Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization

AAMAS 2012 - June 6, 2012

Michael Johanson, Nolan Bard, Marc Lanctot,
Richard Gibson, Michael Bowling
University of Alberta

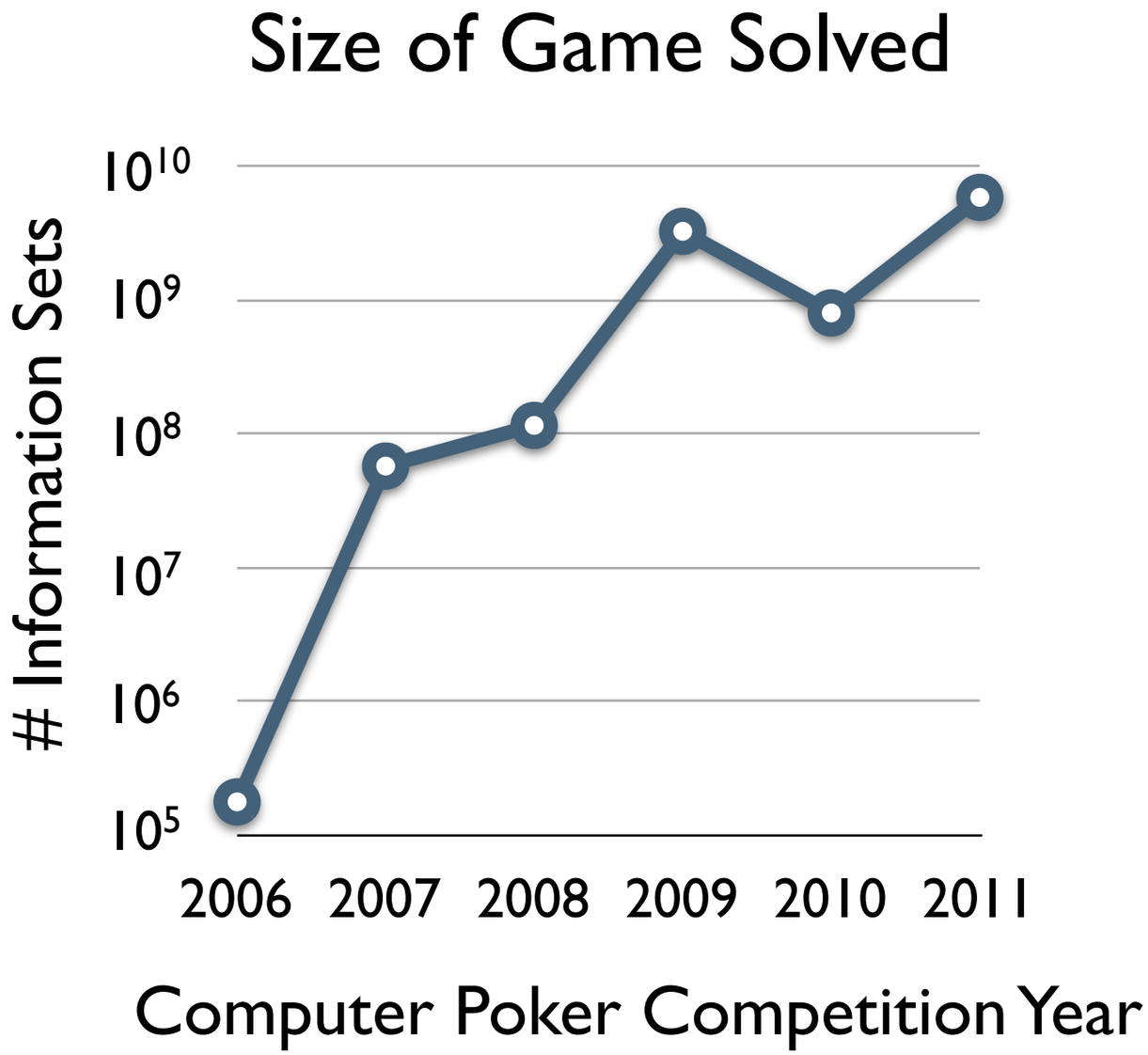
Motivation

Tackling the practical challenge of Nash equilibrium computation in large games

- ♥ Strategy that is guaranteed to not lose on expectation (2-player, zero-sum)

- ♣ Very useful property in practice:
  - ♦ Dominant approach in the Annual Computer Poker Competition
  - ♠ 2008: beat human professionals at 2-player limit Texas hold'em poker

