Sequential Monte Carlo in Probabilistic Planning Reachability Heuristics

Daniel Bryce
Department of Computer Science and Engineering
Arizona State University, Brickyard Suite 501
699 South Mill Avenue, Tempe, AZ 85281
dan.bryce@asu.edu

Abstract

In this work, we apply heuristic search to conformant probabilistic planning by adapting planning graph heuristics developed for non-deterministic planning. We consider a straight-forward application of these planning graph techniques, which amounts to exactly computing the distribution over reachable relaxed planning graph layers. In reality, computing these distributions is costly, so we apply Sequential Monte Carlo to approximate them. We demonstrate on several domains how our approach enables our planner to far outscale existing (optimal) probabilistic planners and still find reasonable quality solutions.

This work adds to a much broader dissertation on improving the scalability of planning under uncertainty. We have designed heuristics for conformant and conditional non-deterministic planning, in addition to this latest addition to probabilistic planning. Future work will build on these heuristic techniques to address conditional probabilistic planning and devise new search algorithms for probabilistic planning.

Introduction

We address the problem of conformant probabilistic planning, where the planning agent has no observability and must formulate plans with uncertain actions. A conformant plan is a sequence of actions that will guarantee goal satisfaction with some probability. While agents can potentially improve their success by devising conditional plans (that use observations), conformant planning is a special case that is useful when sensing is too expensive or impossible (e.g., because of broken sensors). As has been seen in non-deterministic conformant and conditional planning (Bryce, Kambhampati, & Smith 2006; Brafman & Hoffmann 2005), conformant planning heuristics are useful in conditional planning because ignoring observations is sometimes a reasonable relaxation. Future application of the work discussed herein will validate how well conformant probabilistic planning heuristics adapt to conditional probabilistic planning. Initial experiments indicate the adaptation is appropriate.

Despite long standing interest (Kushmerick, Hanks, & Weld 1994; Hyafil & Bacchus 2003; 2004), probabilistic plan synthesis algorithms have a terrible track record in terms of scalability. The current best conformant probabilistic planners are only able to handle very small problems. In contrast, there has been steady progress in scaling deterministic planning. Much of this progress has come from the use of sophisticated reachability heuristics. In this work, we show how to effectively use reachability heuristics to solve conformant probabilistic planning (CPP) problems. We use work on planning graph heuristics for non-deterministic planning (Bryce, Kambhampati, & Smith 2006; Hoffmann & Brafman 2004) as our starting point.

We investigate an extension of the work by Bryce, Kambhampati, & Smith (2006) that uses a planning graph generalization called the labelled uncertainty graph (*LUG*). The *LUG* is used to symbolically represent a set of relaxed planning graphs (much like the planning graphs used by Conformant GraphPlan, Smith & Weld, 1998), where each is associated with a possible world. While the *LUG* (as described by, Bryce, Kambhampati, & Smith, 2006) works only with state uncertainty, it is necessary in CPP to handle action uncertainty. Extending the *LUG* to consider action uncertainty involves symbolically representing how at each level CGP creates a new literal layer for each joint outcomes of the uncertain actions.

With uncertain actions, an explicit or symbolic representation of planning graphs for all possible worlds at each time step is *exactly* representing an exponentially increasing set of literal layers. Since we are only interested in planning graphs to compute heuristics, it is both impractical and unnecessary to exactly represent all of the reachable possible worlds. We turn to approximate methods for representing the possible worlds. Since we are planning in a probabilistic setting, we can use Monte Carlo techniques to construct planning graphs.

There are a wealth of methods, that fall under the name sequential Monte Carlo (SMC) (Doucet, de Freitas, & Gordon 2001) for reasoning about a hidden random variable over time. SMC applied to "on-line" Bayesian filtering is often called particle filtering, however we use SMC for "off-line" prediction. The idea behind SMC is to represent a probability distribution as a set of samples (particles), which evolve recursively over time by sampling a transition function. In our application, each particle is a (simulated) deterministic planning graph and the transition function describes the Conformant GraphPlan (Smith & Weld 1998) construction semantics. By using more particles, we capture more possible worlds, exploiting the natural affinity between SMC

approximation and heuristic accuracy.

The SMC technique requires multiple planning graphs (each a particle), but their number is fixed. We could represent each planning graph explicitly, but they may have considerable redundant structure. Instead, we generalize the *LUG* to symbolically represent the set of planning graph particles in a planning graph we call the Monte Carlo *LUG* ($\mathcal{M}cLUG$). We show that by using the $\mathcal{M}cLUG$ to extract a relaxed plan heuristic we are able to greatly out-scale the current best conformant probabilistic planner CPplan (Hyafil & Bacchus 2004; 2003) in a number of domains, without giving up too much in terms of plan quality.

