# Planning with Robust (L)RTDP

Olivier Buffet, Douglas Aberdeen

# Planning with Robust (L)RTDP

**Olivier Buffet & Doug Aberdeen**
National ICT Australia &
The Australian National University
Canberra, Australia
{olivier.buffet, doug.aberdeen}@nicta.com.au

November 15, 2004 - Revised March 14, 2005

## Abstract

Stochastic Shortest Path problems (SSPs), a subclass of Markov Decision Problems (MDPs), can be efficiently dealt with using *Real-Time Dynamic Programming* (RTDP). Yet, MDP models are often uncertain (obtained through statistics or guessing). The usual approach is robust planning: searching for the best policy under the worst model. This paper shows how RTDP can be made robust in the common case where transition probabilities are known to lie in a given interval.

## Contents

# 1 Introduction

In decision-theoretic planning, Markov Decision Problems [Bertsekas and Tsitsiklis, 1996] are of major interest when a probabilistic model of the domain is available. A number of algorithms make it possible to find plans (policies) optimizing the expected long-term utility. Yet, optimal policy convergence results all depend on the assumption that the probabilistic model of the domain is accurate.

Unfortunately, a large number of MDP models are based on uncertain probabilities (and rewards). Many rely on statistical models of physical or natural systems, such as plant control or animal behavior analysis. These statistical models are sometimes based on simulations (themselves being mathematical models), observations of a real system or human expertise.

Working with uncertain models first requires answering two closely related questions: 1– how to model this uncertainty, and 2– how to use the resulting model. Existing work shows that uncertainty is sometimes represented as a set of possible models, each assigned a model probability [Munos, 2001]. The simplest example is a set of possible models that are assumed equally probable [Bagnell *et al.*, 2001; Nilim and Ghaoui, 2004]. Rather than construct a possibly infinite set of models we represent model uncertainty by allowing each probability in a single model to lie in an interval [Givan *et al.*, 2000; Hosaka *et al.*, 2001].

Uncertain probabilities have been investigated in

- resource allocation problems [Munos, 2001], such as efficient exploration [Strehl and Littman, 2004] and state aggregation [Givan *et al.*, 2000], and
- policy robustness [Bagnell *et al.*, 2001; Hosaka *et al.*, 2001; Nilim and Ghaoui, 2004].

We focus on the later, considering a two-player game where the opponent chooses one of the possible models to reduce the long-term utility.

Our principal aim is to develop an efficient planner for a common sub-class of MDPs for which all policies are guaranteed to eventually terminate in a goal state: Stochastic Shortest Path (SSP) problems. The greedy *Real-Time Dynamic Programming* algorithm (RTDP) [Barto *et al.*, 1995] is particularly suitable for SSPs, as it finds good policies quickly and does not require complete exploration of the state space.

This paper shows that RTDP can be made robust. In Section 2, we present SSPs, RTDP and robustness. Then Sec. 3 explains how RTDP can be turned into a robust algorithm. Finally, experiments are presented to analyse the behavior of the algorithm, before a discussion and conclusion.

# 2 Background

## 2.1 Stochastic Shortest Path Problems

A Stochastic Shortest Path problem [Bertsekas and Tsitsiklis, 1996] is defined here as a tuple $\langle S, s_0, G, A, T, c\rangle$. It describes a control problem where $S$ is the finite set of **states** of the system, $s_0 \in S$ is a starting state, and $G \subseteq S$ is a set of goal states. $A$ is the finite set of possible **actions** $a$. Actions control transitions from one state $s$ to another state $s'$ according to the system's probabilistic dynamics, described by the **transition function** $T$ defined as $T(s,a,s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$. The aim is to optimize a performance measure based on the **cost function** $c : S \times A \times S \to \mathbb{R}^+$.[1]

SSPs assume a goal state is reachable from any state in $S$, at least for the optimal policy, so that one cannot get stuck in a looping subset of states. An algorithm solving an SSP has to find a **policy** that maps states to probability distributions over actions $\pi : S \to \Pi(A)$ which optimizes the **long-term cost** $J$ defined as the expected sum of *costs* to a goal state.

[1] As the model is not certain, we do not make the usual assumption $c(s,a) = \mathbb{E}_{s'}[c(s,a,s')]$.

In this paper, we only consider SSPs for planning purposes, with full knowledge of tuple defining the problem: $\langle S, s_0, G, A, T, c\rangle$. In this framework, well-known stochastic dynamic programming algorithms such as *Value Iteration* (VI) make it possible to find a deterministic policy that corresponds to the minimal expected long-term cost $J^*$. Value Iteration works by computing the function $J^*(s)$ that gives the expected sum of costs of the optimal policies. It is the unique solution of the fixed point Bellman equation:

$$J(s) \quad = \quad \min_{a \in A} \sum_{s' \in S} T(s,a,s') \left[c(s,a,s') + J(s')\right]. \tag{1}$$

Updating $J$ with this formula leads to asymptotic convergence to $J^*$. For convenience, we also introduce the $Q$-value:

$$Q(s,a) = \sum_{s' \in S} T(s,a,s')[c(s,a,s') + V(s')].$$

SSPs can easily be viewed as shortest path problems where choosing a path only probabilistically leads you to the expected destination. They can represent a useful sub set of MDPs, as they are essentially finite-horizon MDPs with no discount factor.

## 2.2 RTDP

Trial based[2] *Real-Time Dynamic Programming* (RTDP), introduced in [Barto *et al.*, 1995], uses the fact that the SSP costs are positive and the additional assumption that every trial will reach a goal state with probability 1. Thus, with a zero initialization of the long-term cost function $J$, both $J$ and $Q$-values monotonically increase during their iterative computation.

The idea behind RTDP (Algorithm 1) is to follow paths from the start state $s_0$, always greedily choosing actions with the lowest long-term cost and updating $Q(s,a)$ as states $s$ are encountered. In other words, the action chosen is the one expected to lead to the lowest future costs, until the iterative computations show that another action may do better.

**Algorithm 1** RTDP algorithm for SSPs

```
RTDP(s:state) // s = s_0
repeat
  RTDPTRIAL(s)
until // no termination condition

RTDPTRIAL(s:state)
while ¬GOAL(s) do
  a =GREEDYACTION(s)
  J(s)=QVALUE(s, a)
  s =PICKNEXTSTATE(s, a)
end while
```

RTDP has the advantage of quickly avoiding plans that lead to high costs. Thus, the exploration looks mainly at a promising subset of the state space. Yet, because it follows paths by simulating the system's dynamics, rare transitions are only rarely taken into account. The use of a simulation makes it possible to get good policies early, but at the expense of a slow convergence, because of the bad update frequency of rare transitions.

[2]We always assume a trial based RTDP implementation.

## 2.3 Robust Value Iteration

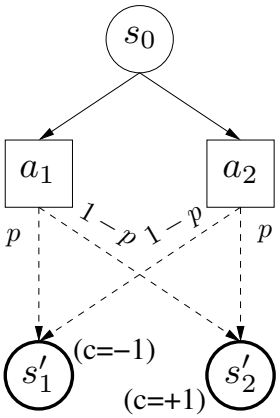
**Pessimism and Optimism —** We now turn to the problem of taking the model's uncertainty into account when looking for a "best" policy. The (possibly infinite) set of alternative models is denoted $\mathcal{M}$.

A simplistic approach computes the average model over $\mathcal{M}$, or the most probable model in $\mathcal{M}$, then uses standard SSP optimization methods. Such approaches guarantee nothing about the long-term cost of the policy if the true model differs from the one chosen for optimization.

