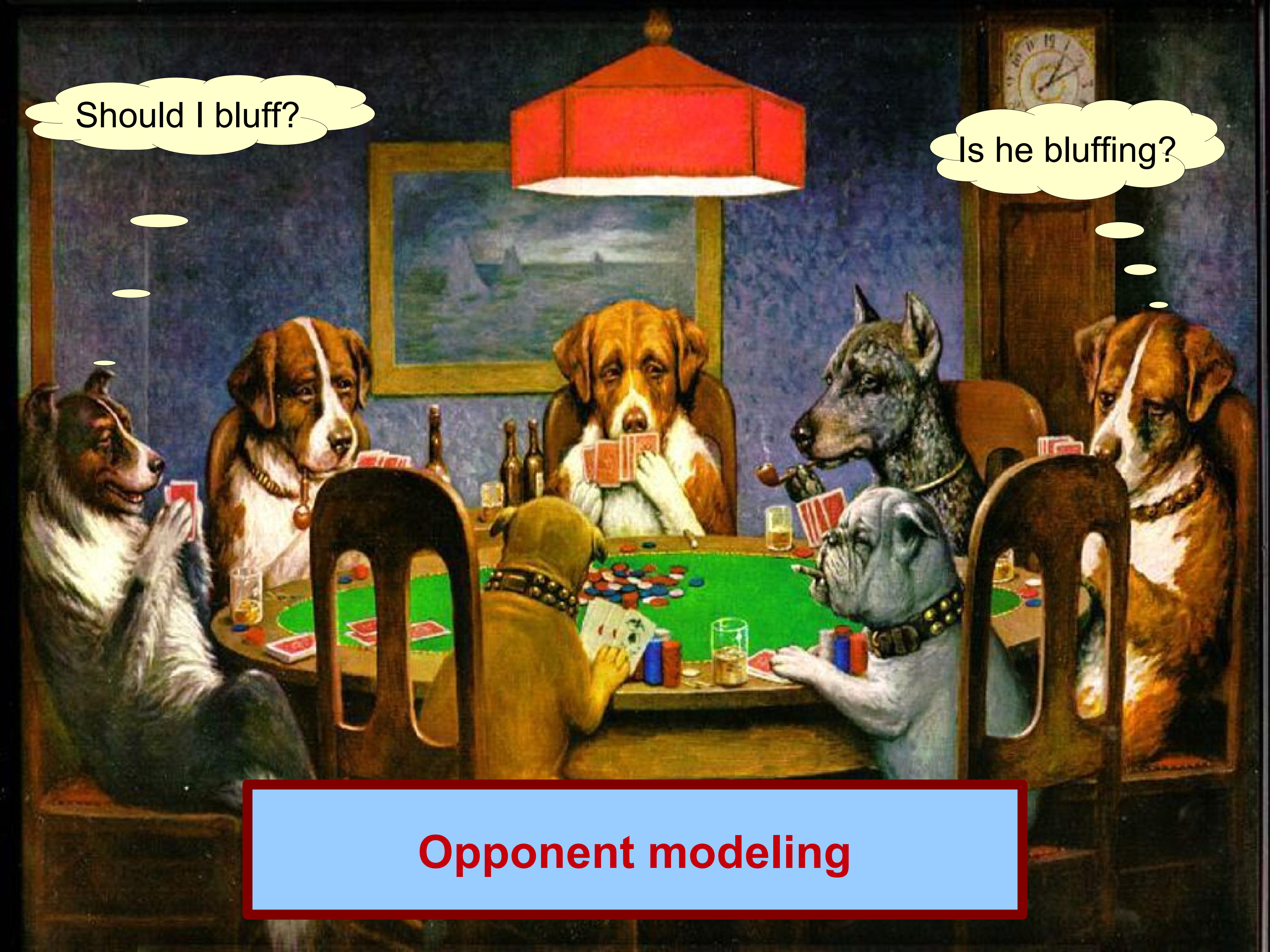
Monte-Carlo tree search for multi-player, no-limit Texas hold'em poker

Guy Van den Broeck

KATHOLIEKE UNIVERSITEIT LEUVEN

DTAI
MACHINE LEARNING


Should I bluff?
Deceptive play

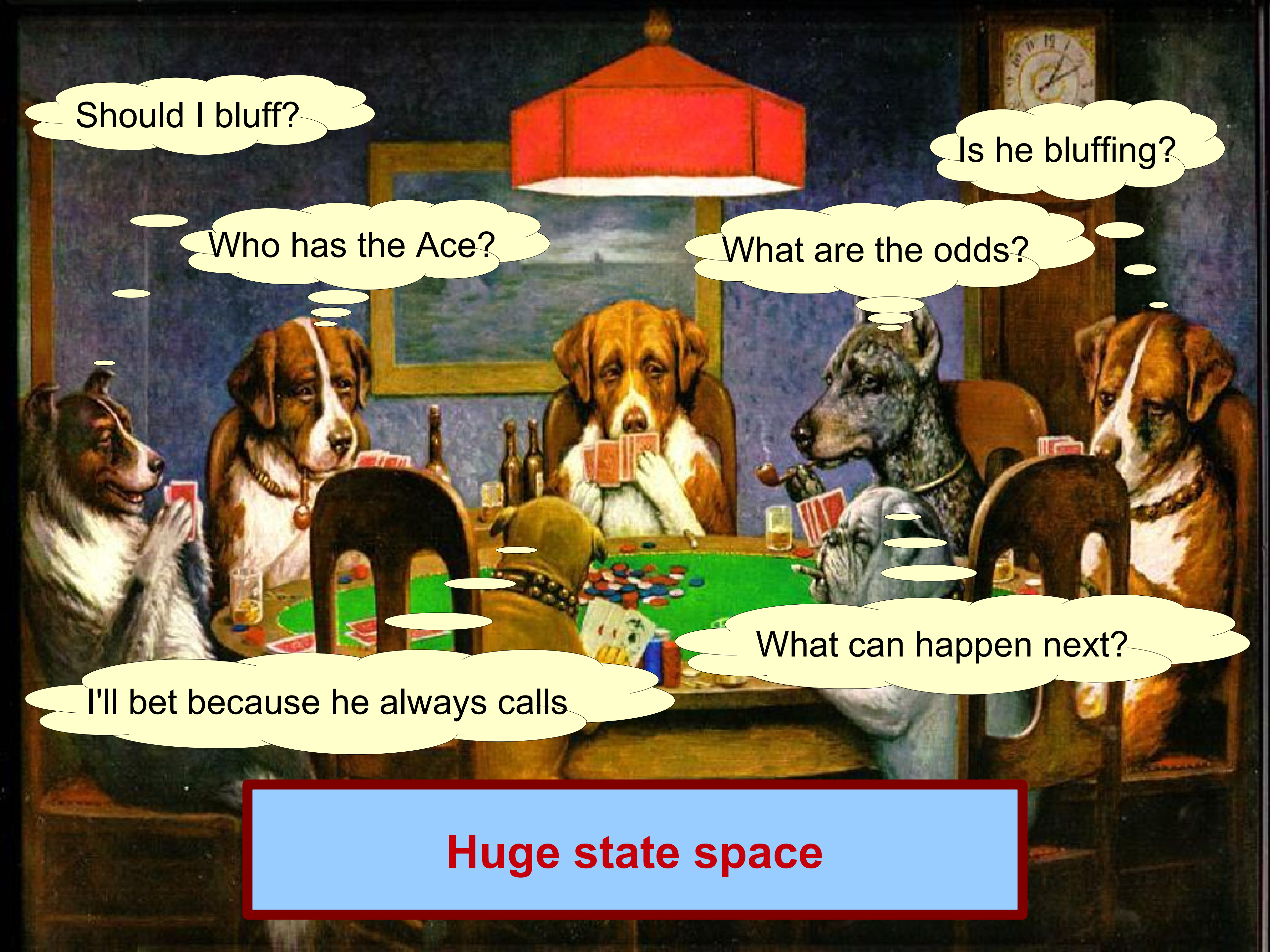


Opponent modeling

Incomplete information

Game of chance

Exploitation


Should I bluff?
Is he bluffing?
Who has the Ace?
What are the odds?
What can happen next?
I'll bet because he always calls
Huge state space

Risk management & Continuous action space

Take-Away Message:

We can solve all these problems!

Problem Statement

- A bot for Texas hold'em poker
  - No-Limit & > 2 players
    - Not done before!
  - Exploitative, not game theoretic
    - Game tree search + Opponent modeling
- Applies to any problem with either
  - incomplete information
  - non-determinism
  - continuous actions

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Outline

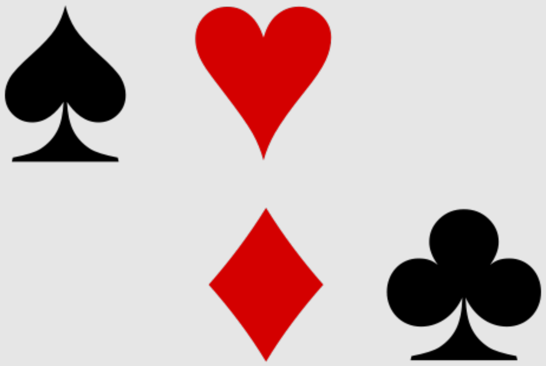
- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Poker Game Tree

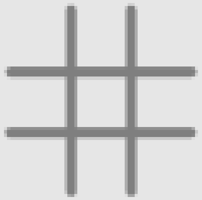
- Minimax trees: deterministic
  - Tic-tac-toe, checkers, chess, go,…

max *min*

Poker Game Tree



- Minimax trees: deterministic
  - Tic-tac-toe, checkers, chess, go,…

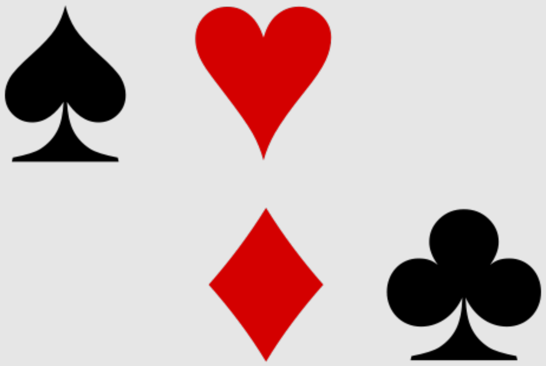


max *min*

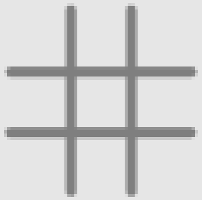
- Expecti(mini)max trees: chance
  - Backgammon, …

max *min* *mix*

Poker Game Tree



- Minimax trees: deterministic
  - Tic-tac-toe, checkers, chess, go,…



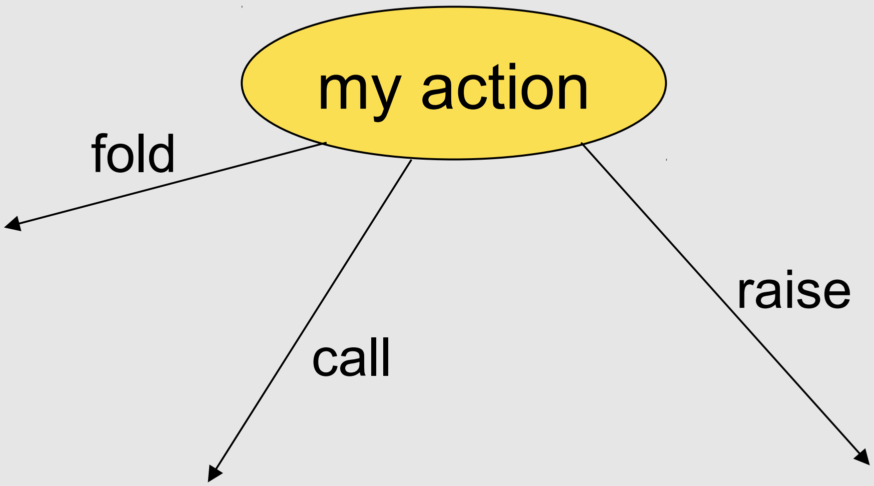
max min

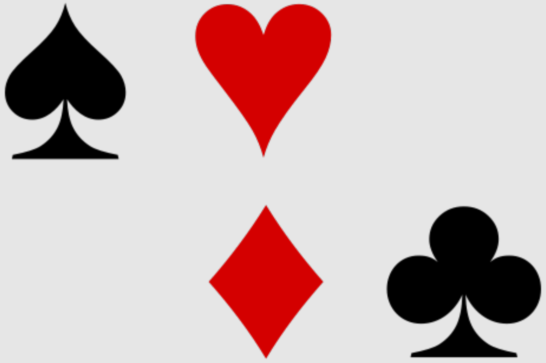
- Expecti(mini)max trees: chance
  - Backgammon, …

