Algorithms for Large Sequential Incomplete-Information Games

Tuomas Sandholm

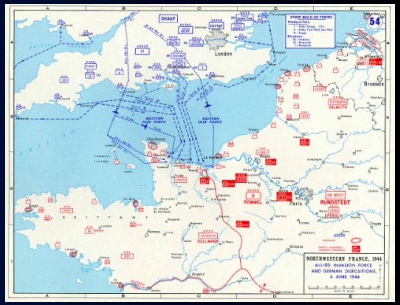
Professor

Carnegie Mellon University

Computer Science Department

Most real-world games are incomplete-information games with sequential (& simultaneous) moves

- Negotiation
- Multi-stage auctions (e.g., FCC ascending, combinatorial auctions)
- Sequential auctions of multiple items
- A robot facing adversaries in uncertain, stochastic envt
- Card games, e.g., poker
- Currency attacks
- International (over-)fishing
- Political campaigns (e.g., TV spending in each region)
- Ownership games (polar regions, moons, planets)
- Allocating (and timing) troops/armaments to locations
  - US allocating troops in Afghanistan & Iraq
  - Military spending games, e.g., space vs ocean
  - Airport security, air marshals, coast guard, rail [joint w Tambe]
  - Cybersecurity ...




Sequential incomplete-information games

- Challenges
  - Imperfect information
  - Risk assessment and management
  - Speculation and counter-speculation: Interpreting signals and avoiding signaling too much
- Techniques for complete-info games don’t apply
- Techniques I will discuss are domain-independent

Game theory

- **Definition.** Strategy is a mapping from known history to action

- In multi-agent systems, an agent's outcome depends on the actions of others'
  =>Agent's *optimal* strategy depends on others' strategies

- **Definition**. A (Bayes) Nash equilibrium is a strategy (and beliefs) for each agent such that no agent benefits from using a different strategy

Simple example

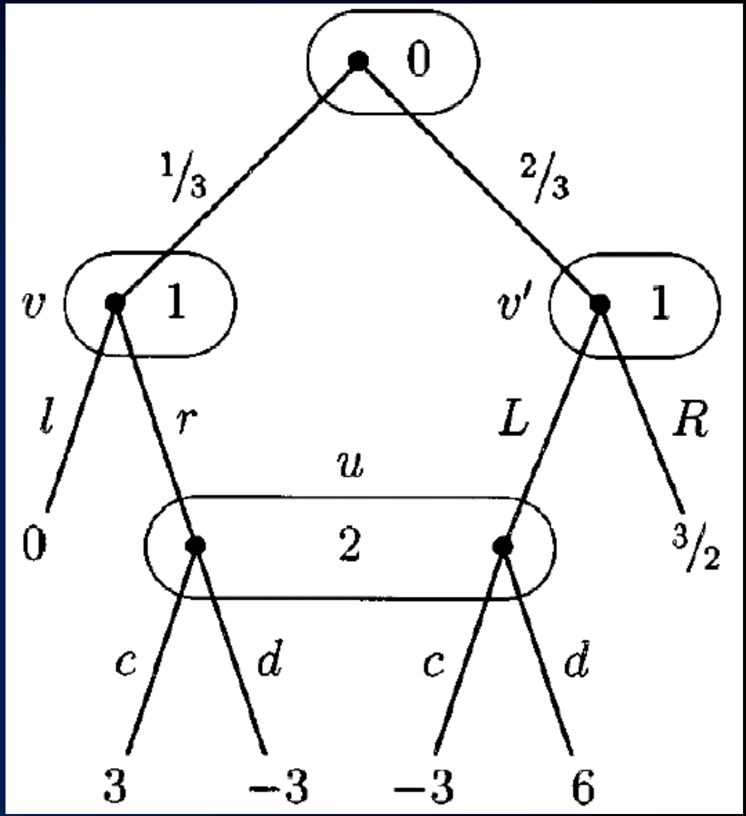
			Player 2	
		Rock 1/3	Paper 1/3	Scissors 1/3
Player 1	Rock 1/3	0, 0	-1, 1	1, -1
	Paper 1/3	1, -1	0, 0	-1, 1
	Scissors 1/3	-1, 1	1, -1	0, 0

Basics about Nash equilibria

- In 2-person 0-sum games,
  - Nash equilibria are minimax equilibria => no equilibrium selection problem
  - If opponent plays a non-equilibrium strategy, that only helps me
- Any finite sequential game (satisfying perfect recall) can be converted into a matrix game
  - Exponential blowup in #strategies
- *Sequence form*: More compact representation based on sequences of moves rather than pure strategies [Romanovskii 62, Koller & Megiddo 92, von Stengel 96]
  - 2-person 0-sum games with perfect recall can be solved in time polynomial in size of game tree using LP
  - Cannot solve Rhode Island Hold'em (3.1 billion nodes) or Texas Hold'em (10^{18} nodes)

Extensive form representation

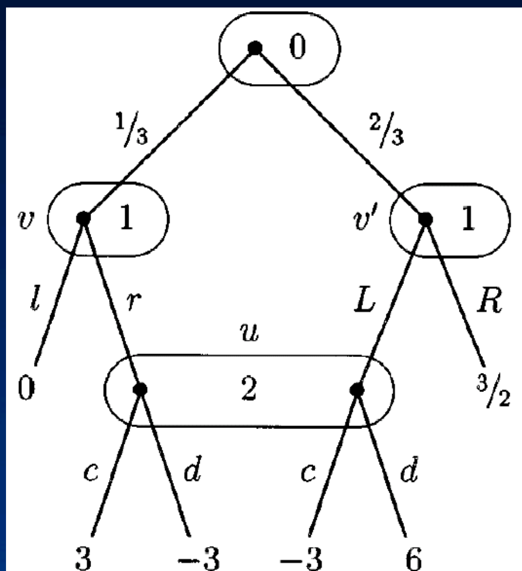
- Players $I = \{0, 1, \ldots, n\}$
- Tree (V,E)
  - Terminals $Z \subseteq V$
- Controlling player P: $V \setminus Z \rightarrow H$
- Information sets $H=\{H_0,\ldots, H_n\}$
- Actions $A = \{A_0, \ldots, A_n\}$
- Payoffs $u : Z \rightarrow R^n$
- Chance probabilities p



Perfect recall assumption: Players never forget information

Game from: Bernhard von Stengel. *Efficient Computation of Behavior Strategies*. In *Games and Economic Behavior* 14:220-246, 1996.

Computing equilibria via normal form



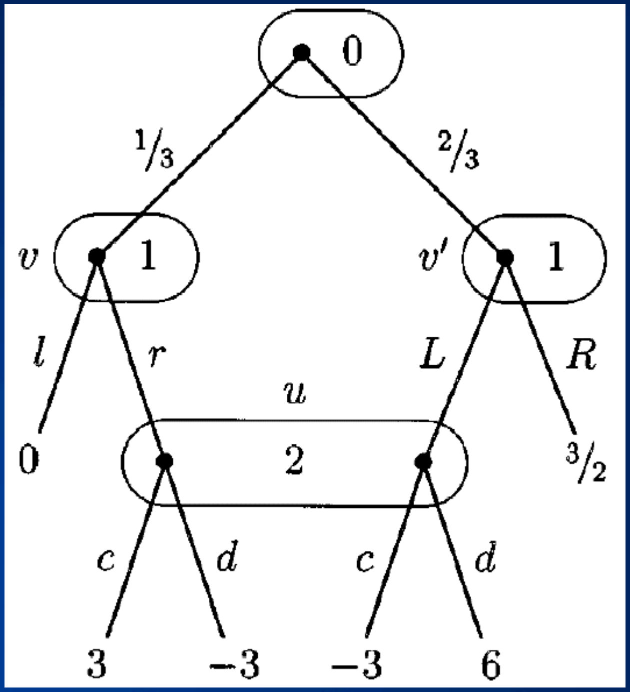
	c	d
(l, L)	-2	4
(l, R)	1	1
(r, L)	-1	3
(r, R)	2	0

- Normal form exponential, in worst case and in practice (*e.g.* poker)

Sequence form

[Romanovskii 62, re-invented in English-speaking literature: Koller & Megiddo 92, von Stengel 96]

- Instead of a move for every information set, consider choices necessary to reach each information set and each leaf
- These choices are *sequences* and constitute the pure strategies in the sequence form



$$S_1 = \{\{\}, l, r, L, R\}$$
$$S_2 = \{\{\}, c, d\}$$

Realization plans

- Players' strategies are specified as realization plans over sequences:

$$r_i(\emptyset) = 1$$

$$-r_i(\sigma_u) + \sum_{c \in C_u} r_i(\sigma_u c) = 0$$

$$r_i(s_i) \geq 0$$

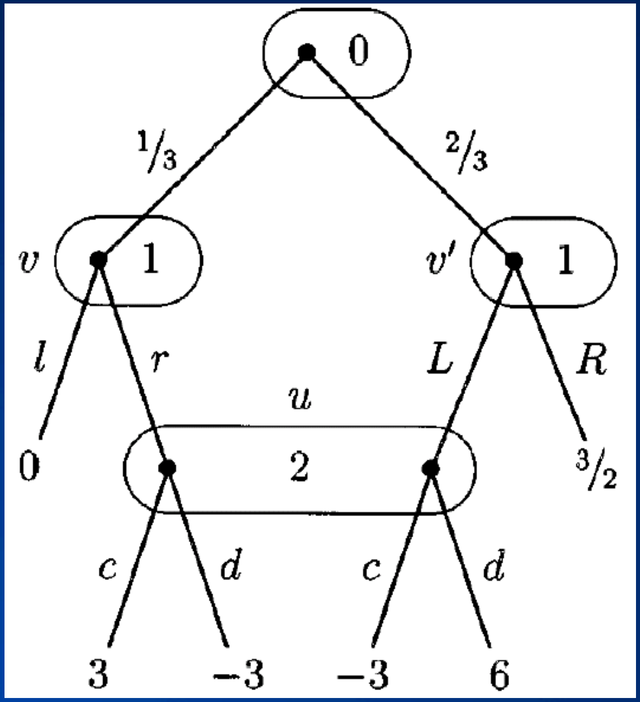
- **Prop**. Realization plans are equivalent to behavior strategies.

Computing equilibria via sequence form

- Players 1 and 2 have realization plans x and y
- Realization *constraint matrices* E and F specify constraints on realizations

$$Ex = e \qquad Fy = f$$

$$e = (1, 0, 0)^T \qquad f = (1, 0)^T$$



$$E = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{ccccc} \{\} & l & r & L & R \end{array}$$

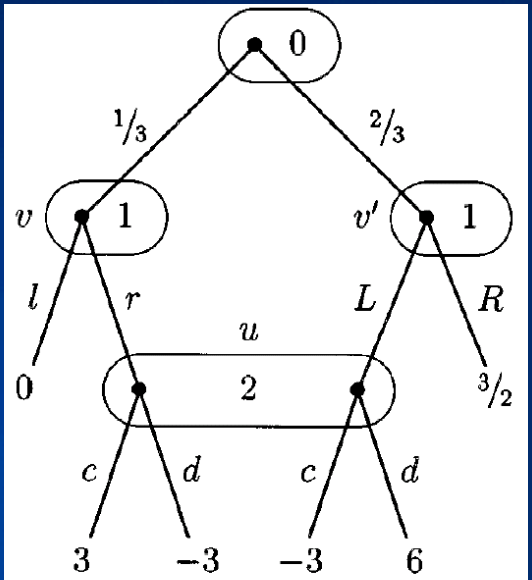
$$\begin{array}{cc} E = \begin{pmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ -1 & & & 1 & 1 \end{pmatrix} & \begin{array}{c} \{\} \\ v \\ v' \end{array} \end{array}$$

$$\begin{array}{ccc} \{\} & c & d \end{array}$$

$$\begin{array}{cc} F = \begin{pmatrix} 1 & & \\ -1 & 1 & 1 \end{pmatrix} & \begin{array}{c} \{\} \\ u \end{array} \end{array}$$

Computing equilibria via sequence form

- Payoffs for player 1 and 2 are: $x^T A y$ and $x^T B y$ for suitable matrices A and B
- Creating payoff matrix:
  - Initialize each entry to 0
  - For each leaf, there is a (unique) pair of sequences corresponding to an entry in the payoff matrix
  - Weight the entry by the product of chance probabilities along the path from the root to the leaf



$$
\begin{array}{c c}
 & \begin{array}{ccc} \{\} & c & d \end{array} \\
A = & \left(\begin{array}{ccc} & & \\ 0 & & \\ & 1 & -1 \\ & -2 & 4 \\ 1 & & \end{array}\right)
\end{array}
\begin{array}{c} \{\} \\ l \\ r \\ L \\ R \end{array}
$$

Computing equilibria via sequence form

	Primal	Dual
Holding x fixed, compute best response	$\begin{array}{ll} \underset{y}{\text{maximize}} & (x^T B) y \\ \text{subject to} & Fy = f, \\ & y \geq 0. \end{array}$	$\begin{array}{ll} \underset{q}{\text{minimize}} & q^T f \\ \text{subject to} & q^T F \geq x^T B \end{array}$
Holding y fixed, compute best response	$\begin{array}{ll} \underset{x}{\text{maximize}} & x^T (Ay) \\ \text{subject to} & x^T E^T = e^T \\ & x \geq 0. \end{array}$	$\begin{array}{ll} \underset{p}{\text{minimize}} & e^T p \\ \text{subject to} & E^T p \geq Ay \end{array}$

Now, assume 0-sum. The latter primal and dual must have same optimal value $e^T p$. That is the amount that player 2, if he plays y, has to give to player 1, so player 2 tries to minimize it:

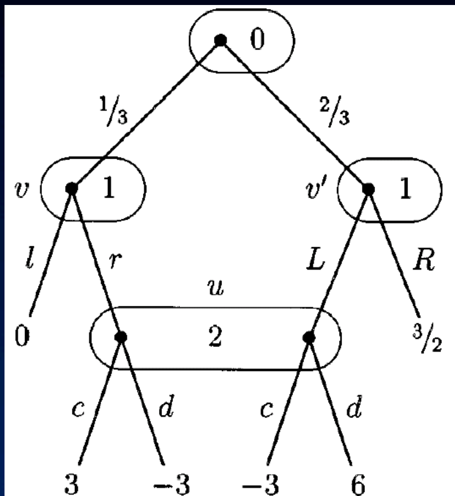
Primal

$$\begin{array}{ll} \underset{y,p}{\text{minimize}} & e^T p \\ \text{subject to} & -Ay + E^T p \geq 0 \\ & -Fy = -f, \\ & y \geq 0. \end{array}$$

Dual

$$\begin{array}{ll} \underset{x,q}{\text{maximize}} & -q^T f \\ \text{subject to} & x^T(-A) - q^T F \leq 0 \\ & x^T E^T = e^T, \\ & x \geq 0. \end{array}$$

Computing equilibria via sequence form: An example



$$A = \begin{pmatrix} 0 & & \\ & 1 & -1 \\ & -2 & 4 \\ 1 & & \end{pmatrix}$$

$$E = \begin{pmatrix} 1 & & & & \\ -1 & 1 & 1 & & \\ -1 & & & 1 & 1 \end{pmatrix}$$

$$F = \begin{pmatrix} 1 & & \\ -1 & 1 & 1 \end{pmatrix}$$

$$\begin{array}{ll} \underset{y,p}{\text{minimize}} & e^T p \\ \text{subject to} & -Ay + E^T p \geq 0 \\ & -Fy = -f, \\ & y \geq 0. \end{array}$$

```
                  min p1
                subject to
x1:                         p1 - p2 - p3 >= 0
x2: 0y1 +                        p2      >= 0
x3:        -y2 +  y3           + p2      >= 0
x4:        2y2 - 4y3                + p3 >= 0
x5: -y1                             + p3 >= 0
        q1: -y1                    = -1
        q2:  y1 - y2 - y3 =  0
                  bounds
       y1 >= 0   y2 >= 0   y3 >= 0
       p1 Free   p2 Free   p3 Free
```

Sequence form summary

- Polytime algorithm for finding a Nash equilibrium in 2-player zero-sum games
- Polysize linear complementarity problem (LCP) for computing Nash equilibria in 2-player general-sum games
- Major shortcomings:
  - Not well understood when more than two players
  - Sometimes, polynomial is still slow and or large (*e.g.* poker)…

Games and information

- Games can be differentiated based on the *information* available to the players
  - *Perfect* information games: players have complete knowledge about the state of the world
    - Examples: Chess, Go, Checkers
  - *Imperfect* information games: players face uncertainty about the state of the world
    - Examples:
      - A robot facing adversaries in an uncertain, stochastic environment
      - Almost any economic situation in which the other participants possess private information (*e.g.* valuations, quality information)
      - Almost any card game in which the other players' cards are hidden
    - This class of games presents several challenges for AI
      - Imperfect information
      - Risk assessment and management
      - Speculation and counter-speculation

Poker

- Recognized challenge problem in AI
  - Hidden information (other players' cards)
  - Uncertainty about future events
  - Deceptive strategies needed in a good player
- Very large game trees
- Texas Hold'em is the most popular variant

