Analysis of Monte Carlo Tree Search technique in Game Isolation

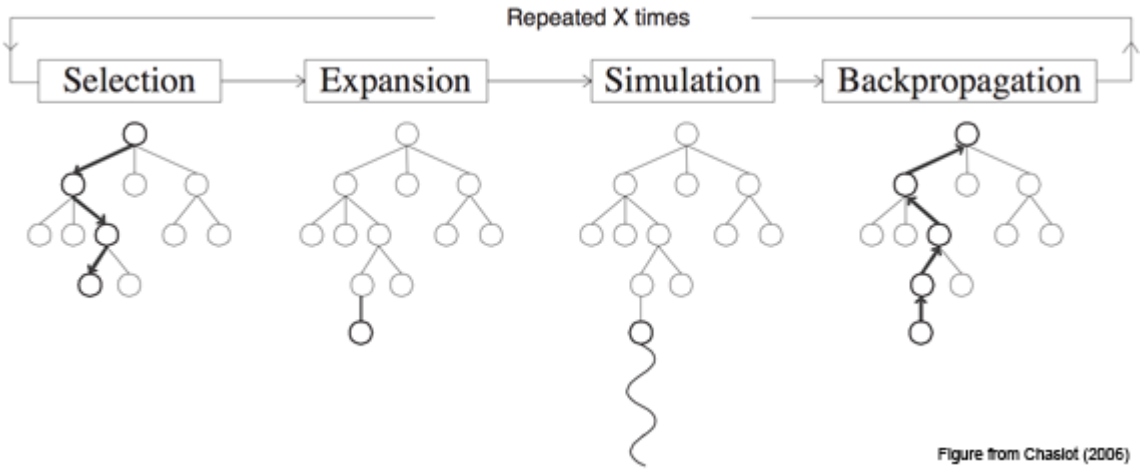
Yazhuo Li

Abstract

This report studies the performance of an adversarial game (Isolation) playing agent based on Monte Carlo Tree Search (MCTS) technique. Detailed performance analysis of this proposed agent will be carried out through game playing against Random agent, Greedy agent and Minimax Agent. The study is based on the comparison of the proposed agent and baseline agent – Alpha-beta search with iterative deepening.

Advanced Search agent – MCTS based

The implementation of the proposed MCTS agent can be summarised as Selection – Expansion-Simulation – Backpropagation processes[1].



During selection process, tree policy with "Upper Confidence Bounds for Trees (UCT)" was implemented. The following formula is used [1]:

$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

This is used to balance exploitation (expanding the same node) and exploration (adding more other nodes). In this study, the E_E_Ratio (Constant C here) is tuned to achieve the best performance.

At Expansion stage, none-terminal nodes are expanded through creating more child nodes and selecting one. Then the simulation will be run from this point on until termination stage, the result will

be back-propagated across all the parent nodes. At the end, based on simulation results, the most visited node will be selected.

Performance

Table 1 and **Figure 1** show the tuning of exploitation to exploration ratio(e-e-ratio) – from 0.75 to 1.5, using winning rate against Minimax agent. Search time has been proven to affect the result. Here we use 150ms, 250ms, 350ms and 500ms search time to study the performance of the agent. We will follow the same protocol and break down all the performance result with different search time limit.

Search Time/E_E_Ratio	0.75	1	1.25	1.5
500ms	67	71.8	72.8	71.6
350ms	67	68.5	68.4	68.6
250ms	63	64.5	67.9	67
150ms	64.5	60.7	63.6	63.1