This work appears as a full paper with Subbarao Kambhampati and David E. Smith in the ICAPS'06 Technical Session. Our presentation starts by describing a worked example of how to construct planning graphs that exactly compute the probability distribution over possible worlds versus using SMC, as well as how one would symbolically represent planning graph particles. We then present an empirical analysis of our technique compared to CPplan, and conclusions. Please consult the full paper for the formal details.

Monte Carlo Planning Graph Construction

We illustrate an example to give the intuition for Monte Carlo simulation in planning graph construction. Consider a simple logistics domain where we wish to load a specific freight package into a truck and loading works probabilistically (because rain is making things slippery). There are two possible locations where we could pick up the package, but we are unsure of which location. There are three fluents, $F = \{$ atP1, atP2, inP $\}$, our initial belief state b_I is 0.5: $s0 = \{$atP1, $\neg$atP2, $\neg$inP $\}$, 0.5: $s1 = \{\neg$atP1, atP2, $\neg$inP $\}$, and the goal is $G = \{$inP$\}$. The package is at location 1 (atP1) or location 2 (atP2) with equal probability, and is definitely not in the truck (inP). Our actions are LoadP1 and LoadP2 to load the package at locations 1 and 2, respectively. Both actions have an empty enabling precondition $\{\}$, so they are always applicable, and have two outcomes. The first outcome with probability 0.8 loads the package if it is at the location, and the second outcome with probability 0.2 does nothing. We assume for the purpose of exposition that driving between locations in not necessary.

Figure 1 illustrates several approaches to planning graph based reachability analysis for our simplified logistics domain. (We assume we are evaluating the heuristic value $h(b_I)$ of reaching G from our initial belief state.) The first is in the spirit of Conformant GraphPlan, where uncertainty is handled by splitting the planning graph layers for all outcomes of uncertain events. CGP creates a planning graph that resembles a tree, where each branch corresponds to a deterministic planning graph.

CGP: In Figure 1a, we see that there are two initial literal layers (denoted by literals in boxes), one for each possible world at time zero. We denote the uncertainty in the source belief state by X_0, which takes on values $s0, s1$ (for each state in our belief state). Both load actions are applicable in both possible worlds because their enabling preconditions are always satisfied. The edges leaving the actions denote the probabilistic outcomes (each a set of conditional effects). While it is possible for any outcome of an action to occur, the effects of the outcome may or may not have their secondary precondition supported. In world $s0$, if outcome the first outcome of LoadP1 occurs, then effect denoted by atP1→inP is enabled and will occur, however even if the first outcome of LoadP2 occurs its effect is not enabled and will not occur.

The set of possible worlds at time one is determined by the cross product of action outcomes in each world at time zero. For instance, possible world $x00$ is formed from world $s0$ when outcomes the first outcome of LoadP1 and the first outcome of LoadP2 co-occur. Likewise, world $x12$ is formed from world $s1$ when outcomes the second outcome of LoadP1 and the first outcome of LoadP2 occur.