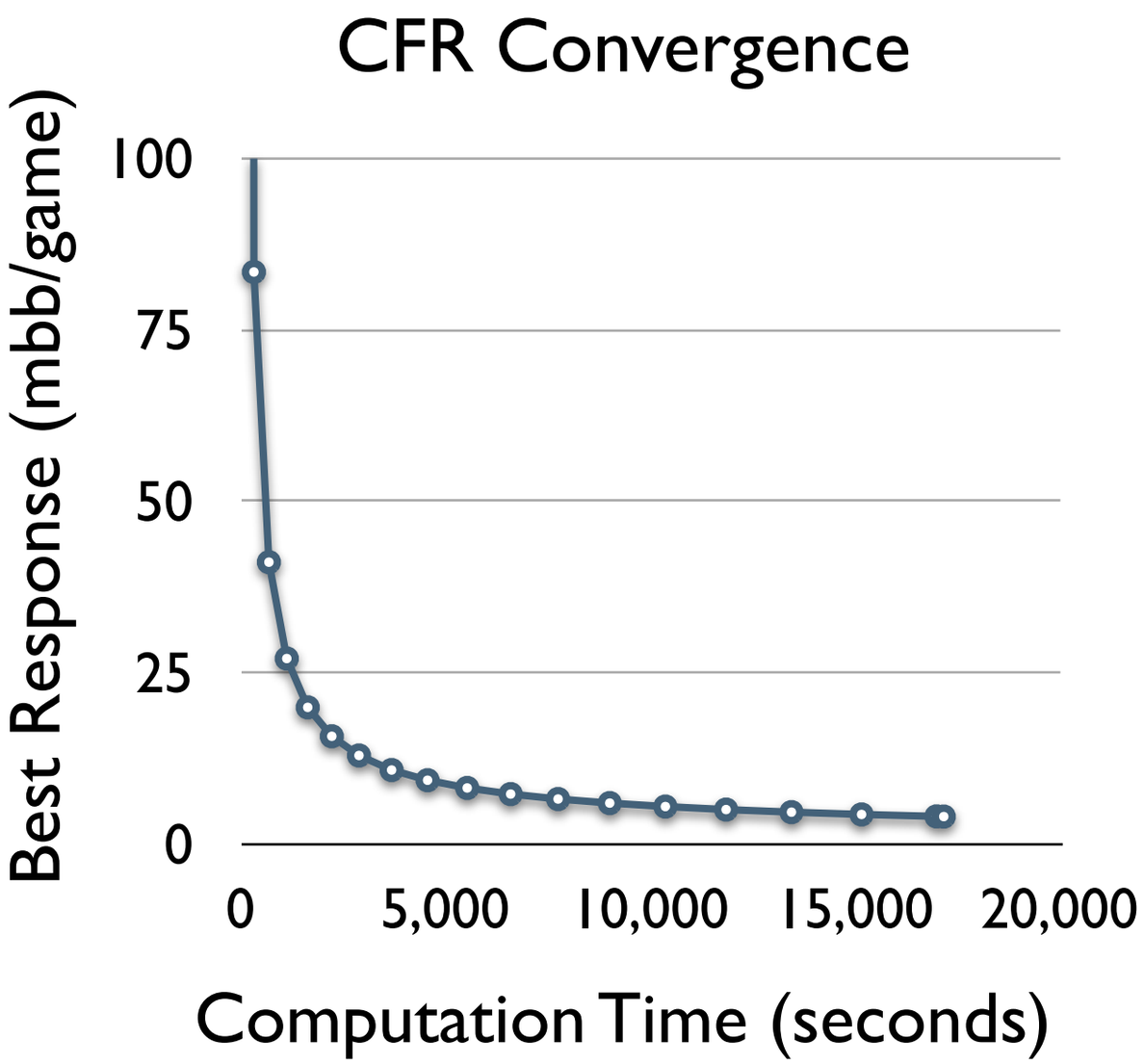
Motivation



The poker community is now solving games with 10^9 decisions (information sets).

LPs don't scale to this size of game. We've made great progress on efficient approximation algorithms. (CFR, EGT)

Counterfactual Regret Minimization (CFR), NIPS 2007



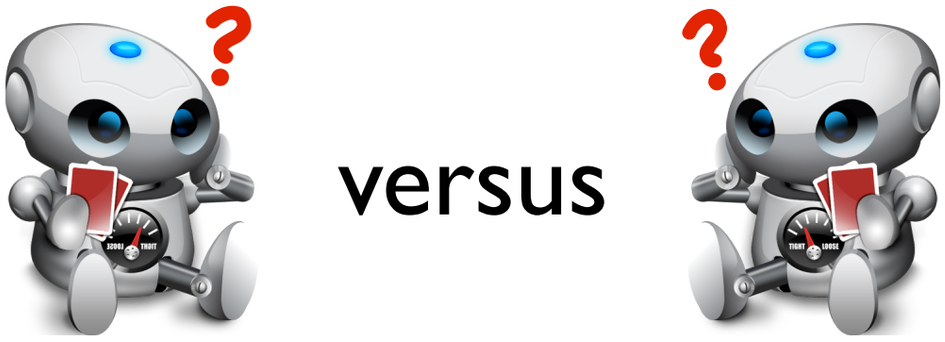
CFR is the competition's most popular algorithm.

Iterative, resembles self-play; reinforcement learning flavour.

- ♦ Memory efficient (2 doubles per infoset-action)
- ♣ Converges quickly ($1/\varepsilon^2$)
- ♥ Programmer Friendly
  - ♠ Easy to implement and optimize
  - ♦ Linear speedup with many cores

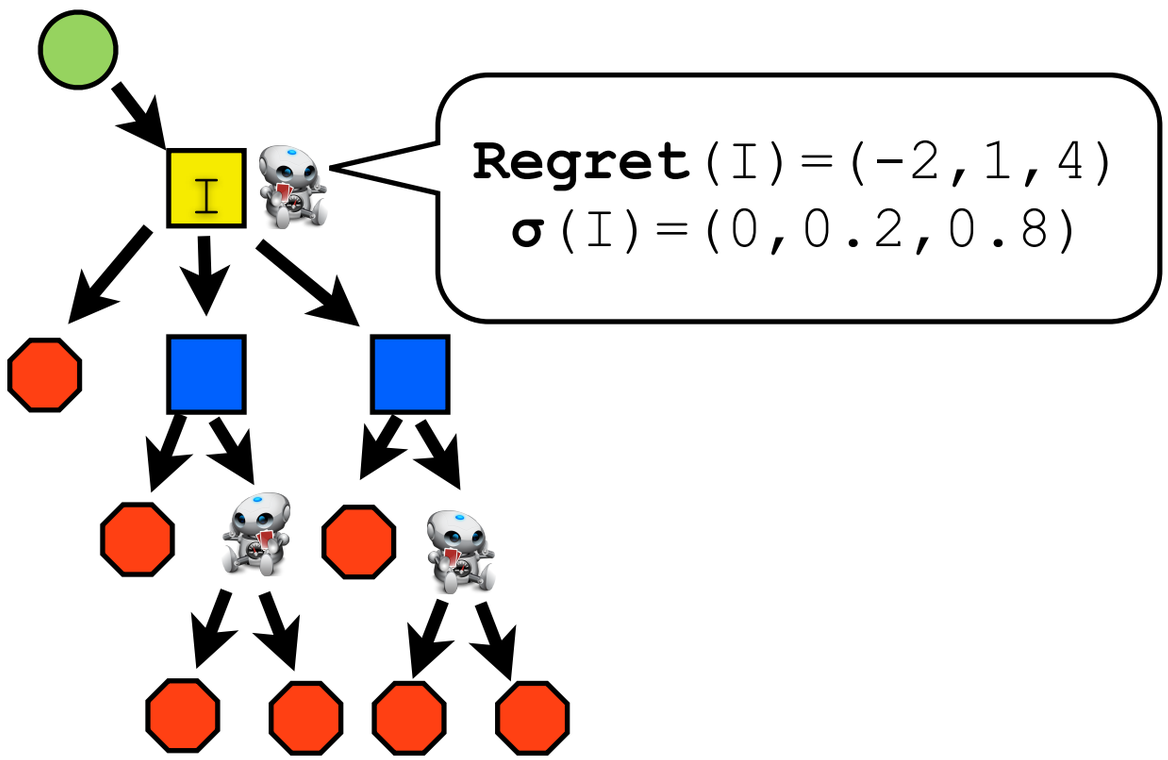
This paper: a new CFR variant that converges more quickly in imperfect information games.

