T_ρFSP: Forward Stochastic Planning using Probabilistic Reachability

Teichteil-Königsbuch Florent and **Infantes Guillaume**
ONERA-DCSD
2 Avenue douard-Belin – BP 74025
31055 Toulouse Cedex 4
FRANCE

Abstract

In order to deal with real-world sequential decision making, stochastic outcomes for actions have to be explicitly taken into account, as in the Markov Decision Processes (MDPs) framework. While MDPs have been studied for a long time, dealing with a large number of states is still an open issue. As the initial state is given, most state-of-the-art algorithms use heuristic information over the MDP domain, in order to deal only with states that are reachable from the initial one. But we think this information still is under-exploited. We give a formal definition of such probabilistic reachability and propose a new algorithm that uses much more explicitly this information in a parametric way. We present results that show that our algorithm outperforms previous algorithms.

Introduction

Planning under uncertainty in large stochastic domains has been a very active field of Artificial Intelligence research for many years. Most of recent algorithms use either approximation techniques (Poupart et al. 2002; O. and Aberdeen 2006) or a *heuristic function* to guide the search of the best solution towards the "good" states that are reachable from a given initial state (Hansen and Zilberstein 2001; Barto, Bradtke, and Singh 1995; Buffet and Aberdeen 2007). In the case of *optimal* heuristic algorithms like LAO*(Hansen and Zilberstein 2001), the information gathered from the reachable states is still under-exploited: the quest of optimality conduces such algorithms to optimize the problem over much more states than the ones that are reachable with the current solution.

In this paper, we propose a new optimal heuristic algorithm for planning under uncertainty, named FSP, that only optimizes the states that are reachable from a given initial state. We introduce as well an extended version named T_ρFSP, that performs a *probabilistic* reachability analysis in order to prune the states whose probability of being reached from the initial state is compared to a given threshold ρ.

In section 2, we present Markov Decision Processes (MDPs), that are a popular framework for planning under uncertainty, and LAO*. In sections 3 and 4, we respectively present our algorithm FSP and its extended version T_ρFSP. In section 5, we compare our algorithms to RTDP and LAO*, that still remain state-of-the-art heuristic algorithms, for two different heuristics: we show that FSP and T_ρFSP perform better than RTDP and LAO* depending on the quality of the heuristic. We then conclude with a discussion and some perspectives for future work.

Background: MDPs and LAO*

In this section, we give a brief overview of Markov Decision Processes (MDPs) (Puterman 1994) and present an in-deep view of a state-of-the art heuristic search algorithm: LAO* (Hansen and Zilberstein 2001).

Markov Decision Processes

Formalism

Definition 1 (MDP) *An MDP is defined by:*

- *a set of states* $\mathcal{S} = \{s^1, \cdots, s^p\}$,
- *a set of actions* $\mathcal{A} = \{a^1, ..., a^q\}$,
- *a transition function* $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *denoted* $T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$*, where* s_t *is state at time* t *and* a_t *the action chosen at time* t*;*
- *a set of reward/cost functions* $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$*, denoted* $R(s, a, s')$.

Definition 2 (Policy) *A policy* π *is a function* $\mathcal{S} \rightarrow \mathcal{A}$ *that associates an action* a *to every state* s *of the MDP such that this* a *is applicable to this state* s*. A partial policy associates actions to a subset of* $\mathcal{S}$.

The solution of an MDP is a policy π that maximizes for each state s the average sum $V^\pi(s)$ of rewards (minimizes the sum of costs) over a given horizon and all stochastic trajectories starting from s. This horizon may be infinite, in this case, a discount factor $0 < \gamma < 1$ is given in order to discount future reward, such that maximizing reward in the short term is preferred.

An MDP problem may come with:

- an initial state, that is the solution policy needs to maximize rewards along trajectories starting from this state;
- a goal state, meaning that only transitions going to only one state lead to reward, and this state is absorbing.

