UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Game Theory and Multi-Agent Learning: A Survey

11/30/2023

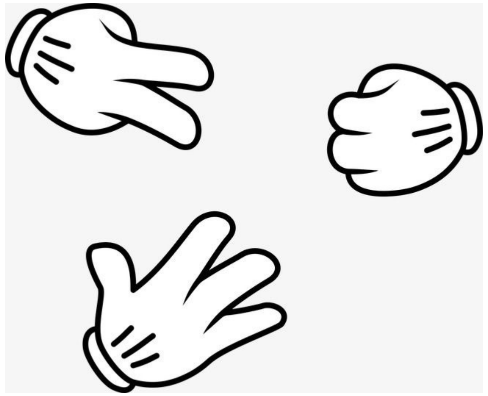
Reinforcement Learning, Multi-Agent, and Game Theory

1. Reinforcement Learning is a natural adaption & extension of behavioral psychology in machine learning: by interacting with the environment and receive rewards, the agent learns to achieve some goals;

2. Multi-agent game is a universal scenario setup: multiple agents act in the same environment, and their actions interacts with each other;

3. Where there is a multi-agent game, there is game theory;

4. If we want to use reinforcement learning algos to teach agents to play the multi-agent game, we need to incorporate game theory to the algorithm;



Markov Decision Process (MDP)

Setup: (S, A, T, R): states, actions, transition function $S \times A \times S \to [0,1]$, rewards function $S \times A \to \mathbb{R}$

Problem Statement: solving MDP means to find a policy function $\pi: S \to A$ that satisfies a specific goal

Under a given policy π, we can measure the value of taking a specific action on a specific state, as well as the value of a specific state:

(1)

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{t\geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \middle| a_0 = a, s_0 = s\right], \forall s \in \mathbb{S}, a \in \mathbb{A}$$

(2)

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t\geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \middle| s_0 = s\right], \forall s \in \mathbb{S}$$

Stochastic Game (SG)

Setup: $(n, S, A_{1,\ldots,n}, T, R_{1,\ldots,n})$: states, **joint** actions, transition function $S \times A_{1,\ldots,n} \times S \to [0,1]$, rewards function $R_i: S \times A_i \to \mathbb{R}$

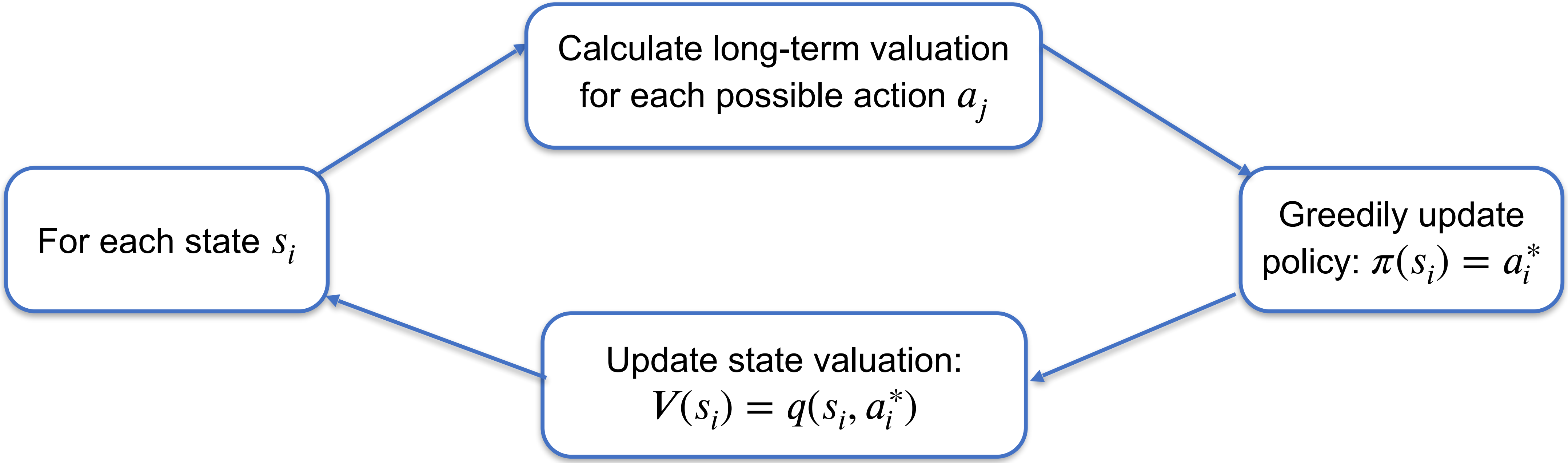
Problem Statement: find a policy function $\pi_i: S \times A_i \to [0,1]$ that satisfies a specific goal