Counterfactual Regret Minimization (CFR)



Basic idea:

♥ Start with two uniform random strategies. Play them against each other.



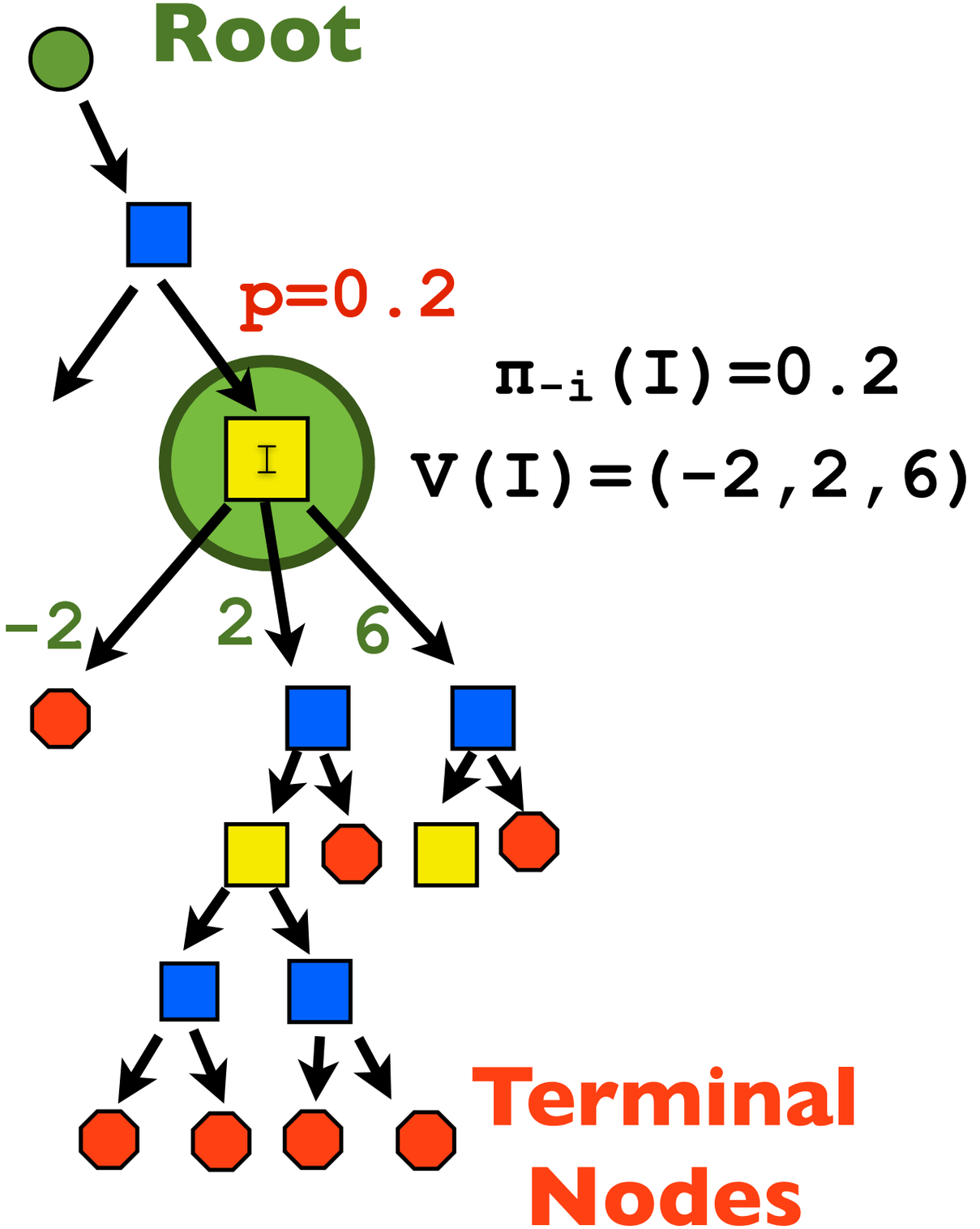
♣ Put a regret minimizing agent at every decision, and let it independently learn its part of the strategy.

♦ Run many iterations: walk the game tree, agents update their parts of the strategy.

$\bar{\sigma}$ → **Nash Equilibrium**

♠ Average strategy profile converges to equilibrium.

Counterfactual Regret Minimization (CFR)



To update a decision, we need:

♥ Probability of other players taking their series of actions

♣ Expected value (or unbiased estimate) of actions' utilities given opponent's strategy

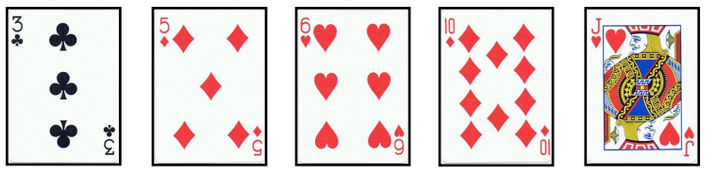
Recursively walk the tree:

♦ Push forwards opponent action probabilities

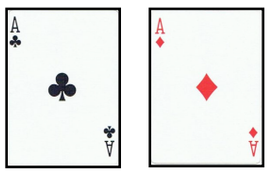
♠ Return EV at this terminal node or in this subtree

Chance-Sampled CFR

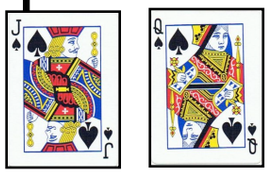
Public Chance

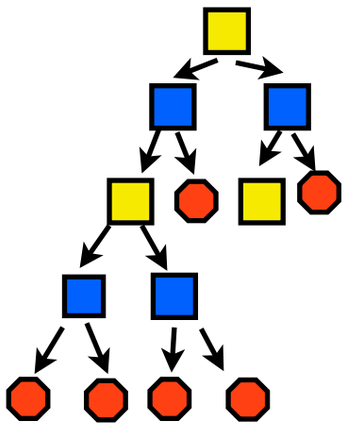


My Private Chance



Opponent Private Chance





```
Recursion:
PASS one scalar
  (opponent reach
   probability)
RETURN one scalar
  (value of subgame)
```

- In practice, a sampling variant of CFR is used.
- Chance Sampling: on each iteration, randomly sample one set of chance events and only update that part of the tree.
- Terminal nodes: Get an unbiased estimate of my state's value. Takes O(1) time.

