Monte Carlo Tree Search with Metaheuristics

Jacek Mańdziuk, Patryk Walczak

Faculty of Mathematics and Information Science
Warsaw University of Technology, Warsaw, Poland
jacek.mandziuk@pw.edu.pl, patryk.walczak.stud@pw.edu.pl

ICAISC 23

Abstract

Monte Carlo Tree Search/Upper Confidence bounds applied to Trees (MCTS/UCT) is a popular and powerful search technique applicable to many domains, most frequently to searching game trees. Even though the algorithm has been widely researched, there is still room for its improvement, especially when combined with metaheuristic or machine learning methods. In this paper, we revise and experimentally evaluate the idea of enhancing MCTS/UCT with game-specific heuristics that guide the playout (simulation) phase. MCTS/UCT with the proposed guiding mechanism is tested on two popular board games: Othello and Hex. The enhanced method clearly defeats the well-known Alpha-beta pruning algorithm in both games, and for the more complex game (Othello) is highly competitive to the vanilla MCTS/UCT formulation.

Monte Carlo Tree Search

MCTS is an iterative algorithm that searches the state space and builds statistical evidence about the decisions available in particular states. Building a game tree starts with a single node (the root) representing the current position. Next, multiple iterations are performed, each of them consisting of four phases, depicted in Figure 1.

1. **Selection** - In the first phase, the algorithm searches the portion of the tree that has already been represented in memory. Selection always starts from the root node and, at each level, selects the next node according to the *selection policy*. *Selection* terminates when a leaf node is reached.
2. **Expansion** - Unless the *selection* process reached a terminal state, the *expansion* phase takes place, which adds a new child node to the currently represented tree. The new node is a child of the node finally reached in the *selection* phase.
3. **Simulation** - In this phase, a **fully random** simulation of the game/problem is performed from the newly-added node, which reaches a terminal state of the game and calculates the players' scores/payoffs.
4. **Backpropagation** - This phase propagates back the payoff of the player represented in the root node along the path from the newly-added leaf node in the tree (the *expansion* node) to the root, and updates the nodes' statistics ($Q(s,a), N(s,a), N(a)$, described below).

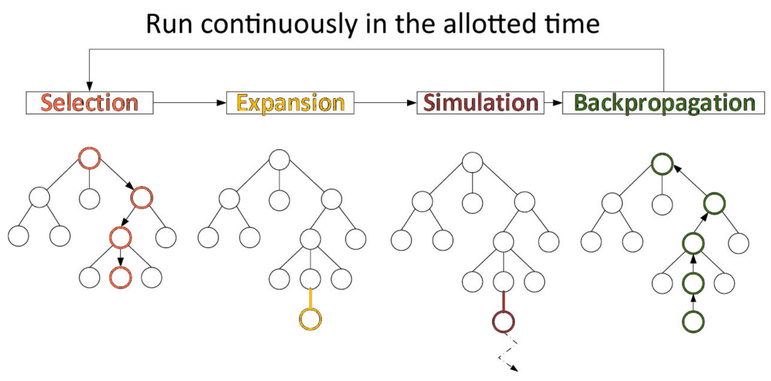


Figure 1: *Monte Carlo Tree Search phases*

Upper Confidence bounds applied to Trees (UCT)

UCT is typically used as the basic *selection policy* in MCTS. The algorithm attempts to balance the exploration (of yet not well tested actions) and the exploitation (of the best actions found so far). In a given node, UCT advises to first check each possible action once and then follow Eq. 1:

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{ln\,(N(s))}{N(s,a)}} \right\} \tag{1}$$

Rapid Action-Value Estimation (RAVE)

Using Q_{RAVE} statistics speeds up the MCTS/UCT state-action estimations. $Q(s,a)$ in Eq. 1 is redefined as follows:

$$Q(s,a) := \beta(s,a) \times Q_{RAVE}(s,a) + (1-\beta(s,a)) \times Q(s,a); \quad \beta(s,a) = \sqrt{\frac{k}{3N(s,a)+k}} \tag{2}$$

where k is a parameter that defines the number of simulations with action a played in state s at which $Q_{RAVE}(s,a)$ and $Q(s,a)$ are given equal weight, i.e. $\beta(s,a) = 1/2$.