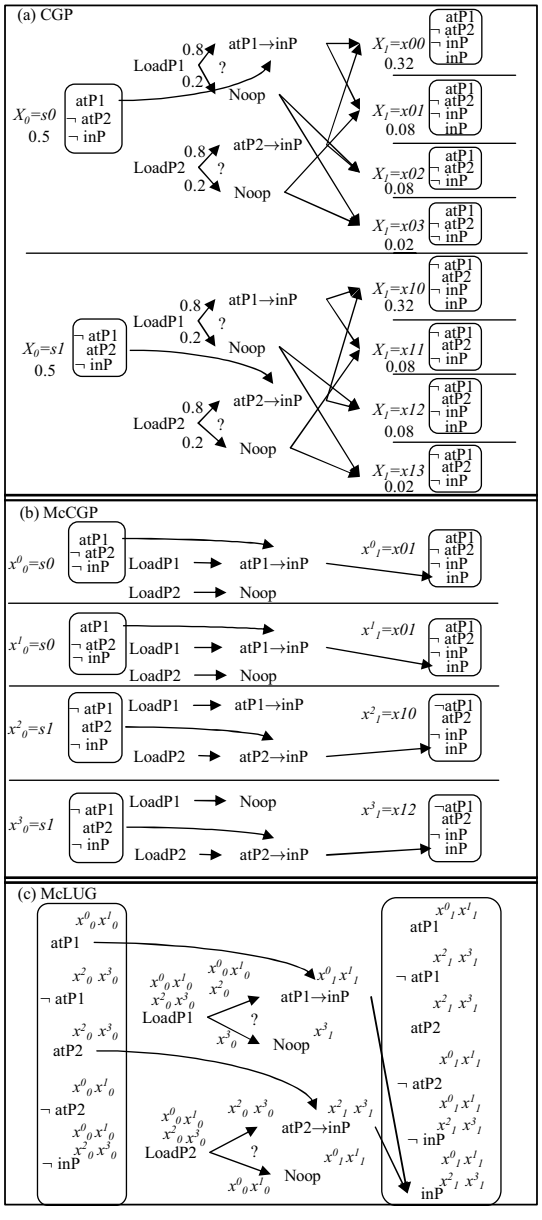

Figure 1: *Variations on planning graph representations.*

CGP is exactly representing the reachable literal layers for all possible worlds. In our example, CGP could determine the exact distribution over X_1 for every value of X_0. We see that our goal is satisfied in half of the possible worlds at time 1, with a total probability of 0.8. It is possible to back-chain on this graph to extract a relaxed plan (by ignoring mutexes) that satisfies the goal with 0.8 probability. However, we note that this is not efficient because it is exactly representing all possible worlds (which can increase exponentially).

McCGP: Next, we illustrate a Monte Carlo simulation approach we call Monte Carlo CGP (McCGP), in Figure 1b. The idea is to represent a set of N planning graph particles. In our example we sample $N = 4$ states $\{x_0^n\}_{n=0}^{N-1} \sim P(X_0) = b_I$ and create an initial literal layer for each. To simulate a particle we first insert the applicable actions. We then insert effects by sampling from the distribution of joint action outcomes. Finally, the subsequent literal layer is constructed, given the sampled outcomes. Note that each particle is a deterministic planning graph.

In our example, the simulation was lucky and the literal layer for each particle at time 1 satisfies the goal, so we may think the best one step plan achieves the goal with certainty. From each of these graphs where the goal is satisfied it's possible to extract a relaxed plan, which can then be aggregated to give a heuristic as described by Bryce, Kambhampati, & Smith (2006). While McCGP improves memory consumption by bounding the number of possible worlds, it still wastes quite a bit of memory. Of the planning graphs many literal layers are identical. Symbolic methods allow us to compactly represent these planning graph particles.

***McLUG*:** Using ideas from Bryce, Kambhampati, & Smith (2006) , we can represent a single literal layer at every time step for all samples in a planning graph called the Monte Carlo *LUG* ($\mathcal{M}cLUG$), in Figure 1c. The idea is to union the connectivity of multiple planning graphs into a single planning graph skeleton, and use labels on the actions and literals to signify the original, explicit planning graphs in which an action or literal belongs. The contribution in the $\mathcal{M}cLUG$ is to represent a set of particles symbolically and provide a relaxed plan extraction procedure that takes advantage of the symbolic representation. From the $\mathcal{M}cLUG$ we are able to extract a relaxed plan that supports the goal for every particle that reaches the goal.

Empirical Analysis

We externally evaluate our planner and its heuristic based on the $\mathcal{M}cLUG$ by comparing with the leading approach to CPP, CPplan (Hyafil & Bacchus 2003; 2004). We also internally evaluate our approach by adjusting the number of particles N that we use in each $\mathcal{M}cLUG$. We show results for a logistics domain. CPplan finds the optimal probability of goal satisfaction for a given plan length, but our planner, like Buridan (Kushmerick, Hanks, & Weld 1994), finds plans that satisfy the goal with probability no less than τ. We find plans with a forward-chaining A* search in the space of belief states. To compare with CPplan, we run CPplan on a problem for each plan length until it exceeds our time or

