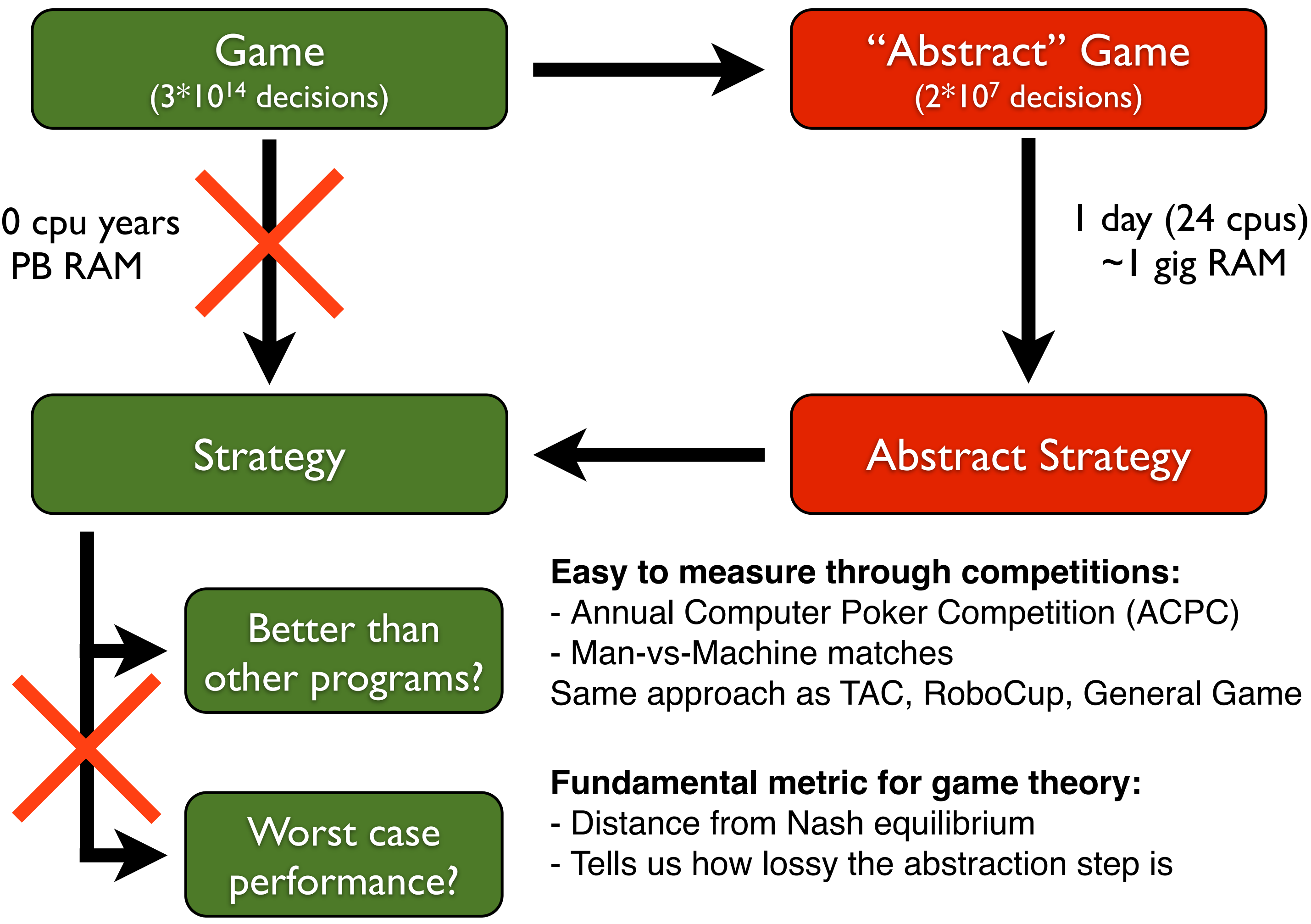
Accelerating Best Response Calculation in Large Extensive Games

Michael Johanson, Michael Bowling - University of Alberta
Kevin Waugh - Carnegie Mellon University
Martin Zinkevich - Yahoo! Inc

University of Alberta
Computer Poker Research Group

How strong are recent computer poker programs?

