Planning Under Uncertainty Using Reduced Models: Revisiting Determinization

Luis Pineda and **Shlomo Zilberstein**
School of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
{lpineda, shlomo}@cs.umass.edu

Abstract

We introduce a family of MDP reduced models characterized by two parameters: the maximum number of *primary outcomes* per action that are fully accounted for and the maximum number of occurrences of the remaining *exceptional outcomes* that are planned for in advance. Reduced models can be solved much faster using heuristic search algorithms such as LAO*, benefiting from the dramatic reduction in the number of reachable states. A commonly used determinization approach is a special case of this family of reductions, with one primary outcome per action and zero exceptional outcomes per plan. We present a framework to compute the benefits of planning with reduced models, relying on online planning when the number of exceptional outcomes exceeds the bound. Using this framework, we compare the performance of various reduced models and consider the challenge of generating good ones automatically. We show that each one of the dimensions—allowing more than one primary outcome or planning for some limited number of exceptions—could improve performance relative to standard determinization. The results place recent work on determinization in a broader context and lay the foundation for efficient and systematic exploration of the space of MDP model reductions.

1 Introduction

Determinization-based algorithms for solving MDPs have gained popularity in recent years, motivated by the surprising success of the FF-Replan solver (Yoon *et al.* 2007) in the IPPC-04 planning competition (Younes *et al.* 2005). The main idea behind these algorithms is to generate a deterministic version of the underlying MDP and solve it using a classical planner such as FF (Hoffmann and Nebel 2001), resulting in a *partial plan* for the original problem. When confronted by an unexpected state during plan execution, the planning process is repeated using the current state as the initial state. The main advantage of determinization-based algorithms is their ability to quickly generate partial plans, particularly in intractable probabilistic domains.

Determinization-based approaches differ in the way they add robustness to the partial plans. FF-Replan (Yoon *et al.* 2007)—the most basic of these approaches—entirely ignores future deviations from the plan. FF-Hindsight (Yoon *et al.* 2008; 2010) uses a *hindsight optimization* approach to approximate the value function of the original MDP by sampling multiple deterministic futures and solving them using the FF planner. RFF (Teichteil-Königsbuch *et al.* 2010) generates a plan for an envelope of states such that the probability of reaching a state outside the envelope is below some threshold. To increase the envelope, RFF chooses states not considered in the current plan and, using an approach similar to FF-Replan, computes an action for each one of these states. HMDPP (Keyder and Geffner 2008) introduces the so-called *self-loop determinization* in order to trick the deterministic planner into generating plans with a low probability of deviations, using a pattern database heuristic to avoid dead-end states.

Despite their success, determinization-based algorithms have some drawbacks due to the fact that they consider each of the outcomes of an action in isolation. This leads to an overly optimistic view of the domain and can result in plans that are arbitrarily worse than an optimal plan. Furthermore, even when optimal plans could be obtained using isolated outcomes, it is not always clear, nor intuitive, which outcomes should be included in the determinization, as we illustrate later. Nevertheless, solving reduced versions of an MDP in an online fashion can be very useful in practice, especially for large domains. Thus, we want to take advantage of this general idea while addressing its inherent drawbacks.

To this end, we introduce a more general paradigm in which the single-outcome variant of FF-Replan (FF_s) (Yoon *et al.* 2007) is just one extreme point on a *spectrum of MDP reductions* that differ from each other along two dimensions: (1) the number of outcomes per state-action pair that are fully accounted for in the reduced model, and (2) the number of occurrences of the remaining outcomes that are planned for in advance. For example, FF_s considers one outcome per state-action pair and plans for zero occurrences of the remaining outcomes in advance (i.e., it completely ignores them). In contrast, we propose an approach that fully considers a subset of the original outcomes—possibly with cardinality greater than one—assuming that the remaining outcomes will occur at most some k times during plan execution. Similar treatments of exceptional outcomes have been explored in fault-tolerant planning (Jensen *et al.* 2004; Domshlak 2013; Pineda *et al.* 2013).

Other planning approaches have been developed for reduced or incomplete models. SSiPP (Trevizan and Veloso 2012), for example, relies on pruning all states beyond a certain reachable envelope. Other works have examined explicitly the question of improving plan robustness to known incompleteness of the domain model (Nguyen *et al.* 2013). In this work we explore an entire spectrum of relaxations using an evaluation metric that minimizes the *comprehensive* cost of reaching the goal, including replanning cost.

We begin with a definition of MDP reduced models, and show how to efficiently solve reduced models using the heuristic search algorithm LAO* (Hansen and Zilberstein 2001). We then introduce a *continual planning* paradigm that handles situations where the number of exceptions exceeds the bound during plan execution, and study the theoretical properties of the resulting plans. In particular, we present a closed-form framework to evaluate the benefits of a particular reduced model. We then show how to use this evaluation technique to choose a good determinization, and more generally, how to find good reduced models. In the experimental results section, we show that both model reduction dimensions contribute significantly to improving the performance of a baseline determinization approach.

2 Problem Formulation

We focus on *stochastic shortest-path* MDPs, where the goal is to minimize the expected cost of reaching a set of goal states. These problems can be defined as tuples $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, s_0, \mathcal{G} \rangle$, where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $\mathcal{T}(s'|s,a)$ is a stationary transition function specifying the probability of reaching state s' when action a is executed in state s; $\mathcal{C}(s,a)$ is the positive cost of executing action a in state s; $s_0 \in \mathcal{S}$ is a given start state; and $\mathcal{G} \subset \mathcal{S}$ is a set of absorbing goal states.

Starting in state s_0, the objective is to reach one of the goal states while minimizing the expected cumulative cost of the actions taken. If dead-ends—states from which a goal cannot be reached—are present in the domain, we circumvent this by allowing the agent to perform a "give up" action in all states at some very high fixed cost (Kolobov *et al.* 2012). Hence, we do not use discounting as there is always a *proper policy* that reaches a goal state with probability 1.

