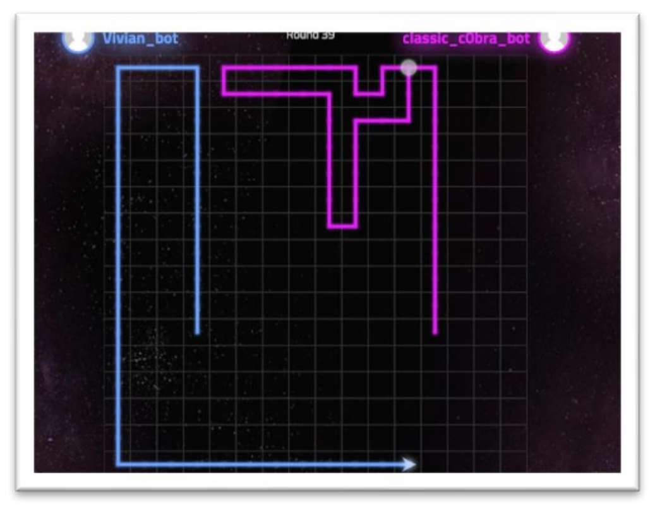
AI Agent for Light Rider

Yipeng He, Wantong Jiang, Di Bai
{yipenghe, wantongj, dibai}@stanford.edu

Overview

Light Rider is an adversarial game where two players on the grid take one step to its adjacent cell and occupy it each round. The player who has no way to go or mistakenly takes an unavailable cell loses the game. *Riddles.io* provides the game platform.



Model and Algorithms

- **Model:**
  We model the game as a two-player simultaneous adversarial game. Each state is composed of the whole board information. The figure below shows an example of a game state on a 4*4 grid.
  - '.' represents empty cells
  - '*' represents occupied cells
  - '0' and '1' represent the position of two players

*	*	.	.
0	.	.	.
.	.	1	.
.	*	*	.

- **Algorithms:**
  To implement the agent, we utilize both game theory methods and reinforcement learning approach.
  - **Minimax with alpha-beta pruning**: We measure how alpha-beta pruning accelerates the search process and how different evaluation functions affect the performance.
  - **Q-learning**: Stochastic gradient descent on feature weights. Learn against a minimax agent.

