

Poker AI:
**Algorithms for Creating Game-Theoretic Strategies
for Large Incomplete-Information Games**

Tuomas Sandholm

Professor

Carnegie Mellon University

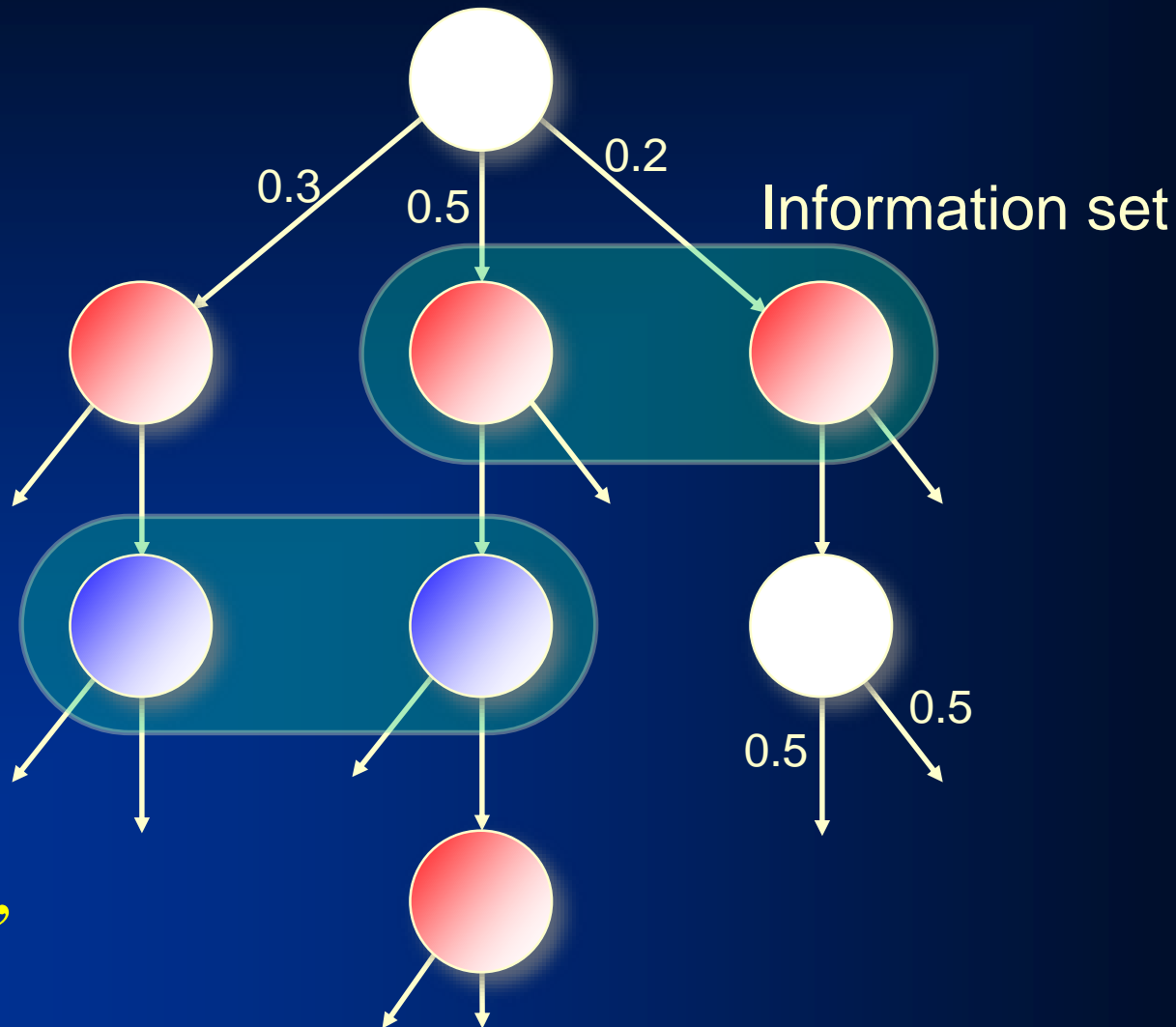
Computer Science Department

Machine Learning Department

Ph.D. Program in Algorithms, Combinatorics, and Optimization

CMU/UPitt Joint Ph.D. Program in Computational Biology

Incomplete-information game tree



Tackling such games

- Domain-independent techniques
- Techniques for complete-info games don't apply
- Challenges
 - Unknown state
 - Uncertainty about what other agents and nature will do
 - Interpreting signals and avoiding signaling too much
- **Definition.** A **Nash equilibrium** is a *strategy* and *beliefs* for each agent such that no agent benefits from using a different strategy
 - Beliefs derived from strategies using Bayes' rule

Most real-world games are like this

- Negotiation
- Multi-stage auctions (FCC ascending, combinatorial)
- Sequential auctions of multiple items
- Political campaigns (TV spending)
- Ownership games (polar regions, moons, planets)
- Military (allocating troops; spending on space vs ocean)
- Next-generation (cyber)security (jamming; OS security)
- Medical treatment [Sandholm 2012]
- ...

Poker

Recognized challenge problem in AI since 1992 [Billings, Schaeffer, ...]

- Hidden information (other players' cards)
- Uncertainty about future events
- Deceptive strategies needed in a good player
- Very large game trees

NBC National Heads-Up Poker Championship 2013



Our approach [Gilpin & Sandholm EC-06, *J. of the ACM* 2007...]

Now used basically by all competitive Texas Hold'em programs

Original game

Abstracted game

Automated abstraction

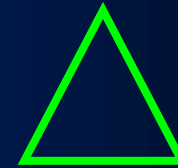
Custom
equilibrium-finding
algorithm

Reverse model

Nash equilibrium

Nash equilibrium

Foreshadowed by Billings et al. IJCAI-03



Lossless abstraction

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 $\{A\spadesuit, A\clubsuit, A\heartsuit, A\diamondsuit\}$ instead of $A\heartsuit$

Signal tree

- Each edge corresponds to the revelation of some signal by nature to at least one player
- Our abstraction algorithm operates on it
 - Doesn'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 their children are isomorphic
 - *Challenge*: permutations of children
 - *Solution*: custom **perfect matching** algorithm between children of the two nodes such that only isomorphic children are matched

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*

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

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

Texas Hold'em poker



- 2-player Limit has $\sim 10^{18}$ nodes
- 2-player No-Limit has $\sim 10^{165}$ nodes
- Losslessly abstracted game too big to solve
=> abstract more
=> lossy

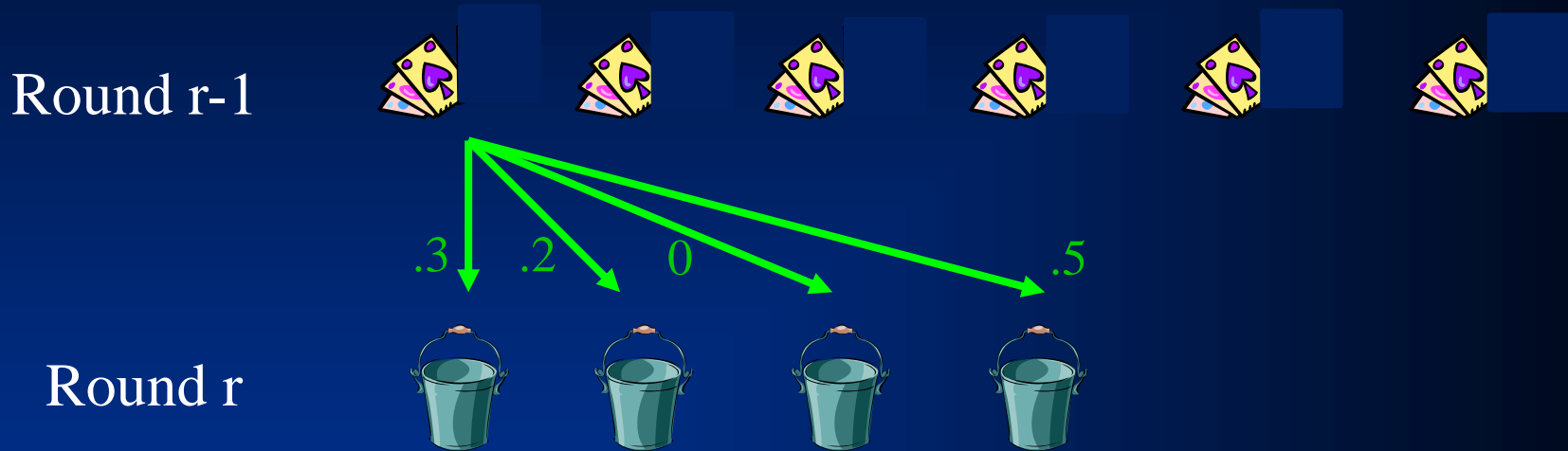
Clustering + integer programming for abstraction

- *GameShrink* can be made to abstract more \Rightarrow lossy
 - Greedy \Rightarrow lopsided abstractions
- Better approach: Abstraction via clustering + IP

Potential-aware abstraction

- All prior abstraction algorithms had probability of winning (assuming no more betting) as the similarity metric
 - Doesn't capture *potential*
- Potential not only positive or negative, but “multidimensional”
- We developed an abstraction algorithm that captures potential ...