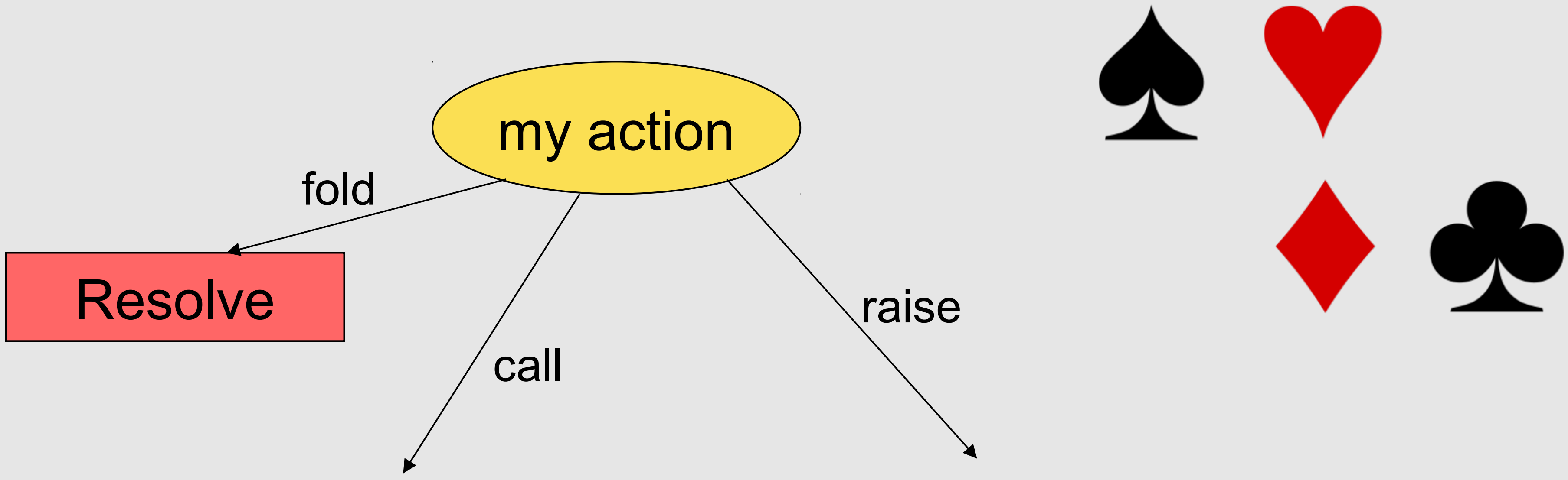
max min mix

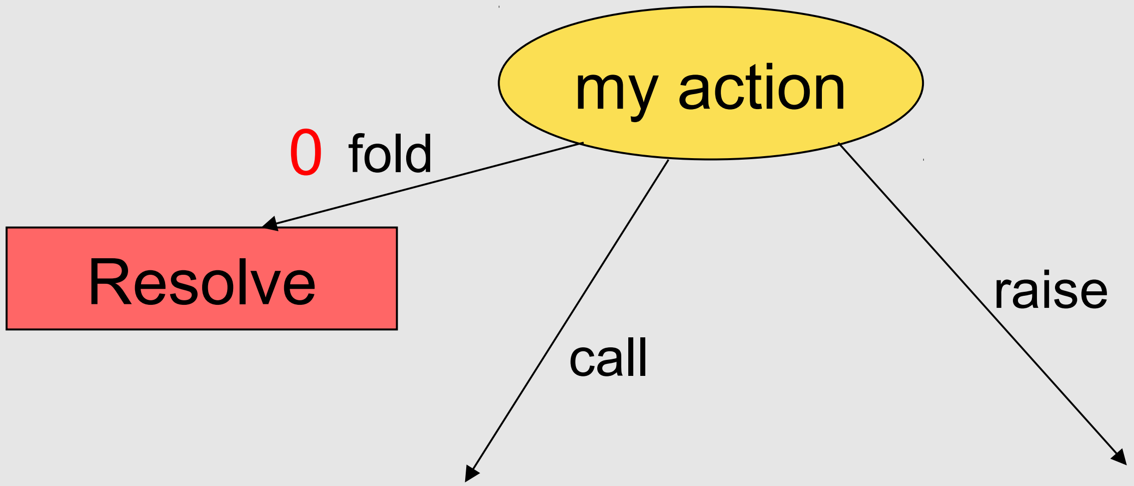
- Miximax trees: hidden information

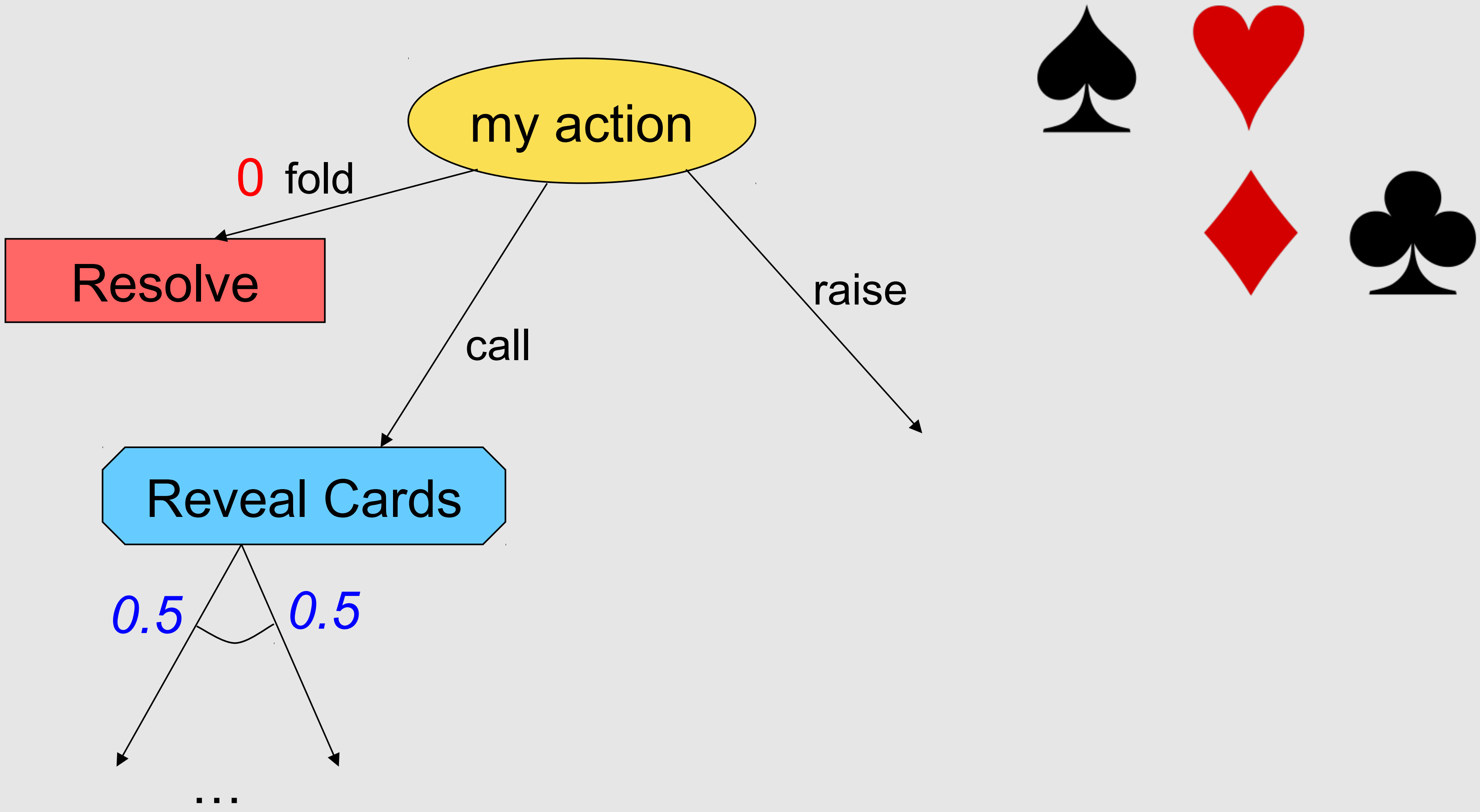
max mix mix + opponent model

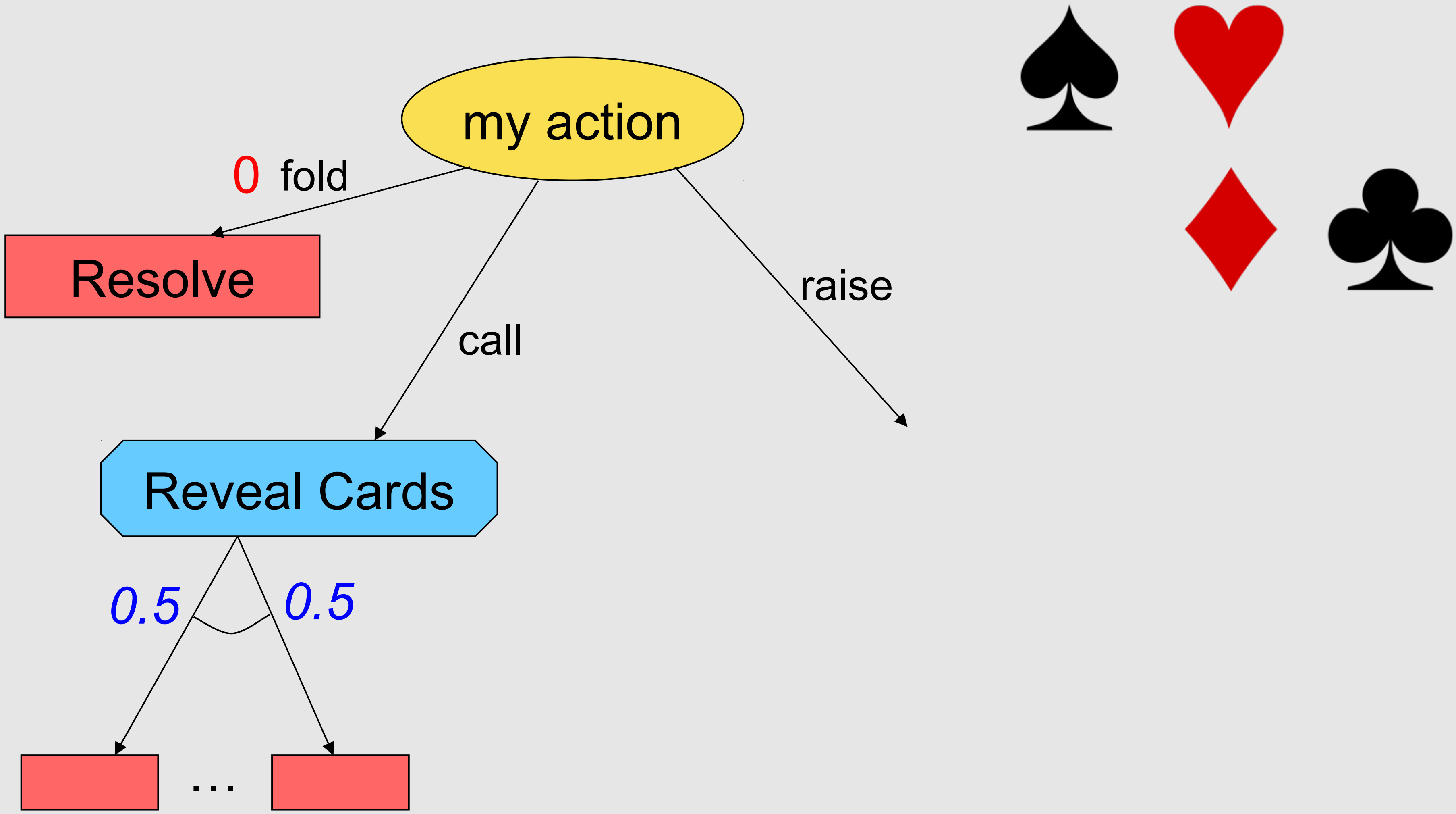

my action
fold
call
raise

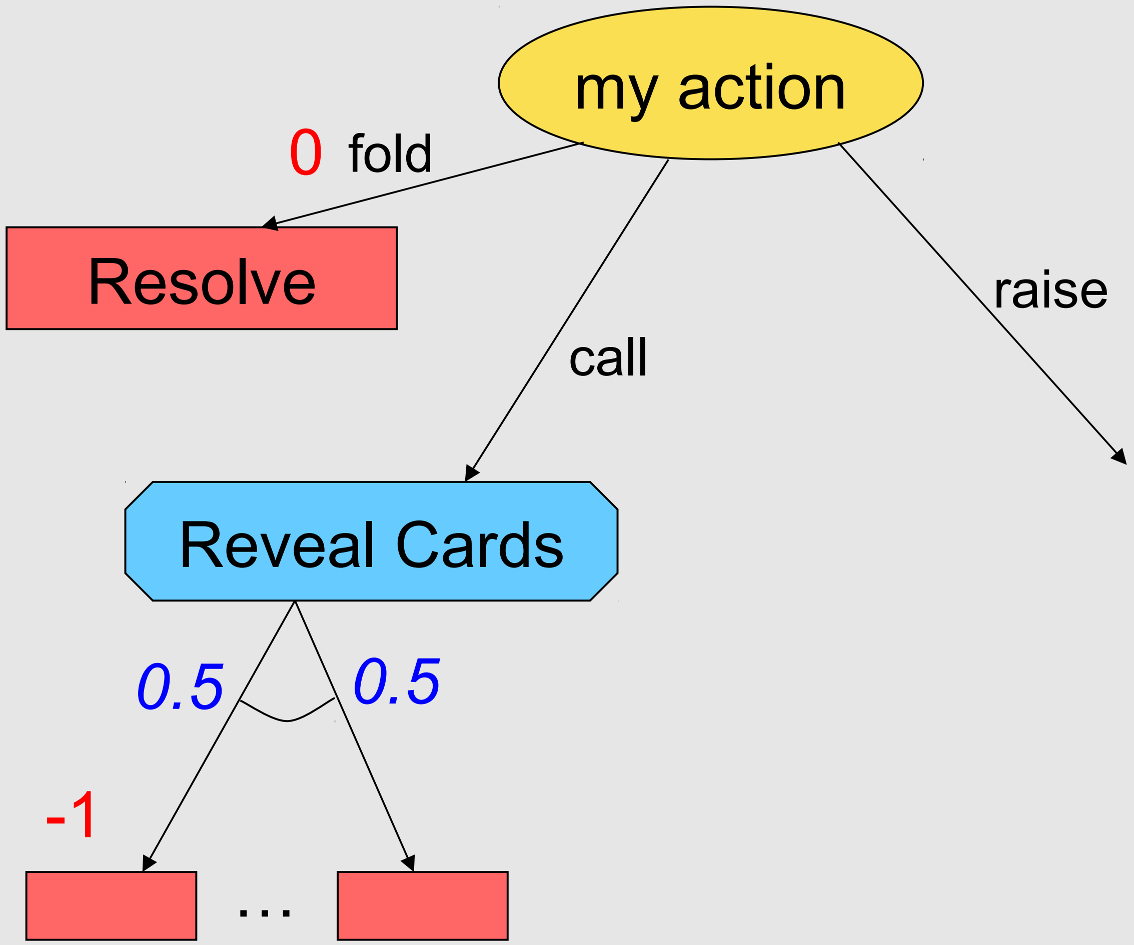


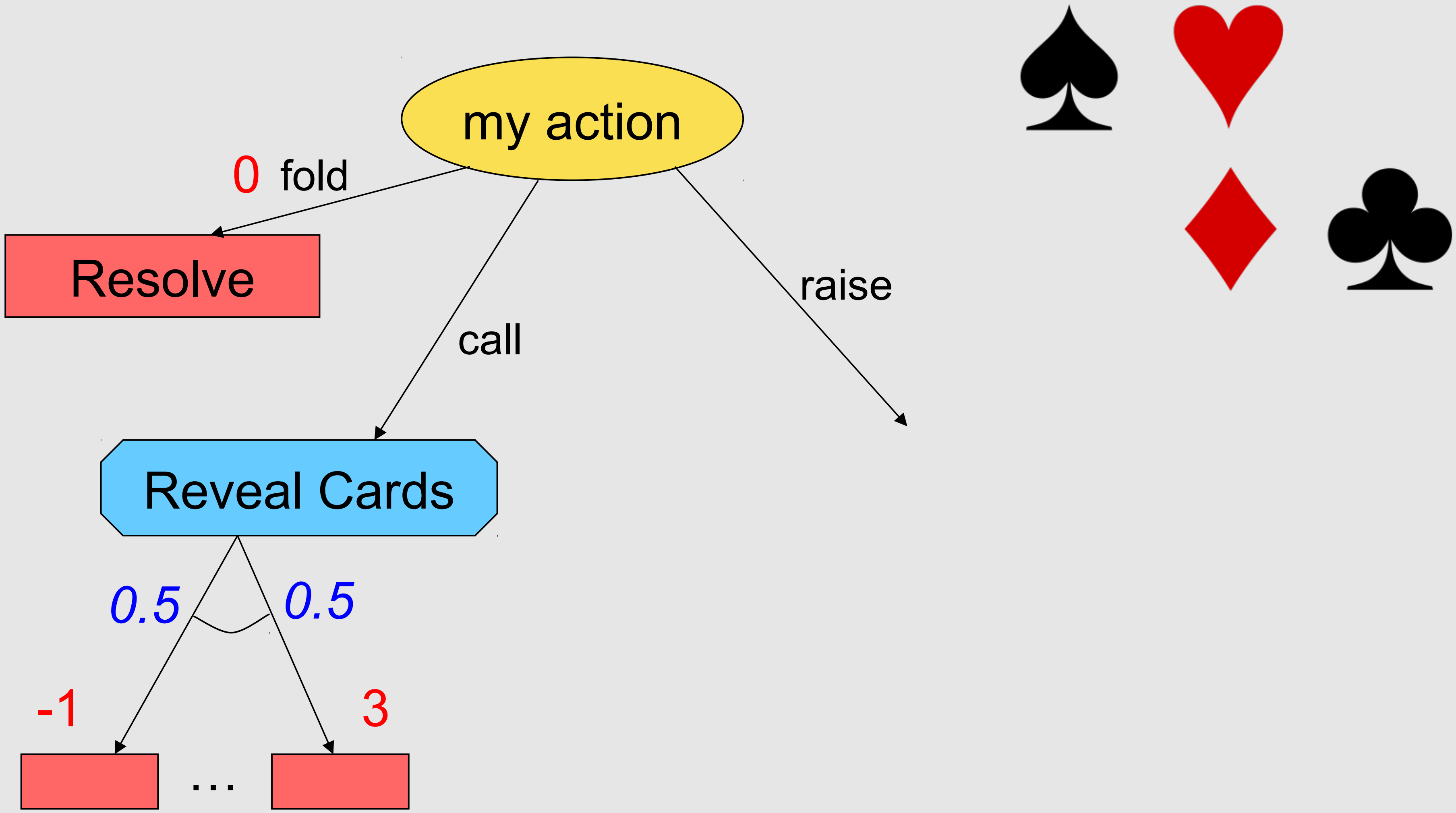

my action
fold
Resolve
call
raise

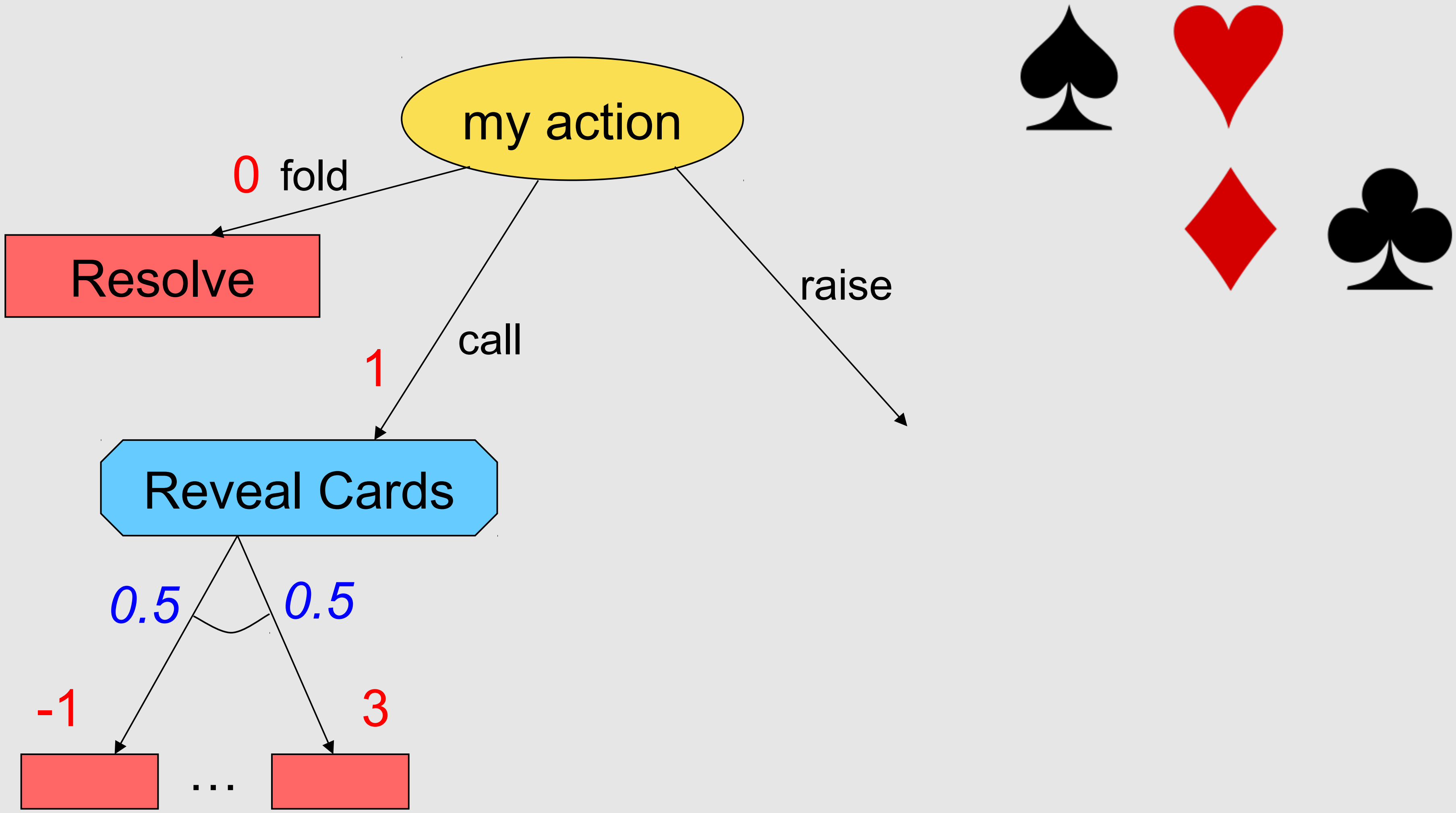

my action
0 fold
Resolve
call
raise


my action
0
fold
Resolve
call
raise
Reveal Cards
0.5
0.5
...

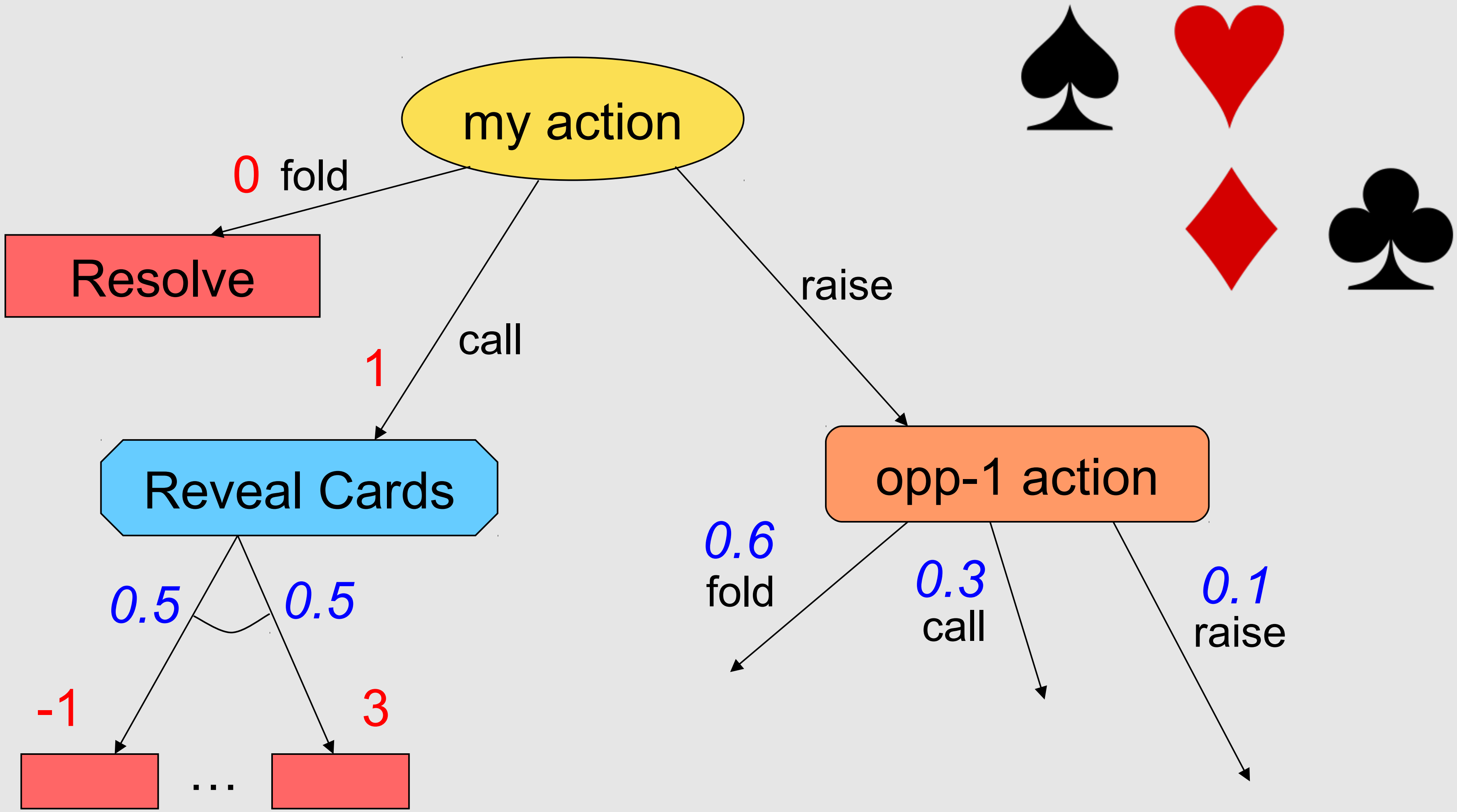

my action
0
fold
Resolve
call
raise
Reveal Cards
0.5
0.5
...

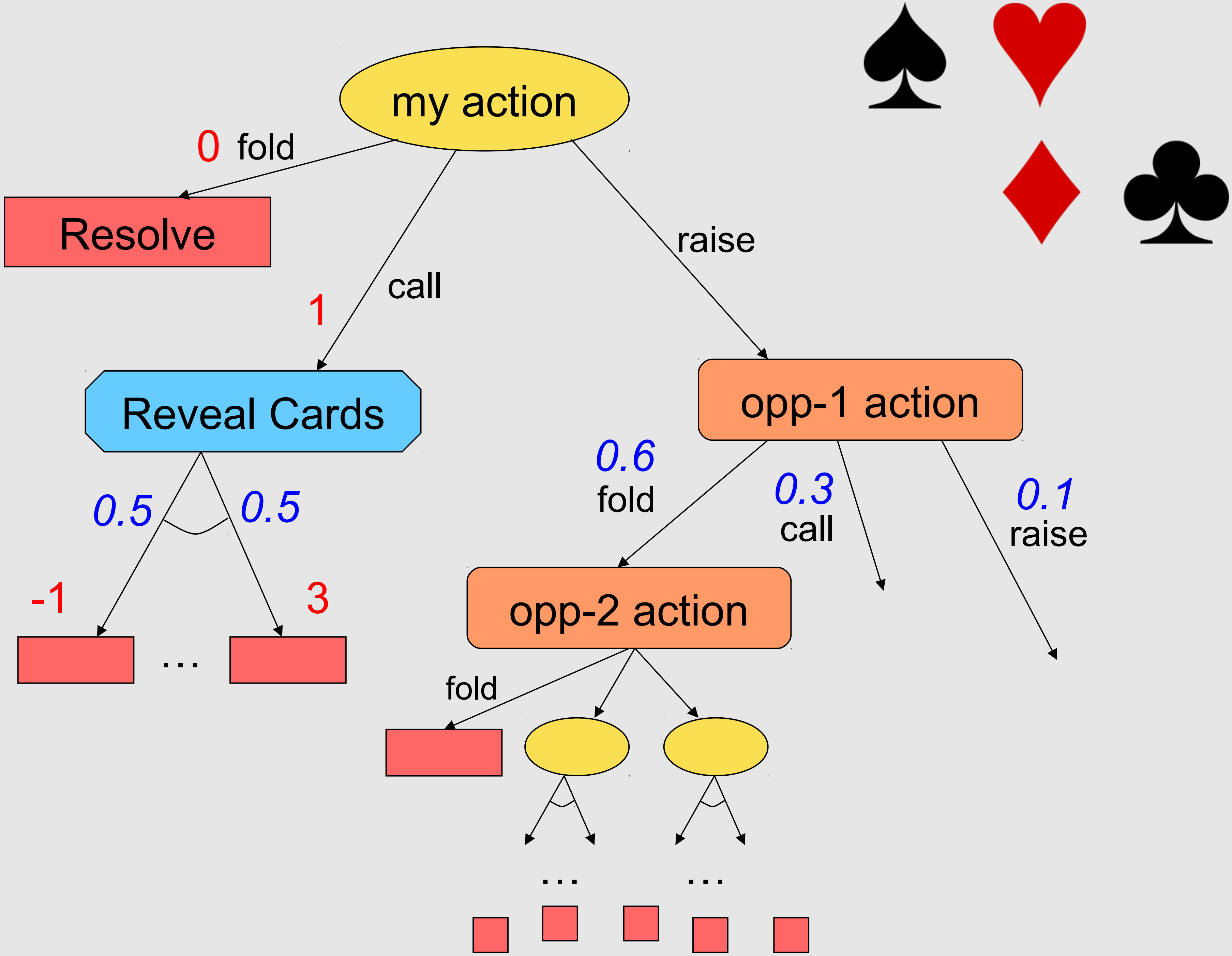

my action
0
fold
Resolve
call
raise
Reveal Cards
0.5
0.5
-1
...

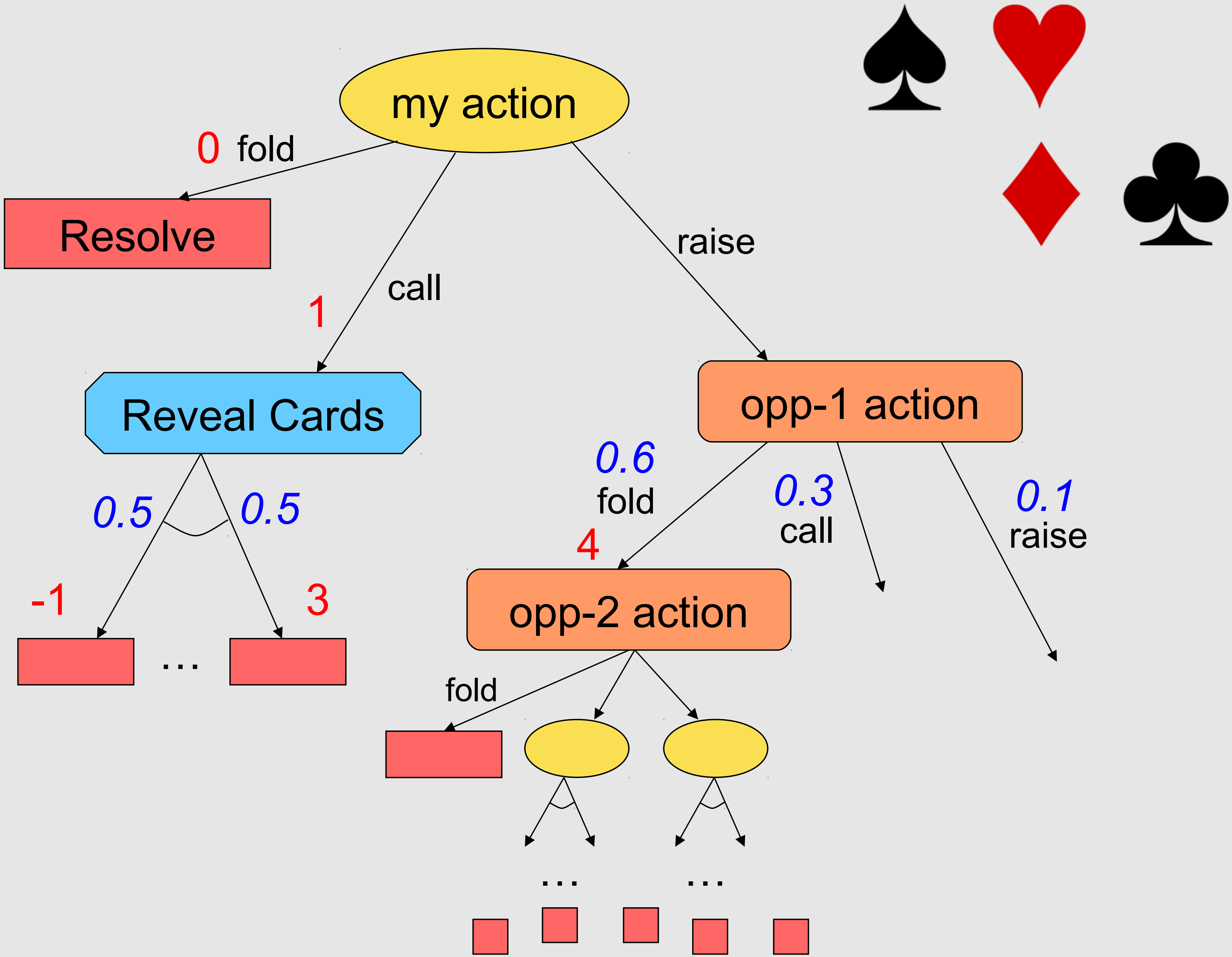

my action
0
fold
Resolve
call
raise
Reveal Cards
0.5
0.5
-1
3
...



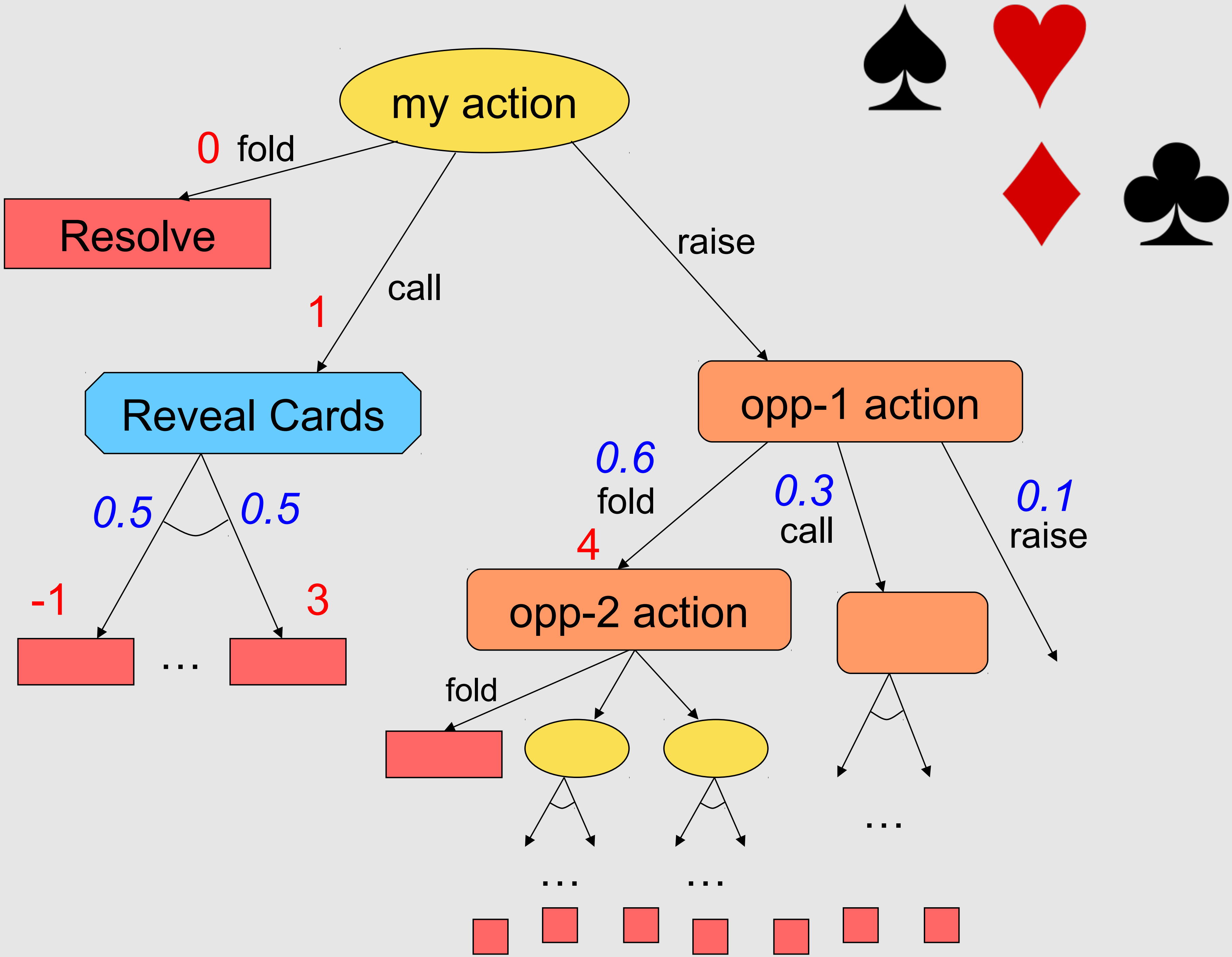
my action
0
fold
Resolve
call
1
raise
Reveal Cards
0.5
0.5
-1
3
…


my action
0 fold
Resolve
1
call
raise
Reveal Cards
opp-1 action
0.6
fold
0.3
call
0.1
raise
0.5
0.5
-1
3
...

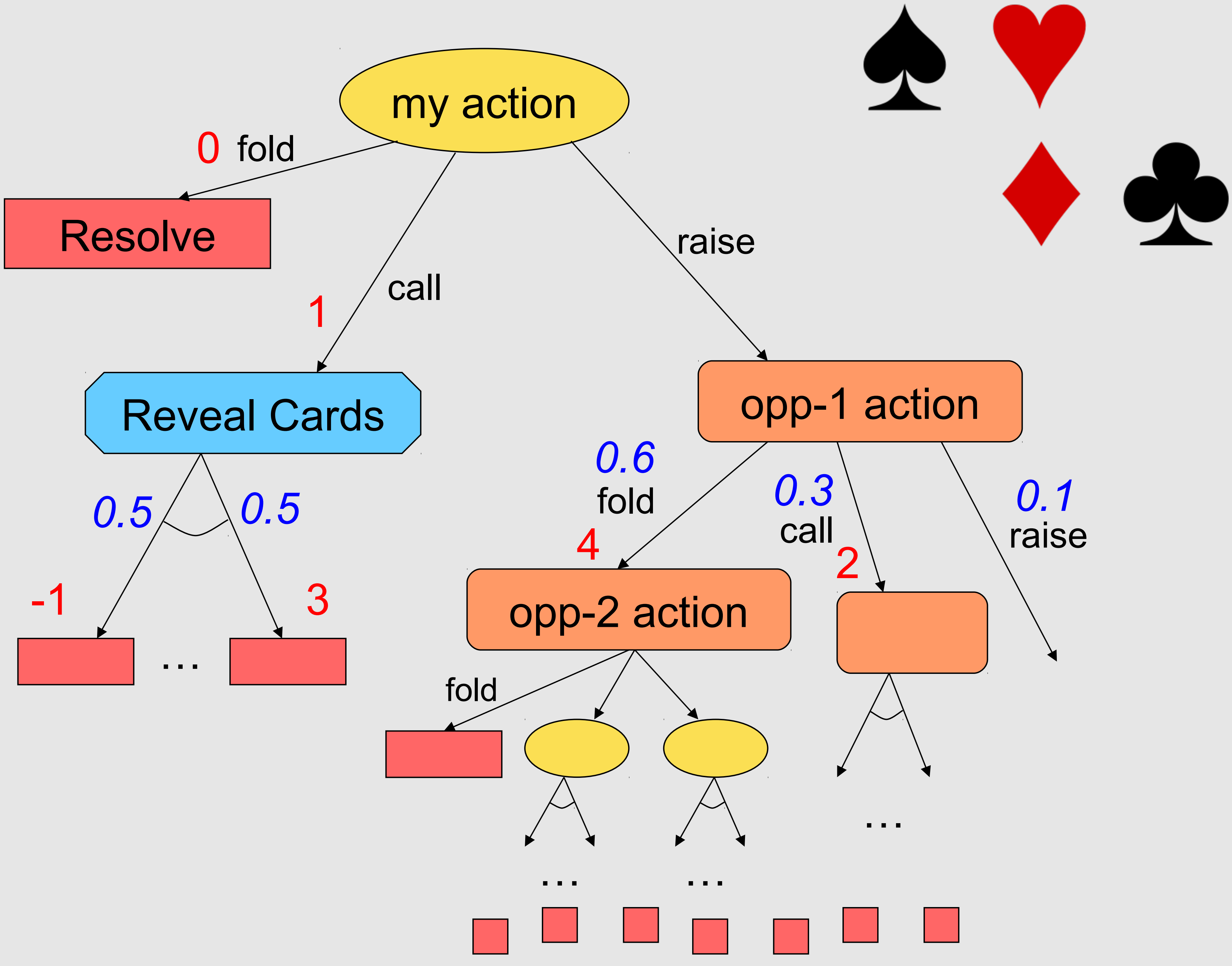

my action
0 fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.5
0.5
-1
3
...
0.6
fold
0.3
call
0.1
raise
opp-2 action
fold
...
...

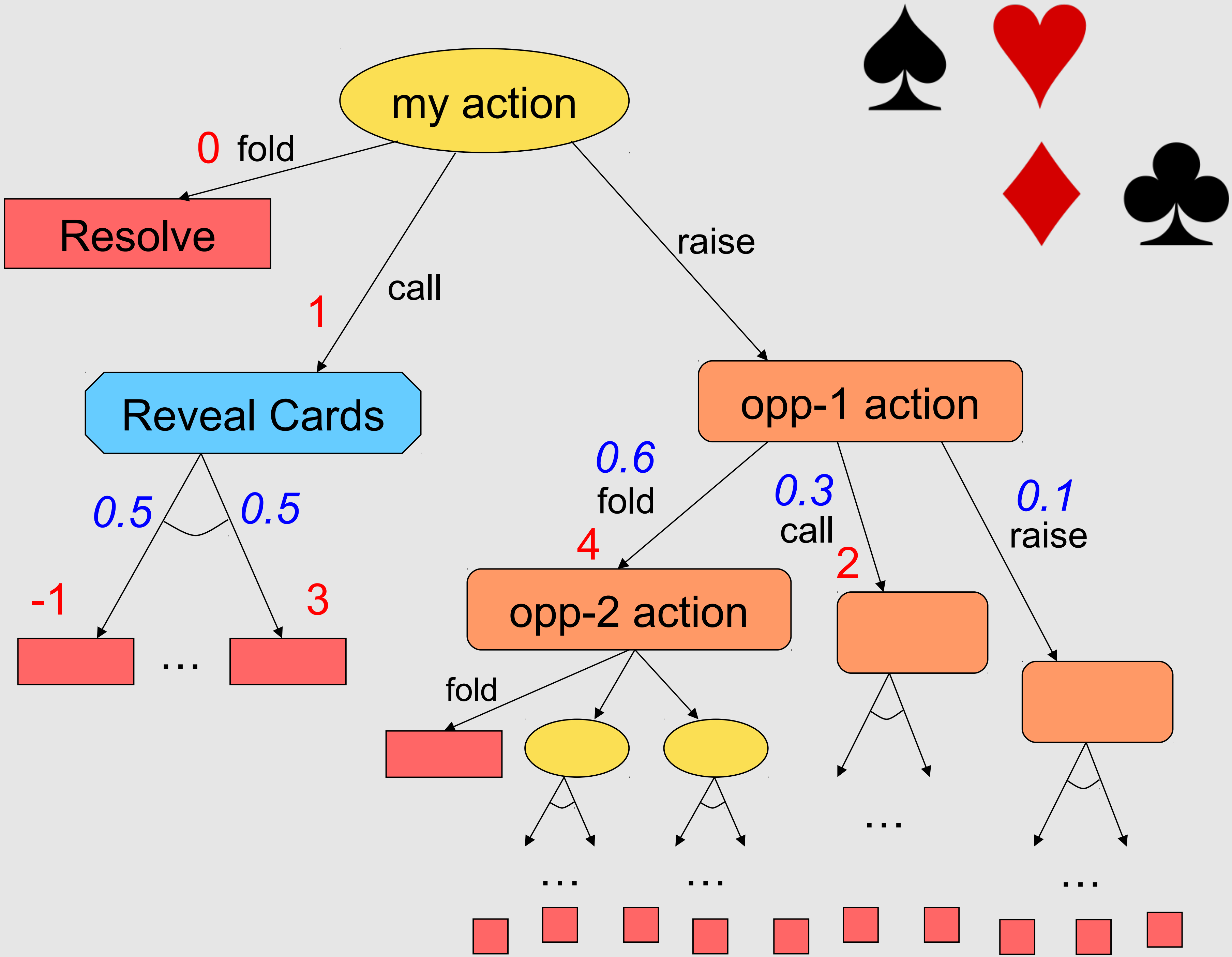


my action
0 fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.6
fold
0.3
call
0.1
raise
0.5
0.5
4
-1
3
opp-2 action
...
fold
...
...