Flashback: Solving MDP with Model from RL perspective

Value Iteration for Solving MDP (Bellman, 1957)

Assumption: transition function $f(S, A) \rightarrow S'$ and reward function $g(S, A) \rightarrow \mathbb{R}$ both known (quite strong assumption!)

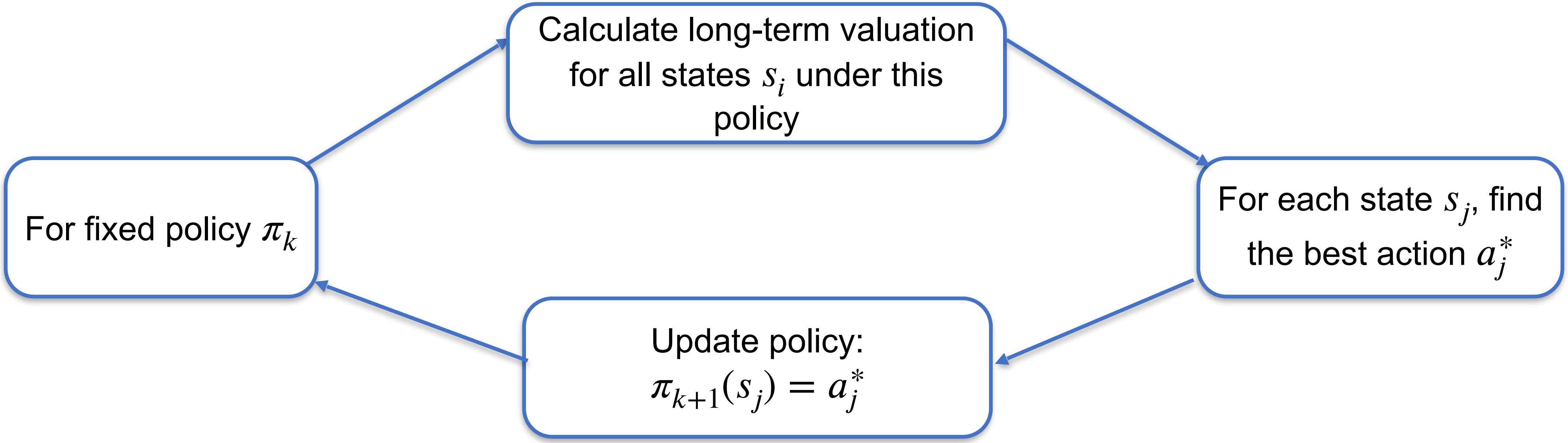
Idea: at each state, search over all possible actions, calculate the one with highest discounted rewards, and pick it



Policy Iteration for Solving MDP (Howard, 1960)

Assumption: transition function $f(S, A) \rightarrow S'$ and reward function $g(S, A) \rightarrow \mathbb{R}$ both known (quite strong assumption!)