A solution is a policy π, represented as a mapping from states to actions: $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The well-known Bellman equation defines a value function over states, $V^*(s)$, from which the optimal policy π^* can be extracted:

$$V^*(s) = \min_a [\mathcal{C}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) V^*(s')] \qquad (1)$$

Solving MDPs optimally is often intractable (Littman 1997), which has led to the development of many approximate algorithms, often based on value function approximation methods. Here we explore a new paradigm to handle the complexity of MDPs by defining a space of MDP reductions that generalizes the idea of single-outcome determinization.

Reducing an MDP We propose a family of MDP reduced models that are characterized by two key parameters: the number of outcomes per action that are fully accounted for, and the maximum number of occurrences of the remaining outcomes that are planned for in advance. We refer to the first set of outcomes as *primary outcomes* and to the remaining outcomes as *exceptional outcomes*.

We consider factored representations of MDPs—such as PPDDL (Younes *et al.* 2005)—in which actions are represented as probabilistic operators of the form:

$$a = \langle prec, cost, [p_1 : e_1, ..., p_m : e_m] \rangle,$$

where $prec$ is a set of conditions necessary for the action to be executed, $cost$ is the cost of the action (assumed to be the same in all states), and for each $i \in [1; m]$, p_i is the probability of outcome (or effect) e_i occurring when the action is executed. The transition function $\mathcal{T}$ can be recovered from this representation through a successor function τ, so that $s' = \tau(s, e_i)$ and $\mathcal{T}(s'|s,a) = p_i$.

For any action a, let $\mathcal{P}_a \subseteq \{e_1, ..., e_m\}$ be the set of primary outcomes. Given sets $\mathcal{P}_a$ for each action $a \in \mathcal{A}$, we define a reduced version of the MDP that accounts for a bounded number of occurrences of exceptional outcomes. The state space of the reduced MDP is $(s, j) \in \mathcal{S} \times \{0, 1, ..., k\}$, where j is a bounded exception counter. The initial state of the reduced MDP is $(s_0, 0)$, indicating that no exceptions have occurred. The transition function $\mathcal{T}'$ of the augmented MDP is defined as follows.

When $j = k$ (the exception bound), we assume that no more exceptions can occur, so the new transition function is:

$$\begin{cases} \forall s, a \;\; \mathcal{T}'((s',k)|(s,k),a) = p'_i & \text{if } e_i \in \mathcal{P}_a \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where we use the shorthand $s' = \tau(s, e_i)$ and p'_i satisfies:

$$\forall i : e_i \in \mathcal{P}_a \;\; p'_i > 0 \quad \text{and} \quad \sum_{i : e_i \in \mathcal{P}_a} p'_i = 1 \qquad (3)$$

In this work we simply normalize the probabilities of the primary outcomes so that they sum up to one. More complex ways to redistribute the probabilities of exceptional outcomes are mentioned in the conclusion.

When $j < k$, exceptions are fully accounted for, thus we need to update the exception counter appropriately. The transition function in this case becomes:

$$\begin{cases} \forall s, a \;\; \mathcal{T}'((s',j)|(s,j),a) = p_i & \text{if } e_i \in \mathcal{P}_a \\ \forall s, a \;\; \mathcal{T}'((s',j+1)|(s,j),a) = p_i & \text{otherwise} \end{cases} \qquad (4)$$

Finally, the costs and goal states of the reduced MDP are the same as the original ones, ignoring the exception counter.

Note that while the complete state space of the reduced MDP is actually larger than that of the original problem, the benefit of the reduction is that, for well-chosen values of k and sets $\mathcal{P}_a$, the set of *reachable states* can become much smaller—by orders of magnitude in our experiments. This is desirable because the runtime of heuristic search algorithms for solving MDPs such as LAO* (Hansen and Zilberstein 1998; 2001) and LRTDP (Bonet and Geffner 2003) depends heavily on the size of the reachable state space. Furthermore, by changing k and the maximum size of the sets $\mathcal{P}_a$, we can adjust the amount of uncertainty we are willing to ignore in order to have a smaller reduced problem. The following definition makes this idea more precise.

Definition 1 **($\mathcal{M}_l^k$-reduction of an MDP).** *An $\mathcal{M}_l^k$-reduction of an MDP is an augmented MDP with the transition function defined by Eqs. 2 and 4, where $j \in \{0, 1, ..., k\}$ and $\forall a \; |\mathcal{P}_a| \leq l$.*

For example, the single-outcome determinization used in the original FF-Replan (Yoon *et al.* 2007) is an instance of an $\mathcal{M}_1^0$-reduction where each set $\mathcal{P}_a$ contains the single most likely outcome of the corresponding action a.

Note that for any given values of k and l there might be more than one possible $\mathcal{M}_l^k$-reduction. We introduce the notation $M \in \mathcal{M}_l^k$ to indicate that M is some instance of an $\mathcal{M}_l^k$-reduction; different instances are characterized by two choices. One is the specific outcomes that are labeled as primary. The other is how to distribute the probability of the exceptional outcomes among the primary ones when $j = k$ (i.e., the choice of p'_i in Eq. 2).

The concept of $\mathcal{M}_l^k$-reductions raises a number of interesting questions about their potential benefits in planning:

1. How should we assess the comprehensive value of an $\mathcal{M}_l^k$-reduction? Can this be done analytically?
2. Considering the space of $\mathcal{M}_l^k$-reductions, is determinization or $\mathcal{M}_1^0$-reduction always preferable?
3. In the space of possible determinizations, can the best ones be identified using a simple heuristic (e.g., choosing the most likely outcome)? Or do we need more sophisticated value-based methods for that purpose?
4. How can we explore efficiently the space of $\mathcal{M}_l^k$-reductions? How can we find good ones or the best one?
5. Given a PPDDL description of an MDP and some k and l, can one automatically generate a PPDDL description of an $\mathcal{M}_l^k$-reduction? How can we explore this space and evaluate different reductions using a compilation scheme and *standard* MDP solvers?

