Sequential imperfect-information games

Case study: *Poker*

Tuomas Sandholm

Carnegie Mellon University

Computer Science Department

Sequential imperfect information games

- Players face uncertainty about the state of the world
- Most real-world games are like this
  - A robot facing adversaries in an uncertain, stochastic environment
  - Almost any card game in which the other players' cards are hidden
  - Almost any economic situation in which the other participants possess private information (*e.g.* valuations, quality information)
    - Negotiation
    - Multi-stage auctions (e.g., English)
    - Sequential auctions of multiple items
  - …
- This class of games presents several challenges for AI
  - Imperfect information
  - Risk assessment and management
  - Speculation and counter-speculation
- Techniques for solving sequential complete-information games (like chess) don't apply
- Our techniques are domain-independent

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: most popular variant

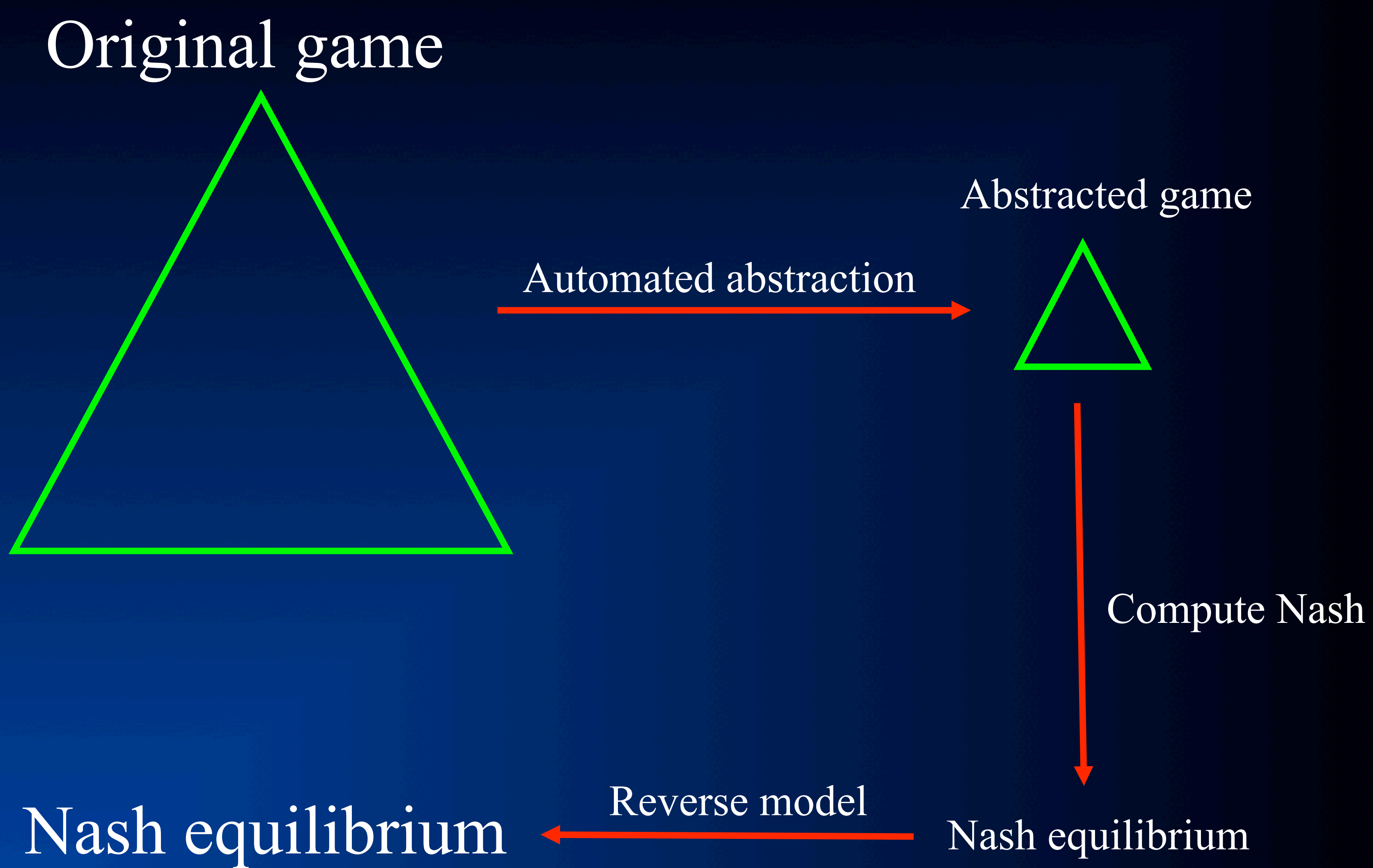
On NBC:

Finding 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 (even in reduced normal form)
- *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)

Our approach [Gilpin & Sandholm EC'06, JACM'07]