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

Other methods for finding equilibria

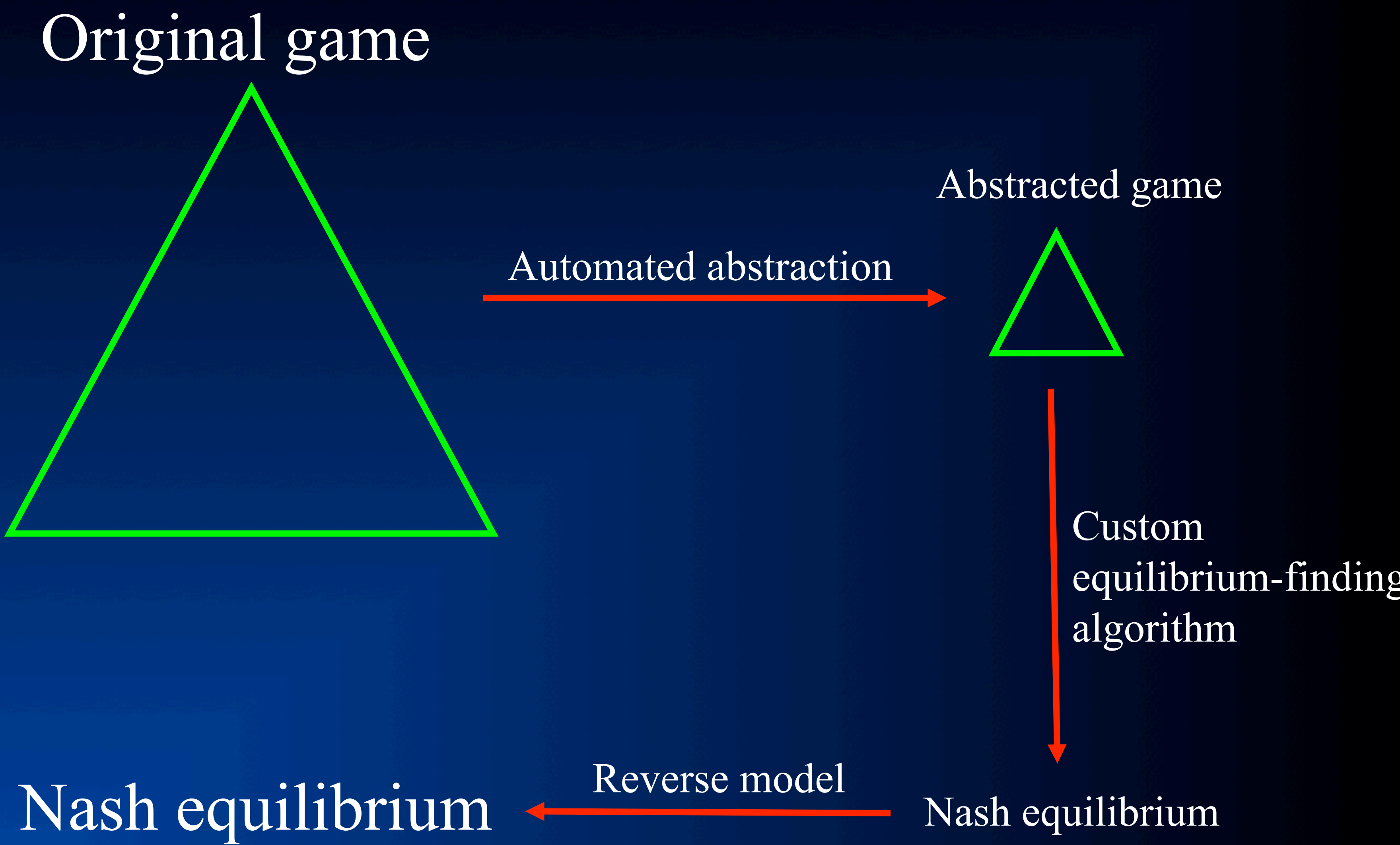
- Fictitious play
  - Convergence only guaranteed for zero-sum games
- Tabu best-response search [Sureka & Wurman 2005]
  - Finds pure strategy equilibria
  - Does not require game to be completely specified
- Lemke-Howson algorithm
  - Pivoting algorithm for finding one Nash equilibrium
  - Very similar to the simplex algorithm for LP
- Support enumeration methods
  - Porter-Nudelman-Shoham [2004]
  - Mixed-Integer Programming Nash [Sandholm *et al* 2005]

Our approach

Automated abstraction + equilibrium finding

Our approach [Gilpin & S., EC'06, JACM'07…]

Now used by all competitive Texas Hold'em programs



Outline

- Automated abstraction
  - Lossless
  - Lossy
- New equilibrium-finding algorithms

Outline

- Automated abstraction
  - Lossless
  - Lossy
- New equilibrium-finding algorithms
- Stochastic games with >2 players, e.g., poker tournaments
- Current & future research

Outline

- Lossless automated abstraction
  - Optimal strategies for Rhode Island Hold'em
- Approximate automated abstraction
  - "Greedy" (*GS1*)
  - Clustering and integer programming (*GS2*)
  - Potential-aware (*GS3*)
- Equilibrium-finding algorithms
  - Adapting Nesterov's excessive gap technique to sequential games
  - Making it scalable
  - New related algorithm with exponentially better speed
- Future research
- Thoughts on application games of national importance

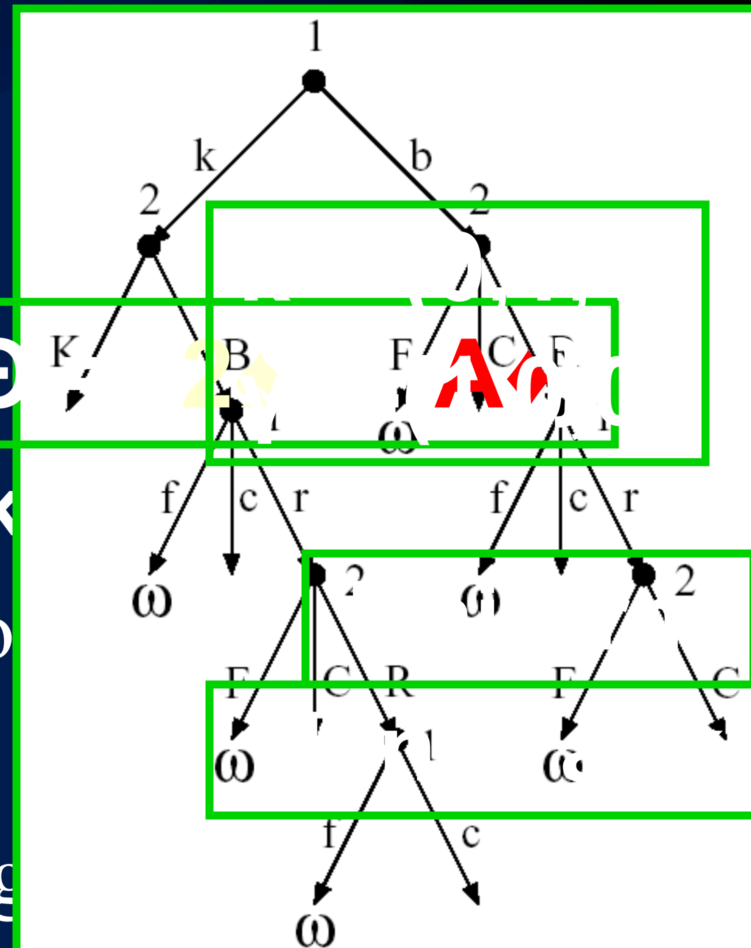
Our approach

- We introduce automated abstraction techniques that result in smaller, (nearly) equivalent games
  - For the optimal version of our algorithm:
    - We prove that a Nash equilibrium in the smaller game corresponds to a Nash equilibrium in the original game
    - The smaller game can then be solved using standard techniques
  - For the approximate versions of our algorithm:
    - We demonstrate their effectiveness by applying the algorithm to Texas Hold'em poker and comparing with other poker-playing programs
- We also improve the equilibrium-finding algorithms themselves

Game with ordered signals (a.k.a. ordered game)

1. Players **I = {1,…,n}**
2. Stage games **G = G$_1$,…,G$_r$**
3. Player label **L**
4. Game-ending nodes **ω**
5. Signal alphabet **Θ**
6. Signal quantities **κ = κ$_1$,…,κ**
7. Signal probability distributio
8. Partial ordering ≥ of subsets
9. Utility function **u** (increasing

I = {1,2}



Reasons to abstract

- Scalability (computation speed & memory)
- Game may be so complicated that can't model without abstraction
- Existence of equilibrium, or solving algorithm, may require a certain kind of game, e.g., finite

Lossless abstraction

[Gilpin & S., EC'06, JACM'07]

Information filters

- Observation: We can make games smaller by filtering the information a player receives

- Instead of observing a specific signal exactly, a player instead observes a **filtered set** of signals
  - *E.g.* receiving signal {A♠,A♣,A♥,A♦} instead of A♥

Signal tree

- Each edge corresponds to the revelation of some signal by nature to at least one player
- Our lossless abstraction algorithm operates on it
  – Don’t load full game into memory

Isomorphic relation

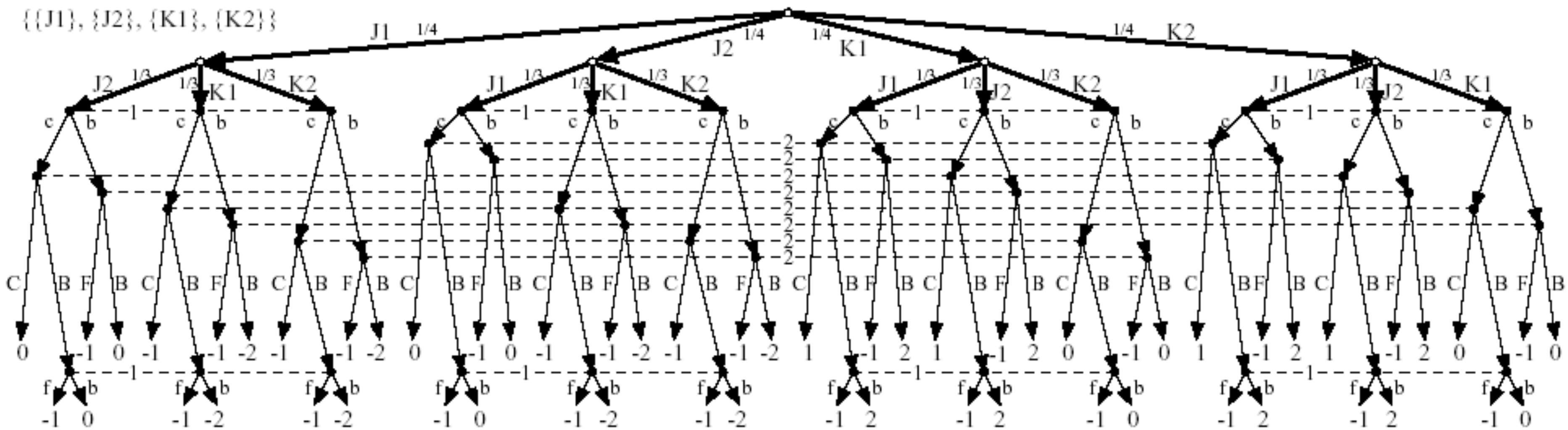
- Captures the notion of strategic symmetry between nodes
- Defined recursively:
  - Two leaves in signal tree are isomorphic if for each action history in the game, the payoff vectors (one payoff per player) are the same
  - Two internal nodes in signal tree are isomorphic if they are siblings and there is a bijection between their children such that only ordered game isomorphic nodes are matched
- We compute this relationship for all nodes using a DP plus custom perfect matching in a bipartite graph

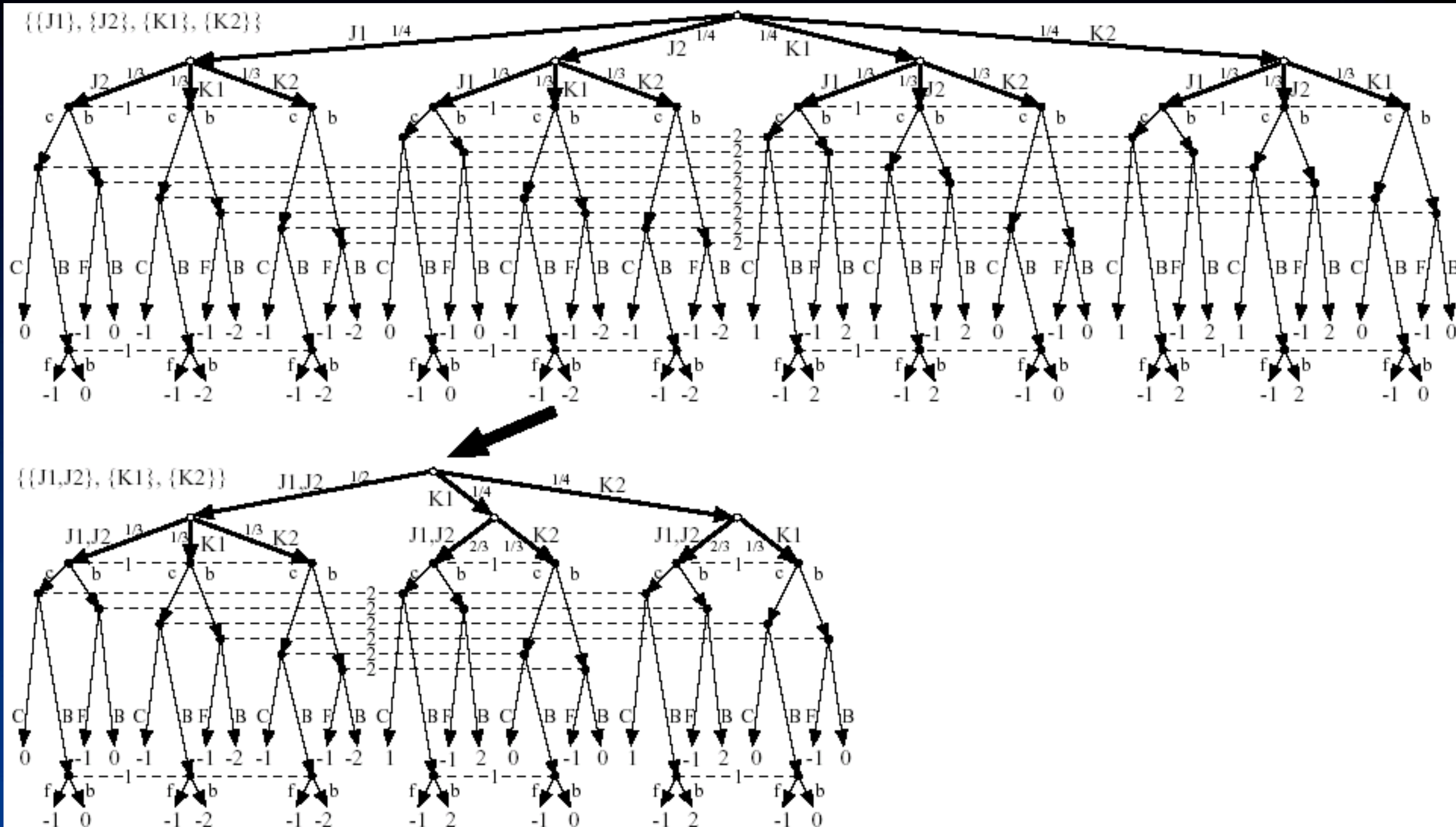
Abstraction transformation

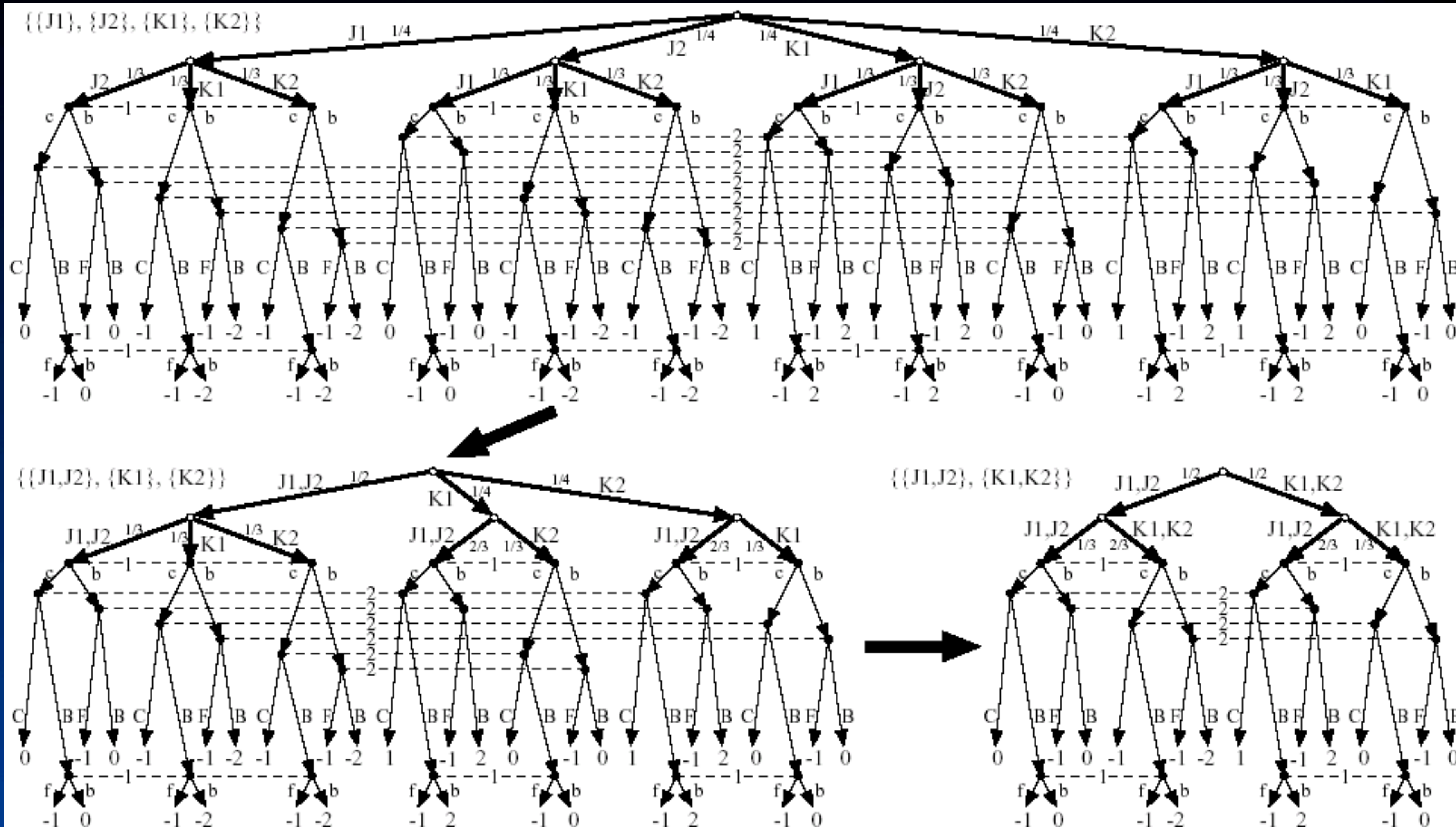
- Merges two isomorphic nodes

- **Theorem.** *If a strategy profile is a Nash equilibrium in the abstracted (smaller) game, then its interpretation in the original game is a Nash equilibrium*