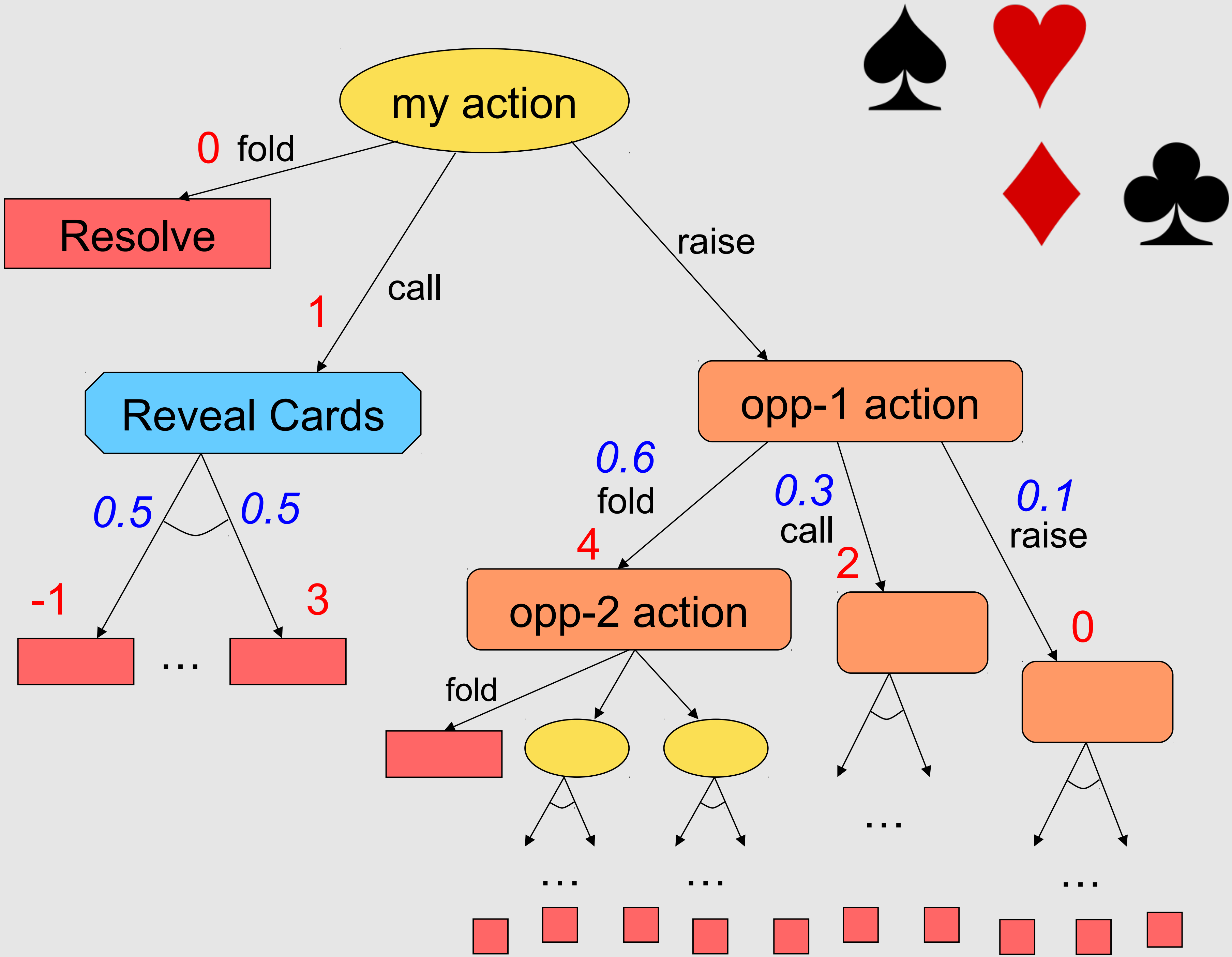


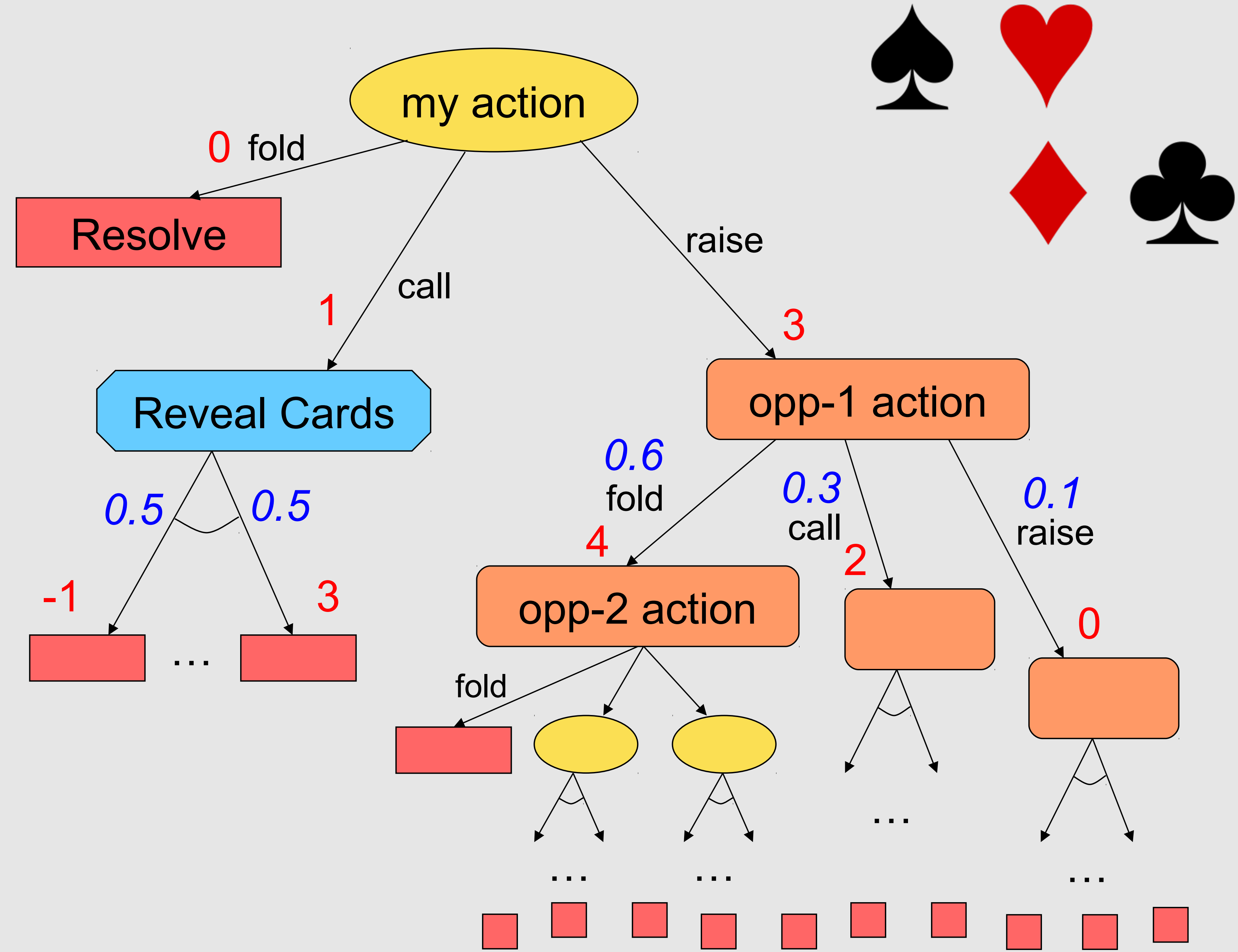
my action
0 fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.5
0.5
-1
3
...
0.6
fold
4
0.3
call
0.1
raise
opp-2 action
fold
...
...
...

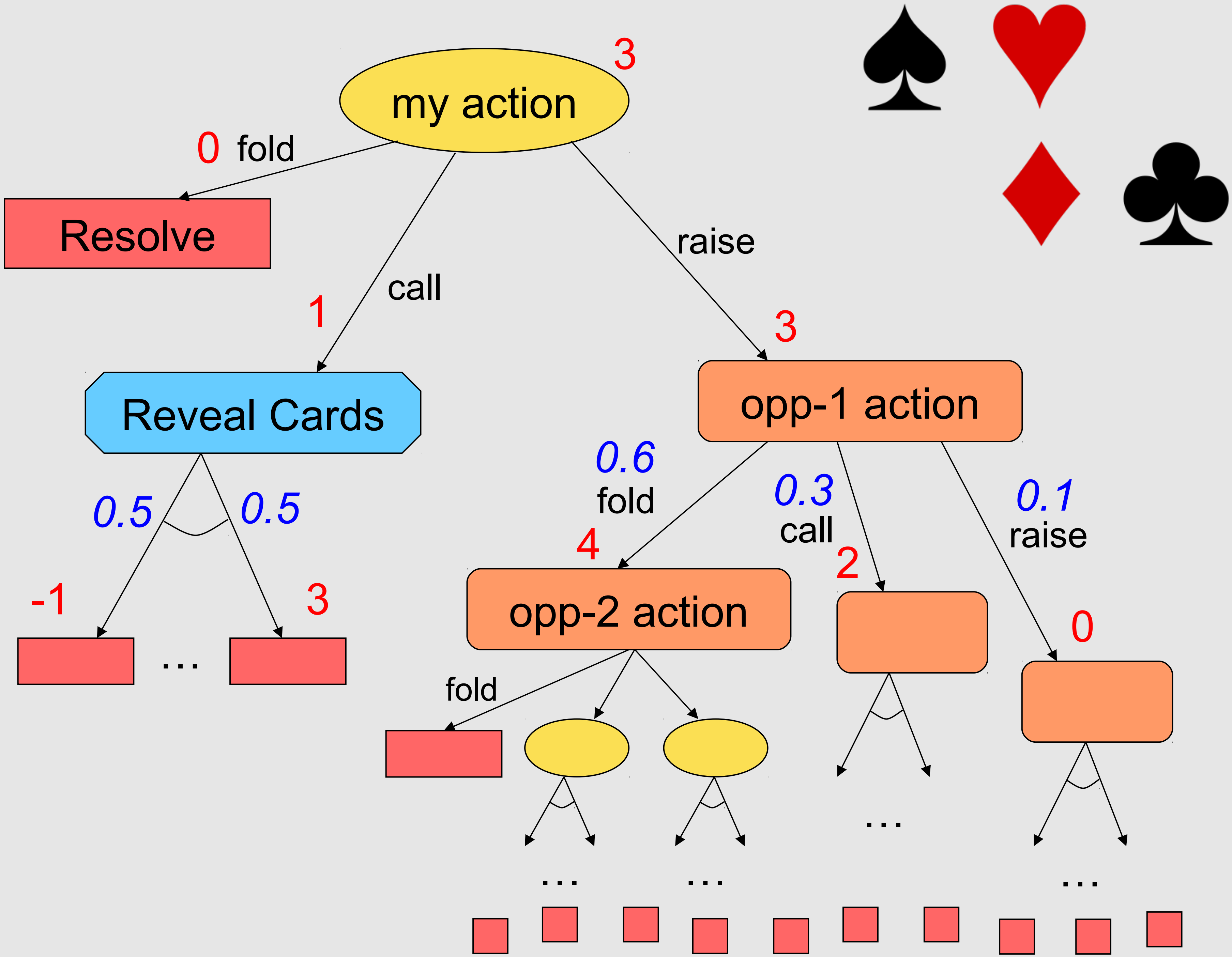

my action
0 fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.5
0.5
-1
3
...
0.6
fold
4
0.3
call
2
0.1
raise
opp-2 action
fold
...
...
...



my action
0
fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.5
0.5
-1
3
...
0.6
fold
4
0.3
call
2
0.1
raise
opp-2 action
fold
...
...
...
...


my action
0 fold
Resolve
call
1
raise
Reveal Cards
opp-1 action
0.5
0.5
-1
...
3
0.6
fold
4
0.3
call
2
0.1
raise
0
opp-2 action
fold
...
...
...
...


my action
0 fold
Resolve
call
1
raise
3
Reveal Cards
opp-1 action
0.5
0.5
-1
3
...
0.6
fold
4
0.3
call
2
0.1
raise
0
opp-2 action
fold
...
...
...
...

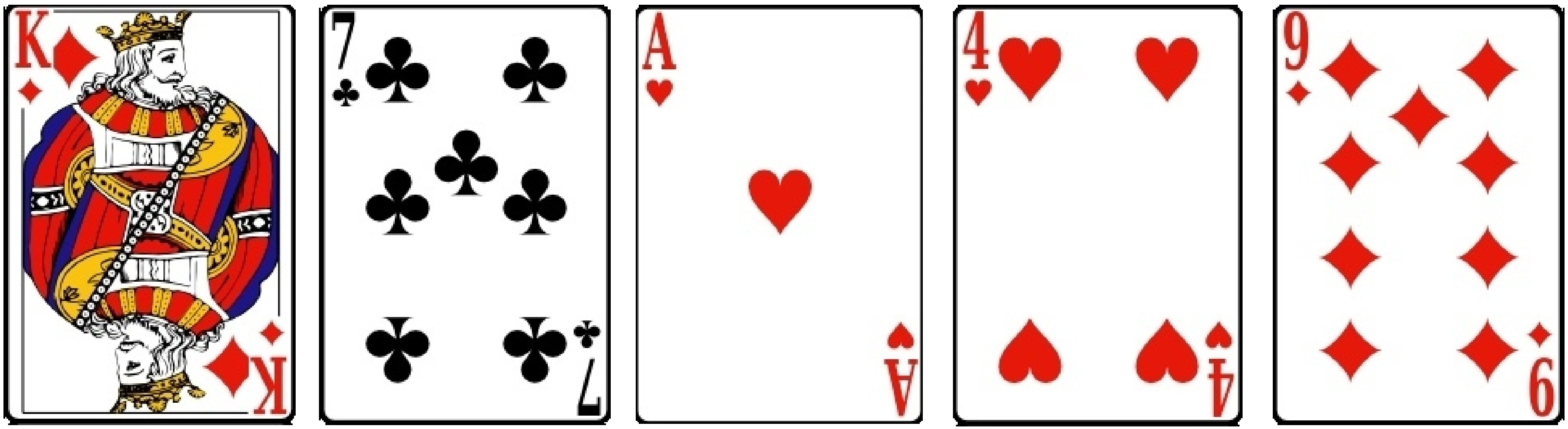


my action
3
0
fold
Resolve
call
1
raise
3
Reveal Cards
opp-1 action
0.5
0.5
-1
3
0.6
fold
4
0.3
call
2
0.1
raise
0
opp-2 action
fold

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Short Experiment



Opponent Model

- Set of probability trees
- Weka's M5'
- Separate model for
  - Actions

    $$P(A_i | A_0 \dots A_{i-1}, C_0 \dots C_i)$$

  - Hand cards at showdown

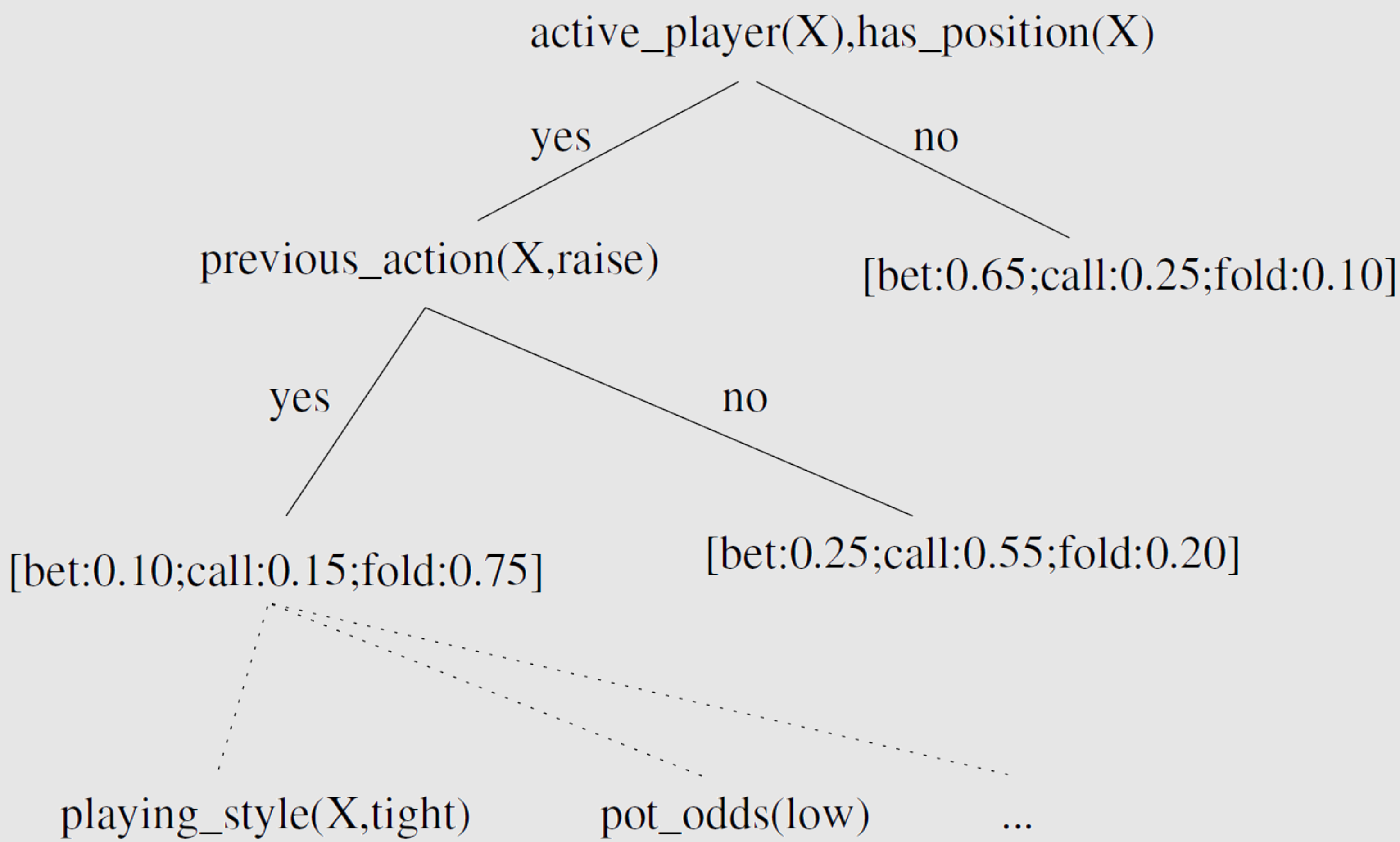
    $$P(H | A_0 \dots A_n, C_0 \dots C_n)$$

Fold Probability

```
nbAllPlayerRaises <= 1.5 :
|   callFrequency <= 0.128 :
|   |   nbActionsThisRound <= 2.5 :
|   |   |   potOdds <= 0.28 :
|   |   |   |   AF <= 2.585 : 0.6904
|   |   |   |   AF >  2.585 :
|   |   |   |   |   potSize <= 3.388 :
|   |   |   |   |   |   round=flop <= 0.5 : 0.8068
|   |   |   |   |   |   round=flop >  0.5 : 0.6896
|   |   |   |   |   potSize >  3.388 : 0.8198
|   |   |   potOdds >  0.28 :
|   |   |   |   stackSize <= 97.238 :
|   |   |   |   |   callFrequency <= 0.038 : 0.8838
|   |   |   |   |   callFrequency >  0.038 :
|   |   |   |   |   |   round=flop <= 0.5 : 0.8316
|   |   |   |   |   |   round=flop >  0.5 :
|   |   |   |   |   |   |   nbSeatedPlayers <= 7.5 : 0.6614
|   |   |   |   |   |   |   nbSeatedPlayers >  7.5 : 0.7793
|   |   |   |   stackSize >  97.238 :
|   |   |   |   |   potSize <= 4.125 :
|   |   |   |   |   |   foldFrequency <= 0.813 : 0.7839
|   |   |   |   |   |   foldFrequency >  0.813 : 0.9037
|   |   |   |   |   potSize >  4.125 : 0.8623
|   |   nbActionsThisRound >  2.5 :
|   |   |   potOdds <= 0.218 :
|   |   |   |   callFrequency <= 0.067 : 0.8753
|   |   |   |   callFrequency >  0.067 : 0.7661
|   |   |   potOdds >  0.218 :
|   |   |   |   AF <= 2.654 : 0.8818
|   |   |   |   AF >  2.654 : 0.921
```

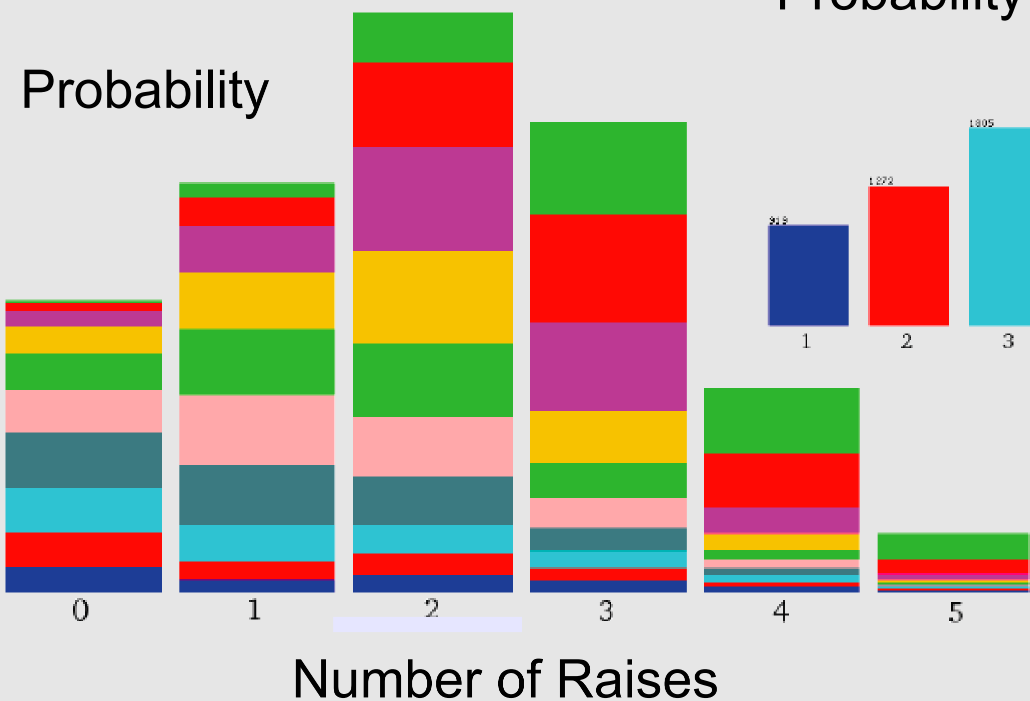
(Can also be relational)

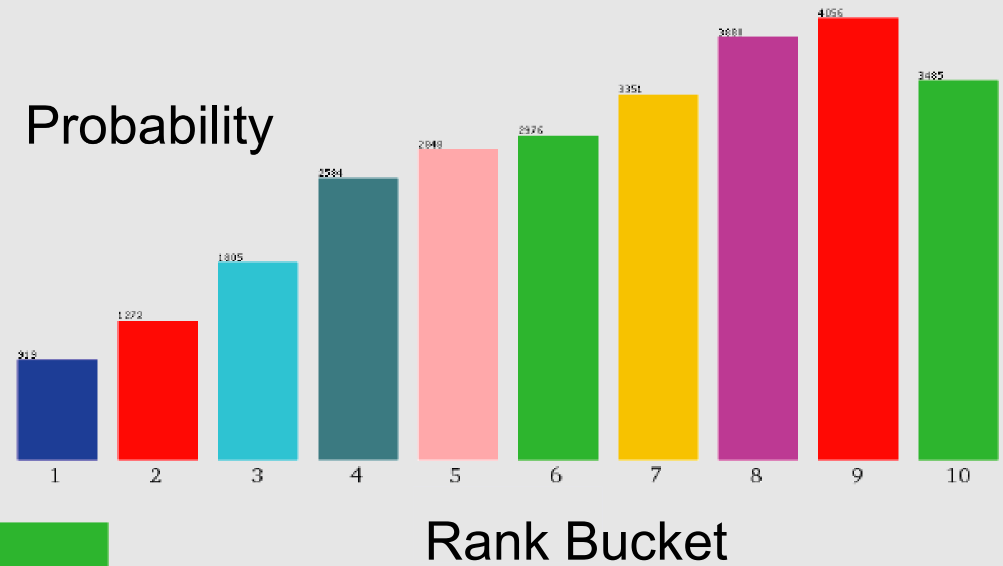
- Tilde probability tree [Ponsen08]



Opponent Ranks

- Learn distribution of hand ranks at showdown





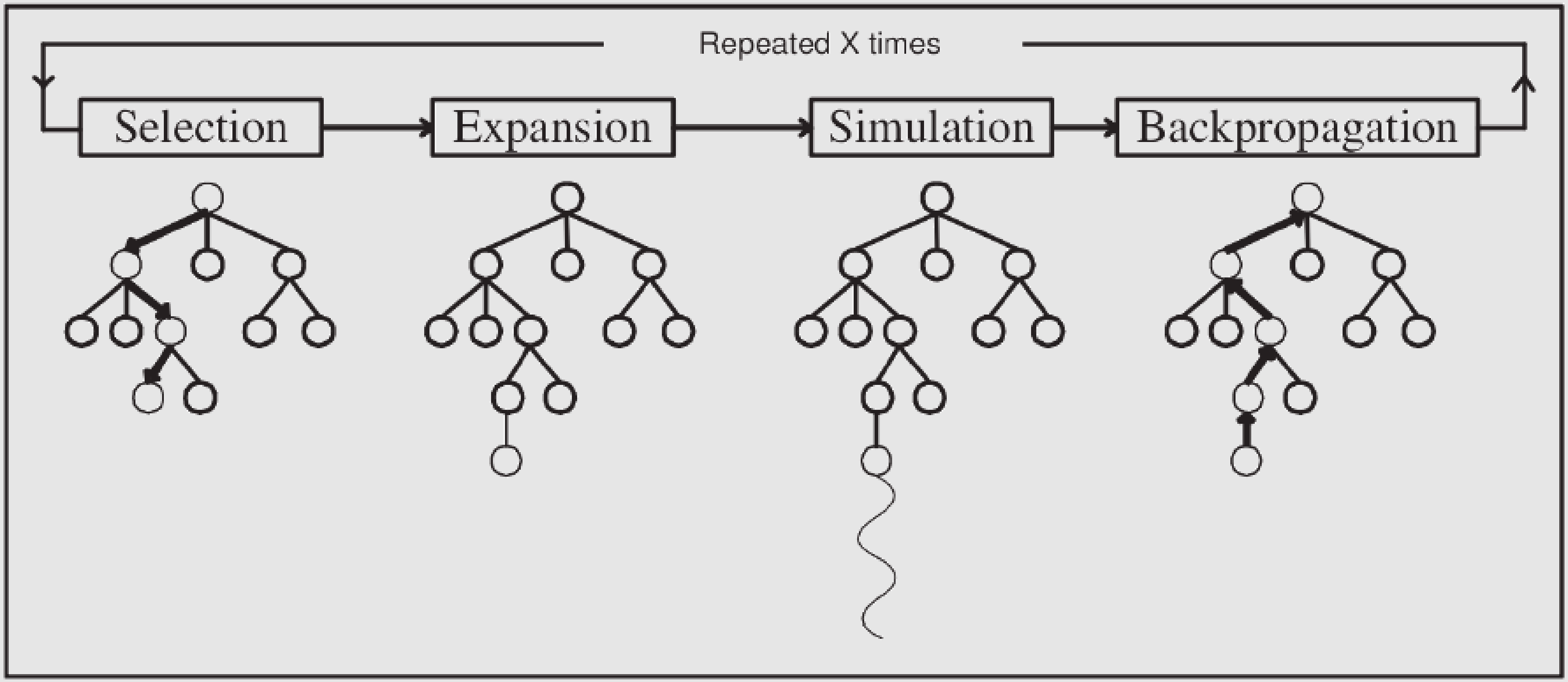
Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Traversing the tree

- Limit Texas Hold'em
  - 10^{18} nodes
  - Fully traversable
- No-limit
  - $>10^{71}$ nodes
  - Too large to traverse
  - Sampled, not searched
  - Monte-Carlo Tree Search

Monte-Carlo Tree Search



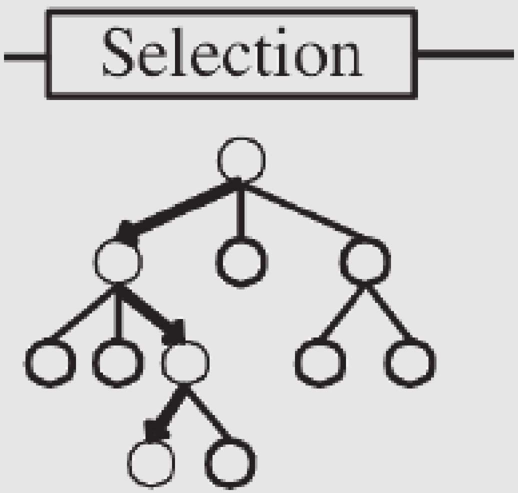
[Chaslot08]

Selection

In each node:

