Probabilistic Reachability Analysis for Structured Markov Decision Processes

Florent Teichteil-Königsbuch and **Patrick Fabiani**
ONERA-DCSD 2 Avenue Édouard-Belin
31055 Toulouse, France
(florent.teichteil,patrick.fabiani)@cert.fr

Abstract

We present a stochastic planner based on Markov Decision Processes (MDPs) that participates to the probablistic planning track of the 2004 International Planning Competition. The planner transforms the PDDL problems into factored MDPs that are then solved with a structured policy iteration algorithm. A probabilistic reachability analysis is performed, approximating the MDP solution over the reachable states subspace, in order to restrict the search space and allow a subsequent heuristic search.

Introduction

We present a planner based on Markov Decision Processes (MDPs) (Puterman 1994) to participate in the probabilistic planning track of the International Planning Competition at ICAPS'04. MDPs provide a decision-theoretic framework for planning with uncertain actions effects. A MDP (Puterman 1994) is a Markov chain controlled by an agent. A control strategy associates to each state the choice of an action, whose result is a stochastic state. The Markov property means that the probability of arriving in a particular state after an action only depends on the previous state of the chain and not on the entire states history. Formally it is a tuple $\langle S, A, T, R\rangle$ where S is the set of states, A is the set of actions, T and R are functions giving respectively the transition probabilities between states (depending on the chosen action) and the immediate or terminal rewards (depending on the starting state, the chosen action and the ending state). The most frequent optimisation criterion consists in maximizing the infinite horizon sum $E\left(\sum_{t=0}^{\infty} \beta\, r_t\right)$ of expected rewards r_t discounted by a factor $0 < \beta < 1$ that insures the convergence of algorithms, but can also be interpreted as a uncontrolled stopping probability between two time points.

The resolution of MDPs is based on dynamic programming and includes two classes of algorithms : value iteration and policy iteration. The first is an iteration on the value function associated with each state, that is to say the expected accumulated reward starting from this state. When the iterated value function stabilizes, the optimal value function is reached and the optimal policy follows. In the policy iteration scheme, the current policy is assessed on the infinite horizon and improved locally at each iteration. The value of a policy π is solution of Bellman's equations (Bellman 1957) :

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s') \cdot \left(R(s, \pi(s), s') + \beta\, V^{\pi}(s')\right)$$

Compared to value iteration, the policy iteration algorithm converges in fewer iterations, but each policy assessment stage may be computationally costly. A large discussion about criteria and resolution algorithms is proposed in (Puterman 1994).

Motivations and issues

Nevertheless, classical exact algorithms (based on stochastic dynamic programming on an explicitly enumerated state space) are not effective enough for realistic applications that often have very large state spaces (Boutilier & Hanks 1999; Verfaillie, Garcia, & Péret 2003). Proposed techniques to solve such problems include approximating or learning methods (Bertsekas & Tsitsiklis 1995) where the computing cost and the error are both controlled. Other approaches exploit the natural structure of planning problems either by using compact factored representations (Boutilier & Hanks 1999; Boutilier, Dearden, & Goldszmidt 2000; Hoey *et al.* 2000), or by decomposing the state space in sub-regions (Hauskrecht *et al.* 1998; Dean & Lin 1995; Parr 1998) that enables a hierarchical resolution that is sometimes more effective.

Our initial motivations are to combine factored and enumerated state representations in probabilistic planning (Teichteil-Knigsbuch & Fabiani 2004). The obtained hybrid MDP model exploits the problem structure in terms of both decomposition and factorization. This approach is adapted for stochastic planning problems involving both intermediate tasks planning and navigation planning. Tools are needed in order to restrict the search space to its useful part and allow an efficient heuristic search in useful regions.

State space factorization

Our planner uses a compact factored representation of MDPs based on Algebraic Decision Diagrams (ADDs) (R.I. Bahar *et al.* 1993) and is inspired from (Hoey *et al.* 2000).

Since the problems of the stochastic planning track of the competition are given in the PPDDL 1.0 language (Younes & Littman 2003), we must translate the PPDDL problem definitions into ADDs-based MDP representation.

The factorization of the state space consist in a cross product involving state variables : $S = \otimes_{i=1}^{n} x_i$. It is a compact representation because the states are no longer enumerated in a list, but rather structured by the set of random state variables: $x_i{}_{i=1}^{n}$. Such variables enable to process sets of states, instead of individual states, whenever useful. For each action, the transition probability into a given state is no longer given as a function of the individual initial state but now depends conditionnally on the state variables. Therefore, they can be represented either as Dynamic Bayesian Networks (Dean & Kanazawa 1989) or with probabilistic STRIPS operators (Dearden & Boutilier 1997).