- Assumptions
  - Observable player actions
  - Players' utility functions rank the signals in the same order


{{J1}, {J2}, {K1}, {K2}}
J1 1/4
J2 1/4
K1 1/4
K2 1/4
J2 1/3 K1 1/3 K2 1/3
J1 1/3 K1 1/3 K2 1/3
J1 1/3 J2 1/3 K2 1/3
J1 1/3 J2 1/3 K1 1/3
c b
C B F B
f b
1
2


{{J1}, {J2}, {K1}, {K2}}
{{J1,J2}, {K1}, {K2}}


{{J1}, {J2}, {K1}, {K2}}
{{J1,J2}, {K1}, {K2}}
{{J1,J2}, {K1,K2}}

GameShrink algorithm

- Bottom-up pass: Run DP to mark isomorphic pairs of nodes in signal tree
- Top-down pass: Starting from top of signal tree, perform the transformation where applicable

- **Theorem.** Conducts all these transformations
  - $\tilde{O}(n^2)$, where n is #nodes in *signal tree*
  - Usually highly *sublinear* in game tree size

Algorithmic techniques for making GameShrink faster

- Union-Find data structure for efficient representation of the information filter (unioning finer signals into coarser signals)
  - Linear memory and almost linear time
- Eliminate some perfect matching computations using easy-to-check necessary conditions
  - Compact histogram databases for storing win/loss frequencies to speed up the checks

Solved Rhode Island Hold'em poker

- AI challenge problem [Shi & Littman 01]
  - 3.1 billion nodes in game tree
- Without abstraction, LP has 91,224,226 rows and columns => unsolvable
- *GameShrink* runs in one second
- After that, LP has 1,237,238 rows and columns
- Solved the LP
  - CPLEX *barrier* method took 8 days & 25 GB RAM
- Exact Nash equilibrium
- Largest incomplete-info game solved by then by over 4 orders of magnitude

Lossy abstraction

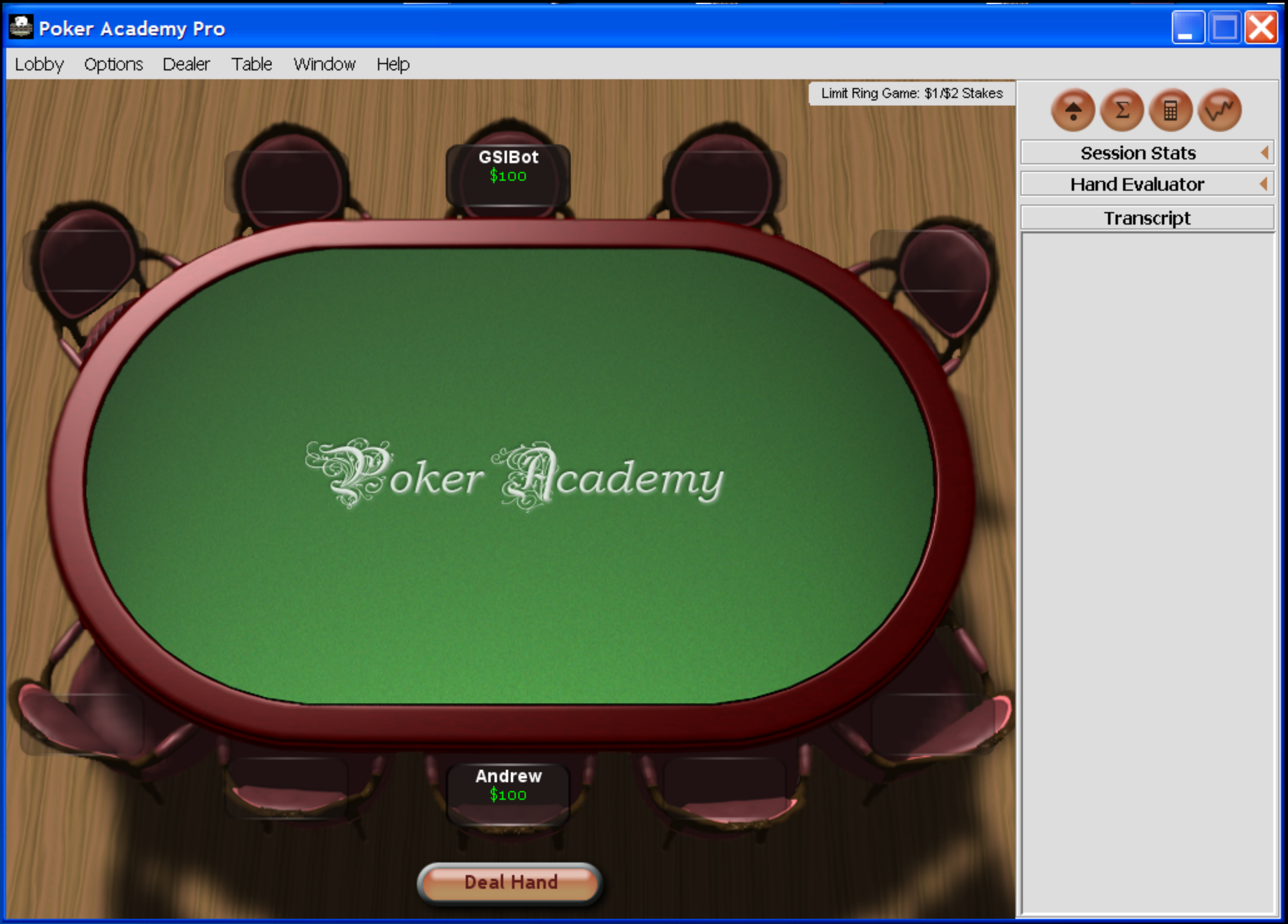
Prior game abstractions (automated or manual)

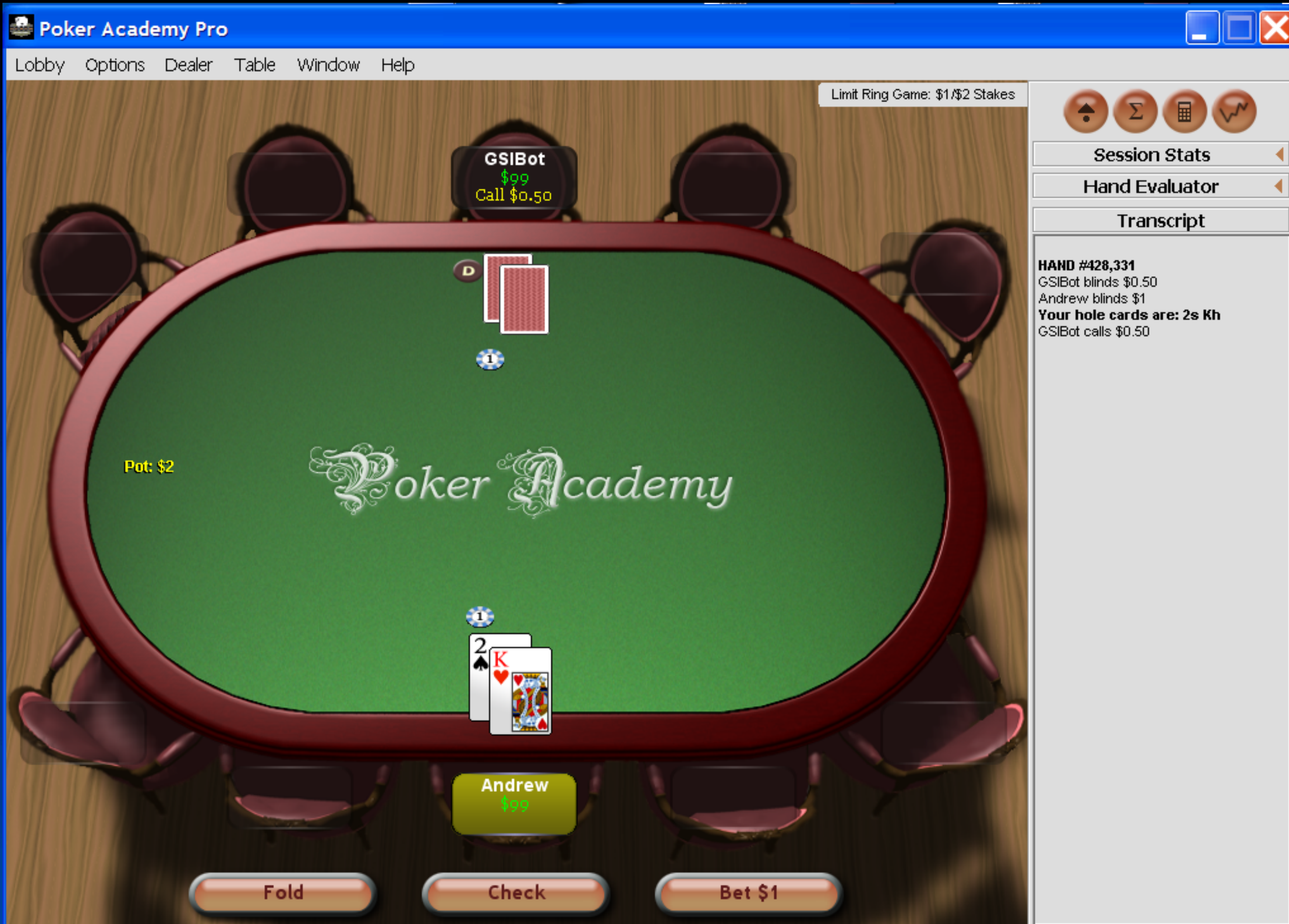
- Lossless [Gilpin & Sandholm, EC'06, JACM'07]
- Lossy without bound [Shi and Littman CG-02; Billings et al. IJCAI-03; Gilpin & Sandholm, AAAI-06, -08, AAMAS-07; Gilpin, Sandholm & Soerensen AAAI-07, AAMAS-08; Zinkevich et al. NIPS-07; Waugh et al. AAMAS-09, SARA-09;…]
  – Exploitability can sometimes be checked *ex post* [Johanson et al. IJCAI-11]

We developed many lossy abstraction algorithms

- Scalable to large n-player, general-sum games, e.g., Texas Hold’em
- Gilpin, A. and Sandholm, T. 2008. Expectation-Based Versus Potential-Aware Automated Abstraction in Imperfect Information Games: An Experimental Comparison Using Poker. *AAAI*.
- Gilpin, A., Sandholm, T., Troels Bjerre Sørensen 2008. A heads-up no-limit Texas Hold'em poker player: Discretized betting models and automatically generated equilibrium-finding programs. *AAMAS*.
- Gilpin, A., Sandholm, T., Soerensen, T. 2007. Potential-Aware Automated Abstraction of Sequential Games, and Holistic Equilibrium Analysis of Texas Hold'em Poker. *AAAI*.
- Gilpin, A., Sandholm, T. 2007. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *AAMAS*.
- Gilpin, A., Sandholm, T. 2006. A competitive Texas Hold'em Poker player via automated abstraction and real-time equilibrium computation. *AAAI*.

Texas Hold'em poker


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$100
Poker Academy
Andrew
$100
Deal Hand
Session Stats
Hand Evaluator
Transcript


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$99
Call $0.50
D
1
Pot: $2
Poker Academy
1
2
K
Andrew
$99
Fold
Check
Bet $1
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50


Poker Academy Pro
Lobby Options Dealer Table Window Help
Limit Ring Game: $1/$2 Stakes
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
GSIBot
$99
Pot: $2
Andrew
$99
Fold
Check
Bet $1


Poker Academy Pro
Lobby Options Dealer Table Window Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$98
Bet $1
Pot: $3
Andrew
$99
Fold
Call $1
Raise $1
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
Andrew checks
GSIBot bets $1


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
Andrew checks
GSIBot bets $1
Andrew calls $1
TURN: Qd 7s 4c 3s
GSIBot
$98
Pot: $4
Andrew
$98
Fold
Check
Bet $2


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
Andrew checks
GSIBot bets $1
Andrew calls $1
TURN: Qd 7s 4c 3s
Andrew bets $2
GSIBot calls $2
RIVER: Qd 7s 4c 3s Qs
GSIBot
$96
Pot: $8
Andrew
$96
Fold
Check
Bet $2


Poker Academy Pro
Lobby Options Dealer Table Window Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$94
Bet $2
Pot: $10
Andrew
$96
Fold
Call $2
Raise $2
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
Andrew checks
GSIBot bets $1
Andrew calls $1
TURN: Qd 7s 4c 3s
Andrew bets $2
GSIBot calls $2
RIVER: Qd 7s 4c 3s Qs
Andrew checks
GSIBot bets $2


Poker Academy Pro
Lobby Options Dealer Table Window Help
Limit Ring Game: $1/$2 Stakes
Session Stats
Hand Evaluator
Transcript
HAND #428,331
GSIBot blinds $0.50
Andrew blinds $1
Your hole cards are: 2s Kh
GSIBot calls $0.50
Andrew checks
FLOP: Qd 7s 4c
Andrew checks
GSIBot bets $1
Andrew calls $1
TURN: Qd 7s 4c 3s
Andrew bets $2
GSIBot calls $2
RIVER: Qd 7s 4c 3s Qs
Andrew checks
GSIBot bets $2
Andrew calls $2
GSIBot shows 2c 7c
Andrew shows 2s Kh
GSIBot wins $12 with Two Pair, Queens and Sevens
GSIBot
$106
GSIBot wins $12 with Two Pair, Queens and Sevens
Andrew
$94
Deal Hand

Texas Hold’em poker

Nature deals 2 cards to each player

Round of betting

Nature deals 3 shared cards

Round of betting

Nature deals 1 shared card

Round of betting

Nature deals 1 shared card

Round of betting

- 2-player Limit Texas Hold’em has ~10^{18} leaves in game tree
- Losslessly abstracted game too big to solve
  => abstract more
  => lossy

GS1 [Gilpin & S., AAAI'06]

- First Texas Hold'em program to use automated abstraction
  - Lossy version of Gameshrink
    - Instead of requiring perfect matching of children, require a matching with a penalty below threshold
- Abstracted game's LP solved by CPLEX
- Phase I (rounds 1 & 2) LP solved offline
  - Assuming rollout for the rest of the game
- Phase II (rounds 3 & 4) LP solved in real time
  - Starting with hand probabilities that are updated using Bayes rule based on Phase I equilibrium and observations

GS1

1/2005 - 1/2006

GS1

- We split the 4 betting rounds into two phases
  - Phase I (first 2 rounds) solved offline using approximate version of GameShrink followed by LP
    - Assuming rollout
  - Phase II (last 2 rounds):
    - abstractions computed offline
      - betting history doesn't matter & suit isomorphisms
    - *real-time* equilibrium computation using anytime LP
      - updated hand probabilities from Phase I equilibrium (using betting histories and community card history):