Now used by all competitive Texas Hold'em programs



Outline

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

Lossless abstraction

[Gilpin & Sandholm 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

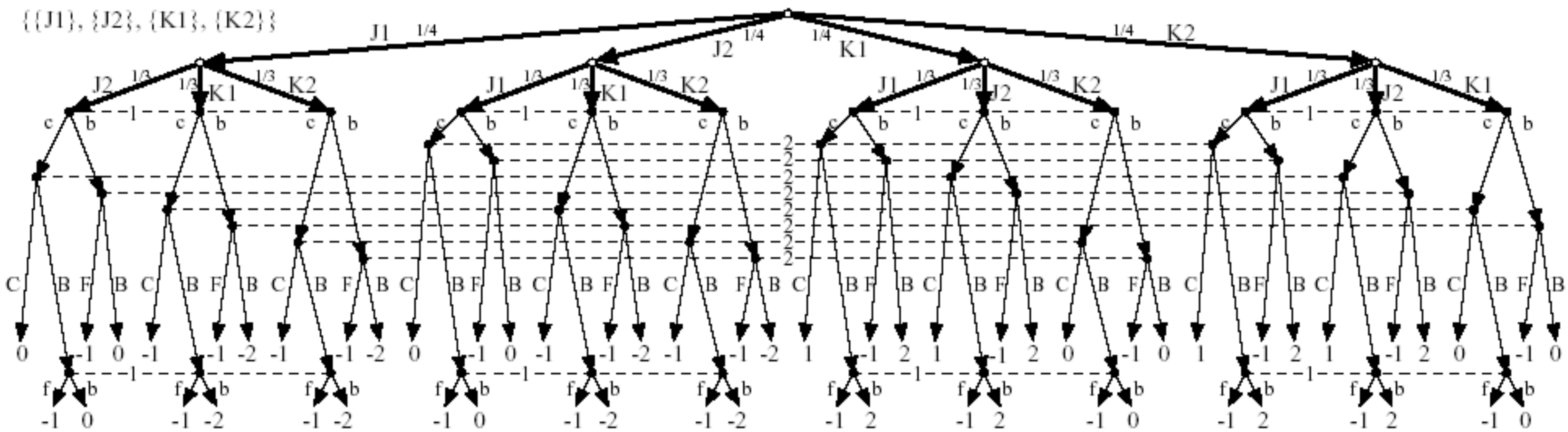
- Each edge corresponds to the revelation of some signal by nature to at least one player
- Our abstraction algorithms operate on it
  - Don't load full game into memory

