Hindsight Optimization for Hybrid State and Action MDPs

Aswin Raghavan[1], Scott Sanner[2], Roni Khardon[3], Prasad Tadepalli[1], Alan Fern[1]
[1]School of EECS, Oregon State University, Corvallis, OR, USA. {nadamuna,tadepall,afern}@eecs.orst.edu
[2]Industrial Engineering, University of Toronto, Toronto, ON, Canada. ssanner@mie.utoronto.ca
[3]Department of Computer Science, Tufts University, Medford, MA, USA. roni@cs.tufts.edu

Abstract

Hybrid (mixed discrete and continuous) state and action Markov Decision Processes (HSA-MDPs) provide an expressive formalism for modeling stochastic and concurrent sequential decision-making problems. Existing solvers for HSA-MDPs are either limited to very restricted transition distributions, require knowledge of domain-specific basis functions to achieve good approximations, or do not scale. We explore a domain-independent approach based on the framework of hindsight optimization (HOP) for HSA-MDPs, which uses an upper bound on the finite-horizon action values for action selection. Our main contribution is a linear time reduction to a Mixed Integer Linear Program (MILP) that encodes the HOP objective, when the dynamics are specified as location-scale probability distributions parametrized by Piecewise Linear (PWL) functions of states and actions. In addition, we show how to use the same machinery to select actions based on a lower-bound generated by straight line plans. Our empirical results show that the HSA-HOP approach effectively scales to high-dimensional problems and outperforms baselines that are capable of scaling to such large hybrid MDPs.

Introduction and Related Work

Many real-world decision-theoretic planning problems are naturally modeled using concurrent, hybrid (discrete and continuous) state and action (HSA) MDPs. Examples include reservoir control under rainfall uncertainty (Reyes et al. 2015) and the unit commitment problem of power generation subject to demand uncertainty (Nikovski and Zhang 2010). Existing approaches to solving expressive HSA-MDPs largely fall into two categories: dynamic programming for special restricted classes of HSA-MDPs, and approximate optimization of restricted value function or policy representations. Unfortunately, each category has critical limitations discussed next.

For the case of exact or bounded approximate solutions, numerous (symbolic) dynamic programming approaches have been proposed for restricted classes of HSA-MDPs ranging from the univariate continuous state setting for time-dependent MDPs (Boyan and Littman 2001) to piecewise value representations (Feng et al. 2004; Li and Littman 2005; Sanner, Delgado, and de Barros 2011; Zamani et al. 2012) and phase-type transition approximations (Marecki, Koenig, and Tambe 2007). Later work introduced real-time dynamic programming extensions yielding more compact value functions (Meuleau et al. 2009; Vianna, De Barros, and Sanner 2015). Sample average approximations (SAA) (Kleywegt, Shapiro, and Homem-de Mello 2002) of dynamic programming can be used for value approximation in an initial state (Mercier and Van Hentenryck 2008). Unfortunately, all these dynamic programming approaches for HSA-MDPs are either too restrictive or computationally intractable for the state-action dependent stochastic, 100-dimensional hybrid state, 50-dimensional hybrid action, and moderate horizon domains we experiment with in this paper.

A different line of research directly optimizes a restricted class of value functions or policies. For value approximation, such methods are exemplified by Hybrid Approximate Linear Programming (HALP) (Kveton, Hauskrecht, and Guestrin 2006), which sought to approximate expressive HSA-MDP value functions via a weighted basis function representation. On the policy approximation side, approaches like Pegasus (Ng and Jordan 2000) sought to optimize restricted parameterized policy classes subject to trajectories sampled in an SAA setting. All of these methods assume *a priori* knowledge of a good value or policy representation, which is typically difficult to have in advance.

In this work, we explore a qualitatively different approach to solving a wide class of HSA-MDPs that does not require domain-specific assumptions on the value function or policy representation. Our approach is based on developing the framework of hindsight optimization (HOP) (Chang, Givan, and Chong 2000; Chong, Givan, and Chang 2000) for HSA-MDPs. HOP provides an upper bound on the finite-horizon action values in the current state, which can be used for action selection. But the challenge is to compute this bound efficiently.

In particular our contributions are as follows: (i) We develop a generic linear space and time compilation of an expressive fragment of the RDDL (Sanner 2010) HSA-MDP representation to a mixed integer linear program (MILP). This compilation is augmented with action constraints to yield different algorithmic variations. (ii) Our main contribution is the hindsight optimization variant. This generalizes previous work on HOP (Issakkimuthu et al. 2015) to handle both continuous random variables and state-action dependent stochasticity. (iii) We develop a second variant based on straight line plans which is complementary in that it provides a lower bound on action values. (iv) Empirical results show that HSA-HOP scales to HSA-MDPs with moderate horizons and high-dimensional state and action spaces and generally outperforms baselines that are capable of scaling to such large hybrid MDPs.

Hybrid State and Action MDPs

A **discrete-time** MDP is a tuple $(\mathbb{S}, \mathbb{A}, T, R)$ where $\mathbb{S}$ is a state set, $\mathbb{A}$ is an action set, $T : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow [0,1]$ denotes the stochastic transition function for time $t+1$ such that $T(s^t, a^t, s^{t+1}) = P(s^{t+1}|s^t, a^t)$, and $R : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$ denotes the state-action reward function. In this paper we focus on finite horizon planning for a specified horizon h with the objective of maximizing the expected total reward accumulated over h steps.