$\hat{V}(P)$ is an estimate of the reward $r(P)$

$\mathrm{T}(P)$ is the number of samples

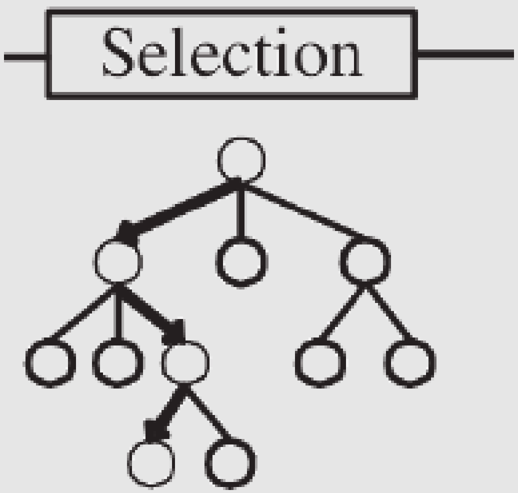


Selection

In each node:

$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples

- UCT (Multi-Armed Bandit)



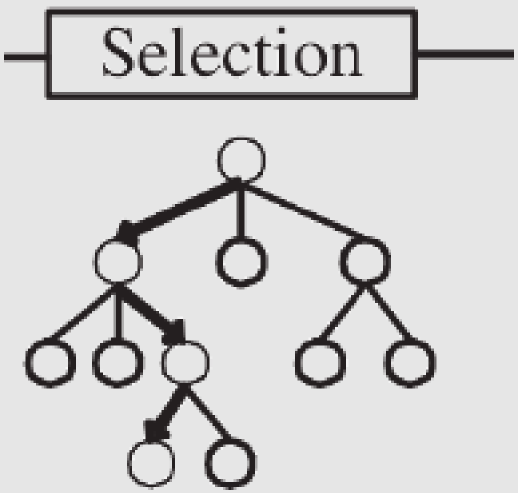
$$\hat{V}(c_i) + C\sqrt{\frac{\ln \mathrm{T}(P)}{\mathrm{T}(c_i)}}$$

Selection

In each node:

$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples

- UCT (Multi-Armed Bandit)



exploitation

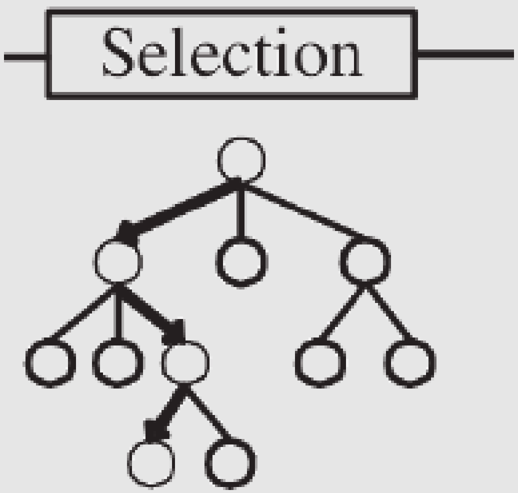
$$\boxed{\hat{V}(c_i)} + C\sqrt{\frac{\ln \mathrm{T}(P)}{\mathrm{T}(c_i)}}$$

Selection

In each node:

$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples

- UCT (Multi-Armed Bandit)



exploitation

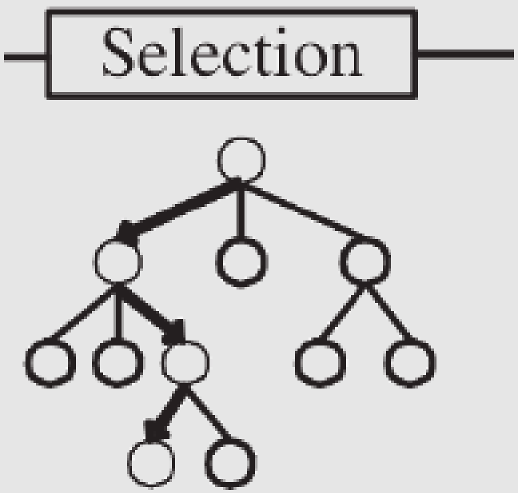
$$\boxed{\hat{V}(c_i)} + \boxed{C\sqrt{\frac{\ln \mathrm{T}(P)}{\mathrm{T}(c_i)}}}$$

exploration

Selection

In each node:

$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples



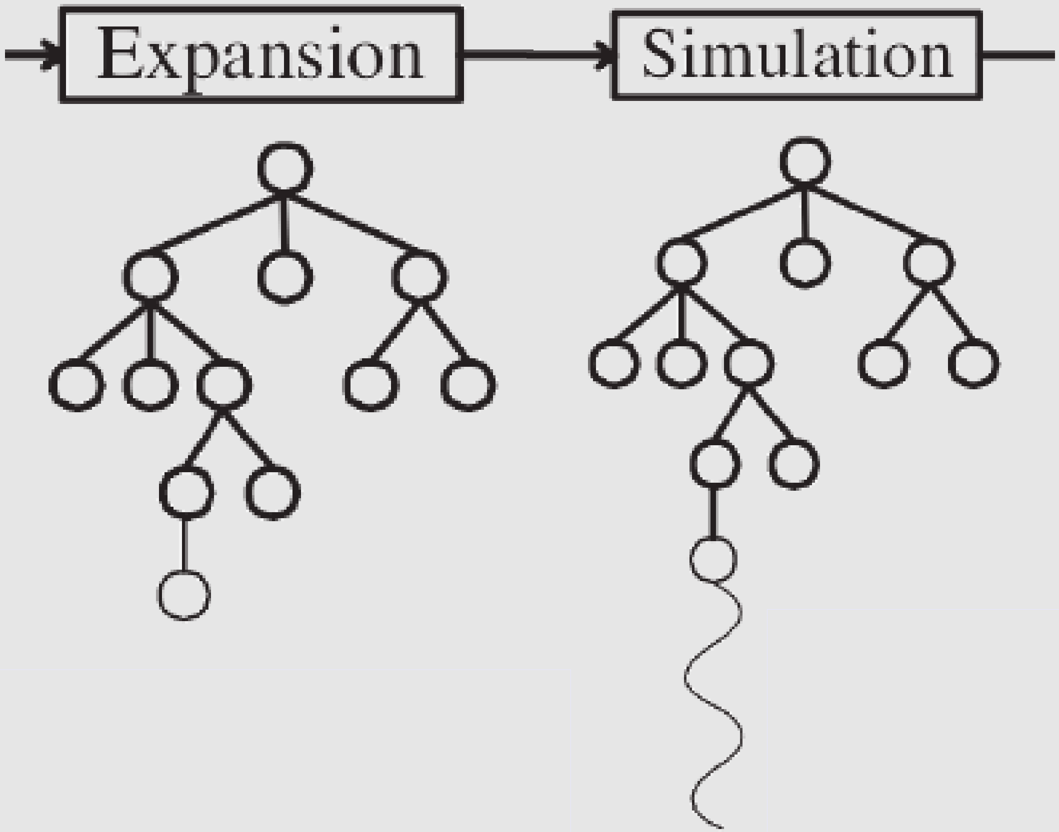
- UCT (Multi-Armed Bandit)

$$\underbrace{\boxed{\hat{V}(c_i)}}_{\text{exploitation}} + \underbrace{\boxed{C\sqrt{\frac{\ln \mathrm{T}(P)}{\mathrm{T}(c_i)}}}}_{\text{exploration}}$$

- CrazyStone

$$\mathrm{P}(c_i) \sim \exp\left(-2.4\frac{\hat{V}(c_{best}) - \hat{V}(c_i)}{\sqrt{2(\overline{\sigma}(c_{best})^2 + \overline{\sigma}(c_i)^2)}}\right)$$

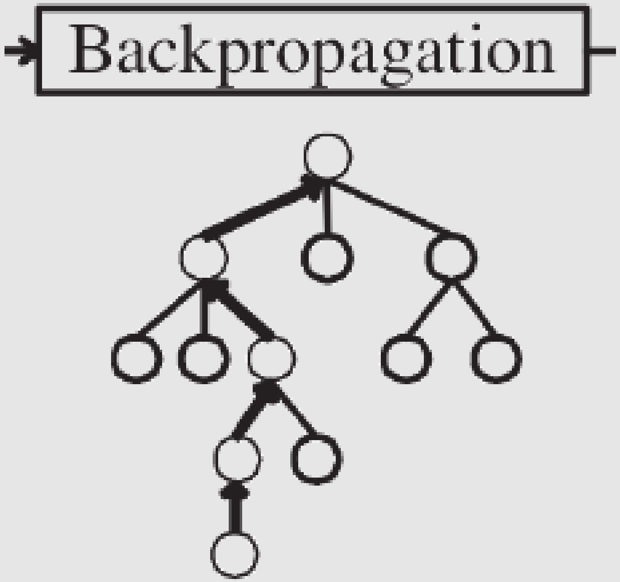
Expansion
Simulation



Backpropagation

$\hat{V}(P)$ is an estimate of the reward $r(P)$

$\mathrm{T}(P)$ is the number of samples

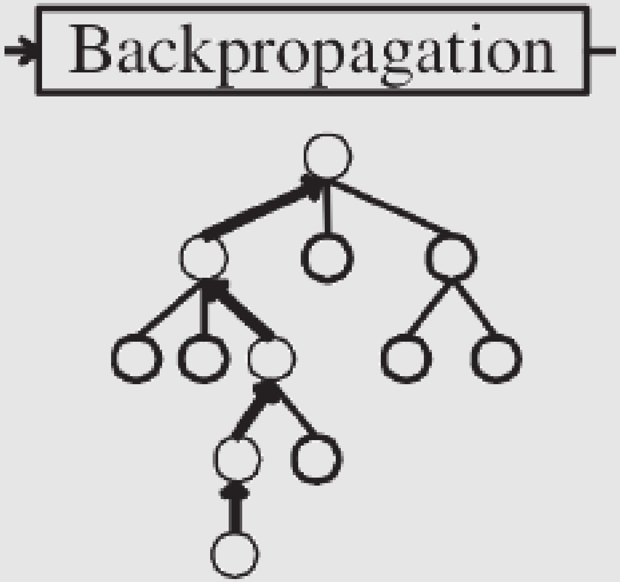


Backpropagation

$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples

- Sample-weighted average

$$\hat{V}(n) = \sum_j \frac{\mathrm{T}(c_j)}{\mathrm{T}(n)} \hat{V}(c_j)$$



Backpropagation

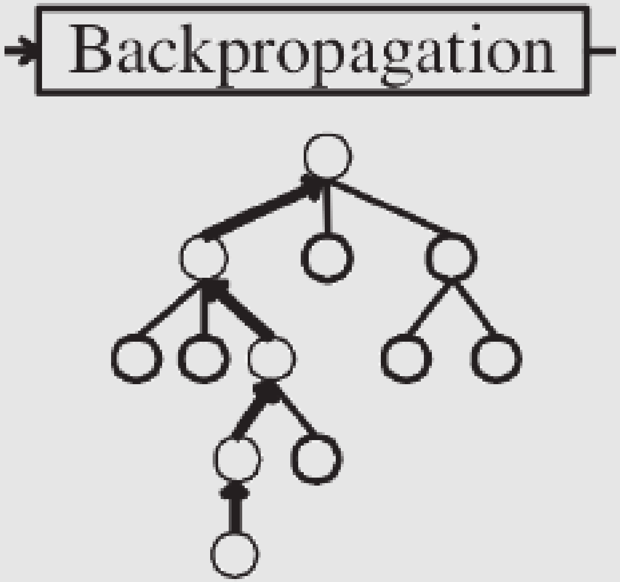
$\hat{V}(P)$ is an estimate of the reward $r(P)$
$\mathrm{T}(P)$ is the number of samples

- Sample-weighted average

$$\hat{V}(n) = \sum_j \frac{\mathrm{T}(c_j)}{\mathrm{T}(n)} \hat{V}(c_j)$$

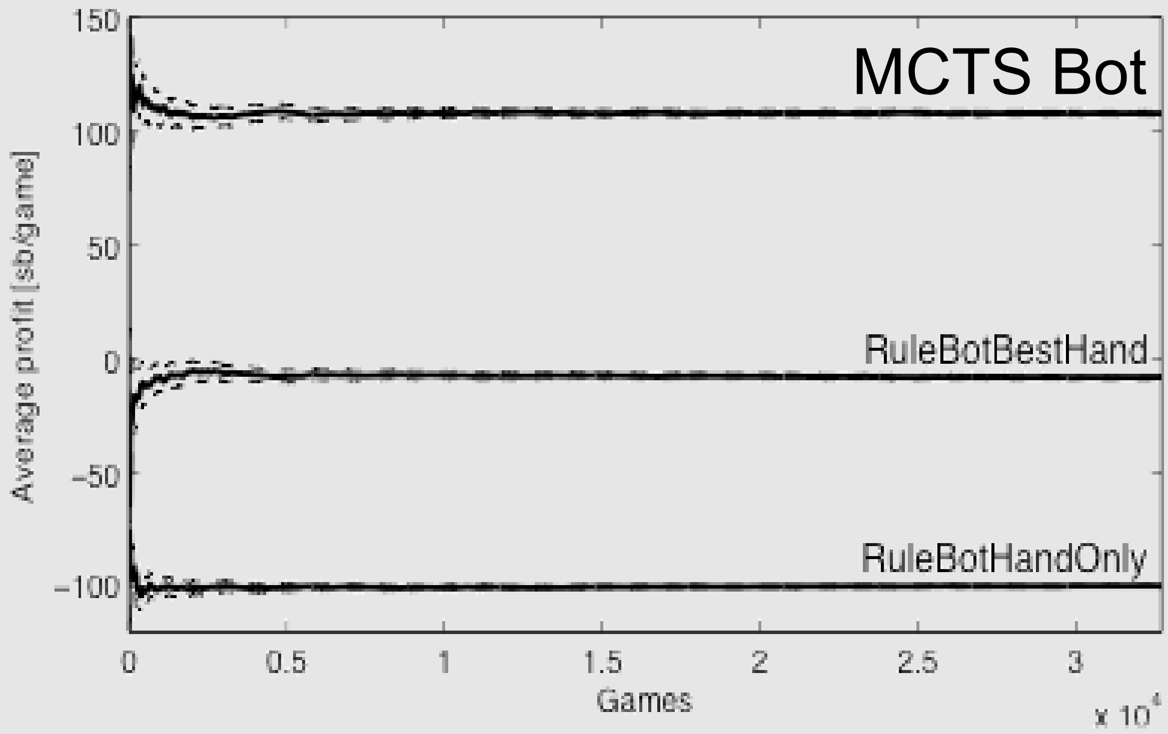
- Maximum child

$$\hat{V}(P) = \max_j \hat{V}(c_j)$$



Initial experiments

- 1*MCTS + 2*rule based
- Exploitative!



Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

MCTS for games with uncertainty?

- Expected reward distributions (ERD)
- Sample selection using ERD
- Backpropagation of ERD

[VandenBroeck09]

Expected reward distribution

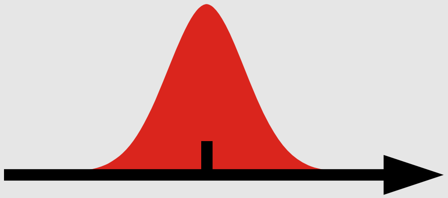
	MiniMax
Estimating	$r(P)$
10 samples	
100 samples	
∞ samples	
Variance	

Expected reward distribution

MiniMax

Estimating $r(P)$

10 samples



100 samples

∞ samples

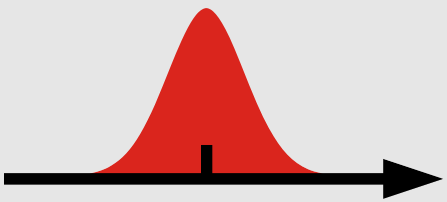
Variance

Expected reward distribution

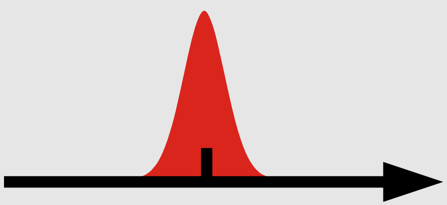
MiniMax

Estimating $r(P)$

10 samples



100 samples



∞ samples

Variance

Expected reward distribution

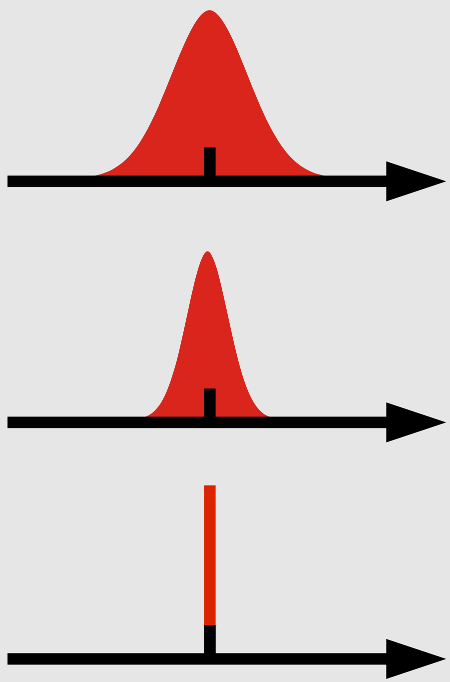
MiniMax

Estimating $r(P)$

10 samples

100 samples

∞ samples



Variance

Expected reward distribution

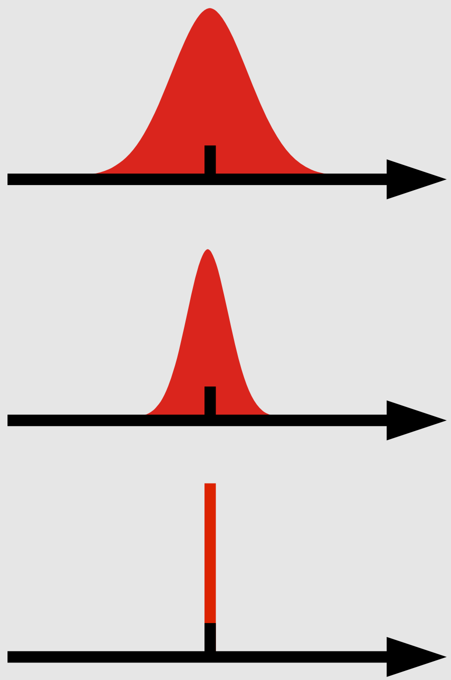
MiniMax

Estimating $r(P)$

10 samples

100 samples

∞ samples



Variance *Sampling*

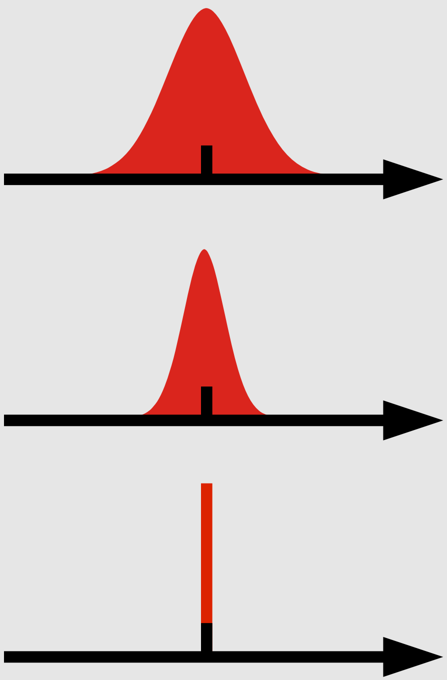
Expected reward distribution

	MiniMax	ExpectiMax/MixiMax
Estimating	$r(P)$	$r(P)$

10 samples

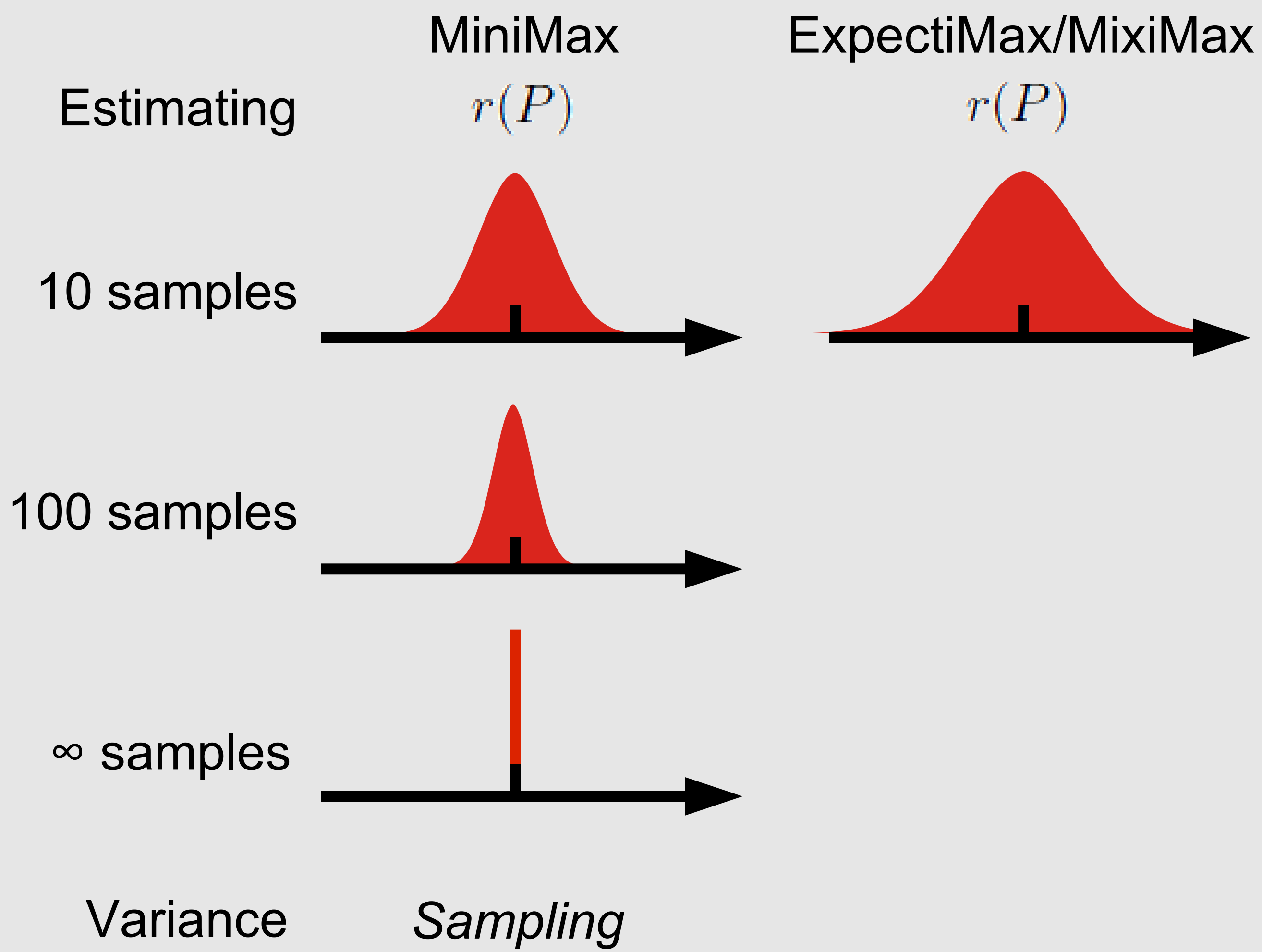
100 samples

∞ samples

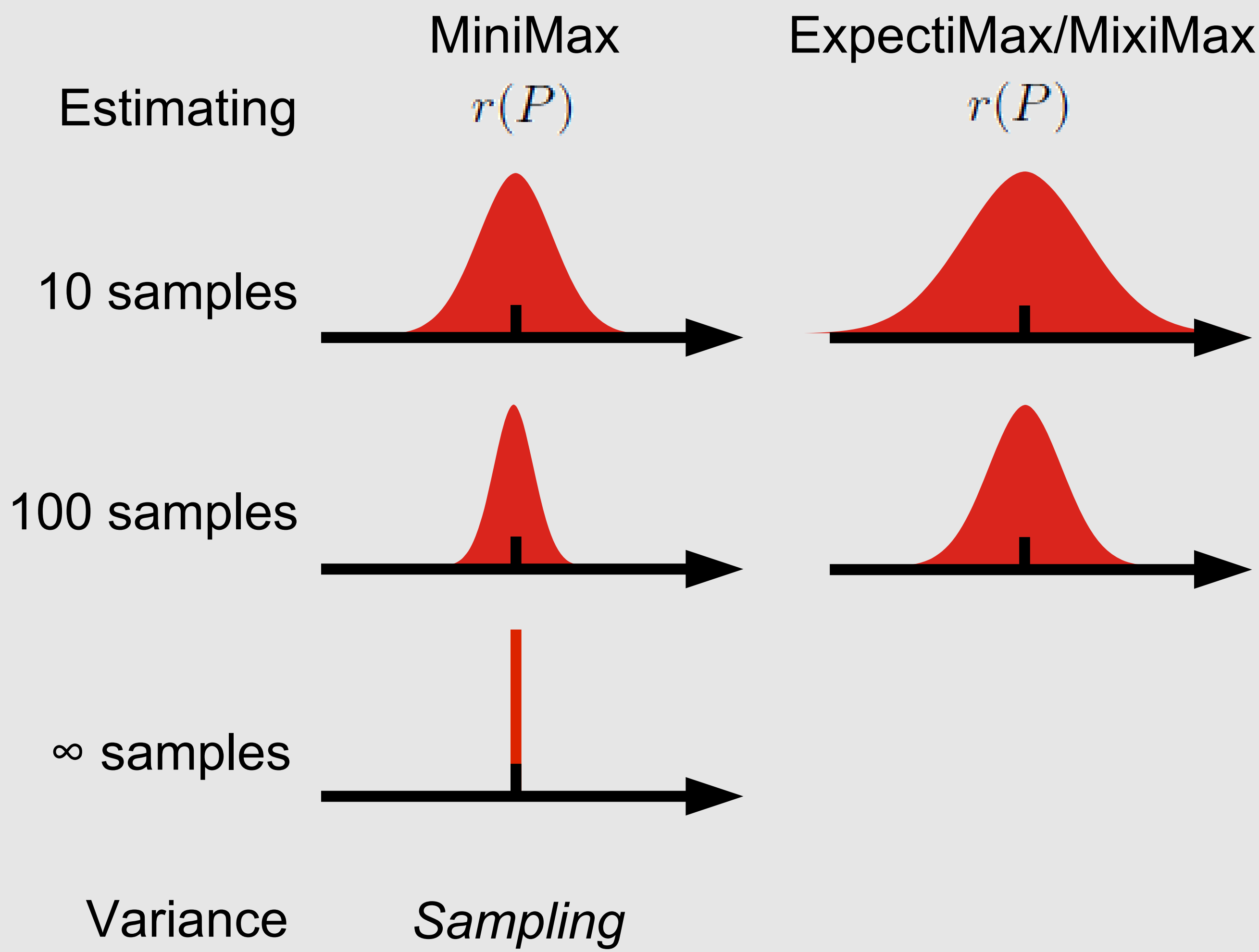


Variance	*Sampling*	

Expected reward distribution

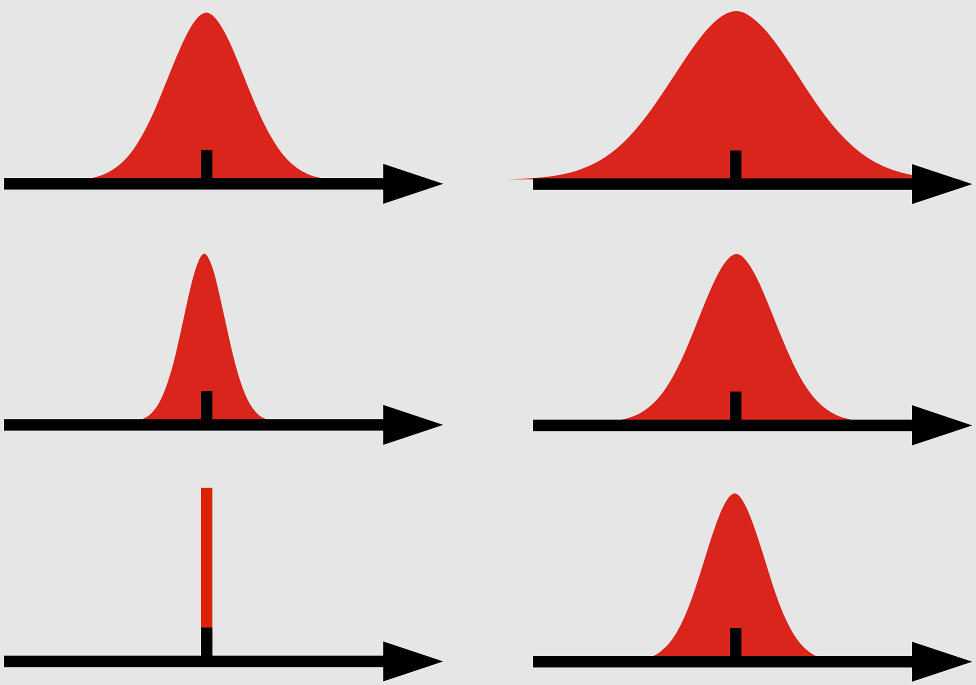


Expected reward distribution



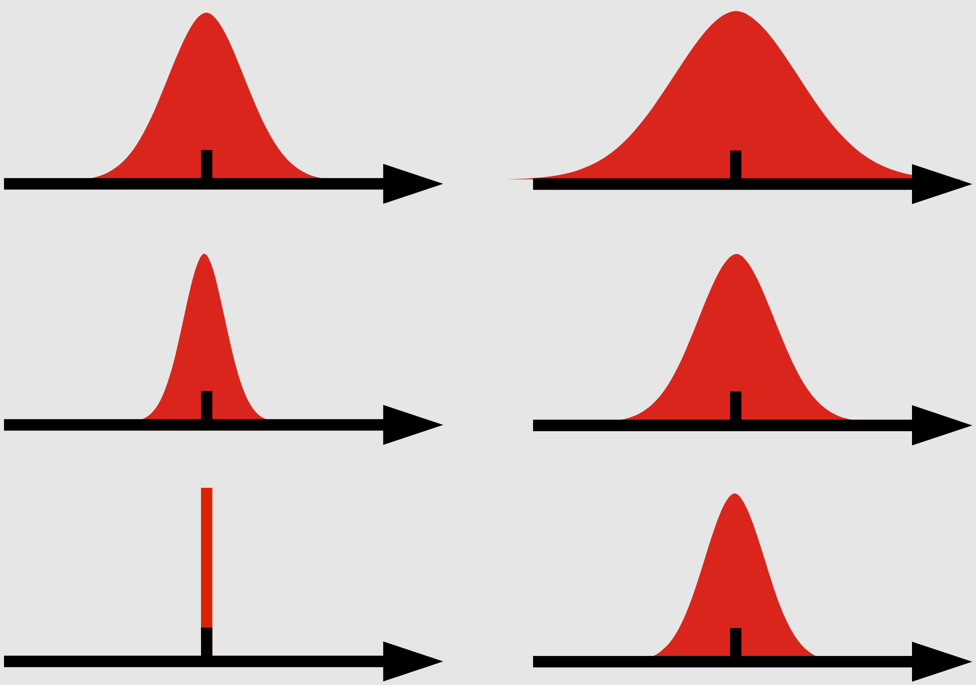
Expected reward distribution

	MiniMax	ExpectiMax/MixiMax
Estimating	$r(P)$	$r(P)$
10 samples		
100 samples		
∞ samples		
Variance	*Sampling*	