The rest of the paper explores these questions, showing that an $\mathcal{M}_1^0$-reduction is not always desirable. Furthermore, even when determinization works well, a value-based approach is needed to choose the most appropriate primary outcome per action. In order to answer the above questions, however, we need an exact method to evaluate the benefits of a particular $\mathcal{M}_l^k$-reduction. In the next section we show how, given k and sets $\mathcal{P}_a$ for all actions in the original MDP, we can evaluate the expected cost of solving the original problem using plans derived by solving the reduced problem.

3 Evaluating Reduced Models

Continual planning paradigm A plan generated using a reduction $M \in \mathcal{M}_l^k$ could possibly be incomplete with respect to the original MDP, because more than k exceptions could occur during plan execution, leading to an outcome state that may not be included in the plan. In order to evaluate plans obtained using reduced models, we need a strategy that guarantees goal reachability during plan execution.

To this end, we employ an online *continual planning* strategy (Algorithm 1) that generates a new plan for M whenever a state is reached with an exception counter $j = k$. Note that at this point of execution there will still be an action ready in the current plan, so we can compute the new plan while simultaneously executing an action from the existing plan. As long as the new plan is ready when the action finishes executing, plan execution will resume without delay.

Algorithm 1: Continual Planning Paradigm

```
input: MDP problem $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, s_0, \mathcal{G} \rangle$, $k$, $l$ and $\mathcal{P}_a$
1 $(s, j) \leftarrow (s_0, 0)$;
2 $\pi \leftarrow$ Find-Reduced-MDP-Plan$((s_0, 0), k, l, \mathcal{P}_a)$;
  while $s \notin \mathcal{G}$ do
      if $j \neq k$ then
3         $(s, j) \leftarrow$ ExecutePlan$(s, \pi(s, j))$;
      else
4         Create new state $\hat{s}$ with one zero-cost action $\hat{a}$
          s.t. $\forall s' \in \mathcal{S}: Pr((s', 0)|\hat{s}, \hat{a}) = \mathcal{T}(s'|s, \pi(s, j))$;
          do in parallel
5             $(s, j) \leftarrow$ ExecutePlan$(s, \pi(s, j))$;
6             $\pi' \leftarrow$ Find-Reduced-MDP-Plan$(\hat{s}, k, l, \mathcal{P}_a)$;
7         $\pi \leftarrow \pi'$; $(s, j) \leftarrow (s, 0)$;
```

There is one complication in this online planning process. Since the new plan will be activated from a start state that is not yet known (when the planning process starts), all the possible start states need to be taken into account, including those reached as result of another exception. Therefore, we create a new dummy start state in Line 4 that leads via a single zero-cost action to all the start states we can possibly encounter when the execution of the current action ends.

For the sake of clarity of the algorithm and its analysis, we describe a straightforward implementation where the execution time of one action is sufficient to generate a plan for the reduced model. When planning requires more time, it will delay the execution of the new plan. Several techniques to avoid such delays are discussed in the conclusions.

Evaluating the continual planning paradigm Our continual planning paradigm is amenable to a precise closed-form evaluation. Let π_k be a *universal* optimal plan computed using value iteration over the entire state space of a reduction $M \in \mathcal{M}_l^k$ of the original problem. At runtime, we always execute π_k using a partial plan and if we reach a state (s, k), we execute $\pi_k(s, k)$, but move to state $(s', 0)$ and start executing $\pi_k(s', 0)$ using the newly computed partial plan (s' is the primary or exceptional outcome of the previous action). More formally, consider the Markov chain defined over states of the form (s, j), with initial state $(s_0, 0)$ and the following transition function, for any $s \in S, 0 \leq j \leq k$:

$$\begin{array}{ll} Pr((s', j')|(s, j)) = \mathcal{T}'((s', j')|(s, j), \pi_k(s, j)) & \text{if } j < k \\ Pr((s', 0)|(s, k)) = \mathcal{T}(s'|s, \pi_k(s, j)) & \text{otherwise} \end{array}$$

The transition probability from (s, k) to $(s', 0)$ indicates the transition to a new plan. Let V_{cp}^M denote the value function defined over this Markov chain with respect to reduction M. Then we have:

Proposition 1. *$V_{cp}^M(s_0, 0)$ provides the expected value of the continual planning paradigm for reduction M when applied to the original problem domain.*

Continual planning paradigms often involve heuristic decisions about the interleaving of planning and execution, making it necessary to evaluate them empirically. In contrast, Proposition 1 enables us to evaluate our paradigm with different reduced models based on their precise value.

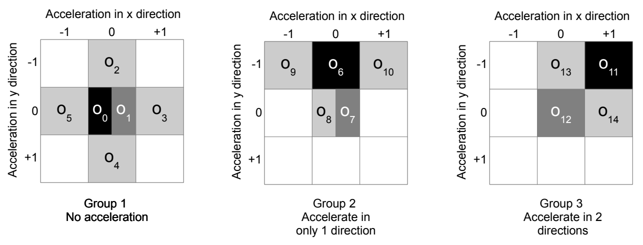
Acceleration in x direction
-1
0
+1
Acceleration in y direction
-1
0
+1
o2
o5
o0
o1
o3
o4
Group 1
No acceleration
o9
o6
o10
o8
o7
Group 2
Accelerate in only 1 direction
o13
o11
o12
o14
Group 3
Accelerate in 2 directions

Figure 1: Action groups in the racetrack domain: dark squares represent the intended action, gray squares represent the acceleration outcome associated with slipping, and the light gray squares represent the remaining outcomes.