An even more special case of MDP is the *shortest stochastic path* problem, where the MDP has an initial state, a goal state and all actions but the ones leading to the goal state have an uniform cost. In some case, a set of adjacent goal states is provided, this case can be easily reduced to the canonical shortest stochastic path problem.

In general, when an initial state is fixed, that means that the answer of the MDP problem can be a partial policy: actions are not given (computed) for un-reachable states.

Heuristic optimization of MDPs If the states have some values $\{V^{\pi}(s)\}_{s\in\mathcal{S}}$ for a given policy π, it is possible to improve the policy of a given state s in one step, by computing a Bellman backup (Puterman 1994). The improved value $V^{\pi^+}(s)$ is given by:

$$V^{\pi^+}(s) = \max_{a\in\mathcal{A}} \left\{ \sum_{s'\in\mathcal{S}} T(s,a,s') \cdot (R(s,a,s') + \gamma V^{\pi}(s')) \right\} \quad (1)$$

If all the states in $\mathcal{S}$ are iteratively improved in that way, then it is proved that the value of states converge to the value function V^* of any optimal policies, whatever the initial value (or initiasl policy) of states (Bellman 1957; Puterman 1994).

Nevertheless, if an initial state s_0 is known, one would like to only improve the values of states that are reachable from s_0. Let $\mathcal{L} \subset \mathcal{S}$ be the set of states that are reachable from s_0 by following a given policy. If the values of states in $\mathcal{S}$ are initialized with *upper bounds* of the optimal values, then the values of states in $\mathcal{L}$ are optimal as soon as $\mathcal{L}$ is stable: i.e. the set of states reachable by following the improved policy is included in the states reachable by following the previous policy. Indeed, this property means that even if the value of a state s' in $\mathcal{S} \setminus \mathcal{L}$ is supposed to have an optimistic upper bound, its discounted value along all possible trajectories leading from s_0 to s' could not be better than the value of a trajectory leading to a reward inside $\mathcal{L}$.

The upper bounds of the optimal values are named *admissible heuristic values*. Formally, the heuristic values of states is a function h such that: $\forall s \in \mathcal{S}, h(s) \geqslant V^*(s)$. The closer the heuristic function h is to the optimal function V^*, the less states are explored. The computation of a good heuristic is an open problem that will not be adressed in this paper. Yet, as we will see in section *Experiments*, the following easily computable uniform upper bound V^*_{sup} of V^* allows to prune a lot of states, since it is not so far from V^*: $V^*_{sup} = \frac{\max_{s\in\mathcal{S}} \max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} T(s,a,s')\cdot R(s,a,s')}{1-\gamma}$. For shortest stochastic path problems, the *Manhattan distance* from any states to the goal state is usually very efficient (Barto, Bradtke, and Singh 1995).

LAO*: Principle and algorithm

LAO* is a heuristic search algorithm that finds partial optimal solutions of MDPs knowing a given initial state. A partial solution is a policy that is only computed over states that are reachable from the initial state. The main loop of LAO* alternates two kinds of computation: reachability analysis and partial optimization.

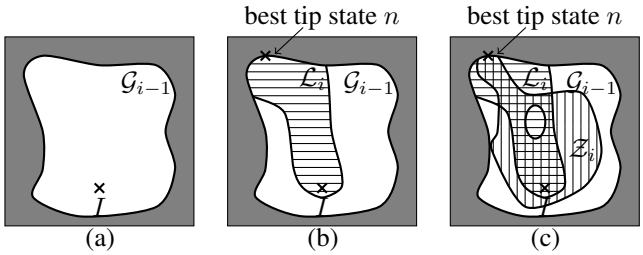

Figure 1: i^{th} loop of LAO*: (a) $\mathcal{G}_{i-1}$ is the set of explored states at iteration $i-1$ — (b) $\mathcal{L}_i$ is the set of states in $\mathcal{G}_{i-1}$ that are reachable from I by following the best current policy ; the best tip state n is identified — (c) $\mathcal{Z}_i$ is the set of states in $\mathcal{G}_{i-1}$ from which it is possible to reach n by following the best current policy from any state in $\mathcal{G}_{i-1}$; value iteration at iteration $i+1$ is performed inside $\mathcal{Z}_i$; the new explored states are $\mathcal{G}_i = \mathcal{G}_{i-1} \cup \{n\}$