Idea: start from a policy π_i, evaluate its value (Eq (2)), and update the policy if improvement available



Value Iteration for Solving Stochastic Game (Sharpley, 1953)

Assumption: 1. transition function $f(S, A) \rightarrow S'$ and reward function $g(S, A) \rightarrow \mathbb{R}$ both known;
2. Zero-sum game

1. Initialize V arbitrarily.
2. Repeat,
   (a) For each state, $s \in \mathcal{S}$, compute the matrix,

$$G_s(V) = \left[g_{a \in \mathcal{A}} : \begin{array}{l} R(s,a)+ \\ \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')V(s') \end{array} \right].$$

   (b) For each state, $s \in \mathcal{S}$, update V,

$$V(s) \leftarrow \text{Value}\left[G_s(V)\right].$$

Construct a Matrix Game from all A

Instead of $max\{\}$ in SR case, solve the matrix game for MR

Q-Learning for Solving MDP (Watkins, 1989)

Idea: without explicitly knowing the transition and reward function (thus no accurate Q value), use temporal-diff to appx

Algorithm 7.3: Optimal policy learning via Q-learning (off-policy version)

Initialization: Initial guess $q_0(s, a)$ for all (s, a). Behavior policy $\pi_b(a|s)$ for all (s, a). $\alpha_t(s, a) = \alpha > 0$ for all (s, a) and all t.

Goal: Learn an optimal target policy π_T for all states from the experience samples generated by π_b.

For each episode $\{s_0, a_0, r_1, s_1, a_1, r_2, \ldots\}$ generated by π_b, do

For each step $t = 0, 1, 2, \ldots$ of the episode, do

Update q-value for (s_t, a_t)*:*

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[q(s_t, a_t) - (r_{t+1} + \gamma \max_a q_t(s_{t+1}, a))\right]$$

Update target policy for s_t*:*

$$\pi_{T,t+1}(a|s_t) = 1 \text{ if } a = \arg\max_a q_{t+1}(s_t, a)$$

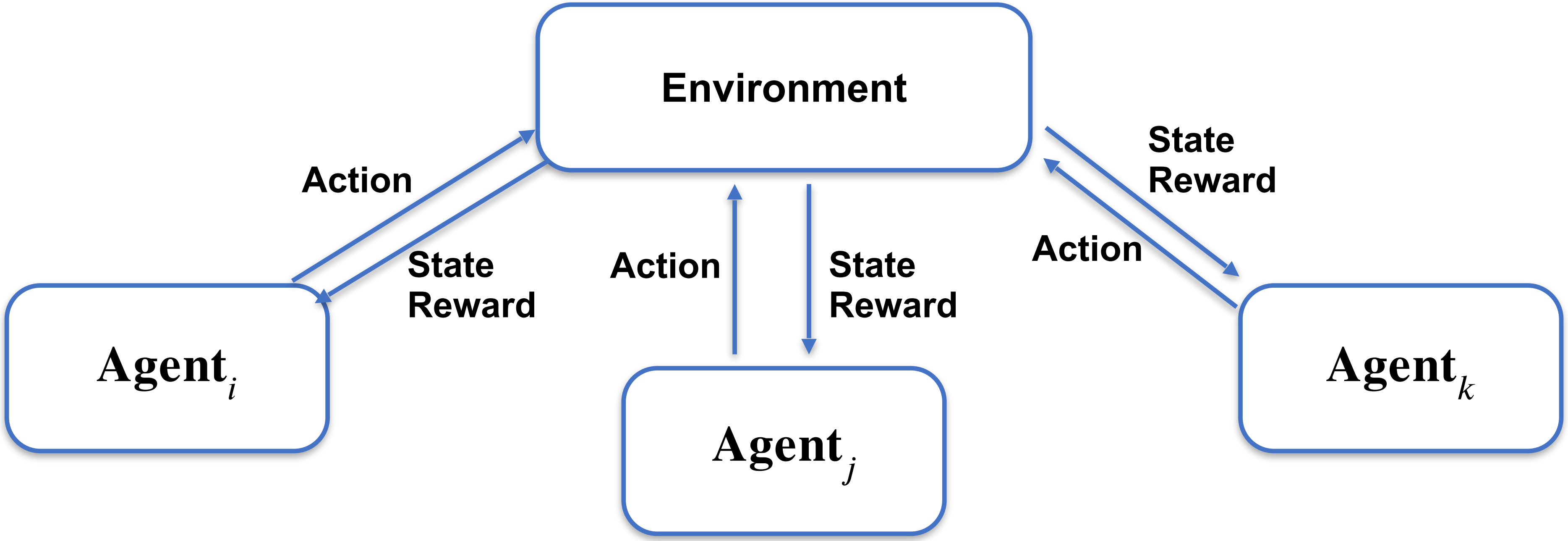
$$\pi_{T,t+1}(a|s_t) = 0 \text{ otherwise}$$

Iteratively estimate q(s, a)

Greedily take the optimal action

IndeQ-Learning (Tan, 1993)

Idea: treat other agents as a part of the environment, each agent trains via Q-learning in a decentralized manner



Critical issue: overfit to other agents' policies!