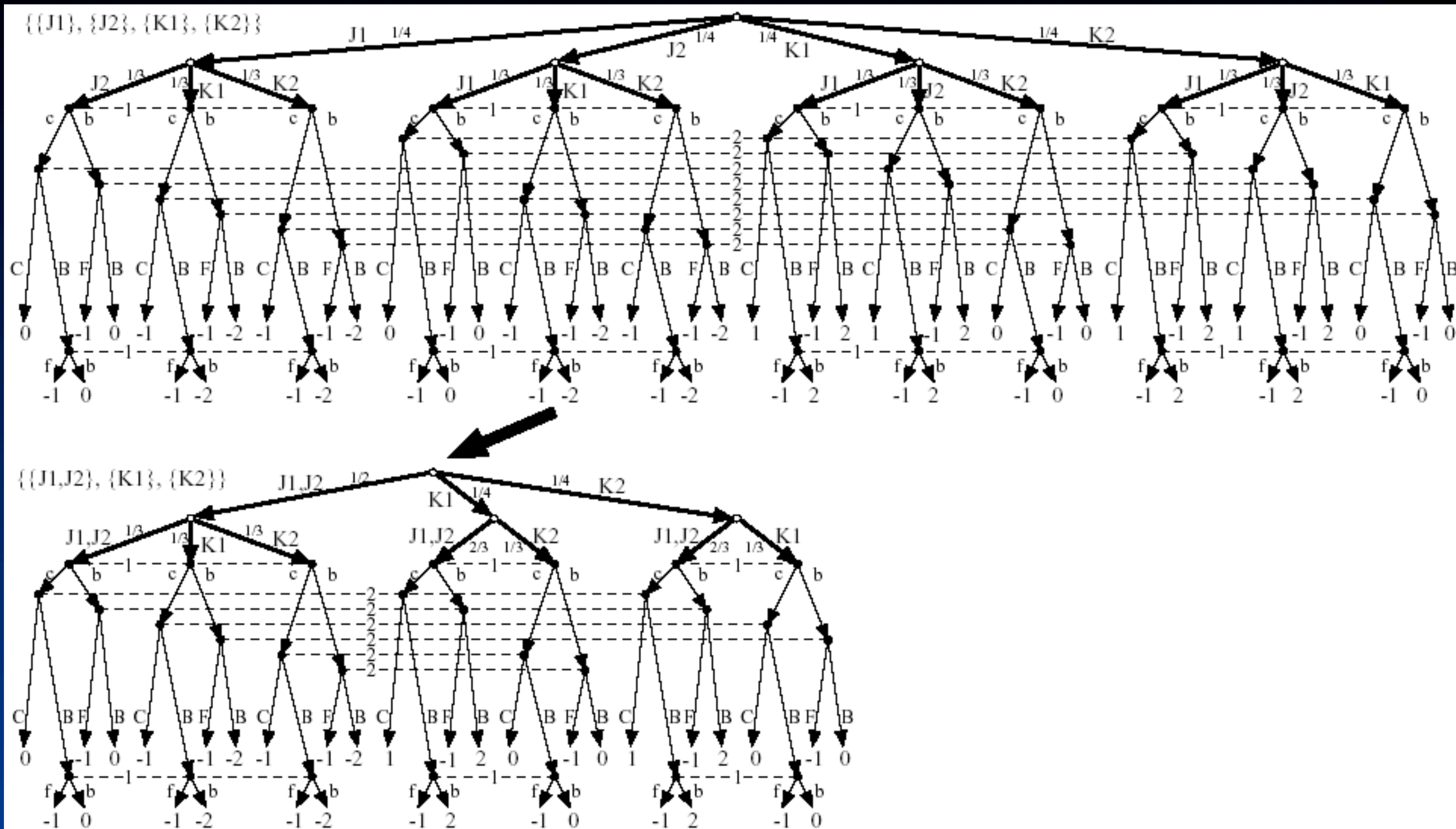
Isomorphic relation

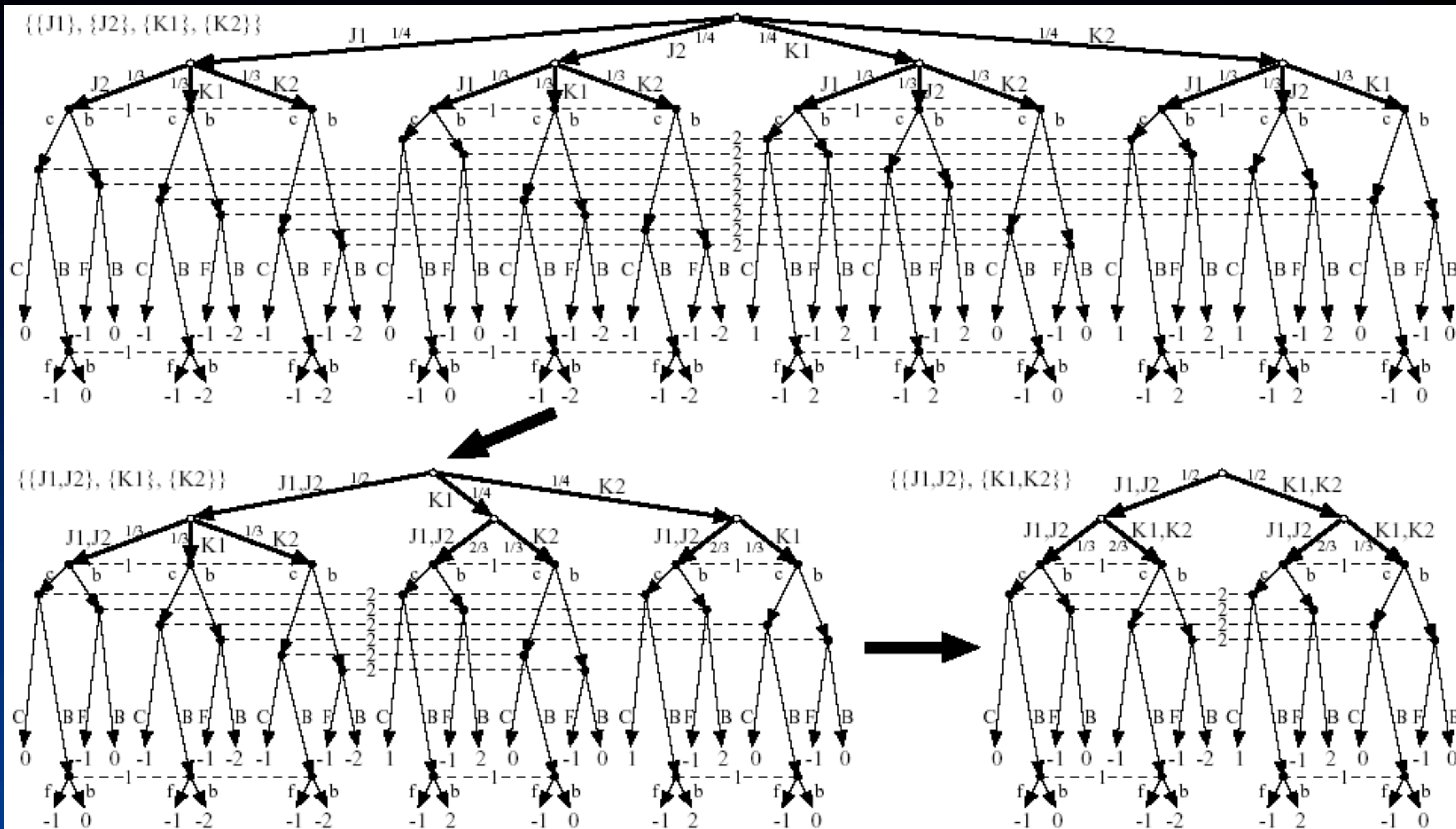
- Captures the notion of strategic symmetry between nodes
- Defined recursively:
  - Two leaves in signal tree are isomorphic if for each action history in the game, the payoff vectors (one payoff per player) are the same
  - Two internal nodes in signal tree are isomorphic if they are siblings and there 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
  - Answer is stored

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}, {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

- One approximation algorithm: instead of requiring perfect matching, require a matching with a penalty below threshold