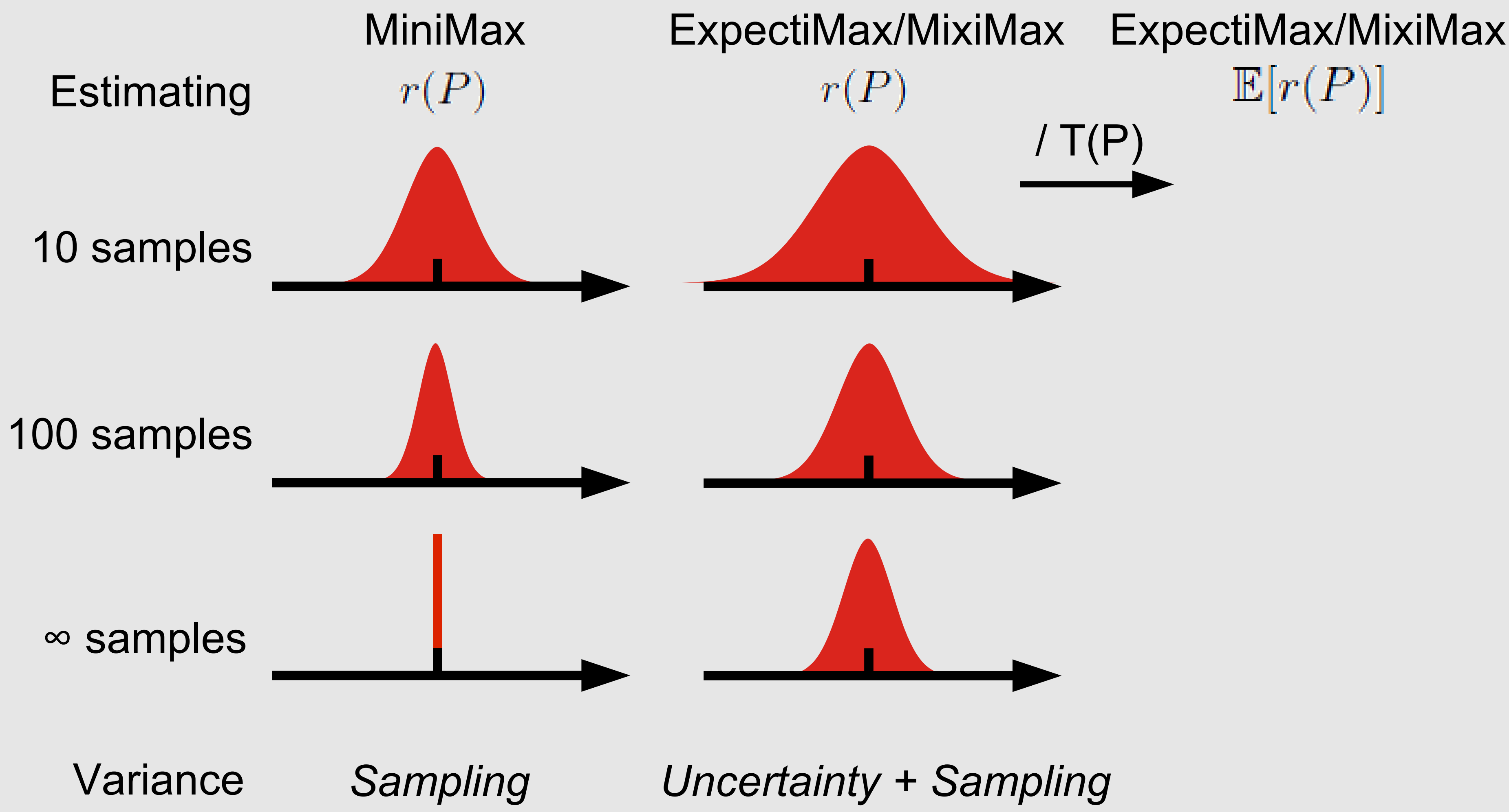


Expected reward distribution

	MiniMax	ExpectiMax/MixiMax
Estimating	$r(P)$	$r(P)$
10 samples		
100 samples		
∞ samples		
Variance	*Sampling*	*Uncertainty + Sampling*



Expected reward distribution



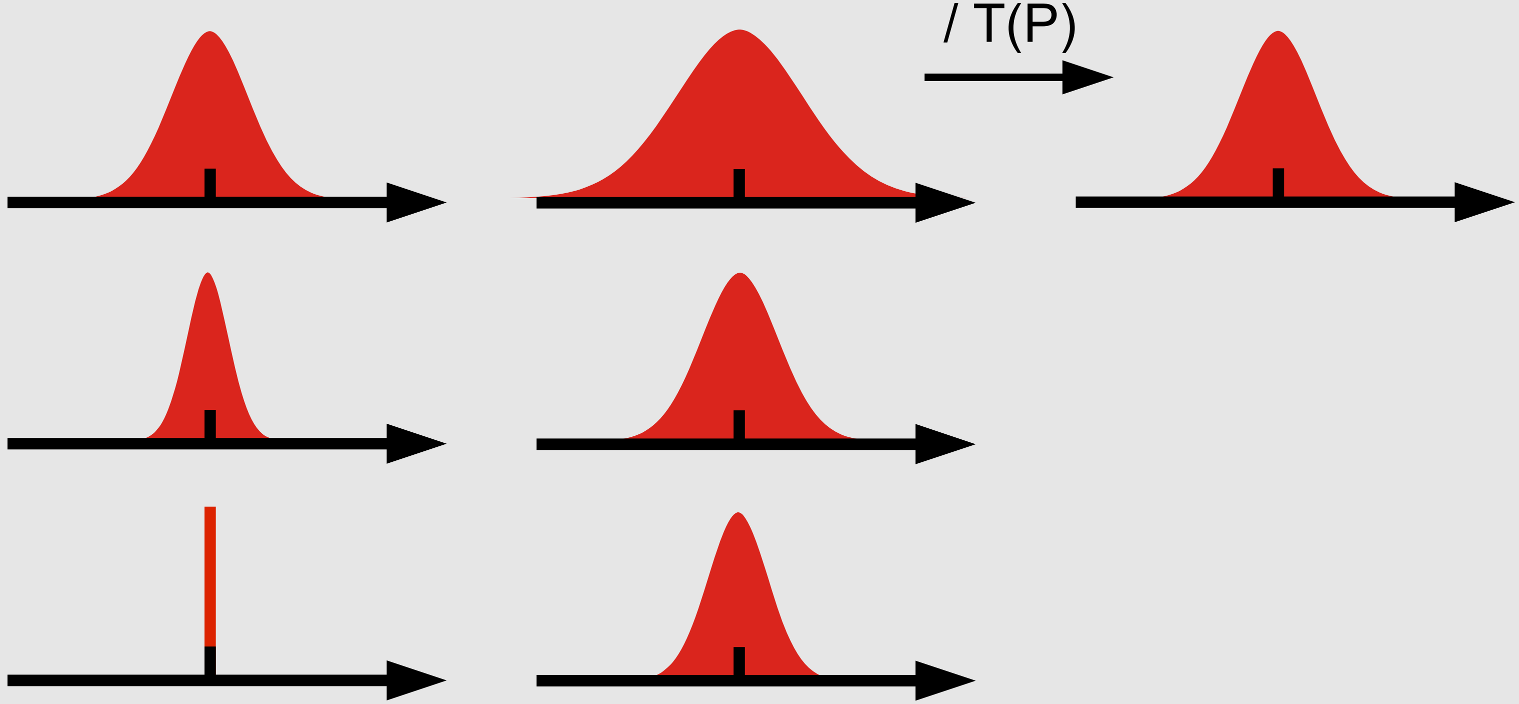
Expected reward distribution

Estimating	MiniMax $r(P)$	ExpectiMax/MixiMax $r(P)$	ExpectiMax/MixiMax $\mathbb{E}[r(P)]$

10 samples

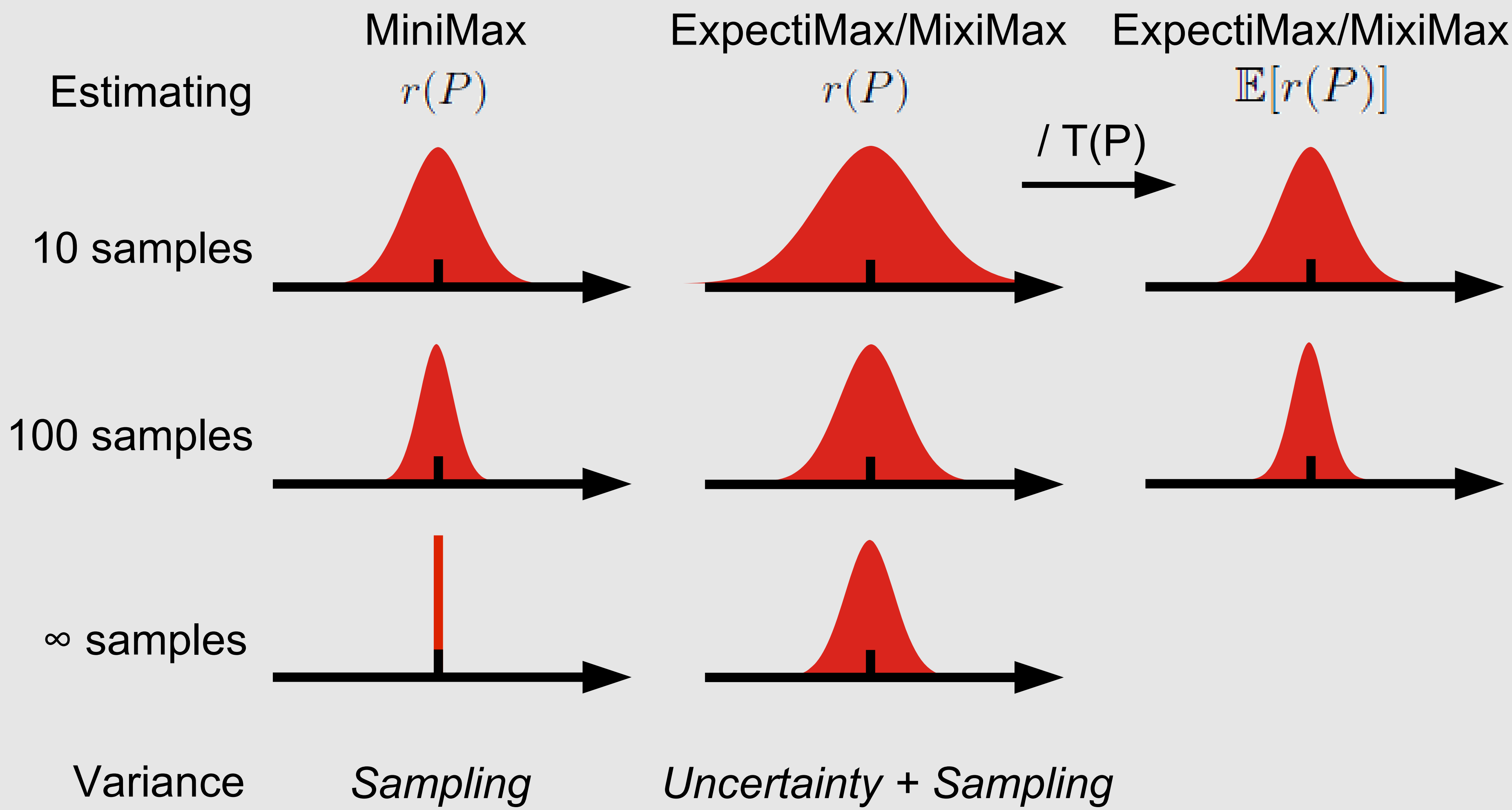
100 samples

∞ samples

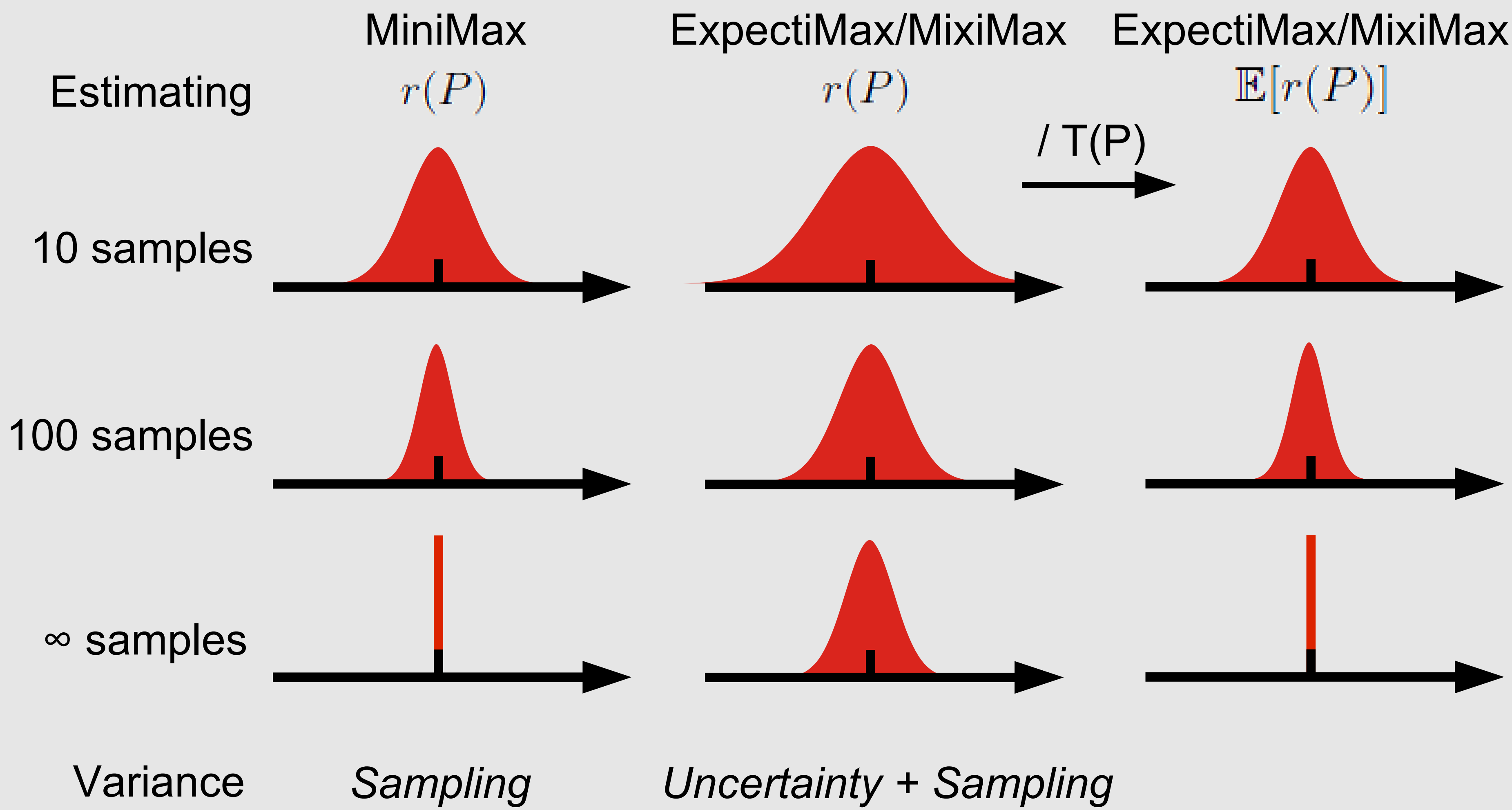


Variance	*Sampling*	*Uncertainty + Sampling*

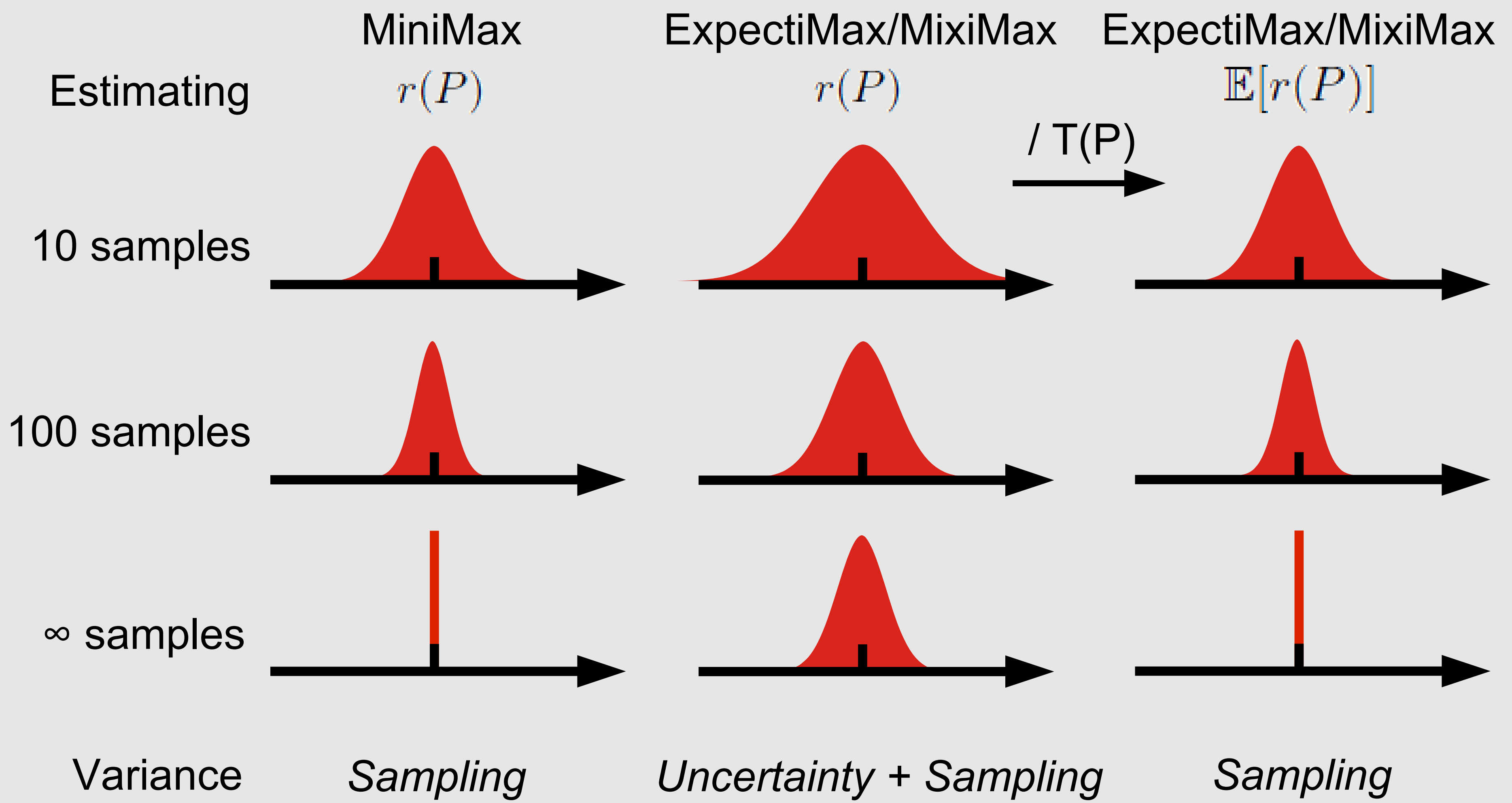
Expected reward distribution



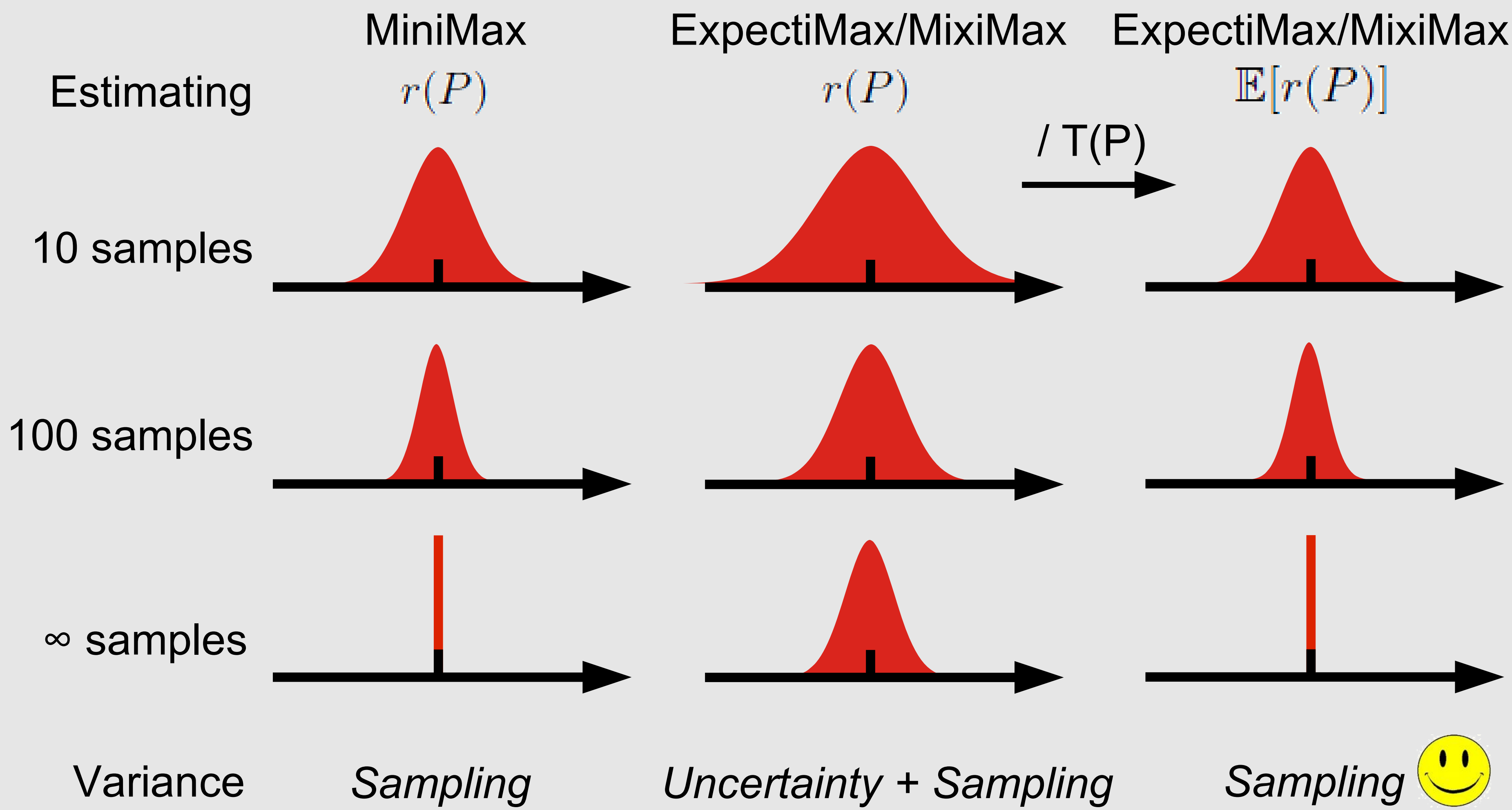
Expected reward distribution



Expected reward distribution



Expected reward distribution



ERD selection strategy

- Objective?
  - Find maximum expected reward
  - Sample more in subtrees with
    - (1) High expected reward
    - (2) Uncertain estimate
- UCT does (1) but not really (2)
- CrazyStone does (1) and (2) for deterministic games (Go)
- **UCT+ selection**: $\underbrace{\hat{V}(c_i)}_{(1)} + \underbrace{C.\sigma_{\hat{V},c_i}}_{(2)}$

ERD selection strategy

- Objective?
  - Find maximum expected reward
  - Sample more in subtrees with
    - (1) High expected reward
    - (2) Uncertain estimate
- UCT does (1) but not really (2)
- CrazyStone does (1) and (2) for deterministic games (Go)
- **UCT+ selection**: $\hat{V}(c_i) + C.\sigma_{\hat{V},c_i}$

"Expected value under perfect play"

ERD selection strategy

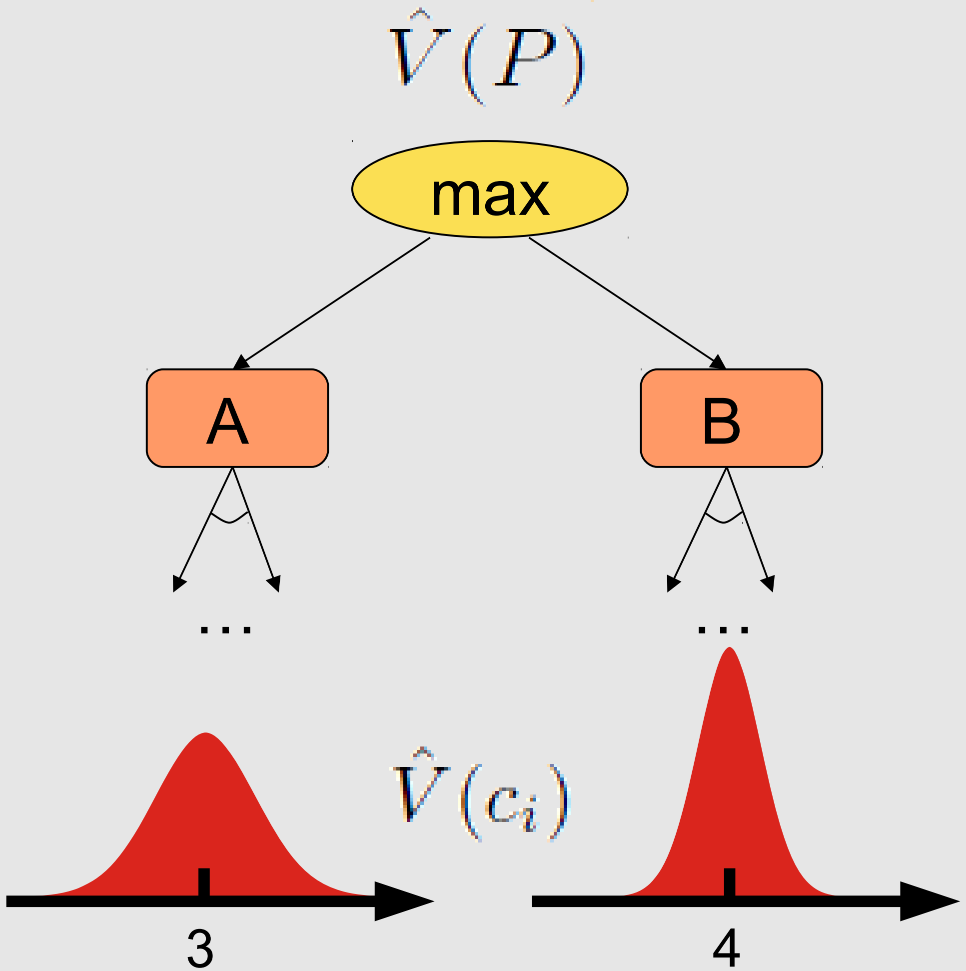
- Objective?
  - Find maximum expected reward
  - Sample more in subtrees with