We follow the approach described in [Bagnell *et al.*, 2001], that consists of finding a policy that behaves well under the worst possible model. This amounts to considering a two-player zero-sum game where a player's gain is its opponent's loss. The player chooses a policy over actions while its "disturber" opponent simultaneously chooses a policy over models (as this is a simultaneous game, optimal policies can be stochastic). This results in a max-min-like algorithm:

$$\max_{\pi_{\mathcal{M}} \in \Pi_{\mathcal{M}}} \min_{\pi_A \in \Pi_A} J_{\pi_{\mathcal{M}}, \pi_A}(s_0).$$

As illustrated by Figure 1, this is a Stochastic Shortest-Path Game (SSPG) as described in [Patek and Bertsekas, 1999], where are proved the existence of a solution and the convergence of Value Iteration.




$a_1$: `shoot-left`
$a_2$: `shoot-right`
$s'_1$: success
$s'_2$: failure

A simple example to show why this problem is a simultaneous game is that of a player trying to score a goal, having the choice between two actions: `shoot-left` ($a_1$) and `shoot-right` ($a_2$). The environment's dynamics are mainly linked to the goal-keeper's behavior, which may (randomly) `dive-left` or `dive-right`. The optimal policy for the "opponent" is a mixture strategy: probability $p$ to `dive-left` and $1 - p$ to `dive-right`, with $p \in [0, 1]$. The result of both decisions is either a success ($s'_1$) or a failure ($s'_2$). If the goal-keeper's behavior were deterministic, it would be easy to adapt and always score goals.

Figure 1: A simple example to show why this problem is a simultaneous game

It is also possible to be optimistic, considering that both players collaborate (as they endure the same costs), which turns the $\max$ into a $\min$ in previous formula. This second case is equivalent to a classical SSP where a decision consists of choosing an action and a local model.

**Locality —** Such a max-min algorithm would be particularly expensive to implement. Even restricting the search to a deterministic policy over models, it requires computing the optimal long-term cost function for each model before picking the worst model found and the optimal policy associated with it. However, a simpler process may be used to compute $J$ while looking simultaneously for the worst model. It requires the hypothesis that next-state distributions $T(s, a, \cdot)$ are independent from one state-action

pair $(s, a)$ to another. This assumption does not always hold. However, this makes things easier for the opponent because it now has larger set of models from which to choose. This consequence is conservative for producing robust policies.

Because we assume independence at the state-action level (not only at the state level), it is equivalent to a situation where the second player makes a decision depending on the current state and the first player's action. This situation amounts to a sequential game (see Fig. 2) where the previous players move is known to the next player: both players can act in a deterministic way without loss of efficiency.

Coming back to the goal-scoring example, the independence assumption amounts to say that the behavior of the goal-keeper will not be the same depending on the player's action (what is here not realistic). In this case, the worst case for the player is the goal-keeper jumping right (respectively left when action $a_1$ (resp. $a_2$) is chosen i.e., $p_1 = 0$ and $p_2 = 1$.

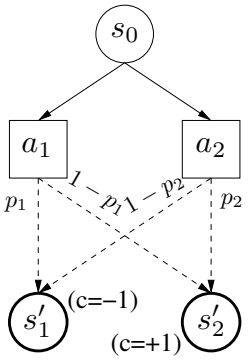



Figure 2: How the independence assumption turns a simultaneous game in a sequential one

The result of this assumption is that the worst model can be chosen locally when $Q$ is updated for a given state-action pair. As can be seen from Algorithm 2, the worst local model $m_s^a$ may change while $Q$-values evolve. Previous updates of reachable states' long-term costs may change their relative ordering, changing which outcomes are considered to be worst.

**Algorithm 2** Robust Value Iteration (for an SSP)

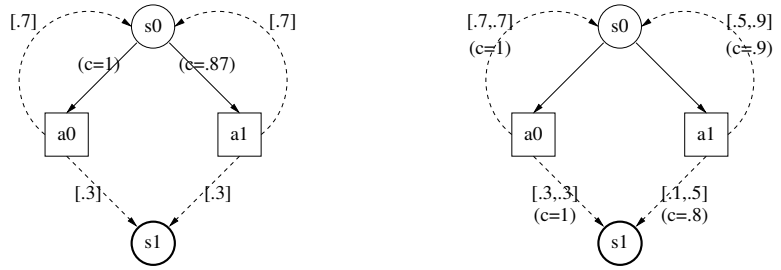
Initialize $J$ to 0.
**repeat**
  **for all** $s$: state **do**
    **for all** $a$: action **do**
      $Q_{\max}(s,a) \leftarrow \max_{m_s^a \in \mathcal{M}_s^a} \sum_{s' \in S} T_{m_s^a}(s,a,s') \left[ J(s') + c_{m_s^a}(s,a,s') \right]$
    **end for**
    $J(i) \leftarrow \min_{a \in A} Q_{\max}(s,a)$
  **end for**
**until** $J$ converges

The key contribution of this paper is to show that RTDP can be made *robust*, allowing for planning in very large and uncertain domains, retaining worst (or best) case behaviour guarantees.

## 3 Robust RTDP

From now on, we consider **interval-based** uncertain SSPs, where $T(s,a,s')$ is known to be in an interval $[Pr^{\min}(s'|s,a), Pr^{\max}(s'|s,a)]$. Figure 3 is an example. We discuss the pessimistic approach, the optimistic one leading to similar results.

For a given state-action pair $(s,a)$, there is a list $R = (s'_1, \cdots, s'_k)$ of reachable states. For each reachable state $T(s,a,s'_i) \in I_i = [p_i^{\min}, p_i^{\max}]$. Thus, pos-




a) certain SSP b) uncertain SSP

Figure 3: Two views of one SSP, depending on whether model uncertainty is taken into account (costs in parenthesis). In the uncertain SSP, action $a_0$ will be prefered as it quickly reaches the goal $s_1$.

sible models are the ones that comply with these interval constraints while ensuring $\sum_i T(s,a,s'_i) = 1$. Fig. 4 illustrates this with three reachable states.

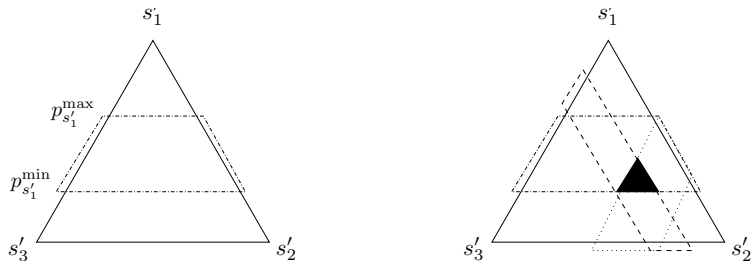



Figure 4: A triangle is a probability simplex representing all possible probability distributions with three different outcomes ($Pr(s'_i) = 1$ at the $s'_i$ vertex). On the left triangle is the trapezium showing the interval constraint for $s'_1$. The right triangle shows possible models at the intersection of the three interval constraints.