Bottom-up pass to determine abstraction for round 1



In the last round, there is no more potential

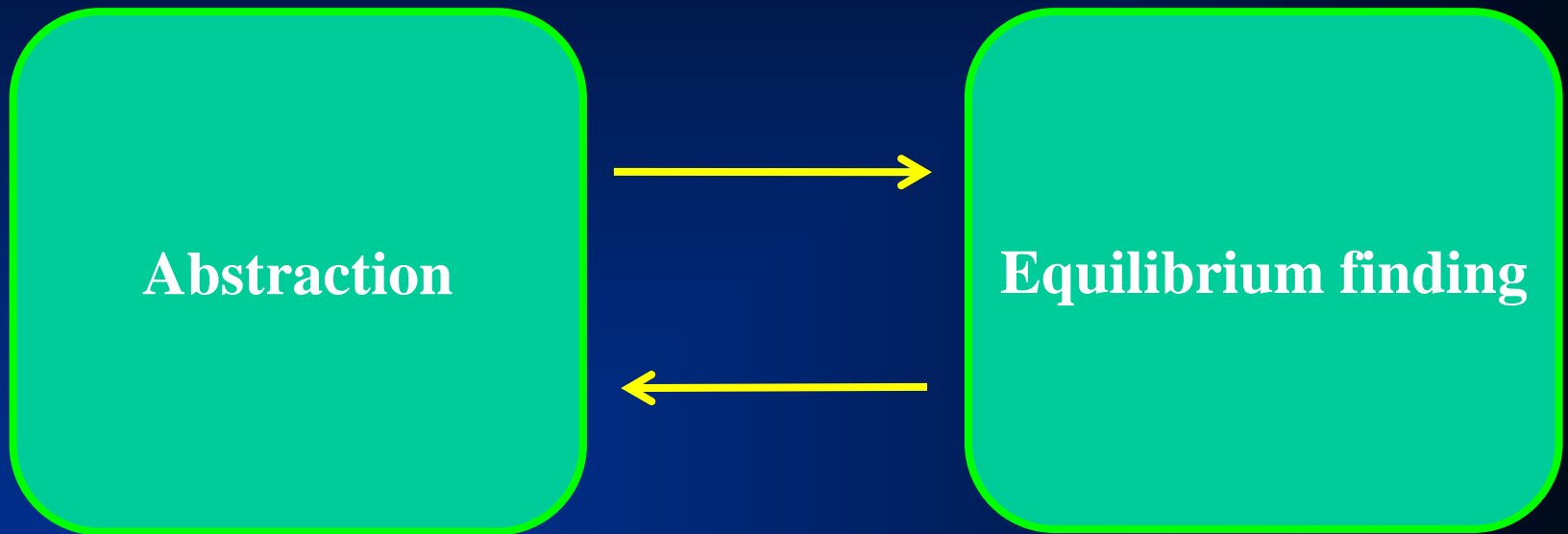
=> use probability of winning as similarity metric

Can combine the abstraction ideas

- Integer programming [Gilpin & Sandholm AAMAS-07]
- Potential-aware [Gilpin, Sandholm & Sørensen AAI-07, Gilpin & Sandholm AAI-08]
- Imperfect recall [Waugh et al. SARA-09, Johanson et al. AAMAS-13]

Strategy-based abstraction

[Ongoing work in my group]



First lossy game abstraction methods with bounds

- Tricky due to abstraction pathology [Waugh et al. AAMAS-09]
- For both action and state abstraction
- For stochastic games
- **Theorem.** Given a subgame perfect Nash equilibrium in an abstract game, no agent can gain more than

$$2k^2 \left[\varepsilon_R + kR_{\max} \varepsilon_T \right]$$

in the real game by deviating to a different strategy

First lossy game abstraction algorithms with bounds

- Proceed level by level from end of game
 - Optimizing all levels simultaneously would be nonlinear
 - **Proposition.** Both algorithms satisfy given bound on regret
- Within level:
 1. Greedy polytime algorithm; does action or state abstraction first
 2. Integer program
 - Does action and state abstraction simultaneously
 - Apportions allowed total error within level **optimally**
 - between action and state abstraction, and
 - between reward and transition probability error
 - **Proposition.** Abstraction is NP-complete
- **One of the first action abstraction algorithms**
 - Totally different than [Hawkin et al. AAAI-11, 12], which doesn't have bounds

Role in modeling

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

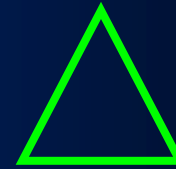
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm



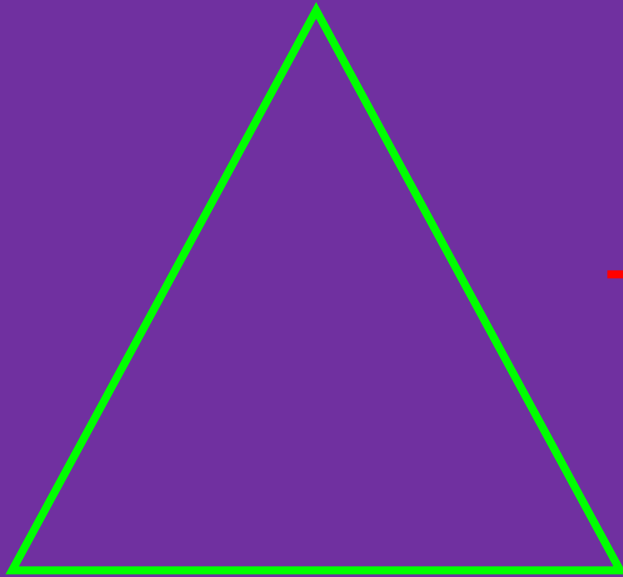
Nash equilibrium

Reverse model



Nash equilibrium

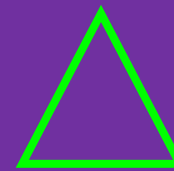
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm



Nash equilibrium

Reverse model



Nash equilibrium

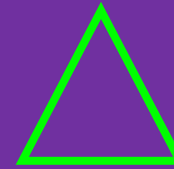
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm



Nash equilibrium

Reverse model



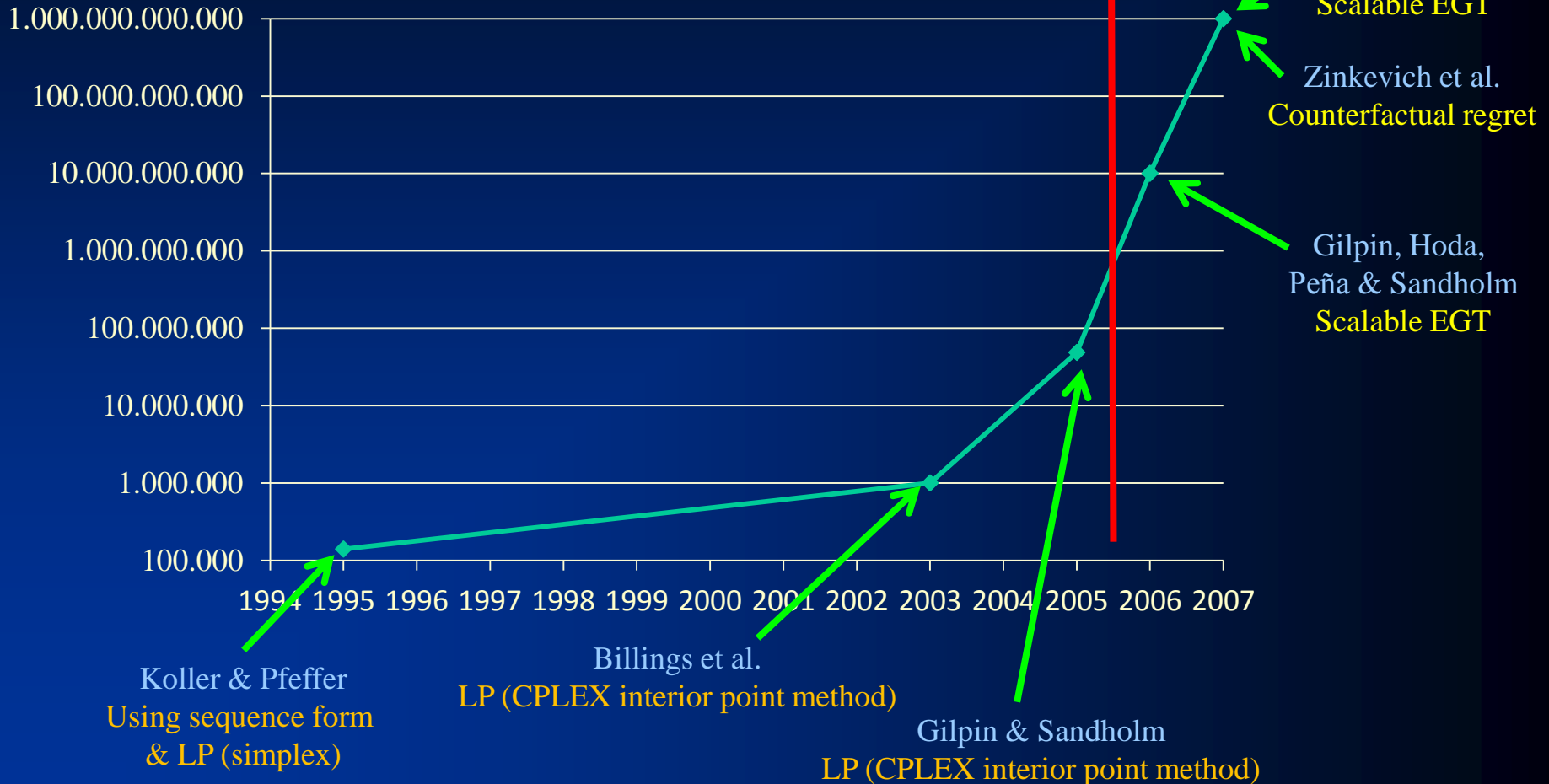
Nash equilibrium



Picture credit: Pittsburgh Supercomputing Center

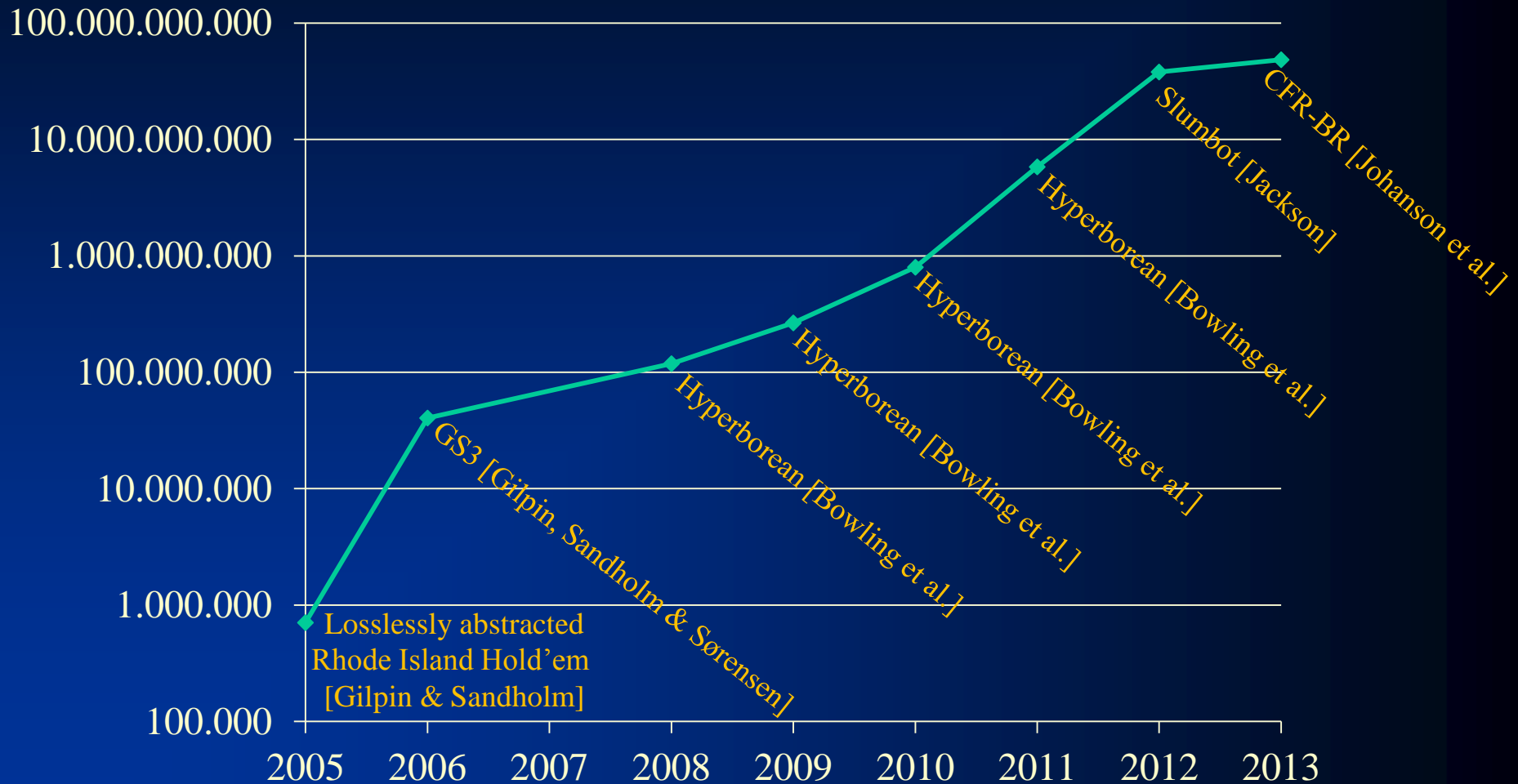
Scalability of (near-)equilibrium finding in 2-player 0-sum games

Nodes in game tree



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

Number of information sets



Best equilibrium-finding algorithms for 2-player 0-sum games

Counterfactual regret (CFR)

- Based on no-regret learning
- Most powerful innovations:
 - Each information set has a separate no-regret learner [Zinkevich et al. NIPS-07]
 - Sampling [Lanctot et al. NIPS-09, ...]
- $O(1/\epsilon^2)$ iterations
 - Each iteration is fast
- Parallelizes
- Selective superiority
- Can be run on imperfect-recall games and with >2 players (without guarantee of converging to equilibrium)

Scalable EGT

- Based on Nesterov's Excessive Gap Technique
- Most powerful innovations:
 - [Hoda, Gilpin, Peña & Sandholm WINE-07, *Mathematics of Operations Research* 2011]
 - Smoothing fns for sequential games
 - Aggressive decrease of smoothing
 - Balanced smoothing
 - Available actions don't depend on chance => memory scalability
- $O(1/\epsilon)$ iterations
 - Each iteration is slow
- Parallelizes
- New $O(\log(1/\epsilon))$ algorithm
 - [Gilpin, Peña & Sandholm AAAI-08, *Mathematical Programming* 2012]

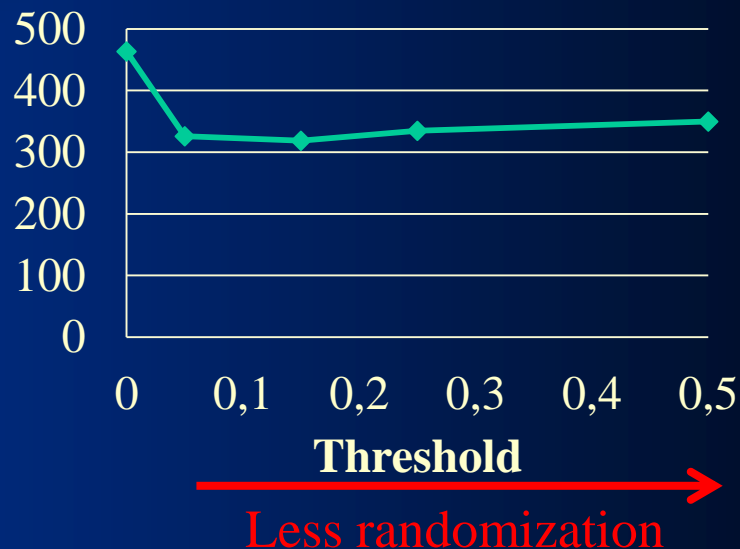
Purification and thresholding

- *Thresholding*: Rounding the probabilities to 0 of those actions whose probabilities are less than c (and rescaling the other probabilities)
 - *Purification* is thresholding with $c = 1/2$
- **Proposition.** Can help or hurt arbitrarily much, when played against equilibrium strategy in unabstracted game

Experiments on purification & thresholding

- No-limit Texas Hold'em: Purification beats threshold 0.15, does better than it against all but one 2010 competitor, and won bankroll competition
- Limit Texas Hold'em:

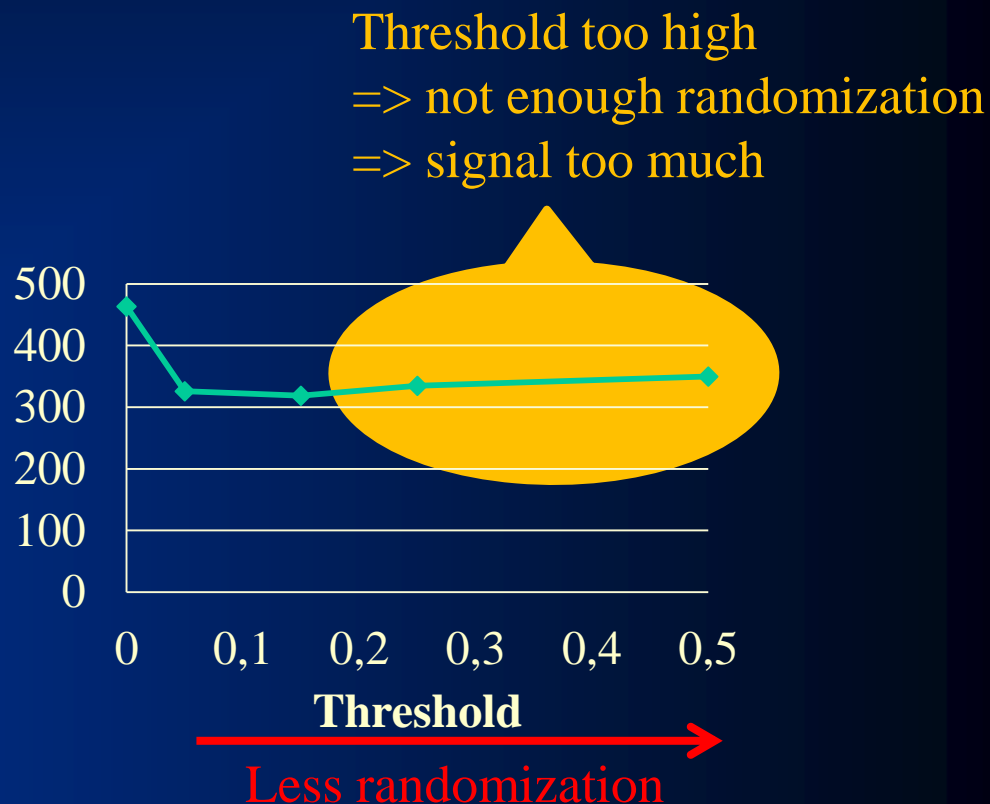
*Exploitability of our 2010 bot
(in milli big blinds per hand)*



Experiments on purification & thresholding

- No-limit Texas Hold'em: Purification beats threshold 0.15, does better than it against all but one 2010 competitor, and won bankroll competition
- Limit Texas Hold'em:

*Exploitability of our 2010 bot
(in milli big blinds per hand)*



Experiments on purification & thresholding

- No-limit Texas Hold'em: Purification beats threshold 0.15, does better than it against all but one 2010 competitor, and won bankroll competition
- Limit Texas Hold'em:

Threshold too low

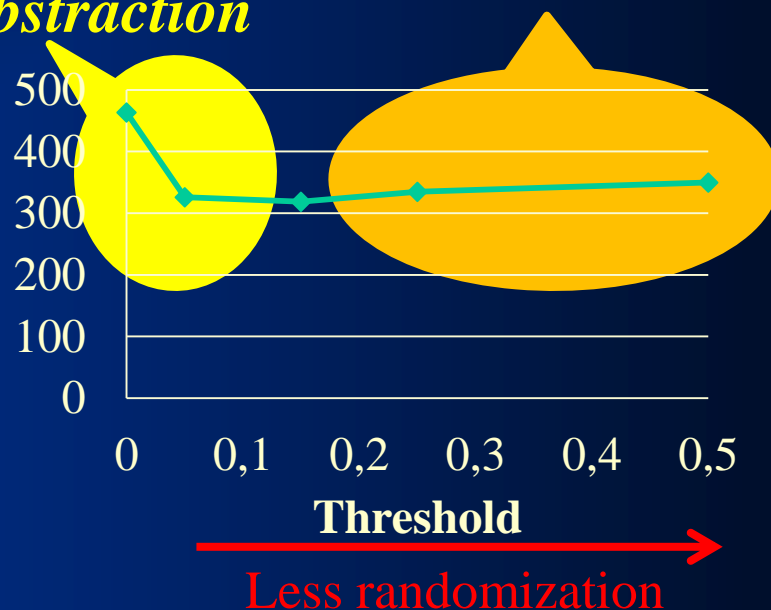
=> *strategy overfit to abstraction*

Threshold too high

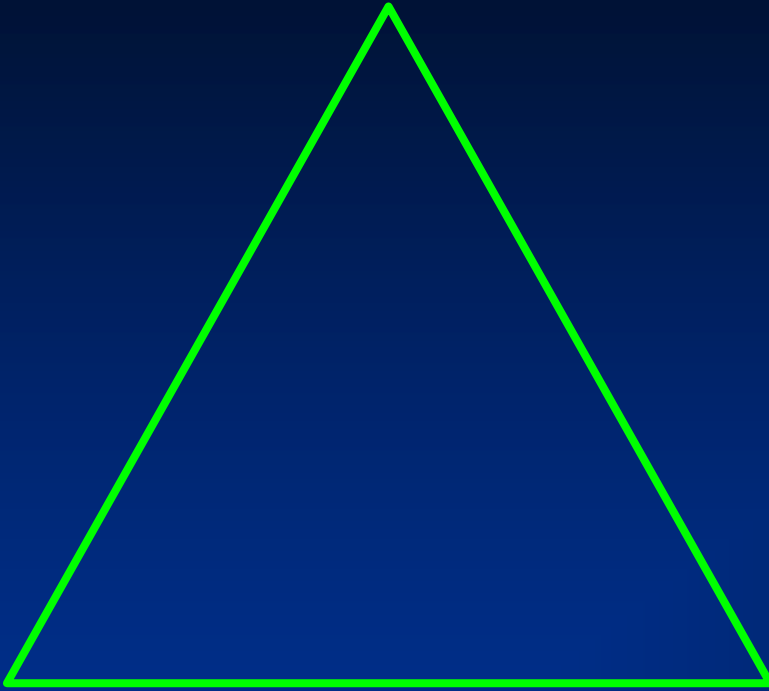
=> not enough randomization

=> signal too much

*Exploitability of our 2010 bot
(in milli big blinds per hand)*

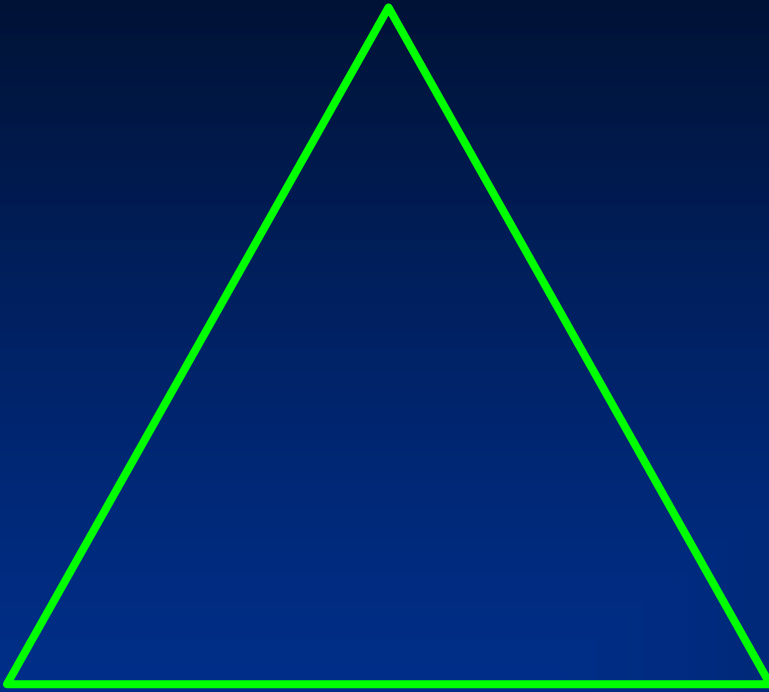


Endgame solving

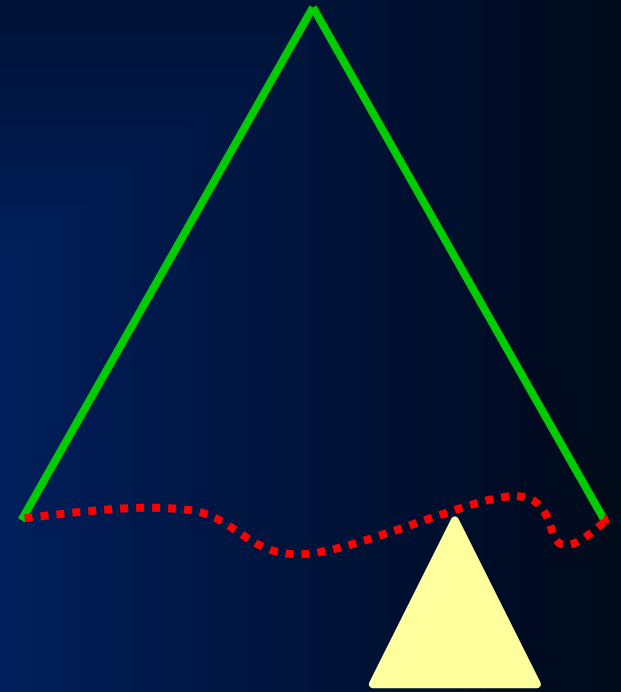


Strategies for entire game computed
offline in a coarse abstraction

Endgame solving



Strategies for entire game computed
offline in a coarse abstraction

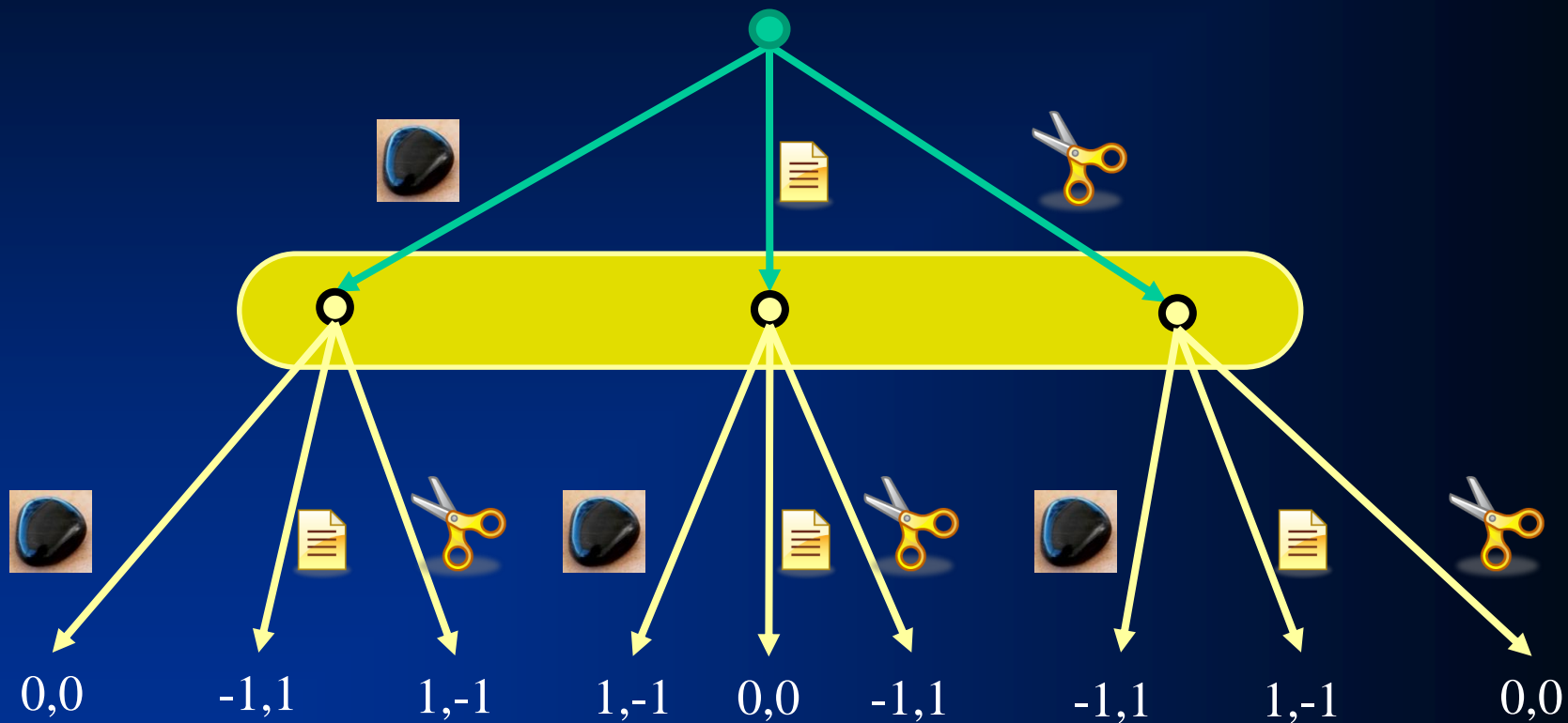


Endgame strategies computed
in real time in finer abstraction

Benefits of endgame solving

- Finer-grained information and action abstraction (helps in practice)
 - Dynamically selecting coarseness of action abstraction
- New information abstraction algorithms that take into account relevant distribution of players' types entering the endgames
- Computing exact (rather than approximate) equilibrium strategies
- Computing equilibrium refinements
- Solving the “off-tree” problem
- ...