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

Solving Rhode Island Hold'em poker

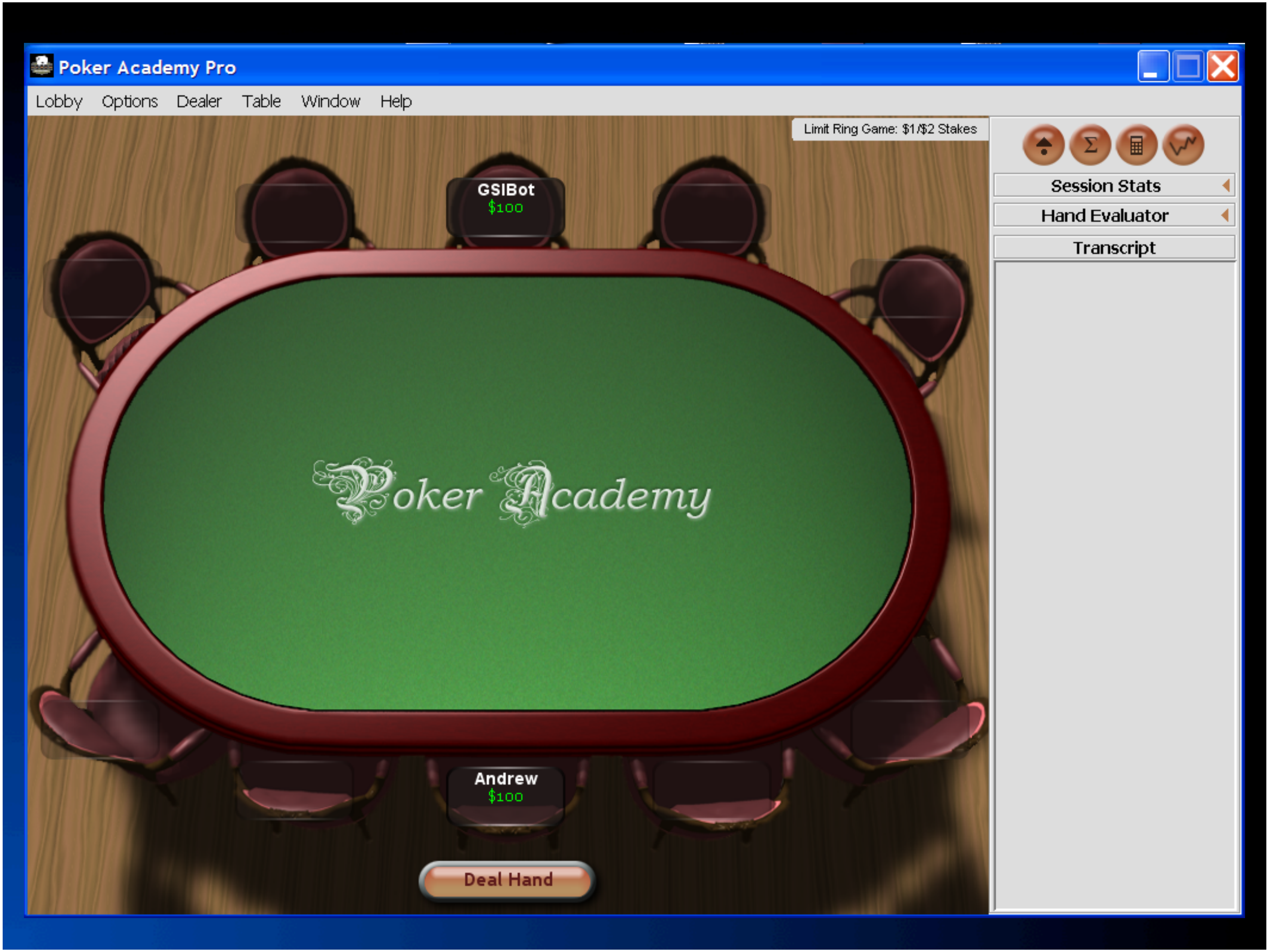
- AI challenge problem [Shi & Littman 01]
  - 3.1 billion nodes in game tree
- Without abstraction, LP has 91,224,226 rows and columns => unsolvable
- *GameShrink* runs in one second
- After that, LP has 1,237,238 rows and columns
- Solved the LP
  - CPLEX *barrier* method took 8 days & 25 GB RAM
- Exact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