4 Determinization as a Reduced Model

The defining property of all determinization-based algorithms is the use of fully deterministic models in planning, entirely ignoring what we call exceptional outcomes. In fact, even the widely used *all-outcome* determinization, which is not an $\mathcal{M}_l^k$ reduction, treats each probabilistic outcome as a fully deterministic one, completely ignoring the relationship between outcomes of the same action. Such oversimplifications can lead planners to optimistically choose an action with a small probability of success, just because one of its outcomes appears very promising. Hence, we argue that an $\mathcal{M}_l^0$-reduction with $l > 1$ could be significantly better for some domains, particularly in comparison to the best $\mathcal{M}_1^0$ reduction. We use $\ll$ to denote that in the following claim:

Claim 1. *There are problems $\mathcal{M}$ for which $\exists M' \in \mathcal{M}_l^0$ s.t. $\forall M \in \mathcal{M}_1^0 : V_{cp}^{M'}(s_0, 0) \ll V_{cp}^{M}(s_0, 0)$, for some $l > 1$.*

We illustrate the validity of this claim through an example, using a modified version of the racetrack domain (Sutton and Barto 1998)—a well-known Reinforcement Learning benchmark. The original problem involves a simulation of a race car on a discrete track of some length and shape, where a starting line has been drawn on one end and a finish line on the opposite end of the track. The state of the car is determined by its location and its two-dimensional velocity. The car can change its speed in each of the two dimensions by at most 1 unit, resulting in a total of nine possible actions. After applying an action there is a probability p_s that the resulting acceleration is zero, simulating failed attempts to accelerate/decelerate because of unpredictably slipping on the track. Additionally, there is a probability p_e that the resulting acceleration is off by one dimension with respect to the intended acceleration. The goal is to go from the start line to the finish line in as few moves as possible. For a complete description of the original problem and its modified form refer to (Sutton and Barto 1998) and (Pineda *et al.* 2013), respectively.

To decrease the number of reductions to consider for this problem, instead of treating the outcomes of all 9 actions separately, we grouped symmetrical actions and applied the same reduction to all actions in the same group. The racetrack domain has three groups of symmetric actions: 4 actions that accelerate/decelerate in both directions, 4 actions that accelerate/decelerate in only one direction, and 1 action that keeps the current speed. Figure 1 illustrates these groups

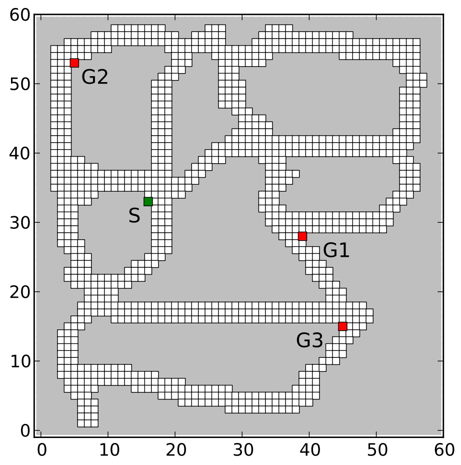

Figure 2: An instance of the racetrack domain.

of actions and their possible outcomes; for each group, a decomposition is specified by the set of outcomes, relative to the intended outcome (shown in darker color), that are labeled as primary.

In our experiments we used the racetrack configuration shown in Figure 2, which results in 34,991 states, and compared the following two reductions, M_1 and M_2:

$$M_1 := \min_{M \in \mathcal{M}_1^0} V_{cp}^{M}(s_0, 0) \quad \text{and} \quad M_2 := \min_{M \in \mathcal{M}_2^0} V_{cp}^{M}(s_0, 0)$$

That is, we compared the best possible $\mathcal{M}_1^0$-reduction (determinization) of this problem, with its best possible $\mathcal{M}_2^0$-reduction. Table 1 shows the increase in cost of these reductions with respect to the lower bound (optimal expected cost obtained by solving the full model offline). In all three cases, the use of determinization resulted in a 14% or higher increase in cost, while the maximum cost increase for the best $\mathcal{M}_2^0$-reduction was less than 7%.

Incidentally, using an all-outcome determinization could result in particularly poor performance in this domain. For example, the no-acceleration action includes low probability outcomes that accelerate in any one direction. A planner based on the all-outcome determinization could potentially choose that action with the intention of accelerating in some direction, which is clearly undesirable.

Choosing the right determinization Determinization does work well for some problems. That is, the cost of using continual planning with the *best* $\mathcal{M}_1^0$-reduction may be close to the optimal cost V^*. However, the choice of primary outcomes by inspecting the domain description may still

	$V_{cp}^{M_1}(s_0, 0)$	$V_{cp}^{M_2}(s_0, 0)$
G1	14.41%	0.83%
G2	28.74%	6.78%
G3	20.10%	3.75%

Table 1: Comparison of the best determinization (M_1) and the best $\mathcal{M}_2^0$-reduction (M_2) for the racetrack problem.

Instance # / Primary outcome	**P01**	**P02**	**P03**	**P04**	**P05**	...	**P10**
(not (not-flattire))	100	100	100	100	100	...	100
(not-flattire)	60	15	4	2	0	...	0

Table 2: Comparison of two different $\mathcal{M}_1^0$-reductions for the triangle-tireworld problem.

present a non-trivial challenge. In particular, the commonly used most-likely-outcome heuristic may not work well.

To illustrate this issue we experimented with different determinizations of the triangle-tireworld domain (Little and Thiebaux 2007). This problem involves a car traveling between locations on a graph shaped like a triangle. Every time it moves there is a certain probability of getting a flat tire when the car reaches the next location, but only some locations include a spare tire that can be used to repair the car. Since the car cannot change its location with a flat tire, the triangle-tireworld domain has dead-ends. As indicated earlier, we circumvent this by allowing the agent to perform a "give up" action in all states at a fixed cost of 500.

This domain has two possible determinizations, depending on whether getting a flat tire is considered an exception or a primary outcome. Table 2 shows the results (number of trials reaching the goal) of evaluating the two determinizations on 10 instances of this domain. The performance for the omitted instances (P06, P07, P08, and P09) is the same as for P10. The best determinization is undoubtedly the one in which getting a flat tire is considered the primary outcome. The resulting plan enabled the car to reach the goal in 100% of the simulations, while the other determinization resulted in complete failure to reach the goal for (P05 ... P10).

As it turns out, the right determinization for this problem is not very intuitive, as one typically expects primary outcomes to correspond to the most likely outcomes of an action or to its intended outcome when it succeeds. (The most likely outcome is not having a flat tire with probability 60%.) This counterintuitive result might lead one to consider the use of conservative heuristics, labeling the worst-case outcome as primary. Although this would indeed work very well in the triangle-tireworld domain, this heuristic performs poorly in other domains such as the racetrack problem.

To sum up, some determinizations can indeed result in optimal performance, but there seems to be no all-purpose heuristic to choose the best one. This suggests that a more principled value-based approach is needed in order to find a good determinization or a good reduction in general.

5 Finding Good Reduced Models

In this section we propose a greedy approach to finding a model $M \in \mathcal{M}_l^k$ with a low cost $V_{cp}^M(s_0, 0)$ for some given k and l. The main premise of the approach is that problems in the given domain share some common structure, and that the relative performance of different $\mathcal{M}_l^k$-reductions generalizes across different problem instances. Although this is a strong assumption, experiments we report in Section 6 confirm that it works in practice.

Let $\mathcal{P}_a^M$ be the set of primary outcomes for action a on a reduction $M \in \mathcal{M}_l^k$ of MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, s_0, \mathcal{G} \rangle$. Given k and l, every reduction $M \in \mathcal{M}_l^k$ is uniquely determined by the set $\bigcup_{a \in \mathcal{A}} \mathcal{P}_a^M$. Therefore, for simplicity of presentation, in the rest of this section the notation $M \in \mathcal{M}_l^k$ refers both to a reduction M or to the set containing its primary outcomes; the right interpretation will be clear from the context. Using this notation, finding a good $\mathcal{M}_l^k$-reduction amounts to solving the following combinatorial optimization problem:

$$\begin{array}{ll} \max\limits_{M \subseteq \mathcal{E}} & -V_{cp}^M(s_0, 0), \quad \mathcal{E} \equiv \bigcup\limits_{a \in \mathcal{A}} outcomes(a) \\ \text{s.t.} & \forall a \in \mathcal{A},\ 1 \leq |\{e : e \in M \cap outcomes(a)\}| \leq l \end{array}$$

This optimization problem is particularly hard to solve due to two complications. First, it is possible that some reductions M introduce dead-ends even if the original MDP had none. This can happen, for example, if all the outcomes that can make progress towards the goal are outside the set of primary outcomes, and the only path towards the goal requires the occurrence of more than k of these outcomes. Second, as we show below, the maximized objective function is not *submodular* (Nemhauser *et al.* 1978), making it harder to develop a bounded approximation scheme.

Proposition 2. *The function* $f(M) = -V_{cp}^M(s_0, 0)$ *is not submodular.*

Proof. The proof is by counterexample, using the MDP shown in Figure 3. Consider two $\mathcal{M}_2^0$-reductions $M_1 = \{e_{A1}, e_{B1}, e_{B2}\}$ and $M_2 = \{e_{A1}, e_{B2}\}$. It is not hard to see that $f(M_1) = f(M_2) = -51$. Now consider adding outcome e_{A2} to both M_1 and M_2. Let $\rho_e(S) = f(S \cup \{e\}) - f(S)$. Then we have $\rho_{e_{A2}}(M_1) = 31$ and $\rho_{e_{A2}}(M_2) = 0$, implying $\rho_{e_{A2}}(M_2) < \rho_{e_{A2}}(M_1)$ and $M_2 \subset M_1$, which contradicts submodularity. $\square$

Similar counterexamples can be constructed for larger values of k. Intuitively, this happens because the benefit of adding a particular outcome to the reduction might not become evident unless some other outcomes had been previously added.

We propose a greedy approach that starts with M equal to the full probabilistic model, and iteratively removes from M an outcome e that minimizes $f(M - \{e\})$. We continue this process until: i) the constraint on the number of primary outcomes is satisfied, and ii) the decrease in f (increase in comprehensive cost) with respect to the value of the full model is larger than some threshold. Additionally, during this greedy process we discard any reduction that introduces dead-ends,

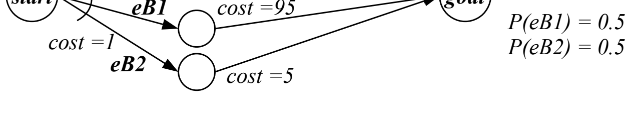

Figure 3: Example showing that $-V_{cp}^M(s_0, 0)$ is not submodular.

	CPU Time					
	VI	LNR	DET	M02	M11	M12
G1	9,205	4,852	5	57	19	396
G2	10,493	4,422	2	8	106	298
G3	8,664	3,521	5	58	100	445

	Total Cost					
	VI	LNR	DET	M02	M11	M12
G1	23.64	19.28	18.82	17.62	14.58	14.88
G2	23.92	17.85	18.76	15.97	13.78	13.82
G3	27.07	21.93	26.12	23.53	18.91	19.16

Table 3: Average planning time (in milliseconds) and total cost of planning and execution for the racetrack domain

thereby ensuring that the value $V_{cp}^{M}(s_0, 0)$ is always well-defined. Note that starting with $M = \emptyset$ and adding outcomes incrementally will inherently generate reduced models with dead-ends, for which the value is undefined.

