Efficient Nash Equilibrium Computation through Monte Carlo Counterfactual Regret Minimization

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

University of Alberta
Computer Poker Research Group

Counterfactual Regret Minimization (CFR)

- In two-player zero-sum games, **Nash equilibrium** strategies (minimax strategies) are **unexploitable**: they will do no worse than tie on expectation against any opponent.
- **CFR** is a state-of-the-art iterative algorithm for approximating Nash equilibria in two-player zero-sum games. It resembles self-play over a series of T games.
- By **minimizing regret** (improving the strategy) at each decision point independently, the entire strategy converges towards a Nash equilibrium.
- CFR is **memory efficient**, **straightforward to implement**, and **easy to optimize and parallelize**.
- Monte-Carlo CFR is a family of **sampling variants** that converge much faster in practice than the base algorithm. This paper proposes **Public Chance Sampling** and shows that it converges faster than earlier approaches.

"Vanilla" CFR, 2007
In each iteration, enumerate all chance events and update the complete game tree. Not useful in practice.

↓ Very fast but noisy iterations.

Chance Sampling (CS), 2007
O(1) terminal node evaluation
[2-4] speed: 1.25m iter/sec
Sample one event for us
Update our strategy considering one opponent private chance event.

→ Slower iterations, Lower variance

Self / Public Chance Sampling (SPCS)
O(n) terminal node evaluation
[2-4] speed: 1952 iter/sec
Sample one event for us, but update while considering all *n* opponent private chance events.

↓ Slower iterations, More updates per iteration (from CS)

↓ Same time complexity, More updates per iteration (from SPCS)

Opponent / Public Chance Sampling (OPCS)
O(n) terminal node evaluation
[2-4] speed: 1414 iter/sec
Update all *n* of our chance events with respect to one sampled opponent event.

→ Same time complexity, Lower variance

Public Chance Sampling (PCS)
Possible O(n) terminal node evaluation
[2-4] speed: 709 iter/sec
Sample public chance events, but consider all *n* private events for each player

Algorithm outline:

Initialize two strategies and repeatedly traverse the game tree. This resembles a self-play algorithm.

At each decision *I*, use recursion to get the **value** of each action *a* and accumulate **regret**:

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)\left(u_i(\sigma^t|_{I\rightarrow a}, I) - u_i(\sigma^t, I)\right)$$

Update the strategies proportional to their accumulated positive regret:

$$\sigma_i^{T+1}(I,a) = \frac{R_i^{T,+}(I,a)}{\sum_{a\in A(I)} R_i^{T,+}(I,a)}$$

Following this procedure, the **average strategy** used by the players converges to a Nash equilibrium.

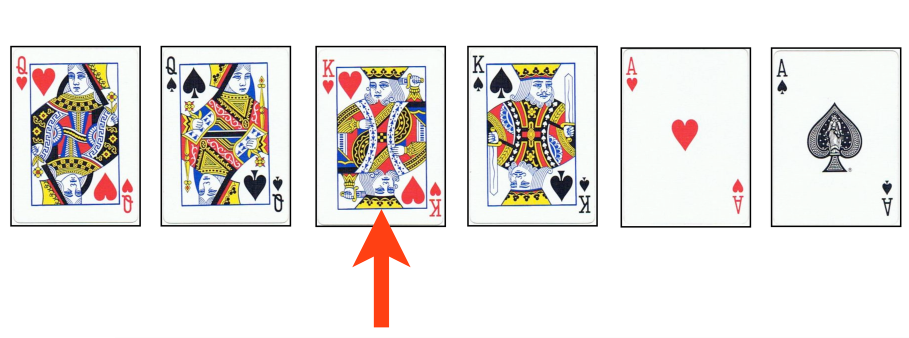
Sampling some or all of the chance events lets us perform fast, noisy updates, and this converges faster. **We can trade off between**:

- Iteration Speed
- Strategy updates per iteration
- Accuracy in estimating action values

Fast Terminal Node Evaluation (IJCAI 2011)

By exploiting game structure, a fast O(*n*) terminal evaluation may be possible when comparing *n* private states for each player.

This allows PCS to do the work of both OPCS and SPCS with the same time complexity!