Factored state and action MDPs (Boutilier, Dean, and Hanks 1999; Raghavan et al. 2012) extend the basic setting by specifying the state and action spaces as products of discrete variables. Hybrid State and Action MDP (HSA-MDP) (Kveton, Hauskrecht, and Guestrin 2006; Sanner, Delgado, and de Barros 2011) keep the factored structure but provide a significant extension by allowing for both continuous and discrete state and action variables. The state space $\mathbb{S}$ and action space $\mathbb{A}$ are represented by finite sets of state variables $\mathbf{X} = \{X_1, \dots, X_l\}$ and action variables $\mathbf{A} = \{A_1, \dots, A_m\}$, where both $\mathbf{X}$ and $\mathbf{A}$ can contain both continuous and discrete random variables.

The transition function of the MDP is factored over the state variables, ie. $P(s'|s,a)$ is represented as a product of conditional probability distributions $P_i(q_i) = P(X_i'|q_i)$ for $i = 1, \dots, l$. Each $P_i(q_i)$ is a function of a (typically small) subset $q_i \subseteq \{\mathbf{X}, \mathbf{A}, \mathbf{X}'\}$ where $\mathbf{X}'$ are the next state variables and the set of dependencies is acyclic. At any time t and state variable $X_i \in \mathbf{X}$, $X_i^{(t+1)} \sim P_i(q_i^{(t)})$.

RDDL Representation of Reward and Dynamics

Following recent work on HSA-MDPs (Sanner, Delgado, and de Barros 2011; Zamani et al. 2012; Vianna, De Barros, and Sanner 2015), we use the description language RDDL (Sanner 2010) to specify HSA-MDPs. RDDL allows for relational templates to specify random variables which are instantiated via a set of objects. As a preprocessing step, we ground the templates with a concrete set of objects and expand relational quantifiers over their finite domains to obtain a propositional HSA-MDP as defined above. We note that the RDDL simply provides a convenient interface and our approach is not restricted to this language.

The RDDL source code specifies the factored transition and reward functions through a sequence of assignment and sampling statements that define intermediate variables and next state variables, using algebraic expressions in these definitions. The crucial point for a well-defined RDDL model is that the dependence among variables arising from the sequence of statements is acyclic.

HSA-MDPs with Piecewise Linear (PWL) dynamics have received significant attention due to their simplicity and expressivity (Feng et al. 2004; Li and Littman 2005; Meuleau et al. 2009; Zamani et al. 2012). Whereas exact Dynamic Programming (DP) approaches for HSA-MDPs with PWL rewards and piecewise constant transition probabilities have been explored (Feng et al. 2004; Zamani et al. 2012), these cannot handle probability distributions whose parameters are continuous functions of state and action variables. In this case the optimal value function need not be PWL as required by previous work, making exact DP impossible. In contrast, our definition allows for a general form of stochasticity which we define next as the **stochasticPWL** class.

Definition 1. An expression for state variable X (or reward variable R) is a **stochasticPWL** expression if it is built recursively using the following 3 cases:

(a) A *deterministic* PWL expression containing (1) scalars, current state, current action, defined intermediate variables and next state variables, (2) boolean operations $\wedge, \vee, \neg, \geq$, (3) linear combination with constant coefficients, and (4) multiplication with a boolean predicate. The syntactic structure of each of these cases is shown in Table 1.

(b) If X a discrete random variable with support $\{1, \dots, n\}$, its probability mass function is parametrized as $(E_1, \dots, E_n)$ where each of $E_1, \dots, E_n$ is a stochasticPWL expression.

(c) If X is a continuous random variable in the location-scale family of distributions (Mukhopadhyay 2000), it is parameterized with a PWL transformation $X = E_1 + E_2 Z$, where E_1 (aka location) and E_2 (aka scale) are stochasticPWL expressions, and Z is a random variable for the standardized form of the distribution with known Cumulative Distribution Function (CDF).

The location-scale family includes many distributions of practical use such as the uniform, gaussian, exponential, logistic, beta, gamma distributions (Mukhopadhyay 2000). For example, the following RDDL expressions are valid according to Definition 1, where x^t is a state variable, a^t is an action variable, and z^t is an intermediate variable used in calculating r^t the reward.

$$x^{t+1} = \max(1, \min(0, 0.2x^t + 0.7a^t)) \quad (1)$$

$$z^{t+1} = \text{Normal}(x^t, a^t), a^t > 0 \quad (2)$$

$$r^{t+1} = \text{if } (x^t < z^{t+1}) \text{ then } x^{t+1} \text{ else } 1 - x^{t+1} \quad (3)$$

These illustrate the recursive structure of expressions, the limitation to PWL, and state-action dependent parameters of random variables. We emphasize that both state variables and action variables can be either discrete or continuous.