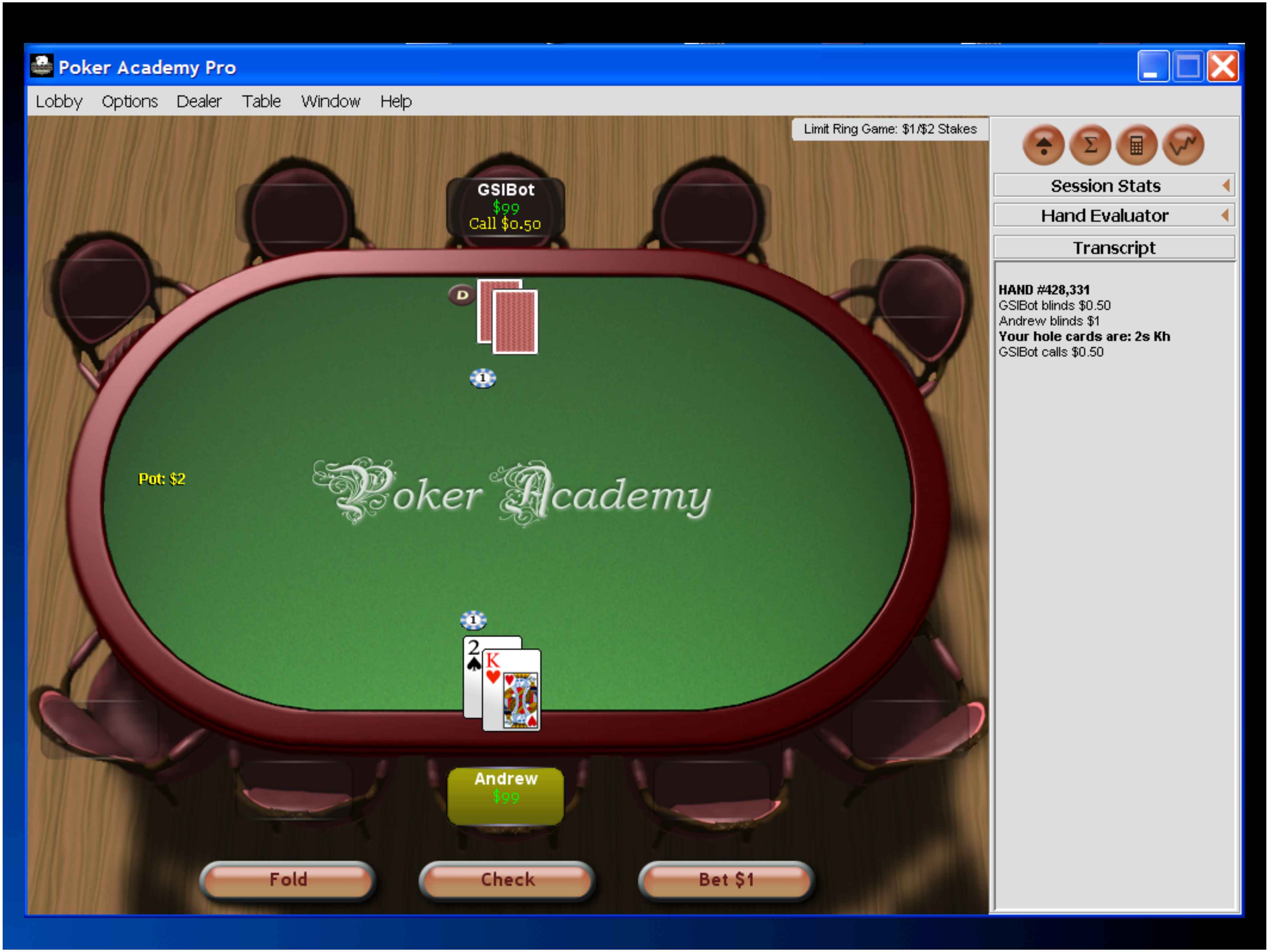
Lossy abstraction

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


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


Poker Academy Pro
Lobby Options Dealer Table Window Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$99
Call $0.50
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
Pot: $2
Poker Academy
Andrew
$99
Fold
Check
Bet $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
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
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
GSIBot
$98
Bet $1
D
Pot: $3
cademy
Andrew
$99
Fold
Call $1
Raise $1


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$98
D
Pot: $4
Andrew
$98
Fold
Check
Bet $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


Poker Academy Pro
Lobby
Options
Dealer
Table
Window
Help
Limit Ring Game: $1/$2 Stakes
GSIBot
$96
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
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
GSIBot
$106
GSIBot wins $12 with Two Pair, Queens and Sevens
Andrew
$94
Deal Hand
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

GS1

1/2005 - 1/2006

GS1 [Gilpin & Sandholm AAAI'06]

- Our first program for 2-person Limit Texas Hold'em
- 1/2005 - 1/2006
- First Texas Hold'em program to use automated abstraction
  - Lossy version of Gameshrink

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):

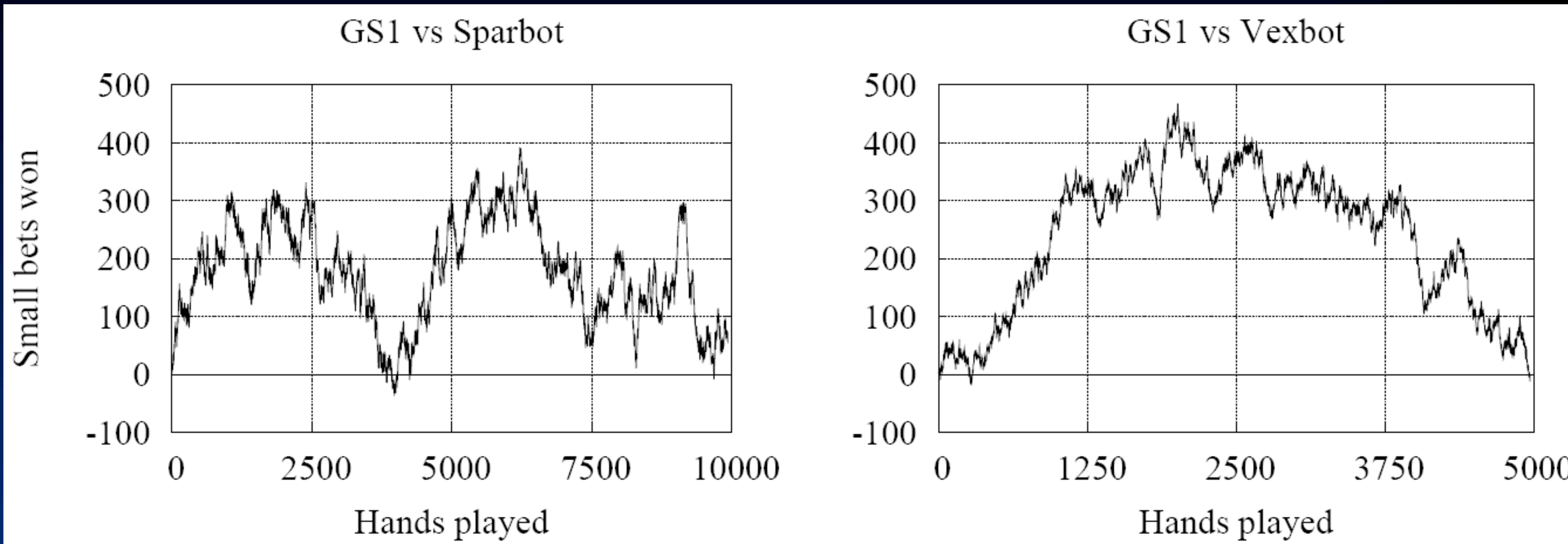
$$\Pr[\theta_i \mid h, s_i] = \frac{\Pr[h \mid \theta_i, s_i]\Pr[\theta_i]}{\Pr[h \mid s_i]} = \frac{\Pr[h \mid \theta_i, s_i]\Pr[\theta_i]}{\sum\limits_{\theta_i' \in \Theta} \Pr[h \mid \theta_i', s_i]}$$