A popular approach to making computer poker agents: approximate a **Nash Equilibrium** strategy. This minimizes the worst-case loss.
- Ties against a worst-case opponent (a **best response**), and may win against weak opponents



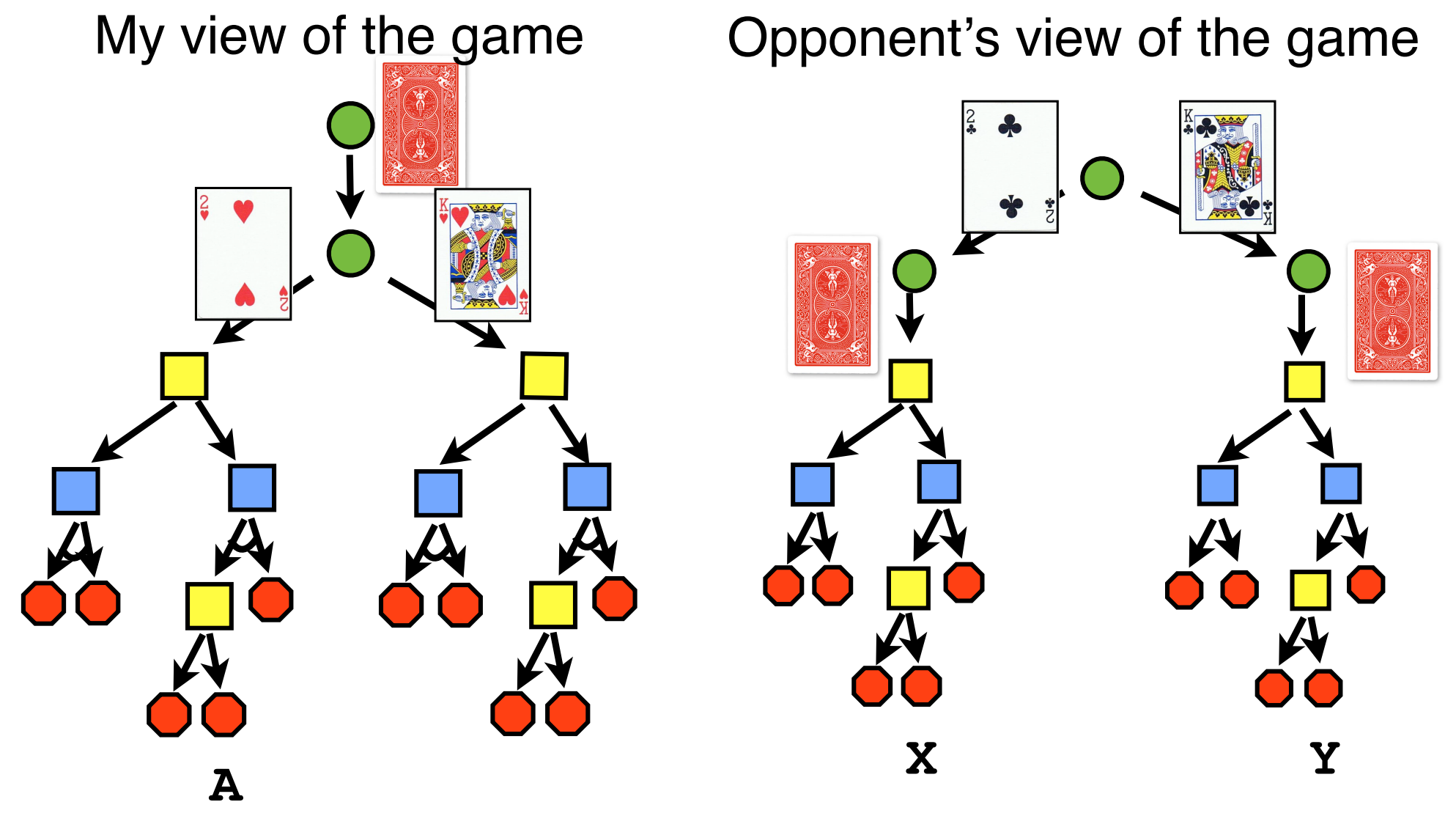
Easy to measure through competitions:
- Annual Computer Poker Competition (ACPC)
- Man-vs-Machine matches

Same approach as TAC, RoboCup, General Game Playing

Fundamental metric for game theory:
- Distance from Nash equilibrium
- Tells us how lossy the abstraction step is

Unfortunately, intractable to compute with conventional approaches!