While PWL is a practical restriction, PWL functions (Keha, de Farias, and Nemhauser 2004) are an arbitrarily good approximation for higher order transition functions (Dunham 1986). As will be clear below, the restriction to PWL is due to translating the RDDL code into mixed integer *linear programs* (MILP) and using MILP solvers. By using more powerful solvers one can expand this approach.

Deterministic PWL Expression	Condition	MILP Constraints
$E \to k$ $E_b \to$ true ($E_b \to$ false, respectively) $E \to p$	k is a constant p is a state or action variable	$v_E = k$ $v_{E^b} = 1.0$ ($v_{E^b} = 0.0$, respectively) $v_E = v_p$
$E \to \wedge_{i=1}^{n} E_b^i \equiv \forall_i E_b^i$ $E \to \vee_{i=1}^{n} E_b^i \equiv \exists_i E_b^i$ $E \to \neg E_b$	E_b^i is a boolean expression	$nv_E \le \sum_{i=1}^{n} v_{E_b^i} \le (n-1) + v_E$ $v_E \le \sum_{i=1}^{n} v_{E_b^i} \le nv_E$ $v_E = 1 - v_{E_b}$ $v_E = 0$ or 1. Each $v_{E_b^i} = 0$ or 1
$E \to kE_1$ $E \to E_1 \; op \; E_2$	k is a constant $op = +$ or $-$	$v_E = kv_{E_1}$ $v_E = v_{E_1}$ op v_{E_2}
$E \to E_b E_1$ $E_b \to E_1 \ge E_2$	E_b is a boolean expression	Same as $E \to$ if E_b then E_1 else 0 $-M(1 - v_{E_b}) \le v_{E_1} - v_{E_2} \le Mv_{E_b}$ $v_{E_b} = 0$ or 1, M is large e.g. $M = 10^6$
$E \to$ if E_b then E_1 else E_2	E_b is a boolean expression	$v_{E_1} - M(1 - v_{E_b}) \le v_E \le v_{E_1} + M(1 - v_{E_b})$ $v_{E_2} - Mv_{E_b} \le v_E \le v_{E_2} + Mv_{E_b}$ $v_{E_b} = 0$ or 1, M is large e.g. $M = 10^6$

Table 1: The syntax of Deterministic PWL statements and their MILP encoding. In each case E_1 and E_2 belong to the same language as E without cycles. The function represented by the expression E is equivalent to the free variable v_E in MILP.

Hindsight Optimization (HOP) for HSA-MDPs

HOP (Chang, Givan, and Chong 2000; Chong, Givan, and Chang 2000) is a computationally attractive heuristic for online action selection. HOP approximates the optimal value function by optimally solving different realizations of the MDP called *futures* or *determinizations* and aggregating their values. While it is easy to construct domains where the HOP heuristic fails, previous work has shown that it performs well in many benchmark probabilistic planning problems (Yoon et al. 2008). Recent work has shown how to apply HOP in discrete factored MDPs through a translation to Integer Linear programs (Issakkimuthu et al. 2015). In this paper, we show how these ideas can be extended in two respects in order to handle HSA-MDPs: First, we allow the more general state-action dependent stochasticity of Definition 1 (previous work restricted stochasticity to a small number of state-independent cases). Second, we allow for continuous and discrete variables.

The notion of a **random future** is central to the idea of HOP. Given a fixed policy, the MDP model induces a distribution over length-h trajectories. Viewing the choice of policy, and the random choices of the MDP as separate processes, we can make the random choices in advance (e.g., fixing the seed for the random number generator). This selection which, conditioned on any policy, produces a random trajectory for the policy is known as a random future.

HOP requires sampling random futures and once the random choices are fixed, a future represents a deterministic planning problem. The optimal value of any state in the MDP is $V_h^*(s^0) = \max_\pi E_f[\sum_{t=0}^{h} R(s_f^t, \pi_f^t)]$, which is the maximum expected value over random futures f of length h. In contrast, the hindsight value $V_h^{hop}(s^0) = E_f[\max_{a_f^0,\ldots,a_f^h} \sum_{t=0}^{h} R(s_f^t, a_f^t)]$ is the expected value of the optimal values of each future, where the inner maximization optimizes a *plan* (instead of a policy) for each future.

Observe that the maximizing values of actions a_f^t are future-dependent, i.e., "in hindsight" assuming a particular outcome of s_f^{t+1}. Due to swapping expectation and maximization and Jensen's inequality the function V^{hop} is an upper bound on V^* (Mercier and Van Hentenryck 2007).

Action selection in HOP uses one-step lookahead using V^{hop} so that the outcome of the first action is not assumed. For each action a, the next states $s_1 \ldots s_F$ are sampled and their V^{hop} value is used. The HOP algorithm picks $\arg\max_{\mathbf{a}} Q_h^{hop}(s_0, a)$ with

$$Q_h^{hop}(s_0, a) \approx R(s_0, a) + \frac{1}{F} \sum_{f=1}^{F} V_{h-1}^{hop}(s_f) \,. \quad (4)$$

Reduction of Deterministic HSA-MDPs to MILP

In this section we show how to translate a deterministic HSA-MDP, for example as generated by the determinization process of the next section. After determinization each P_i and R will be Deterministic PWL as given by Definition 1(a). The optimal plan is the solution to the following Mixed Integer Linear Program (MILP):

$$\max \sum_{t=0}^{h} R(\mathbf{X}^t, \mathbf{A}^t) \quad (5)$$

$$\text{s.t. } X_i^t = P_i(q_i^{t-1}), \;\; i = 1, \ldots, l; t = 1, \ldots, h; \quad (6)$$

where $\mathbf{X}^0 = s_0$ is a given initial state and $\mathbf{X}^t = (X_1^t, \ldots, X_l^t)$ and $\mathbf{A}^t = (A_1^t, \ldots, A_m^t)$ are the optimization variables for each $t = 0, \ldots, h-1$.