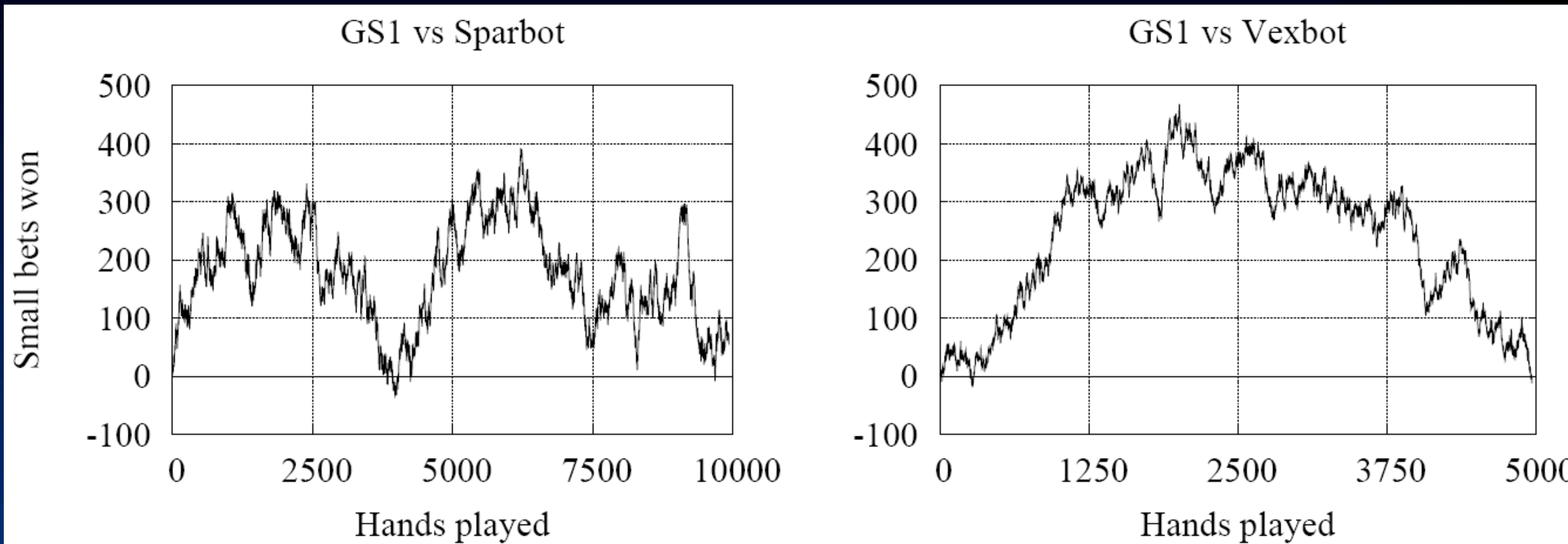
$$\Pr[\theta_i \mid h, s_i] = \frac{\Pr[h \mid \theta_i, s_i]\Pr[\theta_i]}{\Pr[h \mid s_i]} = \frac{\Pr[h \mid \theta_i, s_i]\Pr[\theta_i]}{\sum_{\theta_i' \in \Theta} \Pr[h \mid \theta_i', s_i]}$$

      - s_i is player i's strategy, h is an information set

Some additional techniques used

- Precompute several databases
- Conditional choice of primal vs. dual simplex for real-time equilibrium computation
  - Achieve anytime capability for the player that is us
- Dealing with running off the equilibrium path

GS1 results



- *Sparbot*: Game-theory-based player, manual abstraction
- *Vexbot*: Opponent modeling, miximax search with statistical sampling
- *GS1* performs well, despite using very little domain-knowledge and no adaptive techniques
  - No statistical significance

GS2 [Gilpin & S., AAMAS'07]

- Original *GameShrink* is "greedy" when used as an approximation algorithm => lopsided abstractions
- *GS2* instead finds abstraction via clustering & IP
  - Round by round starting from round 1
  - Operates in signal tree of one player's & common signals at a time
- Other ideas in *GS2*:
  - Overlapping phases so Phase I would be less myopic
    - Phase I = round 1, 2, and 3; Phase II = rounds 3 and 4
  - Instead of assuming rollout at leaves of Phase I (as was done in *SparBot* and *GS1*), use statistics to get a more accurate estimate of how play will go

GS2

2/2006 – 7/2006

[Gilpin & S., AAMAS'07]

Optimized approximate abstractions

- Original version of *GameShrink* is "greedy" when used as an approximation algorithm => lopsided abstractions
- *GS2* instead finds an abstraction via clustering & IP
- For round 1 in signal tree, use 1D k-means clustering
  - Similarity metric is win probability (ties count as half a win)
- For each *round* 2..3 of signal tree:
  - For each group i of hands (children of a parent at *round* – 1):
    - use 1D k-means clustering to split group i into k_i abstract "states"
    - for each value of k_i, compute expected error (considering hand probs)
  - IP decides how many children different parents (from *round* – 1) may have: Decide k_i's to minimize total expected error, subject to $\sum_i k_i \leq K_{round}$
    - K_{round} is set based on acceptable size of abstracted game
    - Solving this IP is fast in practice

Phase I (first three rounds)

- Allowed 15, 225, and 900 abstracted states in rounds 1, 2, and 3, respectively
- Optimizing the approximate abstraction took 3 days on 4 CPUs
- LP took 7 days and 80 GB using CPLEX’s barrier method

Phase I (first three rounds)

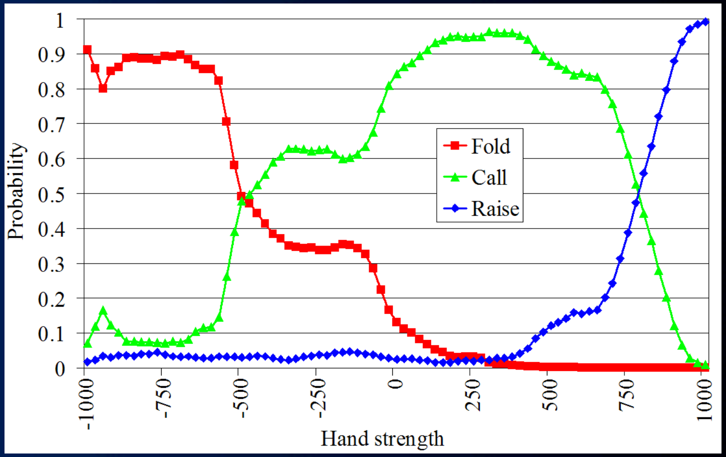
- Optimized abstraction
  - Round 1
    - There are 1,326 hands, of which 169 are strategically different
    - We allowed 15 abstract states
  - Round 2
    - There are 25,989,600 distinct possible hands
      - *GameShrink* (in lossless mode for Phase I) determined there are ~10^6 strategically different hands
    - Allowed 225 abstract states
  - Round 3
    - There are 1,221,511,200 distinct possible hands
    - Allowed 900 abstract states

- Optimizing the approximate abstraction took 3 days on 4 CPUs

- LP took 7 days and 80 GB using CPLEX's barrier method

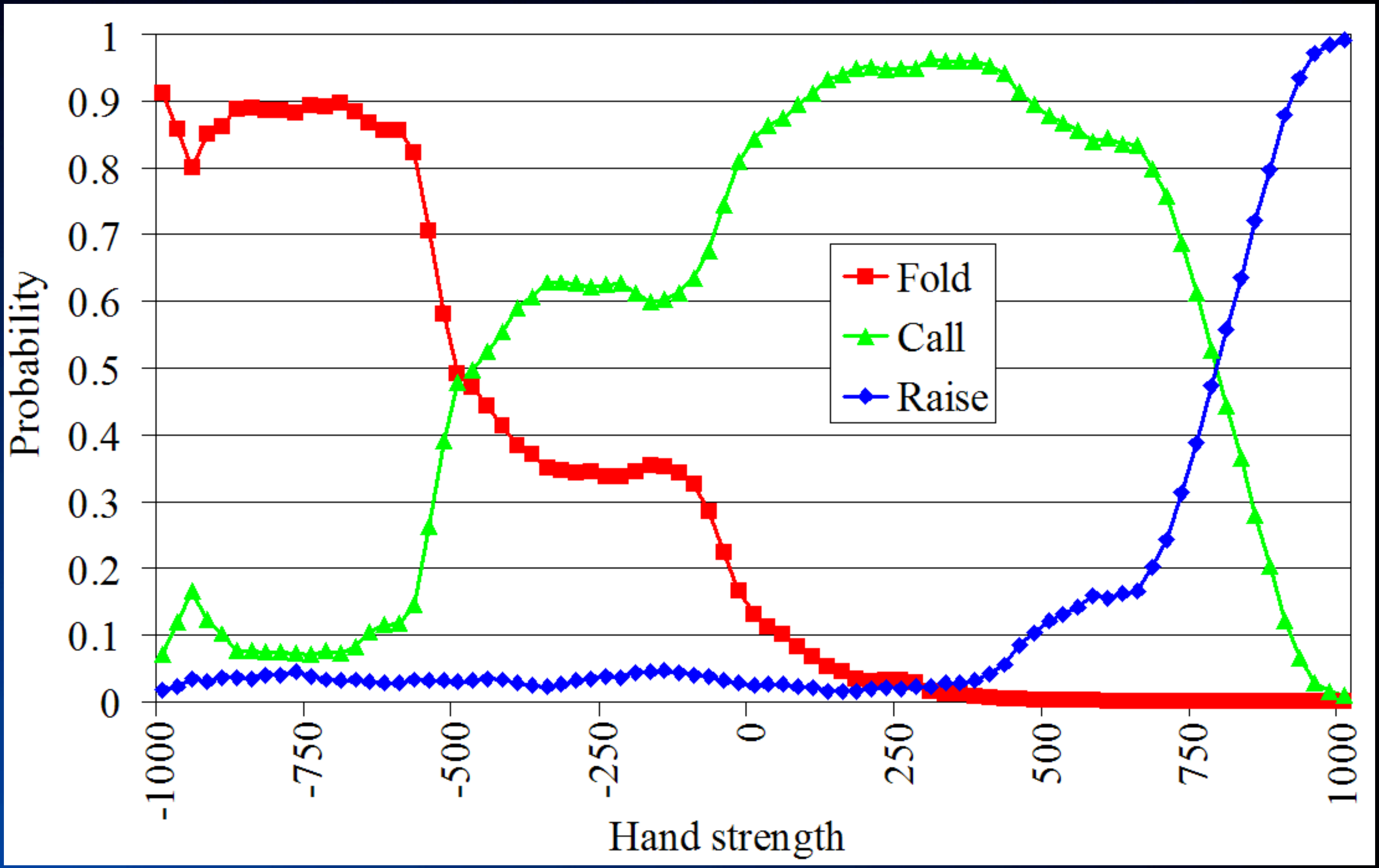
Mitigating effect of round-based abstraction (i.e., having 2 phases)

- For leaves of Phase I, GS1 & SparBot assumed rollout
- Can do better by estimating the actions from later in the game (betting) using statistics
- For each possible hand strength and in each possible betting situation, we stored the probability of each possible action
  - Mine history of how betting has gone in later rounds from 100,000’s of hands that SparBot played
  - E.g. of betting in 4th round
    - Player 1 has bet. Player 2’s turn



Example of betting in 4th round

Player 1 has bet. Player 2 to fold, call, or raise



Phase II (rounds 3 and 4)

- Abstraction computed using the same optimized abstraction algorithm as in Phase I
- Equilibrium solved in real time (as in *GS1)*
  - Beliefs for the beginning of Phase II determined using Bayes rule based on observations and the computed equilibrium strategies from Phase I

Precompute several databases

- `db5`: possible wins and losses (for a single player) for every combination of two hole cards and three community cards (25,989,600 entries)
  - Used by *GameShrink* for quickly comparing the similarity of two hands
- `db223`: possible wins and losses (for both players) for every combination of pairs of two hole cards and three community cards based on a roll-out of the remaining cards (14,047,378,800 entries)
  - Used for computing payoffs of the Phase I game to speed up the LP creation
- `handval`: concise encoding of a 7-card hand rank used for fast comparisons of hands (133,784,560 entries)
  - Used in several places, including in the construction of `db5` and `db223`
- Colexicographical ordering used to compute indices into the databases allowing for very fast lookups

GS2 experiments

Opponent	Series won by *GS2*	Win rate (small bets per hand)
GS1	38 of 50 p=.00031	+0.031
Sparbot	28 of 50 p=.48	+0.0043
Vexbot	32 of 50 p=.065	-0.0062

GS3

8/2006 – 3/2007

[Gilpin, S. & Sørensen AAAI'07]

Our poker bots 2008-2011 were generated with same abstraction algorithm

Entire game solved holistically

- We no longer break game into phases
  - Because our new equilibrium-finding algorithms can solve games of the size that stem from reasonably fine-grained abstractions of the entire game

- => better strategies & real-time end-game computation optional

Clustering + integer programming for abstraction

[Gilpin & Sandholm AAMAS'07]

- *GameShrink* is "greedy" when used as an approximation algorithm => lopsided abstractions
- For constructing *GS2,* abstraction was created via clustering & IP
- Operates in signal tree of one player's & common signals at a time

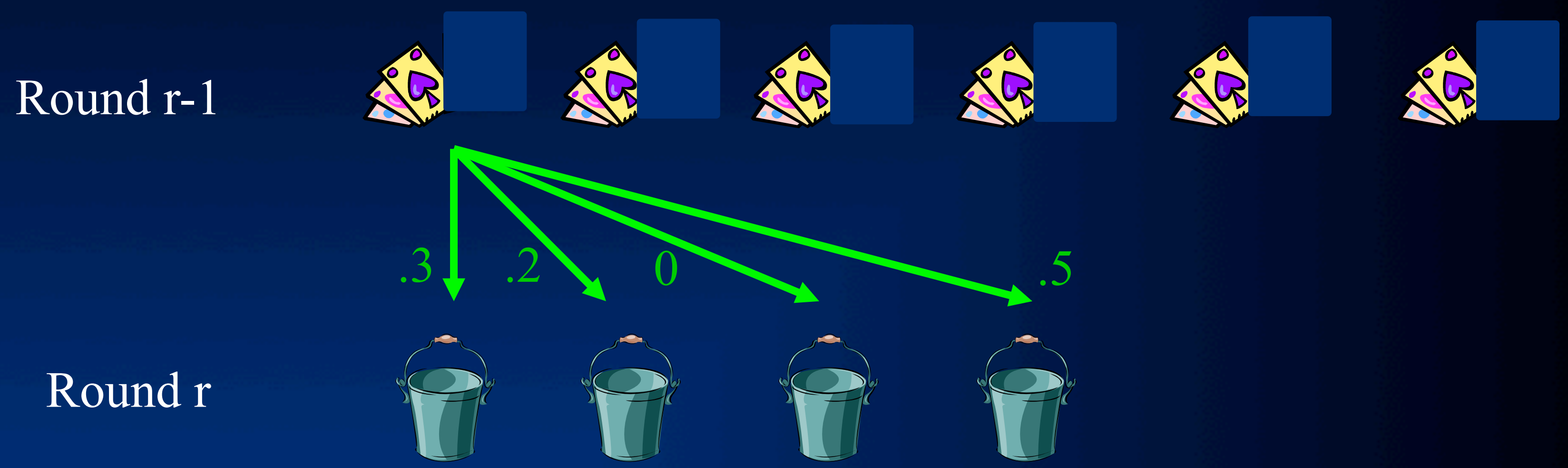
Potential-aware automated abstraction

[Gilpin, S. & Sørensen AAAI'07]

- All prior abstraction algorithms had EV (myopic probability of winning in poker) as the similarity metric
  - Doesn't capture *potential*
- Potential not only positive or negative, but also "multidimensional"
- GS3's abstraction algorithm captures potential …

- Idea: similarity metric between hands at round R should be based on the vector of probabilities of transitions to abstracted states at round R+1
  - E.g., L_1 norm
- In the last round, the similarity metric is simply probability of winning (assuming rollout)
- This enables a bottom

Bottom-up pass to determine abstraction for round 1



- Clustering using L_1 norm
  - Predetermined number of clusters, depending on size of abstraction we are shooting for
- In the last (4th) round, there is no more potential => we use probability of winning (e.g., assuming rollout) as similarity metric

Determining abstraction for round 2

- For each 1st-round bucket i:
  - Make a bottom-up pass to determine 3rd-round buckets, considering only hands compatible with i
  - For $k_i \in \{1, 2, \ldots, \max\}$
    - Cluster the 2nd-round hands into k_i clusters
      - based on each hand's histogram over 3rd-round buckets
- IP to decide how many children each 1st-round bucket may have, subject to $\sum_i k_i \leq K_2$
  - Error metric for each bucket is the sum of L_2 distances of the hands from the bucket's centroid
  - Total error to minimize is the sum of the buckets' errors
    - weighted by the probability of reaching the bucket

Determining abstraction for round 3

- Done analogously to how we did round 2

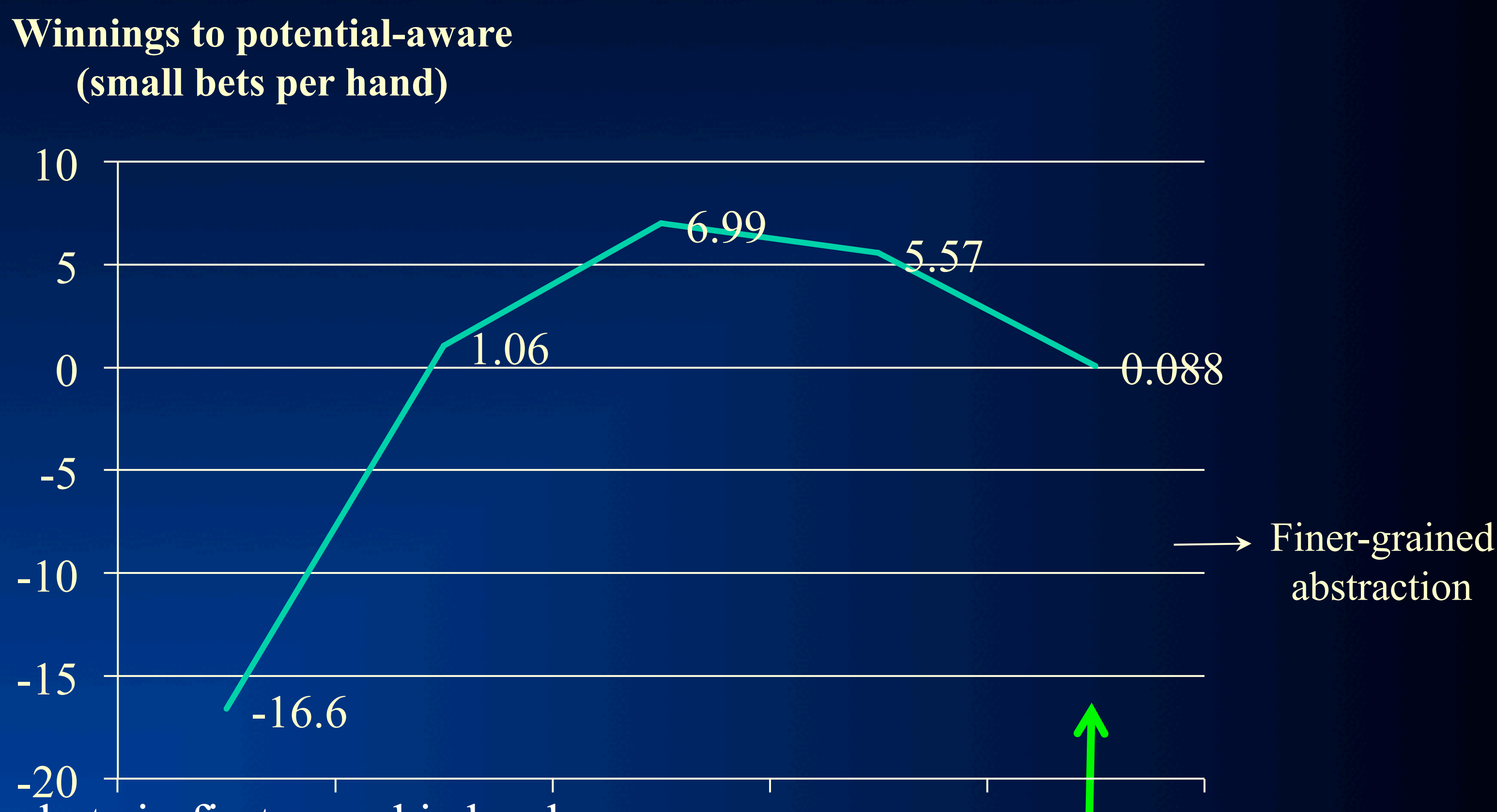
Determining abstraction for round 4

- Done analogously, except that now there is no potential left, so clustering is done based on probability of winning
- Now we have finished the abstraction!

