Combining Deep Reinforcement Learning and Search for Imperfect-Information Games

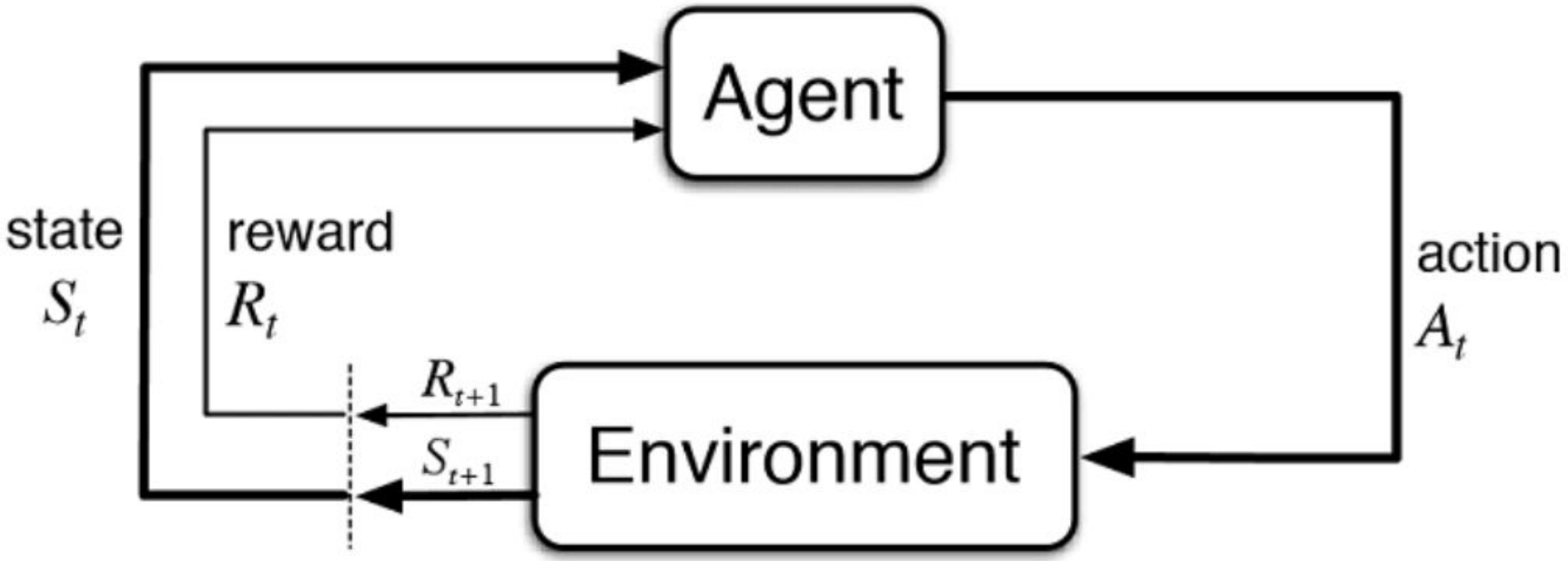
Noam Brown, Anton Bakhtin, Adam Lerer, Qucheng Gong

Facebook AI Research

https://arxiv.org/abs/2007.13544

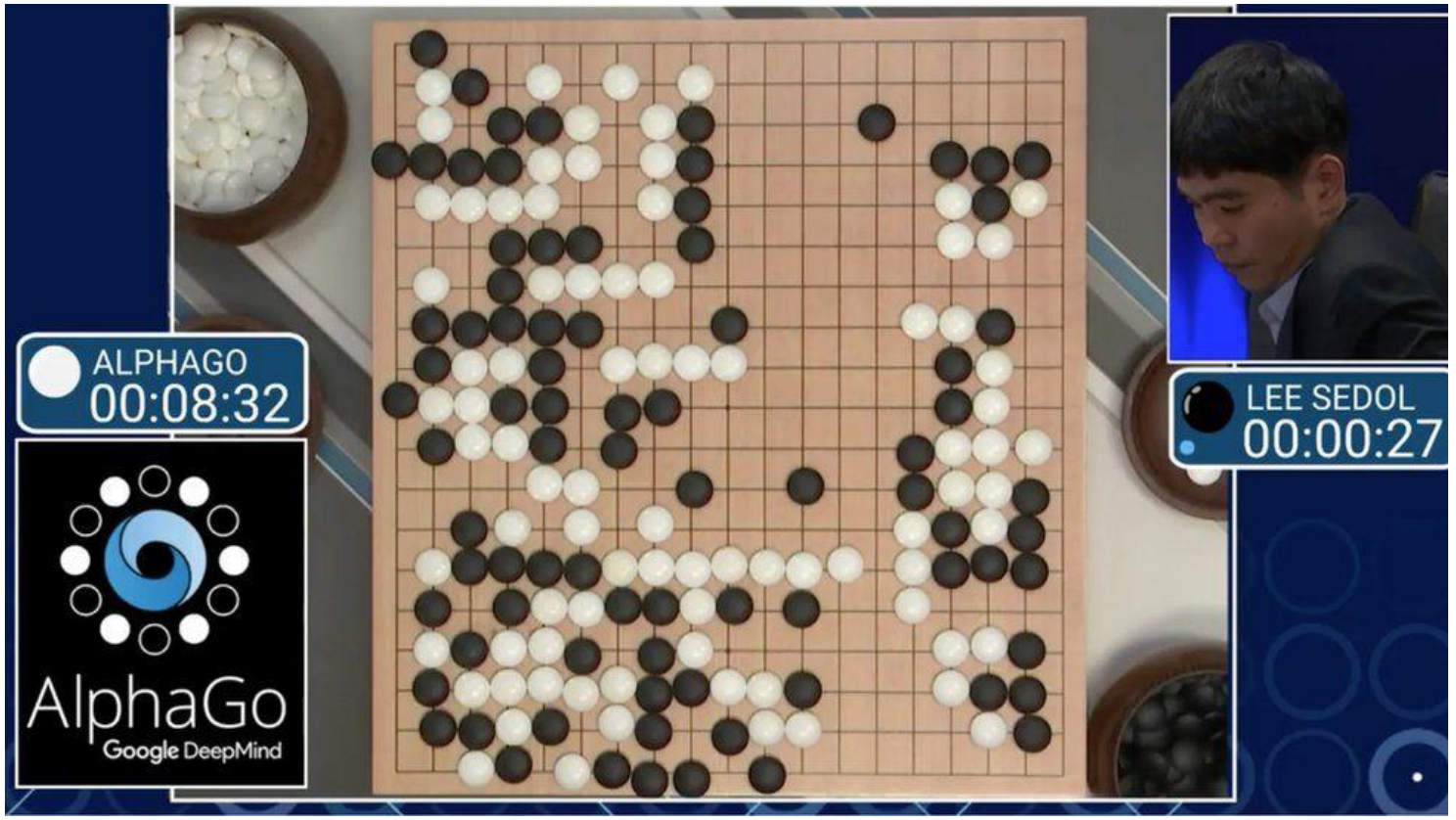
Neural Information Processing Systems (NeurIPS), 2020