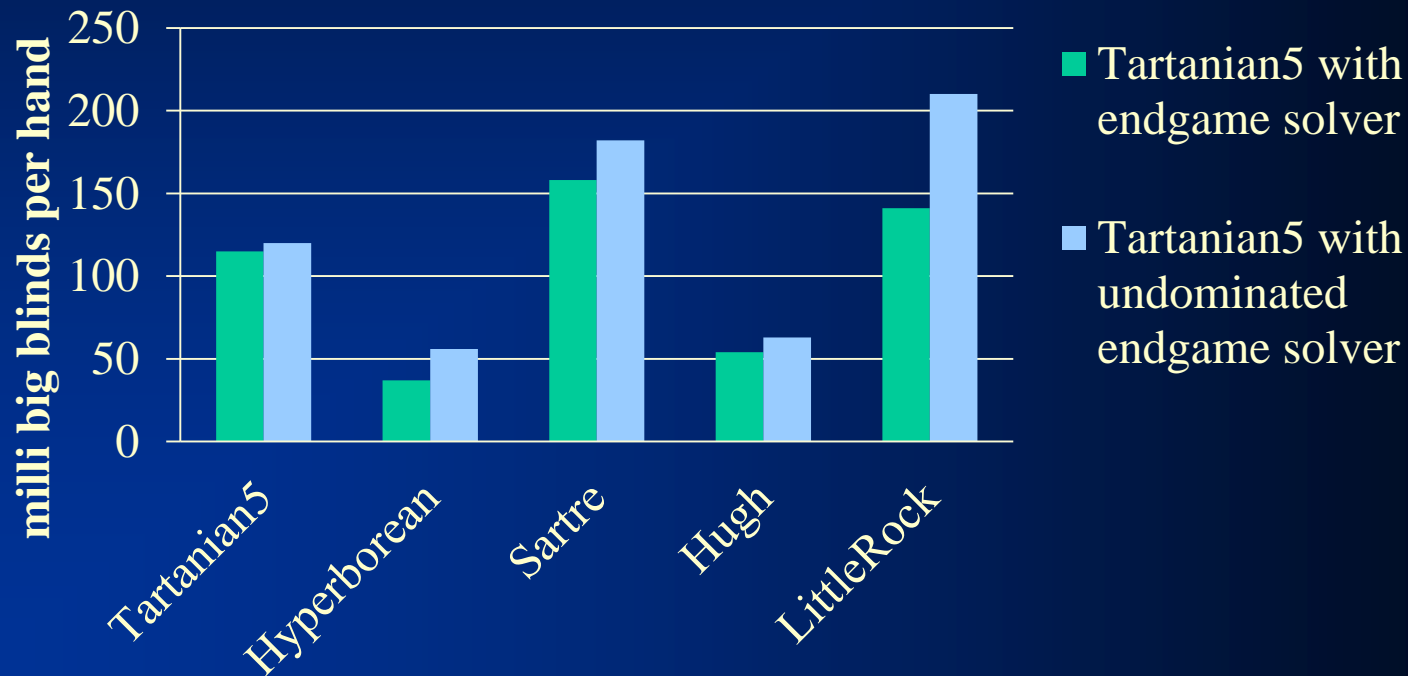
Limitation of endgame solving



Experiments on No-limit Texas Hold'em

- Solved last betting round in real time using CPLEX LP solver
 - Abstraction dynamically chosen so the solve averages 10 seconds

Improvement from adding endgame solver to Tartanian5



Top competitors from 2012

Computing equilibria by leveraging qualitative models

Player 1's strategy Player 2's strategy

Weaker hand
↑
Stronger hand
↓

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

Computing equilibria by leveraging qualitative models

Player 1's strategy Player 2's strategy

Weaker hand
↑
Stronger hand
↓

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

- **Theorem.** Given F_1, F_2 , and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium

Computing equilibria by leveraging qualitative models

Player 1's strategy Player 2's strategy

Weaker hand
↑
Stronger hand
↓

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

- **Theorem.** Given F_1, F_2 , and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium

Computing equilibria by leveraging qualitative models

Player 1's strategy Player 2's strategy

Weaker hand
↑
Stronger hand
↓

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

	BLUFF
CHECK-FOLD	
	CHECK
CHECK-CALL	
	BET

- **Theorem.** Given F_1, F_2 , and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium

Computing equilibria by leveraging qualitative models

Player 1's strategy Player 2's strategy

Weaker hand
↑
Stronger hand
↓

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

BLUFF-FOLD	FOLD/BLUFF
CHECK-FOLD	FOLD/CHECK
	BLUFF/CHECK
CHECK-CALL	CALL/CHECK
BET-FOLD	CALL/BET
	RAISE/BET

	BLUFF
CHECK-FOLD	
	CHECK
CHECK-CALL	
	BET

- **Theorem.** Given F_1, F_2 , and a qualitative model, we have a complete mixed-integer linear feasibility program for finding an equilibrium
- Qualitative models can enable proving existence of equilibrium & solve games for which algorithms didn't exist

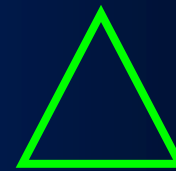
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm



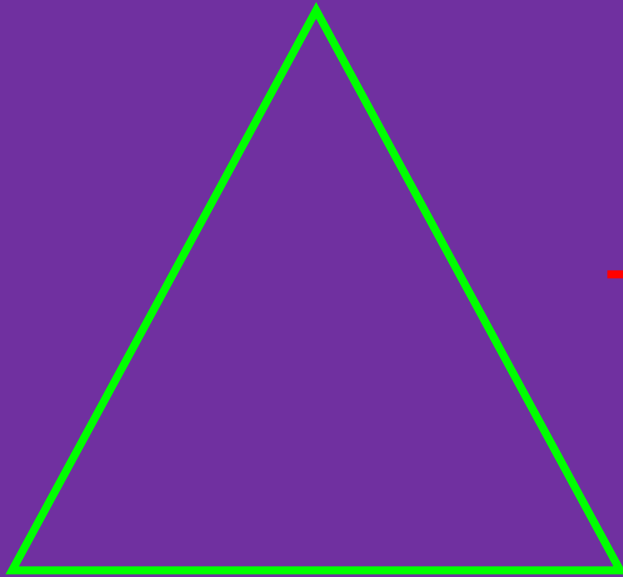
Nash equilibrium

Reverse model



Nash equilibrium

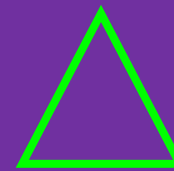
Original game



Automated abstraction



Abstracted game



Custom equilibrium-finding algorithm



Nash equilibrium

Reverse model



Nash equilibrium

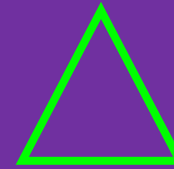
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm



Nash equilibrium

Reverse model



Nash equilibrium

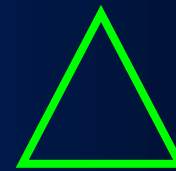
Original game



Automated abstraction



Abstracted game



Custom
equilibrium-finding
algorithm

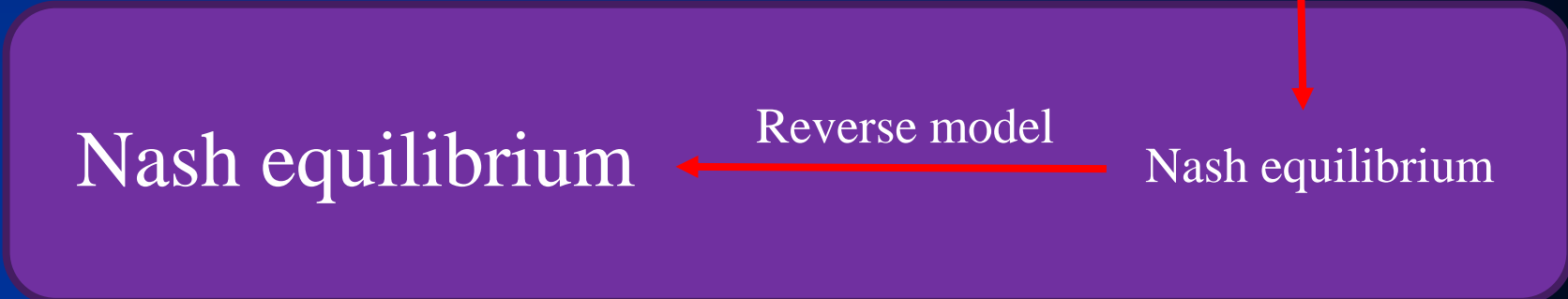


Nash equilibrium

Reverse model



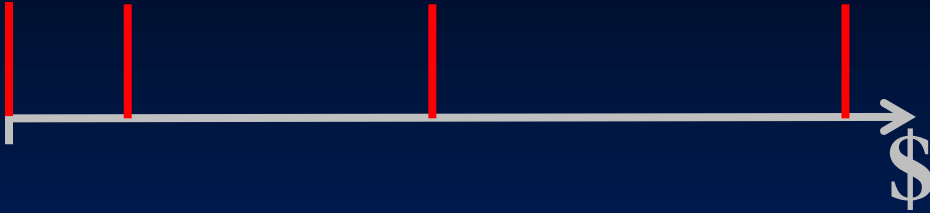
Nash equilibrium



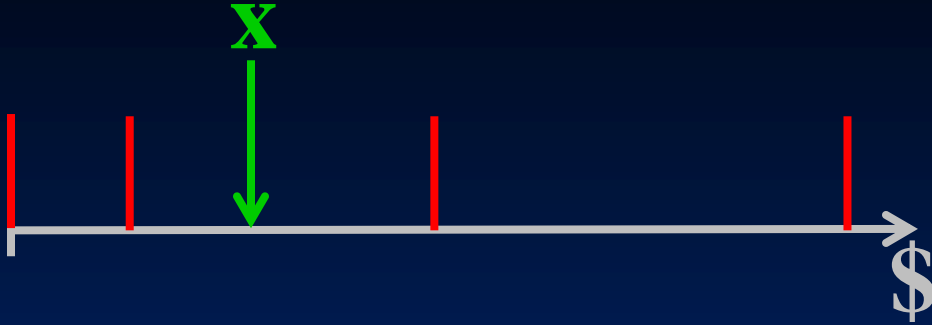
Action translation



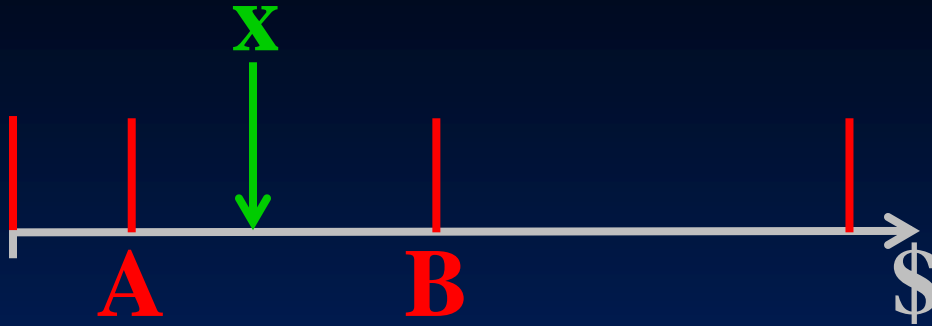
Action translation



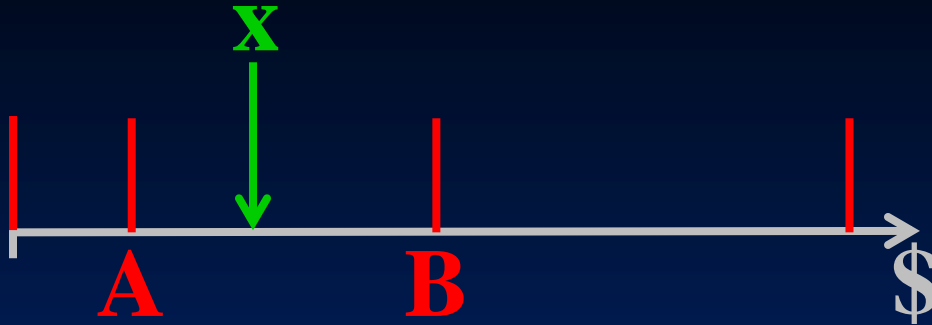
Action translation



Action translation

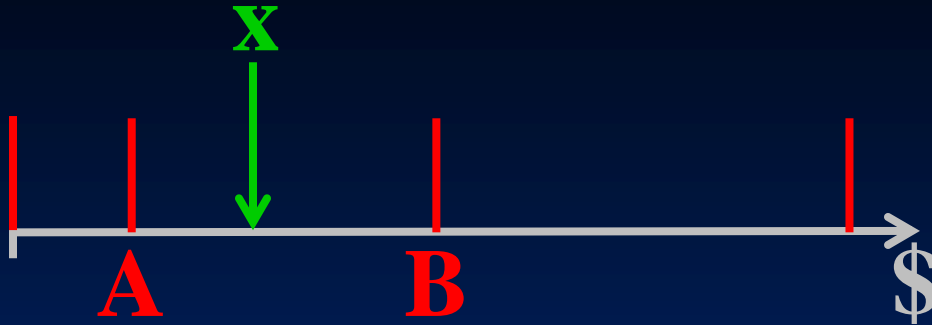


Action translation



$f(x) \equiv$ probability we map x to A

Action translation

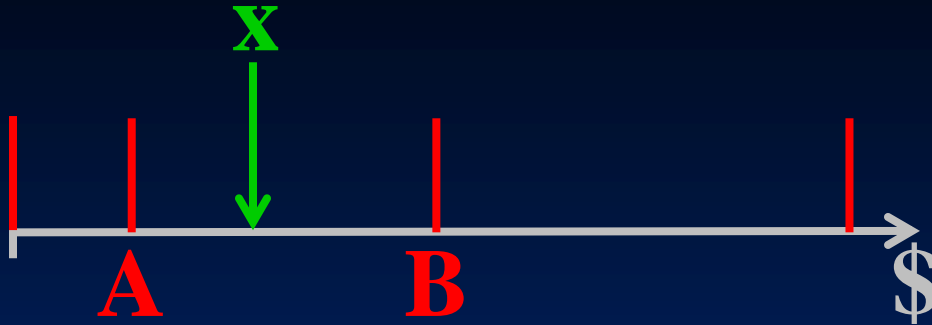


$f(x) \equiv$ probability we map x to A

Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

Action translation



“Pseudo-harmonic mapping”

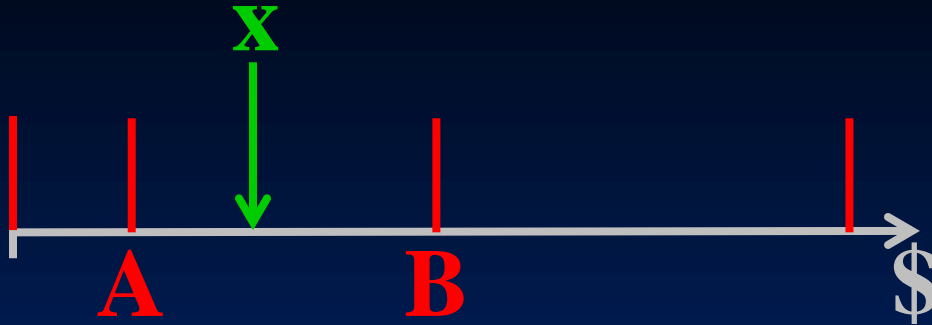
- $f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]$

$f(x) \equiv$ probability we map x to A

Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

Action translation



$f(x) \equiv$ probability we map x to A

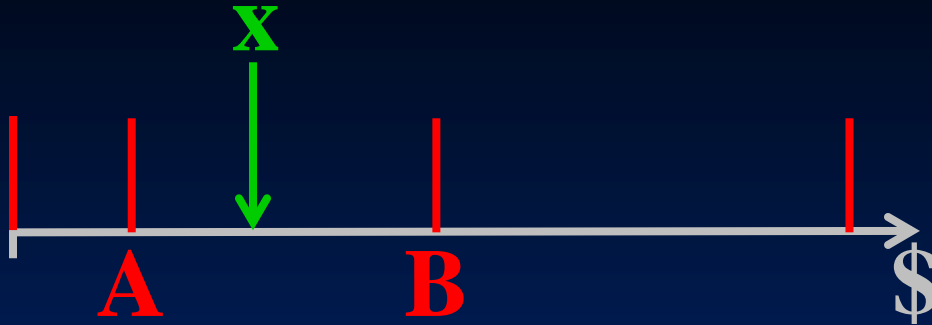
“Pseudo-harmonic mapping”

- $f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]$
- Derived from Nash equilibrium of a simplified no-limit poker game

Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

Action translation



$f(x) \equiv$ probability we map x to A

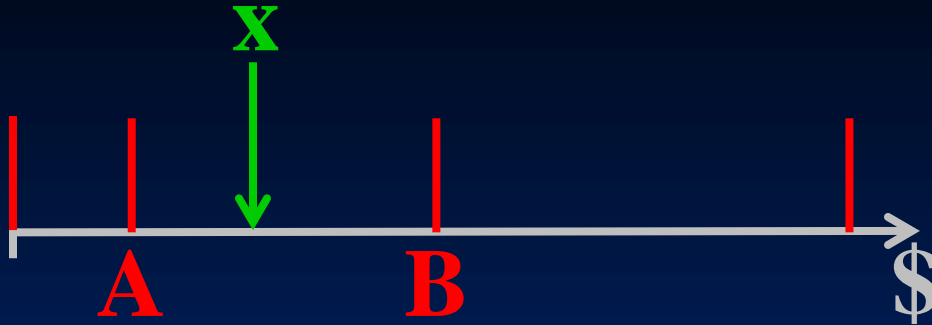
“Pseudo-harmonic mapping”