Potential-aware vs win-probability-based abstraction

[Gilpin & S., AAAI-08]

- Both use clustering and IP
- Experiment on Rhode Island Hold'em => Abstracted game solved exactly



13 buckets in first round is lossless

Potential-aware becomes lossless,
win-probability-based is as good as it gets, *never* lossless

Potential-aware vs win-probability-based abstraction

[Gilpin & S., AAAI-08 & new]

- Both use clustering and IP
- Experiment conducted on Heads-Up Rhode Island Hold'em
  - Abstracted game solved exactly

Granularity	EB payoff versus EB^2	EB payoff versus PA	EB^2 payoff versus EB	EB^2 payoff versus PA	PA payoff versus EB	PA payoff versus EB^2
13-25-125	0.1490	16.6223	-0.1490	17.0938	-16.6223	-17.0938
13-50-250	-0.1272	-1.0627	0.1272	-0.5200	1.0627	0.5200
13-75-500						
13-100-750	0.2340	-6.9880	-0.2340	-7.1448	6.9880	7.1448
13-125-1000						
13-150-1250	0.1813	-5.5707	-0.1813	-5.6879	5.5707	5.6879
13-175-1500						
13-205-1774	0.0000	-0.0877	0.0000	-0.0877	0.0877	0.0877

13 buckets in first round is lossless

Potential-aware becomes lossless,
win-probability-based is as good as it gets, *never* lossless

Other forms of lossy abstraction

- Phase-based abstraction
  - Uses observations and equilibrium strategies to infer priors for next phase
  - Uses some (good) fixed strategies to estimate leaf payouts at non-last phases [Gilpin & Sandholm AAMAS-07]
  - Supports real-time equilibrium finding [Gilpin & Sandholm AAMAS-07]
    - *Grafting* [Waugh et al. 2009] as an extension
- Action abstraction
  - What if opponents play outside the abstraction?
  - Multiplicative action similarity and probabilistic reverse model [Gilpin, Sandholm, & Sørensen AAMAS-08, Risk & Szafron AAMAS-10]

Game abstraction is nonmonotonic

Defender

		A	Between	B
Attacker	A	0, 2	1, 1	2, 0
	B	2, 0	1, 1	0, 2

In each equilibrium:
- Attacker randomizes 50-50 between A and B
- Defender plays A w.p. p, B w.p. p, and Between w.p. $1-2p$
- There is an equilibrium for each $p \in [0, ½]$

An abstraction:

	A	Between	B
A	0, 2	1, 1	2, 0

Defender would choose A, but that is far from equilibrium in the original game where attacker would choose B

Coarser abstraction:

	Between	B
A	1, 1	2, 0

Defender would choose Between. That is an equilibrium in the original game

- Such "abstraction pathologies" also in small poker games [Waugh et al. AAMAS-09]
- We present the first lossy game abstraction algorithm with bounds
  - Contradiction?

First lossy game abstraction algorithms with bounds

[Sandholm and Singh EC-12]

- Recognized open problem; tricky due to pathologies
- For both action and state abstraction; for finite stochastic games
- Evaluations from abstract game are near accurate:

PROPOSITION 4.1. $\forall \sigma', \forall s \in S_k, \forall i,$

$$|V_i^{\sigma^{\uparrow \sigma'}}(s) - W_i^{\sigma'}(h(s))| \leq f_{k,i} \stackrel{\text{def}}{=} \sum_{j=1}^{k} \epsilon_{j,i}^R + \sum_{j=1}^{k-1} \overline{W}_{j,i}^{\sigma'} \epsilon_j^T$$

- Regret is bounded:

THEOREM 5.1. *For any subgame perfect Nash equilibrium (SPNE) strategy σ'^* in M', the corresponding joint strategy $\sigma^{\uparrow \sigma'^*}$ in M has the property that*

$$\forall i, \forall s \in S_k, \forall \pi_i \in S \to A_i(S), V_i^{\langle \pi_i, \sigma_{-i}^{\uparrow \sigma'^*} \rangle}(s) \leq V_i^{\sigma^{\uparrow \sigma'^*}}(s) + 2k f_{k,i} \qquad (15)$$

where $\langle \pi_i, \sigma_{-i}^{\uparrow \sigma'^*} \rangle$ is the joint strategy in M that results from Agent i unilaterally deviating from $\sigma^{\uparrow \sigma'^*}$ to pure strategy π_i, and $f_{k,i}$ is as defined in Proposition 4.1.

First lossy game abstraction methods with bounds
[Sandholm and Singh EC-12]

- Recognized open problem; tricky due to pathologies
- For both action and state abstraction
- For stochastic games

Strategy evaluation in M and M’

- **LEMMA.** If game M and abstraction M’ are “close”, then the value for every strategy in M’ (when evaluated in M’) is close to the value of any corresponding lifted strategy in M when evaluated in M. Formally:

joint strategy

$$\forall i, \forall s \in S_k, \forall \sigma'$$

$$|V_i^{\sigma \uparrow \sigma'}(s) - V'^{\sigma'}_i(h(s))| \leq k[\varepsilon^R + kR_{\max}\varepsilon^T$$

Main abstraction theorem

- Given a subgame perfect Nash equilibrium σ^* in M'

- Let lifted strategy in M be $\sigma^{\uparrow\sigma^{*\prime}}$

- Then maximum gain by unilateral deviation by agent i is

$$\forall s \in S_k$$

$$2k \times k\left[\varepsilon^R + kR_{\max}\varepsilon^T\right]$$

First lossy game abstraction algorithms with bounds

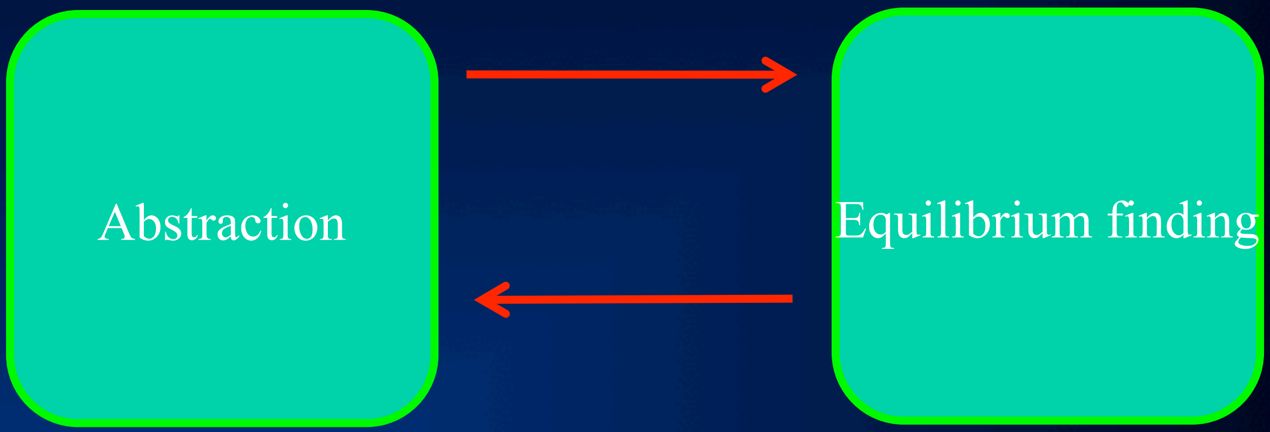
- Greedy algorithm that proceeds level by level from end of game
  - At each level, does either action or state abstraction first, then the other
  - Polynomial time (versus equilibrium finding being PPAD-complete)
- Integer linear program
  - Proceeds level by level from end of game; one ILP per level
    - Optimizing all levels simultaneously would be nonlinear
  - Does action and state abstraction simultaneously
  - Splits the allowed total error within level optimally
    - between reward error and transition probability error, and
    - between action abstraction and state abstraction
- **Proposition.** Both algorithms satisfy the given bounds on regret
- **Proposition.** Even with just action abstraction and just one level, finding the abstraction with the smallest number of actions that respects the regret bound is NP-complete (even with 2 agents)
- One of the first action abstraction algorithms
  - Totally different than the prior one [Hawkin et al. AAAI-11]

Role of this in modeling

- All modeling is abstraction!
- These are the first results that tie game modeling choices to solution quality in the actual setting

Strategy-based abstraction

[unpublished]



Equilibrium-finding algorithms

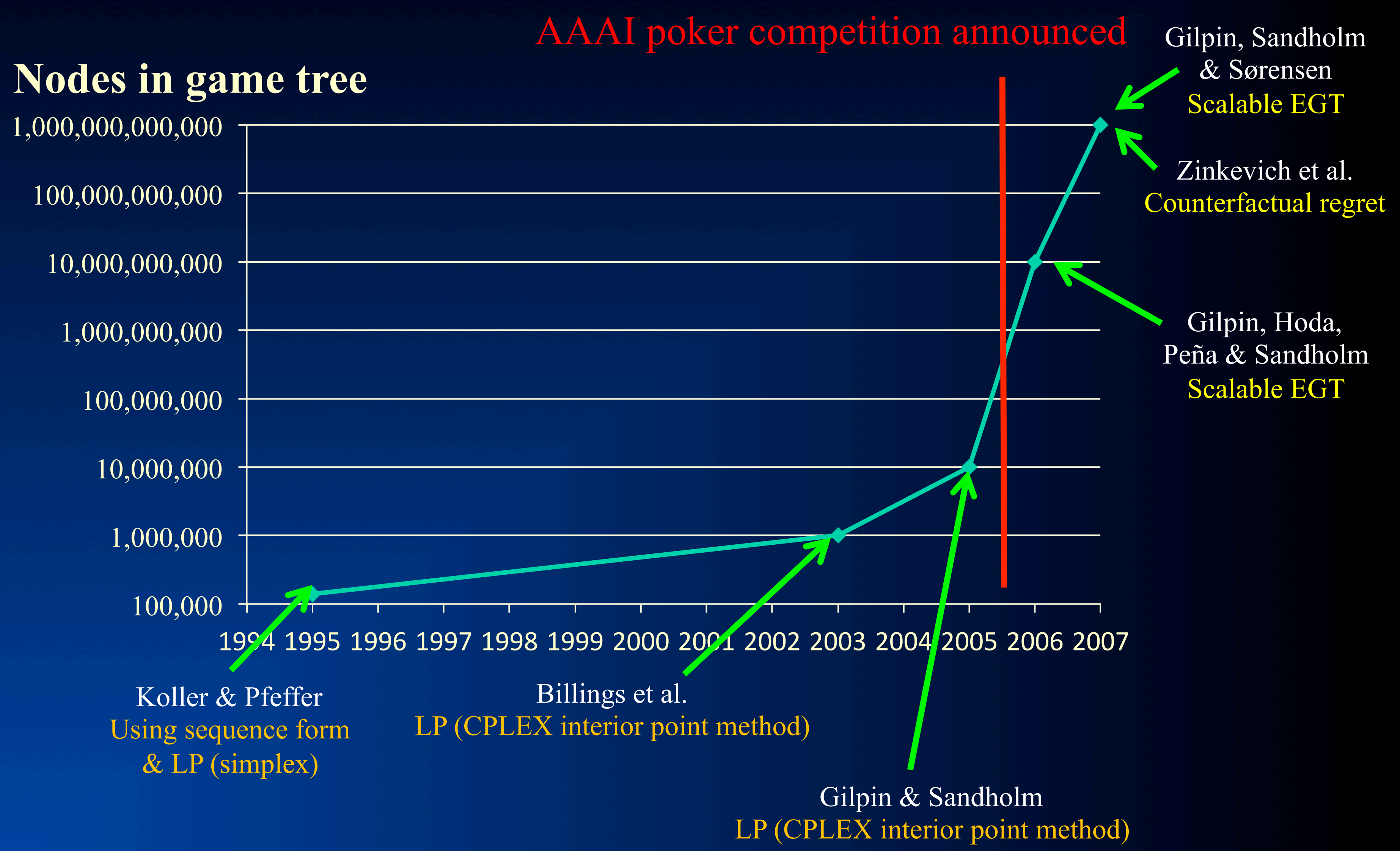
Solving the (abstracted) game

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

Scalability of (near-)equilibrium finding in 2-person 0-sum games

Manual approaches can only solve games with a handful of nodes



(Un)scalability of LP solvers

- Rhode Island Hold'em LP
  - 91,000,000 rows and columns
  - After *GameShrink,* 1,200,000 rows and columns, and 50,000,000 non-zeros
  - CPLEX's barrier method uses 25 GB RAM and 8 days
- Texas Hold'em poker much larger
  - => would need to use extremely coarse abstraction
- Instead of LP, can we solve the equilibrium-finding problem in some other way?

Excessive gap technique (EGT)

- Best general LP solvers only scale to10^7..10^8 nodes. Can we do better?
- Usually, gradient-based algorithms have poor $O(1/\varepsilon^2)$ convergence, but…
- **Theorem** [Nesterov 05]. Gradient-based algorithm, EGT (for a class of *minmax problems*) that finds an ε-equilibrium in $O(1/\varepsilon)$ iterations
- **Theorem** [Hoda, Gilpin, Pena & S., *Mathematics of Operations Research* 2010]. Nice prox functions can be constructed for sequential games

Scalable EGT [Gilpin, Hoda, Peña, S., WINE'07, *Math. Of OR* 2010]

Memory saving in poker & many other games

- Main space bottleneck is storing the game's payoff matrix A
- **Definition.** Kronecker product

$$X \in \mathbb{R}^{m\times n}, Y \in \mathbb{R}^{p\times q}, \qquad X \otimes Y = \begin{bmatrix} x_{11}Y & \cdots & x_{1n}Y \\ \vdots & \ddots & \vdots \\ x_{m1}Y & \cdots & x_{mn}Y \end{bmatrix} \in \mathbb{R}^{mp\times nq}$$

- In Rhode Island Hold'em:

$$A = \begin{bmatrix} A_1 & & \\ & A_2 & \\ & & A_3 \end{bmatrix}$$

- Using independence of card deals and betting options, can represent this as

$$A_1 = F_1 \otimes B_1 \qquad A_2 = F_2 \otimes B_2 \qquad A_3 = F_3 \otimes B_3 + S \otimes W$$

- F_r corresponds to sequences of moves in round r that end in a fold
- S corresponds to sequences of moves in round 3 that end in a showdown
- B_r encodes card buckets in round r
- W encodes win/loss/draw probabilities of the buckets

Memory usage

Instance	**CPLEX barrier**	**CPLEX simplex**	**Our method**
Losslessly abstracted Rhode Island Hold'em	25.2 GB	>3.45 GB	0.15 GB
Lossily abstracted Texas Hold'em	>458 GB	>458 GB	2.49 GB

Memory usage

Instance	CPLEX barrier	CPLEX simplex	Our method
10k	0.082 GB	>0.051 GB	0.012 GB
160k	2.25 GB	>0.664 GB	0.035 GB
Losslessly abstracted RI Hold’em	25.2 GB	>3.45 GB	0.15 GB
Lossily abstracted TX Hold’em	>458 GB	>458 GB	2.49 GB

Scalable EGT [Gilpin, Hoda, Peña, S., WINE'07, *Math. Of OR* 2010]

Speed

- Fewer iterations
  - With *Euclidean prox fn*, gap was reduced by an order of magnitude more (at given time allocation) compared to *entropy-based prox fn*
  - Heuristics that speed things up in practice while preserving theoretical guarantees
    - Less conservative shrinking of μ_1 and μ_2
      - Sometimes need to reduce (halve) τ
    - Balancing μ_1 and μ_2 periodically
      - Often allows reduction in the values
    - Gap was reduced by an order of magnitude (for given time allocation)
- Faster iterations
  - Parallelization in each of the 3 matrix-vector products in each iteration => near-linear speedup

Our successes with these approaches in 2-player Texas Hold'em

- AAAI-08 Computer Poker Competition
  - Won Limit bankroll category
  - Did best in terms of bankroll in No-Limit
- AAAI-10 Computer Poker Competition
  - Won bankroll competition in No-Limit

Iterated smoothing

[Gilpin, Peña & S., AAAI-08, *Mathematical Programming*, to appear]

- Input: Game and ε_{target}
- Initialize strategies x and y arbitrarily
- $\varepsilon \leftarrow \varepsilon_{target}$
- repeat
  - $\varepsilon \leftarrow gap(x, y) / e$
  - $(x, y) \leftarrow SmoothedGradientDescent(f, \varepsilon, x, y)$
  - until $gap(x, y) < \varepsilon_{target}$

$$O(1/\varepsilon) \quad \rightarrow \quad O(\log(1/\varepsilon))$$

Caveat: condition number.