**Worst Local Models —** The maximisation step to compute $Q(s,a)$ in Alg. 2 is done by giving the highest probability to the worst outcome. This requires firstly sorting reachable states in decreasing order of the values: $c(s,a,s'_1) + J(s'_1) \geq c(s,a,s'_2) + J(s'_2) \geq \cdots c(s,a,s'_k) + J(s'_k)$. Then, the worst distribution is the one giving the highest probability to the first state $s'_1$, then to $s'_2$, and so on up to $s'_k$. As pointed out in [Givan *et al.*, 2000], this is equivalent to finding the highest index $r \in [1..k]$ such that

$$\sum_{i=1}^{r-1} p_i^{\max} + \sum_{i=r}^{k} p_i^{\min} \leq 1.$$

The resulting transition probabilities are

$$Pr(s'_i) = \begin{cases} p_i^{\max} \text{ if } i < r \\ p_i^{\min} \text{ if } i > r \end{cases} \quad (2)$$

$$Pr(s'_r) = 1 - \sum_{i=1, i\neq r}^{k} Pr(s'_i). \quad (3)$$

Using the pre-computed bound $B_{\min} = \sum_{i=1}^{k} p_i^{\min}$, Alg. 3 gives a complete implementation. The *insertion sort* algorithm[3] is chosen as the list will usually be ordered

[3] `http://en.wikipedia.org/wiki/Insertion_sort`

from previous $J$ updates.

**Algorithm 3** Worst Model for State-Action Pair $(s, a)$

WORSTMODEL($s$: state, $a$: action)
$R = (s'_1, \cdots, s'_k) =$ REACHABLESTATES($s$,$a$)
SORT($R$)
$i = 1$, $bound = B_{\min}$
**while** $(bound - p_i^{\min} + p_i^{\max} < 1)$ **do**
  $bound \leftarrow bound - p_i^{\min} + p_i^{\max}$
  $Pr(s'_i) \leftarrow p_i^{\max}$
  $i \leftarrow i + 1$
**end while**
$r = i$
$Pr(s'_r) \leftarrow 1 - (bound - p_r^{\min})$
**for all** $i \in \{r + 1, \ldots, k\}$ **do**
  $Pr(s'_i) \leftarrow p_i^{\min}$
**end for**
return $(R, Pr(\cdot))$

To summarise, Robust VI on an interval-based SSP consists of applying normal Value Iteration with the transition probabilities updated through Alg. 3.

We need only a single worst model to compute the corresponding $Q$-value. Yet, because several reachable states $s'_i$ may have the same value $c(s, a, s'_i) + J(s'_i)$ as $s'_r$ (we call this set of states $S'_r$), there may be an infinite number of equivalent worst local models. Any model differing only in how the probability mass is distributed among the equally bad states of $S'_r$ is also a worst local model.

**Worst Global Models —** Contrary to VI, RTDP does not necessarily visit the complete state-space. This is why [Barto *et al.*, 1995] introduces the notion of *relevant* states, which we extend to the uncertain case: a state $s$ is said to be *relevant* for $\mathcal{M}$ if there exists a start state $s_0$, a model $m \in \mathcal{M}$ and an optimal policy $\pi$ under that model such that $s$ can be reached from state $s_0$ when the controller uses that policy under that model.

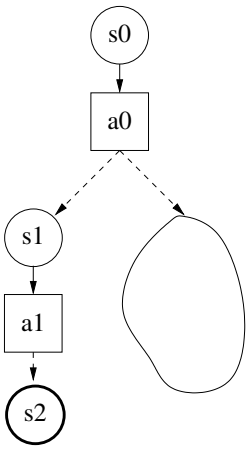
This notion is important because two equally worst local models on a given state-action pair may forbid different states, so that for two models $m_1$ and $m_2$, a state may be relevant in $m_1$ but not in $m_2$. Yet, RTDP should not find an optimal policy just for the relevant states of only one worst global model. Neither does the policy have to apply to all possible states. It should apply to all *reachable* states under *any* model (for optimal policies) i.e., for relevant states. But covering all relevant states in the worst model used for updating $Q$-values does not necessarily cover all relevant states in $\mathcal{M}$: it depends on the model used to choose the next state, i.e., to simulate the system's dynamics.

To avoid missing relevant states, each local model used for the simulation should ensure that all reachable states can be visited. As can be seen in Fig. 6, the set of possible local models for a state-action pair is an $n$-dimensional convex polytope. Any model *inside* this polytope, excluding the boundary, is therefore adequate because, for all $s'_i$, it ensures that $P(s'_i|s, a) > 0$.

So there exists a global model $m_d$ that can be used to effectively simulate the system's dynamics without missing any potentially reachable state.

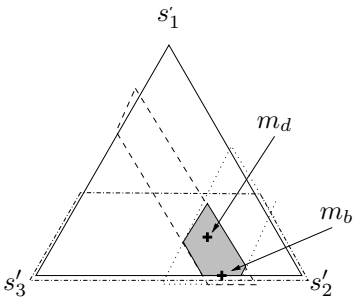
### 3.1 Robust (Trial-Based) RTDP

*Robust* RTDP differs from the original RTDP in that:




Transition $(s_0, a_0) \rightarrow s_1$ (and any other transition leading to $s_1$ may be forbidden in all worst models. Yet, robust planning should tell what to do from state $s_1$, in case the true model can lead to it.

Figure 5: Why relevant states for $\mathcal{M}$ may not be relevant for any worst global model.




A model on the boundary of the simplex (as $m_b$) forbids at least one potentially reachable state. Thus, the model used to simulate the dynamics should be *inside* the polytope of possibles models, in a view to ensure that all relevant states of $\mathcal{M}$ are visited.

Figure 6: Why only a model *inside* the polytope of possible models can be used to simulate the system's dynamics

- Each time the algorithm is updating a state's evaluation, an opponent is looking for a worst local model which serves to compute the $Q$-values.
- For exploration purposes, the algorithm follows dynamics of the system that consider all possible transitions (using model $m_d$).
- "Relevant" states are now the states reachable by following any optimal policy under any possible model.

From this, we adapt to our context convergence Theorem 3 from [Barto *et al.*, 1995] and the corresponding proof, discussing mainly its changes.

**Theorem 1.** *In* **uncertain** *undiscounted stochastic shortest path problems,* **robust** *Trial-Based RTDP with the initial state of each trial restricted to a set of start states, converges (with probability one) to $J^*$ on the set of relevant states, and the controller's policy converges to an optimal policy (possibly nonstationary) on the set of relevant states, under the following conditions:*

1. *the initial cost of every goal state is zero,*
2. *there is at least one proper policy,* [4]

[4] If trials are allowed to time out before a goal state is reached, it is possible to eliminate the requirement that at least one proper policy exists. Timing out prevents getting stuck in fruitless cycles, and the time-out period can be extended systematically to ensure that it becomes long enough to let all the optimal paths be followed without interruption.

3. *all immediate costs incurred by transitions from non-goal states are positive, i.e.,* $c_i(a) > 0$ *for all non-goal states* $i$ *and actions* $a \in A(i)$*, and*
4. *the initial costs of all states are non-overestimating, i.e.,* $J_0(i) \leq J^*(i)$ *for all states* $i \in S$*.*

*Proof.* (more details in Appendix A) The proof in [Barto *et al.*, 1995] shows that states infinitely updated by RTDP are all the relevant ones, so that a classical convergence proof on SSPs can be invoked if restricting the SSP to relevant states.

A first remark is that introducing $\max_{m \in \mathcal{M}}$ in the update formula does not change the fact that $J_t$ is increasing and non-overestimating.

In our case, the use of the model $m_d$ ensures similarly that states infinitely updated by *robust* RTDP are all relevant ones (of the uncertain SSP).

