An intelligent Coup Agent

Wanling Liu, Junming Wang, Chaonan Ye
{liuwl, junmingw, yeco214} @ stanford.edu

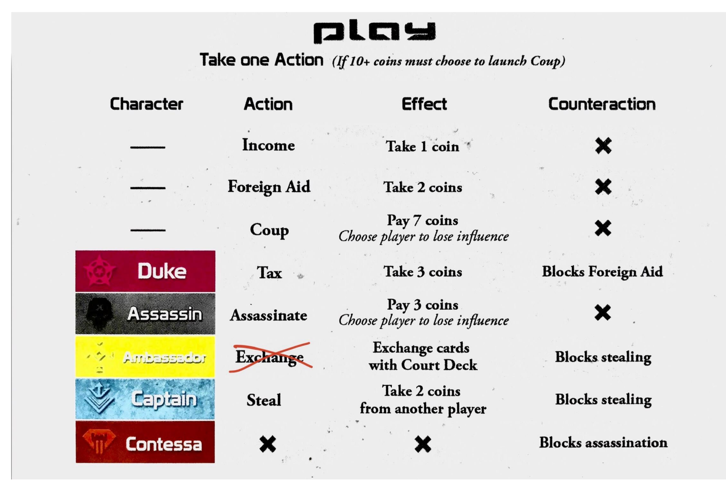
Stanford University

Overview

Board games present a unique challenge for AI agents. Coup, one of the most popular board games, is more challenging than others because deception plays a key role in gameplay. It is a multi-agent game and each agent would pick the optimal policy against other players with specific strategies. We modeled it as a Markov Decision Process (MDP) with unknown transition functions and trained an intelligent agent using Q-learning and feature extractions. The biggest challenge is the large state space.

Due to the complexity of the game, we re-range the game scope by introducing three simplifications:

1. Disable Ambassador's exchange action;
2. Only keep two cards for each character instead of three;
3. Only 3 players are enrolled in this game while the original version allows 2-6 players.



Models

[States]

Game state:

s_{game}= [[["duke", 1], ["ambassador", 1]],
[["assassin", 1], ["contessa", 2]],
[["captain", 1], ["ambassador", 1]]],
[4,5,3], ["steal", 2, 1]

Agent state:

s_{a0}= [[["duke", 1], ["ambassador", 1]],
1,
2],
[4,5,3], ["steal", 2, 1]

[Actions]

s_{a2}= [2, 1, [["captain", 1], ["ambassador", 1]]], [4,5,3], None

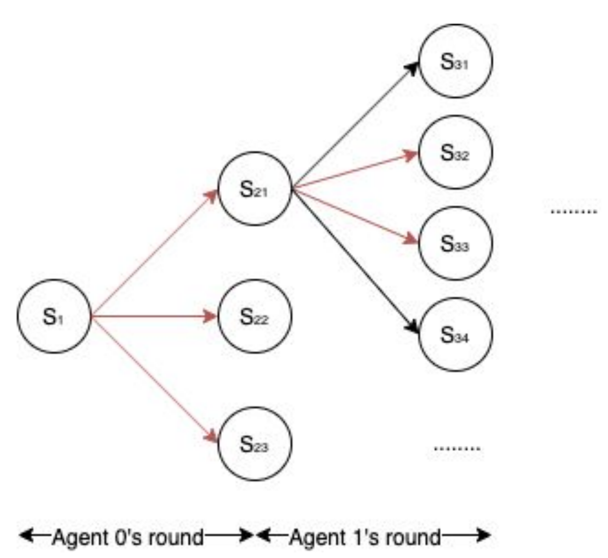
$$Actions(s_{a_2}) = \{[\text{"income"}, 2, 2], [\text{"foreign aid"}, 2, 2], [\text{"assassinate"}, 2, 0], [\text{"assassinate"}, 2, 1], [\text{"tax"}, 2, 2], [\text{"steal"}, 2, 0], [\text{"steal"}, 2, 1]\}$$

s_{a0} = [[["duke", 1], ["ambassador", 1]], 1, 2], [4,5,3], ["steal", 2, 1]

$$Actions(s_{a_0}) = \{[\text{"doubt"}, 0, 2], [\text{"no doubt"}, 0, 2]\}$$

$$Actions(s_{a_1}) = \{[\text{"block steal"}, 1, 2], [\text{"no block steal"}, 1, 2]\}$$

[Transition function]



[Reward]

- Win the game → + 1000
- Opponents lose cards → + 100 /card
- Lose cards → - 500/ card

[Evaluation function]

$$\text{WinRate} = \frac{\#\text{win}}{\#\text{total trials}}$$

Algorithms

- Q-learning
  - 1. Let the current state be s.
    2. Select an action a to perform.
    3. Let the reward received for performing a be r, and the resulting state be t.
    4. Update $Q(s,a)$ to reflect the observation $< s, a, r, t >$ as follows:
       $$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(t,a'))$$
       where α is the current learning rate.
    5. Go to step 1.
- Feature Extractions
  - Convert complicated states to simpler selected features
  - [tax, steal, assassinate, … block assassinate, living cards of agents, coins left of agents]
  - [1 0 … 0 0 | 2 1 1 | 3 2 0]
    Actions Living Cards Coins left
- Epsilon Greedy
  - fixed: ϵ = 0.2