Context : Reinforcement Learning



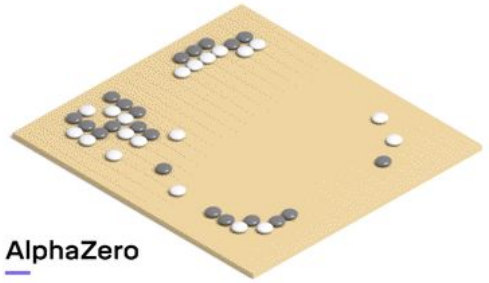
Context : AlphaGo

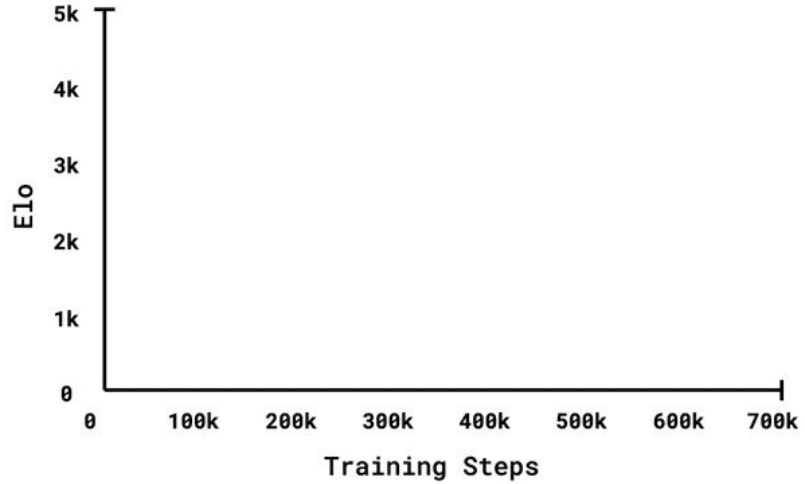
- First time an AI system beat humans in Go
- However, AlphaGo algorithm is:
  - Specific to the game of Go
  - Utilizing human data from expert players
  - Utilizing expert domain knowledge as features
- Therefore, it is difficult to apply this same technique on other domains



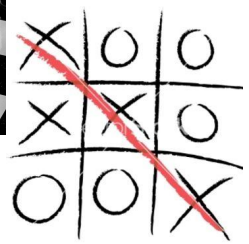
Context : AlphaZero

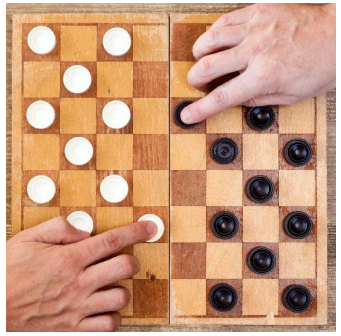
- One same algorithm that can perform at superhuman level for:
  - Chess
  - Go
  - Shogi
- AlphaZero algorithm is a general technique:
  - Uses no human data
  - Uses no expert domain knowledge as features
- However, it is limited to **perfect-information** games only





Context : Perfect-Information vs Imperfect-Information Games





Context : No-Limit Texas Hold'em Poker (NLHE)

- Poker is the classic benchmark challenge for solving imperfect-information games.
- NLHE is the most popular, most played and most widely studied variant of poker.

- 2017: First AI beat expert humans at 2p NLHE
  - DeepStack - University of Alberta in Edmonton
  - Libratus - Carnegie Mellon University
- 2019: First AI beat expert humans at 6p NLHE
  - Pluribus - Carnegie Mellon University and Facebook AI

- Techniques used in Poker AI are very different from techniques used in AlphaGo and AlphaZero

Premise of ReBeL

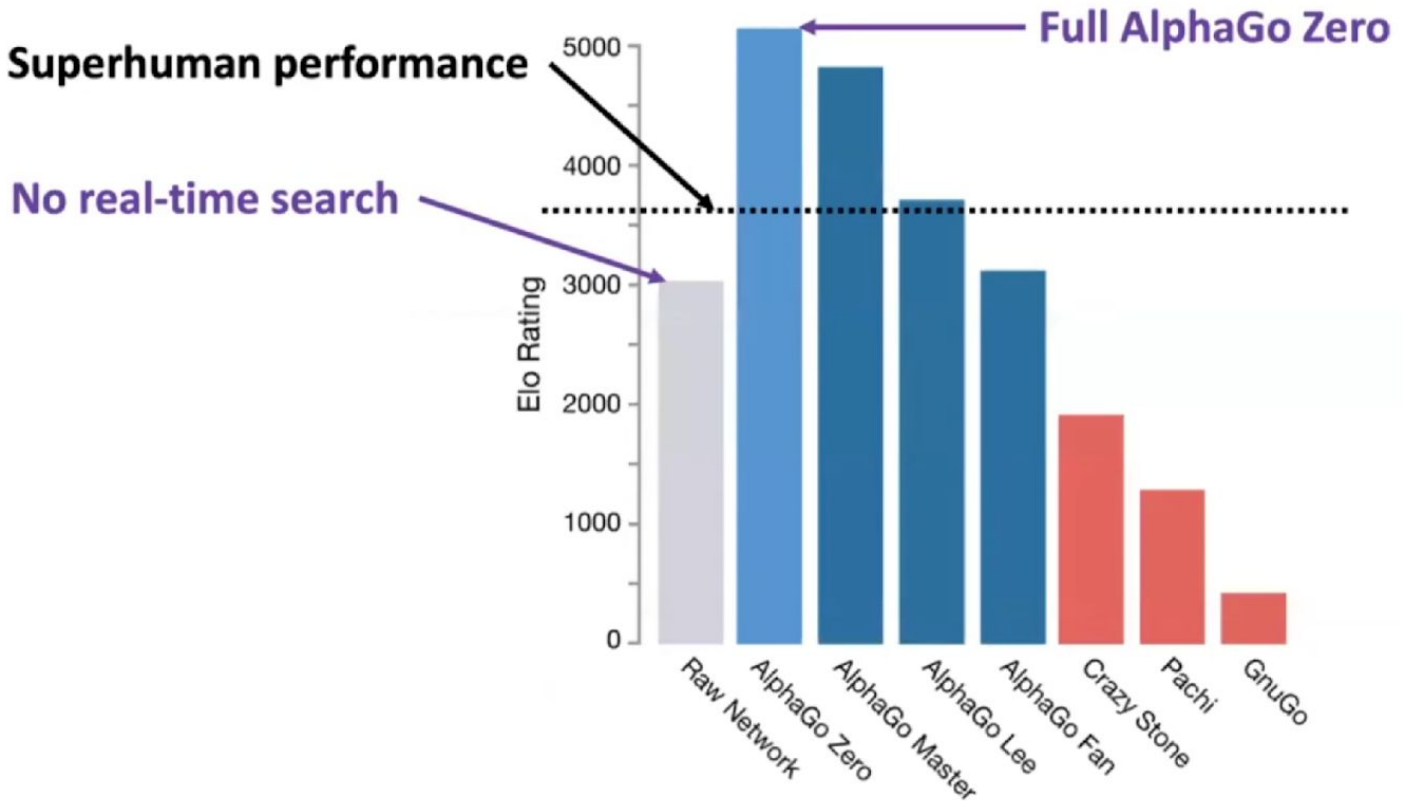
- Main goal: One single algorithm that can solve both perfect-information games and imperfect-information games.

- ReBeL: Recursive Belief-based Learning
  - Not yet achieve superhuman performance in both perfect-information games and imperfect-information games
  - Unify the 2 research domains of perfect and imperfect info game AI
  - Open a new path to future development of one single algorithm that can master both domains
  - When applied to imperfect-information games:
    - Ability to reach optimum strategy, converge to Nash equilibrium in two-player zero-sum games
    - Produce superhuman result in two-player NLHE
    - Use significantly less domain knowledge then prior poker bots
  - When applied to perfect-information games:
    - Reduce to an algorithm like AlphaZero

Challenge : Why can’t the same techniques be applied?