Figure 1 shows a schematic view of its behavior. Starting from the initial state I, it searches in the current set of explored states for the non explored states that are reachable using the current policy. These are called *tip states*. If no tip states are reachable, the optimal partial solution is found and the algorithm stops. Otherwise, it selects the best tip state w.r.t. its admissible heuristic (fig. 1 b). Then it computes the set of all possible ancestors of this best tip state in the set of explored states (fig 1 c) and improves the policy in this set only.

Detailed algorithm is shown as Alg. 1. The set of tip states is denoted $\mathcal{F}$. On line 13, the best one n is selected. On line 14, n is expanded, i.e. all new successors of n for all actions are added to $\mathcal{G}$ and their values are initialized with their heuristic values. Then the set of states that from which it is possible to reach n using best partial policy is computed (lines 15 to 20). Then the policy is improved only on these ancestor states (line 4). It is actually improved at the beginning of the main loop in order to initialize the initial state's policy. The policy is optimized over $\mathcal{Z}$ using the iterative Bellman backup of equation 1. This algorithm ends if they are no more tip states.

Analysis

LAO* performs very efficiently for two main reasons:

- it stops whenever it cannot find any tip state, that is the frontier of explored states is not attractive anymore;
- it updates only the states that have a chance to lead to the currently chosen tip state, and not all explored states.

In is worth noting that both policy iteration or value iteration may be used indifferently to improve policy. Another point is that computation is going significantly faster if selecting *all* best states (line 13), instead of only one. In this case, improvement of the policy (which is by far the most expensive part) will be done far less often. Having more than one best state is directly tied to the quality of the heuristic. If it is well informed, less states are going to have the same heuristic value, whereas if it is only an upper bound, all of them are going to be the best ones.

Algorithm 1: LAO*

```
   // F is a set of tip states
   // L is a set of reachable states
   // Z is a set of ancestor states
   // G is a set of explored states
   // I is the initial state of the
      algorithm
   // π is a partial policy over G
1  G ← {I};
2  Z ← {I};
3  repeat
       // improve policy
4      π ← compute_policy(Z);
       // compute tip states
5      L ← {I};
6      L' ← {I};
7      F ← ∅;
8      repeat
9          L' ← {s ∈ S \ L : ∃s' ∈ L', T(s', π(s'), s) > 0};
10         F ← F ∪ (L' ∩ (S \ G));
11         L ← L ∪ (L' ∩ G);
12     until L' ∩ G = ∅ ;
       // identify best tip state and
          expand it
13     n ← best(F);
14     G ← G ∪ expand_tip_state(n);
       // compute ancestor states
15     Z ← {n};
16     Z' ← {n};
17     repeat
18         Z' ← {s ∈ G \ Z : ∃s' ∈ Z', T(s, π(s), s') > 0};
19         Z ← Z ∪ Z';
20     until Z' ≠ ∅ ;
21 until F = ∅ ;
```

FSP: Forward Stochastic Planner

We present in this section a new algorithm called *Forward Stochastic Planner* (FSP) that uses reachability from start state not only for finding tip nodes, but also for pruning even more the set of states whose policy needs to be updated.

Possible improvement over LAO*

In its fourth phase, LAO* computes the ancestors of the selected tip node, that is the set of nodes from where applying current policy may lead to this tip node. This is clearly a way of pruning the set of states to update: the only new information is about this new tip state (whose value has been initialized to its heuristic value), and this information has influence only over its ancestors. Since the heuristic is an optimistic value, the policy can not be changed aver the states from which the tip state can not be reached.