We have established that we are in sequential Stochastic Shortest Path Games (SSPGs). Bertsekas and Tsitsiklis [1996] establish that these are a special case of general SSPGs. The convergence of general SSPGs is proved in Patek and Bertsekas [1999] (Proposition 4.6), which states that long-term costs converge with probability 1 on the set of relevant states. □

Whatever the true model, the algorithm learns all optimal decisions in any relevant state under the more pessimistic assumption. Importantly, a relevant state $s$ may be not be reachable through a worst global model, but the true environment may lead to that state. So the policy must cover all relevant states but always assumes that the worst model will apply afterwards.

## 4 Experiments

*Labelled* RTDP [Bonet and Geffner, 2003] is a modified version of RTDP which can be made robust in a similar way. The experiments conducted illustrate the behavior of *robust LRTDP*. To that end, it is compared to Bagnell *et al.*'s *Robust Value Iteration*, as well as *LRTDP*. In all cases, the convergence criteria is $\epsilon = 10^{-3}$ for LRTDP, and for VI we stop when the maximum change in a state long-term cost over one iteration is less than $10^{-3}$.

### 4.1 Heart

In this first experiment, we compare a non-robust optimal policy with a robust one on the small example of Fig. 3-b. Table 1 shows the theoretical expected long-term costs of each policy on the normal (most probable) model, as well as the pessimistic and optimistic ones. The robust policy is largely better in the pessimistic case.

Table 1: Theoretical evaluation of robust and non-robust policies on various models, exactly matched by empirical evaluation.

| | Normal | Pessimistic | Optimistic |
|---|---|---|---|
| Non-robust | 2.90 | 8.90 | 1.70 |
| Robust | 3.33 | 3.33 | 3.33 |

Fig. 7 shows the theoretical cost-to-goal, integrated over all possible models in the interval, for a given interval $2r$. Each point represents the integral calculation divided by the width of the interval.

This allows us to state, for example, that for an interval of 0.4 (corresponding to Fig. 3-b and Table 1), the optimal *non*-robust policy has an average cost-to-goal of 3.52 *over all possible models*, and since the average cost-to-goal for the robust case is a constant 3.33, the robust policy is safer if we do not know which model represents reality. This assumes all models are equally probable. The limit case $r = .3$ corresponds to the goal $s_1$ being unreachable through action $a_1$ which loops to $s_0$.

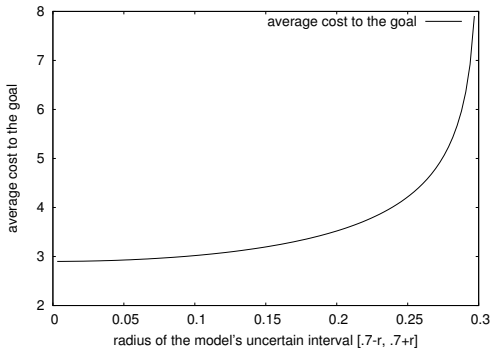



Figure 7: Average cost of the non-robust policy with equiprobable models but with variable uncertainty.

### 4.2 Mountain-Car

We use here the mountain-car problem as defined in [Sutton and Barto, 1998]: starting from the bottom of a valley, a car has to get enough momentum to reach the top of a mountain (see Fig. 8). The same dynamics as described in the mountain car software[5] have been employed. The objective is to minimize the number of time steps to reach goal.

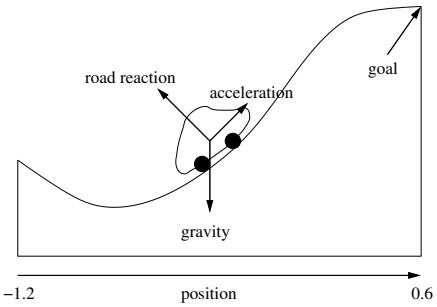



Figure 8: The mountain-car problem.

- State space: $-1.2 \leq x \leq 0.6$ and $-0.07 \leq v \leq 0.07$.
- Actions: $a = +1$ (go right) and $a = -1$ (go left).
- Dynamics of the system [Sutton and Barto, 1998] (time-step = 1 second):

$$\begin{cases} x_{t+1} &= x_t + v_t \\ v_{t+1} &= v_t + 0.001 * a_t - 0.0025 * \cos(3.x_t) \end{cases}$$

- Cost: 1 per time-step (second).
- Starting situation: $x = -0.5$, $v = 0$
- Goal: $x \geq 0.5$ (corresponds to an additional goal state)

The continuous state-space is discretized ($32 \times 32$ grid) and the corresponding uncertain model of transitions is obtained by sampling $1000$ transitions from each state-action pair $(s, a)$. For each transition, we computed intervals in which the true model lies with $95\%$ confidence (using the process described in Appendix B.1).

[5] `http://www.cs.ualberta.ca/~sutton/...MountainCar/MountainCar.html`

**Results —** Preliminary remark: simulating a path generally shows a car oscillating several times before leaving the valley. This has two main explanations: 1- the speed gathering is just sufficient to reach the summit, but not over it; and 2- the discretized model is not accurate enough and applying the policy obtained on the true mathematical model (instead of the discretized one) should be much better.

Fig. 9 shows the long-term cost function obtained by using *value iteration*, LRTDP, and their robust counterparts on the mountain-car problem. The $x$ and $y$ axes are the car's position and speed. The $z$ axis is the expected cost to the goal. On the surface is an example path from the start state to a goal state: it follows the greedy policy under the average model.

The general shape of the surface obtained is always the same, with some unexplored parts of the state-space in LRTDP and Robust LRTDP (as expected). The vertical scales are much larger in robust cases. This reflects the fact that reaching the goal is much more time-consuming under a pessimistic model. Because $J$ can here be interpreted as the average time to the goal, these graphs show how a small uncertainty can lead to longer policies. Here the times are multiplied by more than $2.5$.

While executing the four different algorithms, an evaluation of the current greedy policy was made every $10 * nStates = 10\,240$ $Q$-value updates. The result appears in Fig. 10, the $y$ axis being the expected cost to the goal from the start state. On both sub-figures, LRTDP-based algorithms obtain good policies quickly, but have slow convergence times of VI=$2.83 \times 10^6$ updates, LRTDP=$6.76 \times 10^6$, rVI=$8.31 \times 10^6$, rLRTDP=$11.06 \times 10^6$.

### 4.3 Sailing

The sailing problem used here shares similarities with the mountain-car task. It's complete description can be found in [Vanderbei, 1996], and another use in [Peret and Garcia, 2004]. Here, the space is discretized to a $10 \times 10$ grid, $\times 8$ wind angles and $\times 8$ possible headings. The uncertainty of the system is due to the stochastic changes in the wind's direction. The uncertain model is also learned by drawing $1000$ samples for each state-action pair, using the same $\alpha = 0.05$.

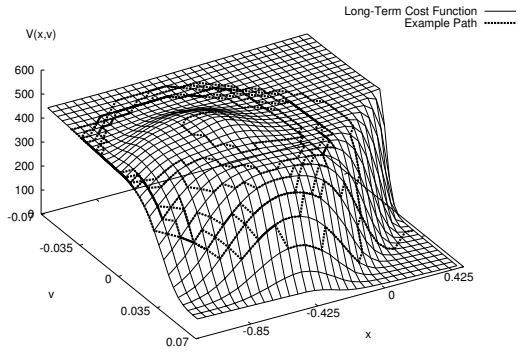
**Results**

The same tests as in the mountain-car problem have been conducted. The long-term cost functions obtained show similar phenomena such as the time increase. Only Figure 11 is of particular interest, as it shows how quickly LRTDP-like algorithms converge. In this higher-dimensional problem, finding solutions takes more time at the beginning, but LRTDP proves to be very efficient at pruning inefficient paths. In fact, most solved states are along the main diagonal of the lake (most side states can be avoided by the optimal policy). For the various algorithms, the time to converge is: VI=$3.67 \times 10^6$, LRTDP=$0.49 \times 10^6$, rVI=$5.22 \times 10^6$, rLRTDP=$0.60 \times 10^6$.