- $f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]$
- Derived from Nash equilibrium of a simplified no-limit poker game
- Satisfies the desiderata

Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

Action translation



$f(x) \equiv$ probability we map x to A

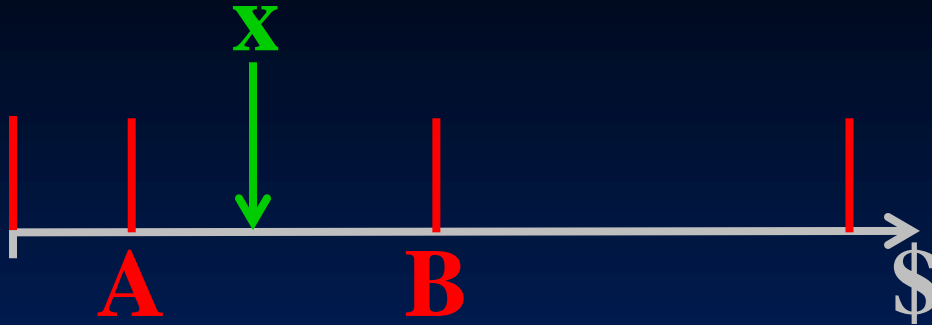
Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

"Pseudo-harmonic mapping"

- $f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]$
- Derived from Nash equilibrium of a simplified no-limit poker game
- Satisfies the desiderata
- Much less exploitable than prior mappings in simplified domains

Action translation



$f(x) \equiv$ probability we map x to A

Desiderata about f

1. $f(A) = 1$, $f(B) = 0$
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

"Pseudo-harmonic mapping"

- $f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]$
- Derived from Nash equilibrium of a simplified no-limit poker game
- Satisfies the desiderata
- Much less exploitable than prior mappings in simplified domains
- Performs well in practice in no-limit Texas Hold'em
 - Significantly outperforms randomized geometric

OPPONENT EXPLOITATION

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**
 - Needs prohibitively many repetitions to learn in large games (loses too much during learning)
 - Crushed by game theory approach in Texas Hold'em
 - Same would be true of no-regret learning algorithms
 - *Get-taught-and-exploited problem* [Sandholm AIJ-07]

Let's hybridize the two approaches

- Start playing based on game theory approach
- As we learn opponent(s) deviate from equilibrium, start adjusting our strategy to exploit their weaknesses
 - Requires no prior knowledge about the opponent

Deviation-Based Best Response algorithm

(generalizes to multi-player games)

- Compute an approximate equilibrium
- Maintain counters of opponent's play throughout the match
- **for** $n = 1$ **to** |public histories|
 - Compute posterior action probabilities at n (using a Dirichlet prior)
 - Compute posterior bucket probabilities
 - Compute model of opponent's strategy at n
- **return** best response to the opponent model

Many ways to define opponent's "best" strategy that is consistent with bucket probabilities

- L_1 or L_2 distance to equilibrium strategy
- Custom weight-shifting algorithm, ...

Experiments on opponent exploitation

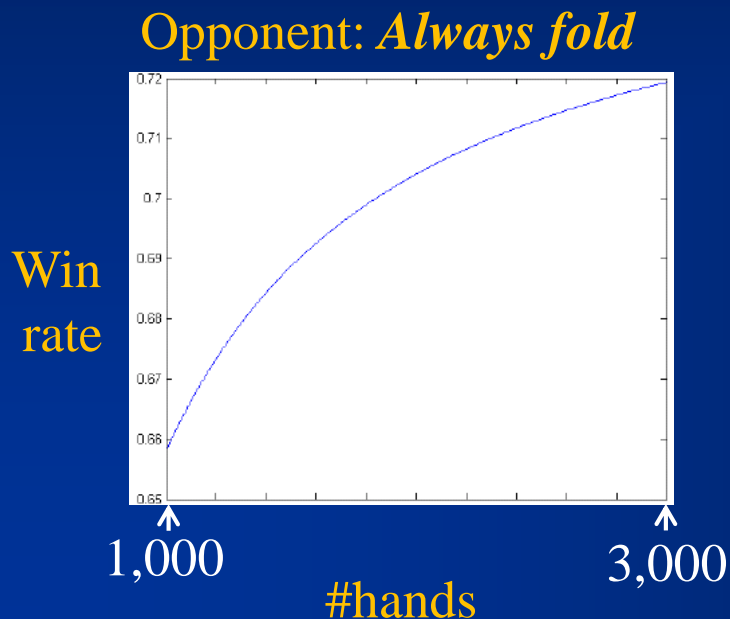
- Significantly outperforms game-theory-based base strategy in 2-player limit Texas Hold'em against
 - trivial opponents
 - weak opponents from AAAI computer poker competitions

Experiments on opponent exploitation

- Significantly outperforms game-theory-based base strategy 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

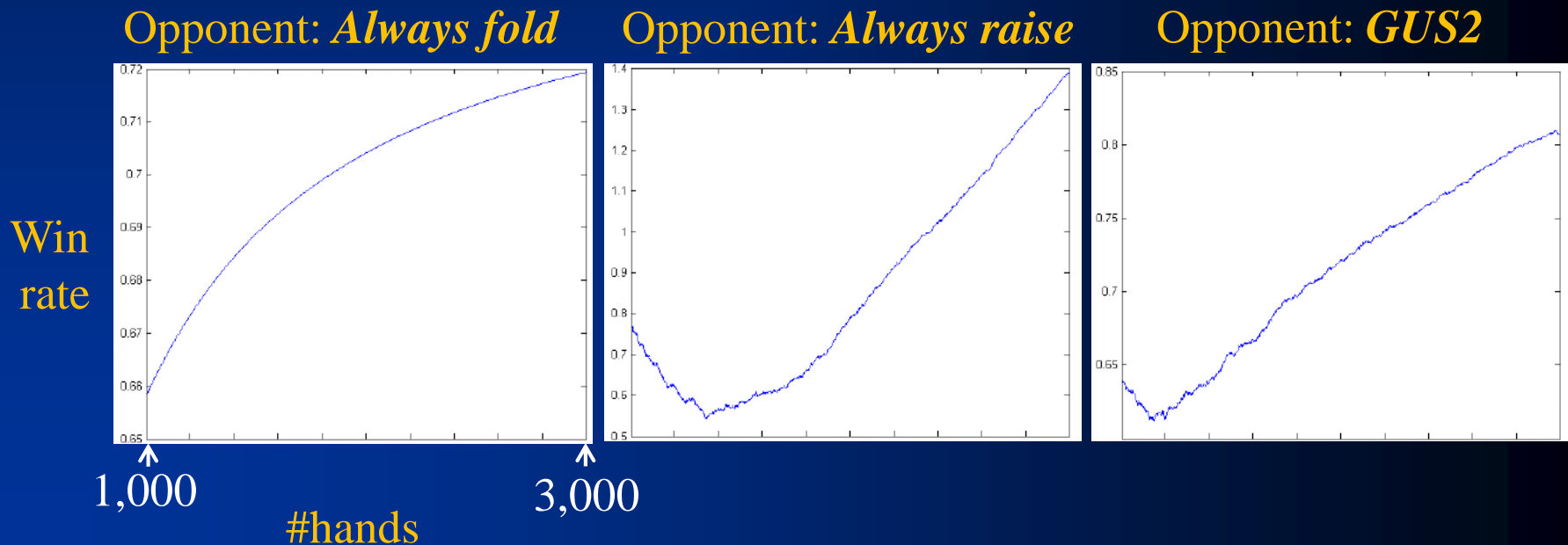
Experiments on opponent exploitation

- Significantly outperforms game-theory-based base strategy 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



Experiments on opponent exploitation

- Significantly outperforms game-theory-based base strategy 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



Other modern approaches to opponent exploitation

- ϵ -safe best response

[Johanson, Zinkevich & Bowling NIPS-07, Johanson & Bowling AISTATS-09]

- Precompute a small number of strong strategies.
Use no-regret learning to choose among them

[Bard, Johanson, Burch & Bowling AAMAS-13]

Safe opponent exploitation

- 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?~~

R is a gift
but not iteratively weakly dominated

	L	M	R
U	3	2	10
D	2	3	0

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

R isn't in the support of any equilibrium
but is also not a gift

	L	R
U	0	0
D	-2	1

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


Exploitation algorithms

1. Risk what you've won so far



Exploitation algorithms

1.  Risk what you've won so far



Exploitation algorithms

1.  Risk what you've won so far
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



Exploitation algorithms

1.  Risk what you've won so far
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



Exploitation algorithms

1.  Risk what you've won so far
 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
- ...



Exploitation algorithms

1.  Risk what you've won so far
 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
- ...
- **Theorem.** A strategy for a 2-player 0-sum game is safe iff it never risks more than the gifts received according to #2



Exploitation algorithms

1.  Risk what you've won so far
 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
- ...
- **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 model / exploitation algorithm safe



Exploitation algorithms

1.  Risk what you've won so far
 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
- ...
- **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 model / exploitation algorithm safe
 - No prior (non-eq) opponent exploitation algorithms are safe

Exploitation algorithms

1.  Risk what you've won so far
 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
- ...
- **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 model / exploitation algorithm safe
 - No prior (non-eq) opponent exploitation algorithms are safe
 - #2 experimentally better than more conservative safe exploitation algs

Exploitation algorithms

1.  Risk what you've won so far
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

...