Table 1. Winning rate against Minimax Agent with different E_E_Ratio

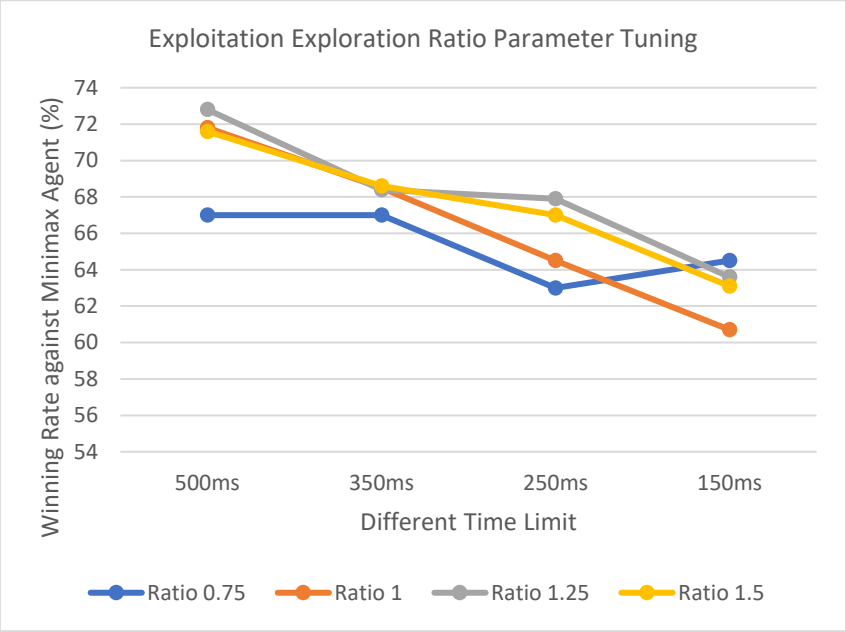


Figure 1. E_E_Ratio Tuning

We can see from figure 1 that when E_E_Ratio = **1.25** the agent shows the best performance and therefore it is chosen for the following study.

Alpha-beta Search		Depth = 3	
	Random	Greedy	Minimax
500ms	92.5	63.7	51
350ms	94.1	65.3	50
250ms	93.9	64	47.2
150ms	94.7	63.6	53.2

Alpha-beta Search		Depth = 4	
	Random	Greedy	Minimax
500ms	93.8	75.7	53.4
350ms	94	77.2	50.8
250ms	93.7	77.5	53.6
150ms	93.2	78	53.3

MCTS		Ratio = 1.25	
	Random	Greedy	Minimax
500ms	99.5	87.6	72.8
350ms	99.5	86.7	68.4
250ms	98	85.7	67.9
150ms	96.2	83.9	63.6