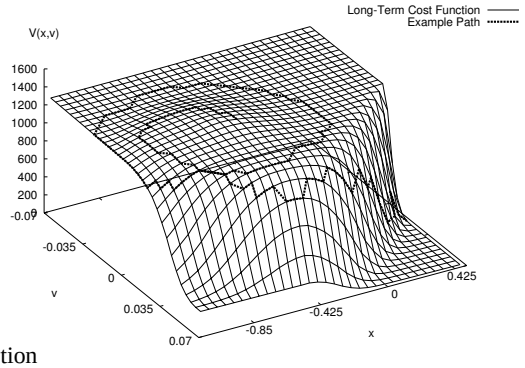
### 4.4 Temporal Decision-Theoretic Planning

Our work with the Australian Defence Science Technology Organisation is exploring a general probabilistic planning domain that considers concurrent tasks, resource allocation, and uncertain outcomes [Aberdeen *et al.*, 2004]. The aim is to maximize the probability of operation success while also minimizing plan duration and resource use. Our planner accepts a modified version of Probabilistic PDDL [Younes and Littman, 2004]. All actions are grounded (no variables), and only conjunctions of (possibly negative) effects and conditions are allowed. Effects and conditions can be applied at the start or end of tasks, as well as over the duration of a task. The space of plans described by our language corresponds closely with the types of plans that can be entered using planning aids such as Microsoft Project.

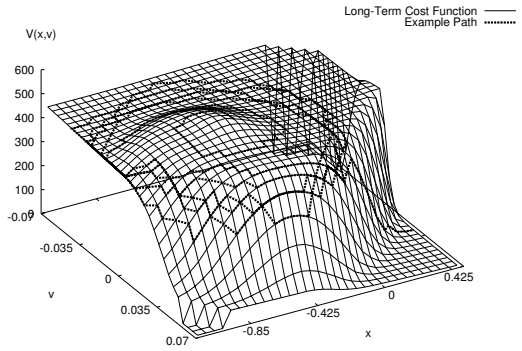
Planning staff use a graphical interface to specify tasks, their preconditions, effects, and resource requirements. They also specify a probability of task failure (which selects different effects and resource use). Staff use their expertise and intuition to choose



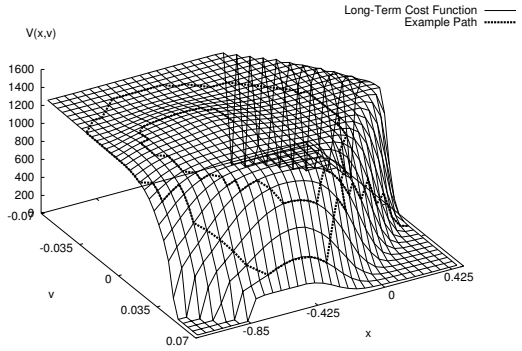

a) Value Iteration




b) Robust Value Iteration

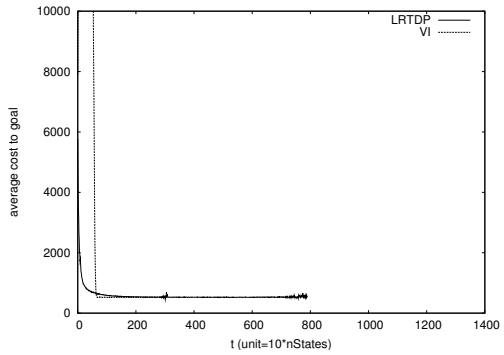



c) LRTDP

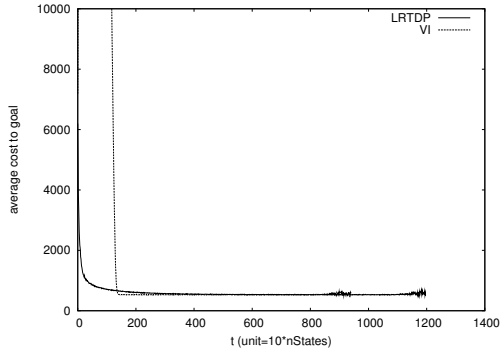



d) Robust LRTDP

Figure 9: Long-term cost functions for the mountain-car problem
In all cases, the most likely model is used to generate the example path.




a) Non-Robust Algorithms

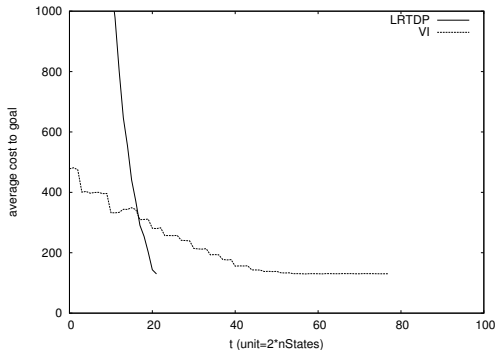



b) Robust Algorithms

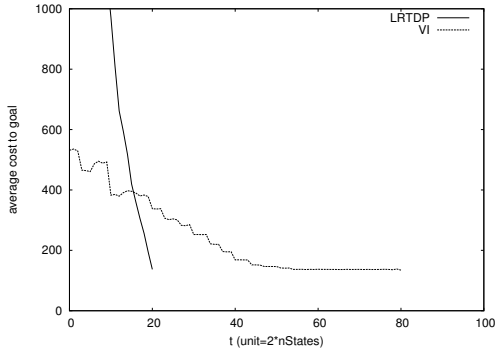
Figure 10: Average cost to goal for the mountain-car problem, measured every $10 * nStates$ updates of $Q$-values.

probabilities, but they are often slightly, or entirely, wrong. For example, in the building industry, the success of a concrete pour might be effected by the probability of rain over the period of the pour — a highly uncertain quantity. There is no feedback mechanism available to check the accuracy of probabilities because the plan is rarely implemented more than once in the real world. If humans also provide a confidence interval on their probability assignments we can significantly improve the plan's worst case performance.

We experiment on two scenarios in our language. The first consists of 15 tasks designed to represent the high level process of building a sky-scraper. These tasks achieve a set of 10 conditions needed for operation success. Four of the effects can be established independently by two different tasks, however resource constraints only allow one of the tasks to be chosen. We have constructed this example to demonstrate the effectiveness of planning with intervals. Thus, for each effect that has two tasks that can achieve it, we have selected one task to have a high probability of success, but also a high uncertainty. The second task has a lower probability of success, but a an interval of 0. The second (lower) probability of success is chosen to be higher than the *lower bound* on the first tasks success probability. The robust plan should (and does) choose tasks with the lower probability of success, but zero interval. Table 2 shows that the results of evaluating the robust and non-robust plans assuming: 1- the original human specified model, 2- the pessimistic model, 3- the optimistic model. Because optimization has a stochastic component, the results presented are the average over 100 training runs each evaluated with 1,000,000 simulations of the resulting plan. Each LRTDP plan optimization ran for a maximum of 10s.




a) Non-Robust Algorithms




b) Robust Algorithms

Figure 11: Average cost to goal for the sailing problem measured every $2 * nStates$ updates of $Q$-values.

