Symbolic Stochastic Focused Dynamic Programming with Decision Diagrams

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 probabilistic planning track of the 2006 International Planning Competition. The planner transforms the PPDDL problems into factored MDPs that are then solved with a structured modified value iteration algorithm based on the safest stochastic path computation from the initial states to the goal states. First, a state subspace is computed by making all the transitions deterministic. Then, a step of modified value iteration on the current reachable state subspace alternates with a step of reachable state expansion by following the current policy.

Introduction

Co-located with the 16th International Conference on Automated Planning and Scheduling, the probabilistic planning track of the 5th International Planning Competition aims at evaluating and at motivating research in the field of non-determinism in planning (Bonet & Givan 2005). In this article, we present a planner based on Markov Decision Processes (MDPs) (Puterman 1994) that have become a popular stochastic framework for planning under uncertainty: the uncertain effects of actions are modeled in a decision-theoretic framework.

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 $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ where S is the set of states, A is the set of actions, T and R are respectively the transition probabilities and rewards that are functions of the starting state, the ending state and the chosen action. The most used optimization 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 probability of a system failure (mission end) between two time points.

The optimization of MDPs produces a *policy*, i.e. a map associating an optimal action to each possible state. It 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 if we start 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 the Bellman equation (Bellman 1957) :

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

In the probabilistic track of IPC'06 (Bonet & Givan 2005), no rewards are explicitly associated to transitions between states. Alternatively, some goal states are defined, meaning that the planner must produce a policy that aims at reaching at least one goal state from some possible initial states. In an MDP framework, this approach is equivalent to define positive rewards to the transitions that lead to goal states. As all goal states have the same significance, all rewards have the same value.

In this particular case, the policy that maximizes the accumulated expected rewards is equal to the policy that maximizes the probability of reaching the goal state subspace $\mathcal{G}$. For a given policy π, the probability P^{π} to reach at least one goal state from any state convergences and it satisfies the following probabilistic dynamic programming equation (Teichteil-Königsbuch 2005):

$$P^{\pi}(s) = \mathbf{1}_{\mathcal{G}}(s) + \mathbf{1}_{\mathcal{S}\setminus\mathcal{G}}(s) \cdot \sum_{s' \in \mathcal{S}} T(s, \pi(s), s') \cdot P^{\pi}(s')$$

where $\mathbf{1}_{\mathcal{E}}$ is the indicator function of a subspace $\mathcal{E} \subset \mathcal{S}$.

State space factorization

Our planner uses a compact factored representation of MDPs based on Algebraic Decision Diagrams (ADDs) (R.I. Bahar *et al.* 1993) that generalize Binary Decision Diagrams (BDDs). Our model is based on work by (Hoey *et al.* 2000) to model and optimize MDPs with decision diagrams. Since the problems of the stochastic planning track of the competition are given in an extension of the PPDDL 1.0 language

(Younes & Littman 2003), we must translate the PPDDL domain and problem definitions into ADDs-based MDP representation. We used the CUDD package (Somenzi 1998) to deal with ADDs and BDDs in the competition.

The factorization of the state space consists in a cross product involving binary state variables: $\mathcal{S} = \otimes_{i=1}^{n} \mathcal{V}_i$. These variables are the instantiations of the PPDDL parametrized predicates for each constant and each object. It is a compact representation because the states are not enumerated in a list, but rather structured by the set of random state variables. Such variables enable to process sets of states, instead of individual states, whenever useful.

The actions are obtained by instantiating all PPDDL parametrized actions for all constants and objects. For each action, the transition probability function can be represented by a Dynamic Bayesian Network (DBN) (Dean & Kanazawa 1989). It encodes the probabilistic effects and rewards obtained on the different values of 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*).

The factored conditional transition probabilities of the DBNs can be encoded as ADDs, that internally use *unprimed* (pre-action) variables and *primed* (post-action) variables (Hoey *et al.* 2000). The action masks, i.e. PPDDL preconditions, can be encoded as BDDs, since they can be defined as indicator functions.