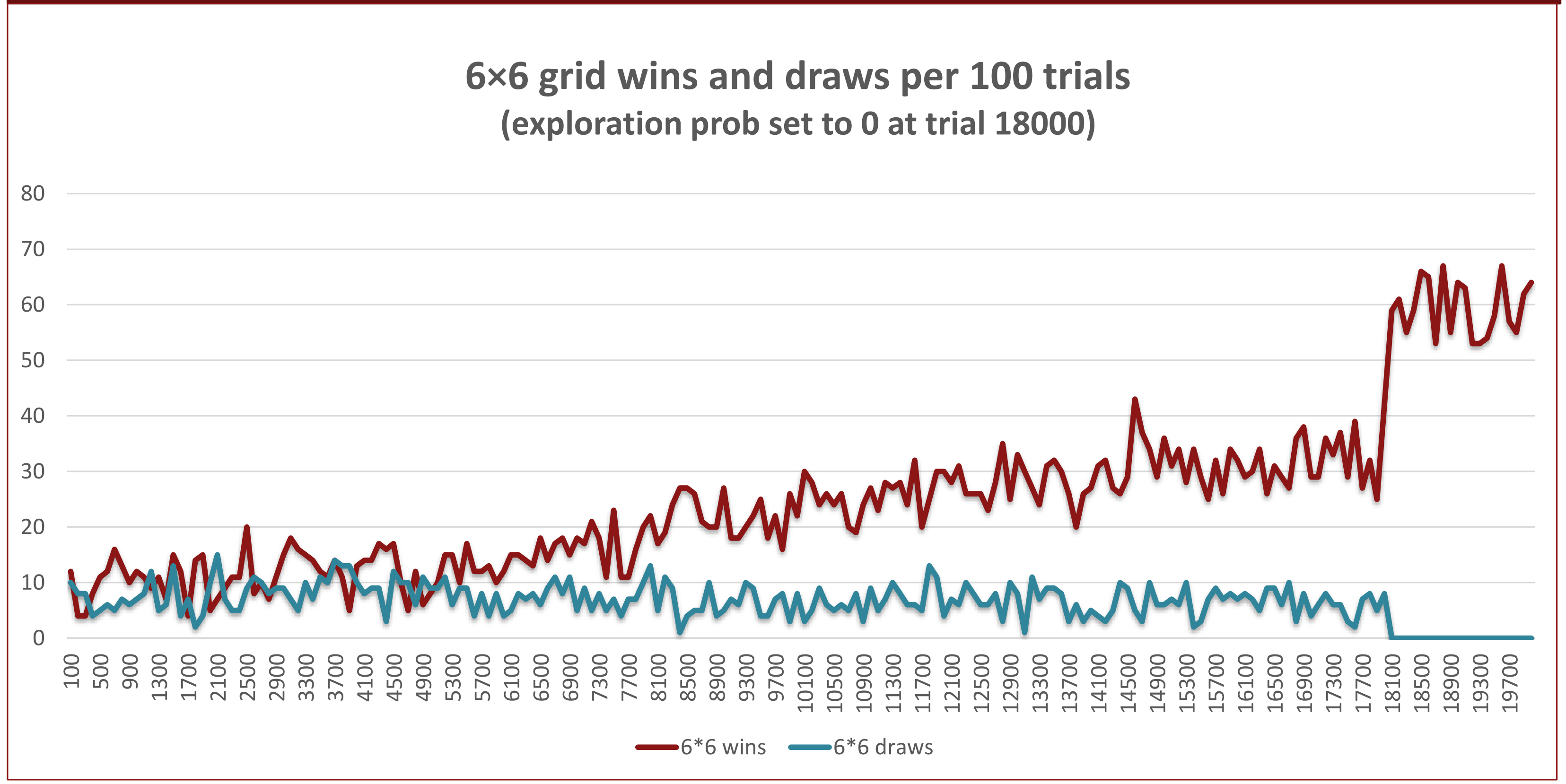
Minimax

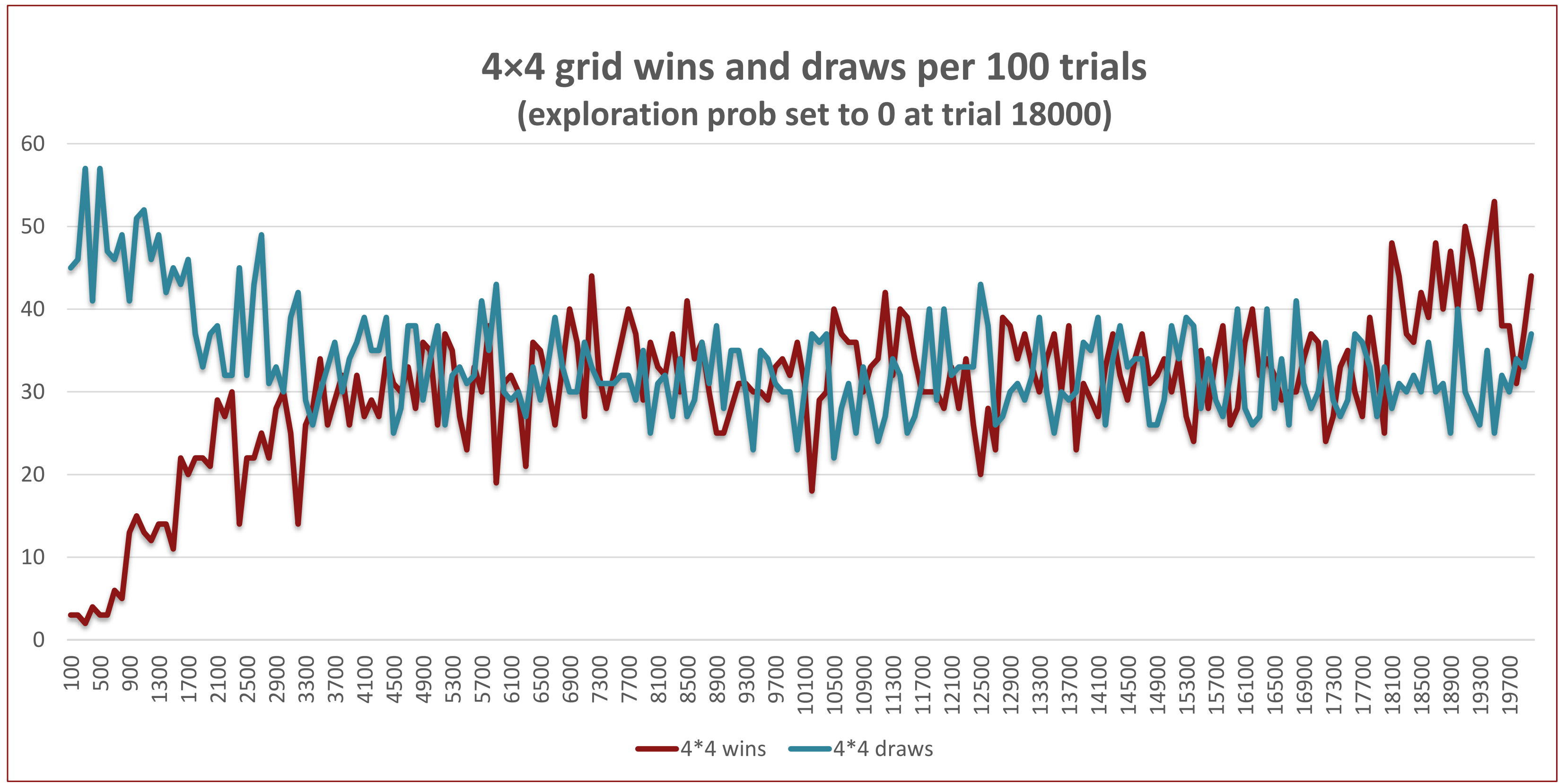
- Simultaneous Game into Turn-based Game
- α-β pruning increases search depth from 2 to 4 within online time limit
- Implement offline simulator for game evaluation
- **Evaluation function**: More space for agent, less space for opponent
  - Score on available spaces and flexibility to change directions on sum / max of all legal actions
  - Score on shorter distance to opponent, "attack" opponent
  - Score on shorter distance to positions surrounding opponent, "block" opponent