MCTS/UCT with *Strategy Switching*

The proposed extension of the MCTS/UCT algorithm utilizes a set of provided heuristics to modify the *simulation (playout)* phase of MCTS. A critical component of SS is proper selection of a **playout strategy** (*policy*) in the current simulation. We follow a solution initially proposed by the first author in the context of General game Playing. Namely, a selection is performed according to the UCB1 formula (Eq. 3), i.e. using the same exploration-exploitation balancing mechanism as the one used in MCTS/UCT in the *Selection* phase:

$$s^* = \arg\max_{s \in S} \left\{ Q(s,n) + b\sqrt{\frac{ln(n)}{T(s,n)}} \right\} \tag{3}$$

In Eq. 3, s^* is the strategy selected in the current simulation, n is the number of completed simulations, $Q(s,n)$ and $T(s,n)$ are the average score of strategy s in n simulations and the number of simulations that have used s in n performed simulations, respectively. b is the exploration-exploitation balancing factor.

Algorithm 1: *Simulation* phase of MCTS/UCT algorithm with SS

$policy$ = strategy switching algorithm chooses the strategy;
while $node$ is not a leaf **do**
  $node$ = a child of $node$ chosen according to $policy$;

Experiments

Two versions of the SS algorithm have been tested: (1) **MCTS/UCT/SS(all)** - all heuristics considered for a given game were available to choose from; (2) **MCTS/UCT/SS(top3)** - only the top 3 most effective heuristics (devised in some number of preliminary tests) were available. The following selection of methods was used in comparative tests to assess both MCTS/UCT/SS versions:

1. **MCTS/UCT** – vanilla approach;
2. **MCTS/UCT/E** – the playout phase is guided by the evaluation function used in Alpha-beta – the bigger its value, the higher the assessment of a given node;
3. **MCTS/UCT/Mp** – the playout phase is guided by the player's mobility – the more actions are available, the higher the assessment of a given node;
4. **MCTS/UCT/Md** – the playout phase is guided by the difference between player's and opponent's mobilities – the bigger the difference, the higher the assessment of a given node;
5. **MCTS/UCT/RAVE** – in the selection phase, UCT is replaced by UCT/RAVE (cf. Eq. 2);

6.-7. **Alpha-beta(10)**, **Alpha-beta(12)** – Alpha-beta pruning with depth 10 and 12, resp.

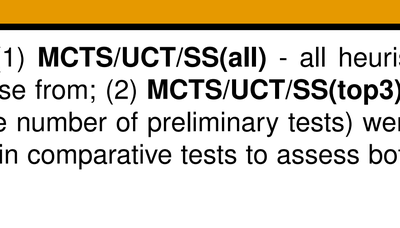
In the last three tested methods, there is no tree search involved, and the move is chosen directly based on:

8. **E** – the evaluation function used in Alpha-beta;
9. **R** – uniform distribution;
10. **WPC** – the WPC evaluation – only in Othello;

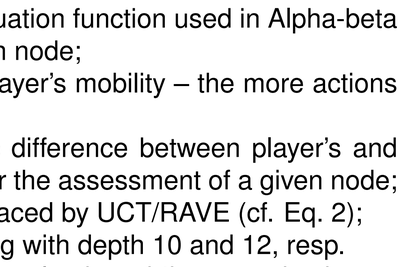
Results

Table 1: *Hex results. The top 3 heuristics used in the MCTS/UCT/SS(top3) setting were player's mobility, the difference between the player's and the opponent's mobilities and simple game state evaluation. In rows 7 and 9 the evaluation function is Dijkstra's algorithm (the same as in the Alpha-beta algorithm).*

