CS221 Project Proposal AI Agent for Chinese Chess

Li Deng[a]

[a] *Stanford University - SUID: dengl11*

Abstract

As a Chinese Chess Lover from childhood, I have great passion for building an AI agent for Chinese chess. I have been destroyed by my father in this game, and now with what I have learned and going to learn from CS 221, I am going to challenge him with my AI-powered chess agent!

The simulator is already built with web interface and web server by *Julia*. Also a greedy player is implemented as the baseline. The next step is to implement several AI agent with different strateties: minimax game tree, alpha-beta pruning, Temporal Difference learning, reinforcement learning for evaluation function, and also potential training in neural network.

Keywords: AI, Game, Chinese Chess

1. Introduction

Chinese chess (Xiang Qi) is one of the most popular board games worldwide. Having a long history, the modern form of Chinese chess was popular during the Southern Song Dynasty (11271279 A.D.).

Chinese chess is a two-player, zero-sum game with complete information. Chinese-chess expert knowledge started to be developed some 800 years ago. Nowadays, the world has many excellent human players. Yet, already now, the strength of the best Chinese-chess programs can be compared to that of human players notwithstanding the fact that the game is considered rather complex. The state-space complexity of Western chess and Chinese chess was estimated by Allis (1994). The game-tree complexity of Chinese chess is based on a branching factor of 38 and an average game length of 95 plies (Hsu, 1990). Detailed game rules can be found at https://en.wikipedia.org/wiki/Xiangqi#Rules

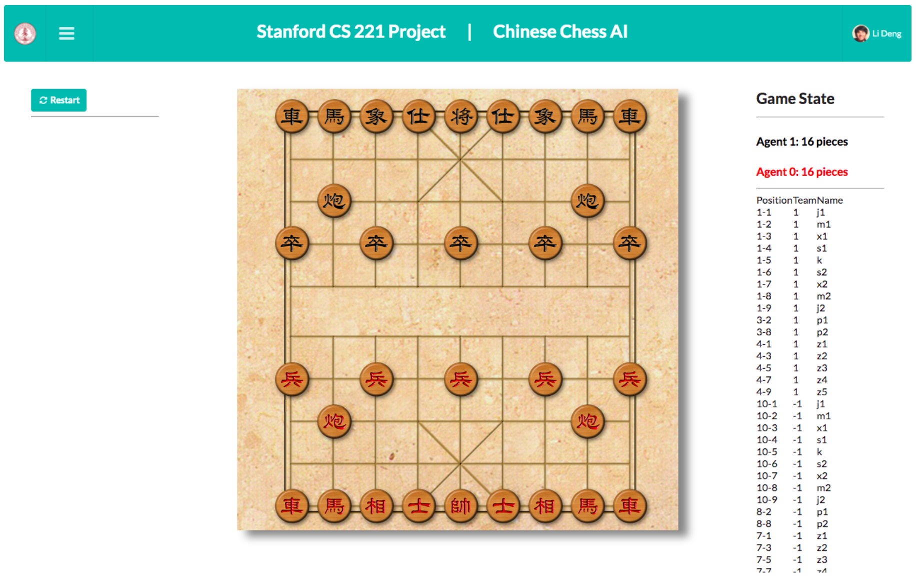
Table 1: State-space complexity and game-tree complexity (log)

Game	Space Complexity	Game Tree Complexity	Branching Factor
Chess	50	123	35
Chinese chess	**48**	**150**	38
Go	160	400	250

So we can see that the complexity of Chinese Chess is between Chess and Go. It is challenging enough to be a course project, but not too complex to analyze.

2. Simulator

The simulator is built with with a web interface by *Angular2* and a web server with *node* powered by *Julia*. Compared to GUI interface by Python, web interface has more flexible and powerful representation. *Julia* is a new programming language, and is famous for its high performance. AI strategies are going to be implemented in *Julia*. *Node* takes output computing result from *Julia*, and *Angular2* takes the data about chess piece move and renders on web. Also *Angular2* sends updated game state to *Node* , and then to *Julia* to compute. So this is the work-flow of the game simulator.



Game State

Agent 1: 16 pieces

Agent 0: 16 pieces

```
PositionTeamName
1-1     1    j1
1-2     1    m1
1-3     1    x1
1-4     1    s1
1-5     1    k
1-6     1    s2
1-7     1    x2
1-8     1    m2
1-9     1    j2
3-2     1    p1
3-8     1    p2
4-1     1    z1
4-3     1    z2
4-5     1    z3
4-7     1    z4
4-9     1    z5
10-1    -1   j1
10-2    -1   m1
10-3    -1   x1
10-4    -1   s1
10-5    -1   k
10-6    -1   s2
10-7    -1   x2
10-8    -1   m2
10-9    -1   j2
8-2     -1   p1
8-8     -1   p2
7-1     -1   z1
7-3     -1   z2
7-5     -1   z3
7-7     -1   z4
```

3. Base Line and Oracle

Greedy strategy is used as the baseline: i.e, as long as there is one opponent piece available to be captured, then capture it. By playing with this greedy strategy, and it winning rate against me is around 10%.

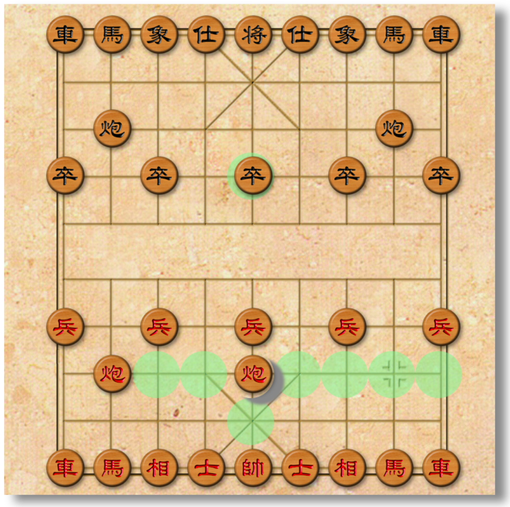
In the papers about Chinese Chess, the benchmark is the top chess player in the world. Due to the lack of such top human player for my game, currently I am using the top level on QQ game online platform. And its winning rate against me is around 95%.

4. Strategies

- Game Tree: Minimax, alpha-beta pruning
- Reinforcement learning with evaluation function
- Temporal Difference Learning
- Neural network

5. Github Repo

The project is hosted on Github:
https://github.com/dengl11/ChineseChessAI



Green spots mean the legal moves of selected piece.