Dynamic Bayesian Networks (DBNs)

A factored MDP can be represented by use of a set of action networks. For each action, an action network (which is a DBN) represents the probabilistic effects and rewards obtained on the variables after the action has been performed (*post-action variables*), conditionally to the possible values of the variables before the action is applied (*pre-action variables*). There can exist *diachronic arcs*, directed from pre-action variables to post-action variables, and *synchronic arcs* encoding for dependences (correlations) between post-action variables. Such DBNs represent the factored conditional (controlled) transition probabilities within the state space, encoded as conditional probabilities of obtaining the post-action variables knowing the pre-action variables. The corresponding immediate rewards are directly associated to the possible transitions. These data are stored respectively in a Conditional Probability Table and in a Conditional Reward Table. Such data structures can be represented either as a set of decision trees (Boutilier, Dearden, & Goldszmidt 2000) or as a set of Algebraic Decision Diagrams (ADDs) (Hoey *et al.* 2000). Although ADDs only deal with binary variables (boolean values), they are in most cases much more effective than decision trees. Non-binary variables are then encoded using a number of boolean variables (Hoey *et al.* 2000).

Resolution scheme

The resolution scheme corresponding to factored MDPs, named *Decision-Theoretic Regression*, avoid the explicit enumeration of all states at each iteration. The corresponding algorithms are structured versions of the classical MDPs resolution algorithms, which use algebraic operations defined on decision trees, or ADDs, in order to solve Bellman's equations for these data structures. For instance, using ADDs, the conditional probabilities ADDs of the possible actions (Probability ADDs) and conditional reward values ADDs of the possible actions (Reward ADDs) are combined in order to provide both Value Function ADDs and Policy ADDs on the factored state space. The algorithms directly perform the operations on ADDs (the same on decision trees naturally). The SPI algorithm (Boutilier, Dearden, & Goldszmidt 2000) is a value iteration scheme based on decision trees. The SPUDD and APRICODD algorithms (Hoey *et al.* 2000), based on ADDs, are respectively value iteration and approximated value interation algorithms for factored MDPs. As SPUDD, we use the CUDD package (Somenzi 1998) as an ADD library in our planner.

Policy iteration with ADDs

However, our planner rather implements a structured version of the modified policy iteration. As a matter of fact, we did not find any implementation of the policy iteration scheme based on the CUDD package. To our experience, the CUDD package does not provide directly a number of operations that appear as useful for policy iteration. For instance, policy evaluation requires an operation on the current Policy ADD Π, which replaces each leaf labelled by the number of an action a (Policy ADDs have leaves labelled by action numbers) with the Reward ADD R_a of this action a, and replaces the other leaves by 0. Let us call *ConcatActionRewardADDPolicy(*Π,a*)* such an operation that outputs an ADD R_a^{Π} having the same leaves values as R_a when applicable according to Π, 0 otherwise. $R_{\Pi} =_{a \in A} R_a^{\Pi}$ is the immediate reward ADD applying Π over the state space.

$R_{\pi} \longleftarrow 0$
For a **from** 1 **to** $|A|$ **do**

$$R_a^{\Pi} \longleftarrow ConcatActionRewardADDPolicy(\Pi, a)$$
$$R_{\pi} \longleftarrow R_{\Pi} + R_a^{\Pi}$$

Similarly, we need a *ConcatActionProbADDPolicy(*Π,a*)* to compute the probability ADDs P_a^{Π} that applies the Probability ADD P_a of action a whenever applicable according to Π, and 0 otherwise. $P_{\Pi} =_{a \in A} P_a^{\Pi}$ is the transition Probability ADD over the state space S applying Π. The implemented version of these operations could possibly be improved by writing new low-level procedures for the CUDD package.

Correlations

The resolution of factored MDPs can sometimes be specifically improved, depending on the specific features of the problem. For instance, dealing with correlations between post-action variables in action networks (*synchronic arcs*) may be an issue. In (Boutilier, Dearden, & Goldszmidt 2000), it is proposed to replace such parasitic post-action variables in decision trees (or ADDs) by modified subtrees containing only pre-action variables. However, this complex operation can be avoided. This is done in our planner by using a single *complete action diagram* per action network (Hoey *et al.* 2000) that represents the product of the conditional probabilities of obtaining the post-action variables knowing the pre-action variables; as a matter of fact, the correlations in that case are implicit and they do not require a specific treatment.