    (1) High expected reward

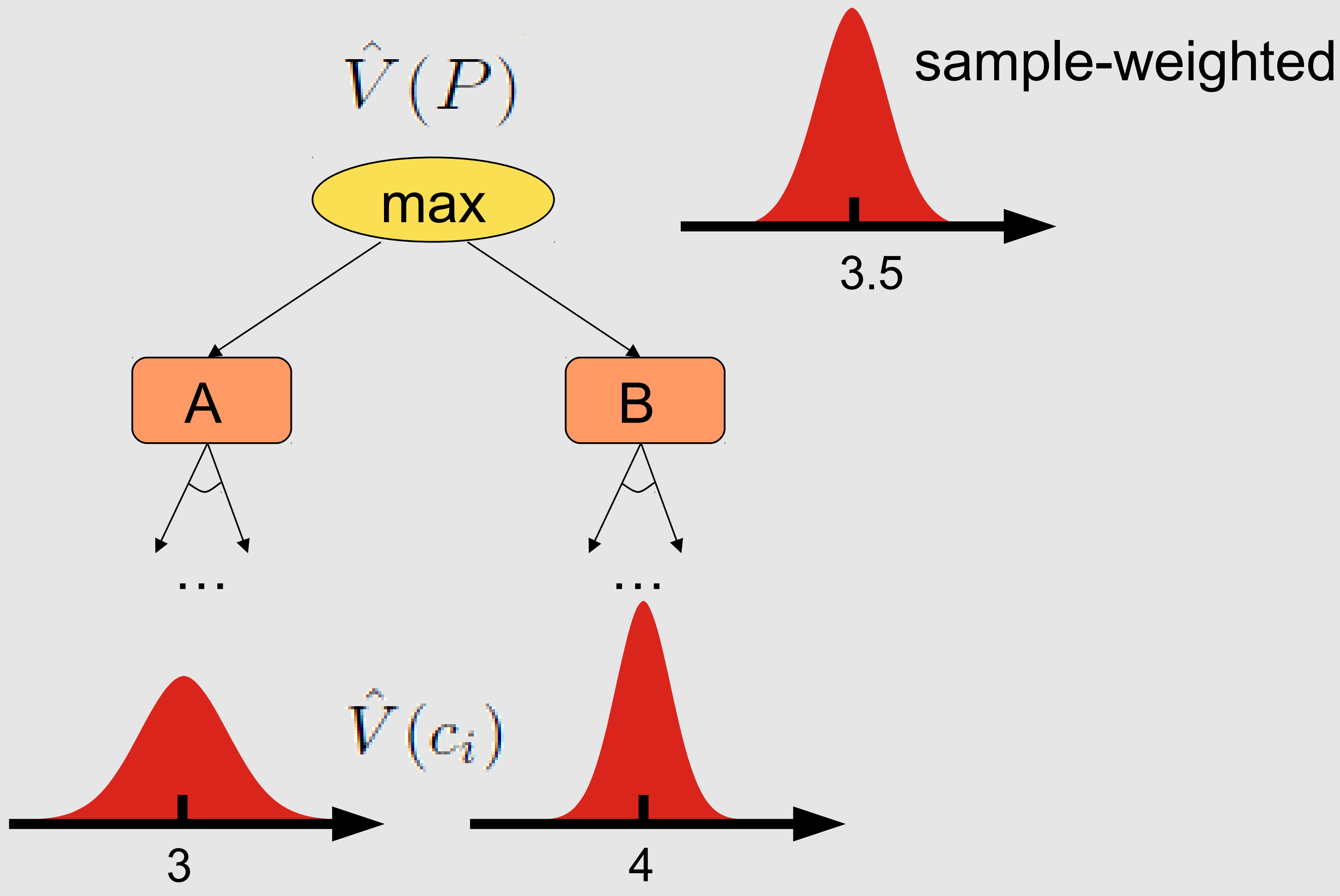
    (2) Uncertain estimate
- UCT does (1) but not really (2)
- CrazyStone does (1) and (2) for deterministic games (Go)
- **UCT+ selection**: $\hat{V}(c_i) + C.\sigma_{\hat{V},c_i}$

"Measure of uncertainty due to sampling"

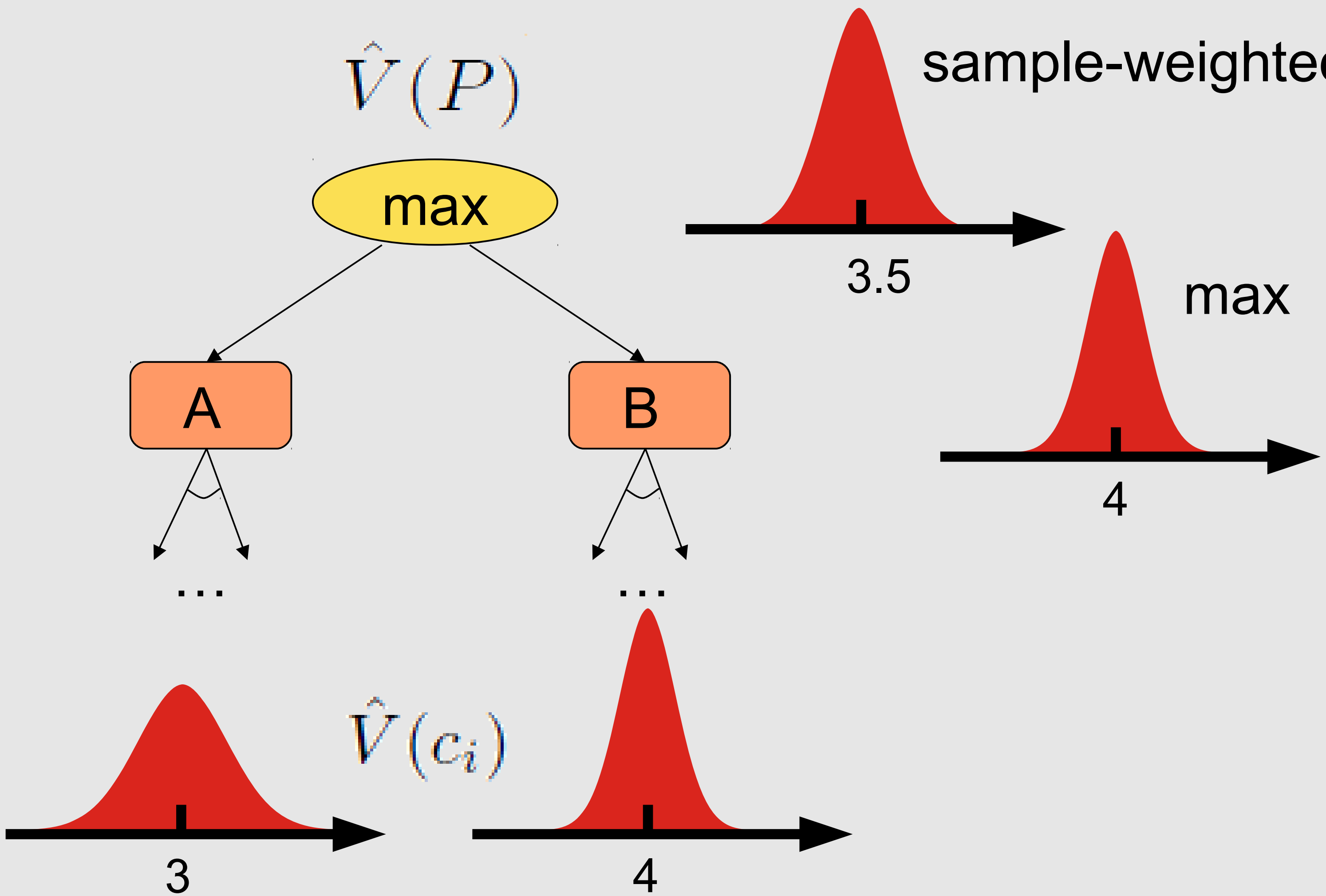
ERD max-distribution backpropagation



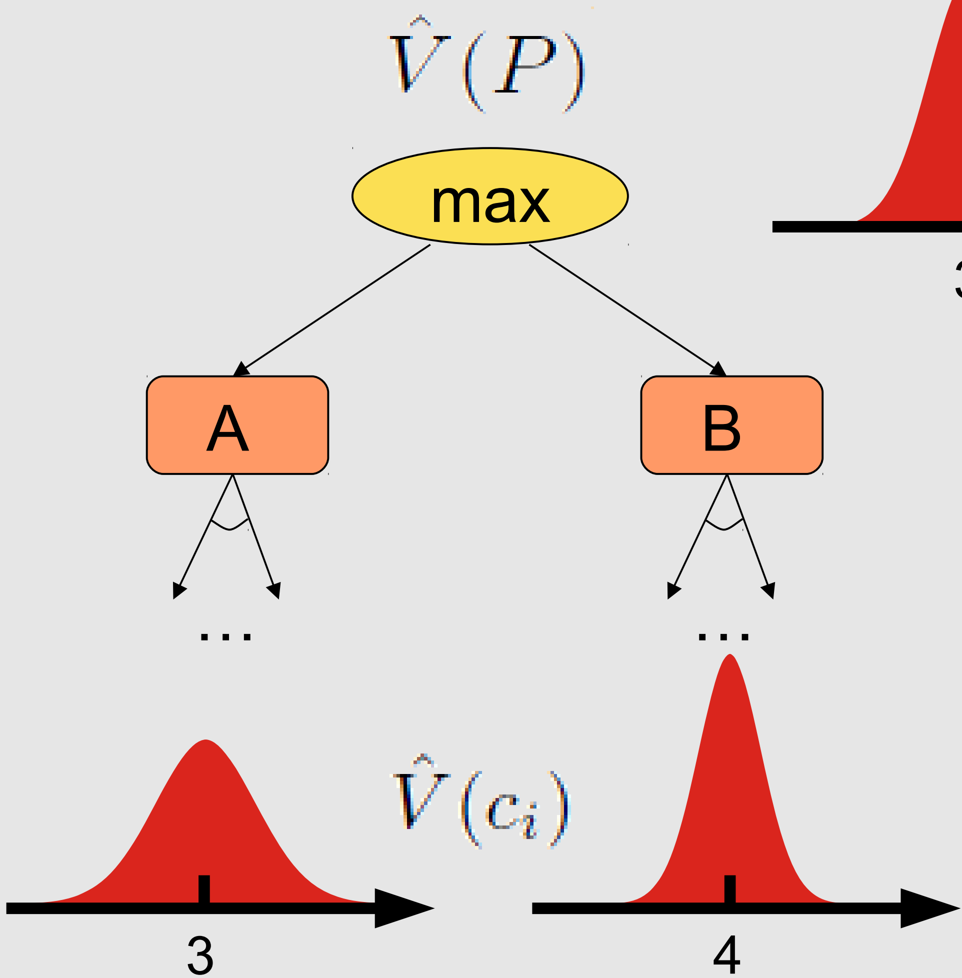
ERD max-distribution backpropagation

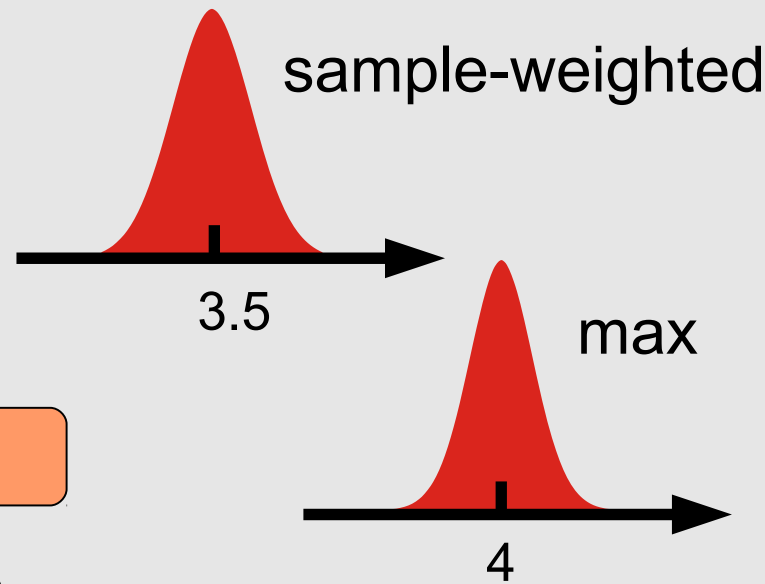


ERD max-distribution backpropagation



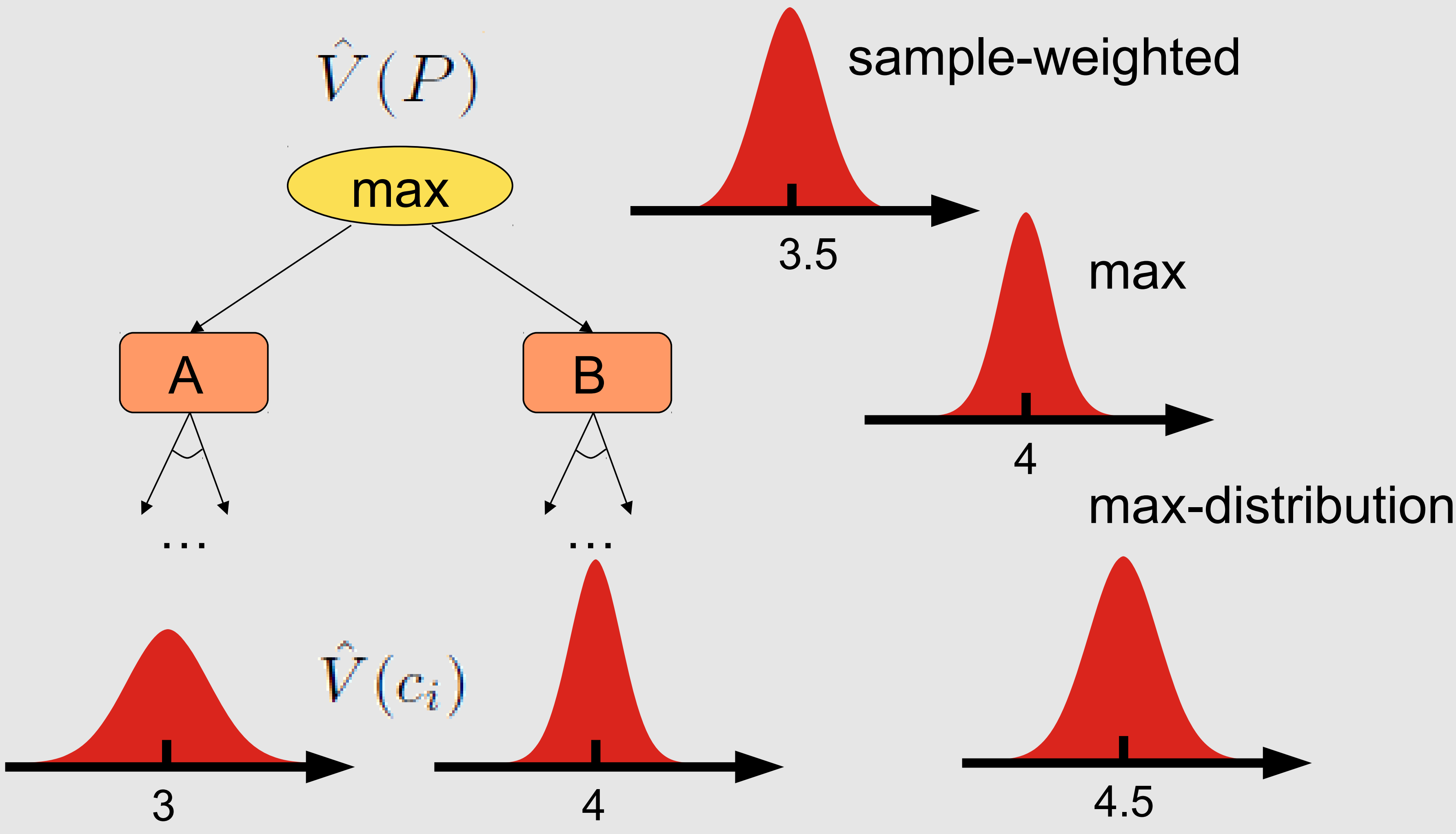
ERD max-distribution backpropagation



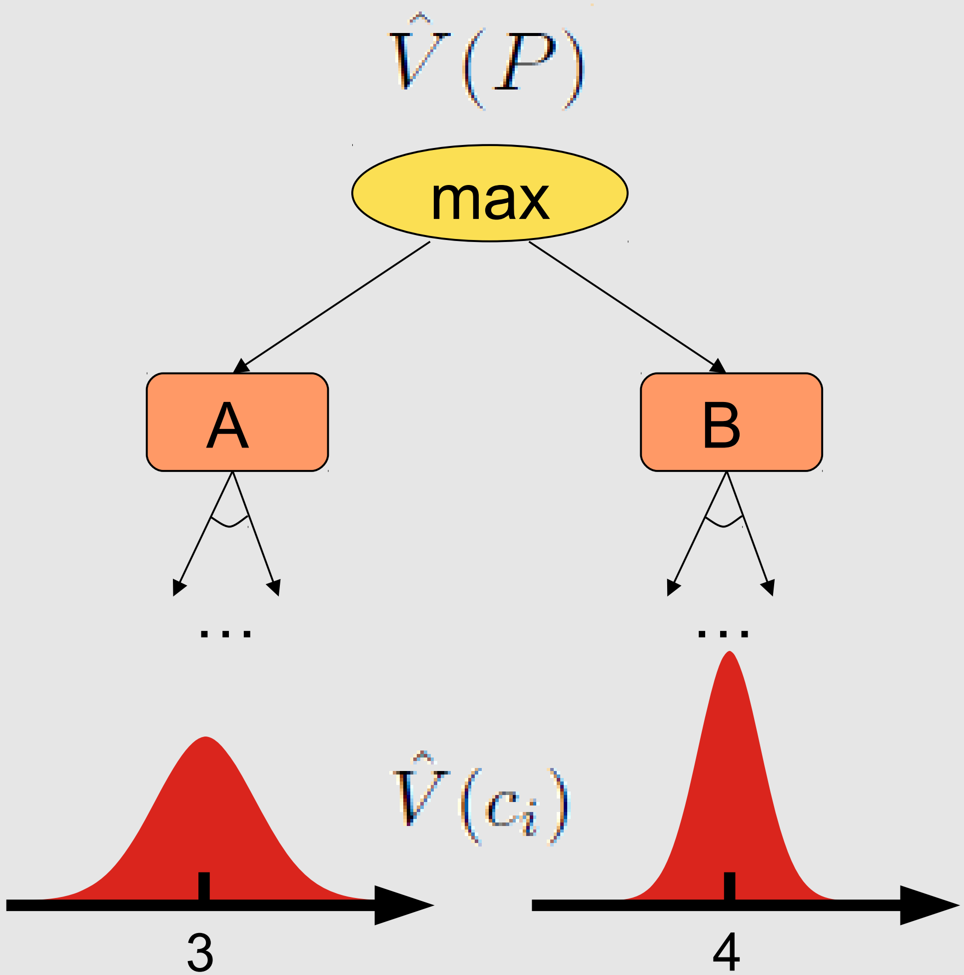


"When the game reaches P, we'll have more time to find the real $\mathbb{E}[r(P)]$ "

ERD max-distribution backpropagation



ERD max-distribution backpropagation

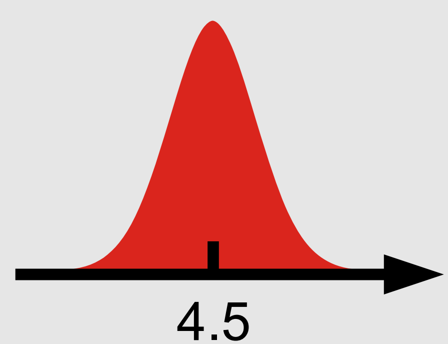


P(B<4) = 0.5 P(B>4) = 0.5

P(A<4) = 0.8 P(A>4) = 0.2

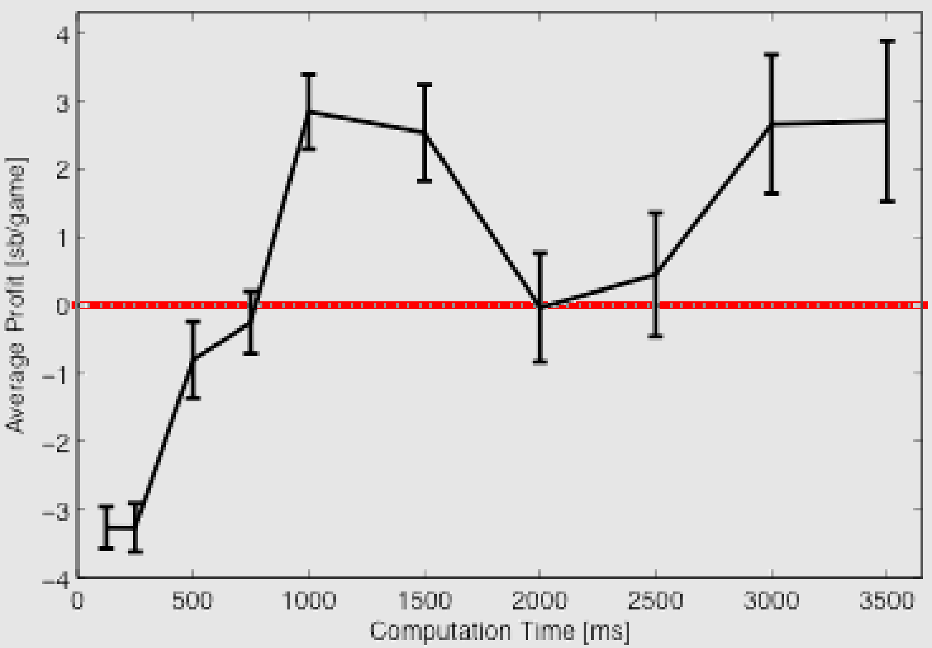
	A<4	A>4
B<4	0.8*0.5	0.2*0.5
B>4	0.8*0.5	0.2*0.5

P(max(A,B)>4) = 0.6
> 0.5

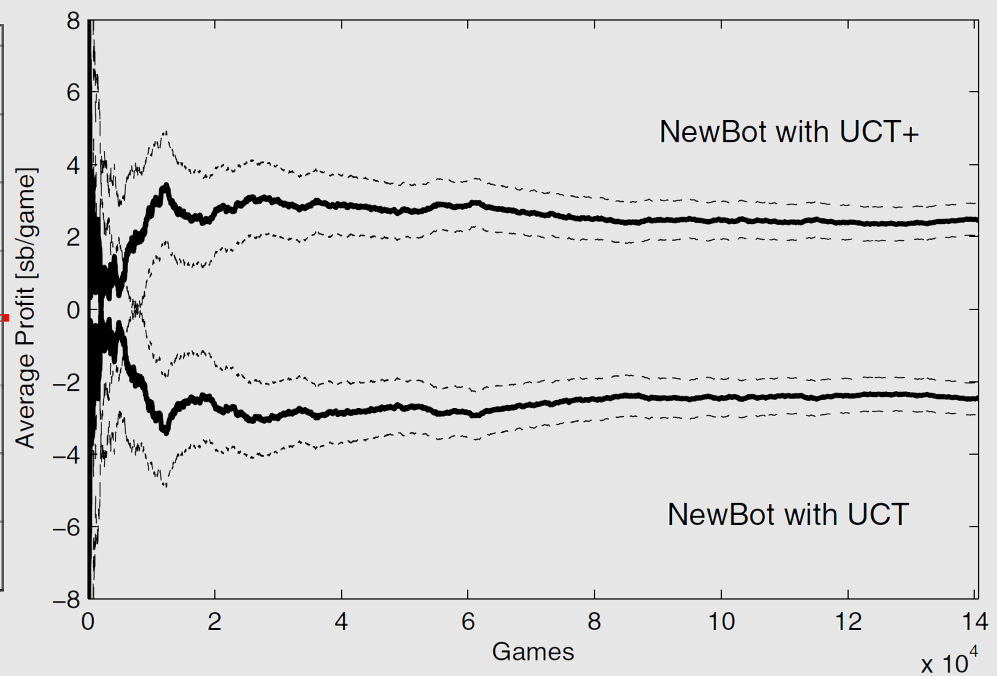


Experiments

- 2*MCTS
  - Max-distribution
  - Sample-weighted



- 2*MCTS
  - UCT+ (stddev)
  - UCT

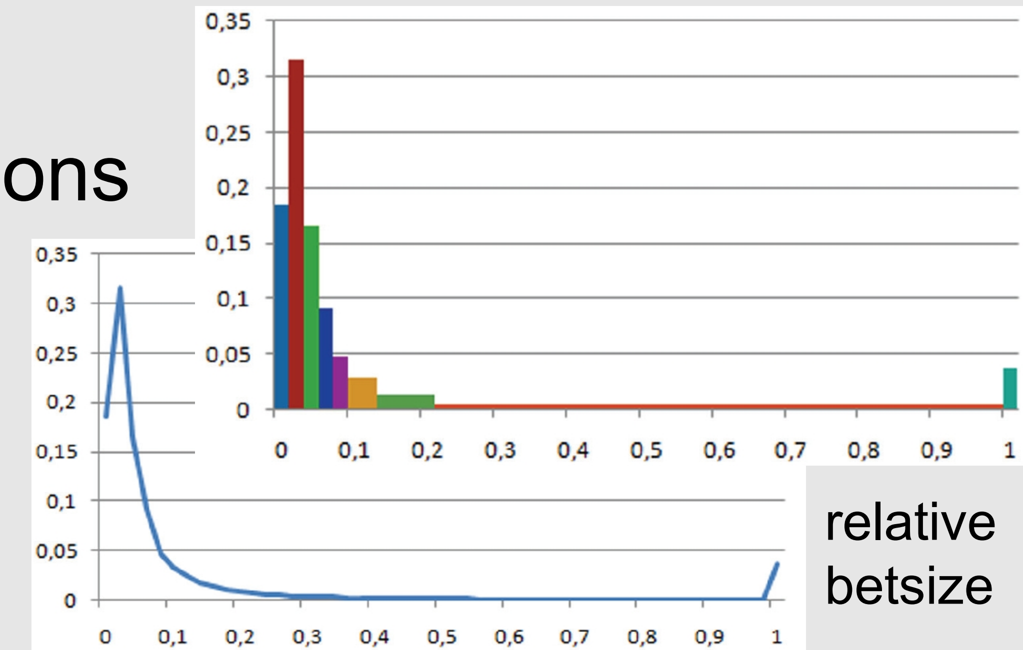


Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Dealing with continuous actions

- Sample discrete actions



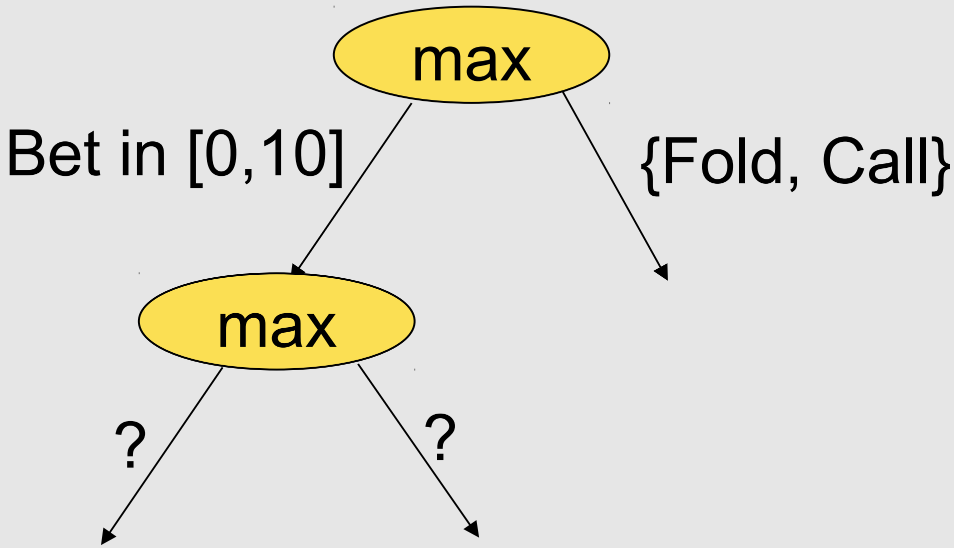
- Progressive unpruning [Chaslot08] (ignores smoothness of EV function)
- ...
- Tree learning search (work in progress)

Tree learning search

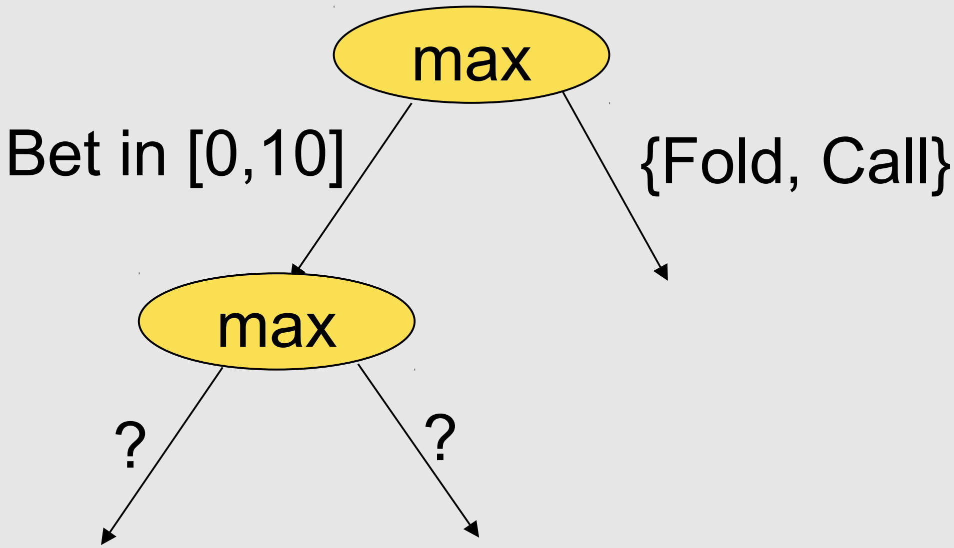
- Based on regression tree induction from ***data streams***
  - training examples arrive ***quickly***
  - nodes ***split*** when significant reduction in stddev
  - training examples are immediately ***forgotten***