We can see that the robust plans perform much better in the pessimistic case, and in general have more uniform cost over different models (but this is not necessarily the case for other domains). However, averaged over just these 3 extreme points, the non-robust has an advantage demonstrating that this method may cause you to be non-optimal under most models.

The second data set is a small military operations planning scenario based on capturing a small island [Aberdeen *et al.*, 2004]. It defines 18 tasks and 11 conditions based on a real military planning data. The original data does not define uncertainty intervals so we added a random interval to each task probability (maximum interval of $\pm 30\%$). The point of this experiment is to demonstrate the sensitivity of the plans to changes in probabilities. On a single plan optimized for 10 minutes both robustly and non-robustly, and evaluated over 1,000,000 evaluation trials, the worst case long-term cost dropped from 1066 to 972 under robust planning and resulted in a different plan.

| | Norm. | Pessimistic | Optimistic | Avg. |
|---|---|---|---|---|
| Non-robust | 284 | 732 | 114 | 376 |
| Robust | 500 | 563 | 404 | 489 |

Table 2: Average total cost of the optimized Building plan. Cost is a function of failure probability, plan duration, and resource use; however in these experiments the cost is dominated by failure probability. (The failure probability is approximately the total cost/1000.)

In general a small change in probability can result in a large change in plan. Planning with intervals can provide some assurance that a small error in probabilities will not catastrophically worsen the average total cost.

## 5 Discussion and Conclusion

An interesting work on robust MDPs is Hosaka *et al.* [2001]. The developed theory is difficult to implement, but it introduces the idea that one should not only look at policies that are optimal under the worst model but choose, among the policies that have optimal long-term cost for the worst models, one that is best under optimistic models. This is possible in the RTDP framework as it can be simply modified to assume the best model after optimizing for the worst model. A problem is to do this efficiently: only a pessimist long-term cost function can benefit from an optimist one (the later underestimates the former). This has recently been addressed (see Appendix C).

The approach to robustness adopted in this paper considers the knowledge of a set of possible models. An open question is whether it is possible to use more information from our uncertain model, by taking the probability distribution over possible models into account.

Model uncertainty has similarly been considered in model learning while planning [Strehl and Littman, 2004]. The algorithm proposed is optimistic, but does not seem to adapt well to our framework as an evolving model can break the "no-overestimation" assumption: $\forall s \in S,\ t \geq 0,\ J_t(s) \leq J^*(s)$. It is still important to notice that *robust* RTDP would not suffer on-line, as the real dynamics can be employed to pick a next state (the worst model appearing only in the long-term cost update formula).

Finally, a crucial assumption in RTDP is that a goal state must be reachable from any state. We present an algorithm tackling this problem in [Buffet, 2004].

**Conclusion —** Recent works show that model uncertainty is a major issue in decision-theoretic planning. We have proposed a modification of the RTDP algorithm enabling it to compute robust policies efficiently in large and uncertain domains. Model uncertainty is represented through confidence intervals on transition probabilities. The convergence of the resulting algorithm is proved. We demonstrate robust LRTDP on various domains where statistics are used to estimate appropriate intervals.

## Acknowledgments

National ICT Australia is funded by the Australian Government. This work was also supported by the Australian Defence Science and Technology Organisation.

## References

[Aberdeen *et al.*, 2004] D. Aberdeen, S. Thiébaux, and L. Zhang. Decision-theoretic military operations planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS'04)*, June 2004.

[Abramowitz and Stegun, 1972] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, 1972.

[Bagnell *et al.*, 2001] J.A. Bagnell, A. Y. Ng, and J. Schneider. Solving uncertain markov decision problems. Technical Report CMU-RI-TR-01-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2001.

[Barto *et al.*, 1995] A.G. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72, 1995.

[Bertsekas and Tsitsiklis, 1996] D.P. Bertsekas and J.N. Tsitsiklis. *Neurodynamic Programming*. Athena Scientific, 1996.

[Bonet and Geffner, 2003] B. Bonet and H. Geffner. Labeled rtdp: Improving the convergence of real time dynamic programming. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS'03)*, June 2003.

[Buffet, 2004] O. Buffet. Robust (l)rtdp: Reachability analysis. Technical report, National ICT Australia, december 2004.

[Givan *et al.*, 2000] R. Givan, S. Leach, and T. Dean. Bounded parameter markov decision processes. *Artificial Intelligence*, 122(1-2):71–109, 2000.

[Hosaka *et al.*, 2001] M. Hosaka, M. Horiguchi, and M. Kurano. Controlled markov set-chains under average criteria. *Applied Mathematics and Computation*, 120(1-3):195–209, 2001.

[Munos, 2001] R. Munos. Efficient resources allocation for markov decision processes. In *Advances in Neural Information Processing Systems 13 (NIPS'01)*, 2001.

[Nilim and Ghaoui, 2004] A. Nilim and L. El Ghaoui. Robustness in markov decision problems with uncertain transition matrices. In *Advances in Neural Information Processing Systems 16 (NIPS'03)*, 2004.

[Patek and Bertsekas, 1999] S. D. Patek and D. P. Bertsekas. Stochastic shortest path games. *SIAM J. on Control and Optimization*, 36:804–824, 1999.

[Peret and Garcia, 2004] L. Peret and F. Garcia. On-line search for solving markov decision processes via heuristic sampling. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'2004)*, 2004.

[Strehl and Littman, 2004] A. L. Strehl and M. L. Littman. An empirical evaluation of interval estimation for markov decision processes. In *Proceedings of the Sixteenth International Conference on Tools with Artificial Intelligence (ICTAI'04)*, 2004.

[Sutton and Barto, 1998] R. Sutton and G. Barto. *Reinforcement Learning: an introduction*. Bradford Book, MIT Press, Cambridge, MA, 1998.

[Vanderbei, 1996] Robert J. Vanderbei. Optimal sailing strategies, statistics and operations research program, 1996. University of Princeton, http://www.sor.princeton.edu/˜rvdb/sail/sail.html.

[Weissman *et al.*, 2003] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M.J. Weinberger. Inequalities for the l1 deviation of the empirical distribution. Technical Report HPL-2003-97R1, Hewlett-Packard Labs, 2003.

[Younes and Littman, 2004] H. L. S. Younes and M. L. Littman. Ppddl1.0: An extension to pddl for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, October 2004.

## A Complete Proof of Theorem 1

*Proof.* This proof follows the convergence proof of Trial-Based RTDP presented in [Barto *et al.*, 1995], the differences being marked by previous text stroken through and new text in bold. We only consider the special case in which only the cost of the current state is backed up at each time interval, i.e.,[6] $B_t = \{s_t\}$ and $k_t = t$, for $t = 0, 1, \ldots$. As in [Barto *et al.*, 1995], we can observe that the proof does not change when each $B_t$ is allowed to be an arbitrary set containing $s_t$.

Let $G$ denote the goal set and let $s_t$, $a_t$, $m_t$ and $J_t$ respectively denote the state, action, local model and evaluation function at time step $t$ in an arbitrary infinite sequence of states, actions, local models and evaluation functions generated by robust Trial-Based RTDP starting from an arbitrary start state.

[6] We follow Barto *et al.*'s notations where $B_t$ is the set of states updated at time step $t$ and $k_t$ is the total number os asynchronous dynamic programming stages completed up to time $t$.