AlphaZero

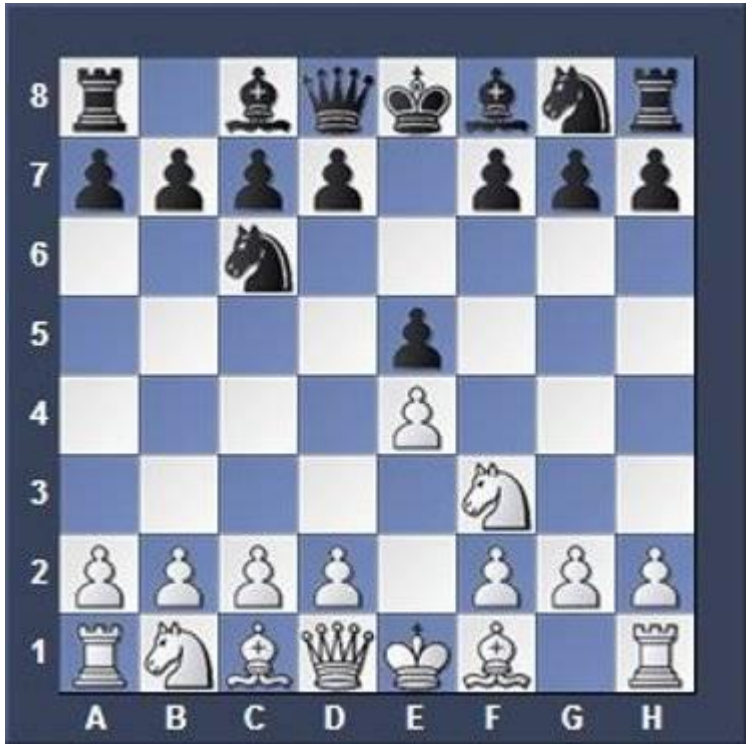
- Train a value network through self-play deep reinforcement learning
- At training time:
  - Use MCTS algorithm to find leaf nodes (states)
  - Values of states are determined via the value function
- RL+Search is critical to achieving superhuman performance
- No AI agent reached superhuman performance in Go without search at both training and test time.
- However, in imperfect-information games, RL+Search algorithms are:
  - not theoretically sound
  - not been shown to be successful



Challenge : Solving Perfect-Information games

Key concepts in perfect-information games AI:

- State: a configuration of the world
- Value of state: a **unique** value assuming all players playing optimally from that point forward
- Value network:
  - Input: a specific state
  - Output: an estimated value of given state



Methods of training value network:

- Handcrafted heuristic function with expert domain knowledge
  - Deep Blue: beat top humans at chess in 1997
- Train on data from expert human games
  - AlphaGo
- **Self-play reinforcement learning**
  - AlphaZero

Challenge : Solving Perfect-Information games

Why value function?

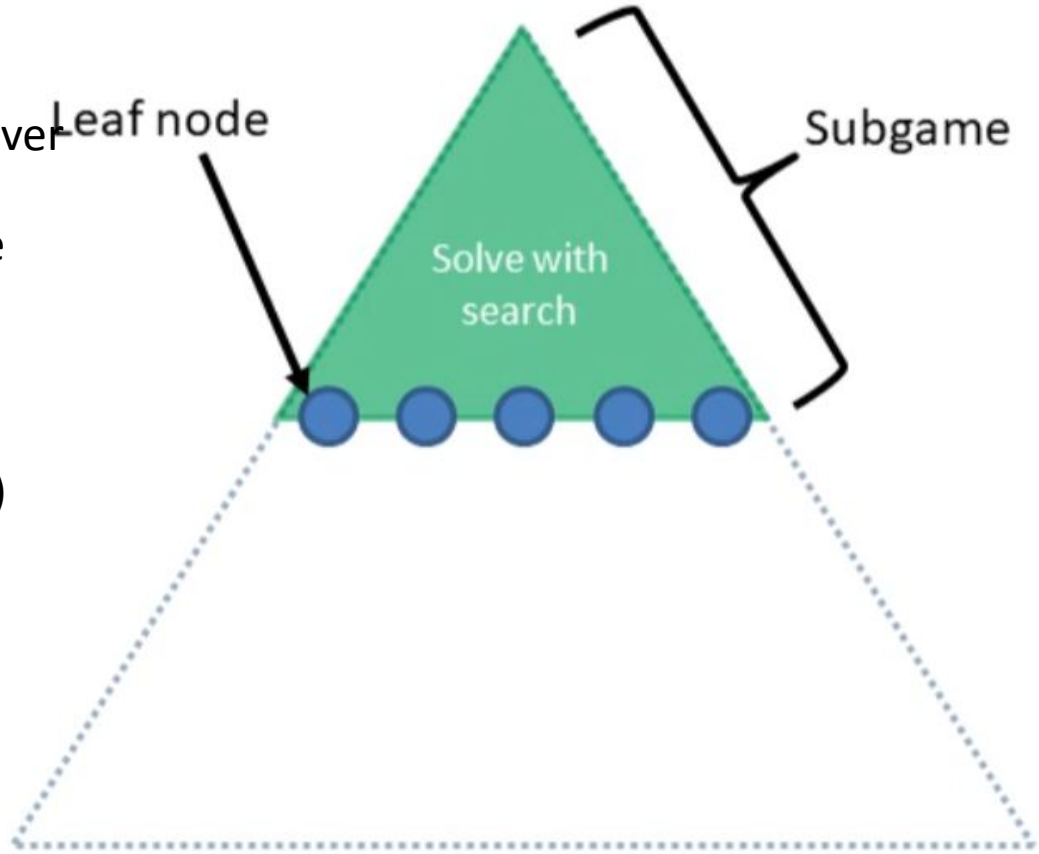
- Can be solved with simple techniques such as backward induction over the whole game (very resource intensive)
- Value function: help approximate optimal policy without solving the entire game

Solution: combine value function with search

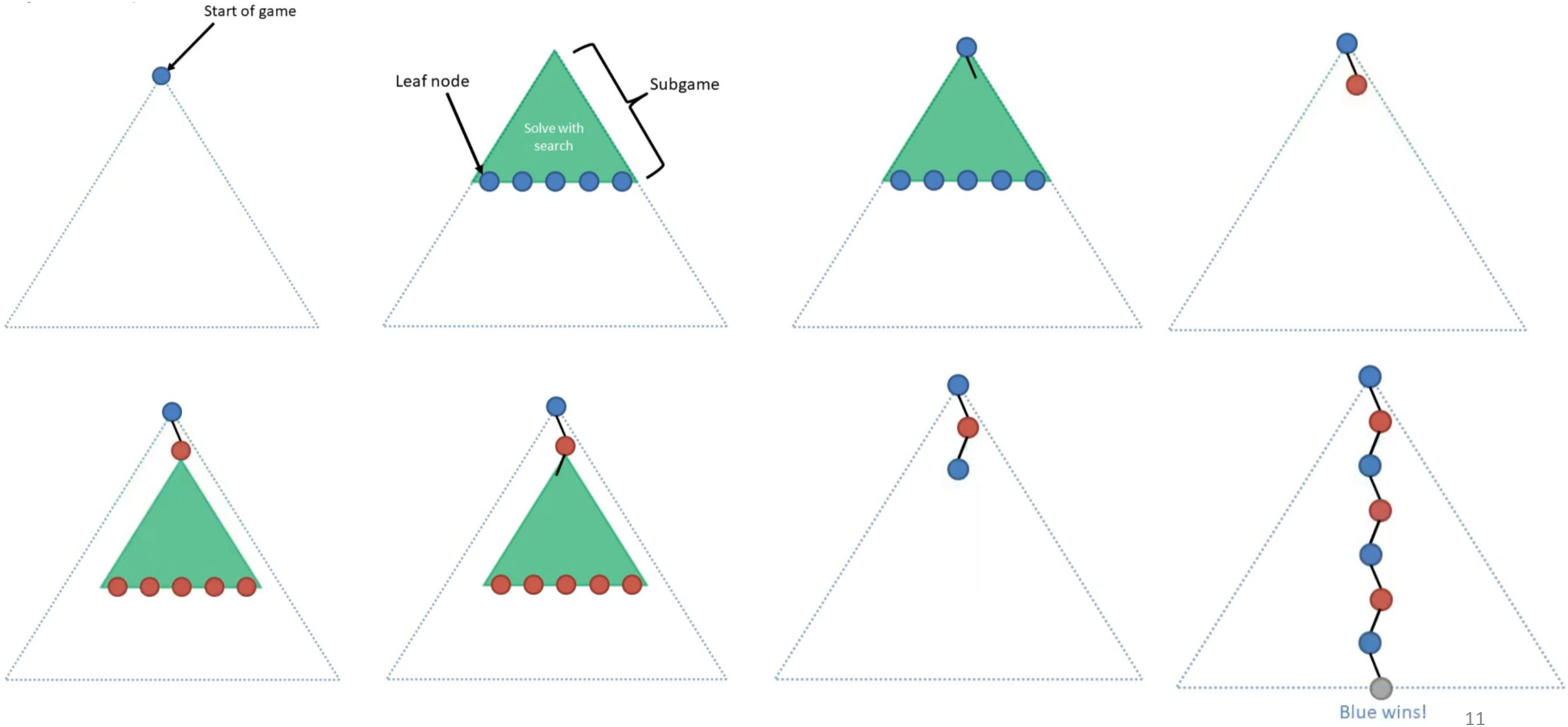
- Look a certain number of moves a head, discover leaf nodes (states)
- Estimate values of states with value network
- Perform backward induction with estimated state values
- Ignore states below leaf nodes

