City University of Hong Kong

Department of Computer Science

CS4386 AI Game Programming

Semester B, 2022-2023

Assignment I report

Name: LI Xiao Yang

SID: 56638660

Contents

1 Methodology

1.1 Monte Carlo Tree Search

A monte carlo tree search (mcts) based algorithm will be adopted in this assignment. One of the most significant reasons is that there will be only 10 seconds for each player to make a move in the game. Therefore, the property of monte carlo tree search is appreciated since it can be terminated anytime. Another issue is that, however, 10 seconds are rather limited for monte carlo based algorithm to approach a precise solution. To tackle this challenge, a heuristic function is introduced to do pruning and avoid blunders, which is demonstrated in the following sections.

1.2 Pseudocode

Algorithm 1 MCTS

$root \leftarrow MonteCarloTreeNode(state)$

$iteration \leftarrow 0$

while $iteration \leq limit_{iteration}$ **do**

$leaf \leftarrow select(root)$

$child \leftarrow expand(leaf)$

$result \leftarrow simulate(child)$

$backpropagate(child, result)$

$iteration \leftarrow iteration + 1$

end while

$solution \leftarrow \max root.children$

1.3 Heuristic function

Intuitively, only a small set of children are required to explore. For instance, some blunders can easily be detected with well defined heuristic functions.

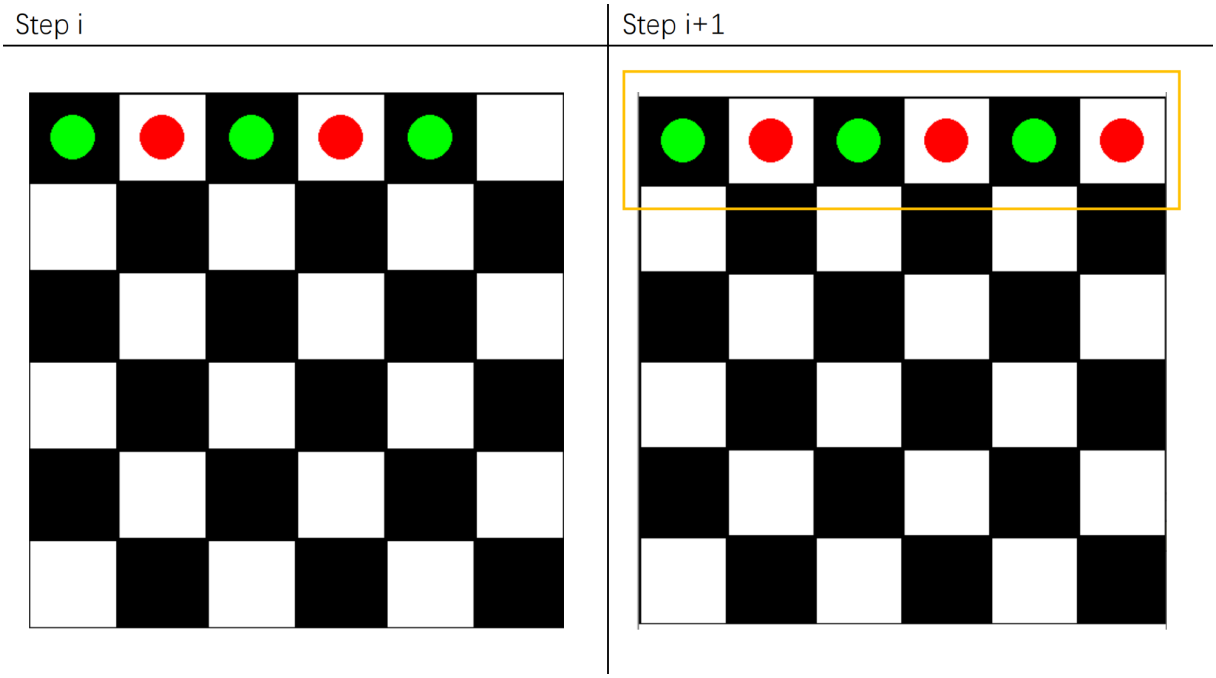

Figure 1: Step i for AI player, Step i+1 for opponent

The basic idea of the heuristic function is to reward when getting scores $s(s \in \{3, 6\})$ by a move and to punish when reaching a score-losing state (see figure above).

Algorithm 2 Heuristic

$utility \leftarrow 0$

$utility \leftarrow utility + reward(move)$

$utility \leftarrow utility - rowPenalty(move)$

$utility \leftarrow utility - columnPenalty(move)$

$utility$

It is worth noting that the number of iterations significantly increases by adding pruning strategy based on heuristic functions. Nevertheless, it is proven by experiment that the calculation of heuristic function does not have much influence on the running time of one single iteration.

Algorithm 3 All possible moves

$moves \leftarrow \emptyset$

$utility_{max} \leftarrow \max_{i \in move} Heuristic(i)$

for $cell \in grid$ **do**

  if $Heuristic(cell) = utility_{max}$ **then**

    $moves \leftarrow moves \cup cell$

  end if

end for

$moves$

Basically, all children with heuristic value no less than the maximum utility will be expanded. In this way, the size of entire search tree is dramatically reduced without much loss of optimality.

2 Experiment

Six groups of experiments are conducted with 10 games each. It is clear to see the win rate and total scores under arbitrary algorithms against different opponents. Most importantly, it can be observed that monte carlo based algorithm frame work does improve the performance in terms of both win rate and total scores. Besides, it is found that late hand player has tremendous advantage against early hand player.

op/AI	CPP	PYTHON
Random	10/10 (678)	10/10 (714)
S	5/10 (528)	6/10 (552)
S+MCTS	5/10 (505)	5/10 (576)

Figure 2: S=pure strategy, S+MCST=Heuristic+Monte Carlo

In terms of the future improvement, parallelization on expansion and simulation can be explored. Also, different tree policies may lead to significant performance improvement as well.

3 Acknowledgements

I would like to thank all TAs support for presubmission testing and suggestions. Apart from that, I would like to acknowledge ZHOU Jun Chen for effective discussions.

References

[Liu et al., 2020] Liu, A., Chen, J., Yu, M., Zhai, Y., Zhou, X., and Liu, J. (2020). Watch the unobserved: A simple approach to parallelizing monte carlo tree search. In *International Conference on Learning Representations*.