First observe that the evaluation functions remain non-overestimating, i.e., at any time $t$, $J_t(i) \leq J^*(i)$ for all states $i$. This is true by induction because $J_{t+1}(i) = J_t(i)$ for all $i \neq s_t$ and if $J_t(j) < J^*(j)$ for all $j \in S$, then for all $t$

$$\begin{aligned} J_{t+1}(s_t) &= \min_{a \in A(i)} \max_{m_i^a \in \mathcal{M}_i^a} \left[ c_{s_t}(a) + \sum_{j \in S} p_{s_t j}^{m_i^a}(a) J_t(j) \right] \\ &\leq \min_{a \in A(i)} \max_{m_i^a \in \mathcal{M}_i^a} \left[ c_{s_t}(a) + \sum_{j \in S} p_{s_t j}^{m_i^a}(a) J^*(j) \right] \\ &= J^*(s_t), \end{aligned}$$

where the last equality restates the Bellman Optimality Equation. **It is also important to note that $J_t$ is here again increasing, what justifies the greediness of (robust) RTDP.**

Let $I \subseteq S$ be the set of all states that appear infinitely often in this arbitrary sequence; $I$ must be nonempty because the state set is finite. Let $Ad(i) \subset A(i)$ be the set of admissible actions for state $i$ that have zero probability of causing a transition to a state not in $I$, i.e., $Ad(i)$ is the set of all actions $a \in A(i)$ such that $p_{ij}(a) = 0$ for all $j \in (S - I)$, **considering the model $m_d$ used to pick the next state**. Because states in $S - I$ appear a finite number of times, there is a finite time $T_0$ after which all states visited are in $I$. Then with probability one any action chosen an infinite number of times for any state $i$ that occurs after $T_0$ must be in $Ad(i)$ (or else with probability one a transition out of $I$ would occur), and so with probability one there must exist a time $T_1 > T_0$ such that for all $t > T_1$, we not only have that $s_t \in I$ but also that $a_t \in Ad(s_t)$.

We know that at each time step $t$, RTDP backs up the cost of $s_t$ because $s_t \in B_t$. We can write the backup operation as follows:

$$\begin{aligned} J_{t+1}(s_t) = \min_{a \in A(i)} \max_{m_i^a \in \mathcal{M}_i^a} \Bigg[ & c_{s_t}(a_t) + \sum_{j \in I} p_{s_t j}^{m_i^a}(a_t) J_t(j) \\ & + \sum_{j \in (S-I)} p_{s_t j}^{m_i^a}(a_t) J_t(j) \Bigg]. \end{aligned} \tag{4}$$

But for all $t > T_1$, we know that $s_t \in I$ and that $p_{s_t i}^{m_i^a}(a_t) = 0$ for all $j \in S - I$ because $a_t \in Ad(s_t)$. Thus, for $t > T_1$ the right-most summation in Eq. (4) is zero ($\forall m \in \mathcal{M}$, $p_{s_t i}^m(a_t) = 0$). This means that the costs of the states in $S - I$ have no influence on the operation of RTDP after $T_1$. Thus, after $T_1$, **robust** RTDP performs asynchronous DP on a ~~Markovian decision problem~~ **sequential stochastic shortest path game** with state set $I$. **From now on, we view our problem as a simultaneous stochastic shortest path game. As pointed out in [Bertsekas and Tsitsiklis, 1996], Section 7.2, a sequential game is just a particular case of simultaneous game.**

If no goal states are contained in $I$, then all the immediate costs in this ~~Markovian decision problem~~ **stochastic shortest path game** are positive. Because there is no discounting, it can be shown that asynchronous DP must cause the costs of the states in $I$ to grow without bound. But this contradicts the fact that the cost of a state can never overestimate its optimal cost, which must be finite due to the existence of a proper policy. Thus $I$ contains a goal state with probability one.

After $T_1$, therefore, **robust** Trial-Based RTDP performs asynchronous DP on a stochastic shortest path ~~problem~~ **game** with state set $I$ that satisfies the conditions of the

convergence theorem for asynchronous DP applied to undiscounted stochastic shortest path ~~problems~~ **games** ~~(Bertsekas and Tsitsiklis [12, Proposition 3.3, p. 318])~~**([Patek and Bertsekas, 1999], Proposition 4.6)**. Consequently, **robust** Trial-Based RTDP converges to the optimal evaluation function of this stochastic shortest path ~~problem~~ **game under the optimal pessimistic models**. We also know that the optimal evaluation function for this problem is identical to the optimal evaluation function for the original problem restricted to the states in $I$ because the costs of the states in $S - I$ have no influence on the costs of states in $I$ after time $T_1$.

Furthermore, with probability one $I$ contains the set of all states reachable from any start state via any optimal policy, **and with any possible model**. Clearly, $I$ contains all the start states because each start state begins an infinite number of trials. **Robust** Trial-Based RTDP always executes a greedy action with respect to the current evaluation function and breaks ties in such a way that it continues to execute all the greedy actions. Because we know that the number of policies is finite and that Trial-Based RTDP converges to the optimal evaluation function restricted to $I$, there is a time after which it continues to select all the actions that are greedy with respect to the optimal evaluation function, i.e., all the optimal actions. Thus with probability one, $I$ contains all the states reachable from any start state via any optimal policy **under any possible model**, and there is a time after which a controller using RTDP will only execute optimal actions.

Finally, with trivial revision the above argument holds if RTDP backs up the costs of states other than the current state at each time step, i.e., if each $B_t$ is an arbitrary subset of S. □

# B Obtaining an Interval-Based Model

This appendix presents practical ways of obtaining an uncertain interval-based model:

- a method for computing confidence intervals from available statistics, and
- a process to turn a temporal planning problem with uncertain task outcomes into a USSP.

## B.1 Computing Confidence Intervals from Statistics

Uncertain models can be learnt if real or simulated state transition data is available. The information we have about a given transition $(s, a) \rightarrow s'$ comes from $N$ uses of action $a$ in state $s$, $n$ of them which led to state $s'$. So, our maximum-likelihood estimate of the true probability $T(s, a, s')$ is $p = n/N$. With a simple uniform prior over $[0, 1]$ (all transition probabilities being equiprobable from our point of view), the central limit theorem states that the sample distribution approaches a normal distribution with a decreasing variance as the sample size $N$ increases. This fact can be used to compute a confidence interval around the estimated probability $p$:

$$I = [p - z.\sigma_p,\ p + z.\sigma_p],$$

where $z$ depends on the desired level of confidence, and the standard error of the proportion is estimated by

$$\sigma_p = \sqrt{\frac{p(1-p)}{N}}.$$

Calculating $z$ requires us to decide of a level of confidence. This is done by choosing the probability $\alpha$ that the true value $T(s, a, s')$ is out of the confidence interval or, conversely that $Pr(T(s, a, s') \in I) = 1 - \alpha$. Using $\alpha = 0$ would make any estimation $p \in (0, 1)$ correspond to probability intervals $[0, 1]$, since $Pr(T(s, a, s') \in I) = 1$ is achieved only when all possible models are considered. With a confidence level $\alpha$, $z$ is the unique solution of

$$erf(\frac{z.\sigma_p}{\sqrt{2}}) = 1 - \alpha,$$

where $erf$ is the error function [Abramowitz and Stegun, 1972][7]

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_{[0,x]} e^{-t^2} dt,$$

yielding

$$z \quad = \quad \frac{\sqrt{2}}{\sigma_p} . erf^{-1}(1-\alpha). \tag{5}$$

This process requires the extra constraint that intervals must stay in $[0, 1]$. If the estimation $p$ is 0 or 1, the estimated standard error $\sigma_p$ is zero and the interval is reduced to a point (see Fig. 12). Whether we accept this or impose a minimum interval is domain dependent.

