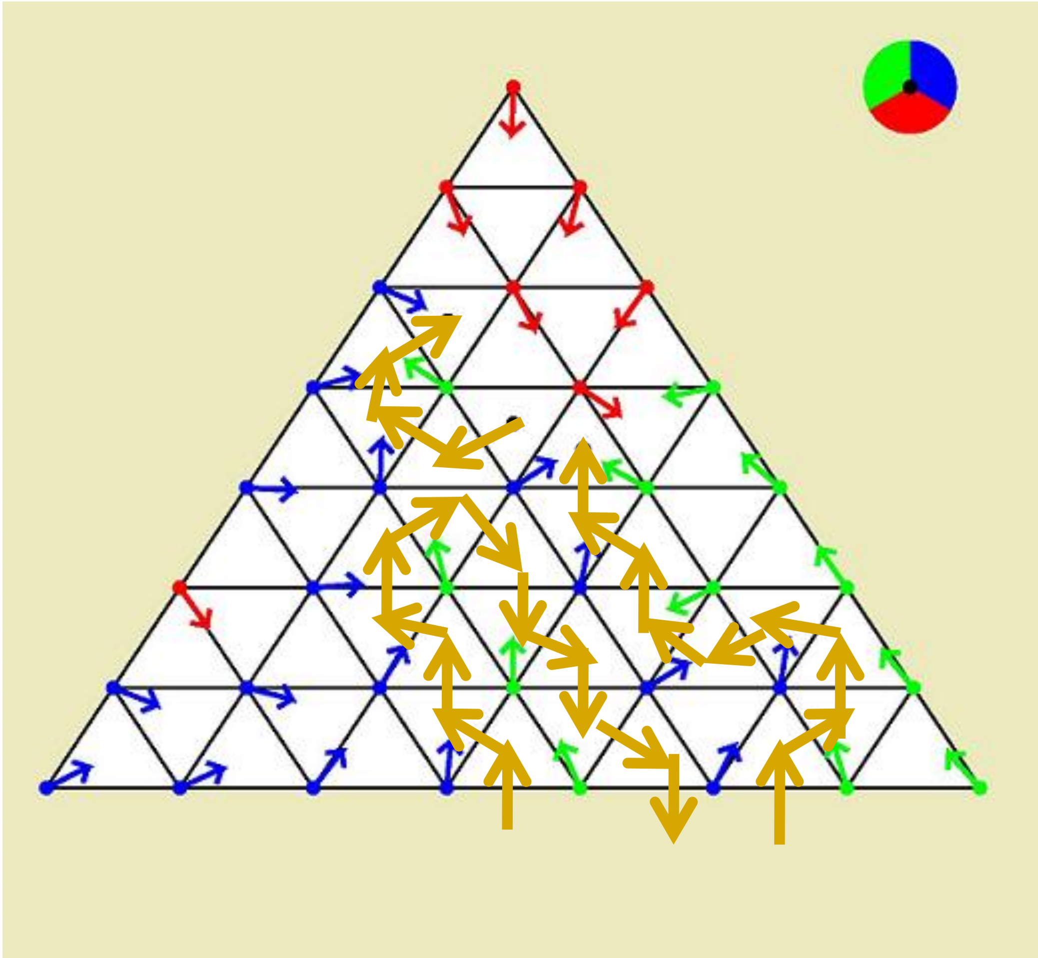
- PPAD is a class of search problems that involve finding something that is known to exists.
- You can find it by following a reversible path.
- You could also find it by other means, e.g. by exhaustive search.
  - In particular, if P=NP, then all PPAD problems er polynomial time solvable.

END-OF-A-LINE task

- END-OF-A-LINE:
  - Input: Given a directed graph with all nodes of indegree and outdegree at most 1 and a node v of indegree 0.
  - Output: A node different from v for which the indegree or the outdegree is 0.



Solving BROUWER



Pictures stolen from talk by Paul Goldberg (thanks, Paul….)

END-OF-A-LINE

- If the graph is given explicitly, END-OF-A-LINE is clearly polynomial time solvable.

- In the actual definition, the graph is given implicitly as two polynomial time subroutines:
  - S: On input y find successor of y or report that none exists.
  - P: On input y find predecessor of y or report that none exists.