Expectimax Search



PASS:	REACH:	X: 0.9	Y: 0.25
RETURN:	VALUE:	A: $0.25	

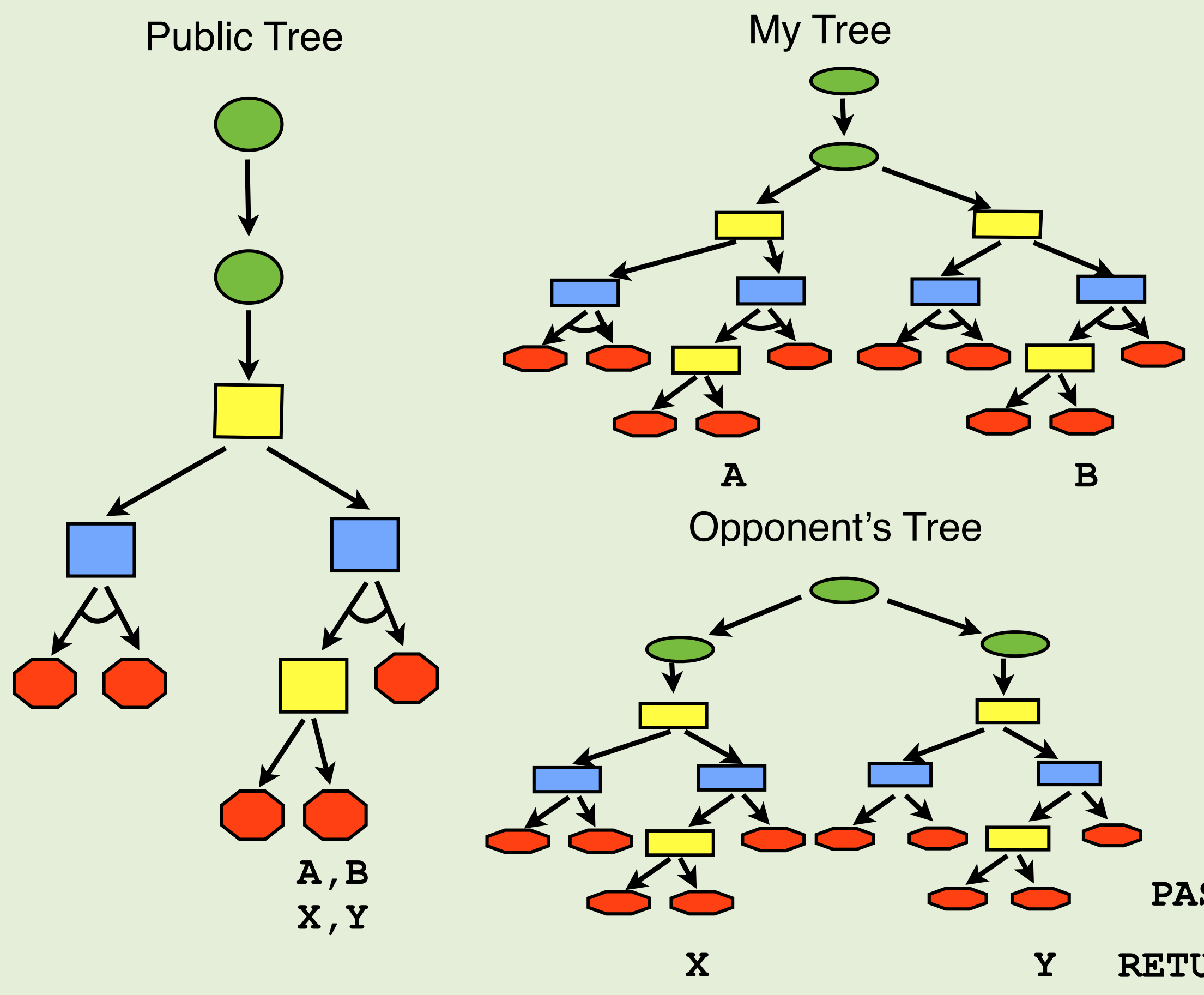
To determine our payoff at A, we need to know the distribution over the opponent being in X and Y.

Recursive tree walk algorithm:
PASS FORWARDS:
An array of probabilities of the opponent being in each of their information sets (X and Y)
RETURN:
Our value at our information set, given the opponent distribution.

Only visits each game state once. But in big domains (10^{18} in our game) this is intractable.

Four steps for accelerating best response computation in imperfect information games

1: Walking the Public Tree



We don't know if the opponent is at X or Y, but they don't know if we are at A or B.

We can reuse the work to compute the distribution over X and Y, when evaluating A and B!