Minimax-Q and Nash-Q for Solving Stochastic Game

Minimax-Q (Littman, 1994) assumes two-player Zero-Sum game, while Nash-Q (Hu & Wellman, 2003) extends to multi-player general-sum

1. Initialize $Q(s \in \mathcal{S}, a \in \mathcal{A})$ arbitrarily, and set α to be the learning rate.
2. Repeat,
   (a) From state s select action a_i that solves the matrix game $\left[Q(s,a)_{a \in \mathcal{A}} \right]$, with some exploration.
   (b) Observing joint-action a, reward r, and next state s',

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma V(s')),$$

where,

$$V(s) = \text{Value}\left(\left[Q(s,a)_{a \in \mathcal{A}} \right] \right).$$

For Nash-Q, it is solving: $V(s) = Q(s) \times_{j=1}^{n} \pi^n(s)$, where $\pi^i(s)$ is the NE at stage s

For minimax-Q, it is solving:
$V(s) = \underset{\pi}{max}\,\underset{o}{min} \sum_{a \in A} Q(s, a, o)\pi_a$

What We Have So Far

Good Points:

1. To solve multi-agent stochastic games, we explored three directions: purely reinforcement learning, purely game-theoretical, and their mixture;

2. For normal form games with full observation, most follow-ups are developed based on Minimax-Q and Nash-Q;

3. A Formula for Innovation: **Game Solver** + **Environment Modeller** = **New Algorithm**

It'd be nice if we can:

1. Extend to extended form games

2. Handle real-world scenarios: how to deal with prohibitively large pure action spaces?

Fictitious Play (Brown, 1951)

Assumption: the opponents play stationary strategies; zero-sum game

Idea: based on opponents' play history, choose the best response

1. Initialize V arbitrarily, $U_i(s \in \mathcal{S}, a \in \mathcal{A}_i) \leftarrow 0$, and $C_i(s \in \mathcal{S}, a \in \mathcal{A}_i) \leftarrow 0$.
2. Repeat: for every state s, let joint action $a = (a_1, a_2)$, such that $a_i = \text{argmax}_{a_i \in \mathcal{A}_i} \frac{U^i(s,a_i)}{C_i(s,a_i)}$. Then,

$$
\begin{aligned}
C_i(s, a_i) &\leftarrow C_i(s, a_i) + 1 \\
U_i(s, a_i) &\leftarrow U_i(s, a_i) + R_i(s, a) + \gamma \left(\sum_{s' \in \mathcal{S}} T(s, a, s') V(s') \right) \\
V(s) &\leftarrow \max_{a_1 \in \mathcal{A}_1} \frac{U_1(s, a_1)}{C_1(s, a_1)}
\end{aligned}
$$

Flash-now: the Modern Variant of Fictitious Play

Extensive FP (Heinrich et al, 2015)

Pure Strategy: a sequence of deterministic actions;

Mix Strategy: a prob distribution over Pure Strategies;

Behaviour Strategy: a prob distribution over actions on a particular information state;

Kuhn's Theorem: there's an equivalence between a BS and a MS → solve EFG using FP in normal form games!