But this set still may be considered as "too large": some states in it may be totally unreachable from the initial state, and thus useless for any trajectory starting from it. We propose, with the FSP algorithm, to update only states that allow to reach the tip state and that are reachable from the initial state.

Unluckily, doing this in a straightforward way does not work. More precisely: if we replace line 4 of algorithm 1 with $\pi \leftarrow compute\_policy(\mathcal{Z} \cap \mathcal{L})$, the algorithm does not find optimal policies anymore. The reason for that is tied to the use of the admissible heuristic and the search for tip states in reachable states from the initial state only. Intuitively, the algorithm should not follow a policy that is not up-to-date with the information given by recent updates. Because of the admissible heuristic, such a policy would tend to lead to tip nodes that are now known to be not as good as formerly hoped, and in some cases this out-of-date policy would lead to miss other interesting tip nodes. As a result, the algorithm would prematurely terminate without being optimal. In LAO*, this case never happens because the new information is spread whenever needed, independently of the reachability ($\mathcal{Z}$ is the set of parents of the tip node).

Defining a new algorithm

In order to avoid this problem, we simply do not consider the policy so far for states that were unreachable and become reachable again ($\mathcal{G} \setminus \mathcal{L}$). We keep in memory the set $\mathcal{L}$ of reachable states from one iteration to the next one of the main loop, because they are the most up-to-date states among the explored states. Contrary to LAO*, the new reachable states are computed inside the previous set of reachable states, but not inside the set of explored states. As soon as a reachable state is found outside $\mathcal{L}$, it is considered as a tip state, even if it has already been explored earlier.

Figure 2 shows FSP in action: the set $\mathcal{Z}$ of states that are optimized are enclosed in the set $\mathcal{L}$ of reachable states, and that the new reachable states are computed inside the previous ones. It is worth noting that the algorithm would not converge if the set $\mathcal{L}_i$ of new reachable states were computed inside the set $\mathcal{Z}_{i-1}$ of optimized states. Indeed, $\mathcal{Z}$ may be non-connex as shown in Figure 2 c, meaning that some absorbant states are reachable from the initial state (no tip states are reachable from these states). If the reachable states would be computed inside $\mathcal{Z}_{i-1}$ instead of $\mathcal{L}_{i-1}$, they would be added to $\mathcal{L}_i$ but never to $\mathcal{Z}_i$: also, the fringe of reachable states would never be empty so that the algorithm would never terminate. Therefore, $\mathcal{L}_{i-1}$ is the smallest set in which the new reachable states can be computed. Algorithm 2 gives an in-deep view of FSP.

Analysis

The main differences with LAO* is that the set of reachable states $\mathcal{L}$ in FSP plays the part of the set of explored states $\mathcal{G}$ in LAO*. When a state is reached with the current policy, that means that its policy is up-to-date and that it can be used to search for tip nodes. This allow to update a lot less states when computing new policy. On the other hand, as less states are updated every time, convergence may be slower when some states switch often from reachable to unreachable. But FSP still performs better than LAO*, as shown in section 5.