Walk the much smaller 'Public Tree':
- Far fewer nodes than our tree
- More work per node
- **110x speedup in practice.**

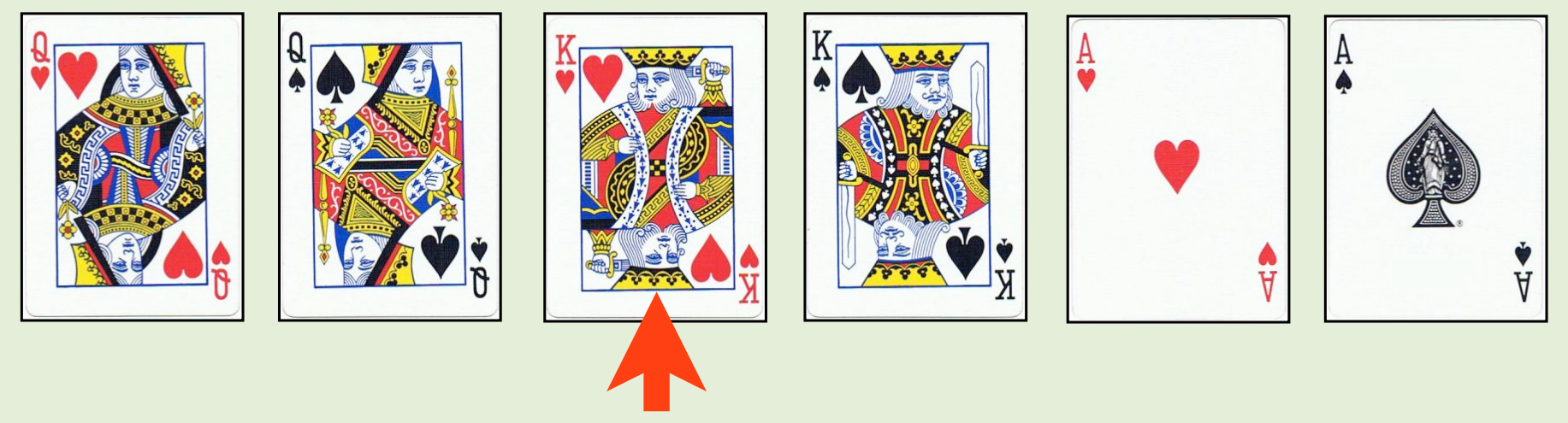
New algorithm:
PASS: Vector of reach probabilities, for their private states
RETURN: Vector of values, for our private states

PASS:	REACH:	X: 0.9	Y: 0.25
RETURN:	VALUE:	A: $0.25	B: $0.75

2: Fast Terminal Node Evaluation

At terminal nodes, we have the opponent's distribution over N states, and we want to evaluate our N states.

There may be structure to exploit.



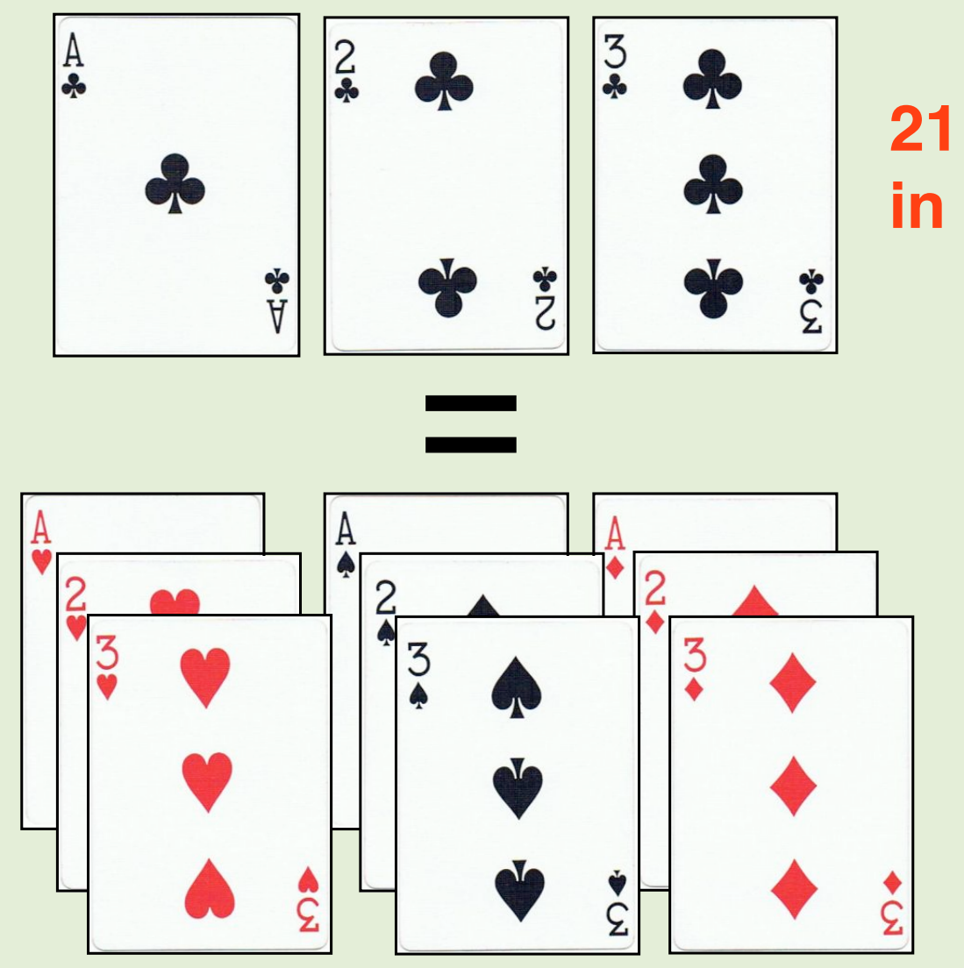
Obvious O(n²) algorithm:

```
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]
// 7.7x speedup in practice
```