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Self-Public Chance Sampling (CS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

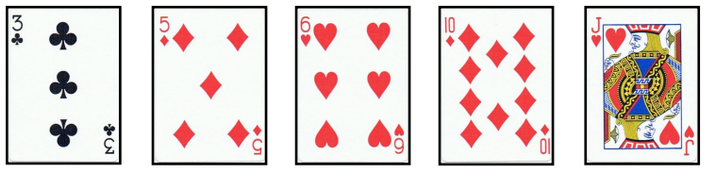
Public Chance Sampling (PCS)

Sample: Public chance

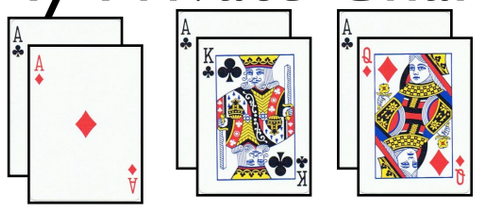
Expand: My Private Chance, Opponent Private Chance

Opponent-Public Chance Sampling (OPCS)

Public Chance

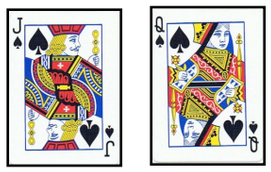


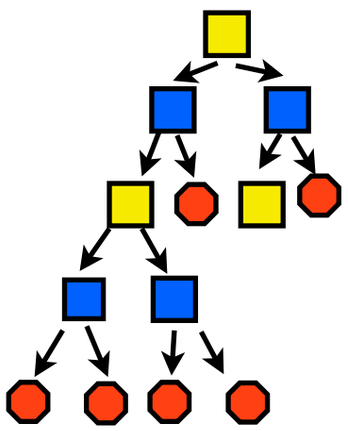
My Private Chance



...(45 choose 2)

Opponent Private Chance





```
Recursion:
PASS one scalar
  (opponent reach
   probability)
RETURN a vector
  (values of
   subgames
```

♥ Sample one Public chance event

♣ Sample one opponent private chance event

♦ Enumerate all of my possible private chance events

♠ KEY OBSERVATION: Opponent can't observe my chance event, so their strategy is the same for all of them. I can efficiently update all of these decisions in the same recursive pass!

♥ Terminal nodes: *n* states to evaluate. Takes O(*n*) time.

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Slower,
Many updates
per iteration

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Self-Public Chance Sampling (SPCS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

Public Chance Sampling (PCS)

Sample: Public chance

Expand: My Private Chance, Opponent Private Chance

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Slower,
Many updates
per iteration

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Self-Public Chance Sampling (SPCS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

Public Chance Sampling (PCS)

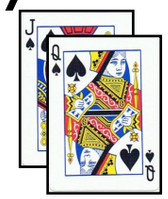
Sample: Public chance

Expand: My Private Chance, Opponent Private Chance

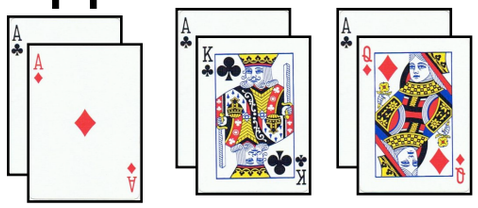
Self-Public Chance Sampling (SPCS)

Public Chance

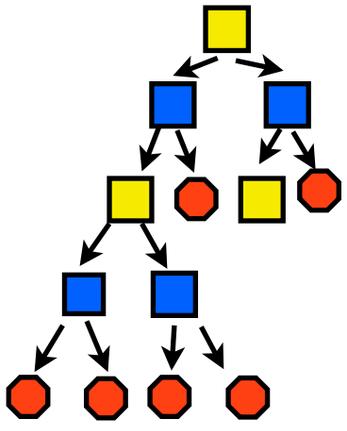
My Private Chance



Opponent Private Chance



...(45 choose 2)



```
Recursion:
PASS one vector
  (opponent reach
   probabilities)
RETURN one scalar
  (value of subgame)
```

♥ Sample one Public chance event

♣ Sample one of my private chance events

♦ Enumerate all of opponent's possible private chance events

♠ Terminal nodes: *n* states to evaluate. Much more precise estimate of my value, since I compare my state to all of theirs!

♥ RESULT:
Slow but very precise updates.

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Slower,
Many updates
per iteration

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Slower,
Very precise
updates

Self-Public Chance Sampling (SPCS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

Public Chance Sampling (PCS)

Sample: Public chance

Expand: My Private Chance, Opponent Private Chance

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Slower, Many updates per iteration

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Slower, Very precise updates

Self-Public Chance Sampling (SPCS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

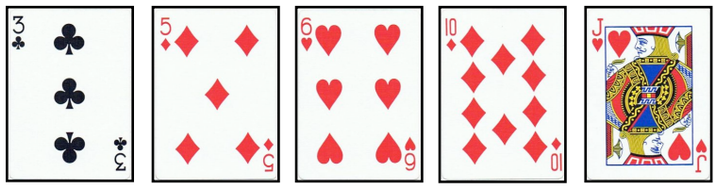
Public Chance Sampling (PCS)

Sample: Public chance

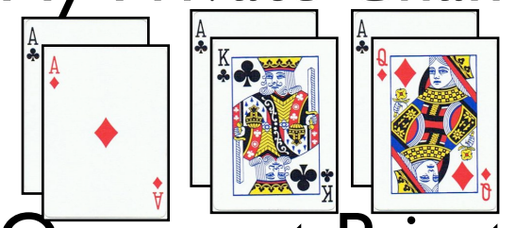
Expand: My Private Chance, Opponent Private Chance