Obvious O(n²) algorithm:

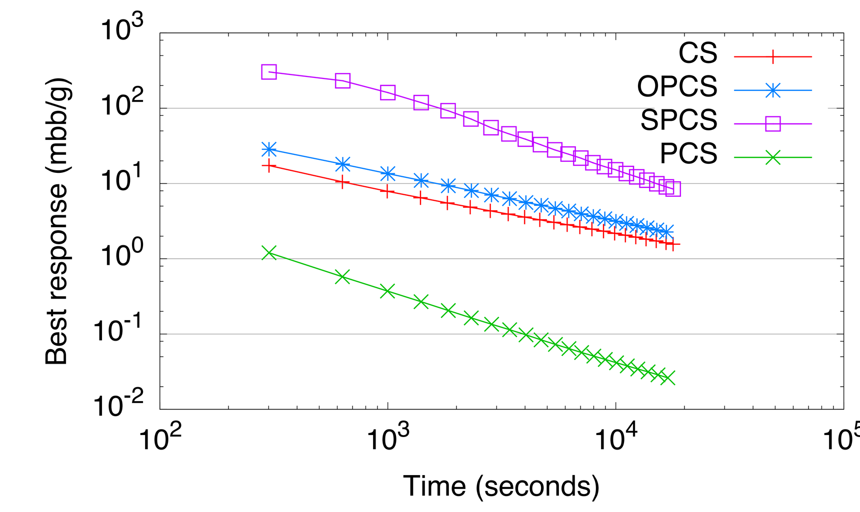
```
for( each of my hands x )
  for( each of their hands y )
    if( x > y )
      util[x] += payoff * P(y)
    else if( x < y )
      util[x] -= payoff * P(y)
```

Faster O(n) algorithm:

```
p_lose = total_prob;  p_win = 0;
for( each hand x )  //red arrow
above
  p_lose -= prob[x]
  util[x] = (p_win - p_lose)*payoff
  p_win += prob[x]
```

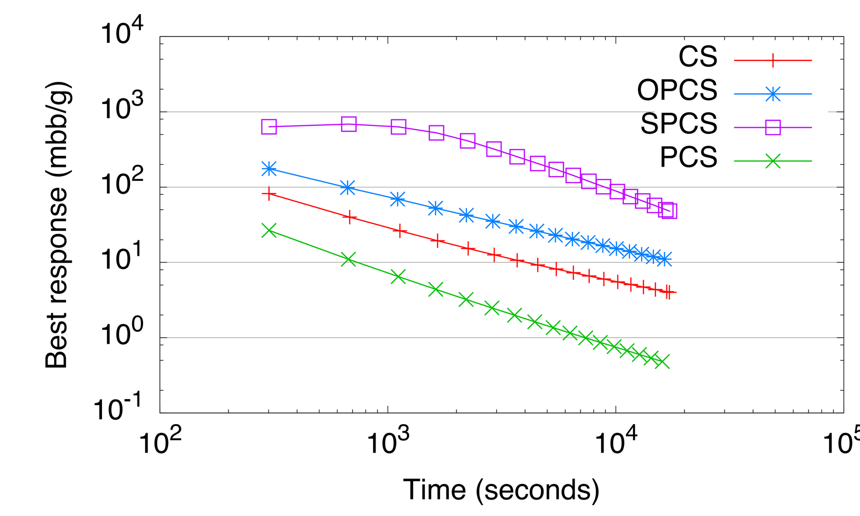
[2-Round, 1-Bet] Hold'em
A small poker game where strategies can be quickly created and evaluated.

- Y-axis shows distance to Nash equilibrium
- Game has 16 million information sets
- First PCS datapoint has already converged closer than final CS datapoint!