⇒ Solve a subgame

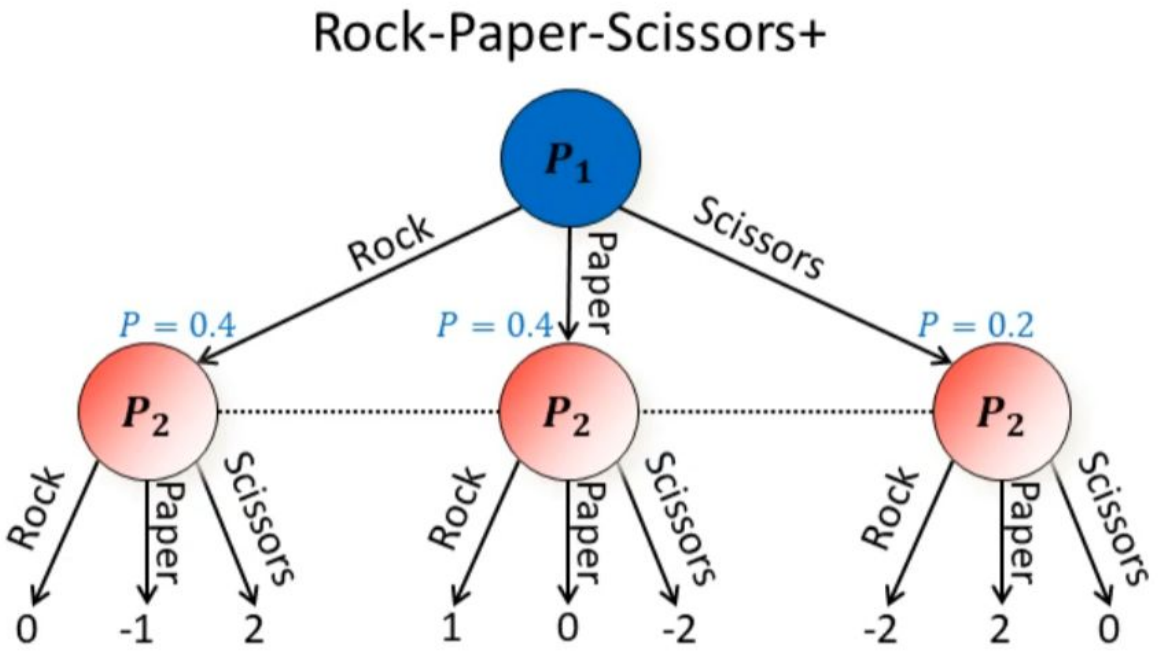
⇒ If perfect value network => optimal policy

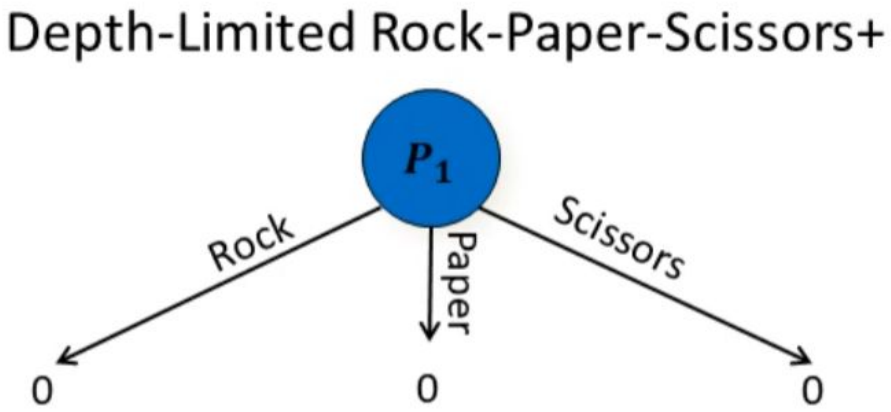


Challenge : Search in AlphaZero



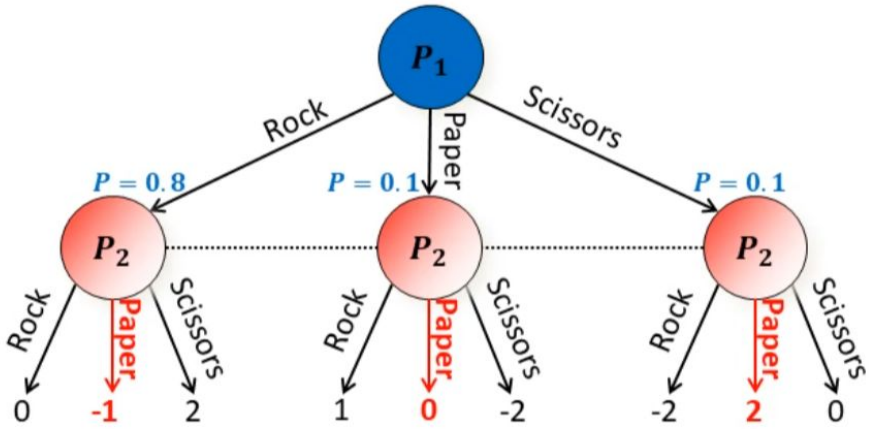
Challenge : Search in Imperfect-Information games