(Almost) formal definition

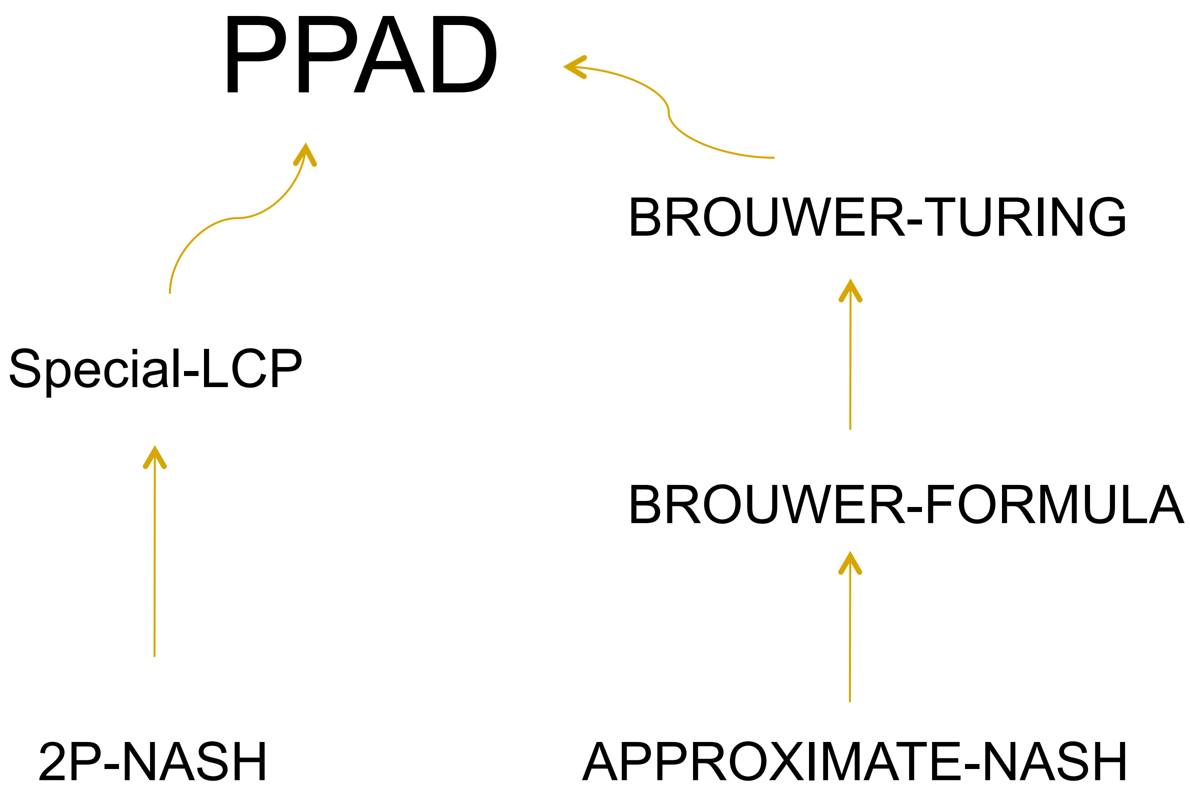
- A task is in PPAD is there are polynomial time procedures S and P and a polynomial p, so that for all strings x,
  - S(x,*) and P(x,*) with * running over all strings of length p(|x|) defines a directed graph of indegree/outdegree at most 1
  - 000…000 is a vertex of indegree 0.
  - The task can be stated as solving the END-OF-A-LINE on this graph, with input vertex 000…000.