No	Algorithm	Win	Loss	Avg. Time (h)
1	MCTS/UCT	88,42%	11,58%	4,85
2	MCTS/UCT/SS(all)	72,73%	27,27%	17,52
3	MCTS/UCT/SS(top3)	64,72%	35,28%	21,95
4	MCTS/UCT/Mp	64,67%	35,33%	18,10
5	Alpha-beta(12)	64,61%	35,39%	6,64
6	Alpha-beta(10)	64,61%	35,39%	6,95
7	MCTS/UCT/E	52,42%	47,58%	12,89
8	MCTS/UCT/Md	52,17%	47,83%	31,66
9	E	37,08%	62,92%	6,69
10	R	21,50%	78,50%	7,50
11	MCTS/UCT/RAVE	12,00%	88,00%	9,89



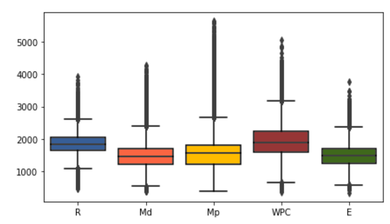
(a) All strategies are available



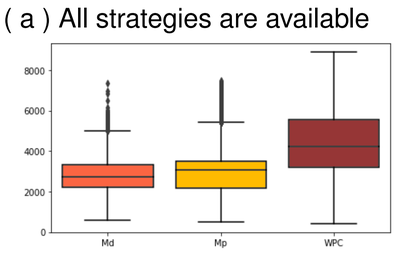
(b) Only top 3 strategies are available

Table 2: *Othello results. The top 3 heuristics used in the restricted MCTS/UCT/SS approach were player's mobility, the difference between the player's and the opponent's mobilities, WPC evaluation. In rows 2 and 10 the evaluation function is the difference between the numbers of the player's and opponent's pieces (the same as in the Alpha-beta algorithm).*

No	Algorithm	Win	Draw	Loss	Avg. Time (s)
1	MCTS/UCT/SS(top3)	81,58%	1,75%	16,67%	1127,28
2	MCTS/UCT/E	79,67%	2,00%	18,33%	2061,46
3	MCTS/UCT/WPC	77,25%	2,25%	20,50%	504,98
4	MCTS/UCT/SS(all)	71,00%	1,17%	27,83%	1203,60
5	MCTS/UCT/Mp	69,67%	1,92%	28,42%	1316,78
6	MCTS/UCT	55,67%	1,92%	42,42%	524,69
7	MCTS/UCT/RAVE	42,33%	1,33%	56,33%	506,90
8	Alpha-beta(12)	19,67%	0,25%	80,08%	374,12
9	Alpha-beta(10)	19,67%	0,25%	80,08%	381,77
10	E	17,00%	0,25%	82,75%	390,81
11	R	9,58 %	1,08%	89,33%	421,08



(a) All strategies are available



(b) Only top 3 strategies are available

Conclusions

- In simple games, such as Hex, enhancing MCTS/UCT with SS does not seem to have much reason. Since MCTS/UCT is sufficiently strong method for this game, making the approach more complex, actually deteriorates the results.
- In more complex problems, e.g. the game of Othello, MCTS/UCT/SS shows its advantages, leading to savings of time and resources.
- The more effective of the two MCTS/UCT/SS setups is the one based on top 3 (generally top n) heuristics, as opposed to using all of them. This observation supports the intuition that selection among a smaller set of stronger strategies is generally more effective (easier) than the selection from a larger set, which also contains relatively weaker strategies.
- MCTS/UCT supported with a proper set of heuristic strategies outperforms the well-known Alpha-beta algorithm, with the evaluation function relying on basic material assessment.
- MCTS/UCT with a lightweight evaluation function (either assigned directly or via the SS mechanism) is clearly a more powerful approach than a direct application of CTS with the same heuristics to move selection. within the entire game tree, including the *selection* phase.