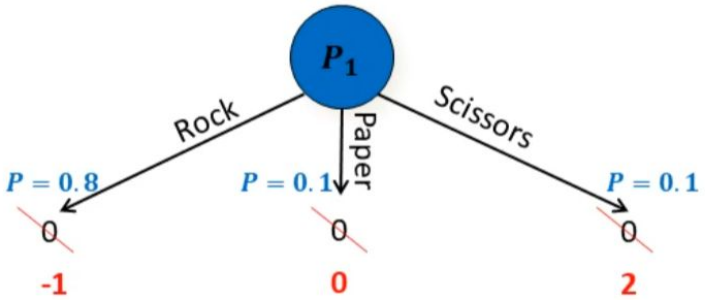


Challenge : Search in Imperfect-Information games

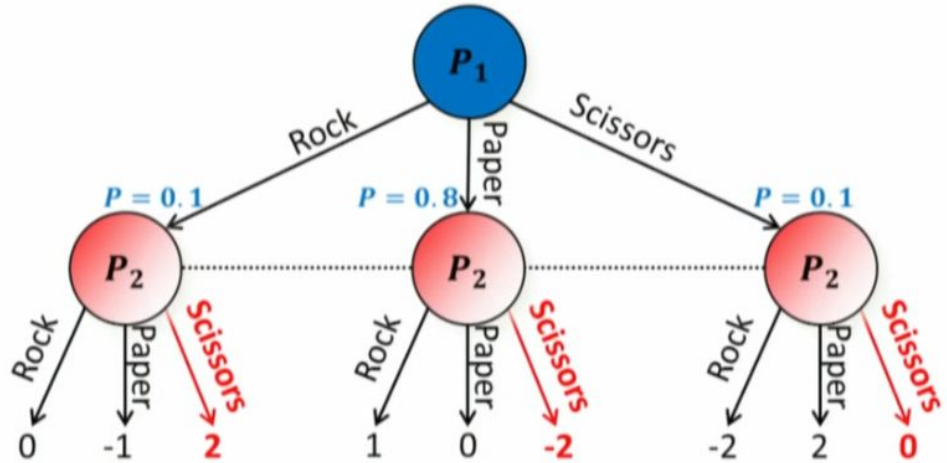
Rock-Paper-Scissors+



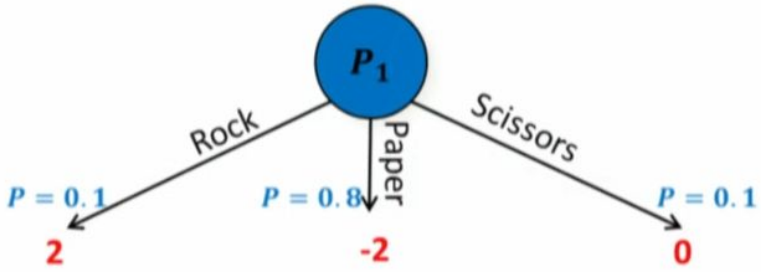
Depth-Limited Rock-Paper-Scissors+



Rock-Paper-Scissors+



Depth-Limited Rock-Paper-Scissors+



Proposed approach

- Apply RL+Search framework to imperfect information games
- Convert imperfect-information games to continuous-state perfect-information games
- Use an expanded notion of “state” as public belief state (PBS)
- PBSs: common-knowledge belief distribution over states, determined by the public observations shared by all agents and the policies of all agents
- Use a PBS value function during search
- Important assumptions:
  - Rules of the game and the agents’ policies (including search algorithms) are common knowledge.
  - Outcome of random processes (i.e., the random seeds) are not common knowledge.

Terminologies

- World state $w \in W$
- Action space A
- Transition function $\mathrm{T}(\mathrm{w}, \mathrm{a}) \in W$
- Reward function $R_i(w, a) \in W$
- Private and public observations $O_{priv(i)}(w, a, w')$ $O_{pub}(w, a, w')$
- History $\mathrm{h} = \{w^0, a^0, w^1, a^1, \ldots, w^t\}$
- **Info state** (action-observation history) $s_i = \{O_i^0, a_i^0, O_i^1, a_i^1, \ldots, O_i^t\}$
- Public state $s_{pub} = \{O_{pub}^0, O_{pub}^1, \ldots, O_{pub}^t\}$
- Policy and policy profile π_i $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$
- Expected value $v_i^{\pi}(h)$ $v_i(\pi)$
- Nash equilibrium policy profile $v_i(\pi^*) = max_{\pi_i} v_i(\pi_i, \pi_{-i}^*)$
- Subgame and depth-limited subgame

Proposed approach

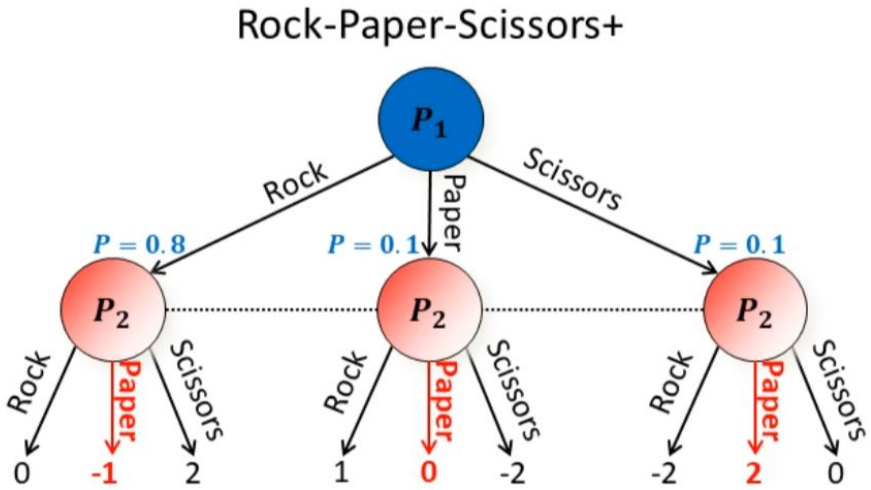
Problem 1: Impossible to use world state, must use info state instead.

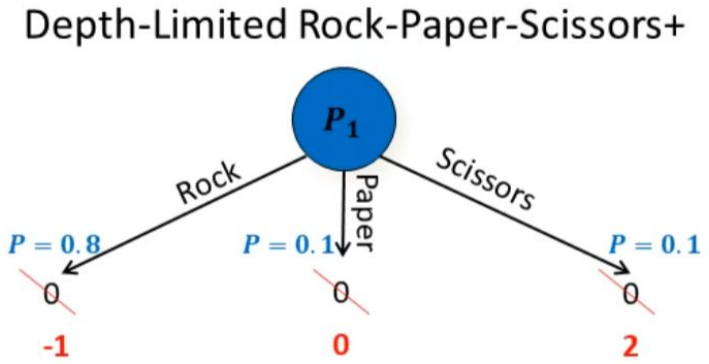
Problem 2: When using info state, there is not enough information for value network to calculate optimal strategy

Solution: change definition of state so that value of state is well defined

v(Rock) => not well-defined

v([0.8 Rock, 0.1 Paper, 0.1 Scissors]) = 0.8*(-1) + 0.1*0 + 0.1*2 = -0.6 => well-defined





Convert imperfect-information games to perfect-information games

I bet

If I have 2, I fold with 100% prob
If I have 3, I bet with 30% prob
…
If I have A, I bet with 100% prob

Convert imperfect-information games to perfect-information games

I bet

If I have 2, I fold with 100% prob
If I have 3, I bet with 30% prob
…
If I have A, I bet with 100% prob

P1 bets

Convert imperfect-information games to perfect-information games

I bet

Discrete representation

I don’t have 2

If I have 2, I fold with 100% prob
If I have 3, I bet with 30% prob
…
If I have A, I bet with 100% prob

P1 bets

He doesn’t have 2

Belief representation

Convert imperfect-information games to perfect-information games

$w(2) = \frac{1}{13}$ $w(3) = \frac{1}{13}$ … $w(K) = \frac{1}{13}$ $w(A) = \frac{1}{13}$




If I have 2, I fold with 100% prob
If I have 3, I bet with 30% prob
…
If I have A, I bet with 100% prob

$w(2) = \frac{1}{13}$ $w(3) = \frac{1}{13}$ … $w(K) = \frac{1}{13}$ $w(A) = \frac{1}{13}$




Convert imperfect-information games to perfect-information games

$w(2) = 0$ $w(3) = \frac{1}{12}$ $w(K) = \frac{1}{12}$ $w(A) = \frac{1}{12}$



…

If I have 2, I fold with 100% prob
If I have 3, I bet with 30% prob
…
If I have A, I bet with 100% prob

P1 bets

$w(2) = 0$ $w(3) = \frac{1}{12}$ $w(K) = \frac{1}{12}$ $w(A) = \frac{1}{12}$



…

Convert imperfect-information games to perfect-information games

$w(2) = 0$ $w(3) = \frac{1}{12}$ … $w(K) = \frac{1}{12}$ $w(A) = \frac{1}{12}$

$w(2) = 0$ $w(3) = \frac{1}{12}$ … $w(K) = \frac{1}{12}$ $w(A) = \frac{1}{12}$

Public Belief State (PBS)

A joint probability distribution over the agents' possible infostates

Convert imperfect-information games to perfect-information games

- Finding: Any imperfect information game can be viewed as a high dimensional continuous perfect information game.
- Question: Is it possible to use an algorithm like AlphaZero on the belief representation of the game, since it is a perfect information game now?
- Answer: Theoretically yes, but due to continuity and high dimensionality of state/action space, it is not tractable.
- Solution: ReBeL