The translation of deterministic $P_i()$ to a set of MILP constraints is done recursively using the syntax in Definition 1(a). We formalize the syntax in the form of a recursive grammar in Table 1. The translation to MILP constraints is given in the third column and is standard for most cases. The encoding of if-then-else expressions requires the use of a large constant (i.e., "big-M trick") that can be chosen generically. Let the size of an expression be characterized by the size of the abstract syntax tree that produces the expression.

Proposition 1. *(Linear-time Reduction) Given a deterministic HSA-MDP specified in the language of Definition 1(a), the MILP in Equations 5-6 is such that (1) the compilation is produced in linear-time w.r.t. the number of constraints, the size of each PWL expression, the number of state variables and the planning horizon, and (2) an optimal solution of the MILP is an optimal plan for the deterministic HSA-MDP.*

Determinization of HSA-MDPs

Next we consider the determinization of HSA-MDPs by determinizing stochasticPWL expressions. The key question is how to sample a random future when the parameters of the random variables are not completely known at compile time, and are dependent on the states within a trajectory. We propose to encode random futures in a MILP using inverse transform sampling. Intuitively, we first sample a $u \sim \text{Uniform}(0,1)$ and encode the quantile of order u as a set of MILP constraints. More precisely for any $u \in [0,1]$,

1. If X is a discrete variable, its cumulative distribution function is $(E_1, E_1 + E_2, \ldots, \sum_{i=1}^{n} E_i)$ and is PWL. The u-quantile is encoded in the PWL constraints: (1) $S_j = \sum_{i=1}^{j} E_j$ for $j = 1, \ldots, n$, (2) $x = \sum_{i=1}^{n} i \times [(S_i \geq u) \wedge (S_{i-1} < u)]$, and (3) $x \in \{1, 2, \ldots, n\}$.
2. If X is a continuous variable as in Definition 1, its quantile function is $F_X^{-1}(u; E_1, E_2) = E_1 + E_2 F_Z^{-1}(u)$ and is PWL. The u-quantile is encoded in the MILP constraint $x = E_1 + E_2 \times F_Z^{-1}(u)$, where $F_Z^{-1}(u)$ is a constant.

Proposition 2. *(Correctness for single variable sampling) Let X_i be a state variable whose transition function is given by a stochasticPWL expression as in Definition 1. Then, the MILP variable x produced by the above procedure encodes a random future for X_i given its parents q_i.*

Note that u and $F_Z^{-1}(u)$ are constants that are known at compile time. Yet the corresponding deterministic expressions for X yield a sample from the state dependent distribution which is not known at compile time. Consider for example the variables in Equation 2 and a pre-determined $u = 0.23$. The constraint $z^{t+1} = x^t + 0.23a^t$ produced by the above procedure encodes the 0.23-quantile outcome for every value of x^t and a^t.

The overall determinization algorithm applies the above procedure recursively on the syntactic structure of each X_i^{t+1}, using the corresponding deterministic or probabilistic MILP translation. By using Proposition 2 inductively over the acyclic structure of the dependencies in the RDDL code:

Proposition 3. *(Correctness of future generation) The MILP constructed using the overall determinization procedure is a random future from the distribution induced by the RDDL source code.*

HOP for HSA-MDPs

We next describe the overall MILP using multiple sampled determinizations. For a given HSA-MDP we produce copies of the state and action variables, annotated with superscripts f, t, where f is the future index, t the time step, and the optimization variables are $X_i^{f,t}$ and $A_j^{f,t}$. We generate F futures using the determinization procedure above. The objective function of the MILP is the h-step accumulated reward averaged over F futures. This objective and the first set of constraints is used by all of our algorithms:

$$\max \frac{1}{F} \sum_{f=1}^{F} \sum_{t=0}^{h} R(\mathbf{X}^{f,t}, \mathbf{A}^{f,t}) \tag{7}$$

$$\text{s.t. } X_i^{f,t} = \text{ Determinization of } P_i(q_i^{f,t-1}) \tag{8}$$

$$i = 1, \ldots, l; f = 1, \ldots, F; t = 1, \ldots, h \tag{9}$$

$$\mathbf{X}^{f,0} = s_0, f = 1, \ldots, F \tag{10}$$

The **HSA-HOP** algorithm uses one additional set of constraints restricting the action variables at time step $t = 0$ to be identical across futures:

$$A_j^{f,0} = A_j^{0,0}, j = 1, \ldots, m; f = 1, \ldots, F \tag{11}$$

Observe that the MILP in Equations 7-10 solves all futures independently. When the constraint in Equation 11 is added, the MILP implements a one-step lookahead where the solutions are coupled by requiring the first action to be the same.

Proposition 4. *(Equivalence to HOP)* **(Issakkimuthu et al. 2015)** *The MILP in Equations 7-11 identifies the same solutions as the explicit HOP construction in Equation 4.*

Proof. The proof is due to the first term of the objective function being identical across futures for any fixed s_0 and any feasible solution to $A_j^{0,0}$, resulting in Equation 4. $\square$

This construction identifies the HOP solutions in factored spaces without state and action enumeration and without the additional enumeration or continuous maximization implicit when using Equation 4 for action selection.

Algorithmic Baselines and Variations

As a baseline we use the simple idea that determinizes the problem using the most likely outcome determinization (for discrete variables) and the expected outcome determinization (for continuous variables). We use this baseline in our experiments denoting it as **Mean**. This idea is not new but the challenge is to encode it without enumeration of state-action dependent parameters of random variables. This can be done using Definition 1 with the MILP encoding: (1) If X is a discrete stochasticPWL variable, its most likely determinization $\max(E_1, \ldots, E_n)$ is PWL and equivalent to a MILP constraint (Table 1). (2) If X is a continuous state variable, its expected outcome determinization $E(X) = E_1 + E_2 E(Z)$ is PWL because $E(Z)$ is known.

We also consider two alternative formulations to HSA-HOP. The first is based on the idea of a **straight line plan** also known as an open loop policy or conformant plan. In this case we commit to a sequence of future actions regardless of the probabilistic outcomes of earlier actions. We can achieve this in the MILP formulation by replacing the constraint in Equation 11 with

$$A_j^{f,t} = A_j^{0,t}, j=1, \ldots, m; f=1, \ldots, F; t=0, \ldots, h-1 \tag{12}$$

The straight line value converges to the optimal value of the best open loop policy as the number of futures increases.

Since in this case we are limiting the set of policies, the value of the optimal straight line plan is a lower bound on the optimal value of any state. The gap between the optimal HOP and straight line value can be used in various ways, e.g., to guide the generation of futures, to detect convergence, and to calculate approximation guarantees. Although this formulation commits to an entire plan, in our evaluation at each step we only use the first action from that plan, exactly like the other algorithms, and then replan for the next step.

The second variant is **Consensus**: determinizations are sampled exactly as in HSA-HOP, but solved independently of other determinizations. An action is selected by majority vote (ties broken randomly) among the $A_j^{0,f}$ across the futures. This trades off the monolithic MILP of HSA-HOP with several independent MILPs (one for each future) and aggregates their solutions heuristically.

Experimental Evaluation

As previously mentioned, we use the description language RDDL (Sanner 2010) to specify the domain dynamics. In all experiments we use the Gurobi optimizer (Gurobi Optimization 2015) for optimizing the MILPs. We compare our algorithms **HOP** and **Straight Line** to the baselines of **Mean** and **Consensus**. When applicable we show the performance of **Noop**, **Random** and hand-coded policies.

The different algorithms are evaluated in an online replanning mode, i.e., planning is repeated at every world state and one action is output. The average accumulated reward over a horizon of 20 steps is measured (averaged over 30 trials) and a 95% confidence interval is shown. Each evaluation has three experimental parameters : (1) Time limit t per decision in minutes, (2) Lookahead h, the length of sampled futures and, (3) Number of sampled futures F per decision. We evaluated the algorithms by setting a reasonable value for t keeping in mind the runtime, then increasing h and F for best performance, until the MILP solver throws a memory error caused by an excessively large MILP. We use the best feasible solution found for any MILP after t minutes.

Domains

Power Generation (Nikovski and Zhang 2010; Angelidakis and Chalkiadakis 2015) : This domain concerns the unit commitment problem for a set of independent power plants represented as a Factored MDP. Each plant i has a current reserve of $\texttt{stock}_i$ ($\texttt{stock}_i \geq 0$) and observes fluctuations in $\texttt{temperature}_i$ with a uniform distribution as $\texttt{temperature}_i^{t+1} \sim \texttt{Uniform}(\mu_i - \delta_i, \mu_i + \delta_i)$. The fluctuations in temperature create a $\texttt{demand}_i$ for heating as well as cooling $\texttt{demand}_i^{t+1} = \mu_i|\texttt{temperature}_i^{t+1} - \mu_i|$. In this simple example, we assume μ_i and δ_i are constant, so this domain has state-independent continuous stochasticity. The demand is always positive with a V-shape, centered with value zero when temperature$_i$ $= \mu_i$. The objective is to optimize $\texttt{order}_i$, the real-valued units of power to be generated at plant i within a prespecified budget $\sum_i \texttt{order}_i \leq B$ (B constant), produced at a cost of \$0.5 per unit and consumed at \$1 per unit $\texttt{reward}^{t+1} = \sum_i[\texttt{min}(\texttt{demand}_i^{t+1}, \texttt{stock}_i^t) - 0.5\texttt{order}_i^t]$. Unconsumed power is carried over as $\texttt{stock}_i^{t+1} =$ $\texttt{if}\ (\texttt{stock}_i^t < \texttt{demand}_i^{t+1})\ \texttt{then}\ (\texttt{order}_i^t)\ \texttt{else}\ (\texttt{stock}_i^t - \texttt{demand}_i^{t+1} + \texttt{order}_i^t)$. This domain consists of a hybrid state space and continuous action space.

The mean temperature is μ_i and the demand for the mean temperature is zero. Thus, the **Noop** policy is optimal with respect to the **Mean** determinization. The **maximum demand** is $\mu_i\delta_i$ per time step and **expected demand** is $\frac{\mu_i\delta_i}{2}$[1].

Reservoirs problem (Reyes et al. 2015) : This problem consists of hybrid states with state-dependent noise in the transitions. The problem consists of a set of reservoirs connected by a set of 2-ended bidirectional pipes. Although any topology can be encoded, we demonstrate a linear topology of reservoirs and say that each reservoir i is downstream of reservoir $i-1$ and connected by pipe i. An MDP state is a list of positive current water levels $\texttt{rlevel}_i$ and $\texttt{rain}_i$ for each reservoir i. An MDP action is a list of real-valued flows $\texttt{flow}_i$, one for each pipe i connecting reservoirs $i-1$ and i. The sign of $\texttt{flow}_i$ determines direction of flow (positive for downstream). The rain level is stochastic as a zero truncated Gaussian distribution $\texttt{rain}_i^{t+1} = \texttt{max}[0, \texttt{Normal}(\texttt{rain}_i^t, \sigma)]$, and the flow is deterministic as $\texttt{rlevel}_i^{t+1} = \texttt{rlevel}_i^t + \texttt{rain}_i^t - \texttt{flow}_i^t + (\texttt{flow}_{i-1}^t > 0)\texttt{flow}_{i-1}^t + (\texttt{flow}_{i+1}^t < 0)\texttt{flow}_{i+1}^t$.

Each reservoir has a preset minimum α_i and maximum β_i water level. The objective is to minimize overflow and underflow of the reservoirs outside the prespecified limits, encoded by the reward function $\texttt{reward}^{t+1} = -5\sum_i|\texttt{max}(0, \alpha_i - \texttt{rlevel}_i^t, \texttt{rlevel}_i^t - \beta_i)| - \sum_i|(\texttt{flow}_i^t < 0)\texttt{flow}_i^t|$.

The second term penalizes upstream flow (checked via the sign of $\texttt{flow}_i$) by the magnitude of the flow. PWL constraints are included in the RDDL to enforce (1) total outflow does not exceed current water level and (2) non-negativity of rain and water levels in each reservoir. In addition, each reservoir has a maximum capacity.

Icetrack : We illustrate a stochastic version of the classical Racetrack problem (Barto, Bradtke, and Singh 1995). In contrast to the previous domains which have an optimization flavor, Icetrack is goal based and requires a large lookahead. Icetrack consists of real valued states of the form (x, y, v_x, v_y) where (x, y) is the real-valued position of a car on ice, and (v_x, v_y) are its velocities in the x and y directions respectively. The control inputs are the accelerations (a_x, a_y) in the two directions. The actions are susceptible to failure with a fixed probability θ with $v_x =$ $\texttt{if Bernoulli}(1-\theta)\ \texttt{then}\ v_x + a_x\ \texttt{else}\ v_x$. The transition for v_y is analogous. So this domain demonstrates state-independent discrete noise. We specify a goal state and measure the negative of the number of steps to reach it. Any collision makes the car unable to move for the rest of the episode. Icetrack shows two limitations of our PWL modeling restriction viz. we cannot use polar form (v, θ) as the state updates would require non-PWL functions, and in our discrete time model, collisions between (x^t, y^t) and (x^{t+1}, y^{t+1}) are given a bilinear function. So we restrict

[1] $E[\text{demand}_i] = \frac{\mu_i}{2\delta_i}[\int_{x=\mu_i-\delta_i}^{\mu_i}(\mu_i - x_i)dx + \int_{x=\mu_i}^{\mu_i+\delta_i}(x_i - \mu_i)dx]$

the track to be rectilinear (axis-parallel walls) to allow PWL collision detection (details omitted due to space constraints).

Instances

In the Power Generation domain, we varied the number of plants between 10 and 50. All the plants start with 0 stock, the demand constants are set $\mu_i = 10$ and each δ_i is a fixed integer sampled uniformly between $\delta_i = 3$ and $\delta_i = 8$. In the Reservoirs problem, we varied the number of reservoirs between 10 and 50 but only show the three largest instances due to space constraints. We set $\alpha_i = 1000$, $\beta_i = 8000$ and capacity to 40000 for all reservoirs. All reservoirs are empty in the initial world state and $\sigma = 1024$. In the Icetrack problem, we tested with a 2 cell wide track along the edges of a 10×10 grid, initial position set to $(5, 1)$ and goal $(5, 9)$. The acceleration is restricted $a_x, a_y \in [-4, 4]$, and the slippage θ is varied from 0% to 20%.

Results

The results are shown in Figure 1 for Power Generation, Table 2 for Reservoirs and Figure 2 for Icetrack. Overall, we see that HSA-HOP (denoted as HOP) performs the best across our three evaluation domains and Straight Line performs equally well in two of them.

Sampled vs. static determinization: In Power Generation, Mean is equivalent to Noop[2] and achieves a total reward of zero, whereas HOP and Straight Line achieve much higher rewards. In the Reservoirs problem (Table 2), we see that the performance of Mean degrades as the number of reservoirs increases in comparison with HOP and Straight Line. Further, Mean has a higher variance and wider confidence interval across episodes whereas HOP appears more stable. Finally in Icetrack, mean determinization assumes that the actions always succeed. This assumption is true in the leftmost instance (Figure 2) which has zero noise, but we see that the performance of Mean is suboptimal, whereas using multiple futures (as in HOP) performs better. We found that this suboptimality is caused by numerical instabilities in the output of Gurobi across runs. As the noise increases, the performance of Mean is never better than HOP, because HOP accounts for the failure of actions.

Action Selection: The algorithms HOP, Straight Line and Consensus differ only in their action selection via the constraints they impose on action variables. Our HOP algorithm performs the best in this regard, and performance of Straight Line and Consensus vary. The Straight Line algorithm performs well in Power Generation and Reservoirs, and the small gap from the performance of HOP suggests the existence of high quality open-loop policies. In Icetrack, the strong action constraints of Straight Line, and the large lookahead of the domain, make the MILP significantly harder and leads to poor performance. The poor performance of Consensus is not surprising when continuous stochasticity is presented. We found that the average consensus among root actions went from 20% in Power Generation, to 55% in Reservoirs, to 95% in the discrete noise case of Icetrack. Overall, HOP strikes a balance between optimality and hardness of the MILP, and gives the best performance over other action selection methods.

[2] $E[\texttt{temperature}_i] = \mu_i$, $\texttt{demand}_i = \mu_i(\mu_i - \mu_i) = 0$.

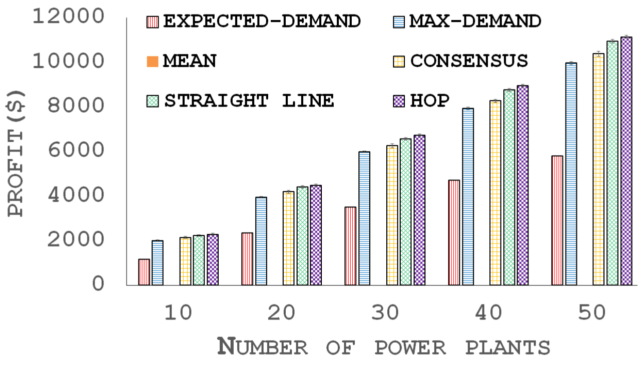

Figure 1: Results in the Power Generation problem with lookahead $h = 4, t = 0.5$ (mins) and $F = 5$ per decision.

Reward ($\times 10^5$)	# Reservoirs		
Algorithms	30	40	50
HOP	**-2.4** (0.22)	**-1.51** (**0.02**)	**-2.27** (0.01)
Mean	$-2.42(0.22)$	$-1.57(0.15)$	$-2.32(0.21)$
Straight Line	$-2.51(0.22)$	$-1.56(0.01)$	$-2.29(0.01)$
Consensus	$-2.78(0.21)$	$-1.78(0.05)$	$-2.61(0.04)$
NoOp	$-2.76(0.25)$	$-3.71(0.14)$	$-4.38(0.13)$
Random	$-2.76(0.25)$	$-6.41(0.17)$	$-7.94(0.19)$

Table 2: Average Accumulated Reward ($\times 10^5$) and 95% Confidence Intervals with increasing reservoirs (columns), setting $h = 4$, $t = 2$ (mins) and $F = 5$ per decision.

Conclusion

We introduced a new approach for online action selection in HSA-MDPs, which is characterized by having both discrete and continuous state and action variables and state-action dependent stochasticity. Our main algorithm HSA-HOP significantly improves on the scalability of previous approaches by leveraging state-of-the-art MILP solvers, scaling to MDPs with 100 continuous state dimensions and 50 continuous action dimensions. The key to the scalability is our linear space and time compilation from the RDDL language to a MILP whose solution returns a hybrid action that

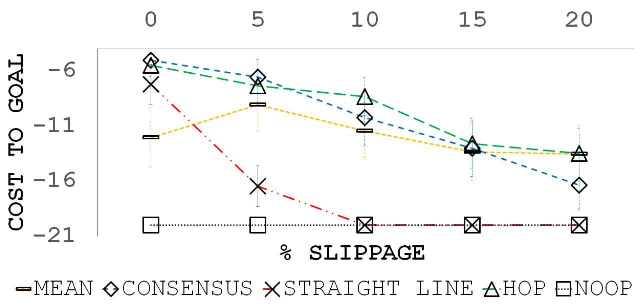

Figure 2: Results in Icetrack with settings $h = 20$, $t = 1$(mins) and $F = 5$.

maximizes an upper bound on action values. Our approach strictly generalizes previous work on ILP reduction of HOP for discrete Factored MDPs (Issakkimuthu et al. 2015) and resolves the critical problem of how to determinize transition distributions whose parameters are state-action dependent. The alternative algorithm using straight line plans provides a complementary heuristic based on a lower bound approximation. This is found to be competitive in some cases but overall less robust than HSA-HOP.