3: Avoid isomorphic states



21.5x reduction in game tree.

4: Parallelize the computation

There are 1755 canonical flop deals, and 7 nonterminal betting sequences to reach them.

7 x 1755 x 2 = 24,570 subtrees to solve. We can solve these independently, at 4m30s per subtree.

4.5 * 24,570 = 76 cpu-days.

By using 72 cpus for a **72x speedup**, we can now solve this formerly intractable problem in just over one day!

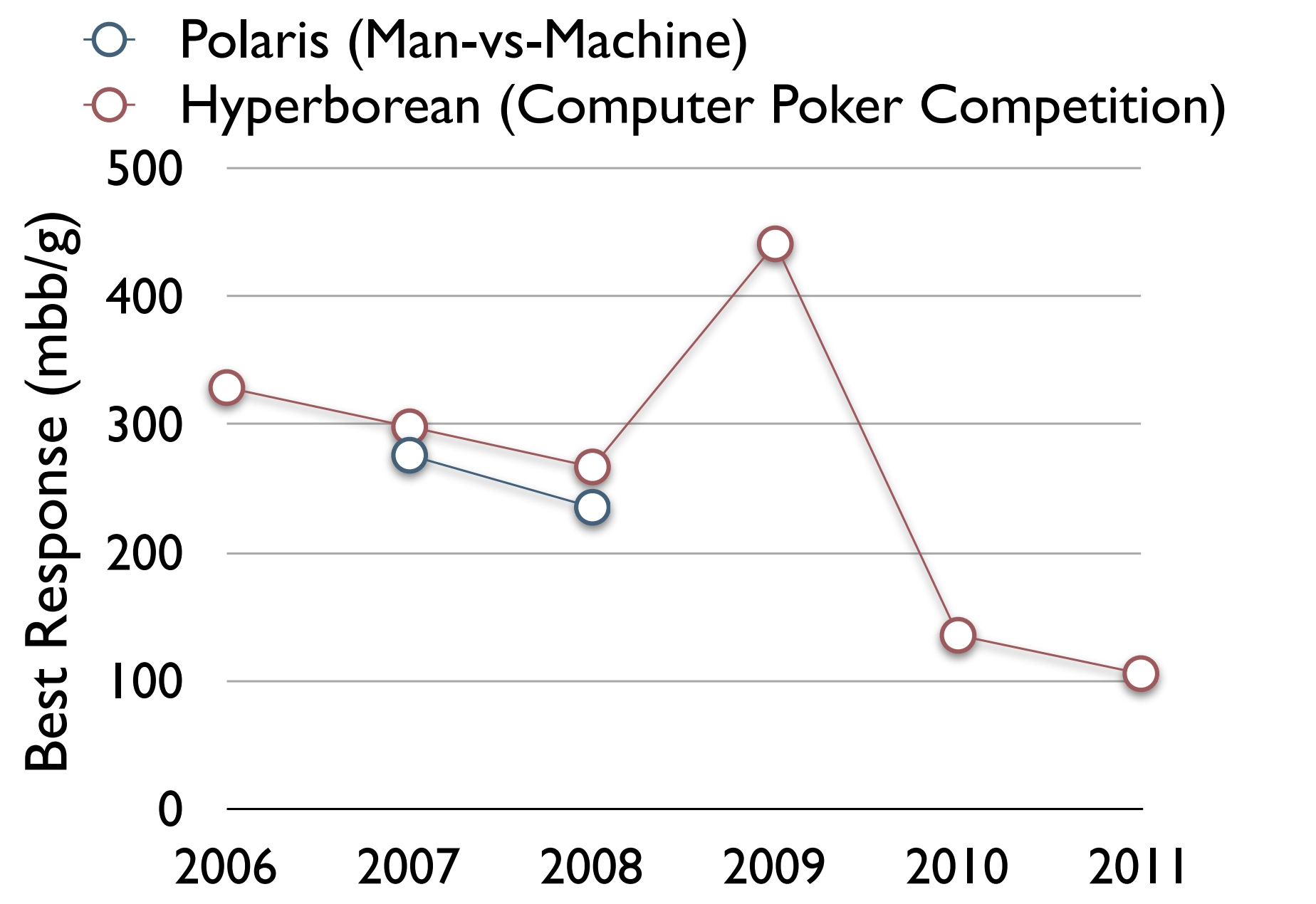
Results in 2-player Limit Texas Hold'em

Trivial opponents

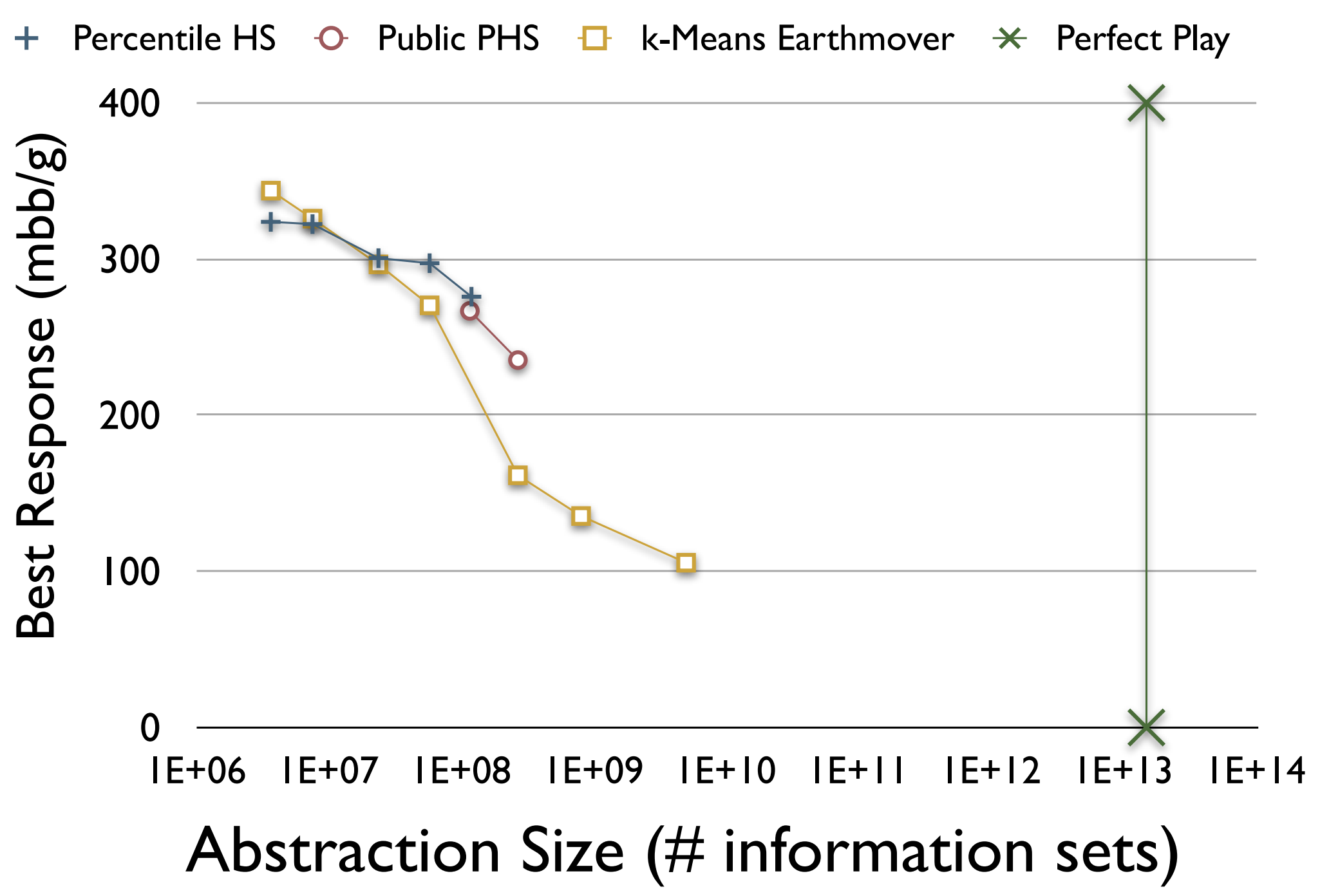
	Best Response (Milliblinds per game)
Always-Fold	750
Always-Call	1163.48
Always-Raise	3697.69
Uniform Random	3466.32

A human pro aims to win at least 50 mbb/g.