ReBeL

Important notes on Public Belief States

- PBSs are identical to perfect-information states in perfect information games
- PBSs always have **unique** value in 2p zero-sum games
  - Possible to use value function
- For imperfect-information games action/state space is continuous and high-dimensional
  - Traditional search methods such as MCTS become impractical
- Fortunately, action space is a **convex optimization problem** (for 2p0s)
  - Algorithms similar to gradient-descent can be used to efficiently solve subgames to find optimal policy
  - Common algorithms: Fictitious Play (FP) or Counterfactual Regret Minimization (CFR)

ReBeL

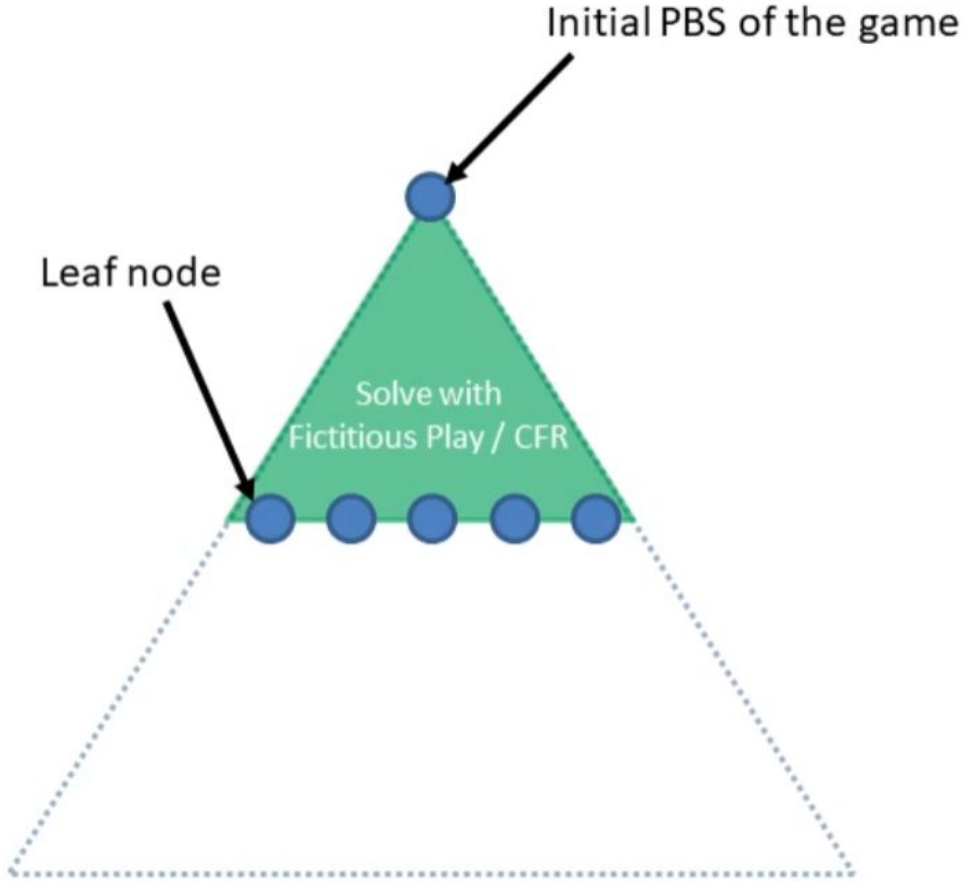
- Reminder: a PBS is a joint probability distribution over the agents' possible infostates

- ReBeL's search algorithm operates on supergradients of the PBS value function at leaf nodes, **not** on PBS values directly
- In 2p0s games, supergradient of the PBS value function can be calculated with infostate values.
- I

$$v_i^{\pi^*}(s_i|\beta) = \max_{\pi_i} \sum_{h \in \mathcal{H}(s_i)} p(h|s_i, \beta_{-i}) v_i^{\langle \pi_i, \pi^*_{-i} \rangle}(h)$$

- Instead of learning a PBS value function, ReBeL learns an infostate-

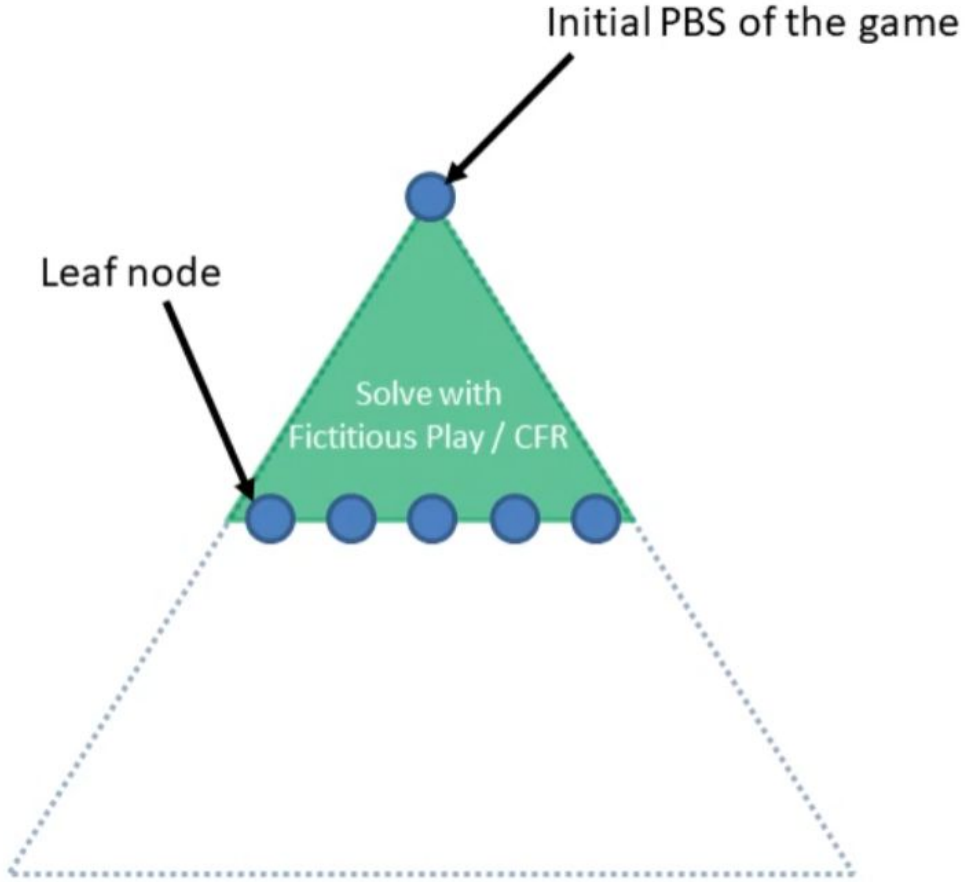
$$\hat{v} : \mathcal{B} \rightarrow \mathbb{R}^{|S_1|+|S_2|}$$



ReBeL

Self Play RL and search for PBSs:

- Generate a depth-limited subgame rooted at the initial PBS
- Subgame is solved by running T iterations of an iterative equilibrium-finding algorithm in the discrete representation of the game
- Use the learned value network to approximate leaf values on every iteration
- During training:
  - the infostate values at root PBS computed during search are added as training examples for value network
  - the subgame policies are added as training examples for the policy network (optional)
- Next, a leaf node is sampled, and the process repeats with the PBS at sampled leaf node being the new subgame root