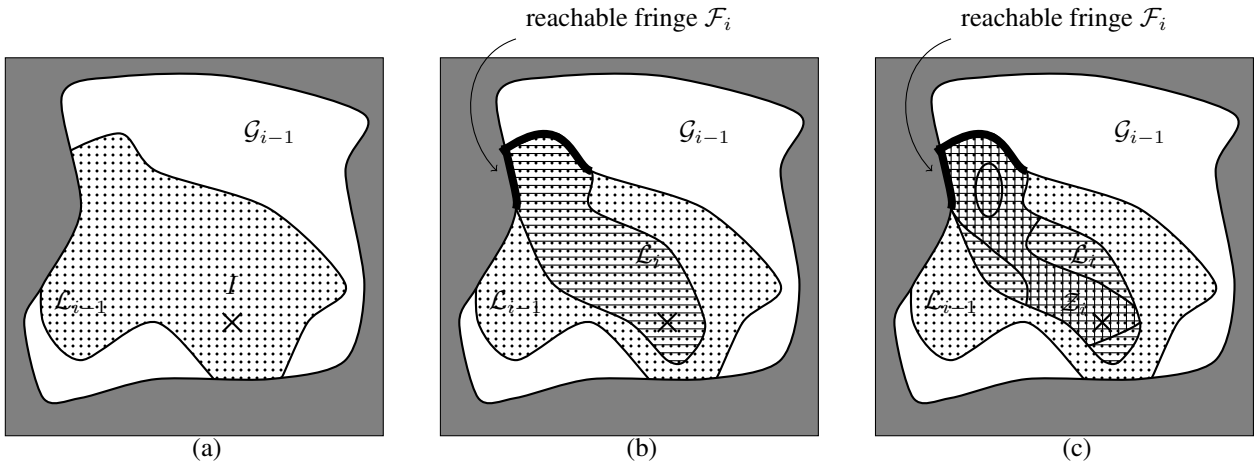

Figure 2: i^{th} loop of FSP: (a) $\mathcal{G}_{i-1}$ is the set of explored states at iteration $i-1$; $\mathcal{L}_{i-1}$ is the set of states that are reachable from I by following the policy at iteration $i-1$ — (b) $\mathcal{L}_i$ is the set of states in $\mathcal{L}_{i-1}$ that are reachable from I by following the best current policy at iteration i ; $\mathcal{F}_i$ is the set of reachable states that are just outside $\mathcal{L}_{i-1}$ — (c) $\mathcal{Z}_i$ is the set of states in $\mathcal{L}_i$ from which it is possible to reach the fringe $\mathcal{F}_i$ by following the best current policy from any state in $\mathcal{L}_i$; value iteration at iteration $i+1$ is performed inside $\mathcal{Z}_i$; the new explored states are $\mathcal{G}_i = \mathcal{G}_{i-1} \cup \mathcal{F}_i$

Extension to thresholded reachability

The notion of reachability does not have to be binary. Clearly, as the MDP is intrinsically stochastic, some states are more reachable than other, using the current policy. This can help to prune even more the computation, focusing on states that are more reachable than others. This information can be very helpful especially if the heuristic is well-informed. In this case, computation will focus even more on "good" states w.r.t. the heuristic function.

Thresholded reachability

Definition 3 (Forward reachability) *We define the forward reachability of a state s, denoted $\widetilde{P}_t^\pi(s \mid I)$, as the probability that s has been visited at least once between timestep 0 and t, by a trajectory that executes the current policy π, starting from the initial state I.*

In order to compute this, we need to compute the probability to be in any state s' at time t, starting from I and without having ever visited a given state s. We denote this probability $U_t(s' \mid s, I)$. By using Bayes' rules and Markov's property, we can demonstrate that:

$$U_t(s' \mid s, I) = \begin{cases} 0 \text{ if } U_{t-1}(s \mid s, I) = 1 \\ \frac{\sum_{s'' \neq s} P(s' \mid s'', \pi(s'')) U_{t-1}(s'' \mid s, I)}{1 - U_{t-1}(s \mid s, I)} \text{ otherwise} \end{cases}$$

with $U_1(s' \mid s, I) = P(s' \mid I, \pi(I))$

We can then compute the forward reachability probability as:

$$\widetilde{P}_t^\pi(s \mid I) = 1 - \Big[(1 - U_t(s \mid s, I))(1 - \widetilde{P}_{t-1}^\pi(s \mid I))\Big]$$

$$\text{with } \widetilde{P}_0^\pi(s \mid I) = \delta_I(s) = \begin{cases} 1 & \text{if } s = I \\ 0 & \text{otherwise} \end{cases}$$

It is easy to demonstrate that the sequence of $\widetilde{P}_t^\pi(\cdot \mid I)$ functions converges when t tends towards $+\infty$, what allows us to iteratively compute it. This demonstration relies on two different cases. If there exists a timestep t such that $U_t(s \mid s, I) = 1$, then it is obvious that $\widetilde{P}_{t'}^\pi(s \mid I) = 1$ for all $t' \geqslant t$. Otherwise, it means that we always have $U_t(s \mid s, I) < 1$, so that $\widetilde{P}_t^\pi(s \mid I) > \widetilde{P}_{t-1}^\pi(s \mid I)$ for all t. In this case, the sequence of $\widetilde{P}_t^\pi(\cdot \mid I)$ functions increases (and 1 is an upper bound): it necessarily converges.