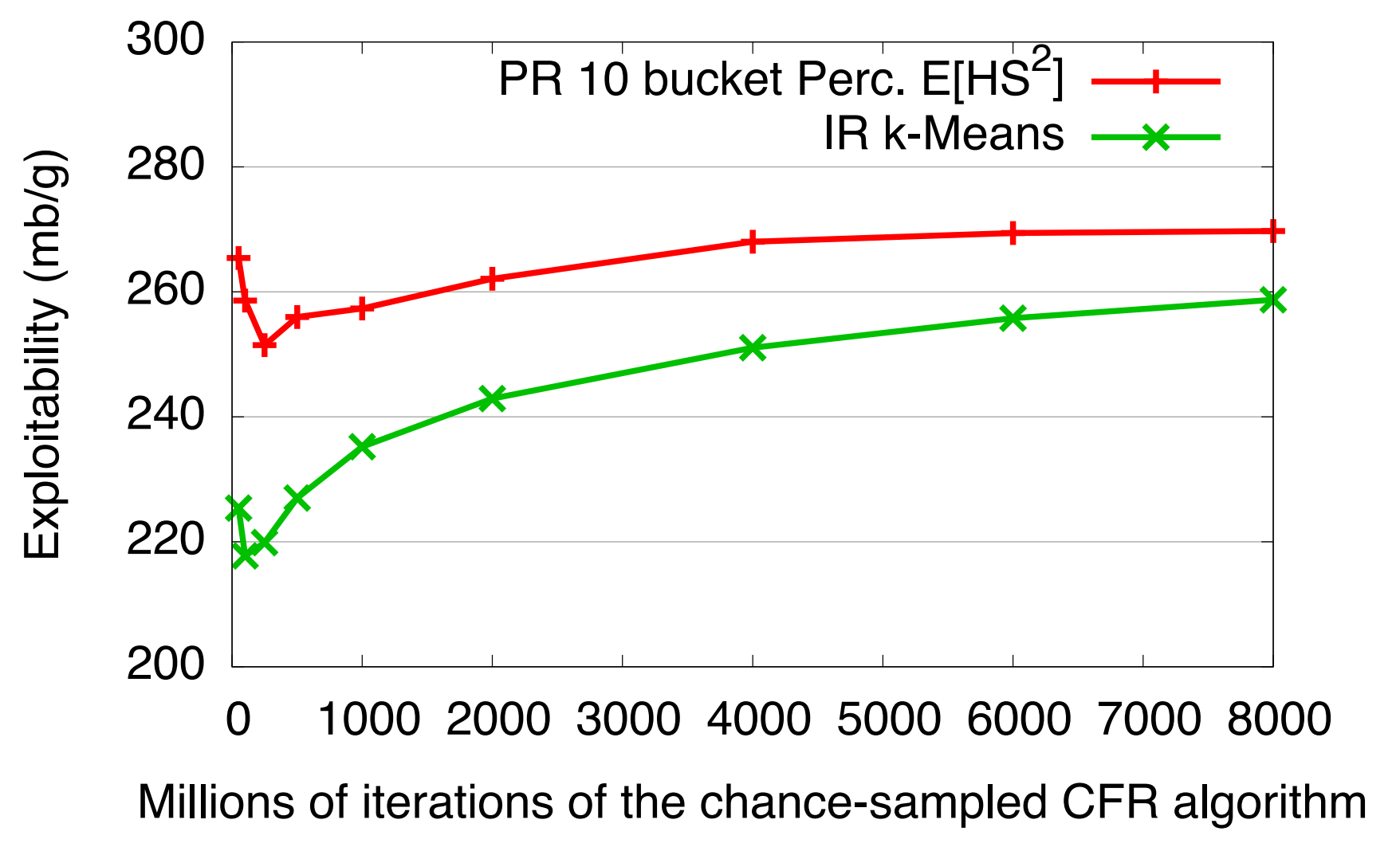
Progress over time



Abstraction and Exploitability



Overfitting



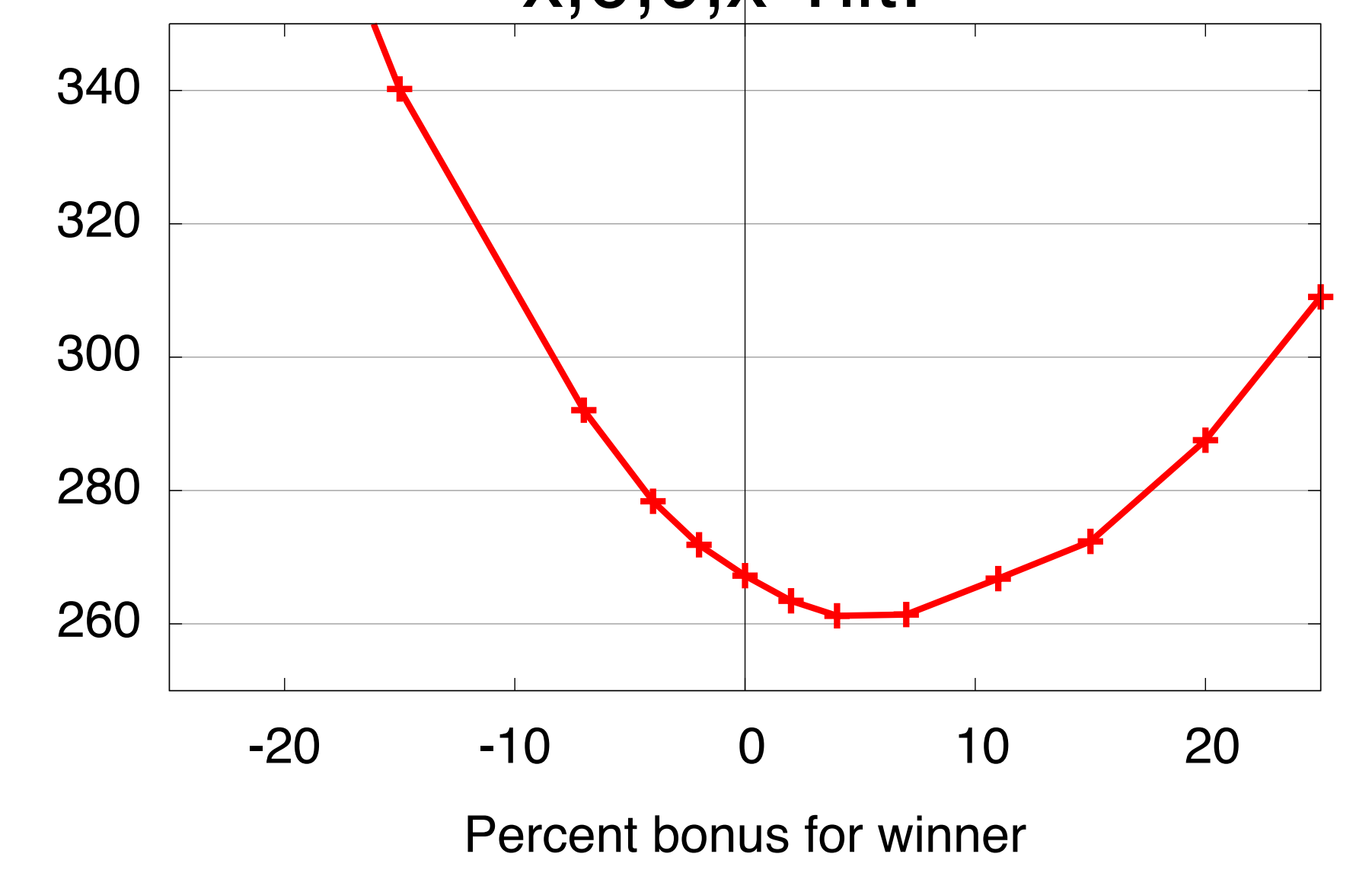
2010 Computer Poker Competition

	Rock hopper	GGValuta	HyperB (UofA)	PULPO	GS6 (CMU)	Littlerock	Best Response
Rock hopper		6	3	7	37	77	300
GGValuta	-6		3	1	31	77	237
HyperB (UofA)	-3	-3		2	31	70	135
PULPO	-7	-1	-2		32	125	399
GS6 (CMU)	-37	-31	-31	-32		47	318
Littlerock	-77	-77	-70	-125	-47		421

Tilting with Polaris 2008

Name	Payoff Adjustment (My win, His Win, My Fold, His Fold)	Best Response
Pink	0,0,0,0	235.294
Orange	7,0,0,7	227.457
Peach	0,0,0,7	228.325
*Red	0,-7,0,0	257.231
*Green	0,-7,0,-7	263.702
*Equilibrium	0,0,0,0	266.797

x,0,0,x Tilt:



UNIVERSITY OF ALBERTA
Alberta Innovates
compute • calcul CANADA
WestGrid
NSERC CRSNG
Making IT happen Computing Science