Dealing with action similarities

In PPDDL, actions often are similar in the sense that some parameter instantiations lead to the same preconditions or effects for two different actions. Therefore, some sub-diagrams of the ADDs encoding the transitions are the same over the different actions. In order to memorize only one time these sub-diagrams, we propose to merge all the transition ADDs (resp. mask BDDs) of each action in a single ADD (resp. BDD) named *Global Action Diagram*. This requires to define action variables shared by all ADDs and BDDs: if PPDDL parametrized actions lead to m instantiated actions for all constants and objects, we must introduce $\mathbb{E}(\log_2 m)$ action binary variables on top of primed and unprimed state variables ($\mathbb{E}(k)$ is the smallest integer bigger or equal to k). Also, our single transition ADD $\tilde{T}$ is defined as:

$$\tilde{T} = \sum_{a \in \mathcal{A}} \mathbf{1}_a \cdot T^a$$

Contrary to work by (Hoey *et al.* 2000), our policy encoding is no more a list of mask BDDs for each action, but rather a single mask BDD that represents the state subspace (unprimed variables) where each action is optimal:

$$\pi = \bigcup_{a \in \mathcal{A}} \mathbf{1}_a \cdot \pi^{-1}(a)$$

In the value iteration scheme, the update of the value requires to compute the maximum of the previous computed value over the actions (Puterman 1994). This can be tedious and ineffective with action variables because the value of an action can only be retrieved by a projection on the variables that encode this action. The value update step would require as many projection computations as the number of actions, and each projection would cause the loss of similarities between actions.

Therefore, we have extended the CUDD package by a new low-level ADD function that we called `Cudd_addMaximumAbstract`, and inspired by `Cudd_addExistAbstract`. This new function computes the maximum of all sub-diagrams over the variables of a given cube. In the case of MDPs, this cube is composed of the action variables. It recursively calls the built-in CUDD function `Cudd_addMaximum` on the sub-diagrams of each action variable, until all action variables are parsed.

Optimization focused on the goal states

In the problems of the competition, the knowledge of possible initial states and of goal states enables to restrict the policy computation to a subspace of the entire state space. This idea was already used in `sLAO*`, but it only used the knowledge of initial states. Moreover, `sLAO*` is based on a heuristic that is an approximation of the value of states over the entire state space, that then helps the optimal optimization of the value on the states that are reachable from the initial states. As a consequence, `sLAO*` requires to initially visit all the states in the heuristic computation step.

Deterministic reachability analysis

Therefore, we propose to compute a subset of reachable states *before* any approximate or optimal dynamic programming computation, by using the knowledge of *both* initial and goal states. This initial step is performed by making all transitions deterministic: in other words, we transform the Global Action Diagram ADD into a BDD by replacing all non-zero discriminants by 1. As a result, we can efficiently propagate the fringe of reachable states from the initial states until at least one goal state is reached, without memorizing the actions that lead from the initial states to the goal states. Moreover, it is known that BDDs are more effective than 1-0 ADDs (R.I. Bahar *et al.* 1993). This forward reachable state search satisfies the following recursive equation:

$$\mathbf{1}_{\mathcal{F}'^{t+1}}(s') = \bigcup_{a \in \mathcal{A}} \bigcup_{s \in \mathcal{S}} \tilde{T}^{det}(s, a, s') \cdot \mathbf{1}_{\mathcal{F}^t}(s)$$

where $\mathcal{F}$ is the current subset of (forward) reachable states and $\tilde{T}^{det}$ is the Global Action Diagram BDD.

This reachable state subset can still be reduced by performing a backtrack search of reachable states from goal states to initial states inside the forward reachable state subset:

$$\mathbf{1}_{\mathcal{B}^{t-1}}(s) = \bigcup_{a \in \mathcal{A}} \bigcup_{s' \in \mathcal{S}} \tilde{T}^{det}(s, a, s') \cdot \mathbf{1}_{\mathcal{F}}(s) \cdot \mathbf{1}_{\mathcal{F}'}(s') \cdot \mathbf{1}_{\mathcal{B}'^t}(s')$$

where $\mathcal{B}$ is the current subset of (backward) reachable states.

Safest stochastic path policy

After we have generated the initial reachable state subspace $\mathcal{W}$ ($= \mathcal{B}$ at the end of the backward deterministic reachability analysis), we compute the policy that maximizes the probability of reaching the goal state subspace $\mathcal{G}$ inside $\mathcal{W}$. We named this policy *safest stochastic path policy*, since it maximizes the chance of reaching at least one goal state. In the probabilistic track of IPC'06, there are no (positive or negative) rewards so that this policy is the same as the classical optimal policy of MDPs obtained if each goal state is rewarded by 1.