Probabilistic Reachability Analysis and Heuristic Search

Coping with large state spaces is a really challenging issue when dealing with realistic problems. This problem has been addressed from at least two different points of view in the literature :

- Reachability analysis : when the initial state is known, a reachability analysis allows to dismiss state variables combinations (sets of states) corresponding to states that will never be reached or traversed. For example, the algorithm REACHABLEK proposed in (Boutilier, Brafman, & Geib 1998) enables to push away from trees (or ADDs in the same way) the nodes corresponding to states that are not reachable when starting from a given starting state.
- heuristic search : an heuristic search algorithm can be used in order to speed up the optimization algorithms, either by producing good initialization values for iterative optimization, or by leading the optimization algorithm to run on more useful regions of the state space. For example, the algorithm proposed in (Feng & Hansen 2001) does both and guarantees to converge towards the optimal solution by using an admissible heuristic. It performs value iteration on a restriction E of the state space. It uses a lower bound estimation as a heuristic initial value assigned on the "fringe" states on the border of E for value iteration on the states of E. This heuristic also determines the "explansion" of E via a reachability analysis using the current "partial" policy Π given by policy iteration at this stage.

The meeting point of both points of view is reached when the heuristic search is based on a reachability analysis. In our planner we perform a probabilistic reachability analysis on the problem. We use it in the policy iteration scheme in order to provide an initial partial policy. We also use it to restrict the resolution algorithm on a useful subspace of the state space. These aspects of the resolution schemeare still under development and require further work.

Conclusion

We have presented our probabilistic planner which is based on Factored Markov Decision Processes (MDPs) as a decision-theoretic framework for planning under uncertainty. The work described in this short paper is still uncomplete at this time, but will be completed for the probabilistic planning track of the International Planning Competition at ICAPS'04. We expect the competition to lead to improvements of our algorithms, to be used later in a more general framework combining factored and enumerated state representations. Such an hybrid MDP model allows to take advantage of the problem structure in terms of both (geographical) decomposition and factorization. It is more dedicated to stochastic planning problems involving both intermediate tasks planning and navigation planning, such as exploration missions. This research is part of the autonomous helicopter project *ReSSAC* project at *ONERA (http://www.cert.fr/dcsd/RESSAC)*.

References

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

Bertsekas, D. P., and Tsitsiklis, J. N. 1995. Neuro-dynamic programming: an overview. In *Proceedings of the 34th Conference on Decision and Control*, 560–564.

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

Boutilier, C.; Brafman, R. I.; and Geib, C. 1998. Structured reachability analysis for Markov decision processes. In *Uncertainty in Artificial Intelligence*, 24–32.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

Dean, T., and Lin, S.-H. 1995. Decomposition techniques for planning in stochastic domains. In *Proceedings of the 14th IJCAI 1995*, 1121–1129.

Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89:219–283.

Feng, Z., and Hansen, E. 2001. Symbolic heuristic search for factored markov decision processes. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 455–460. Edmonton, Canada: AAAI Press / The MIT Press.

Hauskrecht, M.; Meuleau, N.; Kaelbling, L. P.; Dean, T. L.; and Boutilier, C. 1998. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of 14th Conf. UAI 1998*, 220–229.

Hoey, J.; St.Aubin, R.; Hu, A.; and Boutilier, C. 2000. Optimal and approximate stochastic planning using decision diagrams. Technical Report TR-2000-05, University of British Columbia.

Parr, R. 1998. Flexible decomposition algorithms for weakly coupled markov decision problems. In *Proceedings of 14th Conf. UAI 1998*, 422–430.

Puterman, M. L. 1994. *Markov Decision Processes*. John Wiley & Sons, INC.

R.I. Bahar; E.A. Frohm; C.M. Gaona; G.D. Hachtel; E. Macii; A. Pardo; and F. Somenzi. 1993. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, 188–191. Santa Clara, California: IEEE Computer Society Press.

Somenzi, F. 1998. Cudd: Cu decision diagram package. Technical report, University of Colorado at Boulder.

Teichteil-Knigsbuch, F., and Fabiani, P. 2004. Un modle hybride en planification probabiliste d'exploration autonome. In *Proceedings RFIA'04*.

Verfaillie, G.; Garcia, F.; and Péret, L. 2003. Deployment and Maintenance of a Constellation of Satellites: a Benchmark. In *Proceedings of ICAPS'03 Workshop on Planning under Uncertainty and Incomplete Information*.

Younes, H. L., and Littman, M. L. 2003. Ppddl 1.0: An extension to pddl for expressing planning domains with probabilistic effects.