# Bidirectional Online Probabilistic Planning

**Aswin Nadamuni Raghavan** and **Saket Joshi** and **Alan Fern** and **Prasad Tadepalli**

School of Electrical Engineering and Computer Science
Oregon State University, Corvallis, OR 97331

We present Bidirectional Online Probabilistic Planner (BOPP)[1], a novel planner that combines elements of Decision Theoretic Planning(DTP) and forward search. In particular, BOPP uses a combination of SPUDD and Upper Confidence Trees(UCT). We present our approach and some experimental results on the domains presented in the boolean fluents MDP track of the International Probabilistic Planning Competition(IPPC) 2011.

Decision Theoretic Planning (DTP) (Boutilier, Dean, and Hanks 1999) is a well established method for solving probabilistic planning domains by casting them as Markov Decision Processes (MDP) and generating a policy. Classical solutions to DTP (Bellman 1957; Howard 1960) require the entire state space to be enumerated. This approach is usually infeasible for solving planning problems of interest. Recent advances in DTP have mitigated this effect by providing solution algorithms for factored (Boutilier, Dearden, and Goldszmidt 1999) and relational (Boutilier, Reiter, and Price 2001) MDPs. These solutions are abstract and require enumeration only of the relevant conditions that create a partition of the state space into equivalence classes (based on the policy or value), instead of enumerating the entire state space. One factored MDP solver in particular, SPUDD (Hoey et al. 1999), has been very successful at solving planning problems and has spawned numerous variants over the last decade. SPUDD employs Algebraic Decision Diagrams to represent and solve the underlying MDP using value iteration. However, experiments have shown that SPUDD proves to be inefficient for many of the planning problems presented in the recent IPPC.

Forward Search has been another classic approach for AI planning. Brute force search is typically infeasible for large state spaces because the size of the search tree is exponential in the depth (length of the plan). Planners based on heuristic search, however, have shown success at the recent planning competitions (Bonet and Geffner 2001; Yoon, Fern, and Givan 2007; Teichteil-Koenigsbuch, Infantes, and Kuter 2008). The heuristic values of states or state-action pairs are typically derived automatically by solving a relaxation to the probabilistic planning problem. More recently, search algorithms for probabilistic planning based on simulation and sampling have been introduced (Kearns, Mansour, and Ng 1999). The UCT algorithm (Kocsis and Szepesvari 2006), in particular, is an extension of sparse sampling. UCT is a Monte-Carlo value estimation technique where action selection in simulated trials is dependent on the upper confidence bound of the action value estimate. UCT has shown success in planning to play the game of GO (Gelly and Wang 2006; Gelly and Silver 2007) and real time strategy games (Balla and Fern 2009).

Combining forward search with abstract backward reasoning or goal regression is attractive for a number of reasons. It provides the forward search with goal-relevant focus and heuristic guidance derived from the backward reasoning. It reduces the effective search depth for the forward search in cases where the backward reasoning could make signficant progress by computing the description of states which are some steps away from the goal. In harder domains where backward reasoning is not so effective, forward search could still make some incremental progress towards the goal.

## Our Approach

BOPP combines Upper Confidence Trees(UCT) with SPUDD. A sequential approximation of the original problem is constructed by ignoring concurrent actions in the domain. It is this problem that SPUDD solves. SPUDD represents the value function as an algebraic decision diagram. The leaf nodes in the decision diagram contain the value of the state and a path in the diagram corresponds to a state and its value. The value function $V^i$ at every iteration $i$ of value iteration is cached. Note that $V^i$ can be used as a heuristic to evaluate a state at a depth $h - i$ in the search tree constructed by UCT, where $h$ is the horizon. We compute the range of values of states in $V^i$, $\Delta V^i = |max_s V^i(s) - min_s V^i(s)|$. In the absence of domain knowledge, we use $\Delta V^i$ to make intelligent guesses about problem-specific parameters. Firstly, we prune the decision diagram using all pair pruning. The pruning size is set to 10% of $\Delta V^i$, $V^i \leftarrow prune(V^i, all\_pair, 0.1\Delta V^i)$. Note that setting this parameter to more than 50% of $\Delta V^i$ would cause the decision diagram to collapse into one leaf node.

In parallel, the original problem is solved in the forward direction using UCT. UCT is parameterized by the default policy, the exploration constant $C > 0$ and the *forward hori-*

[1]BOPP is a work in progress. The results presented here are after post competition analysis and corrections.

*zon* $0 \leq f_h \leq h$, where $h$ is the horizon of the problem. In the bandit phase, action values are computed as a sum of two terms corresponding to exploration and exploitation. The relative weight of the exploration term is controlled by $C$. In practice, this constant has to be tuned for a given domain. As a rule of thumb, $C$ can be set to the difference between the maximum and minimum expected reward per episode. Let $C^i$ denote the value of $C$ at a depth $i$ in the search tree. We set $C^i = \Delta V^i$. When an unexplored state is encountered, the default policy is followed until the forward horizon. The resulting state is evaluated using the corresponding value function. $V_{tree}(s,d) = evaluate(s, V^{h-d})$. If $V^{h-d}$ is unknown, the most recent value function is used instead. Thus SPUDD provides UCT with a much needed fringe heuristic, especially in domains with sparse rewards. It also allows us to fix the number of decisions taken using the default policy. The parameter $f_h$ can be considered as controlling the bias-variance tradeoff. A large value of $f_h \sim h$ would lead to high variance trajectories while following the base policy. A low value of $f_h$ introduces bias towards the SPUDD value function. Given the time constraints of the IPPC $f_h$ also needs to be tuned such that a sufficiently wide search tree can be built quickly. In domains with exogenous events such as SysAdmin it might even be reasonable to restrict the length of default trajectories.

## Experimental results

The IPPC 2011 consists of eight domains with ten problem instances per domain. The instances are ordered by difficulty in terms of number of objects and/or the degree of randomness. For example, in the SysAdmin domain problems are ordered by the number of computers and the failure probability of each computer. A total time of 24 hours is divided uniformly among the domains. BOPP is parameterized by the default policy $\pi_{default}$ and forward horizon $f_h$ (Other parameters are set as defined in the preceding section). We would like $f_h$ to be a function of the problem horizon and some measure of progress in SPUDD. This might not be practical as the horizon can be arbitrarily large(even unknown in some cases).

We first show the usefulness of truncated default trajectories. Figure 1 shows the average reward obtained in the domain of traffic MDP using UCT($\pi_{RANDOM}$,$f_h = 10$) and UCT($\pi_{RANDOM}$, $f_h = h$), where $h$ is the horizon. The effect of truncating the rollouts seems to be beneficial. Simulations are time consuming in this domain and the instances are such that a lookahead of more than ten is not necessary. Figure 2 shows the average reward in the skill teaching domain presented in IPPC 2011 with parameters BOPP($\pi_{RANDOM}$, $f_h = 1$). We can see the scalability of BOPP as compared to SPUDD. In this domain, the performance comes mainly from UCT. Figure 3 shows the average reward for the SysAdmin domain. In this domain SPUDD seems to be the driving force.

In general, UCT and SPUDD exhibit different strengths and weaknesses, and their combination seems like a promising approach to improve upon both of them. We found that BOPP works better than either approach in all the domains of IPPC. More experiments are needed to evaluate BOPP further and improve its performance especially in domains where both UCT and SPUDD perform poorly.

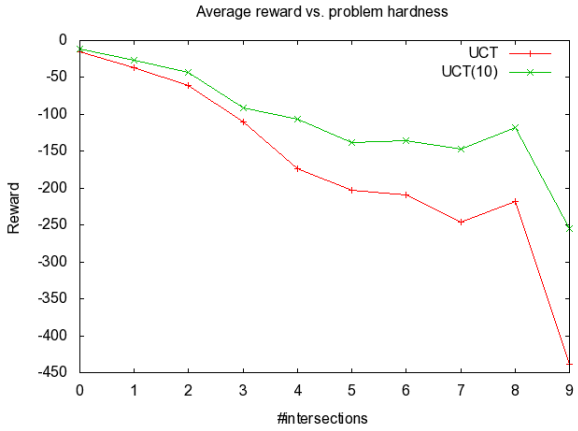


Figure 1: Effect of truncated trajectories

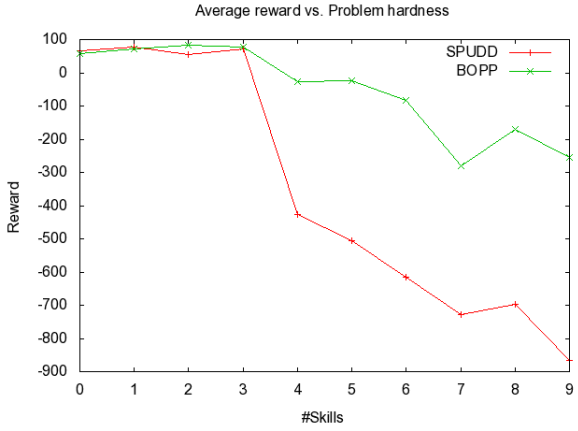


Figure 2: Scalability of BOPP as compared to SPUDD

## References


Balla, R., and Fern, A. 2009. UCT for tactical assault planning in real-time strategy games. In *Proceedings of the International Joint Conference of Artificial Intelligence*, 40–45.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 1999. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for First-Order MDPs. In *Proceed-*

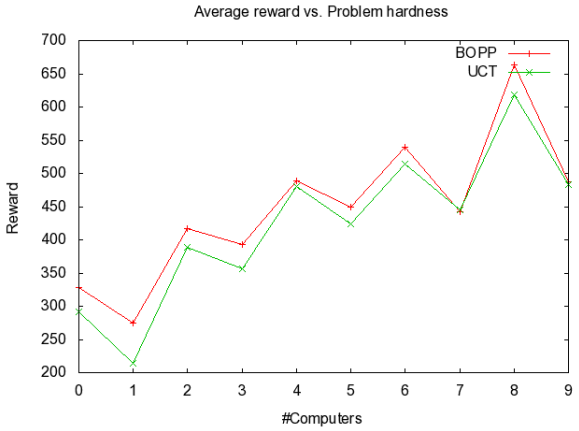


Figure 3: Comparison of UCT and BOPP

*ings of the International Joint Conference of Artificial Intelligence*, 690–700.

Gelly, S., and Silver, D. 2007. Combining online and offline knowledge in UCT. In *Proceedings of the International Conference on Machine Learning*, 273–280.

Gelly, S., and Wang, Y. 2006. Exploration exploitation in Go: UCT for monte-carlo Go. In *NIPS Workshop for Online trading of Exploration and Exploitation*.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of Uncertainty in Artificial Intelligence*, 279–288.

Howard, R. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

Kearns, M.; Mansour, Y.; and Ng, A. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Proceedings of the International Joint Conference of Artificial Intelligence*, 1324–1331.

Kocsis, L., and Szepesvari, C. 2006. Bandit based monte-carlo planning. In *Proceedings of the European Conference on Machine Learning*, 282–293.

Teichteil-Koenigsbuch, F.; Infantes, G.; and Kuter, U. 2008. RFF: A robust FF-based mdp planning algorithm for generating policies with low probability of failure. In *Proceedings of the Sixth IPC at ICAPS*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 352.