      - 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 & Sandholm AAMAS'07]

- 2/2006-7/2006
- Original version of *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
- 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
    - Statistics from 100,000's hands of *SparBot* in self-play

GS2

2/2006 – 7/2006

[Gilpin & Sandholm 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)

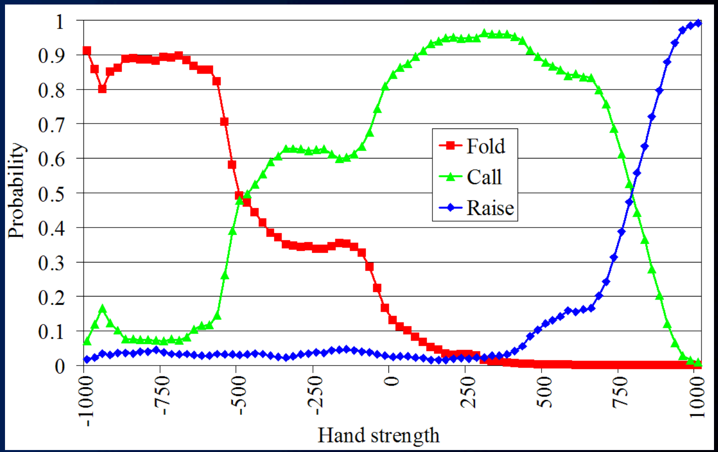
- Optimized abstraction
  - Round 1
    - There are 1,326 hands, of which 169 are strategically different
    - We allowed 15 abstract states
  - Round 2
    - There are 25,989,600 distinct possible hands
      - *GameShrink* (in lossless mode for Phase I) determined there are ~10^6 strategically different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



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, Sandholm & Sørensen AAAI'07]

GS4 is similar

Entire game solved holistically

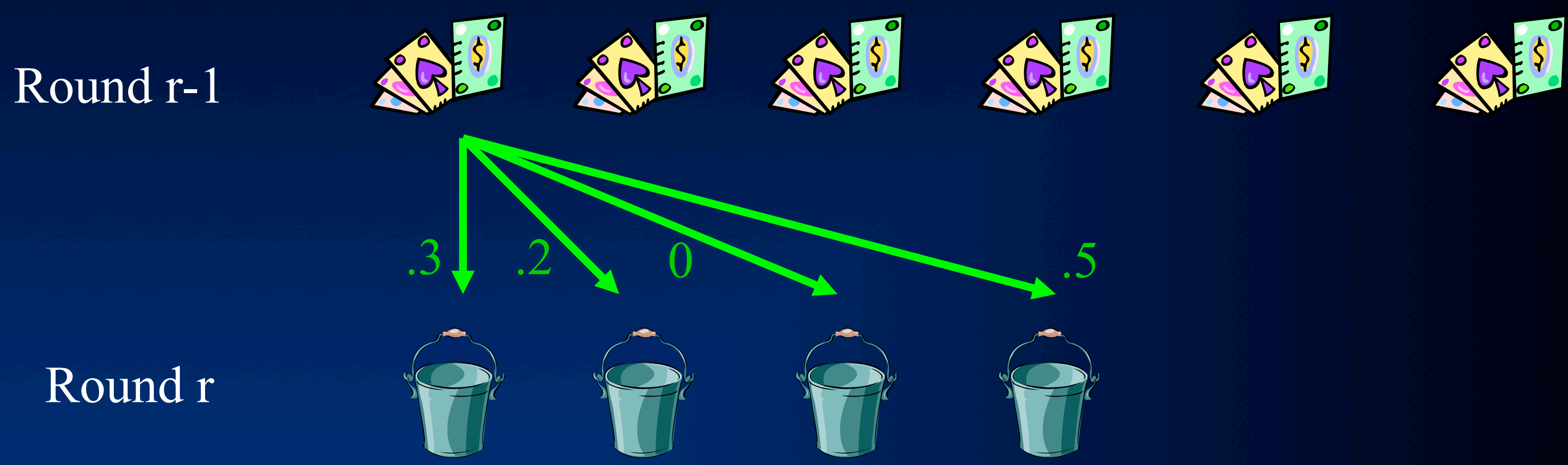
- We no longer break game into phases
  - Because our new equilibrium-finding algorithms can solve games of the size that stem from reasonably fine-grained abstractions of the entire game
- => better strategies & no need for real-time computation

Potential-aware automated abstraction

- All prior abstraction algorithms (including ours) had myopic probability of winning as the similarity metric
  - Does not address *potential*, e.g., hands like flush draws where although the probability of winning is small, the payoff could be high
- Potential not only positive or negative, but also "multidimensional"
- GS3's abstraction algorithm takes potential into account…

- 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 (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