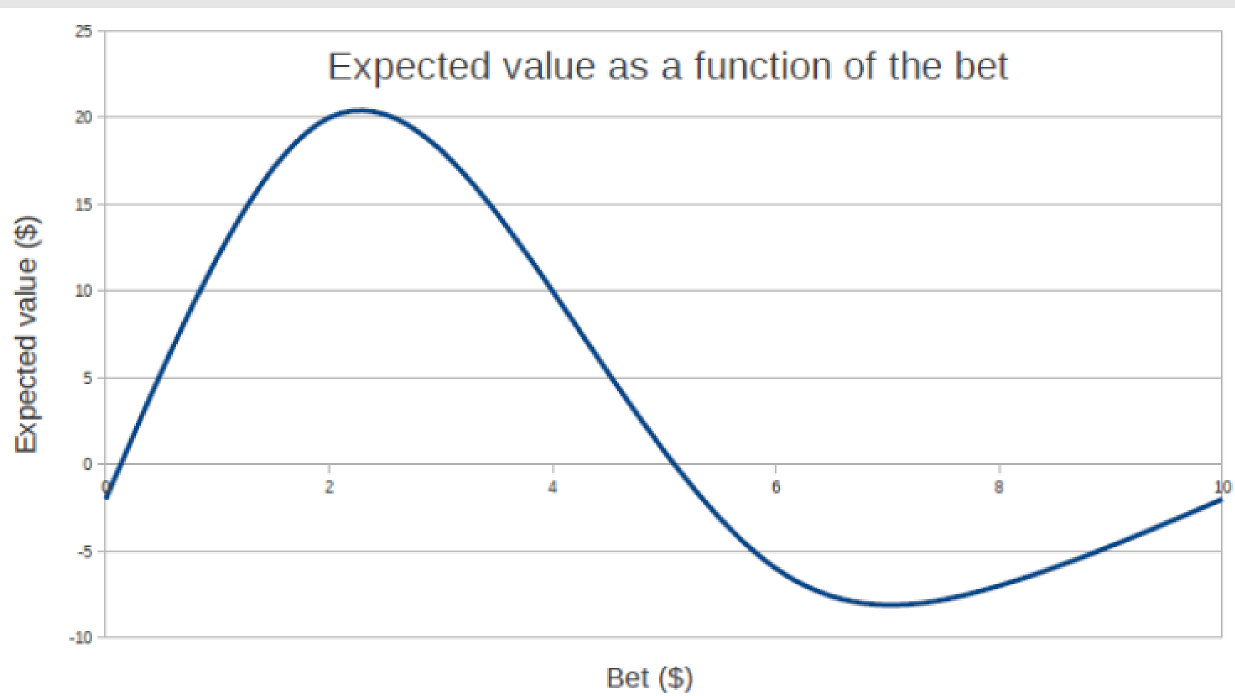
- Edges in TLS tree are not actions, but ***sets of actions***, e.g., (raise in [2,40]), (fold or call)
- MCTS provides a ***stream*** of (action,EV) examples
- Split action sets to reduce stddev of EV (when significant)