Algorithm applies to all linear programming.

Matches iteration bound of interior point methods, but unlike them, is scalable for memory.

Solving GS3's four-round model

[Gilpin, Sandholm & Sørensen AAAI'07]

- Computed abstraction with
  - 20 buckets in round 1
  - 800 buckets in round 2
  - 4,800 buckets in round 3
  - 28,800 buckets in round 4
- Our version of excessive gap technique used 30 GB RAM
  - (Simply representing as an LP would require 32 TB)
  - Outputs new, improved solution every 2.5 days
  - 4 1.65GHz CPUs: 6 months to gap 0.028 small bets per hand

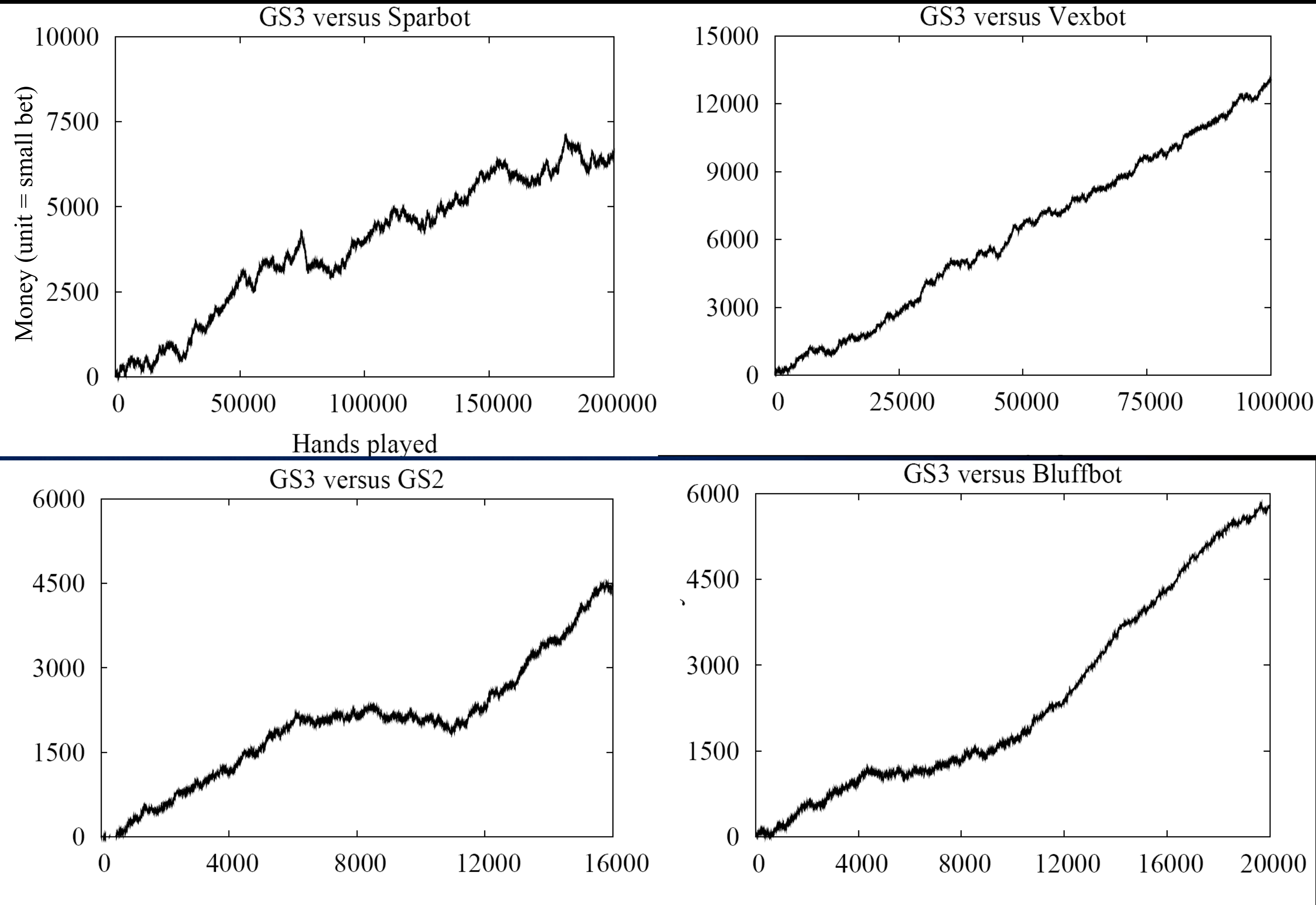
AAAI Computer Poker Competitions won

- 2008
  - *GS4* won Limit Texas Hold’em bankroll category
    - Played 4-4 in pairwise comparisons. 4th of 9 in elimination category
  - *Tartanian* did best in terms of bankroll in No-Limit Texas Hold’em
    - 3rd out of 4 in elimination category
- 2010
  - *Tartanian4* won Heads-Up No-Limit Texas Hold'em bankroll category
    - 3rd in Heads-Up No-Limit Texas Hold'em bankroll instant run-off category


Read this brilliant and timely book.

Going live with $313 million on PokerStars.com

- April fools!



All wins are statistically significant at the 99.5% level

Comparison to prior poker AI

- Rule-based
  - Limited success in even small poker games
- Simulation/Learning
  - Do not take multi-agent aspect into account
- Game-theoretic
  - Small games
  - Manual abstraction [Billings et al. IJCAI-03]
  - **Ours**
    - **Automated abstraction**
    - **Custom solver for finding Nash equilibrium**
    - **Domain independent**

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

Purification and thresholding

[Ganzfried, S. & Waugh, AAMAS-12]

- *Thresholding*: Rounding the probabilities to 0 of those strategies whose probabilities are less than c (and rescaling the other probabilities)
  - *Purification* is thresholding with c=0.5

- **Proposition** (performance of strategy from abstract game against equilibrium strategy in actual game): Any of the 3 approaches (standard approach, thresholding (for any c), purification) can beat any other by arbitrarily much depending on the game
  - Holds for any equilibrium-finding algorithm for one approach and any equilibrium-finding algorithm for the other

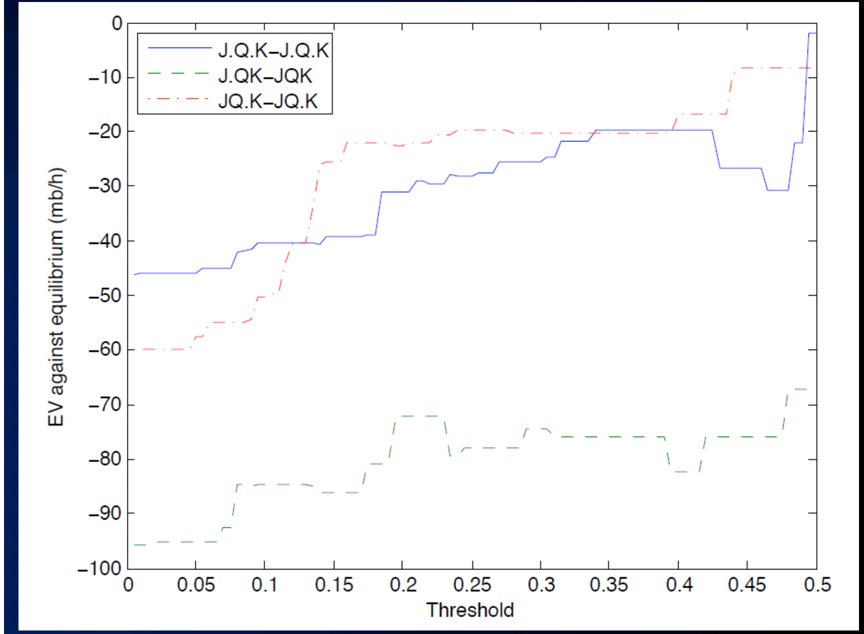
Experiments on random matrix games

- 2-player 4x4 zero-sum games
- Abstraction that simply ignores last row and last column
- Purified eq strategies from abstracted game beat non-purified eq strategies from abstracted game at 95% confidence level when played on the unabstracted game

Experiments on Leduc Hold'em

Strategy	Base EV	Purified EV	Improvement
JQ.K-J.QK	-119.46	-37.75	81.71
J.QK-full	-115.63	-41.83	73.80
J.QK-J.Q.K	-96.66	-27.35	69.31
JQ.K-J.Q.K	-96.48	-28.76	67.71
JQ.K-full	-99.30	-39.13	60.17
JQ.K-JQK	-80.14	-24.50	55.65
JQ.K-JQ.K	-59.97	-8.31	51.66
J.Q.K-J.QK	-60.28	-13.97	46.31
J.Q.K-J.Q.K	-46.23	-1.86	44.37
J.Q.K-JQ.K	-44.61	-3.85	40.76
full-JQK	-43.80	-10.95	32.85
J.QK-J.QK	-96.60	-67.42	29.18
J.QK-JQK	-95.69	-67.14	28.55
full-J.QK	-52.94	-24.55	28.39
J.QK-JQ.K	-77.86	-52.62	25.23
J.Q.K-full	-68.10	-46.43	21.66
full-JQ.K	-55.52	-36.38	19.14
full-J.Q.K	-51.14	-40.32	10.82
JQK-J.QK	-282.94	-279.44	3.50
JQK-full	-273.87	-279.99	-6.12
JQK-J.Q.K	-258.29	-279.99	-21.70
J.Q.K-JQK	-156.35	-188.00	-31.65
JQK-JQK	-386.89	-433.64	-46.75
JQK-JQ.K	-274.69	-322.41	-47.72

Table 2: Effects of purification on performance of abstract strategies against an equilibrium opponent in mb/h.



Experiments on no-limit Texas Hold'em

- We submitted bot Y to the AAAI-10 bankroll competition; it won
- We submitted bot X to the instant run-off competition; finished 3[rd]

	Bot 1	Bot 2	Bot 3	Bot 4	Bot 5	Bot 6	Bot X	Bot Y
Bot X	5334 $\pm$ 109	8431 $\pm$ 156	-248 $\pm$ 49	-364 $\pm$ 42	108 $\pm$ 46	-42 $\pm$ 38		-80 $\pm$ 23
Bot Y	4754 $\pm$ 107	8669 $\pm$ 168	-122 $\pm$ 38	-220 $\pm$ 39	159 $\pm$ 40	13 $\pm$ 33	80 $\pm$ 23	

Table 3: Results from a recent AAAI computer poker competition for 2-player no limit Texas Hold'em. Values are in milli big blinds per hand (from the row player's perspective) with 95% confidence intervals shown. Bot X and bot Y both use the same abstraction and equilibrium-finding algorithms. The only difference is that X uses thresholding with a threshold of 0.15, and Y uses purification.

Experiments on limit Texas Hold'em

- Worst-case exploitability

Threshold	Our 2010 competition bot Exploitability	U. Alberta 2010 competition bot Exploitability
None	463.591	**235.209**
0.05	326.119	243.705
0.15	**318.465**	258.53
0.25	335.048	277.841
Purified	349.873	437.242

- Too much thresholding => not enough randomization
  => signal too much to the opponent

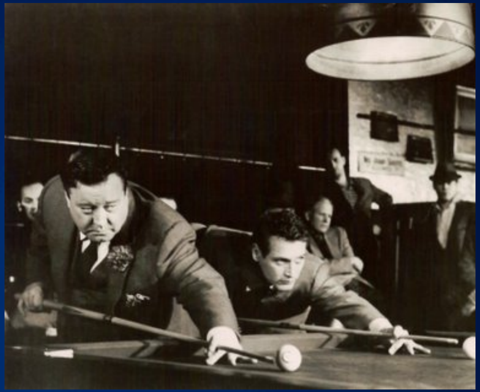
- Too little thresholding => strategy is overfit to the particular abstraction

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

Traditionally two approaches

- Game theory approach (abstraction+equilibrium finding)
  - Safe in 2-person 0-sum games
  - Doesn't maximally exploit weaknesses in opponent(s)
- Opponent modeling
  - *Get-taught-and-exploited problem* [Sandholm AIJ-07]



  - Needs prohibitively many repetitions to learn in large games (loses too much during learning)
    - Crushed by game theory approach in Texas Hold'em, even with just 2 players and limit betting
    - Same tends to be true of no-regret learning algorithms

Let's hybridize the two approaches
[Ganzfried & Sandholm AAMAS-11]

- Start playing based on game theory approach
- As we learn opponent(s) deviate from equilibrium, start adjusting our strategy to exploit their weaknesses

Deviation-Based Best Response (DBBR) algorithm (can be generalized to multi-player non-zero-sum)

Compute an approximate equilibrium of the game
Maintain counters from observing opponent's play throughout the match.
for $n = 1$ **to** $|PH_{-i}|$ **do**
  Compute posterior action probabilities at n.
  Compute posterior bucket probabilities at n.
  Compute full model of opponent's strategy at n.
end for
return Best response to the opponent model.

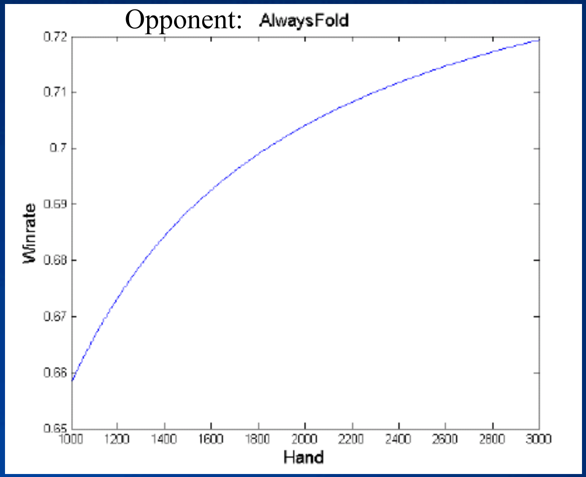
Public history sets

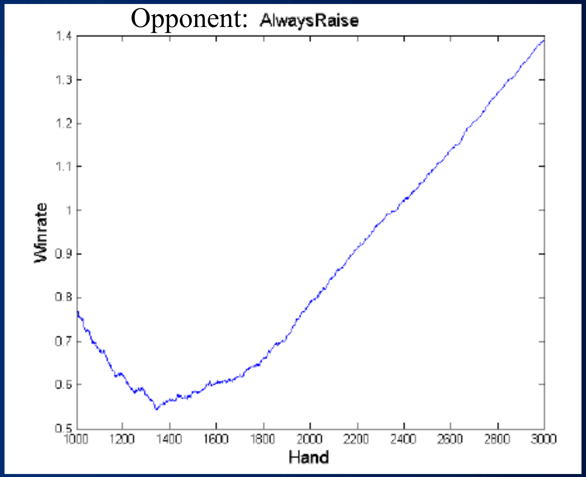
Dirichlet prior

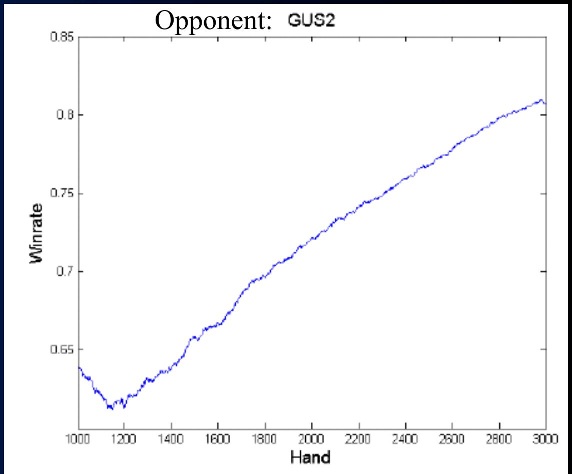
- Many ways to determine opponent's "best" strategy that is consistent with bucket probabilities
  - L_1 or L_2 distance to equilibrium strategy
  - Custom weight-shifting algorithm
  - ...

Experiments

- Significantly outperforms game-theory-based base strategy (*GS5*) in 2-player limit Texas Hold'em against
  - trivial opponents
  - weak opponents from AAAI computer poker competitions
- Don't have to turn this on against strong opponents
- Examples of winrate evolution:





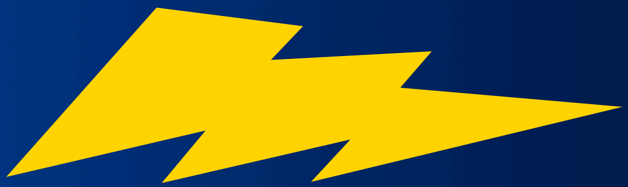


Safe opponent exploitation

[Ganzfried & Sandholm EC-12]

- Definition. *Safe* strategy achieves at least the value of the (repeated) game in expectation

- Is safe exploitation possible (beyond selecting among equilibrium strategies)?



When can opponent be exploited safely?

- ~~Opponent played an (iterated weakly) dominated strategy?~~

	L	M	R
U	3	2	10
D	2	3	0

A game with a gift strategy that is not weakly iteratively dominated.