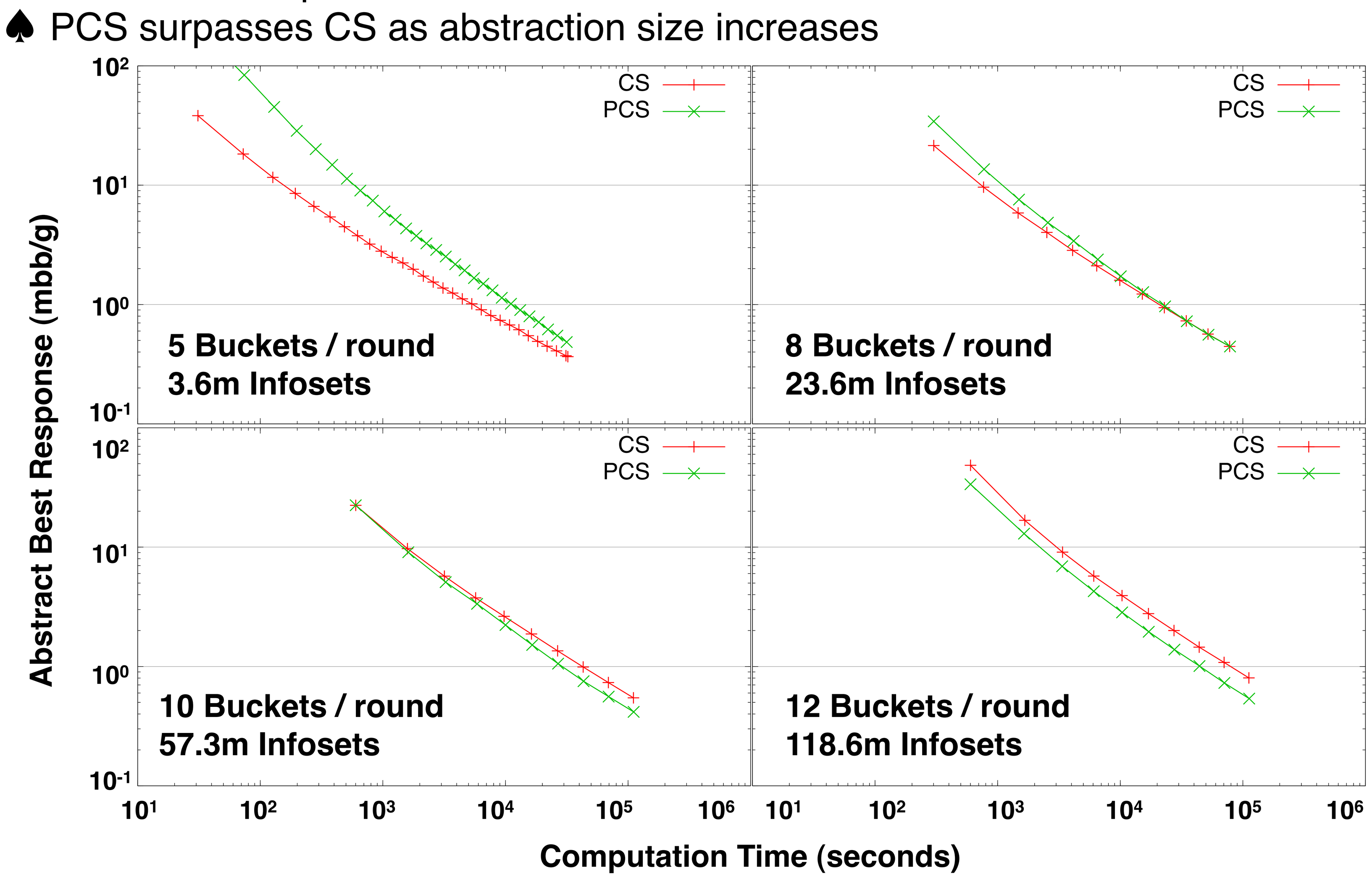
[2-Round, 4-Bet] Hold'em
A larger test domain that increases the players' action space

- 94 million information sets
- PCS curve is both lower and has a steeper slope



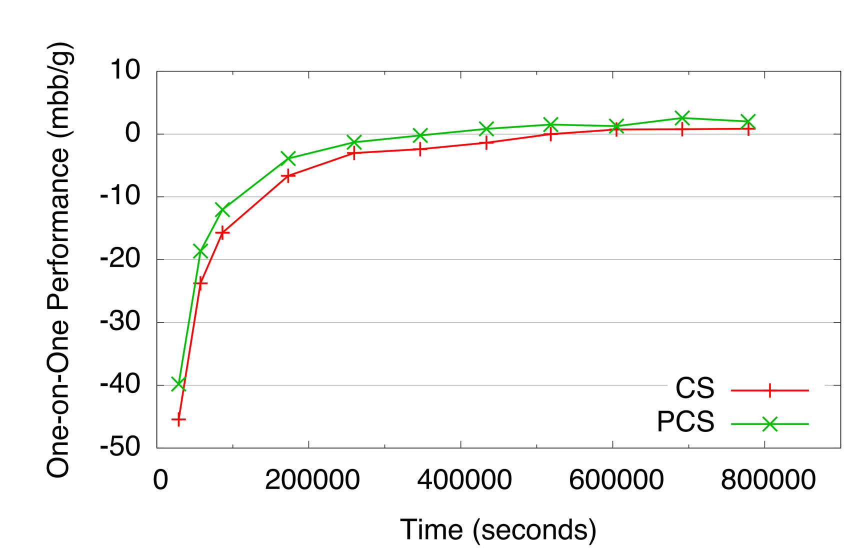
Limit Texas Hold'em: Abstract Best Response