Some Followups:

1. Fictitious Self-Play: a "weakened" fictitious play framework, uses a RL architecture to appx BR, supervised learning arch to update strategy;

2. DFSP: uses neural network as the function appx;

Algorithm 1 Full-width extensive-form fictitious play

function FICTITIOUSPLAY(Γ)
  Initialize π_1 arbitrarily
  $j \leftarrow 1$
  while within computational budget **do**
    $\beta_{j+1} \leftarrow$ COMPUTEBRS(π_j)
    $\pi_{j+1} \leftarrow$ UPDATEAVGSTRATEGIES(π_j, β_{j+1})
    $j \leftarrow j + 1$
  end while
  return π_j
end function

function COMPUTEBRS(π)
  Recursively parse the game's state tree to compute a best response strategy profile, $\beta \in b(\pi)$.
  return β
end function

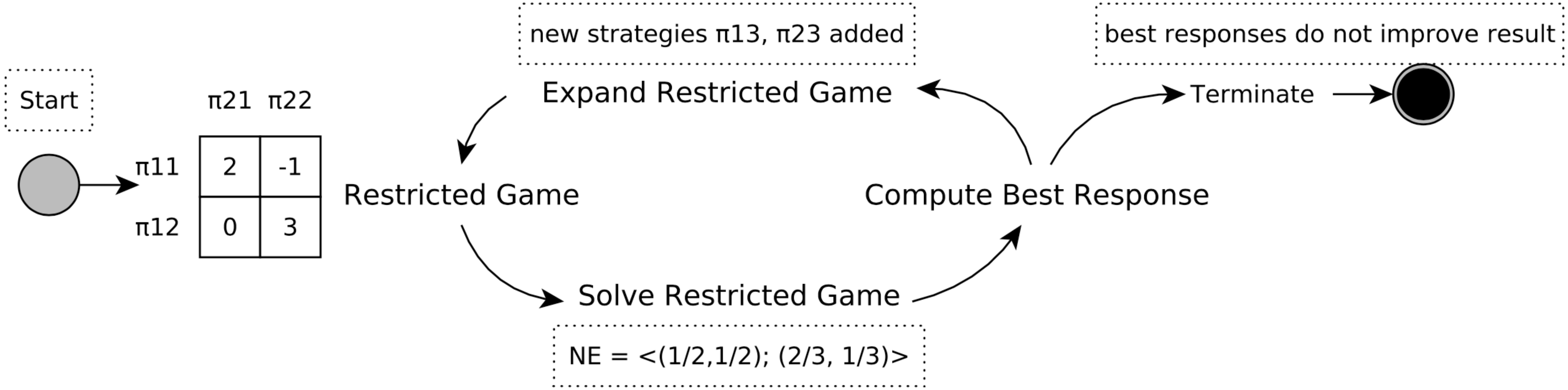
function UPDATEAVGSTRATEGIES(π_j, β_{j+1})
  Compute an updated strategy profile π_{j+1} according to Theorem 7.
  return π_{j+1}
end function

Double Oracle (McMahan, 2003)

In a real-world game, the opponents’s possible move space is numerously large, and we need to efficiently strategize to optimise for the worse case;

Assumption: Zero-sum, two-player matrix game

Idea: by iteratively compute & update the strategy set of a subgame, we eventually reaches equilibrium;



Policy-Space Response Oracle (Lanctot, 2017)

In some real-world games, the strategy space is not only large but in fact prohibitively expensive to enumerate (think about MoBA games like Dota 2 & StarCraft). Thus we focus more on **meta-games:** inducing the game via simulation

Assumption: Zero-sum, two-player EFG

Algorithm 1 A General Solver for Open-Ended Meta-Games

Initialise: the “high-level” policy set $\mathbb{S} = \prod_{i \in \mathcal{N}} \mathbb{S}^i$, the meta-game
and meta-policy $\boldsymbol{\pi}^i = \text{UNIFORM}(\mathbb{S}^i)$.
for iteration $t \in \{1, 2, ...\}$ **do**:
    for each player $i \in \mathcal{N}$ **do**:
        Compute the meta-policy $\boldsymbol{\pi}_t$ by meta-game solver $\mathcal{S}(\mathbf{M}_t)$.
        Find a new policy against others by Oracle: $S_t^i = \mathcal{O}^i($
        Expand $\mathbb{S}_{t+1}^i \leftarrow \mathbb{S}_t^i \cup \{S_t^i\}$ and update meta-payoff
    terminate if: $\mathbb{S}_{t+1}^i = \mathbb{S}_t^i, \forall i \in \mathcal{N}$.
Return: $\boldsymbol{\pi}$ and $\mathbb{S}$.