The central limit theorem provides us with a complete probability distribution over possible models.[8] But our approach to taking uncertainty ignores the probability of each feasible model.

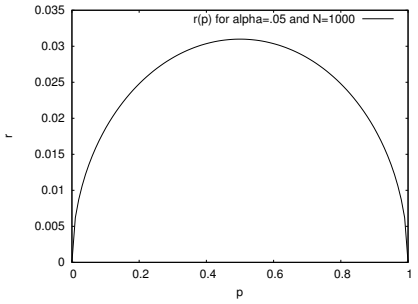



Figure 12: Radius of a confidence interval depending on the estimated proportion

## B.2 Uncertain Model for Temporal Planning

We consider a temporal planning setting where each action can start several tasks. The resulting state depends on the probabilistic outcome of running tasks, on their durations and on the resources they consume. Here we assume that the task outcome probabilities are:

- assigned by a human expert who also gives an estimate of how certain they are, in the form of intervals around the probability values, or
- learned through experience (using the process described in previous section).

States are generated from task descriptions the first time they are visited so we never generate the entire state space. The process starts from the current state. A list of valid actions for this state is generated, corresponding to the power set of tasks with satisfied pre-conditions. Future states are generated by applying the effects of each action and adding future — possibly probabilistic — effects of the action to an event queue for the new *intermediate* state. To generate the successor states for a particular action, we take the intermediate state and process events in the queue until an event allows new tasks to be started, i.e., a state in which we can choose a new action. Processing probabilistic events splits the search for successors, as shown in Figure 13. The root is the current state $s$ after applying the *immediate* effects of actions. Events $e_1$, $e_2$ are processed and each arc going out of an event corresponds to a possible outcome, labeled with its probability interval, and leaves are reachable states from $s$ under a particular action.

[7] `http://mathworld.wolfram.com/Erf.html`

[8] A better approach should be that of [Weissman *et al.*, 2003], used for example by [Strehl and Littman, 2004].

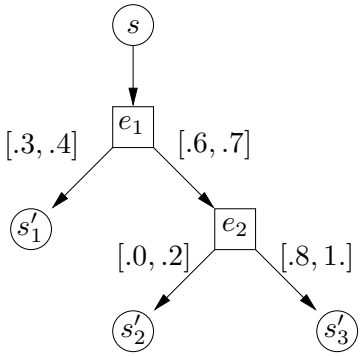



Figure 13: Computing the states reachable from current state.

In Figure 13 the probability of event $e_1$'s left outcome (leading to $s'_1$) lies in the interval $I_{e_1}^{\text{left}} = [.3, .4]$, and the probability of reaching event node $e_2$ is $Pr(e_2) \in I_{e_1}^{\text{right}} = [.6, .7]$. This tree is built by considering the events one after another. States appearing several times in the tree are merged. The uncertain probabilities known for the outcomes of an event are used to compute the probability intervals of next event or reachable state in the tree. The probability of reaching $s'_2$ is for example: $Pr(s'_2) \in I_{e_1}^{\text{right}}.I_{e_2}^{\text{left}} = [.6 * .0, .2 * .7]$.

Robust RTDP requires that the model parameters are independent from one state-action pair to another. However, in the temporal planning domain, an action starts multiple tasks and several actions may start the same task. States represents the status of all tasks. Thus selecting the worst model, by choosing a fixed point in a $T(s, a, s')$ transition probability interval, implies that we have selected a fixed probability of failure for all tasks involved in that transition, and that fixed probability should be applied to all other transition intervals $T(s'', a', s''')$ involving the same tasks. Ignoring these dependencies has the effect of allowing the disturber to select a model that is not in the original set $\mathcal{M}$, and can consequently be even more pessimistic than is strictly necessary.

# C Being Optimistic After Being Pessimistic (work in progress)

The main idea, taken from [Hosaka *et al.*, 2001] is that being robust should not mean only looking for a best policy under worst models. Among these policies, one can extract one that behaves better than others under the best model. In other words, one can be optimistic after having been pessimistic.

In the case of the uncertain SSP presented on Fig. 14, optimal actions under worst models are $a_0$ and $a_1$ (because of the propability $0.9$ of coming back to state $s_0$. Yet, $a_1$ should be prefered as it leads to a better outcome under an optimistic model.

## C.1 How to Compute the Various Long-Term Cost Functions

The set of relevant states we care about is that of relevant states under worst models (and corresponding best policies).

We define three long-term cost functions:

- $J_{pess}$, the usual one in robust algorithms: the long-term cost function under worst models (pessimistic long-term cost),
- $J_{opt}$, the opposite notion: the long-term cost function under best models (optimistic long-term cost), and
- $J_{oap}$, the function we are interested in: the long-term cost function under best models when allowing only decisions which are optimal with respect to $J_{pess}$.

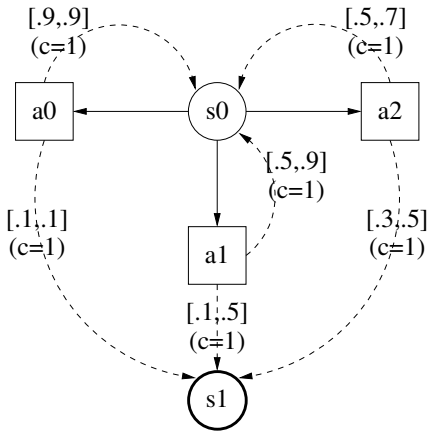



Figure 14: An uncertain SSP with two deterministic robust policies, one being better when adopting an optimistic point of view.

One can note that, for all $s \in S$: $J_{opt}(s) \leq J_{oap}(s) \leq J_{pess}(s)$. It follows, considering the greedy nature of RTDP, that $J_{opt}$ can be used as an initialization of $J_{oap}$ or $J_{pess}$. But $J_{pess}$ cannot use $J_{oap}$ since the later is computed thanks to the former ($V_{oap}$ requires restraining the set of actions to the "robust" ones).

Yet, computing $J_{opt}$ could help saving computation time only in the first of the two possible algorithms we propose:

- 1-Use RTDP (with optimistic models) to compute $J_{opt}$ which will serve as an initialization, 2- use RTDP (with pessimistic models) to compute $J_{pess}$, and 3- use RTDP (with optimistic models) with actions restricted to the ones optimal under $J_{pess}$ to compute $J_{oap}$.
- Use RTDP (with pessimistic models) to compute $J_{pess}$ and simultaneously compute $J_{oap}$. $J_{oap}$ will converge only after $J_{pess}$, as it needs that optimal decisions are stable.

We have decided to follow the second approach, as it involves only minor modifications. The fact that $J_{oap}$ may not be non-decreasing is not a problem as this property is required only for exploration purposes.

## C.2 Modifications of LRTDP

As we are using in practice LRTDP [Bonet and Geffner, 2003], we have to take particular care of a view implementation details. Indeed, one can notice that LRTDP does not respect one requirement of RTDP [Barto *et al.*, 1995]: the fact that all equally greedy actions should be equally chosen. This could be forgotten if being pessimistic only, but it is essential in the current process. This problem can be corrected through minor changes:

- `greedyAction()` has to pick an action randomly among all best actions (not always the same),
- `checkSolved()` has to consider all reachables states of state $s$, considering all best actions (not one best action only), and
- `checkSolved()` has to check simultaneously for the residual related to $J_{pess}$ and the one related to $J_{oap}$ before deciding if a state is `solved`.