- Real game: 10^14 information sets. Abstraction lets us produce tractable games.
- Increasing abstraction granularity results in better real game strategies, but increases computational costs
- PCS surpasses CS as abstraction size increases



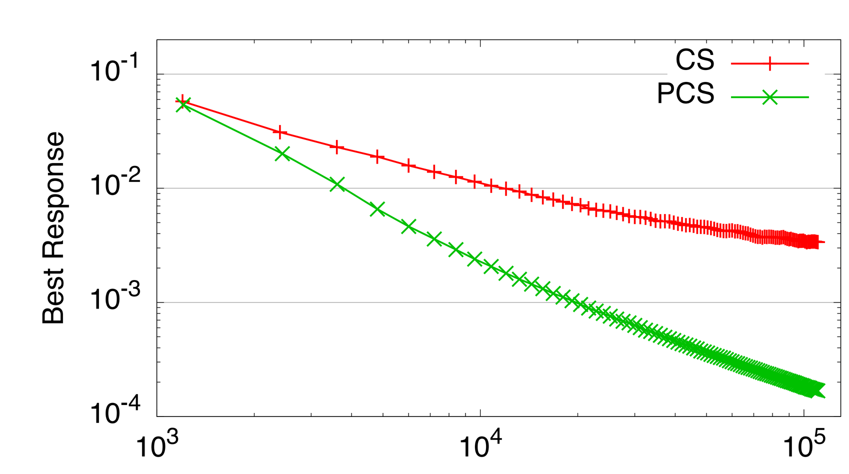
Limit Texas Hold'em: In-Game Performance

- In large abstractions, we can evaluate by in-game performance against a strong opponent (Hyperborean2011)
- Note the horizontal distance. CS must be run for much longer to reach the same level of performance.
- Abstraction has 880m information sets.



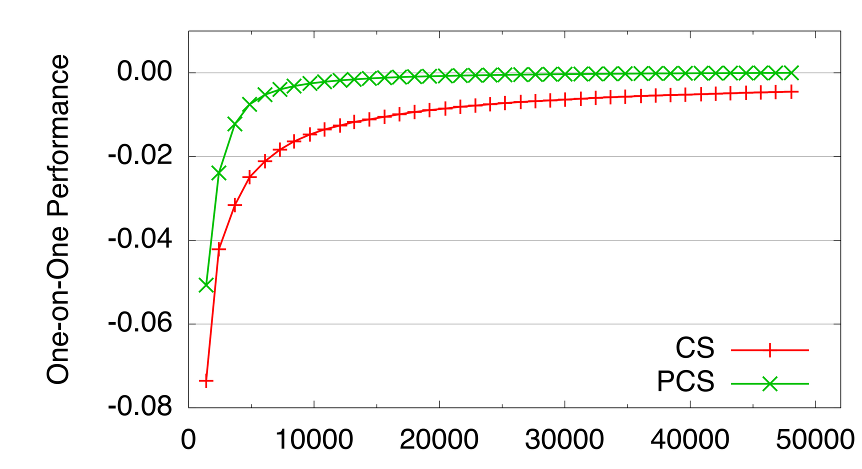
(2,2) Bluff: Exploitability

- Bluff is a 2-player dice game. Each player secretly rolls 2 dice and players bid on how many of each side was rolled.
- No public chance events, so PCS does efficient complete traversals.
- PCS' curve is both lower and steeper at each timestep.



(2,2) Bluff: In-Game Performance

- In this graph, we use the PCS and CS strategies to play against the final PCS strategy.
- PCS generates strong strategies much more quickly than CS. Consider the horizontal distance.



University of Alberta · Alberta Innovates · compute • calcul CANADA · WestGrid · NSERC CRSNG · Computing Science