Nested Monte-Carlo Tree Search for Online Planning in Large MDPs[1]

Hendrik Baier Mark H. M. Winands

Games and AI Group, Department of Knowledge Engineering, Maastricht University, The Netherlands

Monte-Carlo Tree Search (MCTS) [3, 5] is a promising choice for online planning in large MDPs. It is a best-first, sample-based search algorithm in which every state in the search tree is evaluated by the average outcome of Monte-Carlo rollouts from that state. Since MCTS is based on sampling, it does not require a transition function in explicit form, but only a generative model of the domain. Because it grows a highly selective search tree guided by its samples, it can handle search spaces with large branching factors. By using Monte-Carlo rollouts, MCTS can take long-term rewards into account even with distant horizons. Combined with multi-armed bandit algorithms to trade off exploration and exploitation, MCTS has been shown to guarantee asymptotic convergence to the optimal policy [5], while providing approximations when stopped at any time.

For the consistency of MCTS, i.e. for the convergence to the optimal policy, uniformly random rollouts beyond the tree are sufficient. However, heuristically informed rollout strategies typically speed up convergence [4]. In this paper, we propose *Nested Monte-Carlo Tree Search* (NMCTS), using the results of lower-level searches recursively to provide rollout policies for searches on higher levels.

So far, no nested search algorithm has made use of the selectivity and exploration-exploitation control that MCTS provides. In the context of MCTS, nested search has so far only been performed offline to provide opening databases for the underlying online game playing agent. The different levels of search therefore used different tree search algorithms adapted to their respective purpose, and nested and regular MCTS have not been compared on the same task. In this extended abstract, we propose Nested Monte-Carlo Tree Search (NMCTS) as a general online planning algorithm for MDPs.

We define a level-0 *Nested Monte-Carlo Tree Search* (NMCTS) as a single rollout with the base rollout policy—either uniformly random, or guided by a simple heuristic. A level-1 NMCTS search corresponds to MCTS, employing level-0 searches as state evaluations. A level-n NMCTS search for $n \geq 2$ recursively utilizes the results of level-$(n-1)$ searches as evaluation returns.

As the selection, expansion and backpropagation steps of MCTS are preserved in NMCTS, many successful techniques from MCTS research such as the UCB1-TUNED selection policy can be applied in NMCTS as well. Parameters can be tuned for each level of search independently.

In some domains, it is effective not to spend the entire search time on the initial position of a problem, but to distribute it over all actions in the episode (or the first z actions). Search and execution are thus interleaved. We call this technique *action-by-action search* as opposed to *global search*, and it is optionally applicable at all levels of NMCTS. In case NMCTS is used with action-by-action search, a decision has to be made which action to choose and execute at each step of the search. Two possible options are a) choosing the most-sampled action—as traditionally done in MCTS—, or b) choosing the next action in the overall best solution found so far. Setting NMCTS to action-by-action search, using only one rollout per legal action in each action search, and then choosing the next action of the best known solution leads to NMCS [2] as a special case of NMCTS. This special case does not provide for an exploration-exploitation tradeoff, nor does it build a tree going deeper than the number of nesting levels used, but it allows relatively deep nesting due to the low number of rollouts per search level.

[1]This work is funded by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project Go4Nature, grant number 612.000.938.

The full version of this paper is accepted at the 20th European Conference on Artificial Intelligence (ECAI 2012).

We tested Nested Monte-Carlo Tree Search on three different deterministic, fully observable MDPs: The puzzles named “SameGame”, “Clickomania” and “Bubble Breaker”. These domains have identical transition functions, but different reward functions, resulting in different distributions of high-quality solutions. The decision problem associated with these optimization problems is NP-complete [1].

We compared regular MCTS and level-2 NMCTS in all three domains, using a random rollout policy. For SameGame, we also employed a state-of-the-art informed rollout policy, consisting of the TabuColorRandomPolicy [6] (setting a “tabu color” at the start of each rollout that is not chosen as long as groups of other colors are available) in combination with a multi-armed bandit learning the best-performing tabu color for the position at hand (based on UCB1-TUNED).

As it has been shown for SameGame that restarting several short MCTS runs on the same problem can lead to better performance than a single, long run [6], we tested several numbers of randomized restarts for MCTS and tuned the selection policy for each of them. The same settings were then used for NMCTS, with the number of nested level-1 NMCTS searches equivalent to the number of restarts for multi-start MCTS.

Results show that in Bubble Breaker and SameGame—in the latter using both random and informed rollouts—level-2 NMCTS significantly outperformed multi-start MCTS in all experimental conditions. The best results in SameGame were achieved building a level-2 tree out of 36,480 level-1 searches of 250 ms each, with informed base-level rollouts. In comparison to the best performance of multi-start MCTS, achieved with 2280 restarts of 4-second searches, the use of a nested tree increased the average best solution per position from 3395.9 to 3465.96. As a comparison, a doubling of the search time to 4560 restarts only resulted in a performance increase to 3431.0.

In Clickomania, level-2 NMCTS also achieved the highest score. While the results of multi-start MCTS for different numbers of restarts suggest that a single, global MCTS search could perform relatively well in Clickomania, memory limitations reduced the effectivity of this approach. NMCTS however is able to constantly reuse tree nodes of lower-level searches, and therefore suffers less from this problem. We observed that the best-performing NMCTS setting tested used less than 15% of memory of what a single, global MCTS search would have required for optimal performance.

In further experiments, we compared level-2 NMCTS to level-3 NMCS. Here, NMCTS used action-by-action search on level 2, and advanced from action to action by choosing the next action of the best solution found so far. NMCS was not able to complete a level-3 search in the given time; consequently, the best solutions found after the given computation time had elapsed were used for the comparisons. NMCTS outperformed NMCS in SameGame with random playouts, SameGame with informed playouts and Clickomania. For Bubble Breaker, manual tuning has not revealed parameter settings superior to NMCS yet. Automatic parameter tuning is in preparation.

In conclusion, empirical results in the test domains of SameGame, Bubble Breaker and Clickomania show that NMCTS significantly outperforms regular MCTS. Experiments in SameGame and Clickomania suggest performance superior to NMCS. Since both MCTS and NMCS represent specific parameter settings of NMCTS, correct tuning of NMCTS has to lead to greater or equal success in any MDP domain.

References

[1] T. C. Biedl, E. D. Demaine, M. L. Demaine, R. Fleischer, L. Jacobsen, and J. I. Munro. The Complexity of Clickomania. In R. J. Nowakowski, editor, *More Games of No Chance, Proceedings of the MSRI Workshop on Combinatorial Games*, pages 389–404, 2002.

[2] T. Cazenave. Nested Monte-Carlo Search. In C. Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 456–461, 2009.

[3] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, editors, *5th International Conference on Computers and Games (CG 2006). Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2007.

[4] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. Technical report, HAL - CCSd - CNRS, France, 2006.

[5] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *17th European Conference on Machine Learning, ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2006.

[6] M. P. D. Schadd, M. H. M. Winands, H. J. van den Herik, G. M. J.-B. Chaslot, and J. W. H. M. Uiterwijk. Single-Player Monte-Carlo Tree Search. In H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Proceedings of the 6th International Conference on Computers and Games*, pages 1–12, 2008.