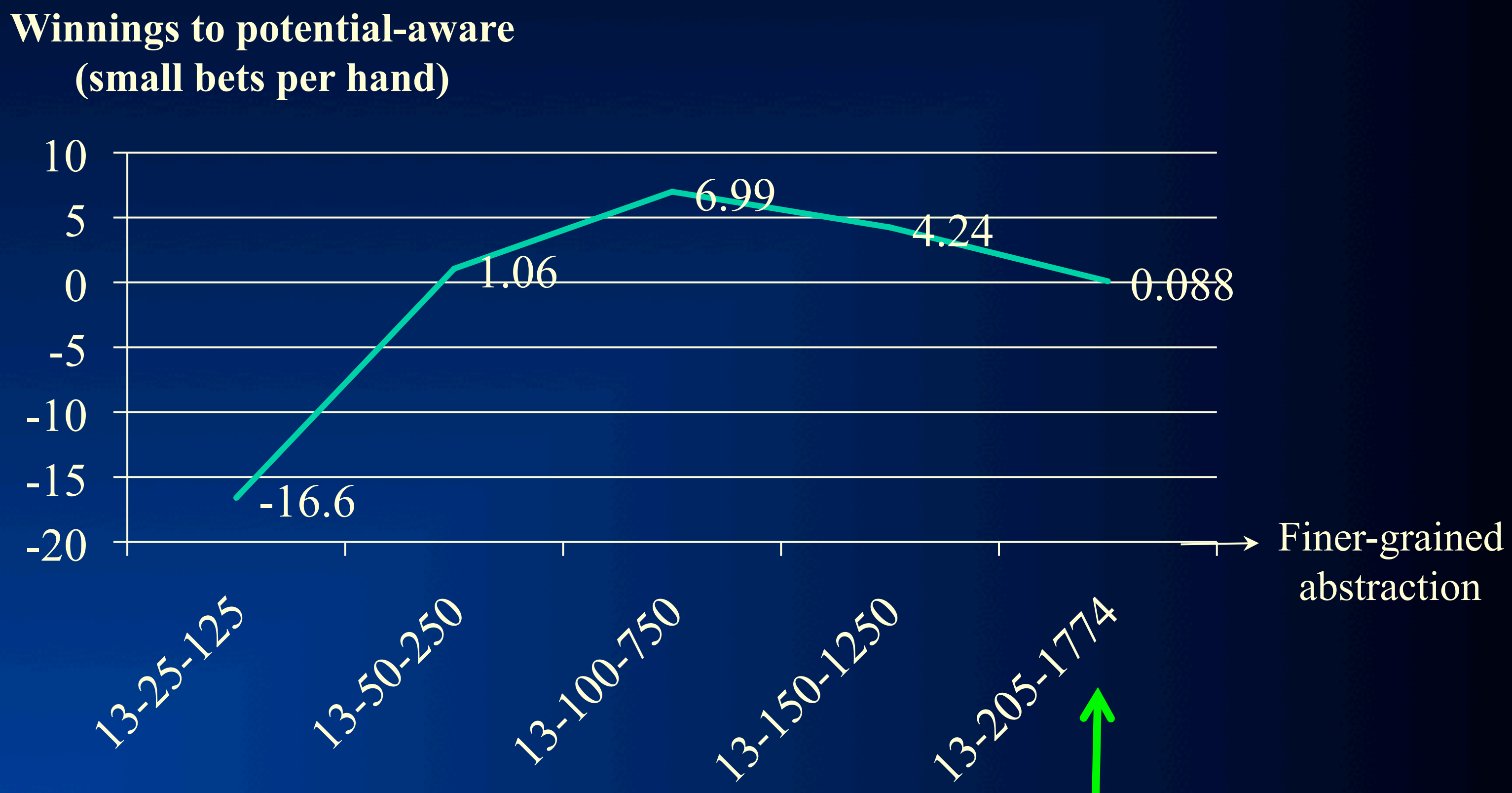
Determining abstraction for round 4

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

Potential-aware vs win-probability-based abstraction

[Gilpin & Sandholm AAAI-08]

- Both use clustering and IP
- Experiment conducted on Heads-Up 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 & Sandholm AAAI-08 & new]