- ~~Opponent played a strategy that isn't in the support of any eq?~~

	L	R
U	0	0
D	-2	1

Strategy R is not in the support of an equilibrium for player 2, but is also not a gift.

- **Definition.** We received a *gift* if the opponent played a strategy such that we have an equilibrium strategy for which the opponent's strategy is not a best response
- **Theorem.** Safe exploitation is possible in a game iff the game has gifts
- E.g., rock-paper-scissors doesn't have gifts
- Can determine in polytime whether a game has gifts

Exploitation algorithms (both for matrix and sequential games)

1. Risk what you've won so far
   – Doesn't differentiate whether winnings are due to opponent's mistakes (gifts) or our luck
2. Risk what you've won so far in expectation (over nature's & own randomization), i.e., risk the gifts received
   – Assuming the opponent plays a nemesis in states where we don't know
3. Best(-seeming) equilibrium strategy
4. Regret minimization between an equilibrium and opponent modeling algorithm
5. Regret minimization in the space of equilibria
6. Best equilibrium followed by full exploitation
7. Best equilibrium and full exploitation when possible

- **Theorem.** A strategy for a 2-player 0-sum game is safe iff it never risks more than the gifts received according to #2
- Can be used to make any opponent modeling algorithm safe
- No prior (non-eq) opponent exploitation algorithms are safe
- Experiments on Kuhn poker: #2 > #7 > #6 > #3
- Suffices to lower bound opponent's mistakes

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

>2 players

(Actually, our abstraction algorithms and opponent exploitation, presented earlier in this talk, apply to >2 players)

Computing Equilibria in Multiplayer Stochastic Games of Imperfect Information

Sam Ganzfried and Tuomas Sandholm
Computer Science Department
Carnegie Mellon University

Stochastic games

- $N = \{1,\ldots,n\}$ is finite set of players
- S is finite set of states
- $A(s) = (A_1(s),\ldots, A_n(s))$, where $A_i(s)$ is set of actions of player i at state s
- $p_{s,t}(a)$ is probability we transition from state s to state t when players follow action vector a
- r(s) is vector of payoffs when state s is reached
- Undiscounted vs. discounted

- A stochastic game with one agent is a Markov Decision Process (MDP)

Stochastic game example: poker tournaments

- Important challenge problem in artificial intelligence and computational game theory
- Enormous strategy spaces:
  - Two-player limit Texas hold'em game tree has ~10^{18} nodes
  - Two-player no-limit Texas hold'em has ~ 10^{71} nodes
- Imperfect information (unlike, e.g., chess)
- Many players
  - Computing a Nash equilibrium in matrix games PPAD-complete for > 2 players
- Poker tournaments are undiscounted stochastic games

Rules of poker

- No-limit Texas hold’em
- Two private hole cards, five public community cards
  - Only best 5-card hand matters
- 4 rounds of betting: preflop, flop, turn, river
- Preflop investments: *small blind (SB) & big blind (BB)*
- Actions: fold, call, raise (any amount), go all-in

Poker tournaments

- Players pay entry fee (e.g., $10)
- Players given some number of chips (e.g., 1500)
- A player is eliminated when he has no more chips
  - The order of elimination determines the payouts
  - E.g., winner gets $50, 2nd place $30, 3rd place $20
  - Blinds escalate quickly
- Tournaments are stochastic games: each game state corresponds to a vector of stack sizes
- We study 3-player endgame with fixed high blinds
  - Potentially infinite duration

Jam/fold strategies

- All-in or fold preflop: no postflop play
- 169 strategically distinct starting hands (pocket pairs, unsuited non-pairs, suited non-pairs)
- For any given stack vector, the sizes of the players' strategy spaces are 2^{169}, 2^{2*169}, and 2^{3*169}
- With two players left, jam/fold strategies are near-optimal when blinds sufficiently high [Miltersen/Sörensen AAMAS '07]
  - They show that probability of winning is approximately equal to fraction of chips player has

VI-FP: Prior algorithm for equilibrium finding in multiplayer stochastic games [Ganzfried & Sandholm AAMAS '08]

- Initialize payoffs V_0 for all game states using ICM
- Repeat
  - Run "inner loop":
    - Assuming the payoffs V_t, compute an approximate equilibrium s_t at each non-terminal state (stack vector) using an extension of smoothed fictitious play to imperfect information games

$$s_{i,t} = \left(1 - \frac{1}{t}\right) s_{i,t-1} + \frac{1}{t} s'_{i,t}$$

  - Run "outer loop":
    - Compute the values V_{t+1} at all non-terminal states by using the probabilities from s_t and values from V_t
- until outer loop converges

Drawbacks of VI-FP

- Neither the inner nor outer loop guaranteed to converge
- Possible for outer-loop to converge to a non-equilibrium
  - Initialize the values to all three players of stack vectors with all three players remaining to $100
  - Initialize the stack vectors with only two players remaining according to ICM
  - Then everyone will fold (except the short stack if he is all-in), payoffs will be $100 to everyone, and the algorithm will converge in one iteration to a non-equilibrium profile

Ex post check

- Determine how much each player can gain by deviating from strategy profile s^* computed by VI-FP
- For each player, construct MDP M induced by the components of s^* for the other players
- Solve M using variant of policy iteration for our setting (next slide)
- Look at difference between the payoff of optimal policy in M and payoff under s^*

- Converged in just two iterations of policy iteration.
- No player can gain more than $0.049 (less than 0.5% of tournament entry fee) by deviating from s^*

Optimal MDP solving in our setting

- Our setting:
  - Objective is expected total reward
  - For all states s and policies p, the value of s under p is finite
  - For each state s there exists at least one available action a that gives nonnegative reward
- Value iteration: must initialize pessimistically
- Policy iteration:
  - Choose initial policy with nonnegative total reward
  - Choose minimal non-negative solution to system of equations in evaluation step (if there is a choice) $v(i) = r(i) + \sum_j p_{ij}^{\pi^n} v(j)$
  - If the action chosen for some state in the previous iteration is still among the optimal actions, select it again

New algorithms for equilibrium finding in multiplayer stochastic games

One algorithm from [Ganzfried & S., AAMAS-08, IJCAI-09]

Repeat until ε-equilibrium

At each state

Run fictitious play until regret < thres, given values of possible future states

Adjust values of all states (using modified policy iteration) in light of the new payoffs obtained

First algorithms for ε-equilibrium in large stochastic games for small ε

Proposition. If outer loop converges, the strategy profile is an equilibrium

Found ε-equilibrium for tiny ε in jam/fold strategies in 3-player No-Limit Texas Hold'em tournament (largest multiplayer game solved to small ε?)

Algorithms converged to an ε-equilibrium consistently and quickly despite not being guaranteed to do so -- new convergence guarantees?

PI-FP: Policy Iteration as outer loop

- Similar to VI-FP except value updates follow the evaluation step of policy iteration in our setting

Algorithm *VI-FP*(γ, δ)

V^0 = initializeValues()
diff $= \infty$
$i = 0$
while diff $> \delta$ **do**
    $i = i + 1$
    regret = ∞
    S = initializeStrategies()
    while regret $> \gamma$ **do**
        S = fictPlay()
        regret = maxRegret(S)
    end while
    V^i = getNewValues(V^{i-1},S)
    diff = maxDev(V^i, V^{i-1})
end while
return S

Algorithm *PI-FP*(γ, δ)

V^0 = initializeValues()
diff $= \infty$
$i = 0$
while diff $> \delta$ **do**
    $i = i + 1$
    regret = ∞
    S^0 = initializeStrategies()
    while regret $> \gamma$ **do**
        S^i = fictPlay()
        regret = maxRegret(S^i)
    end while
    M^i = createTransitionMatrix(S^i)
    V^i = evaluatePolicy(M^i)
    diff = maxDev(V^i, V^{i-1})
end while
return S^i

- Proposition: if the outer loop of PI-FP converges, then the final strategy profile is an equilibrium
  - Can recover from poor initialization since it uses values resulting from evaluating the policy, not the values from the initialization

FP-MDP: switching the roles of fictitious play and MDP-solving

- Again we prefer policy iteration to value iteration because it allows us to get a good warm start more easily
- Use policy iteration to perform best response calculation
- Use fictitious play to combine new best response with previous strategy
- Like VI-FP, FP-MDP can recover from poor initializations and can provably never converge to a non-equilibrium

Algorithm *FP-MDP*

S^0 = initializeStrategies()
$i = 0$
while termination criterion not met **do**
  M^i = constructMDP(S^i)
  S' = solveMDP(M^i)
  $S^{i+1} = \frac{i}{i+1} S^i + \frac{1}{i+1} S'$
  $i = i + 1$
end while
return S^i

FTPL-MDP: a polynomial-time algorithm for regret minimization

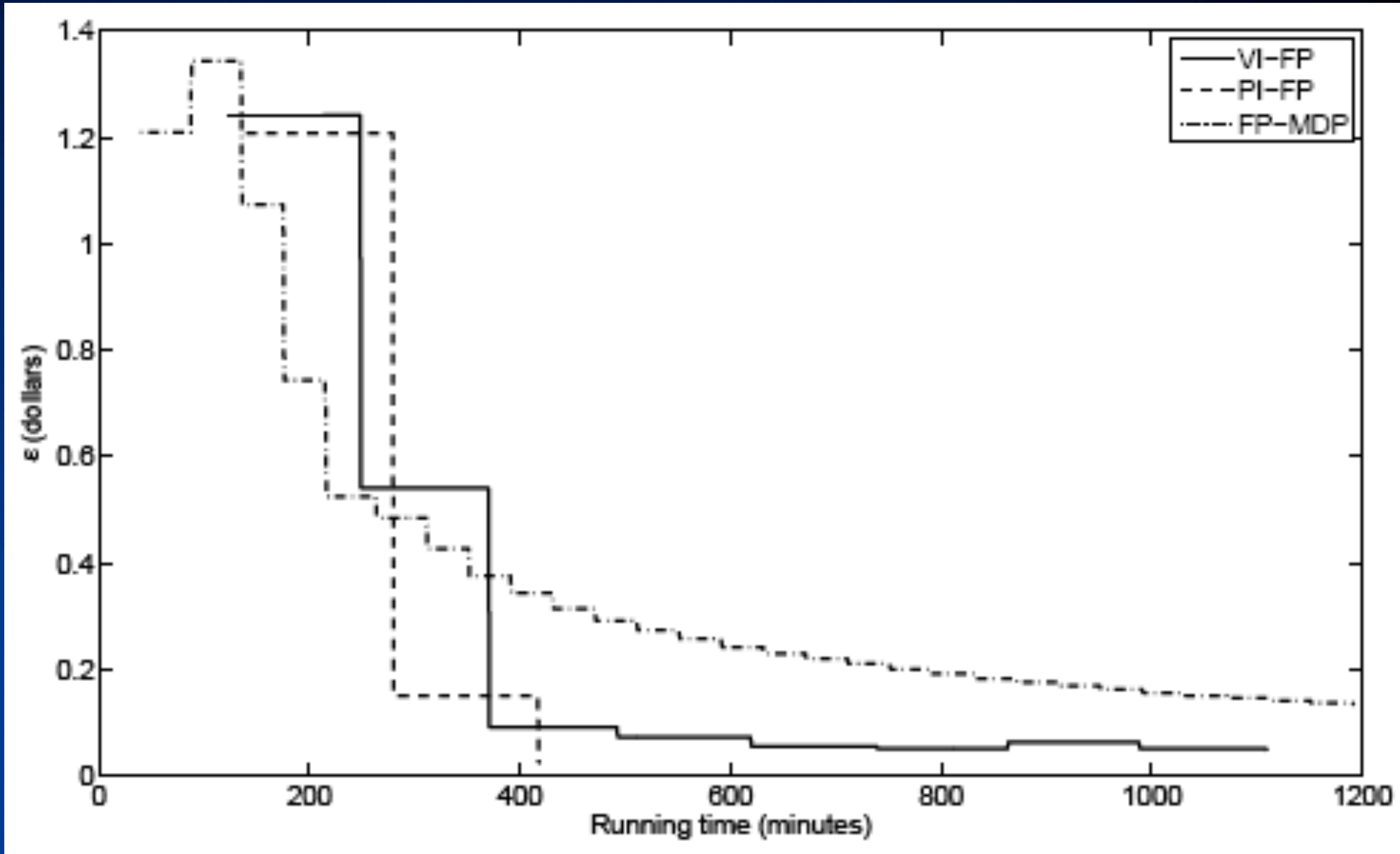
- Similar to FP-MDP
- Polynomial-time LP algorithm for MDP-solving in inner loop
- *Follow-the-perturbed-leader* algorithm for outer loop: like fictitious play, but add random noise before computing best response [Kalai & Vempala, JCSS ’05]
- Minimizes external regret in repeated game

Algorithm 6 *FTPL-MDP*

S^0 = initializeStrategies()
$i = 0$
while termination criterion not met **do**
  $\hat{S}^i$ = randomPerturbation(S^i)
  M^i = constructMDP($\hat{S}^i$)
  S' = solveMDP-LP(M^i)
  $S^{i+1} = \frac{i}{i+1} S^i + \frac{1}{i+1} S'$
  $i = i + 1$
end while
return S^i

Experimental Results

- Each data point corresponds to an outer loop iteration
- Target accuracy: \$0.05 = 0.1% of first place payoff
- PI-FP first to reach target accuracy, followed by VI-FP
- FP-MDP never reached target accuracy



Conclusions and future work

- Presented first algorithm for provably computing an ε-equilibrium of large stochastic games for small ε
- First provable near-equilibrium strategies for jam/fold poker tournament with more than 2 players
- Algorithms converged to an ε-equilibrium consistently and quickly despite not being guaranteed to do so
- Hopefully can lead to investigation of more general settings under which convergence properties can be proven
  – Fictitious play converged consistently despite not being guaranteed to do so
  – Outer loop of VI-FP converged despite not being guaranteed to do so
  – Maybe value iteration for solving MDP's can be proven to converge for some optimal initializations in this setting

Games with >2 players

- Matrix games:
  - 2-player zero-sum: solvable in polytime
  - >2 players zero-sum: PPAD-complete [Chen & Deng, 2006]
  - No previously known algorithms scale beyond tiny games with >2 players
- Stochastic games (undiscounted):
  - 2-player zero-sum: Nash equilibria exist
  - 3-player zero-sum: Existence of Nash equilibria still open

Poker tournaments

- Players buy in with cash (e.g., \$10) and are given chips (e.g., 1500) that have no monetary value
- Lose all you chips => eliminated from tournament
- Payoffs depend on finishing order (e.g., \$50 for 1st, \$30 for 2nd, \$20 for 3rd)
- Computational issues:
  - >2 players
  - Tournaments are stochastic games (potentially infinite duration): each game state is a vector of stack sizes (and also encodes who has the button)

Jam/fold strategies

- Jam/fold strategy: in the first betting round, go all-in or fold
- In 2-player poker tournaments, when blinds become high compared to stacks, provably near-optimal to play jam/fold strategies [Miltersen & Sørensen 2007]

- Solving a 3-player tournament [Ganzfried & Sandholm AAMAS'2008]
  - Compute an approximate equilibrium in jam/fold strategies
  - Strategy spaces 2^{169}, 2×2^{169}, 3×2^{169}
  - Algorithm combines
    - an extension of fictitious play to imperfect-information games
    - with a variant of value iteration
  - Our solution challenges *Independent Chip Model (ICM)* accepted by poker community
  - Unlike in 2-player case, tournament and cash game strategies differ substantially

Our first algorithm

- Initialize payoffs for all game states using heuristic from poker community (ICM)
- Repeat until "outer loop" converges
  - "Inner loop":
    - Assuming current payoffs, compute an approximate equilibrium at each state using fictitious play
    - Can be done efficiently by iterating over each player's information sets
  - "Outer loop":
    - Update the values with the values obtained by new strategy profile
    - Similar to value iteration in MDPs

Ex-post check

- Our algorithm is not guaranteed to converge, and can converge to a non-equilibrium (we constructed example)
- We developed an *ex-post* check to verify how much any player could gain by deviating [Ganzfried & Sandholm draft]
  - Constructs an undiscounted MDP from the strategy profile, and solves it using variant of policy iteration
  - Showed that no player could gain more than 0.1% of highest possible payoff by deviating from our profile

New algorithms [Ganzfried & Sandholm draft]

- Developed 3 new algorithms for solving multiplayer stochastic games of imperfect information
  - Unlike first algorithm, if these algorithms converge, they converge to an equilibrium
  - First known algorithms with this guarantee
  - They also perform competitively with the first algorithm
- The algorithms combine fictitious play variant from first algorithm with techniques for solving undiscounted MDPs (i.e., maximizing expected total reward)

Best one of the new algorithms

- Initialize payoffs using ICM as before
- Repeat until "outer loop" converges
  - "Inner loop":
    - Assuming current payoffs, compute an approximate equilibrium at each state using our variant of fictitious play as before
  - "Outer loop": update the values with the values obtained by new strategy profile S_t using a modified version of policy iteration:
    - Create the MDP M induced by others' strategies in S_t (and initialize using own strategy in S_t):
    - Run modified policy iteration on M
      - In the matrix inversion step, always choose the minimal solution
      - If there are multiple optimal actions at a state, prefer the action chosen last period if possible

Second new algorithm

- Interchanging roles of fictitious play and policy iteration:
  - Policy iteration used as inner loop to compute best response
  - Fictitious play used as outer loop to combine BR with old strategy

- Initialize strategies using ICM
- Inner loop:
  - Create MDP M induced from strategy profile
  - Solve M using policy iteration variant (from previous slide)
- Outer loop:
  - Combine optimal policy of M with previous strategy using fictitious play updating rule