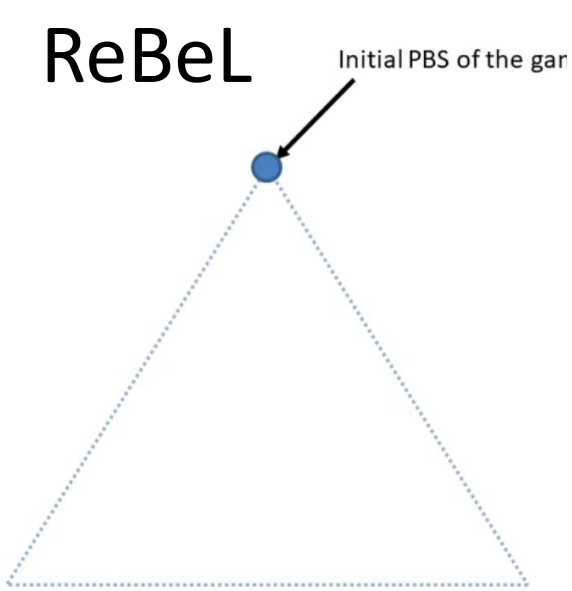


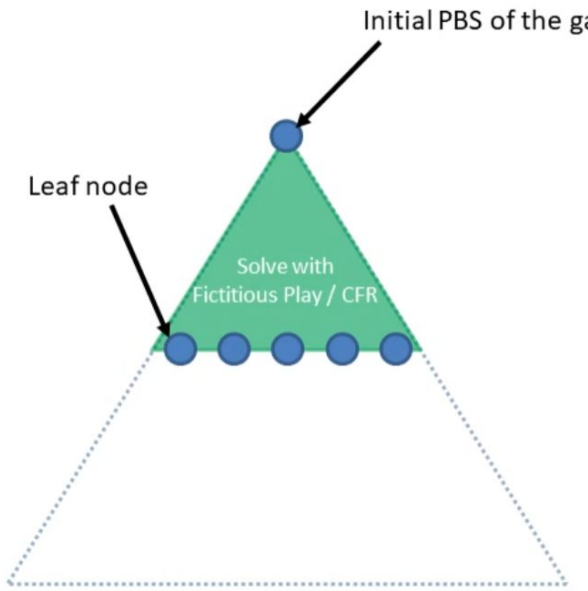
ReBeL

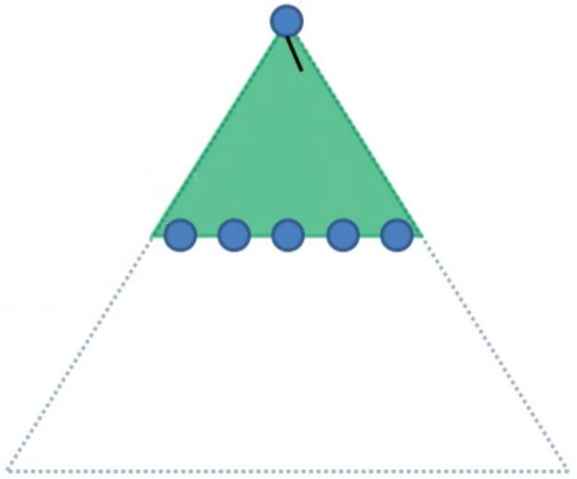
Algorithm 1 ReBeL: RL and Search for Imperfect-Information Games

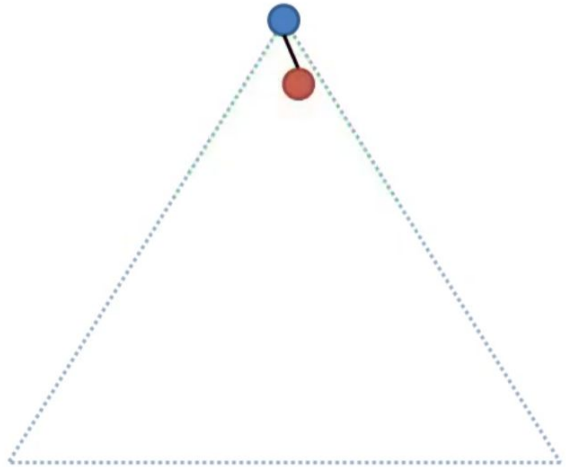
function SELFPLAY($\beta_r, \theta^v, \theta^\pi, D^v, D^\pi$) ▷ β_r is the current PBS
  while !ISTERMINAL(β_r) **do**
    $G \leftarrow$ CONSTRUCTSUBGAME(β_r)
    $\bar{\pi}, \pi^{t_{\text{warm}}} \leftarrow$ INITIALIZEPOLICY(G, θ^π) ▷ $t_{\text{warm}} = 0$ and π^0 is uniform if no warm start
    $G \leftarrow$ SETLEAFVALUES($G, \bar{\pi}, \pi^{t_{\text{warm}}}, \theta^v$)
    $v(\beta_r) \leftarrow$ COMPUTEEV($G, \pi^{t_{\text{warm}}}$)
    $t_{sample} \sim \text{unif}\{t_{\text{warm}} + 1, T\}$ ▷ Sample an iteration
    for $t = (t_{\text{warm}} + 1)..T$ **do**
      if $t = t_{sample}$ **then**
        $\beta'_r \leftarrow$ SAMPLELEAF(G, π^{t-1}) ▷ Sample one or multiple leaf PBSs
      $\pi^t \leftarrow$ UPDATEPOLICY(G, π^{t-1})
      $\bar{\pi} \leftarrow \frac{t}{t+1}\bar{\pi} + \frac{1}{t+1}\pi^t$
      $G \leftarrow$ SETLEAFVALUES($G, \bar{\pi}, \pi^t, \theta^v$)
      $v(\beta_r) \leftarrow \frac{t}{t+1}v(\beta_r) + \frac{1}{t+1}$ COMPUTEEV(G, π^t)
    Add $\{\beta_r, v(\beta_r)\}$ to D^v ▷ Add to value net training data
    for $\beta \in G$ **do** ▷ Loop over the PBS at every public state in G
      Add $\{\beta, \bar{\pi}(\beta)\}$ to D^π ▷ Add to policy net training data (optional)
    $\beta_r \leftarrow \beta'_r$