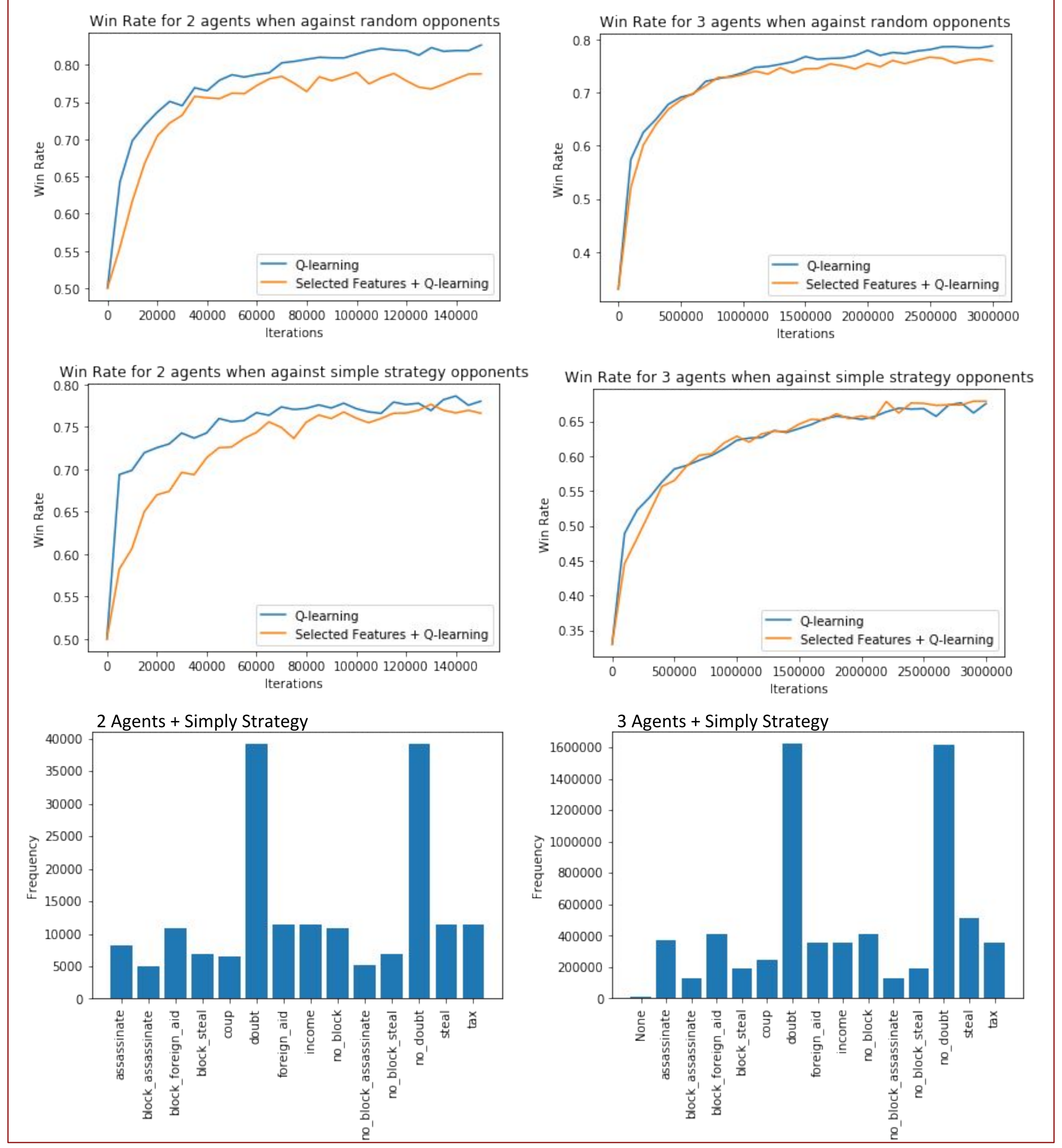
Experiments & Results

Iterations	Training	Testing
2 Agents	150,000	10,000
3 Agents	3,000,000	10,000

Runtime(s)	Q-learning	Features + QL
2 Agents	79.9	68.1
3 Agents	2350.9	2245.9

Simple Strategy vs. Simple Strategy

Experiments & Results



Conclusions

- Both our Q-learning algorithms provide substantial results. Trained agents perform impressively and reach high win rates.
- Q-learning with feature extractions is generally faster than Q-learning in the same experiment setting, but there is a tradeoff between win rate and running time.
- At a high level, actions distributions on states in learned policies are similar.
- The win rate of random vs. random is higher than simple vs. simple for both 2 and 3 agents.

Reference

[WD92] Christopher JCH Watkins and Peter Dayan. "Q-learning". In:Machine learning 8.3-4 (1992), pp. 279–292.
[RN94] Gavin A Rummery and Mahesan Niranjan. Online Q-learning using connectionist systems. Vol. 37. University of Cambridge, Department
of Engineering Cambridge, England, 1994.
[Mni+15] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: Nature 518.7540 (2015), p. 529.
[Sil+16] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: nature 529.7587 (2016), p. 484.