Obviously, this greedy approach could be costly in terms of computation time, since every evaluation of the objective function involves computing an universal plan for the reduced model, and for $k > 0$ this is in fact more costly than solving the original problem using value iteration. In order to overcome this difficulty, we apply the greedy approach to relatively small problem instances so as to learn a good reduced model for the domain. We use a small enough problem so that computing universal plans for its reduced models can be done very quickly. The underlying assumption is that if a small problem instance captures the relevant structure of the domain, then a good reduction for this instance generalizes to larger problems. This is of course a strong assumption, but it seems to work well in practice and is used as an initial approach to this problem. We discuss some way to improve this approach in the conclusions.

6 Evaluating Planning with Reduced Models

In this section we show that $\mathcal{M}_l^k$-reductions with $k = 1$ can be used to quickly compute plans that produce near-optimal performance. To this end, we evaluate and compare the use of our continual planning approach with several reductions on the racetrack problem shown in Figure 2, and compare their performance with both value iteration and LAO* using no reduction (referred to as VI and LNR, respectively). In all the experiments we use a constant value of one second to measure the execution time of an action, which also bounds the re-planning time for the continual planning approaches.

We evaluate four continual planning approaches: DET uses an $\mathcal{M}_1^0$-reduction (single-outcome determinization), M02 uses an $\mathcal{M}_2^0$-reduction, M11 uses an $\mathcal{M}_1^1$-reduction and M12 uses an $\mathcal{M}_2^1$- reduction. Reductions with equal l (e.g., M02 and M12) have the same set of primary outcomes, generated automatically using the greedy approach described in Section 5 on a smaller track with 1,371 states, using $k = 0$.

Note that neither DET nor M02 will have actions available when exceptions occur, so the agent must idle while a new plan is computed. We simulate this situation by quickly computing a reactive plan to stop the car (and adding its cost to the total cost) before the next plan is computed. On the other hand, this is not an issue for M11 and M12, since re-planning can be done at the same time an action is being executed (as explained in Section 3).

Table 3 (top) shows the average CPU time spent on planning for the three different goals, computed based on 50 simulations for each planner. Planning time includes the cumulative time of all activations that delay plan execution (i.e., the initial planning time before the first action is taken plus any subsequent planning time when planning is not done during plan execution). Note that LNR doesn't reduce planning time significantly with respect to VI (less than 60% reduction). However, all the continual planning approaches reduce time by more than an order of magnitude with respect to LNR, with DET being the faster approach by far. Although using reduced models with $k = 1$ increases the planning time with respect to using $k = 0$, both are significantly faster than using no reduction at all (i.e., VI and LNR).

We quantify the trade-off between execution and planning time by accounting for the execution time of an action, using 1 second per action in this example. This is equivalent to associating a cost of 0.001 per millisecond of planning, allowing us to combine planning and execution costs and compute the total expected cost of all four approaches. These numbers are shown in Table 3 (bottom). For all three goals the total cost of using $\mathcal{M}_l^1$-reductions is smaller than that of LNR by more than 12% (22% for G1 and G2). Even though M02 and DET are the fastest approaches, they lose their advantage w.r.t. planning with $k = 1$, due to the approximate nature of the plans (ignoring exceptions altogether) and the overhead cost of stopping the car before re-planning.

Finally, we compare the total expected cost of the four approaches with a theoretical (loose) lower bound equal to the optimal cost obtained by solving the full model, without considering planning time. Figure 4 shows the relative increase in expected cost per method with respect to this bound. In all cases, M11 and M12 differ from the bound by 4% or less, while the rest of the approaches yield a much greater increase of (at least) 19%, 19%, 30% and 47% for M02, LNR, DET and VI, respectively. In general, the best approach is M11, since the faster planning times makeup for a small decrease in expected cost of the policy.

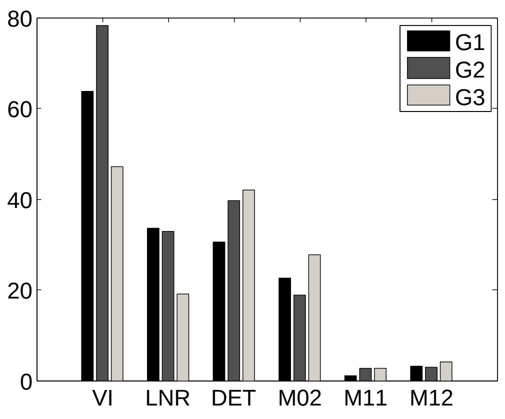

Figure 4: Relative increase in total cost with respect to the lower bound for the racetrack domain shown in Figure 2.

7 Solving Reduced Models via Compilation

In this section we propose a compilation scheme for generating $\mathcal{M}_l^k$-reductions of a given PPDDL file, making it easy to leverage existing probabilistic planners to solve $\mathcal{M}_l^k$-reductions. The input to the compiler is a PPDDL description of the *original* problem, along with a set of *tags* that identify the set of primary outcomes. The output is a PPDDL description of the $\mathcal{M}_l^k$-reduction of the original problem with the specified set of primary outcomes. For the sake of clarity, we describe the approach for the case where $k = 1$, although it can be extended to higher values of k.

Tagging primary outcomes Our tagging scheme preserves the integrity of the original problem description, allowing the tagged file to be used as input for any planner that accepts PPDDL descriptions. Our tags are designed to be intuitive and easy to use, while adding only a negligible computational cost for planners operating on the tagged PPDDL file. To accomplish these goals we introduce a new predicate that will be used as a *tag* to designate primary outcomes; we will refer to this predicate as `primary`. Let

$$(\texttt{probabilistic}\ p_1\, e_1\, p_2\, e_2\, ...\, p_m\, e_m)$$

be a probabilistic outcome in the PPDDL description and let e_i be one of its primary outcomes. To tag e_i we simply replace e_i with the conjunctive outcome $(\texttt{and}\,(\texttt{primary})\,(e_i))$.