Acknowledgements

This work was partly supported by NSF under grants IIS-0964457, IIS-1219258, IIS-1616280, and IIS-1619433.

References

Angelidakis, A., and Chalkiadakis, G. 2015. Factored MDPs for Optimal Prosumer Decision-Making. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 503–511. International Foundation for Autonomous Agents and Multiagent Systems.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to Act Using Real-Time Dynamic Programming. *Artificial Intelligence* 72(1):81–138.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *J. of Artif. Intell. Res.* 11(1):94.

Boyan, J. A., and Littman, M. L. 2001. Exact Solutions to Time-Dependent MDPs. In *Advances in Neural Information Processing Systems*, 1026–1032.

Chang, H. S.; Givan, R. L.; and Chong, E. K. 2000. On-line scheduling via sampling. In *Proceedings of the Conference on Artificial Intelligence Planning and Scheduling*.

Chong, E. K.; Givan, R. L.; and Chang, H. S. 2000. A framework for simulation-based network control via hindsight optimization. In *IEEE CDC '00: IEEE Conference on Decision and Control*.

Dunham, J. G. 1986. Optimum Uniform Piecewise Linear Approximation of Planar Curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (1):67–75.

Feng, Z.; Dearden, R.; Meuleau, N.; and Washington, R. 2004. Dynamic Programming for Structured Continuous Markov Decision Problems. In *UAI*.

Gurobi Optimization, I. 2015. Gurobi Optimizer Reference Manual.

Issakkimuthu, M.; Fern, A.; Khardon, R.; Tadepalli, P.; and Xue, S. 2015. Hindsight Optimization for Probabilistic Planning with Factored Actions. In *ICAPS*.

Keha, A. B.; de Farias, I. R.; and Nemhauser, G. L. 2004. Models for Representing Piecewise Linear Cost Functions. *Operations Research Letters* 32(1):44–48.

Kleywegt, A. J.; Shapiro, A.; and Homem-de Mello, T. 2002. The Sample Average Approximation Method for Stochastic Discrete Optimization. *SIAM Journal on Optimization* 12(2):479–502.

Kveton, B.; Hauskrecht, M.; and Guestrin, C. 2006. Solving Factored Mdps with Hybrid State and Action Variables. *J. Artif. Intell. Res.(JAIR)* 27:153–201.

Li, L., and Littman, M. L. 2005. Lazy Approximation for Solving Continuous Finite-Horizon MDPs. In *AAAI*, volume 5, 1175–1180.

Marecki, J.; Koenig, S.; and Tambe, M. 2007. A Fast Analytical Algorithm for Solving Markov Decision Processes with Real-Valued Resources. In *IJCAI*, 2536–2541.

Mercier, L., and Van Hentenryck, P. 2007. Performance Analysis of Online Anticipatory Algorithms for Large Multistage Stochastic Integer Programs. In *IJCAI*, 1979–1984.

Mercier, L., and Van Hentenryck, P. 2008. AMSAA: A Multistep Anticipatory Algorithm for Online Stochastic Combinatorial Optimization. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 173–187.

Meuleau, N.; Benazera, E.; Brafman, R. I.; Hansen, E. A.; and Mausam, M. 2009. A Heuristic Search Approach to Planning with Continuous Resources in Stochastic Domains. *Journal of Artificial Intelligence Research* 34(1):27.

Mukhopadhyay, N. 2000. *Probability and Statistical Inference*. CRC Press.

Ng, A. Y., and Jordan, M. 2000. Pegasus: A Policy Search Method for Large MDPs and POMDPs. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 406–415. Morgan Kaufmann Publishers Inc.

Nikovski, D., and Zhang, W. 2010. Factored Markov Decision Process Models for Stochastic Unit Commitment. In *Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES), 2010 IEEE Conference on*, 28–35. IEEE.

Raghavan, A.; Joshi, S.; Fern, A.; Tadepalli, P.; and Khardon, R. 2012. Planning in Factored Action Spaces with Symbolic Dynamic Programming. In *AAAI*.

Reyes, A.; Ibarguengoytia, P. H.; Romero, I.; Pech, D.; and Borunda, M. 2015. Open Questions For Building Optimal Operation Policies for Dam Management using Factored Markov Decision Processes. In *2015 AAAI Fall Symposium Series*.

Sanner, S.; Delgado, K. V.; and de Barros, L. N. 2011. Symbolic Dynamic Programming for Discrete and Continuous State MDPs.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description.

Vianna, L. G.; De Barros, L. N.; and Sanner, S. 2015. Real-Time Symbolic Dynamic Programming for Hybrid MDPs.

Yoon, S. W.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic Planning via Determinization in Hindsight. In *AAAI*, 1010–1016.

Zamani, Z.; Sanner, S.; Fang, C.; et al. 2012. Symbolic Dynamic Programming for Continuous State and Action MDPs. In *AAAI*.