Monte Carlo *-Minimax Search[1]

Marc Lanctot[a] Abdallah Saffidine[b] Joel Veness[c] Chris Archibald[c]
Mark H.M. Winands[a]

[a] *Department of Knowledge Engineering, Maastricht University, The Netherlands*
[b] *LAMSADE, Université Paris-Dauphine, France*
[c] *Department of Computing Science, University of Alberta, Canada*

1 Introduction

Monte Carlo sampling in game-tree search has received much attention in recent years due to successful application to Go-playing programs. While the community has focused mainly on deterministic two-player games, such as Go, Hex, and Lines of Action, there has been a growing interest in studying these sample-based approaches outside this traditional setting. The class of perfect information games with chance events– which includes, for example, Backgammon– has received comparatively little attention.

Classic algorithms such as minimax perform a depth-limited search from the root (current position), returning a heuristic value if the depth limit is reached, or the value of the best available move otherwise. $\alpha\beta$ pruning prevents searching provably wasteful portions of the tree. The largest and most famous application of these techniques was in IBM's Deep Blue chess program which defeated the human world champion. Expectimax is an extension of minimax that will return the expected values over children at chance nodes [4]. The *-minimax algorithm extends $\alpha\beta$ pruning to perfect information games with chance nodes [1].

2 Sparse Sampling in *-Minimax Search

Monte Carlo *-Minimax Search (MCMS) samples a subset of the chance event outcomes at chance nodes during its search. In essence, the algorithm applies *-minimax (Star1 or Star2) search to a sampled and significantly smaller subgame to effectively increase the depth reached in a fixed time limit. This way of using sampling to reduce computation is inspired by sparse sampling methods from the MDP planning literature [3] and is in contrast with recent Monte Carlo search algorithms such as Monte Carlo Tree Search (MCTS) [2], which are simulation-based and build a model of the game tree incrementally.

Consider Figure 1. Suppose the number of chance event outcomes is N. For example, in a game where players roll two six-sided dice, it may be that $N = 36$. Suppose the algorithm returns a value of v_i for the subtree below outcome i, and the probability of outcome i is p_i. Expectimax and *-Minimax will return the weighted sum $\sum_{i=1}^{N} v_i p_i$. MCMS, however, first samples $c < N$ outcomes (with replacement), sets $p'_i = 1/c$ and returns $\sum_{i=1}^{c} p'_i v'_i$, where v'_i is the value that MCMS returns for the subtree under outcome i.

3 Results and Remarks

In the full paper, we show that the value returned by MCMS approaches the value computed by *-Minimax as the sample width, c, increases. Furthermore, the convergence does not depend on the number of states.

[1]Extended abstract of: M. Lanctot, A. Saffidine, J. Veness, C. Archibald and M.H.M. Winands. Monte Carlo *-minimax search. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 580–586, 2013.

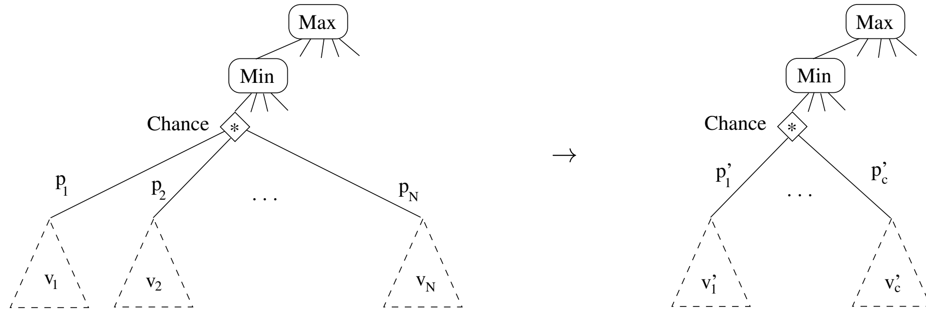

Figure 1: Example of sampling in MCMS.

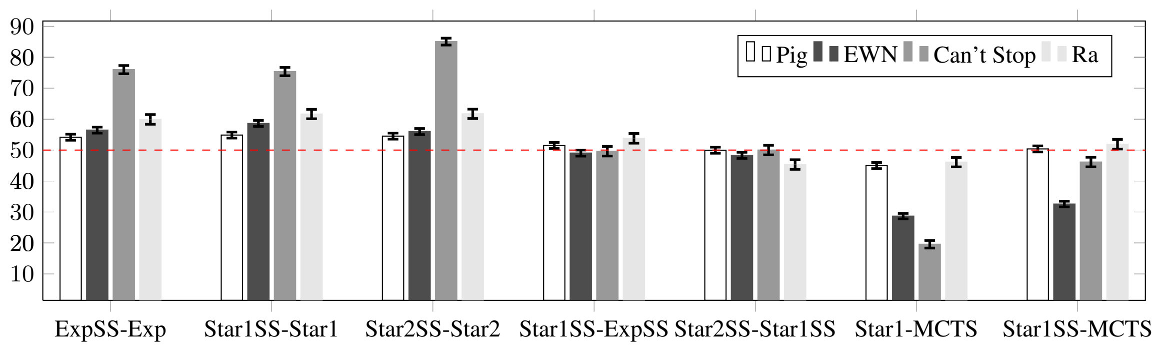

Figure 2: Results of playing strength experiments. Each bar represents the percentage of wins for p_{left} in a p_{left}-p_{right} pairing. (Positions are swapped and this notation refers only to the name order.) Errors bars represent 95% confidence intervals. Here, Exp refers to expectimax, XSS refers to algorithm X with sparse sampling, and Star1 and Star2 represent two different pruning variants of *-minimax.

In practice, MCMS is shown to exhibit lower regret and bias than *-minimax on Pig. This comes at a cost of increased variance due to sampling. As seen in Figure 2 results across four games (Pig, EinStein würfelt Nicht! (EWN), Can't Stop, and Ra) show that MCMS (ExpSS, Star1SS, or Star2SS) consistently outperforms its classic counterpart (expectimax, Star1, or Star2). When playing against MCTS, MCMS wins 4-16% more than *-minimax. MCMS is also competitive against state-of-the-art MCTS in two of the four chosen games, outperforming MCTS in Ra.

References

[1] B.W. Ballard. The *-minimax search procedure for trees containing chance nodes. *Artificial Intelligence*, 21(3):327–350, 1983.

[2] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the 5th international conference on Computers and games*, pages 72–83. Springer-Verlag, 2007.

[3] M.J. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. In *IJCAI*, pages 1324–1331, 1999.

[4] D. Michie. Game-playing and game-learning automata. *Advances in Programming and Non-numerical Computation*, pages 183–196, 1966.