Tree learning search

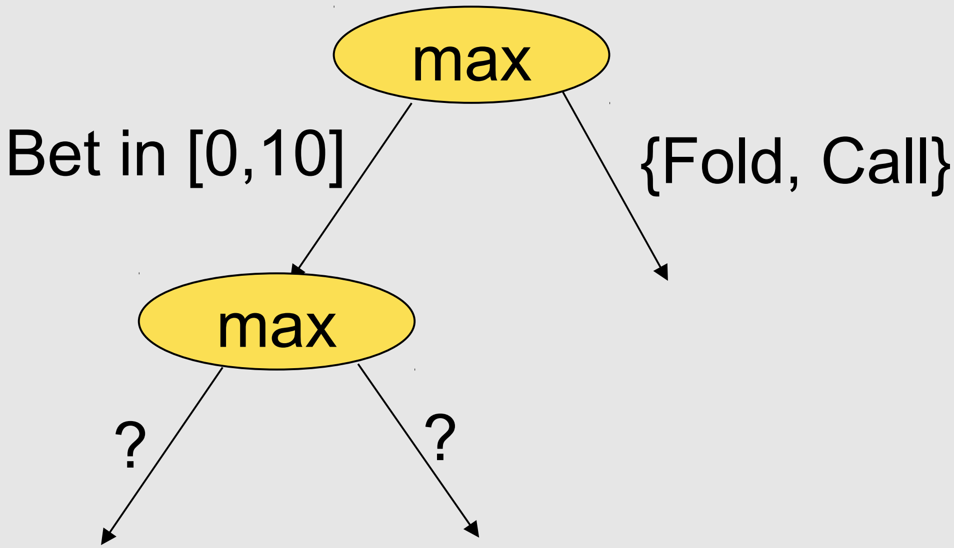


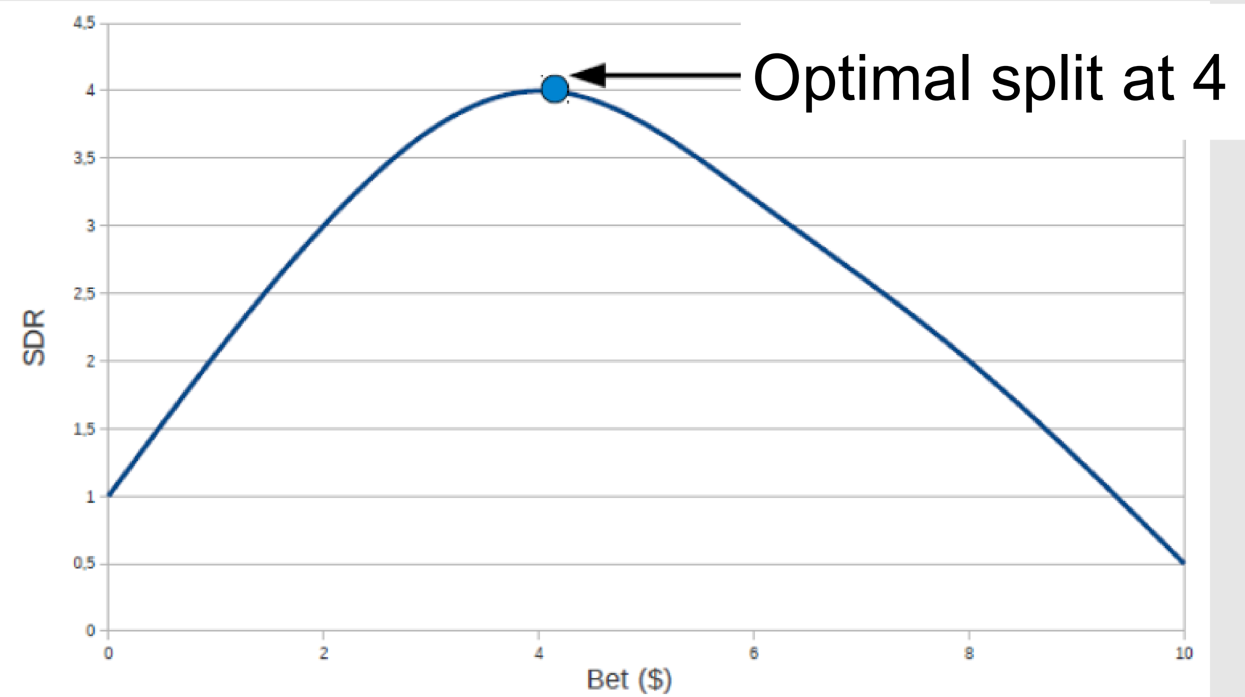
Tree learning search



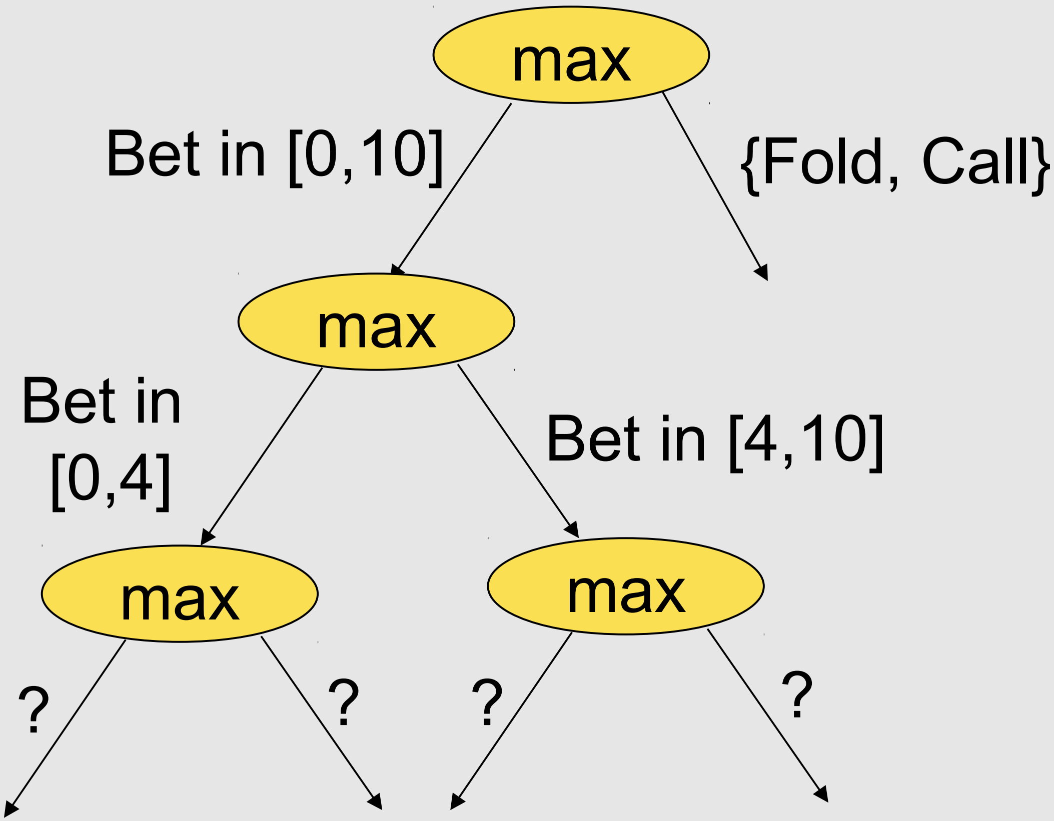


Tree learning search

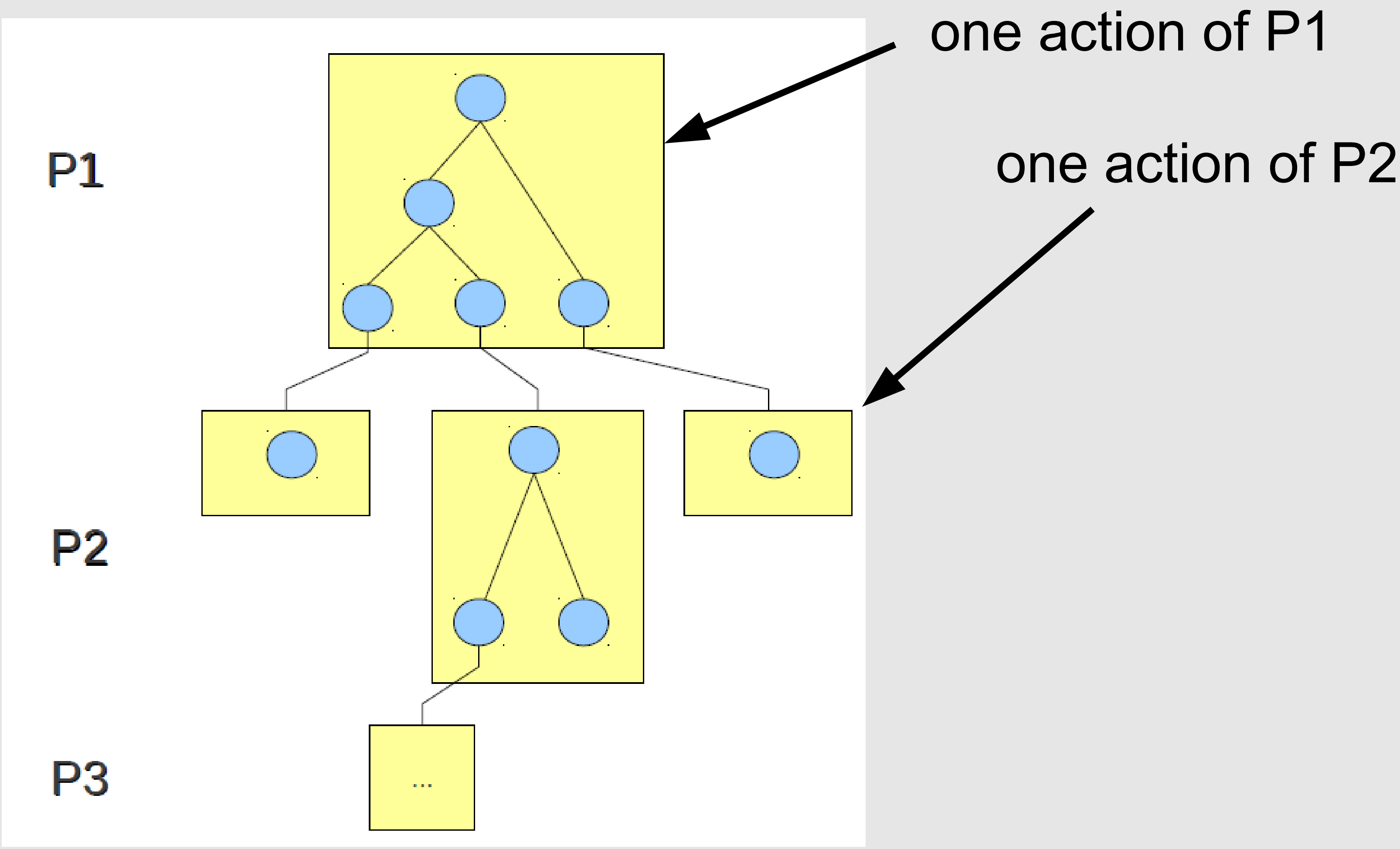




Tree learning search

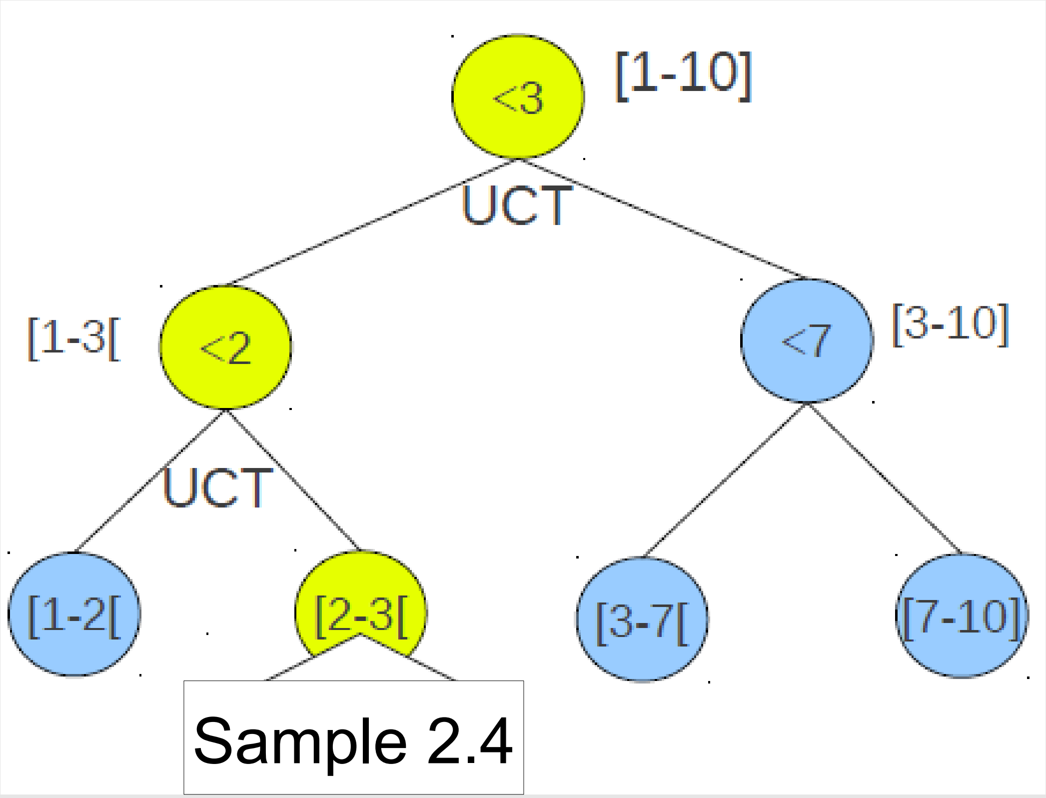


Tree learning search



Selection Phase

P1

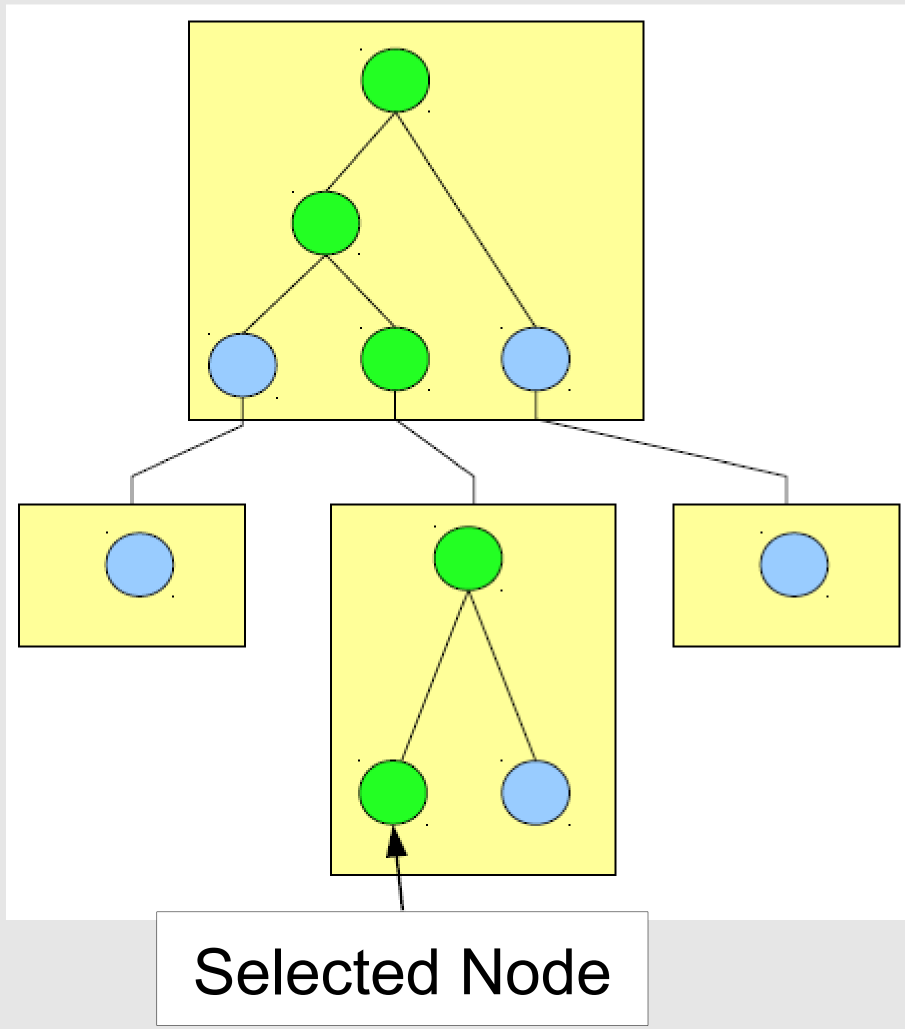


Each node has EV estimate, which generalizes over actions

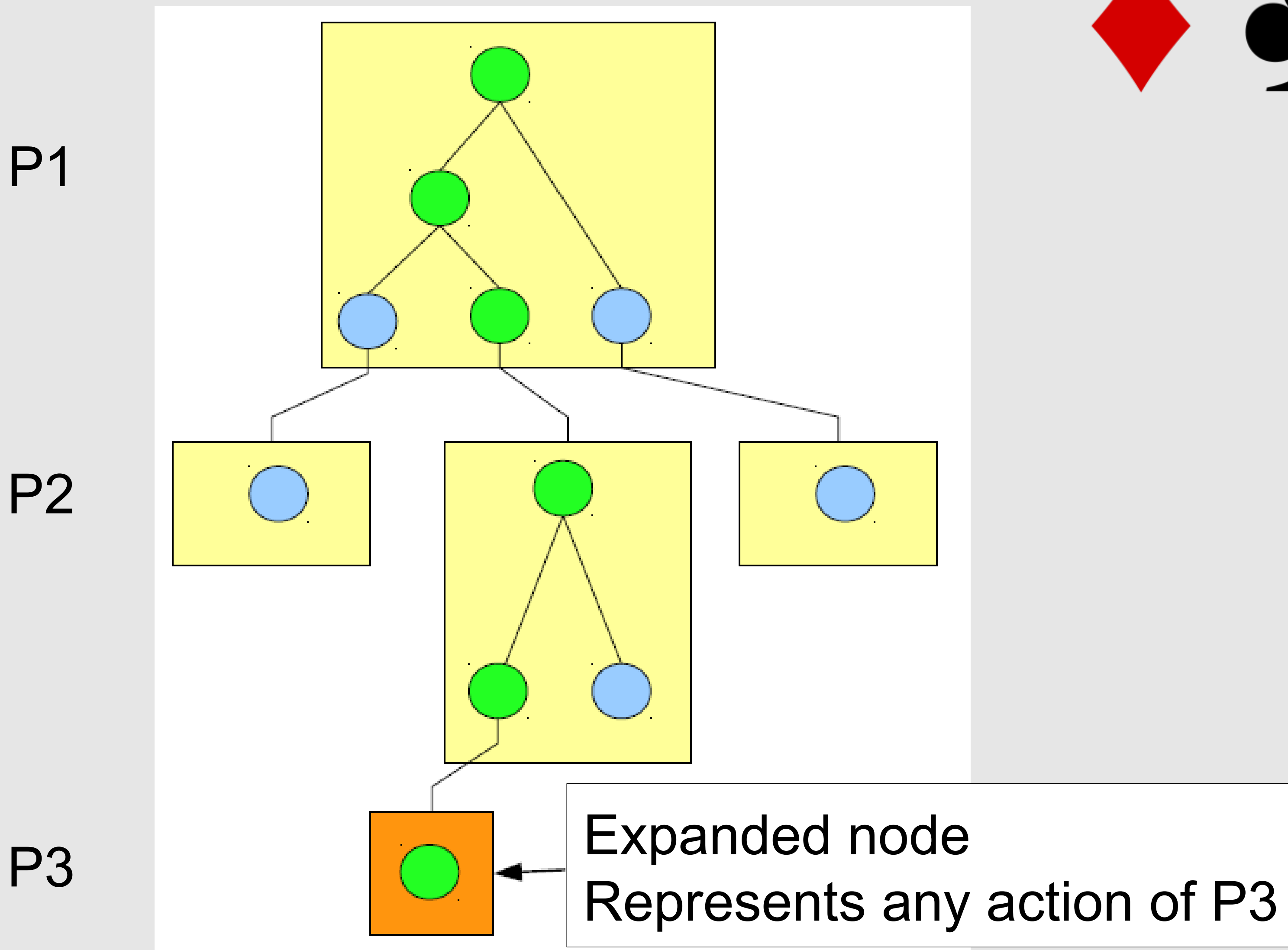
Expansion

P1

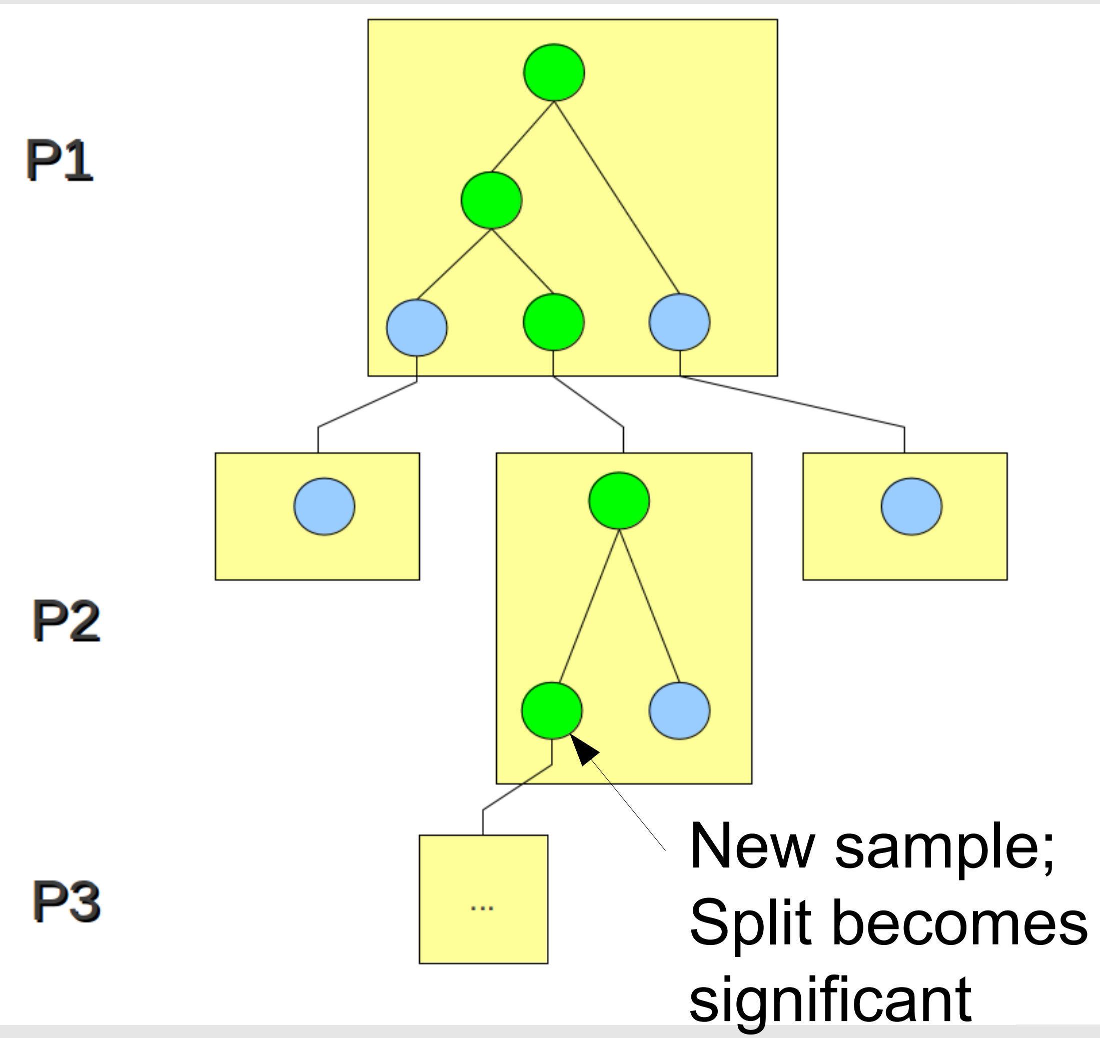
P2



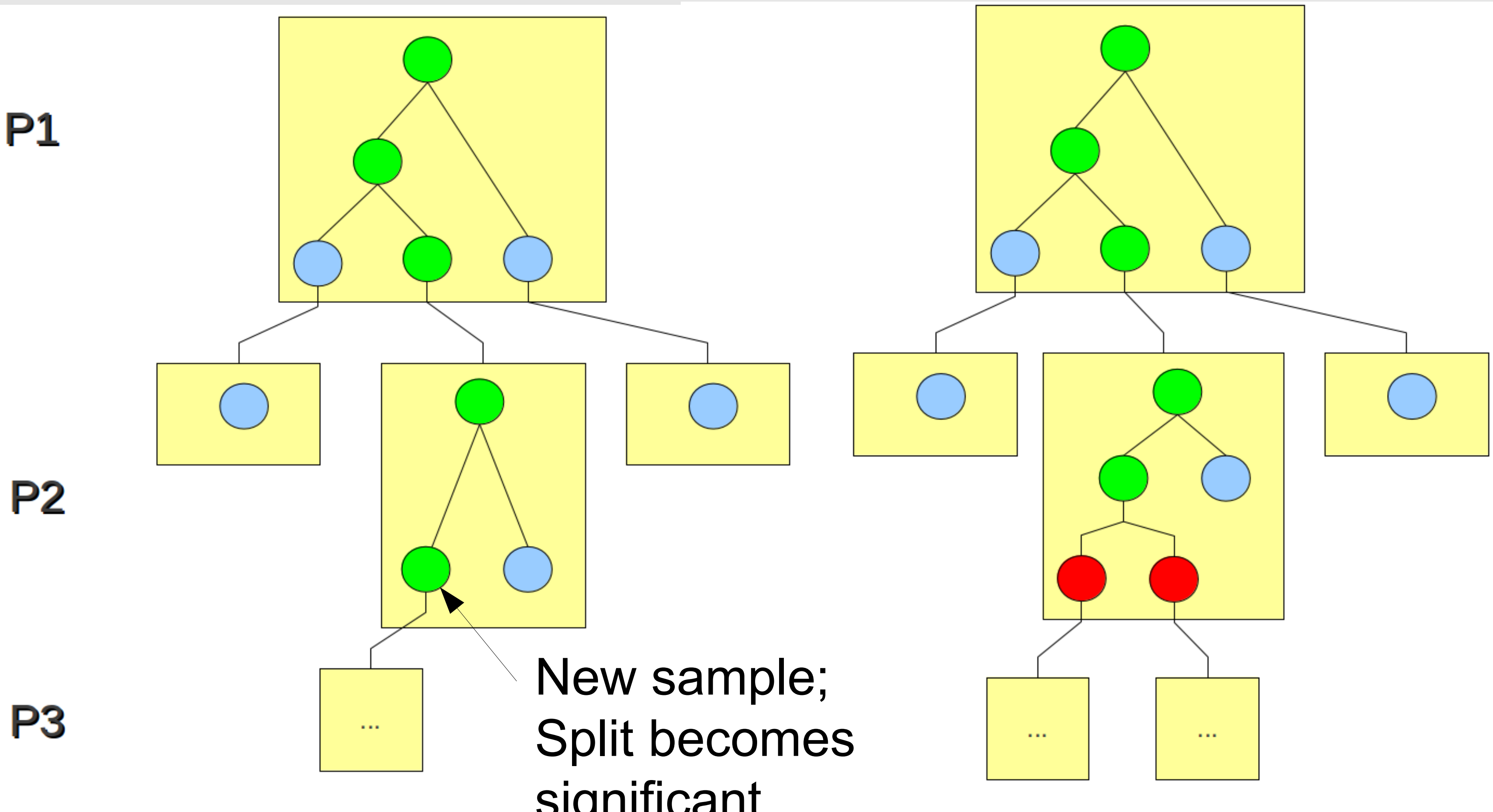
Expansion



Backpropagation



Backpropagation



Outline

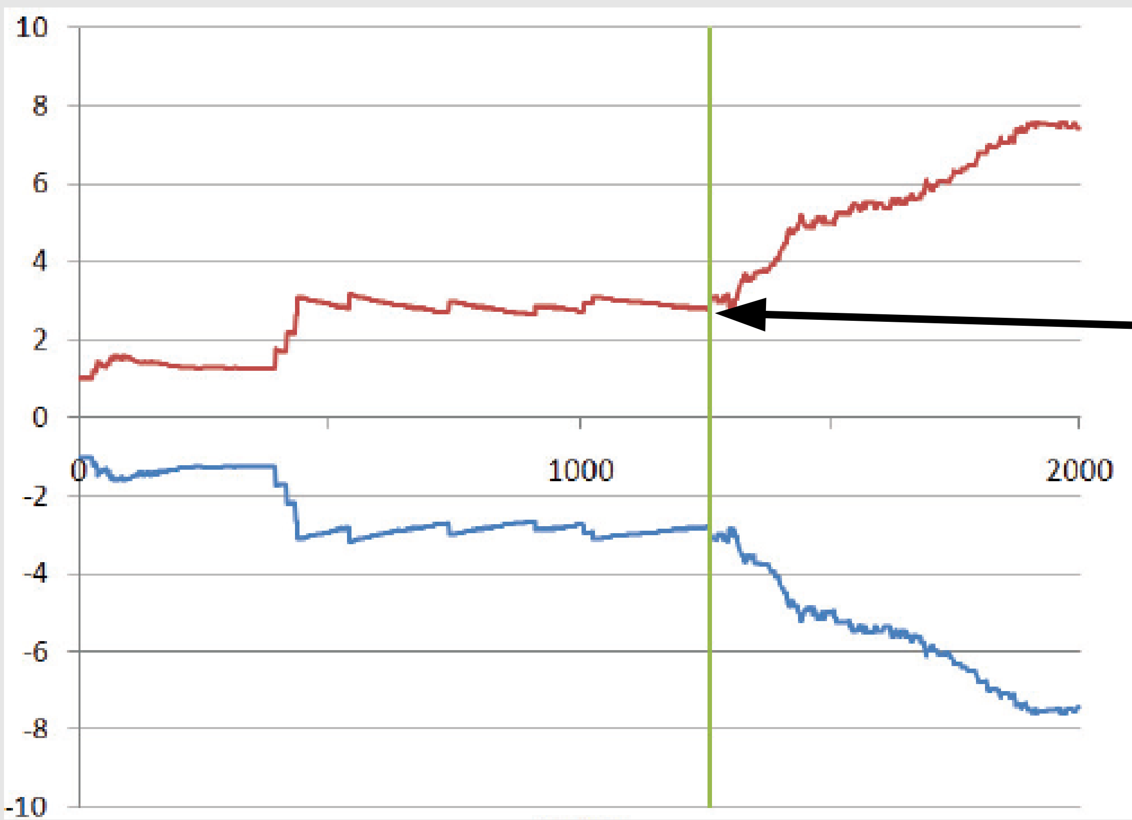
- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Online learning of opponent model

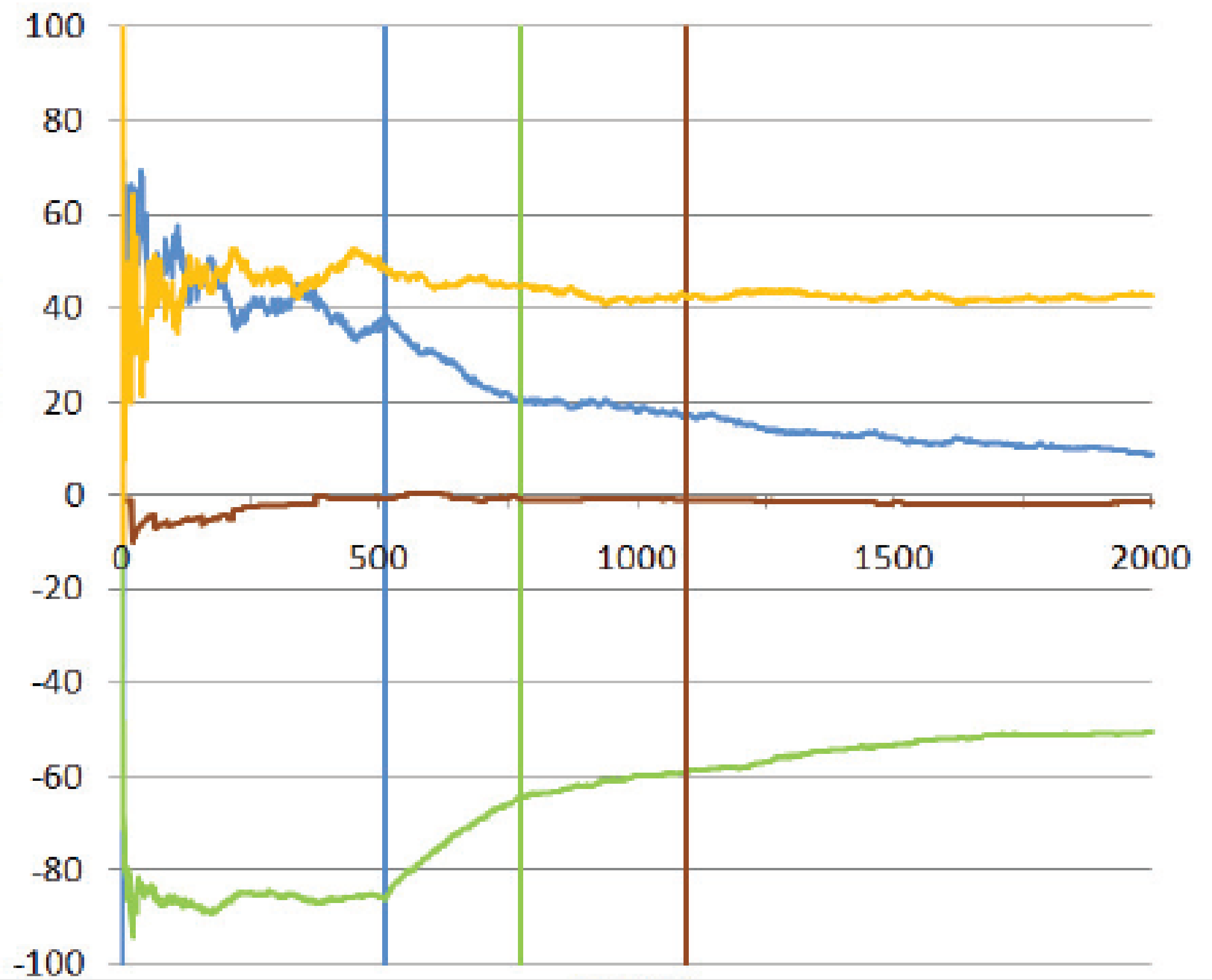
- Start from (safe) model of general opponent
- Exploit weaknesses of specific opponent



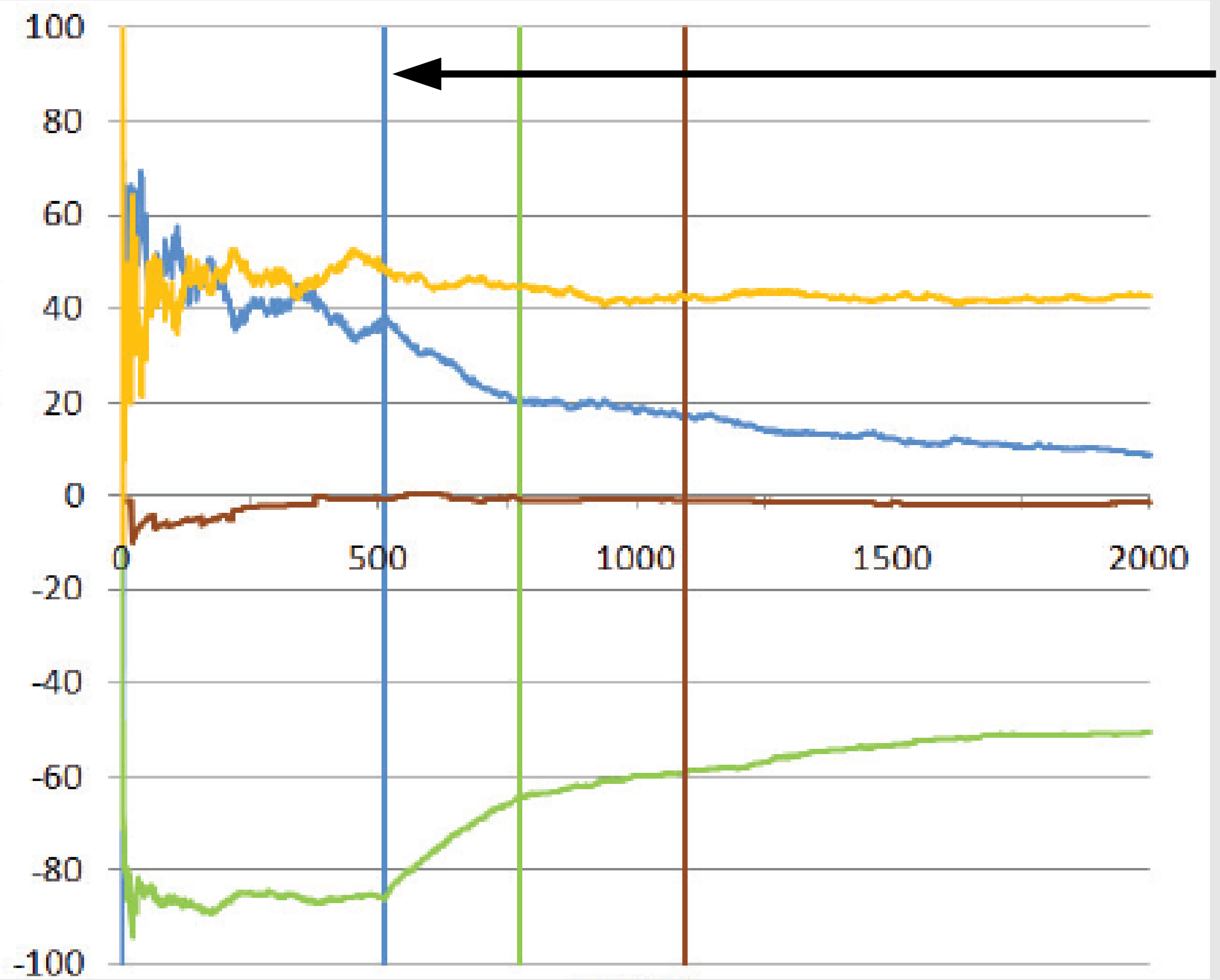
Start to learn model of specific opponent

(exploration of opponent behavior)

Multi-agent interaction

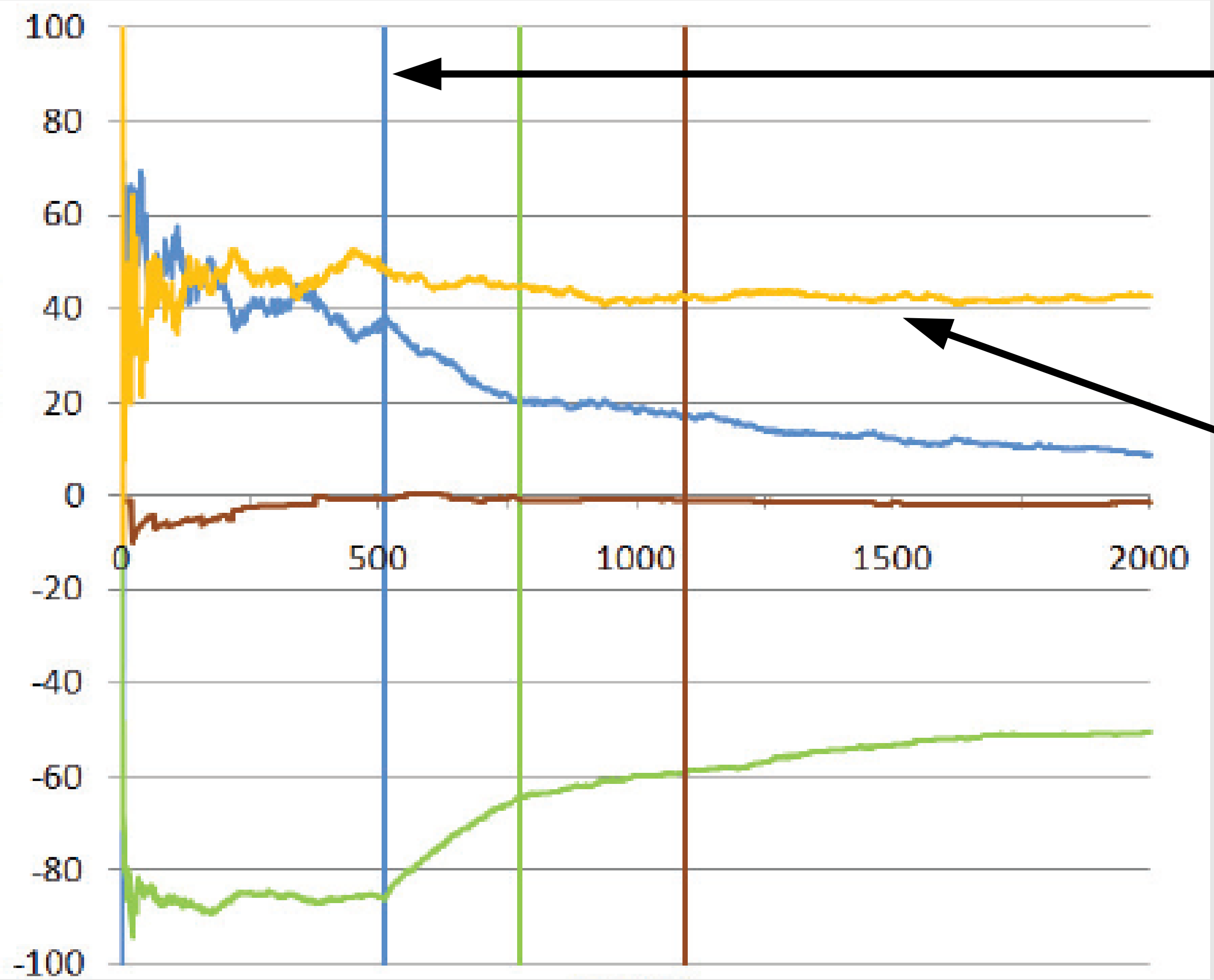


Multi-agent interaction



Yellow learns model for Blue and changes strategy

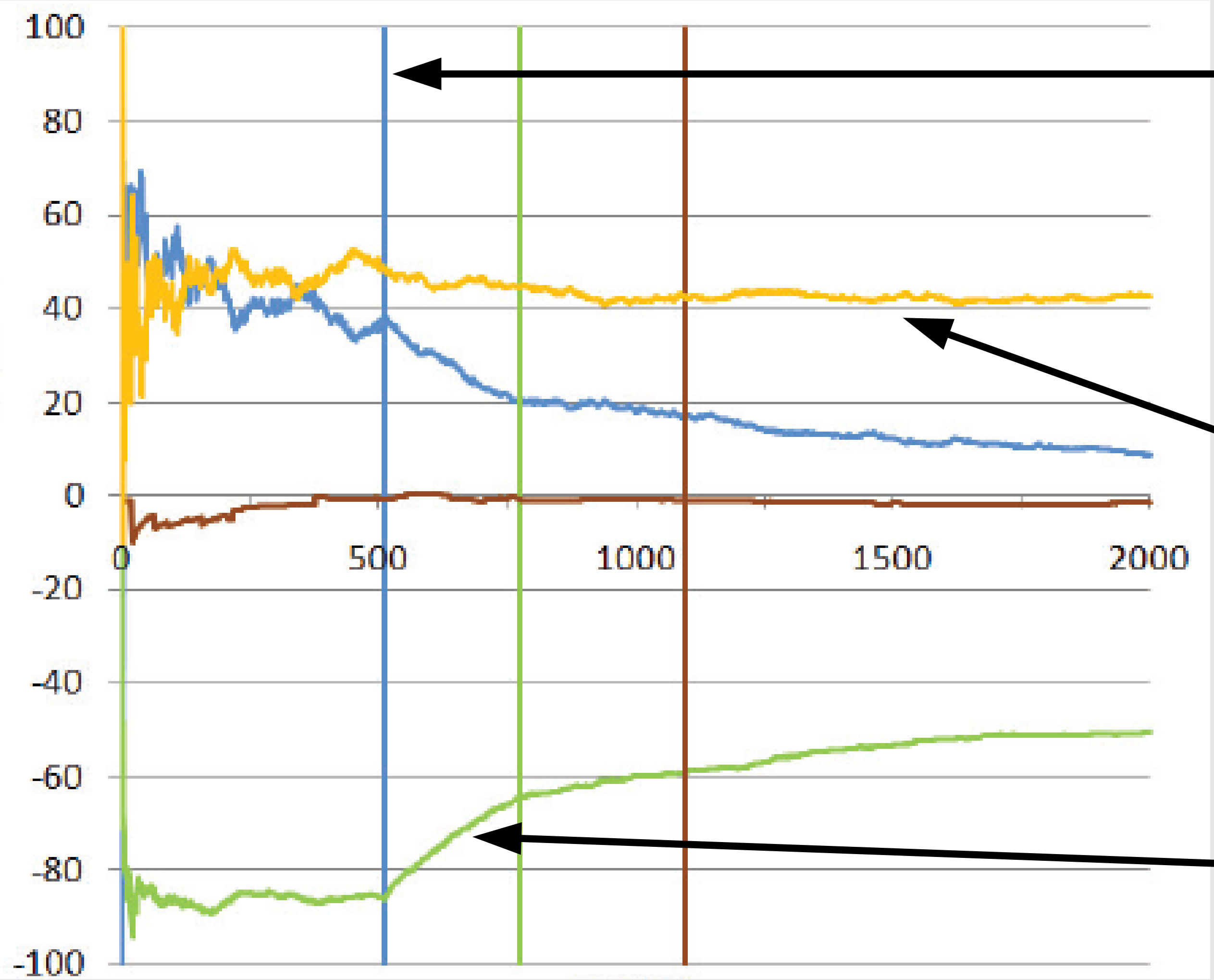
Multi-agent interaction



Yellow learns model for Blue and changes strategy

Yellow doesn't profit!

Multi-agent interaction



Yellow learns model for Blue and changes strategy

Yellow doesn't profit!

Green profits without changing strategy!!

Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
    - Uncertainty in MCTS
    - Continuous action spaces
  - Opponent model
    - Online learning
    - Concept drift
- Conclusion

Concept drift

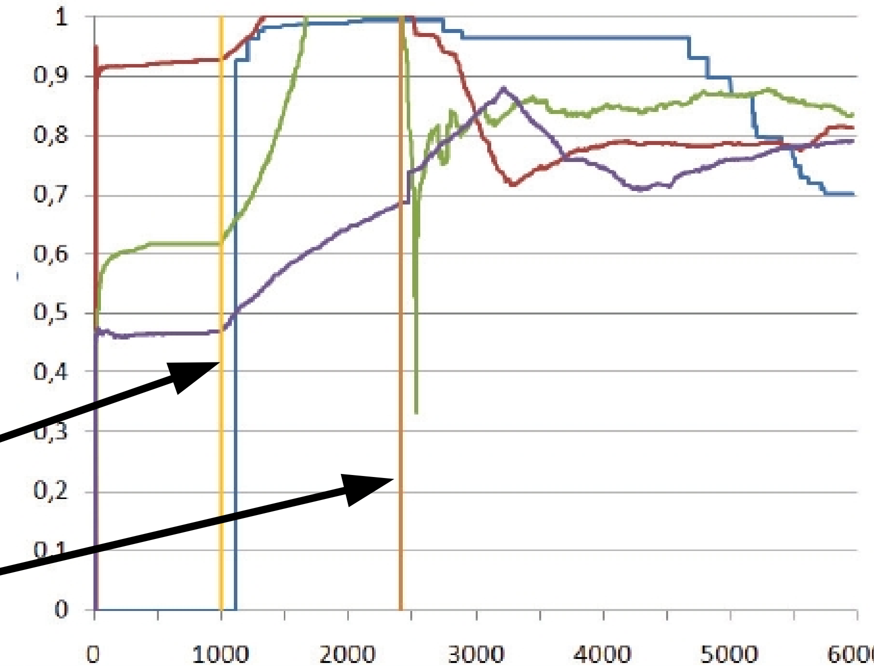
- While learning from a stream, the training examples in the stream change
  - In opponent model: changing strategy
- "***Changing gears*** is not just about bluffing, it's about changing strategy to achieve a goal."
- Learning with concept drift
  - ***adapt*** quickly to changes
  - yet ***robust*** to noise
  - (recognize recurrent concepts)

Basic approach to concept drift

- Maintain a window of training examples
  - large enough to learn
  - small enough to adapt quickly
  - without 'old' concepts
- Heuristics to adjust window size
  - based on FLORA2 framework [Widmer92]

4 components of a single opponent model

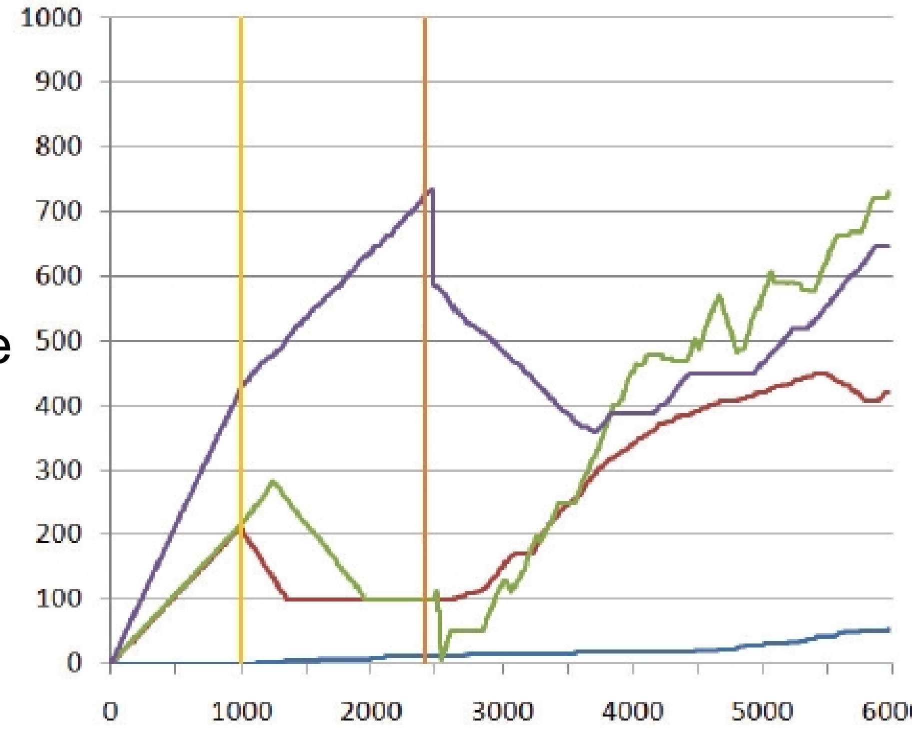
Accuracy



Start online learning

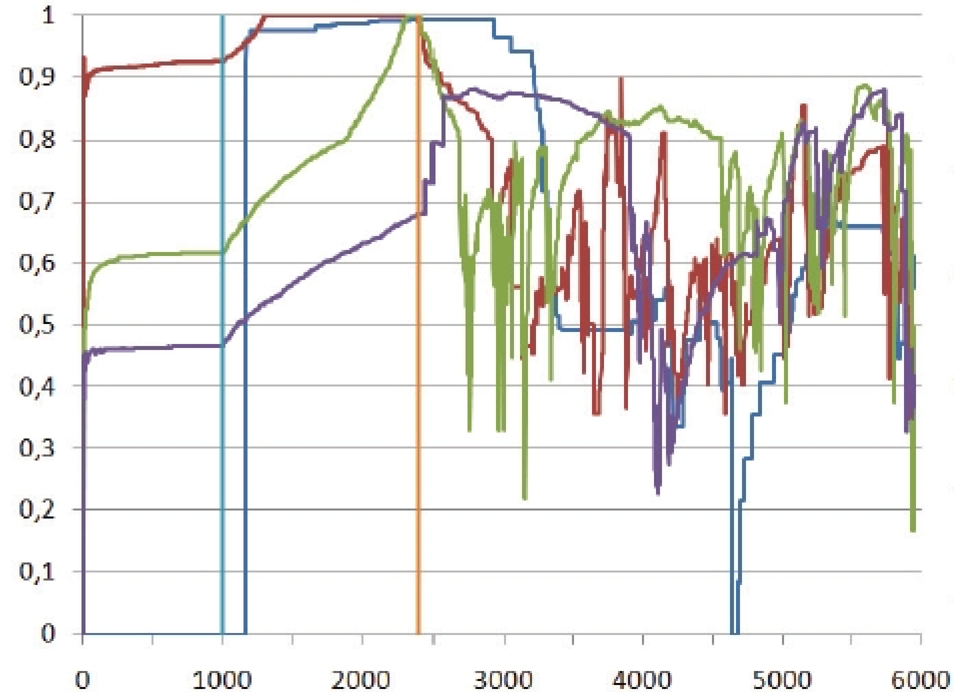
Concept drift

Window size

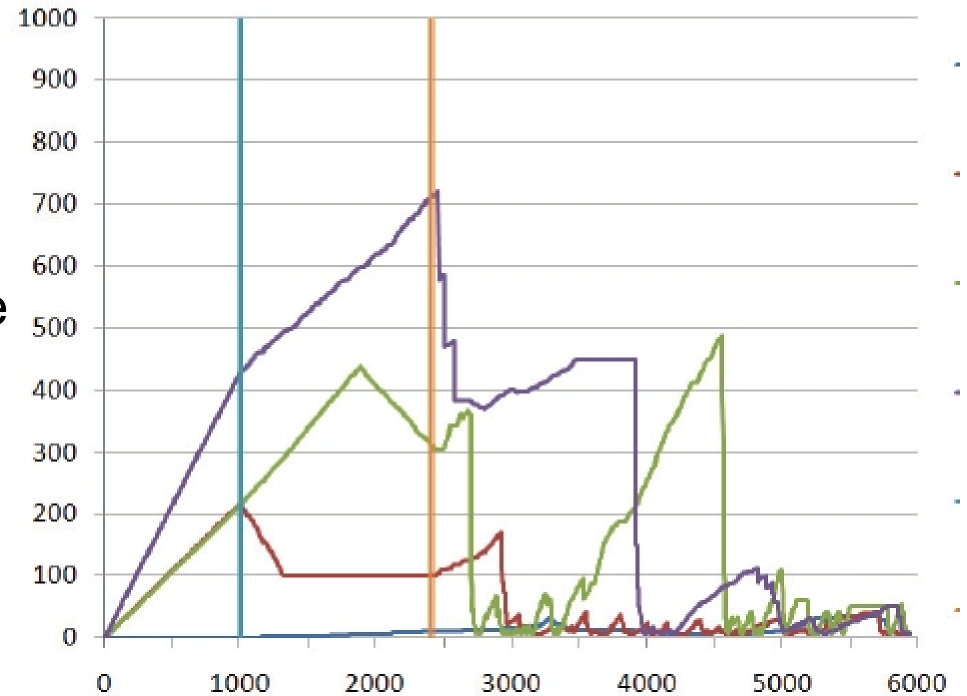


Bad parameters for heuristic

Accuracy



Window size



Outline

- Overview approach
  - The Poker game tree
  - Opponent model
  - Monte-Carlo tree search
- Research challenges
  - Search
  - Opponent model
- Conclusion

Conclusions

- First exploitive poker bot for
  - *No-limit* Holdem
  - > 2 players
- Apply in other games
  - backgammon
  - computational pool
  - ...
- Challenge for **MCTS**
  - games with uncertainty
  - continuous action space
- Challenge for **ML**
  - online learning
  - concept drift
  - (relational learning)

Thanks for listening!