Jakub Skalski

Monte Carlo Tree Search

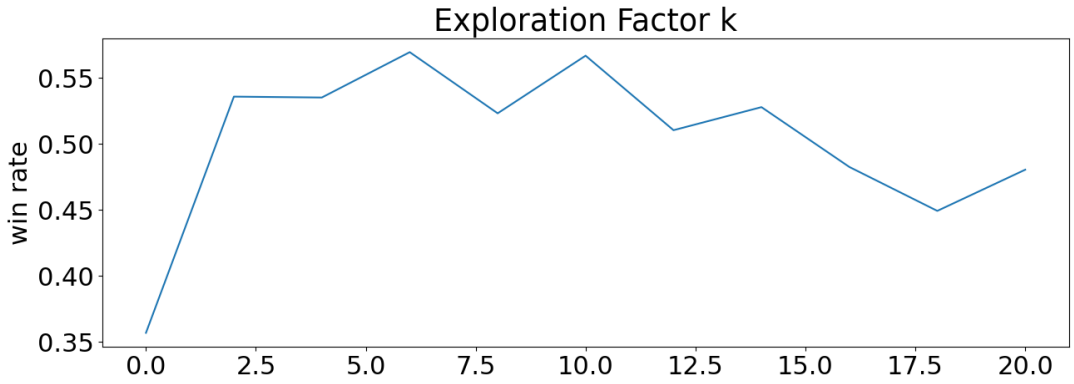
17th October 2021

Introduction

MCTS has seen a lot of success in the field of artificial intelligence. In this report we will be applying it to the *Ultimate Tic Tac Toe* game on the CodingGames platform. All mentioned algorithms are implemented in Javascript. The custom built game engine is able to handle 30000 simulations per second (measured with CodingGames during the preprocessing phase). Unsurprisingly Flat MC scores pretty much the same amount of simulations while the pure MCTS finishes at around 16000 simulations per second. It's worth mentioning that this implementation of MCTS recycles branches from the previous turns and thus it is quite common to see trees with initial sizes of around 500 simulations which should amount to just a little less than 2000 simulations per 100ms. Both algorithms were run a few dozens of times and the time limit was never exceeded thus it is rather safe to assume this to be the lower bound for the algorithms' efficiency.

Enchantments and tuning

All test samples consist of 1000 matches with 200 simulations per turn each. We begin with tuning of exploration factor for the USB oriented algorithms. The choice will be made based on a series of simulated games between pure MCTS agents with exploration factor of 1 and *k*. Results for different *k* values are as follows:

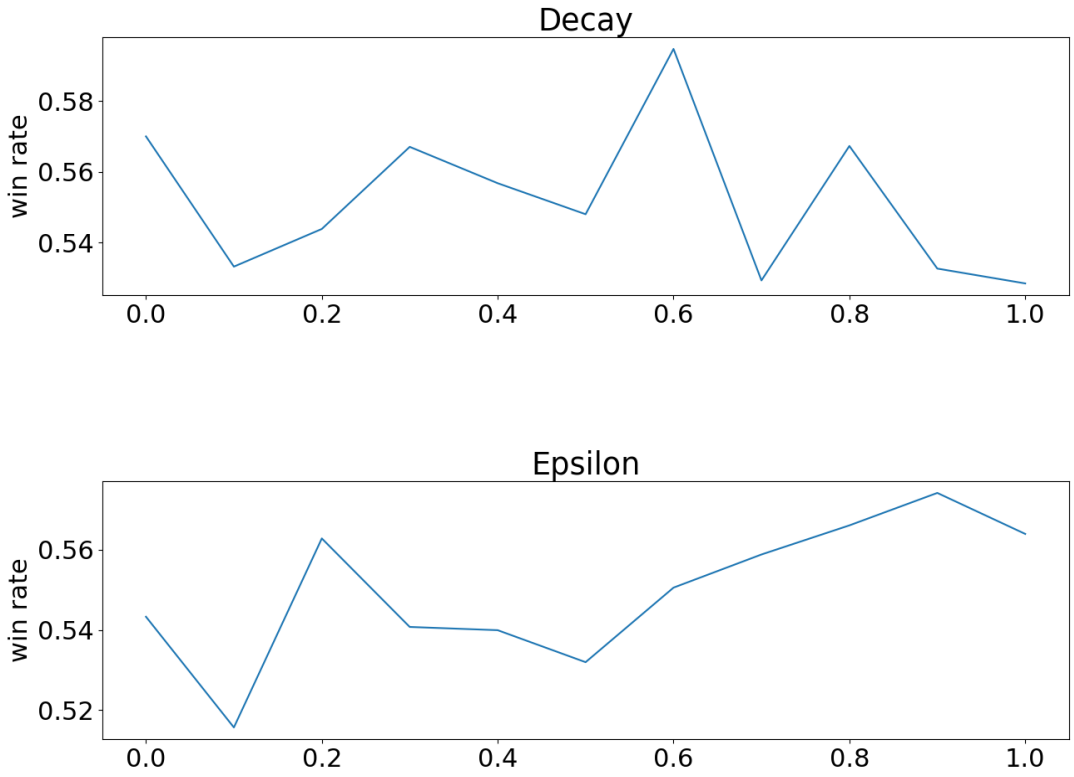


SHOT

Although there is no tuning to be done here we can still tweak the algorithm by choosing the implementation of the transposition table. In general when it comes to turn based board games rotations and color swappings are the most obvious choices to go for. Given the somewhat peculiar board of the *Ultimate Tic Tac Toe* we have some extra space for creativity. For instance we could say that one board is a transposition of the other if we can pair up their sub boards so that each pair contains the same sub boards. For example if we consider the number of moves available to us in the first turn, the reduction of 72 actions is pretty significant.

MAST

Tuning here is done under the assumption that there is little to no correlation between the *decay* and *epsilon* parameters (satisfactory tests for all combinations between these values would be too expensive to compute in a simple manner). The agents were tested against pure MCTS with exploration factor of 6.



Method comparison

Agents will face off against each other in a series of duels. This time around we will adjust the number of simulations so that each agent will take the same amount of time per turn.

	mcts	mast-mcts	shot	mast-shot
mcts	0.5	0.47	0.39	0.37
mast-mcts	0.53	0.5	0.44	0.38
shot	0.61	0.56	0.5	0.49
mast-shot	0.63	0.62	0.51	0.5

Discussion

As seen in the table, switching from regular MCTS type algorithms to SHOT results in significant gain in win rate. Although SHOT is somewhat of a wild card its high performance might be justified. According to the source it is expected to work well in combinatorial games where regular MCTS would struggle. As for the MAST enchantment it is generally expected for it to be a straight up improvement since it raises the quality of the simulations. In *Ultimate Tic Tac Toe,* however, the increase is rather insignificant supposedly because the moves themselves are highly dependent on the scenario in which the move is being performed, thus keeping track of them doesn't amount to all that much.

So.. how good is it?

Well... the agent would often encounter difficulty finishing off the opponent or blocking an obviously winning move. Its opening moves could use some improvements as well, since the agent has yet to figure out that choosing the center spot in some sub boards is actually pretty advantageous, especially when going first. In the end the agent was able to rank at 60th place in the silver league.