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

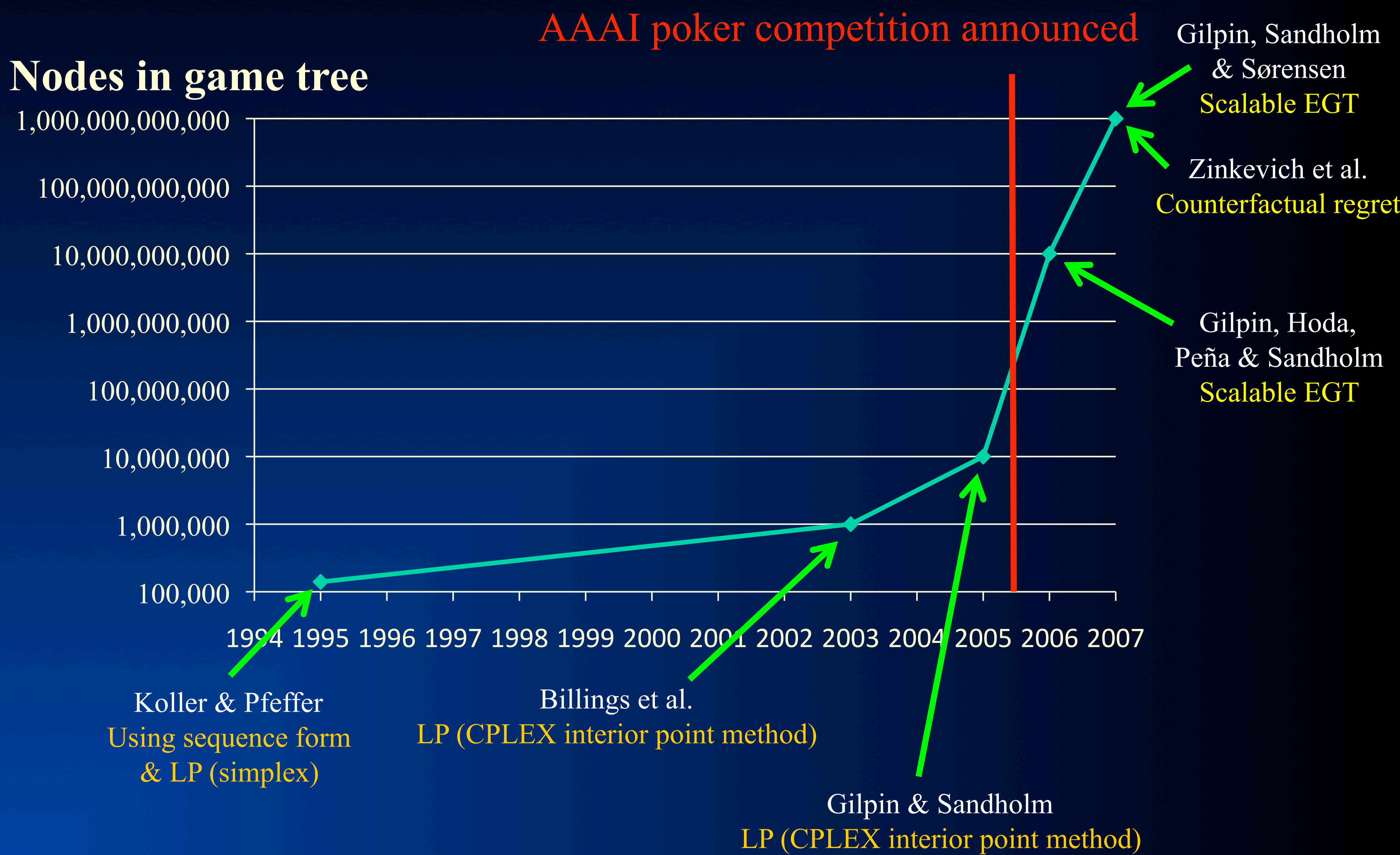
Equilibrium-finding algorithms

Solving the (abstracted) game

Now we move from discussing general-sum n-player games to discussing 2-player 0-sum games

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)

- LP solvers only scale to ~10^7 nodes. Can we do better than use LP?
- Usually, gradient-based algorithms have poor convergence, but…
- **Theorem** [Nesterov 05]. There is a gradient-based algorithm (for a class of *minmax problems*) that finds an ε-equilibrium in O(1/ ε) iterations
- In general, work per iteration is as hard as solving the original problem, but…
- Can make each iteration faster by considering problem structure:
- **Theorem** [Hoda et al. 06]. In sequential games, each iteration can be solved in time linear in the size of the game tree

Scalable EGT [Gilpin, Hoda, Peña, Sandholm WINE'07]

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, Sandholm WINE'07]

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
    - Less conservative shrinking of [illegible]$_1$ and [illegible]$_2$
      - Sometimes need to reduce (halve) t
    - Balancing [illegible]$_1$ and [illegible]$_2$ 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

Iterated smoothing [Gilpin, Peña & Sandholm AAAI-08]

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

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

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

Results (for *GS4*)

- AAAI-08 Computer Poker Competition
  - *GS4* won the Limit Texas Hold'em bankroll category
    - Played 4-4 in the pairwise comparisons. 4^{th} of 9 in elimination category
  - *Tartanian* did the best in terms of bankroll in No-Limit Texas Hold'em
    - 3^{rd} out of 4 in elimination category

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 + LP for equilibrium finding [Billings et al. IJCAI-03]
  - **Ours**
    - **Automated abstraction**
    - **Custom solver for finding Nash equilibrium**
    - **Domain independent**

>2 players

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

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-08]
  - 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 IJCAI-09]
  - 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 IJCAI-09]

- 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

Summary

- Domain-independent techniques
- Automated lossless abstraction
  - Solved Rhode Island Hold'em exactly
    - 3.1 billion nodes in game tree, biggest solved before had 140,000
- Automated lossy abstraction
  - k-means clustering & integer programming
  - Potential-aware
- Novel scalable equilibrium-finding algorithms
  - Scalable EGT & iterated smoothing
- DBs, data structures, …
- Won AAAI-08 Computer Poker Competition Limit Texas Hold'em bankroll category (and did best in bankroll in No-Limit also)
  - Competitive with world's best professional poker players?
- First algorithms for solving large stochastic games with >2 players (3-player jam/fold poker tournaments)

Current & future research

- Abstraction
  - Provable approximation (*ex ante / ex post*)
  - Action abstraction (requires reverse model) -> *Tartanian* for No-Limit Texas Hold'em [Gilpin, Sandholm & Sørensen AAMAS-08]
  - Other types of abstraction
- Equilibrium-finding algorithms with even better scalability
- Other solution concepts: sequential equilibrium, coalitional deviations,…
- Even larger #players (cash game & tournament)
- Opponent modeling
- Actions beyond the ones discussed in the rules:
  - Explicit information-revelation actions
  - Timing, …
- Trying these techniques in other games