Public Chance Sampling (PCS)

Public Chance

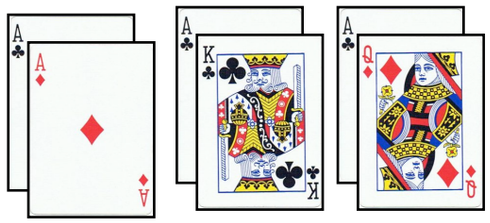


My Private Chance

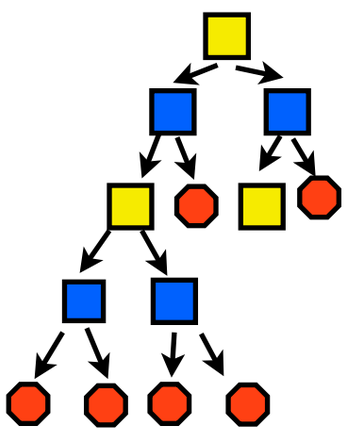


...(47 choose 2)

Opponent Private Chance



...(47 choose 2)



```
Recursion:
PASS one vector
  (opponent reach
   probability)
RETURN one vector
  (value of subgame)
```

♥ Sample one Public chance event

♣ Enumerate all of my private chance events

♦ Enumerate all of opponent's possible private chance events

♠ Terminal nodes: *n* states to evaluate against n states. Looks like $O(n^2)$ work. But depending on game structure, $O(n)$ is often possible, making it as fast as OPCS or SPCS!

RESULT:

♥ Slower, but do many precise updates on each iteration.

New CFR Sampling Variants

Chance Sampling (CS)

Sample: Public chance, My Private Chance, Opponent Private Chance

Slower, More updates per iteration

Opponent-Public Chance Sampling (OPCS)

Sample: Opponent Private Chance, Public chance

Expand: My Private Chance

Slower, Very precise updates

Same speed, very precise updates

Self-Public Chance Sampling (SPCS)

Sample: My Private Chance, Public chance

Expand: Opponent Private Chance

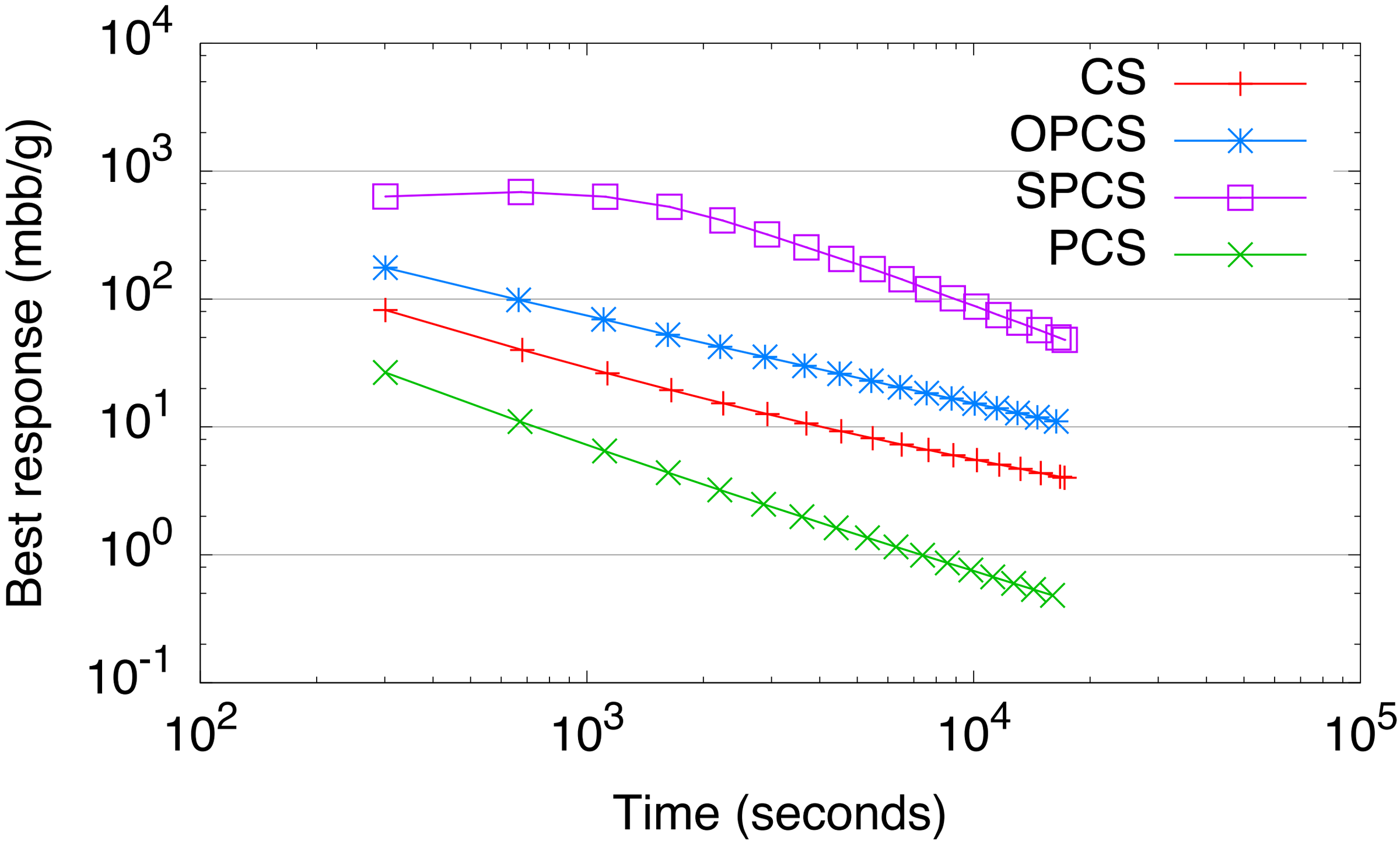
Same speed, many updates per iteration

Public Chance Sampling (PCS)