- **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 model / exploitation algorithm safe
- No prior (non-eq) opponent exploitation algorithms are safe
- #2 experimentally better than more conservative safe exploitation algs
- Suffices to lower bound opponent's mistakes

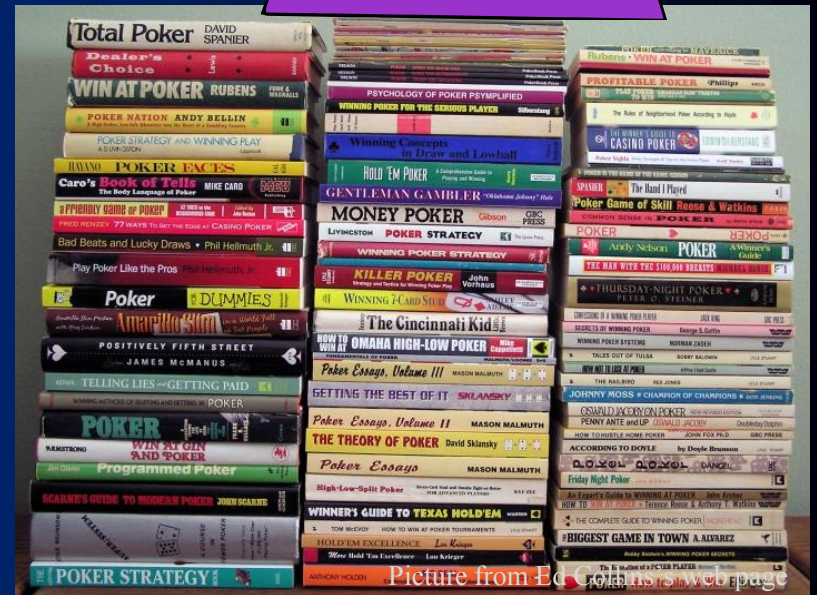
Bots versus top pros

- 2-player poker
 - Rhode Island Hold'em: Bots play optimally [Gilpin & Sandholm EC-06, J. of the ACM 2007]
 - Limit Texas Hold'em: Bots surpassed pros in 2008 [U. Alberta Poker Research Group]



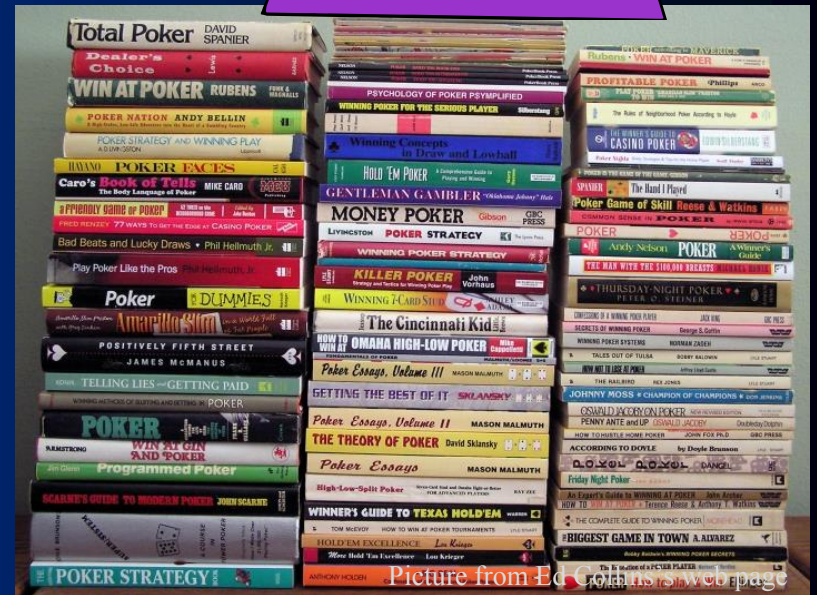
- No-Limit Texas Hold'em: Bots surpass pros soon?
- Multiplayer poker: Bots aren't very strong yet

Learning from bots



Picture from Ed Collins's web page

Learning from bots



Ground truth

Picture from Ed Collins's webpage

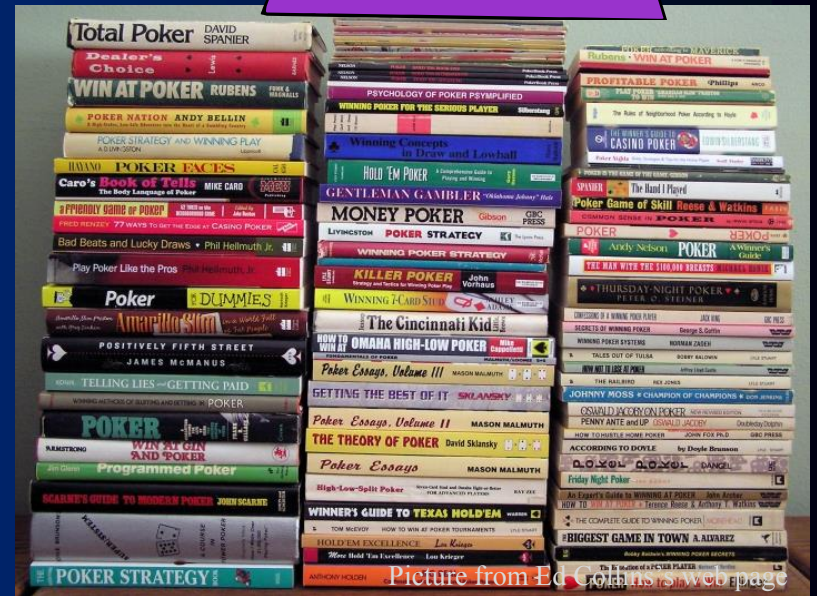
Learning from bots



0.03647, 0.39408, 0.0,
0.43827, 0.0, 0.0, 0.04147,
...



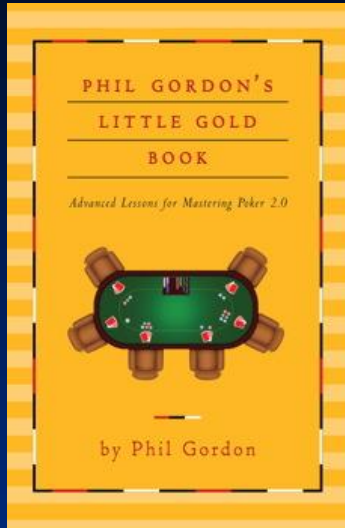
Ground truth



Picture from Ed Collins's webpage

First action:

To fold, “limp”, or raise (the typical 1×pot)?



"Limping is for Losers

This is *the most important fundamental* in poker--for every game, for every tournament, every stake: If you are the first player to voluntarily commit chips to the pot, open for a raise. Limping is inevitably a losing play. If you see a person at the table limping, you can be fairly sure he is a bad player. Bottom line: If your hand is worth playing, it is worth raising."

Daniel Cates: “we're going to play 100% of our hands...We will raise ... We will be making small adjustments to that strategy depending on how our opponent plays ... Against the most aggressive players ... it is acceptable to fold the very worst hands ..., around the bottom 20% of hands. It is probably still more profitable to play 100% ...”

With 91.1% of hands, our bot randomizes between limp and raise (plus with one hand it always limps)

- Probability mix not monotonic in hand strength
- Aggregate limping probability is 8.0%

“Donk bet”

- A common sequence in 1st betting round:
 - First mover raises, then second mover calls
 - The latter has to move first in the second betting round. If he bets, that is a “donk bet”
- Considered a poor move
- Our bot donk bets ~8% of the time

1 or more bet sizes (for a given betting sequence and public cards)?

- Using more than 1 risks signaling too much
- Most pros use 1 (some sometimes use 2)
 - Typical bet size is $1 \times \text{pot}$ in the first betting round, and between $\frac{2}{3} \times \text{pot}$ and $\frac{3}{4} \times \text{pot}$ in later rounds
- Our bot sometimes randomizes between 6 sizes (even with a given hand)
 - Both with bluff hands and “value hands”
 - Includes unusually small and large bets (all-in $37 \times \text{pot}$)

Conclusions

- Domain-independent techniques
- Game abstraction
 - Automated lossless abstraction - exactly solved game with billions of nodes
 - Practical lossy abstraction: integer programming, potential-aware, imperfect recall
 - Automated lossy abstraction with bounds
 - For action and state abstraction
 - Also for modeling
- Equilibrium-finding
 - Can solve 2-person 0-sum games with over 10^{14} nodes to small ϵ
 - $O(1/\epsilon^2) \rightarrow O(1/\epsilon) \rightarrow O(\log(1/\epsilon))$
 - Purification and thresholding help
 - Endgame solving helps
 - Leveraging qualitative models \Rightarrow existence, computability, speed, insight
- Scalable practical online opponent exploitation algorithm
- Fully characterized safe exploitation & provided algorithms
- New poker knowledge

Current & future research

- Lossy abstraction with bounds
 - General sequential games
 - With structure
 - With generated abstract states and actions
- Equilibrium-finding algorithms for 2-person 0-sum games
 - Understanding the selective superiority of CFR and EGT
 - Making gradient-based algorithms work with imperfect recall
 - Parallel implementations of our $O(\log(1/\epsilon))$ algorithm and understanding how #iterations depends on matrix condition number
 - Making interior-point methods usable in terms of memory
- Equilibrium-finding algorithms for >2 players [Ganzfried and Sandholm AAMAS-08, IJCAI-09]
- Theory of thresholding, purification, and other strategy restrictions
- Other solution concepts: sequential equilibrium, coalitional deviations, ...
- Understanding exploration vs exploitation vs safety
- Applying these techniques to other games

Thank you

Students & collaborators:

- Sam Ganzfried
- Andrew Gilpin
- Noam Brown
- Javier Peña
- Sam Hoda
- Troels Bjerre Sørensen
- Satinder Singh
- Kevin Waugh
- Kevin Su

Sponsors:

- NSF
- Pittsburgh Supercomputing Center
- IBM
- Intel
- Comments, figures, etc.: Michael Bowling, Michael Johansen, Ariel Procaccia, Christina Fong