NP

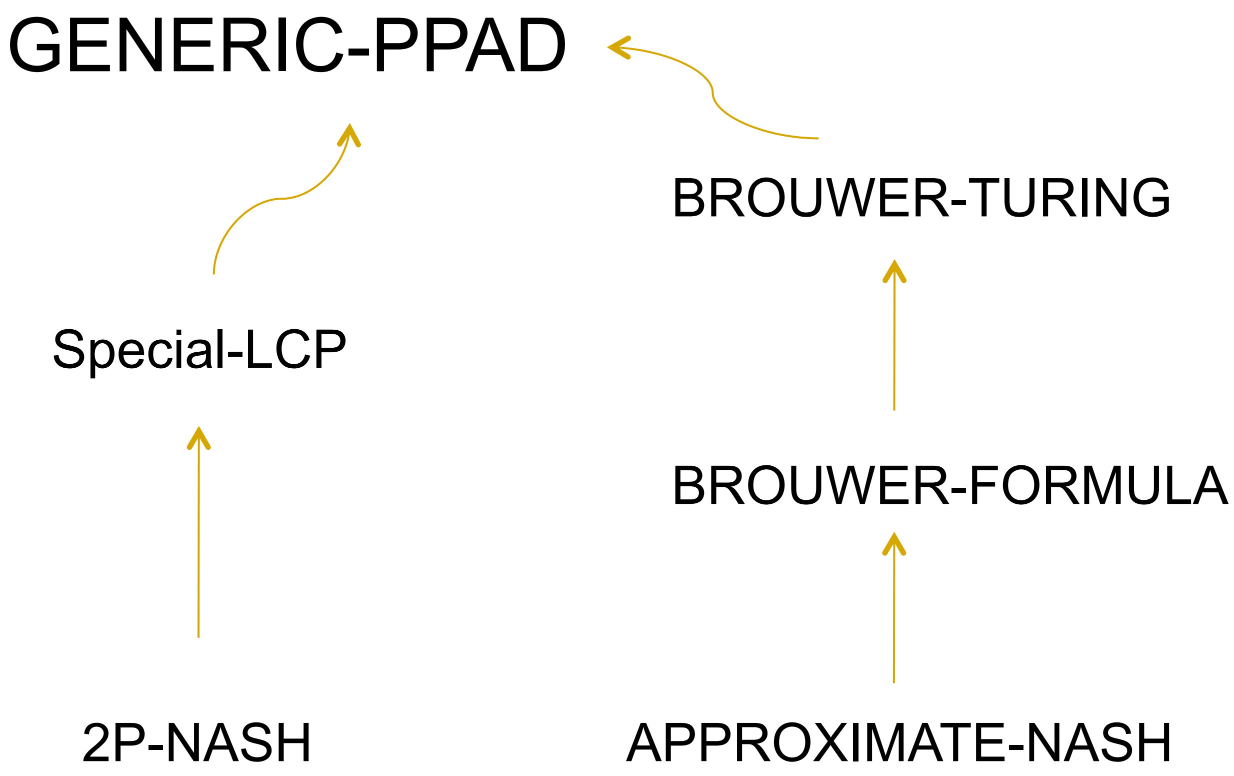
- NP is the class of decision problems that can be solved by any algorithm of the following kind (p polynomial, A polynomial time)

- Let x be the input
- For each string y of length p(|x|):
    - If A(x,y) returns ”**yes**” then return ”**yes”**
- If no A(x,y) returns ”**yes**”, then return ”**no**”