Sample: Public chance

Expand: My Private Chance, Opponent Private Chance

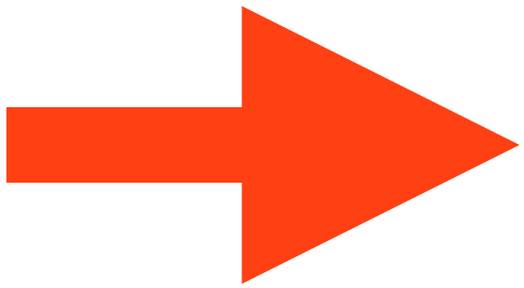
Results: 2-round, 4-bet Poker
94 million decision points (information sets)



Abstracted Limit Texas Hold'em Poker

Real Poker Game

$3*10^{14}$ Decisions (infosets)



Abstraction

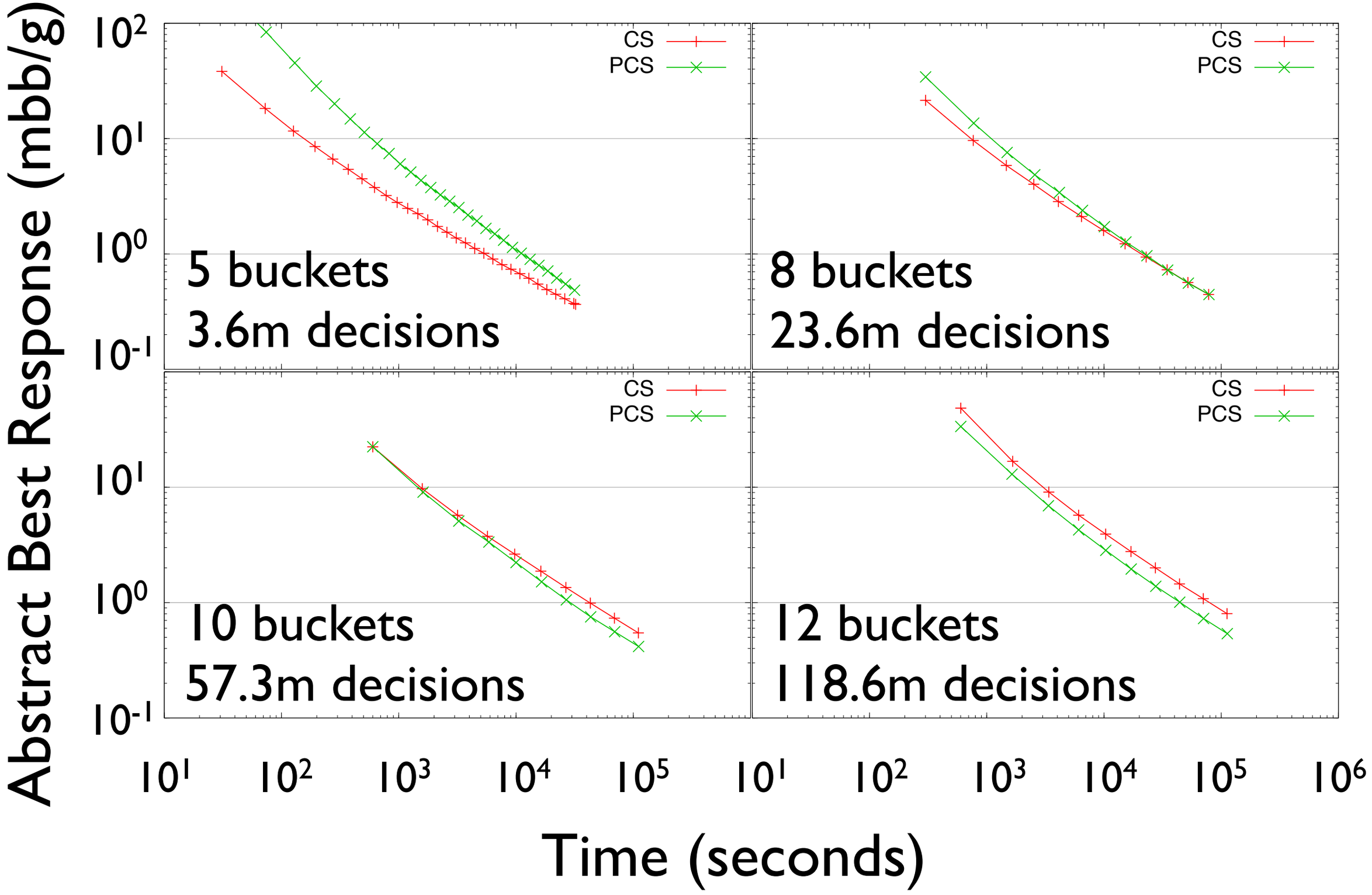
Abstract Poker Game

10^{9} Decisions (infosets)