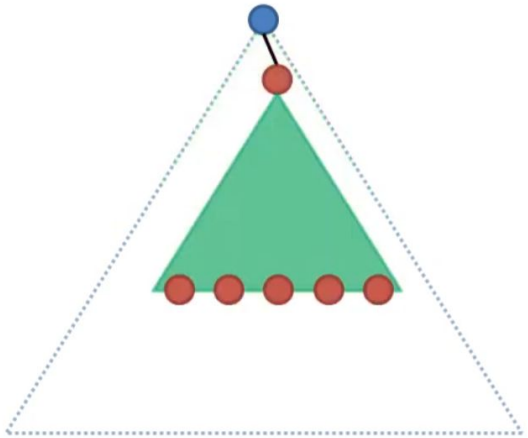
ReBeL

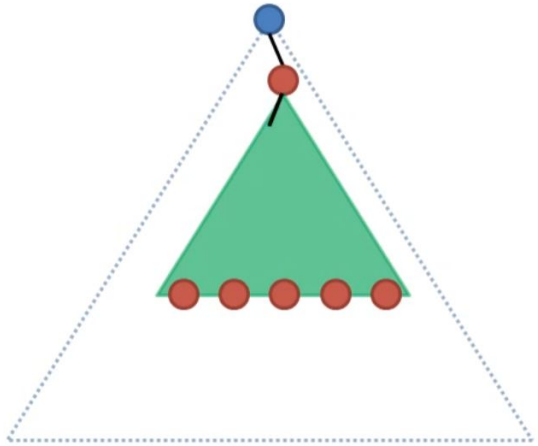

Initial PBS of the gar

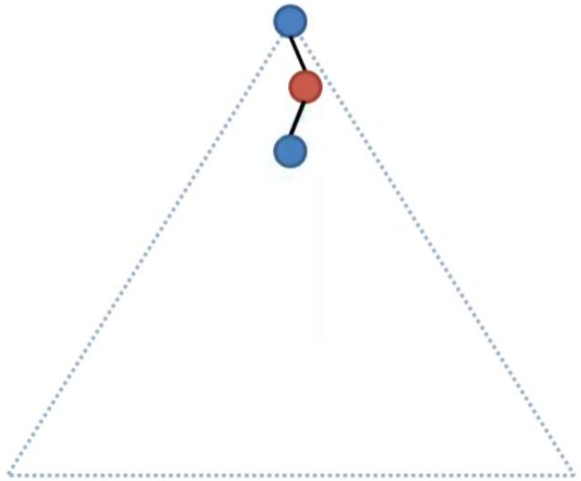

Initial PBS of the game
Leaf node
Solve with
Fictitious Play / CFR

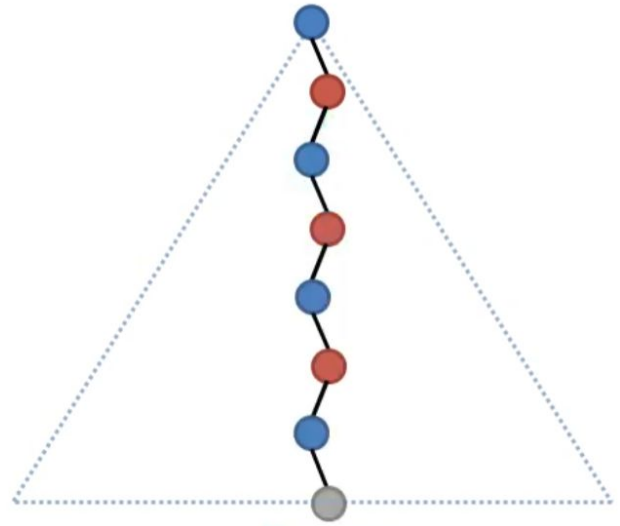











Blue wins!

ReBeL: Test time strategy

To play Nash equilibrium without being exploited easily at test time:

- Stop CFR on a random iteration and assume beliefs from this iteration
- Opponent will not know ReBeL's beliefs, and therefore cannot predict its policy
- ReBeL's subgame policy will be a Nash equilibrium in expectation
- This is the exact same algorithm used in training

Experiment setup

Benchmark games:

- 2p NLHE
- Liar's Dice
- Turn endgame Hold'em (THE)
  - Reduce action space to maximum 9 actions
  - Bet and stack size are randomized during training

Value and policy networks:

- Multiplayer perceptron
- GeLU activation function
- LayerNorm
- Adam optimizer
- Pointwise Huber loss (value network)
- Mean squared error loss (policy network)

Environment

- PyTorch
- One single machine for training
- 128 machines with 8 GPUs each for data generation

Experimental Results in THE

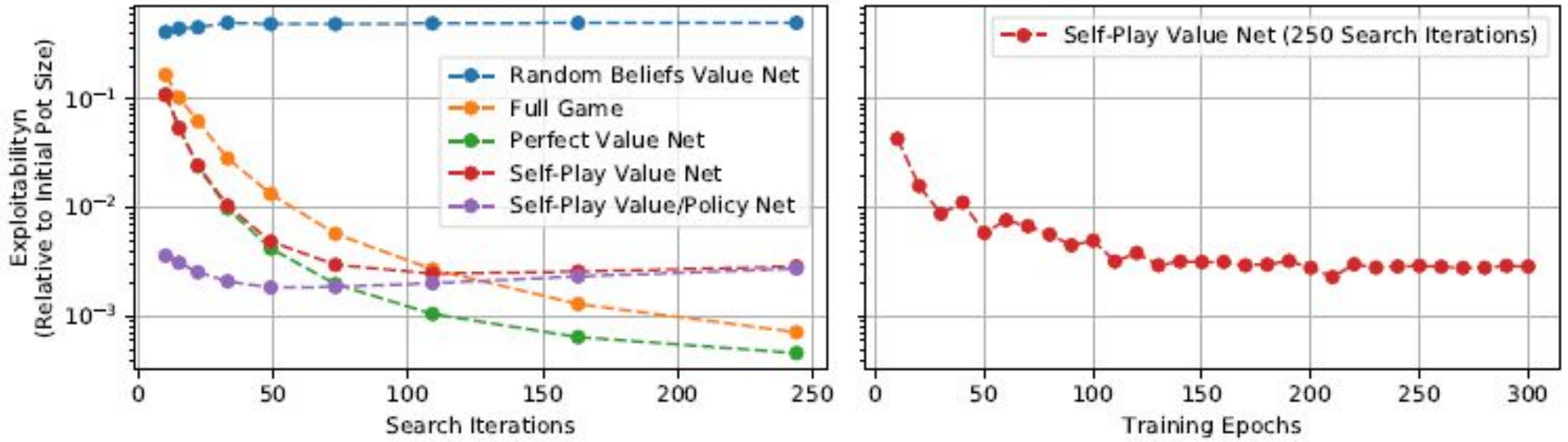


Figure 2: Convergence of different techniques in TEH. All subgames are solved using CFR-AVG. Perfect Value Net uses an oracle function to return the exact value of leaf nodes on each iteration. Self-Play Value Net uses a value function trained through self play. Self-Play Value/Policy Net additionally uses a policy network to warm start CFR. Random Beliefs trains the value net by sampling PBSs at random.

Experimental Results in 2p NLHE

Bot Name	Slumbot	BabyTartanian8 [9]	LBR [39]	Top Humans
DeepStack [40]	-	-	383 ± 112	-
Libratus [12]	-	63 ± 14	-	147 ± 39
Modicum [15]	11 ± 5	6 ± 3	-	-
ReBeL *(Ours)*	45 ± 5	9 ± 4	881 ± 94	165 ± 69

Table 1: Head-to-head results of our agent against benchmark bots BabyTartanian8 and Slumbot, as well as top human expert Dong Kim, measured in thousandths of a big blind per game. We also show performance against LBR [39] where the LBR agent must call for the first two betting rounds, and can either fold, call, bet $1\times$ pot, or bet all-in on the last two rounds. The $\pm$ shows one standard deviation. For Libratus, we list the score against all top humans in aggregate; Libratus beat Dong Kim by 29 with an estimated $\pm$ of 78.

Experimental Results in Liar's Dice

Algorithm	1x4f	1x5f	1x6f	2x3f
Full-game FP	0.012	0.024	0.039	0.057
Full-game CFR	0.001	0.001	0.002	0.002
ReBeL FP	0.041	0.020	0.040	0.020
ReBeL CFR-D	0.017	0.015	0.024	0.017

Table 2: Exploitability of different algorithms of 4 variants of Liar's Dice: 1 die with 4, 5, or 6 faces and 2 dice with 3 faces. The top two rows represent baseline numbers when a tabular version of the algorithms is run on the entire game for 1,024 iterations. The bottom 2 lines show the performance of ReBeL operating on subgames of depth 2 with 1,024 search iterations. For exploitability computation of the bottom two rows, we averaged the policies of 1,024 playthroughs and thus the numbers are upper bounds on exploitability.

Conclusions and Broader Impact

- ReBeL: a major step toward developing universal techniques for multi-agent interactions
  - Generalizes the paradigm of self-play RL and search to imperfect-information games
  - Converges to a Nash equilibrium in 2p0s games

- Limitations:
  - The input to value and policy functions grows linearly with the number of infostates in a public state (intractable in games with strategic depth and little common knowledge)
  - Theoretical proves only limited to 2p0s games

- Broader impact:
  - Potential future applications in auctions, negotiations, cybersecurity, and autonomous vehicle navigation (imperfect-information multi-agent interactions)
  - Potential risk if used for cheating in recreational games such as poker

Useful resources

- ReBeL implementation for Liar's Dice here
- DeepMind blog post on AlphaZero here
- DeepMind Reinforcement Learning Course here
- A paper on Deep CFR here
- Pluribus article and demo here
- A paper on Poker and Game Theory here