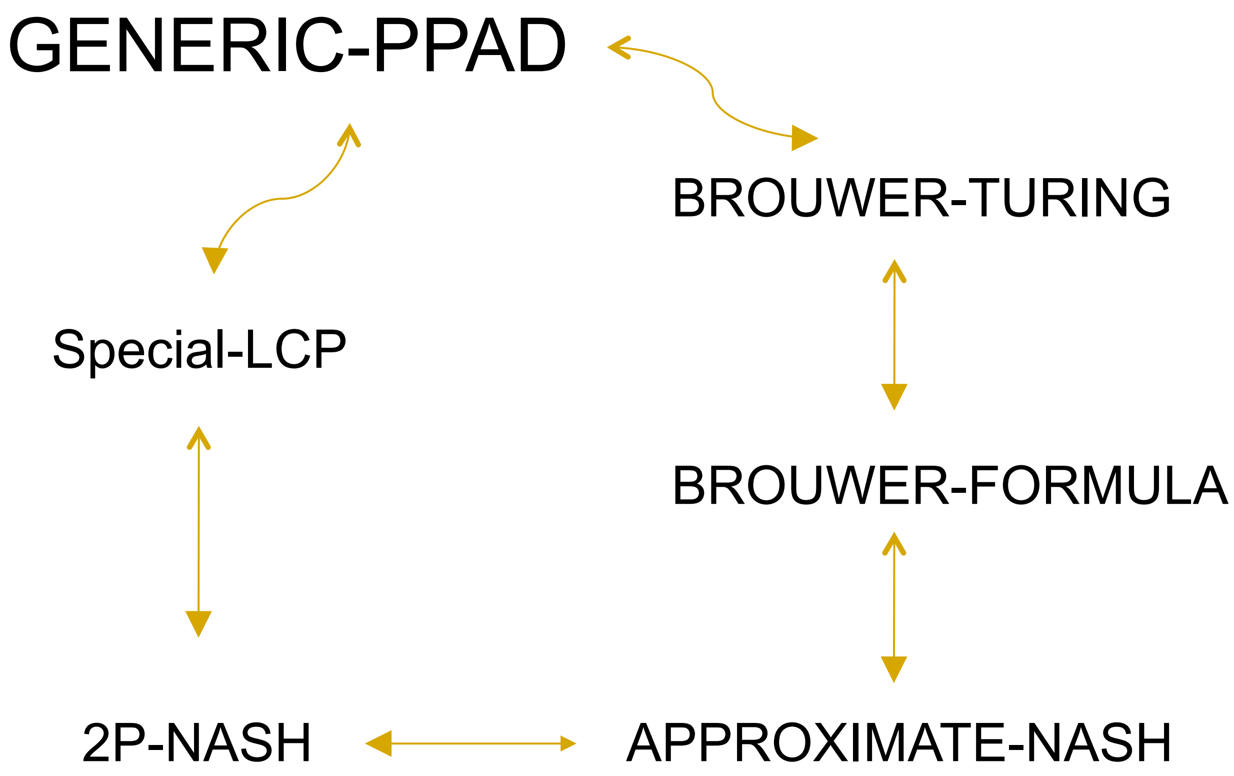
Papadimitriou 1988



Papadimitriou 1988



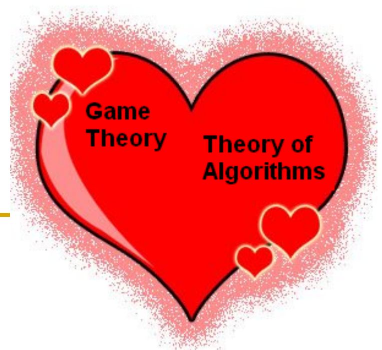
Daskalakis-Goldberg-Papadimitriou 2005, Chen-Deng 2005



All these tasks are polynomial time equivalent !!

Brouwer-Turing vs. Brouwer-Formula

- Brouwer-Turing looks like a ***much*** more general problem that Brouwer-Formula
  - Finding a fixed point of a function given by a procedure for numerically computing it, (e.g. in C) vs.
  - Finding a fixed point of a function given by a closed formula.
- We now know that Brouwer-Turing is polynomial time equivalent to Brouwer-Formula.
- Proved ***only*** because we studied Nash equilibrium notion and used Nash' Brouwer-based proof of the existence of Nash equilibrium…..



Finding approximate Nash equilibria

- APPROXIMATE-NASH:
  - Input: A multi-player game in normal form with rational payoffs and ε.
  - Output: An ε-Nash equilibrium of the game

- Fair substitute?
- A rather unsatisfactory notion to people who care about infinitisimals! As we do!
- More discussion later!

Finding approximations of Nash equilibria

- APPROXIMATION-NASH:
  - Input: A multi-player game in normal form with rational payoffs and ε.
  - Output: A strategy profile of Euclidean distance at most ε to an actual Nash equilibrium.
- Etessami and Yanakakis 2007.
  - APPROXIMATE-NASH polynomially reduces to APPROXIMATION-NASH.
  - But APPROXIMATION-NASH seems much harder…

Generic task of numerical computation

- A straight line program, involving rational numbers, additions, subtractions, multiplications and divisions, and a positive integer d.
- Output: The result, to d significant digits.

- We do not know that this task is polynomial time solvable.
- We also don’t know how (or if) it compares to the NP-complete problems in terms of poly time reductions.

Etessami and Yanakakis, 2007

- The generic task of numerical computation polynomial time reduces to APPROXMATION-NASH (which is ”approx-FIXP”-complete).
- Non-computational byproduct by passing interesting computation through reduction:
  - Games with ε-approximate Nash equilibria for ***very*** small ε that are extremely far away from any exact Nash equilibrium.

NP-hardness

It is **NP**-hard to decide if a given pure Nash equilibrium for a given 3-player game in normal form (i.e., as a table of payoffs) is trembling hand perfect.

How about finding a trembling hand perfect equilibrium?

How about *finding* a trembling hand perfect equilibrium?

- No notion of ”approximate trembling hand perfect equilibrium”, so no analogy to PPAD-completeness results for three or more players.

With Kousha Etessami (in writing):

- Approximating an actual trembling hand perfect equilibrium in a multi-player game *is polynomial time equivalent* to approximating an actual Nash equilibrium in a multi-player game
  - both are ”approxFIXP”-complete