In this special case, the maximum probability of reaching the goal state subspace always converges (Teichteil-Königsbuch 2005). Therefore, contrary to the computation of the value function in MDPs, this probability does not require to be pondered by an empirical discount factor at each iteration (Puterman 1994). The maximum probability P of reaching $\mathcal{G}$ inside $\mathcal{W}$ is given by:

$$P^{t-1}(s) = \mathbf{1}_{\mathcal{G}}(s) + \mathbf{1}_{\mathcal{W}\setminus\mathcal{G}}(s) \cdot \max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} T(s,a,s') \cdot \mathbf{1}_{\mathcal{W}}(s) \cdot \mathbf{1}_{\mathcal{W}'}(s') \cdot P^t(s')$$

The maximum over the actions is performed by the function `Cudd_addMaximumAbstract` that we added to CUDD (Somenzi 1998) as an extension.

Policy refinement

The previously obtained policy is not guaranteed to be optimal over the entire state space since it is only optimized over $\mathcal{W} \subset \mathcal{S}$. In order to improve the policy, we alternate a step of deterministic reachability analysis and a step of safest stochastic path policy optimization, in a loop that ends when the policy convergences over the previous reachable state subspace. The current reachable state subspace is generated with the same function as the one responsible for the initial computation of the reachable states, but the fringe of state expansion is propagated by following the current policy (and not by applying all possible actions).

The forward reachable state subspace expansion is:

$$\mathbf{1}_{\mathcal{F}'^{t+1}}(s') = \bigcup_{a\in\mathcal{A}} \bigcup_{s\in\mathcal{S}} \tilde{T}^{det}(s,a,s') \cdot \pi(a,s) \cdot \mathbf{1}_{\mathcal{F}^t}(s)$$

where π is a BDD depending on action variables and on unprimed state variables. The backward expansion is given by:

$$\mathbf{1}_{\mathcal{B}^{t-1}}(s) = \bigcup_{a\in\mathcal{A}} \bigcup_{s'\in\mathcal{S}} \tilde{T}^{det}(s,a,s') \cdot \pi(a,s) \cdot \mathbf{1}_{\mathcal{F}}(s) \cdot \mathbf{1}_{\mathcal{F}'}(s') \cdot \mathbf{1}_{\mathcal{B}'^t}(s')$$

Related work

`sLAO*` (Feng & Hansen 2002) already uses such alternation of a step of reachable state expansion by following the current policy and of a step of policy optimization over the current reachable state subspace. However, there are some key differences between `sLAO*` and our algorithm. First, `sLAO*` does not use the knowledge of goal states so that the state space expansion stops as soon as the new reachable states are the same as the previous reachable states in the high-level loop. Second, for the same reason, the state space expansion step of `sLAO*` does not perform a backward search of reachable states. Third, `sLAO*` takes into account general (positive or negative) rewards so that the policy is updated with the classical discounted dynamic programming equation of MDPs. Nevertheless, the latter is not really a drawback of our algorithm since we can replace the safest stochastic path policy by a classical MDPs policy without changing the spirit of our framework. Moreover, in case of general rewards, the safest stochastic path policy can be a heuristic to the policy optimized with rewards (Teichteil-Königsbuch 2005).

Conclusion

We have presented `sfDP` (*Symbolic Focused Dynamic Programming*) that participates to the probabilistic track of the 2006 International Planning Competition. Based on Binary and Algebraic Decision Diagrams, it is a symbolic dynamic programming algorithm for planning under action uncertainty that focuses the policy on goal states. A step of reachable state subspace expansion by following the current policy alternates with a step of policy optimization over the current reachable state subspace, in a loop that ends if the policy stabilizes over the previous reachable state subspace. We think that the competition is a good forum to improve our algorithm, to compare it with other approaches, and to exchange interesting ideas between researchers.

References

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

Bonet, B., and Givan, R. 2005. 5th international planning competition: Non-deterministic track call for participation.

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

Feng, Z., and Hansen, E. 2002. Symbolic heuristic search for factored markov decision processes. In *Proceedings 18th AAAI*, 455–460.

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.

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.

Somenzi, F. 1998. Cudd: Cu decision diagram package release.

Teichteil-Königsbuch, F. 2005. *Symbolic and Heuristic Approach of Planning under Uncertainty*. Ph.D. Dissertation, SUPAERO.

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