Definition 4 (Backward reachability) *Symmetrically, we define a backward reachability for a state s, denoted $\widehat{P}_t^\pi(s \mid \mathcal{F})$ as the probability to reach in t steps the current frontier $\mathcal{F}$ (of tip states) starting from s and executing the current policy π.*

We have that:

$$\widehat{P}_t^\pi(s \mid \mathcal{F}) = \sum_{s' \in S} P(s' \mid s, \pi(s)) \widehat{P}_{t-1}^\pi(s' \mid \mathcal{F})$$

$$\text{with} \quad \widehat{P}_o^\pi(s \mid \mathcal{F}) = \delta_{\mathcal{F}}(s)$$

The idea now is to only focus on states that have a *high reachability value*, that is are likely to be on trajectories given by the current policy. The sequence of $\widehat{P}_t^\pi(\cdot \mid \mathcal{F})$ functions also convergences when t tends towards $+\infty$, but the demonstration is slightly more difficult than the convergence proof of the forward reachability probability. Unfortunately, place is insufficient to present this demonstration in this paper.

Definition 5 (Thresholded Reachability) *At iteration i, we define the thresholded reachability $0 \leqslant \rho \leqslant 1$ such that the subspaces $\mathcal{L}_i$ and $\mathcal{Z}_i$ are filtered by ρ:*

$$\mathcal{F} = \{s \in \mathcal{S} : \exists s' \in \mathcal{L}_{i-1}, T(s', \pi(s'), s) > 0\} \quad (2)$$

$$\mathcal{L}_i = \{s \in \mathcal{L}_{i-1} \cup \mathcal{F} : \widetilde{P}_{+\infty}^\pi(s \mid I) > 1 - \rho\} \quad (3)$$

$$\mathcal{Z}_i = \{s \in \mathcal{L}_i : \widehat{P}_{+\infty}^\pi(s \mid \mathcal{F}) > 1 - \rho\} \quad (4)$$

Algorithm 2: FSP

```
   // F is a set of tip states
   // Z is a set of ancestor states
   // G is a set of explored states
   // L_i is a set of reachable states at
      iteration i
   // I is the initial state of the
      algorithm
   // π is a partial policy over G
1  G ← {I};
2  Z ← {I};
3  L_0 ← {I};
4  i ← 0;
5  repeat
6  |  i ← i + 1;
   |  // improve policy
7  |  π ← compute_policy(Z);
   |  // compute (reachable) tip nodes
8  |  L_i ← {I};
9  |  L' ← {I};
10 |  F ← ∅;
11 |  repeat
12 |  |  L' ← {s ∈ S \ L_i : ∃s' ∈ L', T(s', π(s'), s) > 0};
13 |  |  F ← F ∪ (L' ∩ (S \ L_{i-1}));
14 |  |  L_i ← L_i ∪ (L' ∩ L_{i-1});
15 |  until L' ∩ L_{i-1} = ∅ ;
   |  // expand all non-explored tip
      |  states
16 |  G ← G ∪ expand_tip_states(F \ G);
   |  // compute reachable ancestor
      |  states
17 |  Z ← F;
18 |  repeat
19 |  |  Z' ← {s ∈ L_i \ Z : ∃s' ∈ Z', T(s, π(s), s') > 0};
20 |  |  Z ← Z ∪ Z';
21 |  until Z' ≠ ∅ ;
22 until F = ∅ ;
```

Clearly, if $\rho = 1$, this reachability is the same as the one used in FSP. The thresholded reachability allows to filter the states whose reachability probability is considered too low w.r.t the real tracking of the final policy by an autonomous agent. If the agent arrives during the mission in an unexplored state that is though reachable, the policy is recomputed from this new initial state. Statistically, this recomputation occurs with a probability of $1 - \rho$.

