

# Project 3 Report

---

郭庭佑

28<sup>th</sup> November, 2022

## 1 DESIGN

Besides the basic Monte-Carlo Tree Search, here are the techniques or tricks I applied:

1. RAVE
2. UCB1-Tuned
3. Choice of parameter
4. Bitboard

The rest of this report points out some details of these techniques.

### 1.1 RAVE

With RAVE, the win rate to strong was raised from 60% to 70%

### 1.2 UCB1-TUNED

It is easy to modify so I tried this. The win rate to strong was about the same.

### 1.3 CHOICE OF PARAMETER

There are some parameters to be optimized:

1.  $c$  is the constant balancing the exploration and exploitation. That is to maximize

$$\bar{X}_j + c \sqrt{\frac{2 \log n}{T(n)_j}}$$

2.  $k$  is the decreasing factor in RAVE. That is

$$\beta = \sqrt{\frac{k}{n+k}}$$

With some self-competition test, I set  $c = 1.5$  and  $k = 10$ .

## 1.4 BITBOARD

Store black and white pieces on the board by 2 `__uint128_t`. Also maintain feasible moves for black and white each. In the code of the `random` player, we used to first shuffle all moves and try all moves, this do a lot of unnecessary attempts calculating liberty, and even worse in late game. With bitboard, the check liberty process speeds up, we can fetch a feasible moves table in  $O(1)$ . We notice the time cost to do fixed times of Monte-Carlo simulation is much faster in late game. Originally,  $T$ , the times of Monte-Carlo simulation in one action, is set to  $10^4$  to meet the time condition. With bitboard,  $T$  can be set to  $6 \cdot 10^4$ . The win rate to *strong* is increased from 70% to 95%.