Third new algorithm

- Using value iteration variant as the inner loop
- Again we use MDP solving as inner loop and fictitious play as outer loop
- Same as previous algorithm except different inner loop

- New inner loop:
  - Value iteration, but make sure initializations are pessimistic (underestimates of optimal values in the MDP)
  - Pessimistic initialization can be accomplished by matrix inversion using outer loop strategy as initialization in induced MDP

Outline

- Abstraction
- Equilibrium finding in 2-person 0-sum games
- Strategy purification
- Opponent exploitation
- Multiplayer stochastic games
- Leveraging qualitative models

Computing Equilibria by Incorporating Qualitative Models

Sam Ganzfried and Tuomas Sandholm
Computer Science Department
Carnegie Mellon University

Introduction

- Key idea: often it is much easier to come up with some aspects of an equilibrium than to actually compute one
- E.g., threshold strategies are optimal in many settings:
  - Sequences of take-it-or-leave-it offers
  - Auctions
  - Partnerships/contracts
  - Poker…
- We develop an algorithm for computing an equilibrium in imperfect-information games given a qualitative model of the structure of equilibrium strategies
  - Applies to both infinite and finite games, with 2 or more players

Continuous (i.e., infinite) games

- Games with infinite number of pure strategies
  - E.g., strategies correspond to amount of time, money, space (such as computational billiards)
- N is finite set of players
- S_i is (a potentially infinite) pure strategy space of player i
- u_i: $S \rightarrow R$ is utility function of player i
- **Theorem** [Fudenberg/Levine]: If strategy spaces are nonempty compact subsets of a metric space and payoff functions are continuous, then there exists a Nash equilibrium

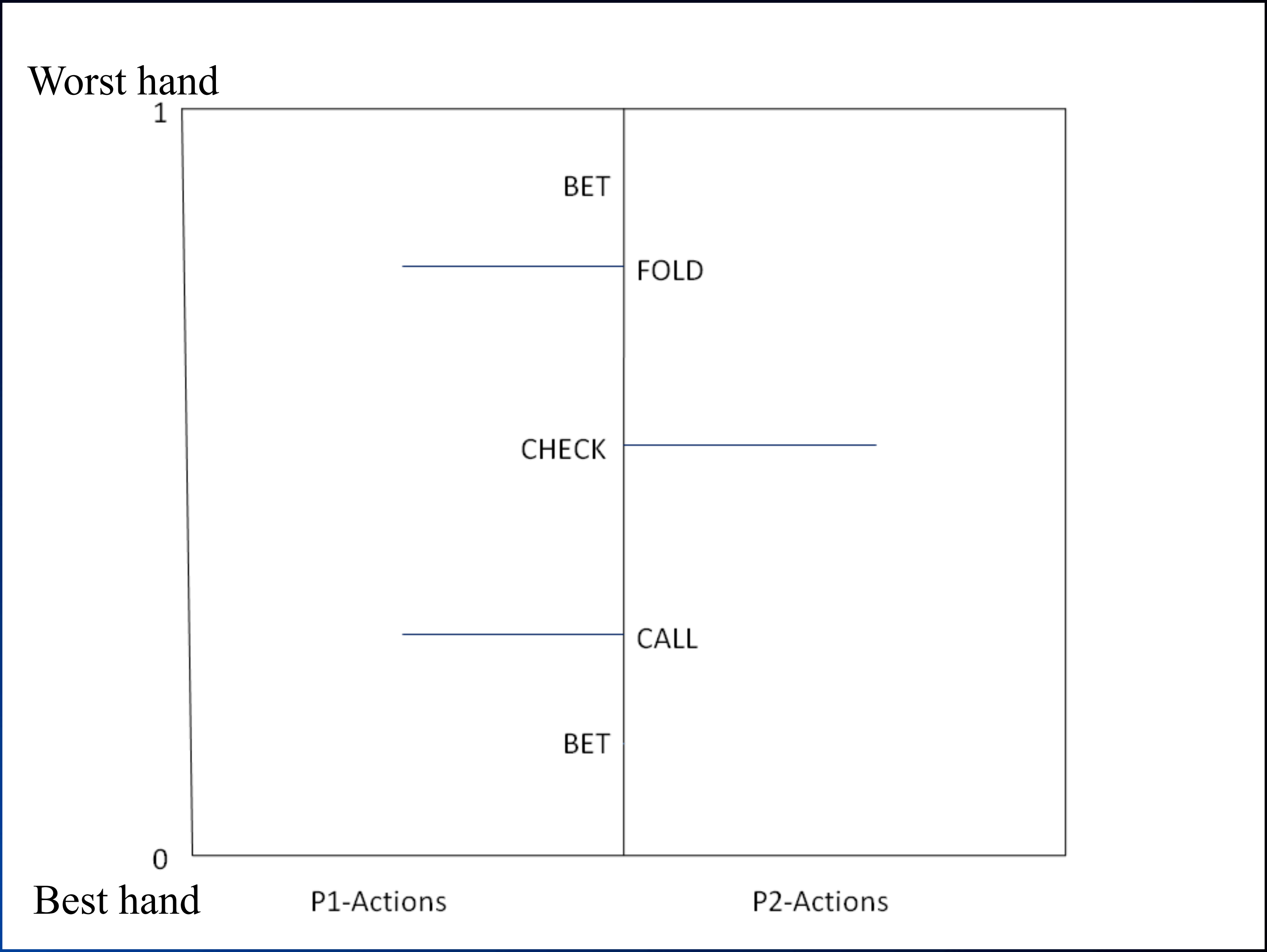
Poker example

- Two players given private signals x_1, x_2 independently and uniformly at random from [0,1]
- Pot initially has size P
- Player 1 can bet or check
- If player 1 checks, game is over and lower signal wins
- If player 1 bets, player 2 can call or fold
- If player 2 folds, player 1 wins
- If player 2 calls, player with lower private signal wins P+1, while other player loses 1

Example cont'd

- Strategy space of player 1: Set of measurable functions from [0,1] to {bet, check}
  - Similar for player 2
- **Proposition.** The strategy spaces are not compact
- **Proposition.** All strategies surviving iterated dominance must follow a specific threshold structure (on next slide)
- New strategy spaces are compact subsets of R
- **Proposition.** The utility functions are continuous
- Game can be solved by extremely simple procedure…

Example cont'd

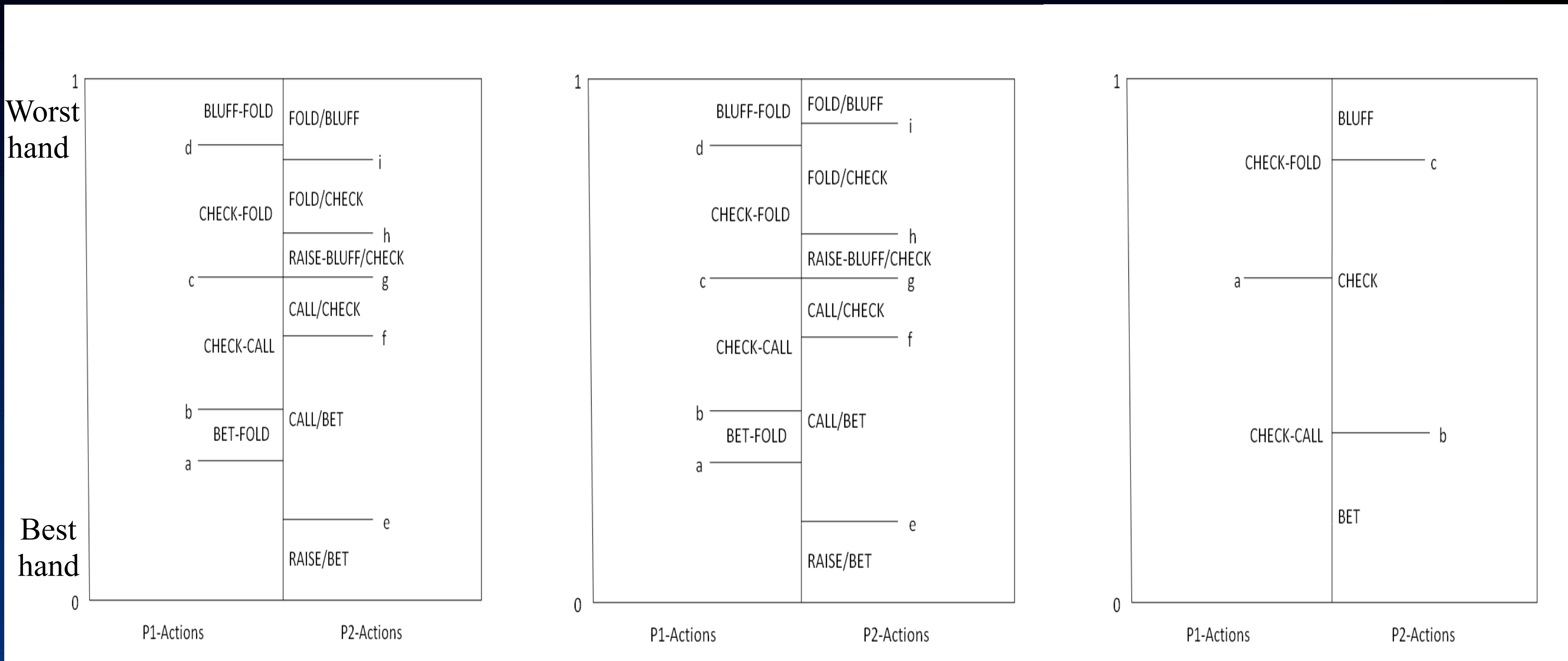


Setting: Continuous Bayesian games

[Ganzfried & Sandholm AAMAS-10 & newer draft]

- Finite set of players
- For each player i:
  - X_i is space of private signals (compact subset of R or discrete finite set)
  - C_i is finite action space
  - F_i: $X_i \rightarrow [0,1]$ is a piece-wise linear CDF of private signal
  - u_i: C x X $\rightarrow$ R is continuous, measurable, *type-order-based utility function*: utilities depend on the actions taken and order of agents' private signals (but not on the private signals themselves)

Qualitative models



Analogy to air combat

- Qualitative models can enable proving existence of equilibrium
- **Theorem.** Given F_1, F_2, and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium

Parametric models

- Way of dividing up signal space qualitatively into "action regions"
- $P = (T, Q, <)$
- T_i is number of regions of player i
- Q_i is sequence of actions of player i
- $<$ is partial ordering of the region thresholds across agents
- We saw that forcing strategies to conform to a parametric model can allow us to guarantee existence of an equilibrium and to compute one, when neither could be accomplished by prior techniques

Computing an equilibrium given a parametric model

- Parametric models => can prove existence of equilibrium
- Mixed-integer linear feasibility program
- Let $\{t_i\}$ denote union of sets of thresholds
- Real-valued variables: x_i corresponding to $F_1(t_i)$ and y_i to $F_2(t_i)$
- 0-1 variables: $z_{i,j} = 1$ implies $j\text{-}1 \leq t_i \leq j$
  - For this slide we assume that signals range 1, 2, …, k, but we have a MILFP for continuous signals also
  - Easy post-processor to get mixed strategies in case where individual types have probability mass
- Several types of constraints:
  - Indifference, threshold ordering, consistency
- **Theorem.** Given a candidate parametric model P, our algorithm outputs an equilibrium consistent with P if one exists. Otherwise it returns "no solution"

Works also for

- >2 players
  - Nonlinear indifference constraints => approximate by piecewise linear
    - Theorem & experiments that tie #pieces to ε
    - Gives an algorithm for solving multiplayer games without qualitative models too
- Multiple qualitative models (with a common refinement) only some of which are correct
- Dependent types

Once we obtain the x_i and y_i by solving the MILFP, we must map them into mixed strategies of the game. Suppose player 1 is dealt private signal $z \in [1, \overline{n}]$ and consider the interval $I = [F_1(z-1), F_1(z)]$. Now define the intervals $J_i = [x_{i-1}, x_i]$ where we define $x_{-1} = 0$. Let O_i denote the overlap between sets I and J_i. Then player 1 will play the strategy defined by region i with probability $\frac{O_i}{\sum_i O_i}$. The strategy for player 2 is determined similarly, using the y_i and F_2.

Multiple players

- With more than 2 players, indifference constraints become nonlinear
- We can compute an ε-equilibrium by approximating products of variables using linear constraints
  - We provide a formula for the number of breakpoints per piecewise linear curve needed as a function of ε
- Our algorithm uses a MILFP that is polynomial in #players
- Can apply our technique to develop a MIP formulation for finding ε-equilibria in multiplayer normal and extensive-form games without qualitative models

Multiple parametric models

- Often have several models and know at least one is correct, but not sure which
- We give an algorithm for finding an equilibrium given several parametric models that have a common refinement
  - Some of the models can be incorrect
  - If none of the models are correct, our algorithm says so

Experiments

- Games for which algs didn’t exist become solvable
  – Multi-player games
- Previously solvable games solvable faster
  – Continuous approximation sometimes a better alternative than abstraction (e.g., n-card Kuhn poker)
- Works in the large
  – Improved performance of *GS4* when used for last betting round

Experiments

Texas Hold'em experiments

- Once river card dealt, no more information revealed
- Use GS4 and Bayes' rule to generate distribution over possible hands both players could have
- We developed 3 parametric models that have a common refinement (for 1-raise-per-player version)
  - All three turned out necessary

Texas Hold'em experiments cont'd

- We ran it against top 5 entrants from 2008 AAAI Computer Poker Competition
- Performed better than GS4 against 4
- Beat GS4 by 0.031 (± 0.011) small bets/hand
- Averaged 0.25 seconds/hand overall

Multiplayer experiments

- Simplified 3-player poker game
- Rapid convergence to ε-equilibrium for several CDFs
- Obtained $\varepsilon = 0.01$ using 5 breakpoints
  - Theoretical bound $\varepsilon \approx 25$

Approximating large finite games with continuous games

- Traditional approach: abstraction
- Suppose private signals in {1,..,n} in first poker example
  - Runtime of computing equilibrium grows large as n increases
  - Runtime of computing x_∞ remains the same
- Our approach can require much lower runtime to obtain given level of exploitability

Approximating large finite games with continuous games

- Experiment on Generalized Kuhn poker [Kuhn '50]
- Compared value of game vs. payoff of x_∞ against its nemesis
- Agree to within .0001 for 250 signals
- Traditional approach required very fine abstraction to obtain such low exploitability

Conclusions

- Qualitative models can significantly help equilibrium finding
  - Solving classes of games for which no prior algorithms exist
  - Speedup
- We develop an algorithm for computing an equilibrium given qualitative models of the structure of equilibrium strategies
  - Sound and complete
  - Some of the models can be incorrect
  - If none are correct, our algorithm says so
- Applies to both infinite and large finite games
  - And to dependent type distributions
- Experiments show practicality
  - Endgames of 2-player Texas Hold'em
  - Multiplayer games
  - Continuous approximation superior to abstraction in some games

Future research

- How to generate parametric models? Can this be automated?
- Can this infinite projection approach compete with abstraction for large real-world games of interest?
- In the case of multiple parametric models, can correctness of our algorithm be proven without assuming a common refinement?

Summary

- Domain-independent techniques
- Automated lossless abstraction
  - Exactly solved game with 3.1 billion nodes
- Automated lossy abstraction
  - k-means clustering & integer programming
  - Potential-aware
  - Phase-based abstraction & real-time endgame solving
  - Action abstraction & reverse models
  - First lossy game abstraction algorithms with bounds
  - Strategy-based abstraction
- Equilibrium-finding for 2-person 0-sum games
  - $O(1/\varepsilon^2)$ -> $O(1/\varepsilon)$ -> $O(\log(1/\varepsilon))$
  - Can solve games with over 10^{14} nodes to small ε
- Purification and thresholding help – surprising
- Scalable practical online opponent exploitation algorithm
- Fully characterized safe exploitation & provided algorithms
- Solved large multiplayer stochastic games
- Leveraging qualitative models => existence, computability, speed

Summary

- Domain-independent techniques
- Game abstraction
  - Automated lossless abstraction -- exactly solved game with 3.1 billion nodes
  - Automated lossy abstraction with bounds
    - For action and state abstraction
    - Also for modeling
- Equilibrium-finding for 2-person 0-sum games
  - $O(1/\varepsilon^2)$ -> $O(1/\varepsilon)$ -> $O(\log(1/\varepsilon))$
  - Can solve games with over 10^{14} nodes to small ε
- Purification and thresholding help – surprising
- Scalable practical online opponent exploitation algorithm
- Fully characterized safe exploitation & provided algorithms
- Solved large multiplayer stochastic games
- Leveraging qualitative models => existence, computability, speed

Did not discuss…

- DBs, data structures, …

Some of our current & future research

- Lossy abstraction with bounds
  - Extensive form
  - With structure
  - With generated abstract states and actions
- Equilibrium-finding algorithms for 2-person 0-sum games
  - Can CFR be parallelized or fast EGT made to work with imperfect recall?
  - Fast implementations of our $O(\log(1/\varepsilon))$ algorithm and understanding how #iterations depends on matrix condition number
  - Making interior-point methods usable in terms of memory
- New game classes where our algs for stochastic multiplayer games (and their components) are guaranteed to converge
- Other solution concepts: sequential equilibrium, coalitional deviations,…
- Actions beyond the ones discussed in the rules:
  - Explicit information-revelation actions
  - Timing, …
- Understanding exploration vs exploitation vs safety
- Theoretical understanding of thresholding and purification
- Using & adapting these techniques to other games, esp. (cyber)security