Updating FSP algorithm

The modification to Alg. 2 are quite straightforward. Lines 8 to 15 need to be replaced by Equations 2 and 3. Lines 17 to 21 are replaced by Equation 4 too.

The other main change is on line 22. FSP (as LAO*) stops when the set of tip nodes is empty. In our case, we stop whenever there is no tip node n such that $\widetilde{P}^{\pi}_{+\infty}(n \mid I) > 1 - \rho$.

As we already noticed, the reachability probabilities can be computed at a relatively low cost, especially if they are approximately computed. Indeed, we do not need to reach the same precision in the probabilities computation as in the computation of states values, because the precision of MDPs algorithms is generally related to the Bellman's error on states values (Puterman 1994).

Comparison

In order to compare T$_\rho$FSP to LAO* and RTDP, we randomly generated MDPs with a variable number of states. States have the same applicable actions that lead to neighbor states with some probability of success, and a probability of "sweeping" to another neighbor state. Some states are removed in order to make the domain more complex, creating dead-ends randomly. We generated both:

- *general case problems* where 1 % of rewards or costs are randomly spread in $[-1; 1]$;
- *shortest stochastic path* problems where a goal state is randomly defined that is absorbing and leads by itself to all rewards of the domain; all other actions have the same uniform cost -1.

As RTDP and T$_\rho$FSP (with $\rho < 1$) do not compute a policy for every reachable state, we use them as "real-time" algorithms, meaning that we simulate the execution of the policy and recompute a policy whenever an unknown state is reached. In this case, a new policy starting from this unknown state is computed, and played. For shortest stochastic path, this is done until goal is reached, while in general case, this is done over a given horizon H for which rewards gathered at timestep $H + 1$ has no more influence on the optimal value of the initial state (Puterman 1994).

Heuristic: uniform upper bound

The uniform upper bound of the optimal values of states presented in paragraph is used to solve general case problems. Clearly, this heuristic is quite poor in the sense that no tip state is preferred to the others during the search. Yet, such heuristic is still usefull because it allows to stop the search as soon as explored rewards become better than any unexplored rewards (whose cumulated value is guaranteed to be less than the heuristc value of tip states).

Table presents a comparison of computation time and value of the initial state. 3×10^5 is the size of the biggest problem we could generate on our Centrino Duo computer with 2Go of memory. These results confirm the optimality of FSP and show that FSP is slightly faster than LAO*. RTDP is always the fastest algorithm, but it is not optimal. Moreover, it appears that the use of thresholded reachability does not help when the heuristic is very little informative: the current policy has a high chance to be very different from the optimal one, so that good states are too often pruned by T$_\rho$FSP ($\rho < 1$). The next subsection shows that results are totally different for good heuristics.

Heuristic: Manhattan distance

For shortest stochastic path problems, we use the Manhattan distance heuristic (Barto, Bradtke, and Singh 1995)

size	RTDP	LAO*	FSP	$T_{0.95}$FSP	$T_{0.9}$FSP	$T_{0.85}$FSP
10^2	**0.01**	**0.008**	**0.002**	**0.0218**	**0.0237**	**0.0282**
	1.63	*1.58*	*1.61*	*1.58*	*1.53*	*1.55*
10^3	**0.188**	**0.418**	**0.14**	**10.2**	**1.62**	**1.21**
	2.16	*2.42*	*2.43*	*2.08*	*1.92*	*1.68*
10^4	**0.443**	**0.96**	**0.442**	**20.9**	**4.22**	**2.51**
	1.62	*1.81*	*1.81*	*1.28*	*0.683*	*0.608*
10^5	**0.657**	**1.37**	**1.07**	**19.9**	**3.03**	**2.06**
	2.19	*2.53*	*2.54*	*1.93*	*1.32*	*0.787*
3.10^5	**2.16**	**3.75**	**3.26**	**31.5**	**4.54**	**2.9**
	2.03	*2.31*	*2.27*	*1.68*	*1.3*	*0.89*