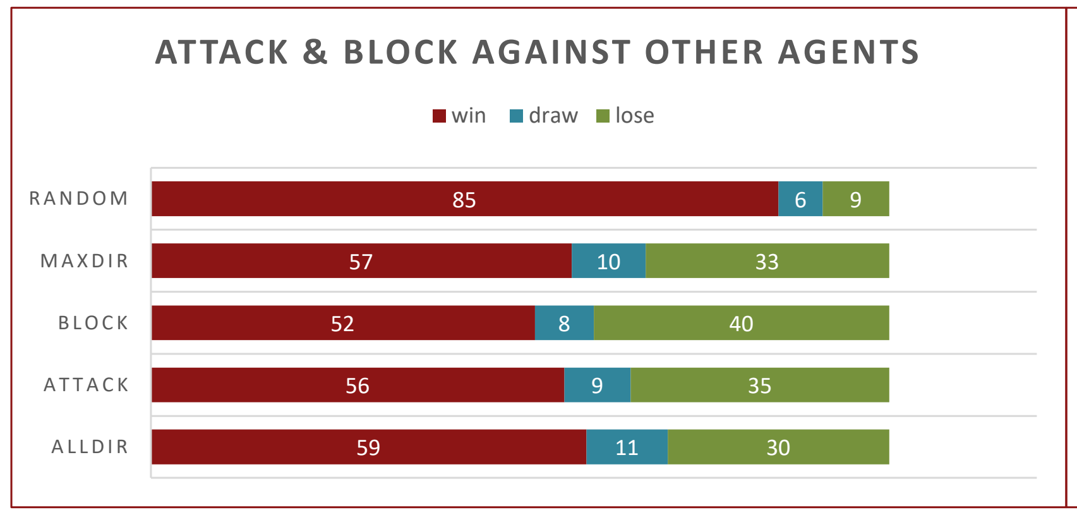
Q-Learning

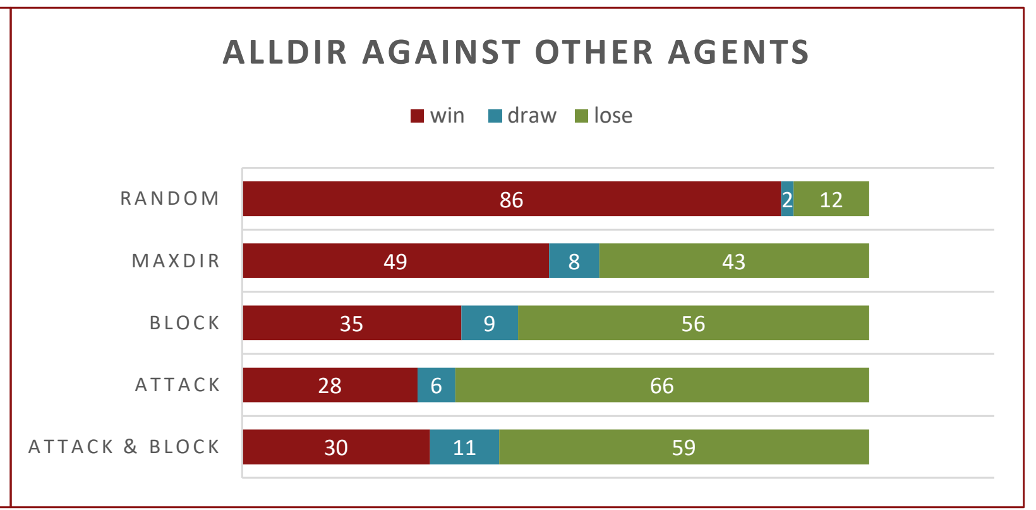
- **Feature**: (board layout, action) tuple
- **Reward**: win +100, lose or draw -100, otherwise 0
- **Training**:
  - Play with minimax agent for 20000 rounds
  - Epsilon-greedy: ε = 0.2
  - discount = 0.99
  - Stochastic gradient descent on feature weights
  - $\mathbf{w} \leftarrow \mathbf{w} - \eta[\hat{Q}_{opt}(s,a;\mathbf{w}) - (r + \gamma \hat{V}_{opt}(s'))]\phi(s,a)$

Test Results









Conclusion & Discussion

- **Minimax:**
  - α-β pruning makes search **2X** faster with search depth 4
  - Evaluation function with "block" and "attack" effectively improve win rate
- **Q-Learning:**
  - On a 4*4 grid, The agent performs as well as the minimax agent it was trained against. the agent is able to achieve around 40% win rate at test time after 18000 training trials.
  - At test time on a 6*6 grid, the agent is able to achieve around 60% win rate, while the draw rate drops to 0%, meaning the agent is able to outperform the minimax agent it was trained against.
  - The significant difference between the draw rates may be caused by the grid size. Minimax agent with depth 3 is also harder to beat on a smaller grid, since 3 steps on a 4*4 grid is a lot (every agent can at most take 7 moves).

Future Work

- **Minimax:**
  - TD-Learning for evaluation function
  - Faster evaluation function for online playing purpose
- **Q-Learning**:
  - Better feature extractors for generalization to work on large grids
  - Use neural networks or other representation functions for function approximation
- **Other Algorithm**:
  - Try Monte Carlo Tree Search

Acknowledgement

We would like to thank Prof. Percy Liang, Dr. Kelvin Guu and CA Andrew Han for their generous instructions and insightful suggestions.
We would also like to thank user DogOfHouseDawg for providing some starter codes on Riddle.io.