

Monte Carlo *-Minimax Search: An Overview

First author ^a

Second author ^a

Third author ^a

^a *Department of Knowledge Engineering, Maastricht University,
P.O. Box 616, 6200 MD Maastricht, The Netherlands*

Abstract

This paper presents an overview of Monte Carlo *-Minimax Search (MCMS), a Monte Carlo search algorithm for turned-based, stochastic, two-player, zero-sum games of perfect information [3]. MCMS combines sparse sampling techniques used in MDP planning with classic pruning techniques developed for adversarial expectimax planning. MCMS has been compared to the traditional *-Minimax approaches, as well as MCTS on four games: Pig, EinStein Würfelt Nicht!, Can't Stop, and Ra. Results show that MCMS can be compete with enhanced MCTS variants in some domains, while outperforming the equivalent classic approaches given the same amount of thinking time.

1 Introduction

Monte Carlo sampling in game-tree search has received much attention in recent years due to successful applications 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 is commonly applied to prevent 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. The *-minimax algorithm extends $\alpha\beta$ pruning to perfect information games with chance nodes [1].

1.1 Sparse Sampling in *-Minimax Search

Monte Carlo *-Minimax Search (MCMS) samples a subset of the chance event outcomes at chance nodes during its search [3]. 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 [2] and is in contrast with recent Monte Carlo search algorithms such as Monte Carlo Tree Search (MCTS), 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 $N = 36$. The 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 .

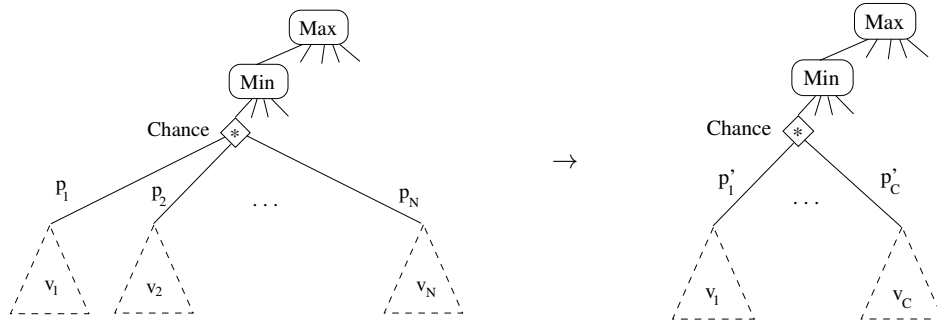


Figure 1: Example of sampling in MCMS

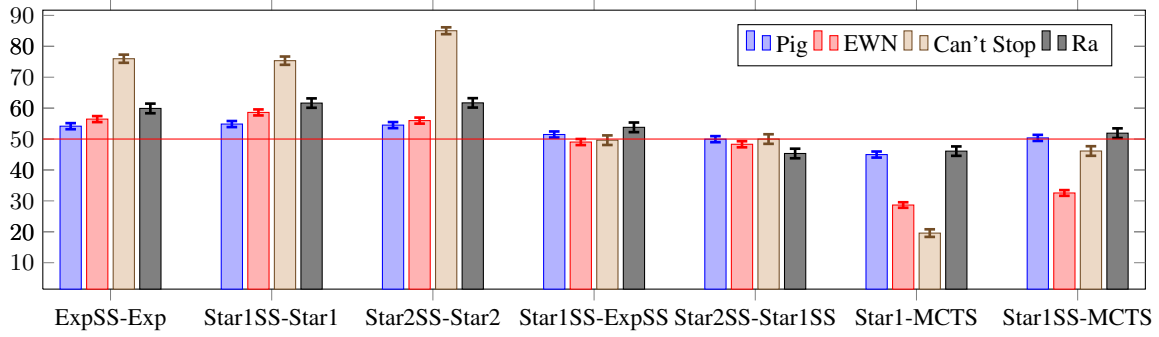


Figure 2: Results of playing strength experiments. Each bar represents the percentage of wins for p_{left} in a $p_{\text{left}}-p_{\text{right}}$ pairing. (Positions are swapped and this notation refers only to the name order.) Errors bars represent 95% confidence intervals. Here, Exp refers to expectimax,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

2 Results and Remarks

In [3], 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. In practice, MCMS is shown to exhibit lower regret and bias than $*$ -minimax on a small, solvable game. This comes at a cost of increased variance due to sampling. As seen in Figure 2, results across four games show that MCMS (ExpSS, Star1SS, or Star2SS) consistently outperforms its classic counterpart (expectimax, Star1, or Star2). When played 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] M.J. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1331, 1999.
- [3] Marc Lanctot, Abdallah Saffidine, Joel Veness, Chris Archibald, and Mark Winands. Monte carlo $*$ -minimax search. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.