The compiler recognizes such special conjunctions (where one component is the `primary` predicate) as a signal that the tagged outcome is a primary outcome. Note that, as long as `primary` is set to true in the initial state of a problem instance, this simple tagging scheme will result in the same set of reachable states as the original problem (ignoring the `primary` predicate). As a consequence, the overhead for search-based probabilistic planners solving the tagged version of the original MDP will be negligible, as we confirmed experimentally using a set of tagged problems.

The compilation process We next show how to generate the reduced version of a tagged probabilistic outcome. To accomplish that, we introduce an additional predicate, `exc`, that represents the exception counter component of the state. In the case $k = 1$, we simply use `true` for $j = 1$ and `false` for $j = 0$. Using this predicate, we substitute every tagged probabilistic outcome $(\texttt{probabilistic}\ p_1\ e_1\ p_2\ e_2\ ...\ p_m\ e_m)$ by a conjunction of two conditional outcomes ce_1 and ce_2, with conditions $(\texttt{not}\ (\texttt{exc}))$ and $(\texttt{exc})$, respectively. For ce_1 (no exception occurred so far), the outcome is a probabilistic outcome of equal size and probability distribution as the original, but with every *exceptional* outcome e_i substituted by the conjunction $(\texttt{and}\ (e_i)\ (\texttt{exc}))$. For ce_2, the outcome is a probabilistic outcome containing only primary outcomes with normalized probabilities.

Figure 5 shows an example of a tagged PPDDL action schema (top) and its reduced version (bottom). Note that the `primary` predicate only appears in the tagged version of the original MDP, but not on the reduced version. For brevity we only show the parts that differ from the original.

```
(:action move-car
 :parameters (?l1 - loc ?l2 - loc)
 :precondition
   (and (vehicle-at ?l1) (road ?l1 ?l2)
    (not-flattire))
 :effect
  (and (vehicle-at ?l2) (not (vehicle-at ?l1))
    (probabilistic
      0.4(and (primary)
             (not (not-flattire))))))
```

```
(:action move-car
       ⋮
 :effect
  (and (vehicle-at ?l2) (not (vehicle-at ?l1))
   (when (not (exc))
    (probabilistic
      0.4 (not (not-flattire))
      0.6 (and (exc))))
   (when (exc)
    (probabilistic
      1.0 (not (not-flattire))))))
```

Figure 5: Example of a tagged PPDDL description (top) and the corresponding reduced PPDDL action (bottom).

We adopt a few conventions in our compilation scheme. First, PPDDL allows a probability-outcome pair to be left out from a `probabilistic` outcome description (i.e., $\sum_{i=1}^{m} p_i < 1$), implying that the remaining probability is assigned to the empty outcome (see Figure 5 for an example). If this occurs on a tagged PPDDL file, the compiler assumes the intended meaning is that the empty outcome is considered an exception. Otherwise, the probability of the empty outcome should be explicitly included in the description with a single `primary` predicate used to indicate the desired outcome.

Second, PPDDL allows arbitrary nesting of probabilistic and conditional outcomes to provide more compact representations. The compilation scheme described here can handle such nesting, using the convention that normalization of primary outcomes is done at the *local* level. For example, consider the following outcome:

```
(probabilistic
  0.2 *e1
  0.6 *(probabilistic 0.25 e2 0.3 *e3 0.45 *e4)
  0.2 e5)
```

where a star indicates a primary outcome. In this case, for $j = 1$ the reduced outcome will include the following probabilities: 0.25 to `e1`, 0.75×0.40 to `e3`, and 0.75×0.60 to `e4`.

Thus normalization at inner levels does not affect normalization at outer levels. This is in fact a desired property, since the probability of exceptional outcomes for $j = 1$ is being redistributed among those outcomes that share structural similarity in the state space.

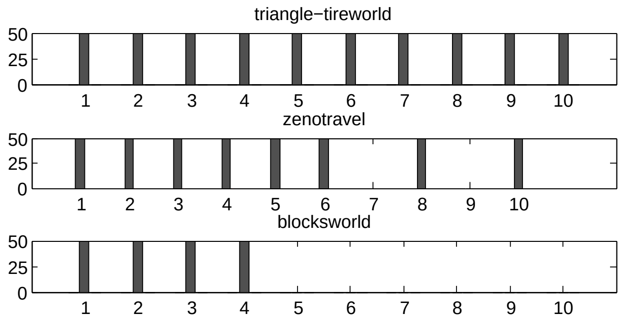

Figure 6: Number of rounds ending up in success for each problem in the domains considered.

Experimental Results

To validate our compilation approach we applied it to several problems taken from IPPC'08 (Bryce and Buffet 2008). In particular, we used the first 10 instances of three domains: blocksworld, triangle-tireworld and zenotravel. We use a methodology similar to that of the IPPCs, in which the planner is given a fixed amount of time to solve several rounds of the same problem (20 minutes for 50 rounds). In all problems we used the non-admissible FF heuristic for LAO* and $\mathcal{M}_1^1$-reductions for planning (generated using our compiler). We allow LAO* to reuse information obtained during past rounds of the same problem, by labeling solved states and keeping all values in the explicit graph for future use.

We chose the set of primary outcomes based on the performance of the continual planning approach on the first two problem instances of each domain. For the triangle-tireworld domain the primary outcome was the one corresponding to getting a flat tire. For blocksworld and zenotravel, to reduce complexity we only considered two possibilities for primary outcomes, namely, the most-likely outcome for all actions, or the least-likely outcome. For zenotravel, the best was labeling the least-likely outcomes as primary, while for blocksworld the best was choosing the most-likely outcomes.