Table 2. Winning rate against different agents – baseline and MCTS

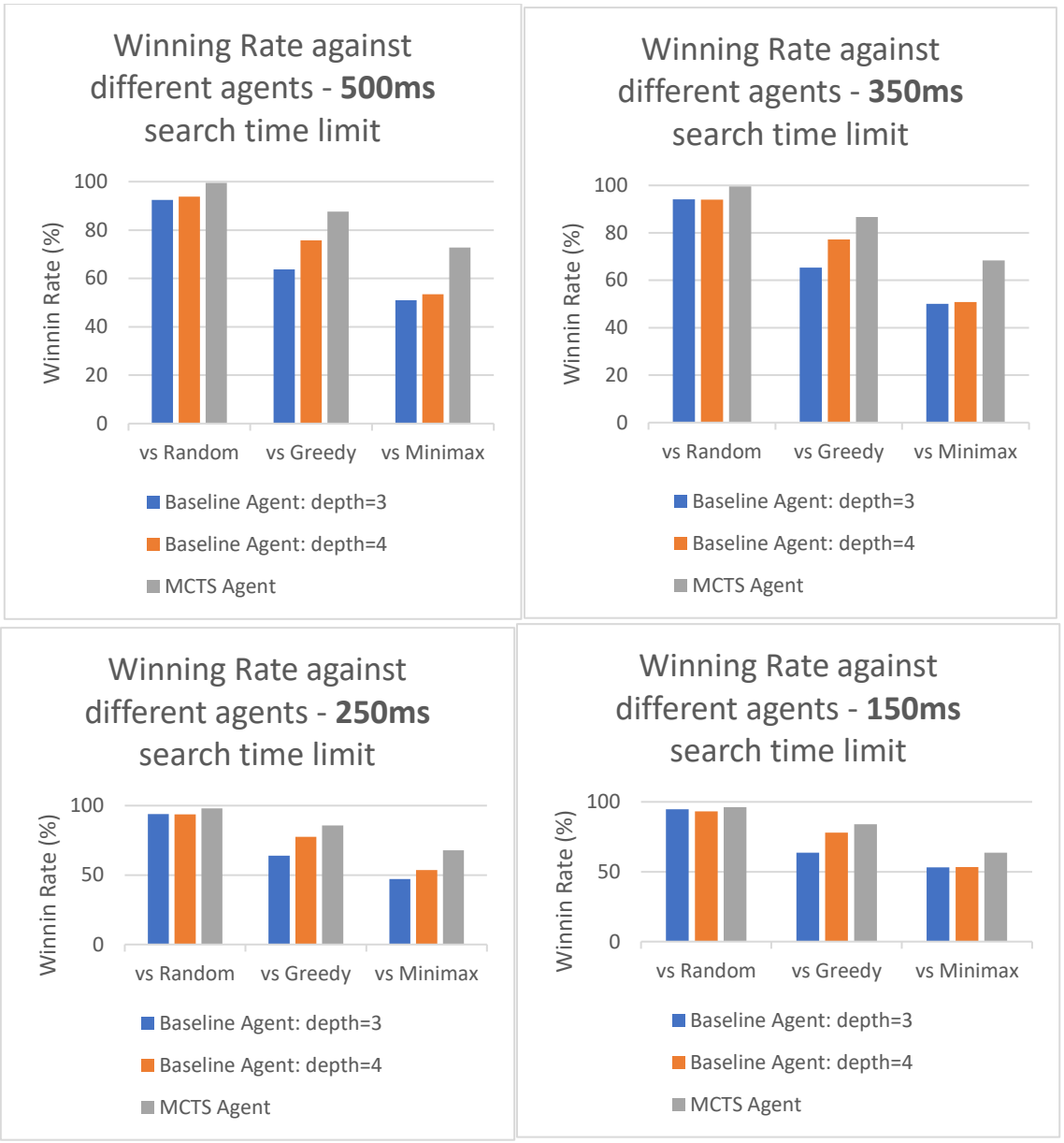


Figure 2. Winning rate against different agents – baseline and MCTS

Table 2 and Figure 2 illustrate the winning rate of the MCTS agent and baseline agent against Random, Greedy and Minimax agents. Search depth 3 and 4 are used for baseline agent study.

We can see that the proposed MCTS agent shows around 20 to 40 percent performance increase compared to baseline agent. It is worth noting that when search time is increased, the MCTS has more advantages against baseline. For example, when search time is increased to 500ms, the proposed MCTS agent can achieve a winning rate of 72.8% against minimax agent, whereas it doesn't give as much an increase in baseline agent's performance.

Questions

- Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?

  Compared to baseline agent (based on alpha-beta search with iterative deepening at the depth of 3 and 4), the proposed MCTS agent has shown around 20% - 40% improvement in performance.

- Why do you think the technique you chose was more (or less) effective than the baseline?

  MCTS search has a better balance between depth-based search and breadth-based search. MCTS will always reach terminal state first before propagating back. Alpha-beta pruning with the depth of 3 or 4 will not necessarily reach terminal stage before the cut-off time. MCTS as an asymmetric search technique also is more effective in going further into the search tree within the same amount of search time.

Note:

1. All the results are based in 1000 Games played.
2. More search time limit has given MCTS agent more advantages.
3. Fair Game play flag is switched off because MCTS uses simulation to decide on starting node. Using opponent's opening move is not needed for finding the best solutions.
4. Further fine turning of the E_E_Ratio can achieve even better performance.

Reference

[1] http://mcts.ai/about/

[2] C. Browne, E. Powley, etc. "*A Survey of Monte Carlo Tree Search Methods*", Vol.4, No.1, IEEE Transactions on Computational Intelligence and AI Games", March 2012