Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

OPTIMAL COMPUTING BUDGET ALLOCATION FOR MONTE CARLO TREE SEARCH IN OTHELLO

Daniel Qiu

Thomas Jefferson High School
Alexandria, VA 22312, USA

Jie Xu

Dept. of Systems Eng. & Operations Research
George Mason University
Fairfax, VA 22030, USA

ABSTRACT

Upper Confidence bounds applied to Trees (UCT) is the most popular tree policy for Monte Carlo Tree Search (MCTS). However, UCT focuses on minimizing cumulative regret rather than maximizing the Probability of Correct Selection (PCS) of the best action, which is often preferred in game engines. To address this, we examine an Optimal Computing Budget Allocation (OCBA) tree policy that provides a rigorous way for maximizing the PCS rather than minimizing regret. MCTS-OCBA has been shown to work well with simple games such as Tic-Tac-Toe, where the search space is small enough to simulate through, but not unsolved games such as Othello or Go. We report numerical results showing that MCTS-OCBA performs better in Othello than MCTS-UCT and thus demonstrate OCBA is a more efficient tree policy for MCTS for game engines.

1 INTRODUCTION

MCTS performs a tree search using simulations on a current game state to identify the best move. It is the algorithm used by Google DeepMind's AlphaGo that beat world champion Lee Sedol (Silver et al. 2016). MCTS has 4 steps: selection, expansion, simulation, and backpropagation. The goal of selection is to use a tree policy to select a leaf node of a game tree for simulation evaluation. The goal of expansion is to add a new leaf node to the game tree. The goal of simulation is to use a default policy to simulate the game until its end. The goal of backpropagation is to update information for the tree. A tree policy controls how the tree is grown and is of fundamental importance. There has recently been a proposed algorithm that focuses on maximizing the PCS of the best move called OCBA (Li et al. 2022), which is achieved by solving an optimization problem with PCS as the objective function. While MCTS-OCBA has been demonstrated to outperform MCTS-UCT on Tic-Tac-Toe (Li et al. 2022), its performance has yet to be tested on an unsolved game. We do so on an Othello game engine, which is an unsolved game widely used in computer game literature (Browne et al. 2012).

2 METHODOLOGY AND RESULTS

2.1 MCTS-UCT vs MCTS-OCBA

In MCTS-UCT, the child node chosen is the node that maximizes $\frac{w_i}{n_i} + c\sqrt{\frac{\ln N}{n_i}}$, where w_i is the sum of all values of simulations starting from that node, n_i is the number of visits to that node, c is a constant, and N is the number of visits to the parent node. In our code, we set c to 1, and calculate the value of a simulation by using the formula $\frac{a-b}{a+b}$, where a is the number of the player's tokens and b is the number of the opponent's tokens. In MCTS-OCBA, we denote by I_x a set that contains the nodes that can be reached from state x, and let δ_i be the difference between the performance of the best node $\hat{i}_x^*$ and node i, and σ_i^2 be the variance of the simulation noise. We use $\tilde{n}_i$ to denote the number of simulations to be allocated to

node i. A key feature of MCTS-OCBA is the role of σ^2 as in the following equations for solving $\tilde{n}_i$:

$$\forall i, j \in I_x, i, j \neq \hat{i}_x^* : \frac{\tilde{n}_j}{\tilde{n}_i} = \left(\frac{\sigma_j / \delta_j}{\sigma_i / \delta_i} \right)^2 \text{ and } \tilde{n}_{\hat{i}_x^*} = \sigma_{\hat{i}_x^*} \sqrt{\sum_{i \in I_x, i \neq \hat{i}_x^*} \frac{\tilde{n}_i^2}{\sigma_i^2}}.$$

We update σ_i and δ_i as simulation data is collected and simulate the next node $\hat{i_x} = \arg\max_{i \in I_x} (\tilde{n}_i - n_i)$.

2.2 Simulation Policy

We test two simulation policies: i) random play, ii) a heuristic policy. The heuristic policy builds upon Othello domain knowledge (Lee and Mahajan 1990) and uses the number of corners, X-squares (the squares directly diagonal to corners), number of moves, and potential mobility (p-mob, the total number of empty spaces next to opponent pieces) to estimate the desirability of a move:

$$\begin{aligned}(\#\text{ of your corners} - \#\text{ of enemy corners}) * 400 + (\#\text{ of enemy X-squares} - \#\text{ of your X-squares}) * 100 \\ + (\#\text{ of your moves}) / (\#\text{ of opponent moves}) * 50 + (\text{your p-mob}) / (\text{enemy p-mob}) * 80.\end{aligned}$$

2.3 Results

Table 1 reports numerical results. MCTS-OCBA consistently outperformed MCTS-UCT with random simulation policy, delivering 60.2% wins, 3.4% ties with 50 simulations, 54.0% wins, 2.2% ties with 100 simulations, and 52.0% wins, 3.8% ties for 500 simulations. MCTS-OCBA's performance is also better than MCTS-UCT with simulation heuristic (60.8% wins and 2.2% ties with 50 simulations, 59% wins and 1.4% ties with 100 simulations, and 50.8% wins and 3.6% ties with 500 simulations). MCTS-OCBA's advantage is larger when simulation budget is relatively small or when using heuristic rollout policy.

Table 1: Experimental Results from 250 games (ties not reported)

(White vs Black, 250 games)	No Heuristic	Heuristic
MCTS-OCBA vs MCTS-UCT (50 simulations)	149 - 93	144 - 102
MCTS-UCT vs MCTS-OCBA (50 simulations)	89 - 152	83 - 160
MCTS-OCBA vs MCTS-UCT (100 simulations)	142 - 102	149 - 96
MCTS-UCT vs MCTS-OCBA (100 simulations)	117 - 128	102 - 146
MCTS-OCBA vs MCTS-UCT (500 simulations)	130 - 111	134 - 109
MCTS-UCT vs MCTS-OCBA (500 simulations)	110 - 130	119 - 120

3 CONCLUSION

We reported new numerical results that help illustrate MCTS-OCBA is a better tree policy than MCTS-UCT in an unsolved game like Othello and show the performance improvement is robust with different simulation policies. Further work could be done by building state-of-the-art game engines with MCTS-OCBA.

REFERENCES

Browne, C., E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. "A Survey of Monte Carlo Tree Search Methods". *IEEE Transactions on Computational Intelligence and AI in Games* 4:1–49.

Lee, K., and S. Mahajan. 1990. "The Development of a World Class Othello Program". *Artificial Intelligence* 43:21–35.

Li, Y., M. C. Fu, and J. Xu. 2022. "An Optimal Computing Budget Allocation Tree Policy for Monte Carlo Tree Search". *IEEE Transactions on Automatic Control* 67:2685–2699.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search". *Nature* 529(7587):484–489.