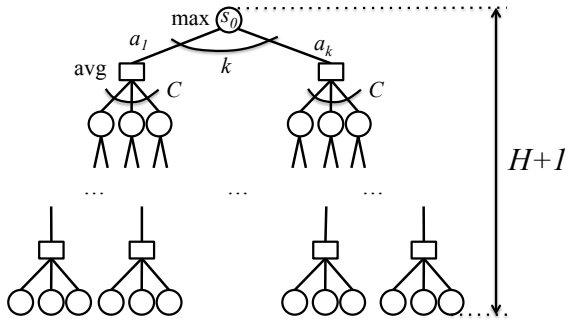
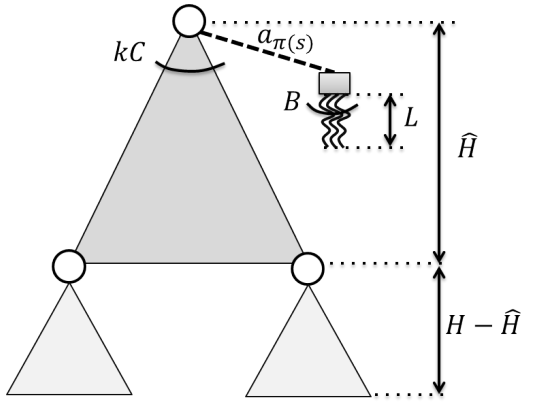


(a) Sparse Sampling



(b) SS-Aux

Figure 1: The look-ahead trees of SS and SS-Aux; each state node (circles) estimates V_h^{SS} while its state-action children (rectangles) estimate Q_h^{SS} for each height $h \in [H-\hat{H}, H]$.

sampling methods use a similar tree-like structure as SS, so there is potential to enhance them using the Aux method. The latest in this series is the Forward Search Sparse Sampling (FSSS) (Walsh, Goschin, and Littman 2010) that tries to smartly allocate the computation to promising branches of the look-ahead tree in order to more quickly approximate the optimal policy in large domains. In this section, we investigate the application of the Aux method to enhance FSSS.

Forward Search Sparse Sampling (FSSS)

FSSS is the latest successor of SS that is able to combine the performance guarantee of SS with selective sampling (Walsh, Goschin, and Littman 2010). Instead of directly forming point estimates of the Q- and V-values in a look-ahead tree, it maintains interval estimates that allow it then to guide the sampling to the promising and/or unexplored branches of the look-ahead tree based on the widths and upper bounds of the intervals.

Similar to SS, FSSS also constructs a tree of height H and branching factor k. However, instead of constructing the whole tree in one go, FSSS traverses the tree from root down using simulated episodes of length H called *trials*. At each state node s of the trial, the algorithm *expands* the state by simulating all the k actions C times. Instead of then recursing to all the next states like SS, the FSSS picks the action a^* with a highest upper-bound for Q(s,a) and *selects* the next state s^* with the widest interval for V(s') among the states s' that were obtained by simulating a^* at s. The selected state s^* is then expanded until the leaf node is reached, after which the interval bounds of V- and Q-values are *updated* recursively on the path from leaf to root. When there is an action at the root with the Q lower bound greater than all other arms' upper bounds, i.e., the action is surely more highly rewarding than its siblings, the algorithm terminates.

As such, FSSS may also skip subtree expansions at branches where no further exploration would be helpful. When no pruning occurs, FSSS requires at most $(kC)^H$ trials. However, its running time may be up to $O(H(kC)^H)$.

Proposition 1 *The running time of FSSS is bounded by* $O(H(kC)^H)$.

Proof Sketch. At each non-leaf state node, *expansion* samples the simulator kC times, thus incurring a cost of $O((kC)^H)$ in total. In one FSSS trial, *selection* and *update* take $O(H)$, so since there are at most $(kC)^H$ trials, altogether these two actions cost $O(H(kC)^H)$. The total running time is therefore dominated by $O(H(kC)^H)$. □

This means that in the worst case FSSS may require longer running time than SS.

FSSS-Aux: FSSS with π-guided auxiliary arms

Similar to SS-Aux, we add one auxiliary arm at every internal state node of the tree up to a certain depth. At each of these auxiliary arms, the lower and upper bounds are computed using B π-guided rollouts of length L. The procedure is detailed in Algorithm 2 with the *aux* flag set to true and $EstimateQ(s, a, \pi)$ being the average bounds returned by π-guided simulations. Similar to the behavior of SS-Aux, it is straightforward to show that in the worst case, when auxiliary arms do not help to prune any branch, FSSS-Aux yields the same estimation as that of FSSS. In the good case when the heuristic policy is near-optimal, we expect many state nodes have their lower-upper bound gap closed faster. In fact, the following proposition states that, after terminating, FSSS-Aux is as good as SS-Aux.

Proposition 2 *On termination, the action chosen by FSSS-Aux is the same as that chosen by SS-Aux.*

Proof sketch. Note that if there is no pruning possible, FSSS-Aux expands the same tree as SS-Aux. Otherwise, upon termination, there is no point for further tightening the bounds because one arm surely has higher value than the rest, as its lower bound exceeds the upper bounds of other arms. This means that the selected arm is guaranteed to be the same as the one selected by SS-Aux, which expands the full tree. □

Note that however at premature termination, there is no guarantee on the performance of FSSS as compared to that of SS (or FSSS-Aux to SS-Aux for that matter) due to their dissimilarity in exploring the state space. Specifically, FSSS uses upper-lower bound gap to guide the search, while SS distributes sampling equally among all child nodes to collect rewards' statistics. As such, they may discover the optimal values at different time points.

Chang, H. S.; Fu, M. C.; Hu, J.; and Marcus, S. I. 2005. An adaptive sampling algorithm for solving Markov Decision Processes. *Operations Research* 53(1):126–139.

Chaslot, G.; Fiter, C.; Hoock, J. B.; Rimmel, A.; and Teytaud, O. 2010. Adding expert knowledge and exploration in Monte-Carlo Tree Search. *Advances in Computer Games* 1–13.

Finnsson, H., and Björnsson, Y. 2008. Simulation-based approach to General Game Playing. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*, 259–264. AAAI Press.

Gelly, S., and Silver, D. 2007. Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, 273–280. New York, NY, USA: ACM.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 2002. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. *Machine Learning* 49(2):193–208.

Keller, T., and Helmert, M. 2013. Trial-based Heuristic Tree Search for Finite Horizon MDPs. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS '13)*, 135–143.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo Planning. In *Proceedings of the 17th European conference on Machine Learning (ECML '06)*, ECML'06, 282–293. Berlin, Heidelberg: Springer-Verlag.

Nguyen, T.-H. D.; Lee, W.-S.; and Leong, T.-Y. 2012. Bootstrapping Monte Carlo Tree Search with an Imperfect Heuristic. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD'12)*, 164–179. Berlin, Heidelberg: Springer-Verlag.

Péret, L., and Garcia, F. 2004. On-line search for solving Markov Decision Processes via heuristic sampling. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, 530–534.

Walsh, T. J.; Goschin, S.; and Littman, M. L. 2010. Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*.