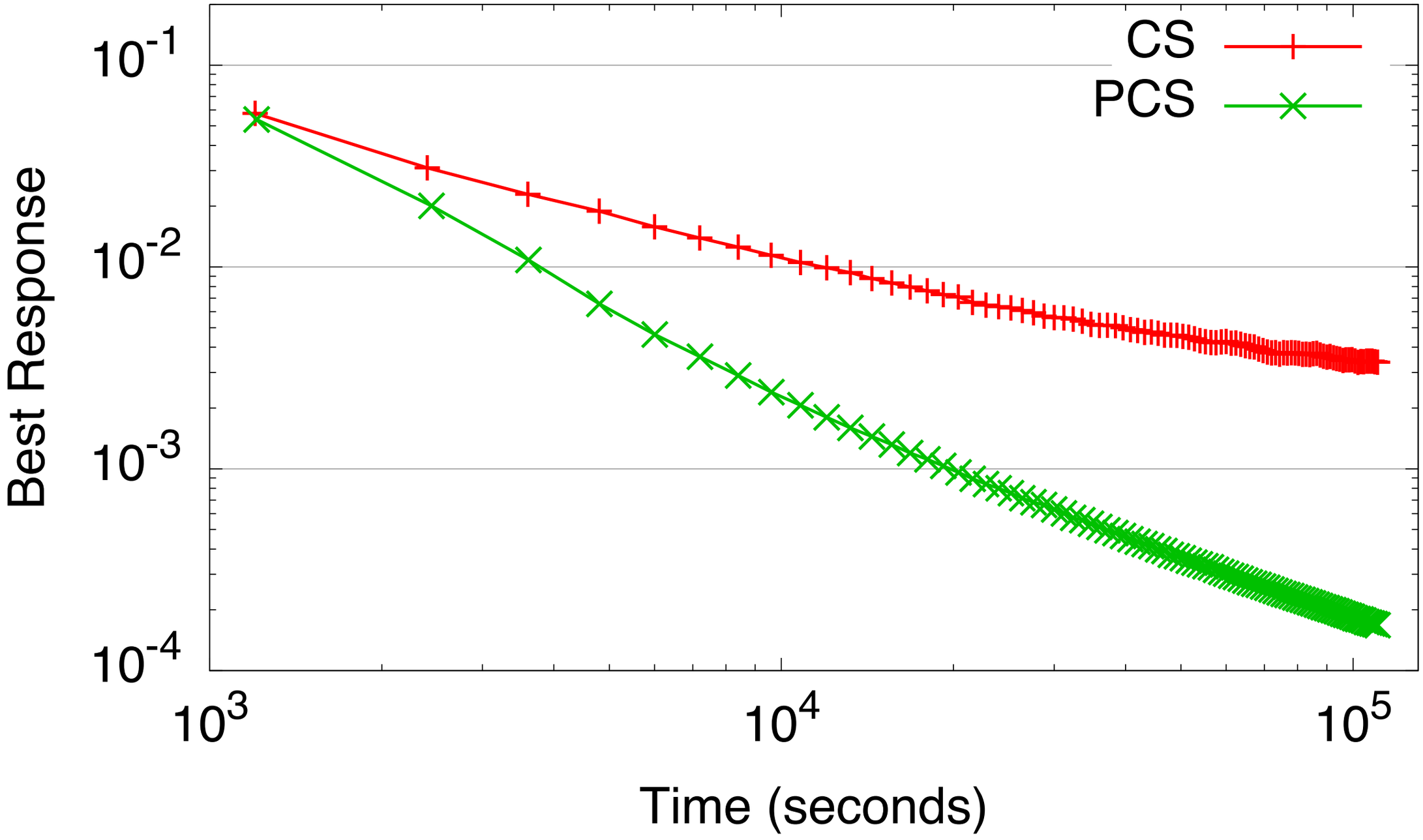
Larger abstractions are better in practice, but take longer to solve.

Can evaluate by measuring exploitability in abstract game.

Results: Abstracted Limit Texas Hold'em Poker



Alternate domain:
Bluff, an imperfect information dice game



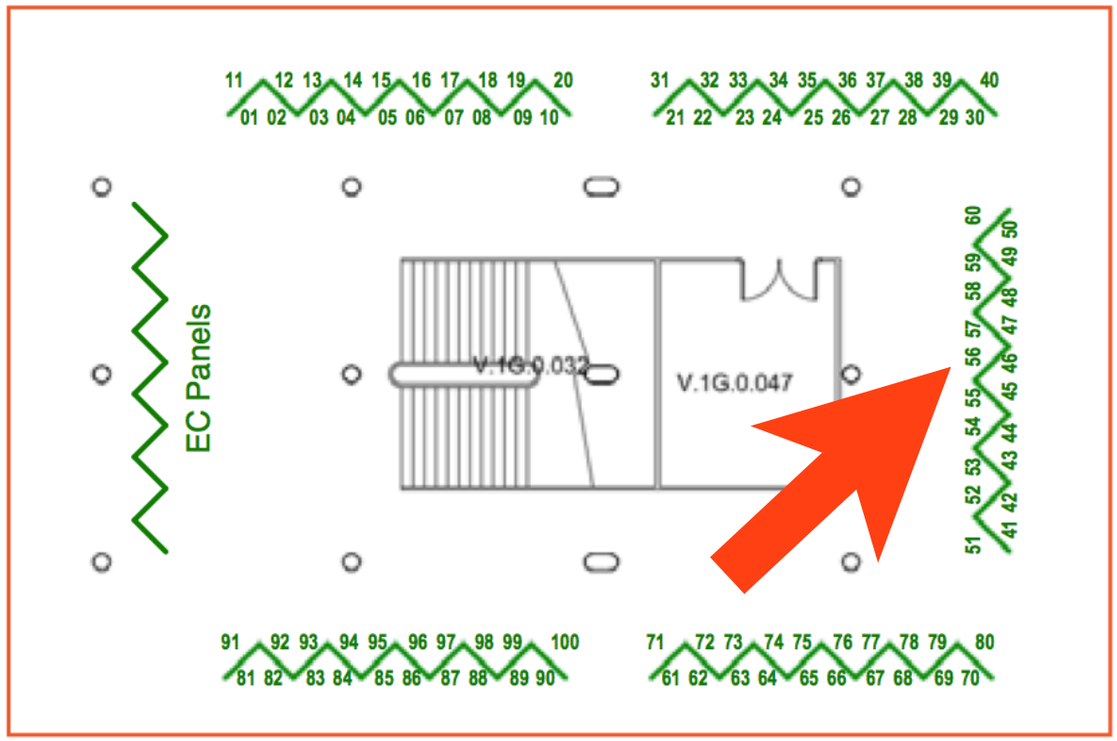
Conclusion

- ♥ Counterfactual Regret Minimization is a state-of-the-art algorithm for Nash equilibrium approximation in 2-player zero sum games.

- ♣ Public Chance Sampled CFR:
  - ♦ Takes advantage of structure of imperfect information
  - ♠ Converges faster in practice