Table 1: General problems (10 random problems per size, 20 simulation trials per problem): the first column is the number of states — for each cell, the **bold** line is the average computation time in seconds, and the *italic* line is the average value of the initial state

size	RTDP	LAO*	FSP	$T_{0.95}$FSP	$T_{0.9}$FSP	$T_{0.85}$FSP
10^2	**0.0024**	**0.002**	**0.004**	**0.0078**	**0.00585**	**0.00395**
	-5.63	*-5.52*	*-5.61*	*-5.6*	*-5.69*	*-5.6*
10^3	**0.0154**	**0.086**	**0.057**	**0.233**	**0.0531**	**0.0348**
	-7.37	*-7.38*	*-7.37*	*-7.4*	*-7.37*	*-7.37*
10^4	**0.611**	**12.7**	**22.2**	**14.4**	**1.02**	**0.481**
	-9.98	*-9.98*	*-9.98*	*-9.99*	*-9.99*	*-9.99*
10^5	**18.6**	**86.6**	**140**	**17**	**1.14**	**0.612**
	-9.99	*-9.99*	*-9.99*	*-9.99*	*-9.99*	*-10*
3.10^5	**23**	**158**	**23.5**	**11.5**	**2.36**	**2.01**
	-10	*-10*	*-10*	*-10*	*-10*	*-10*

Table 2: Shortest stochastic path problems (10 random problems per size, 20 simulation trials per problem): the first column is the number of states — for each cell, the **bold** line is the average computation time in seconds, and the *italic* line is the average value of the initial state

which is the minimum number of steps required to reach the goal state from any state. Summary Table shows that $T_{\{0.85,0.9,0.95\}}$FSP outperforms LAO* and RTDP as the state space size increases. The computation time drastically decreases as ρ decreases, without affecting the optimality of the solution.

Conclusion

We have proposed a new heuristic algorithm named T_ρFSP that performs probabilistic reachability analysis in order to prune the states that have a low chance to be reached from a given initial state. For general MDPs problems solved with a "poor" heuristic, T_1FSP is slightly faster than LAO* while being optimal. For shortest stochastic path problems, for which a "good" heuristic is known, T_ρFSP ($\rho < 1$) outperforms LAO* and RTDP.

One interesting feature of the T_ρFSP algorithm class is that any update fo the policy may be used, and that probabilistic reachability may be defined for non-discrete state spaces. One of our future goals is to use this algorithmic framework to work on hybrid MDPs (Guestrin, Hauskrecht, and Kveton 2004), where state space is both continuous and discrete. Another interesting perspective of this work is its extension to symbolic factored representations using decision diagrams (Feng and Hansen 2002).

Acknowledgment

This research was supported by the French *Délégation générale pour l'armement* grant 07.60.031.00.470.75.01.

References

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

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

Buffet, O., and Aberdeen, D. 2007. Ff+fpg: Guiding a policy-gradient planner. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*, volume 17.

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

Guestrin, C.; Hauskrecht, M.; and Kveton, B. 2004. Solving factored MDPs with continuous and discrete variables. In *Proceedings of UAI*.

Hansen, E. A., and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129:35–62.

O., O. B., and Aberdeen, D. 2006. The factored policy gradient planner (ipc-06 version). In *Proceedings of the Fifth International Planning Competition*.

Poupart, P.; Boutilier, C.; Patrascu, R.; and Schuurmans, D. 2002. Piecewise linear value function approximation for factored mdps. In *Proceedings 18th AAAI*, 292–299.

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