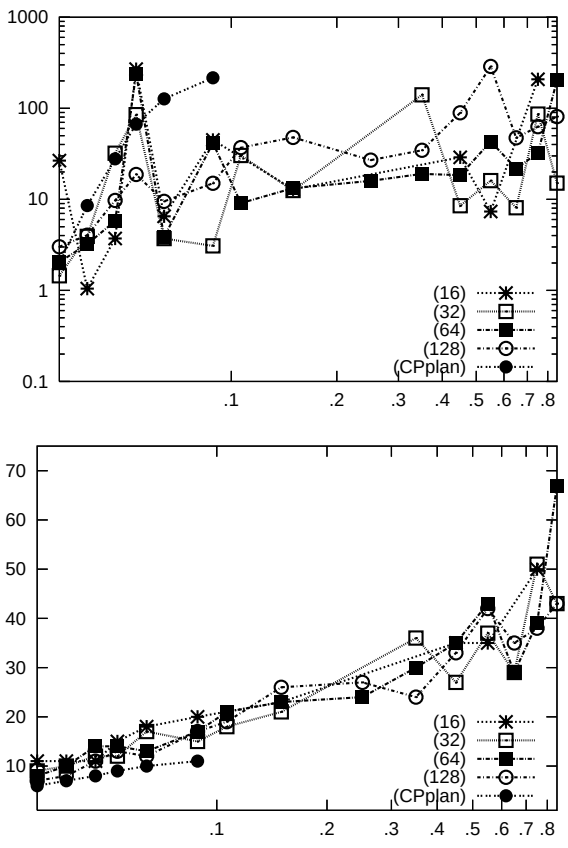

Figure 2: *Run time in seconds (top), and Plan lengths (bottom) vs. τ (log scale) for Logistics p4-2-2*

memory limit. We record the probability that CPplan satisfies the goal for each plan length. We then give our planner a series of problems with increasing values of τ that match the values found by CPplan (and fixed increments thereafter). We ran our planner five times on each problem and present the average run time, and plan length.

The logistics domain has the standard logistics actions of un/loading, driving, and flying, but adds uncertainty. Hyafil & Bacchus (2004) enriched the domain to not only include initial state uncertainty, but also action uncertainty. In each problem there are some number of packages whose probability of initial location is uniformly distributed over some locations and un/loading is only probabilistically successful. Plans require several loads and unloads for a single package at several locations, making a relatively simple deterministic problem a very difficult stochastic problem. We compare on problem p4-2-2, where there are 4 possible initial locations for a package, 2 cities, and 2 packages.

The plots in Figure 2 compare the total run time in seconds (top) and the plan lengths (bottom) of our planner with 16/32/64/128 particles in the $\mathcal{M}cLUG$ versus CPplan. In this domain we also use helpful actions from the relaxed plan. We notice that CPplan is able to at best find solutions where $\tau \leq 0.09$. In most cases our planner is able to find plans much faster than CPplan for the problems they

both solve. It is more interesting that our planner is able to solve problems for *much larger* values of τ. Our planner finds solutions where $\tau \leq 0.85$ which is 9.6 times the maximum values of τ solved by CPplan. In terms of plan quality, the average increase in plan length for the problems we both solved was 4.2 actions. Where CPplan exactly evaluates plan suffixes to find pruning conditions for plan prefixes, we use a heuristic to estimate plan suffixes. As the results demonstrate, our heuristic effectively guides search toward good plans.

Conclusion & Future Work

We have presented an approach called $\mathcal{M}cLUG$ to integrate Monte Carlo into heuristic computation on planning graphs. The $\mathcal{M}cLUG$ enables us to quickly compute effective heuristics for conformant probabilistic planning. With the heuristics, our planner is able to far out-scale the current best conformant probabilistic planner. At a broader level, our work shows one fruitful way of exploiting the recent success in deterministic planning to scale stochastic planners.

Our future work will concentrate on adapting the heuristics described here to handle conditional probabilistic planning. We also intend to develop a Monte Carlo based heuristic search algorithm for search in belief space that combines the work of Thrun (2000) and Barto, Bradtke, & Singh (1995).

References

Barto, A. G.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72:81–138.

Brafman, R., and Hoffmann, J. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of ICAPS'05*.

Bryce, D.; Kambhampati, S.; and Smith, D. 2006. Planning graph heuristics for belief space search. *JAIR*. (To appear).

Doucet, A.; de Freitas, N.; and Gordon, N. 2001. *Sequential Monte Carlo Methods in Practice*. New York, New York: Springer.

Hoffmann, J., and Brafman, R. 2004. Conformant planning via heuristic forward search: A new approach. In *Proceedings of ICAPS'04*.

Hyafil, N., and Bacchus, F. 2003. Conformant probabilistic planning via CSPs. In *Proceedings of ICAPS' 03*.

Hyafil, N., and Bacchus, F. 2004. Utilizing structured representations and CSPs in conformant probabilistic planning. In *Proceedings of ECAI'04*.

Kushmerick, N.; Hanks, S.; and Weld, D. 1994. An algorithm for probabilistic least-commitment planning. In *Proceedings of AAAI'94*.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings of AAAI'98*.

Thrun, S. 2000. Monte Carlo POMDPs. In *Advances in Neural Information Processing 12*, 1064–1070.