Thanks!
Poster: Panel 056

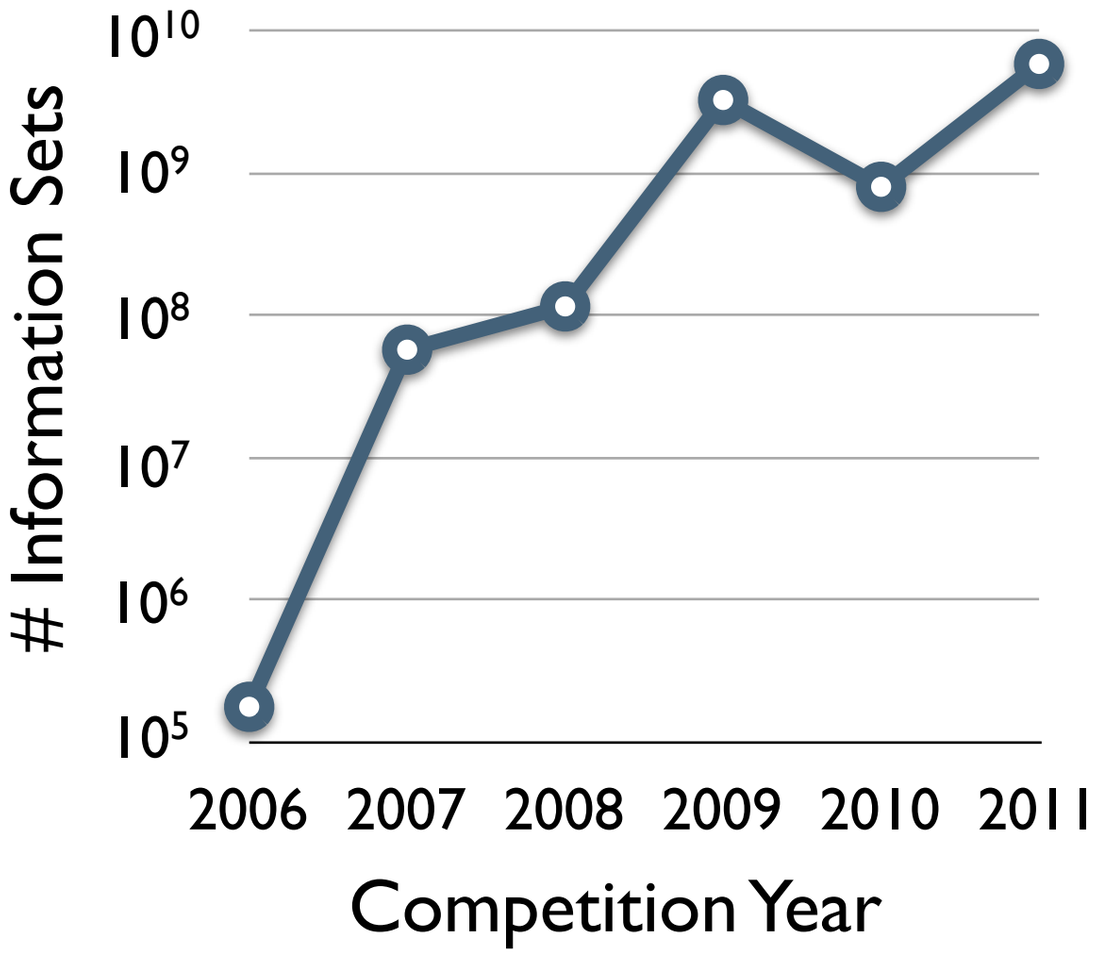
Ground Floor



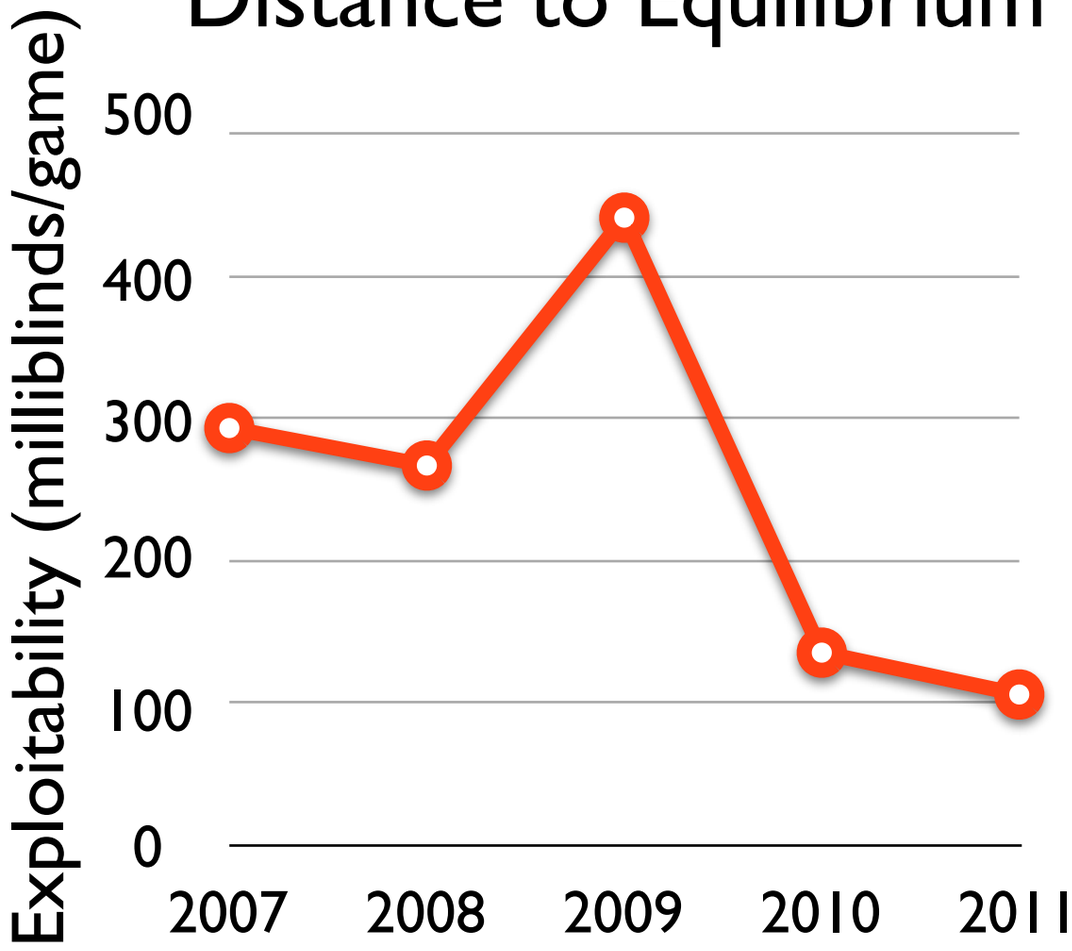
University of Alberta
Computer Poker Research Group

Games Solved for the Annual Computer Poker Competition

Size of Game Solved



Distance to Equilibrium



Results: Limit Texas Hold'em Poker
One-on-One performanace against a strong opponent (Hyperborean2010.IRO)