Figure 6 shows the results of these experiments. Using the $\mathcal{M}_1^1$-reductions generated by our compiler, LAO* was able to generate successful plans for many of the problem instances in these domains. Although we didn't perform a direct comparison with other approaches (our goal in these experiments was just to validate our PPDDL compiler), we note that these results are on par with those reported for state-of-the-art planners for these domains (Trevizan and Veloso 2012).

8 Conclusions

We present a spectrum of MDP reduced models that can help reduce planning time and improve the overall performance of stochastic planning algorithms, when the cost of planning is taken into consideration. Each reduced model is characterized by two parameters: the maximum number of primary outcomes per action and the maximum number of exceptional outcomes in the plan. We show that reduced models can accelerate planning by orders of magnitude. We also introduce a continual planning paradigm that generates new plans in parallel to plan execution when the number of exceptions reaches the maximum allowed. A benefit of this paradigm is that it is amenable to a precise evaluation of the benefits of planning with reduced models.

Although the all-outcome determinization does not belong to this class of reductions, other commonly used determinization approaches represent reductions with one primary outcome per action and zero exceptions. We show that reduced models with either more than one primary outcome per action or with some exceptional outcomes (or both) can be beneficial, producing significantly better value.

We also investigate how to generate a good reduced model, be it a determinization or not. We show that the choice of primary outcomes is non-trivial, even when reductions are limited to determinization. We introduce a greedy algorithm that can produce good reduced models automatically, and evaluate its performance, again showing that performance can be improved relative to a baseline determinization technique or directly solving the original model.

Finally, to facilitate the use of these methods in practice, we introduce a compilation technique that allows one to easily solve reduced models by compiling a PPDDL description of the original problem into a description of the desired reduced model. Solving such compiled reduced models, we were able to obtain results that are competitive with state-of-the-art planners on a range of benchmark problems.

These results extend previous work demonstrating the value of reduced models in reinforcement learning (Ravindran and Barto 2002) and planning under uncertainty (Dean *et al.* 1997; Dean and Givan 1997; Givan *et al.* 2003). In particular, they place work on determinization in a broader context and lay the foundation for further work to increase the benefits offered by planning with reduced models.

There are a number of aspects of our approach that can be further improved. First, we are exploring better ways to redistribute the probabilities of exceptional outcomes among the primary outcomes based on various measures of *structural similarity* of the outcomes (e.g., similarity in outcome states or their values). Second, we can use an anytime version of LAO* (Zilberstein 1996), allowing us to produce the new partial plan within any given budget of time. The incomplete plan could be executed while we continue to increase the envelope of reachable states until it covers fully the desired reduced model. Third, we can start re-planning as soon as the first exception occurs, rather than wait for k exceptions to occur. In fact, planning could even start before any exception happens, simply by projecting the most likely exceptions to happen and planning ahead just in case, as in the *continual computation* framework (Horvitz 2001). Finally, we are developing more efficient ways to explore the space of $\mathcal{M}_l^k$-reductions and find good ones, for example, by using sampling to estimate the regret of labeling an outcome as primary. These promising research avenues will further increase the impact of planning with reduced models.

Acknowledgments

Support for this work was provided in part by the National Science Foundation under grant IIS-1116917.

References

Blai Bonet and Hector Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, pages 12–21, 2003.

Daniel Bryce and Olivier Buffet. 6th international planning competition: Uncertainty part. In *Proceedings of the 6th International Planning Competition (IPC'08)*, 2008.

Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, pages 106–111, 1997.

Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI'97)*, pages 124–131, 1997.

Carmel Domshlak. Fault tolerant planning: Complexity and compilation. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, pages 64–72, 2013.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223, 2003.

Eric A. Hansen and Shlomo Zilberstein. Heuristic search in cyclic AND/OR graphs. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, pages 412–418, Madison, Wisconsin, 1998.

Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14(1):253–302, 2001.

Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126(1-2):159–196, 2001.

Rune M. Jensen, Manuela M. Veloso, and Randal E. Bryant. Fault tolerant planning: Toward probabilistic uncertainty models in symbolic non-deterministic planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, pages 335–344, 2004.

Emil Keyder and Hector Geffner. The HMDPP planner for planning with probabilities. *The ICAPS 3rd International Probabilistic Planning Competition (IPPC'08)*, 2008.

Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented MDPs with dead ends. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'12)*, pages 438–447, 2012.

Iain Little and Sylvie Thiebaux. Probabilistic planning vs. replanning. In *Proceedings of the ICAPS'07 Workshop on the International Planning Competition: Past, Present and Future*, 2007.

Michael L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, pages 748–754, 1997.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.

Tuan Nguyen, Subbarao Kambhampati, and Minh Do. Synthesizing robust plans under incomplete domain models. *Neural Information Processing Systems (NIPS'13)*, 2013.

Luis Pineda, Yi Lu, Shlomo Zilberstein, and Claudia V. Goldman. Fault-tolerant planning under uncertainty. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 2350–2356, Beijing, China, 2013.

Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation (SARA'02)*, pages 196–211, 2002.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

Florent Teichteil-Königsbuch, Ugur Kuter, and Guillaume Infantes. Incremental plan aggregation for generating policies in MDPs. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 1231–1238, 2010.

Felipe W. Trevizan and Manuela M. Veloso. Short-sighted stochastic shortest path problems. *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 288–296, 2012.

Sung Wook Yoon, Alan Fern, and Robert Givan. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pages 352–359, 2007.

Sungwook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic planning via determinization in hindsight. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pages 1010–1016, 2008.

Sungwook Yoon, Wheeler Ruml, J. Benton, and Minh B. Do. Improving determinization in hindsight for online probabilistic planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 209–216, 2010.

Håkan L. S. Younes, Michael L. Littman, David Weissman, and John Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24(1):851–887, 2005.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.