If the meta-solver calculates the NE, it degrades to Double Oracle

If the meta-solver selects the most recently added strategy, it degrades to self-play

Some “Recent” Advances: toward Competitive Games

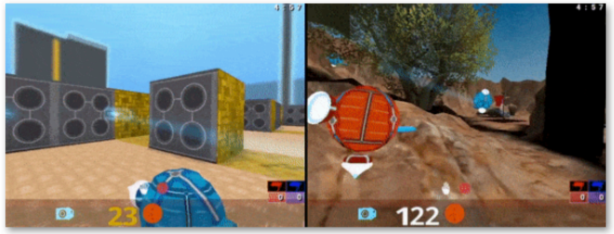
Research Advance + GPU computing Power = Remarkable Progress

Jan 2016 — Dec 2017 — July 2018 — Jan 2019 — Apr 2019 — July 2019 — Sep 2019

AlphaGO Series

milestone of single-agent decision-making technique

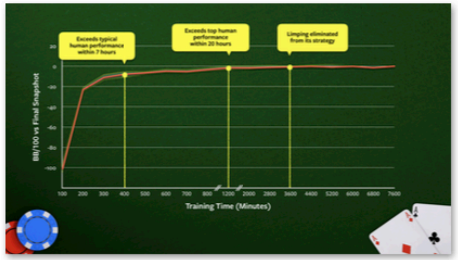
Capture-the-flag (DeepMind)



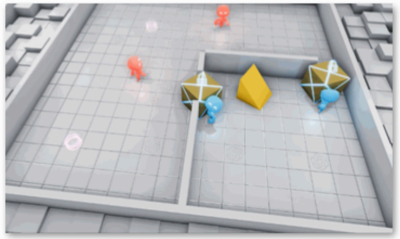
AlphaStar (DeepMind)

Dota2 (OpenAI)

Pluribus Poker (FAIR)



Hide and Seek (OpenAI)



References

[1]R. Bellman, “A Markovian Decision Process,” Indiana Univ. Math. J., vol. 6, no. 4, pp. 679–684, 1957.

[2]L. S. Shapley, “Stochastic games,” Proceedings of the national academy of sciences, vol. 39, no. 10, pp. 1095–1100, 1953.

[3]R. A. Howard, “Dynamic programming and markov processes.,” 1960.

[4]C. J. C. H. Watkins, “Learning from delayed rewards,” 1989.

[5]M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.

[6]M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in Machine learning proceedings 1994, Elsevier, 1994, pp. 157–163.

[7]J. Hu and M. P. Wellman, “Nash Q-learning for general-sum stochastic games,” Journal of machine learning research, vol. 4, no. Nov, pp. 1039–1069, 2003.

[8]G. W. Brown, “Iterative solution of games by fictitious play,” Act. Anal. Prod Allocation, vol. 13, no. 1, p. 374, 1951.

[9]J. Heinrich, M. Lanctot, and D. Silver, “Fictitious self-play in extensive-form games,” in International conference on machine learning, 2015, pp. 805–813.

[10] J. Heinrich and D. Silver, “Deep reinforcement learning from self-play in imperfect-information games,” arXiv preprint arXiv:1603.01121, 2016.

[11]H. B. McMahan, G. J. Gordon, and A. Blum, “Planning in the presence of cost functions controlled by an adversary,” in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 536–543.

[11]M. Lanctot et al., “A unified game-theoretic approach to multiagent reinforcement learning,” Advances in neural information processing systems, vol. 30, 2017.

[12]O. Vinyals et al., “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” Nature, vol. 575, no. 7782, pp. 350–354, 2019.

[13]N. Brown and T. Sandholm, “Superhuman AI for multiplayer poker,” Science, vol. 365, no. 6456, pp. 885–890, 2019.

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN