Symbolic Dynamic Programming for Continuous State MDPs with Linear Program Transitions

Jihwan Jeong*, **Parth Jaggi***, **Scott Sanner**†
Department of Mechanical & Industrial Engineering, University of Toronto
jhjeong@mie.utoronto.ca, parth.jaggi@mail.utoronto.ca, ssanner@mie.utoronto.ca

Abstract

Recent advances in symbolic dynamic programming (SDP) have significantly broadened the class of MDPs for which exact closed-form value functions can be derived. However, no existing solution methods can solve complex discrete and continuous state MDPs where a linear program determines state transitions — transitions that are often required in problems with underlying constrained flow dynamics arising in problems ranging from traffic signal control to telecommunications bandwidth planning. In this paper, we present a novel SDP solution method for MDPs with LP transitions and continuous piecewise linear dynamics by introducing a novel, fully symbolic *argmax* operator. On three diverse domains, we show the first automated exact closed-form SDP solution to these challenging problems and the significant advantages of our SDP approach over discretized approximations.

1 Introduction

Many real-world stochastic planning problems naturally involve some component of continuous state such as resources, time, or spatial configurations. A specialized and important subclass of these problems require *linearly constrained optimization* in order to compute their transitions. For example, in traffic signal control problems, it is often customary to model flows of traffic according to a linear program (LP) that tries to advance all cars as far as possible subject to flow constraints and queue capacities [Lin and Wang, 2005]. In another example, telecommunications companies need to plan bandwidth purchases from providers to serve user demand at every time step [Adler *et al.*, 2011]; here, the company selects providers and capacities (edges in the graph). Then, a max flow problem [Cormen *et al.*, 2001] — cast as an LP — computes the maximum traffic routed through the network.

Existing exact solutions fall short of addressing MDPs with such LP transitions. For a discrete state space, it is possible in principle to compute an exact solution by enumerating all state-action pairs and solving an LP for each [Nicol *et al.*, 2013]. In discrete and continuous state MDPs (DC-MDPs), such an enumeration is obviously impossible for an infinite state space. While we might consider an approximate discretized state approach, this suffers from the well-known curse of dimensionality. I.e., for n continuous variables discretized into K segments, the Bellman backup would scale as $\mathcal{O}(K^{2n} \times |A|)$ where $|A|$ is the size of the action space; for $n = 2$ and $K = 100$, this would require 100 million LP evaluations! As we will demonstrate empirically, coarse discretizations lead to high approximation error, leaving us to search for alternative non-discretization solution techniques.

An alternative to discretization is to take a more symbolic approach. To this end, a recently developed class of methods known as symbolic dynamic programming (SDP) has provided exact closed-form solutions to DC-MDPs [Sanner *et al.*, 2011] and their extension to continuous action parameters [Zamani *et al.*, 2012]. The latter introduced the symbolic $\max$ operation over continuous parameters. The LP transition, however, specifically requires a symbolic multivariate $\arg\max$ operator that does not exist in the present literature.

This work aims to extend the symbolic $\max$ operator to the multivariate $\arg\max$ operator that can be leveraged for exact, closed-form SDP solutions of DC-MDPs with LP transitions. To be more concrete about our contributions, let us formalize a SIMPLE TRAFFIC MANAGEMENT problem[1]:

Example 1 (SIMPLE TRAFFIC MANAGEMENT)**.** *Figure 1 shows a road configuration where an Eastbound road (r_1) intersects a Southbound road (r_4). r_1 diverges into r_2 and r_3 after the intersection, and r_4 transitions to r_5. Each road is one-way, and the state of the environment consists of traffic volumes on the roads, i.e. $s = (q_1, q_2, q_3, q_4, q_5)$. For simplicity, we assume cars do not leave r_2, r_3 and r_5. The controller needs to decide which traffic approach to give a green light: r_1 ($a = 0$) or r_4 ($a = 1$). A positive reward is received for each vehicle transitioning through the intersection. The max capacities of roads r_1, r_2, r_3, r_4 and r_5 are 100, 120, 100, 100 and 100 respectively. The max number of cars leaving r_1 and r_4 are capped at 20 and 15, respectively.*

Notice that the Δq_i^* terms in the transition equations (Fig-

*Authors contributed equally.
†Affiliate to Vector Institute, Toronto, Canada.

[1]For purposes of concise exposition and explanation of the optimal value function and policy, this DC-MDP example uses continuous states and deterministic transitions; the empirical results will later discuss a range of DC-MDPs with stochastic transitions.

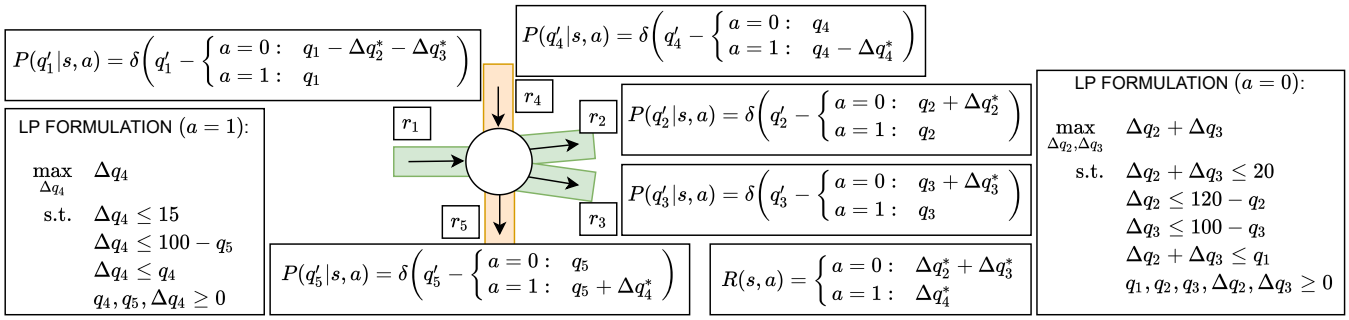

Figure 1: SIMPLE TRAFFIC MANAGEMENT. $\delta[\cdot]$ is the Dirac function, and Δq_i $(i = 2, 3, 4)$ is the volume of traffic entering or leaving r_i.

ure 1) are the $\arg\max$ solutions of the LP transition. Since the state transitions depend on the $\arg\max$ of the LP, we have to symbolically obtain closed-form optimal solutions of the LP to provide an *exact closed-form* solution to the overall MDP. Hence, as our primary technical contribution in this work, we develop a multivariate symbolic $\arg\max$ operator by breaking it into respective $\arg$ and $\max$ operations and showing that each can be computed in closed-form.

After preliminaries in Section 2, we first describe a univariate $\arg\max$ operation in Section 3, followed by the extension to the multivariate case. The $\arg\max$ operation facilitates a novel and exact closed-form SDP approach that we evaluate in Section 4 on problems from the Operations Research domain: SIMPLE TRAFFIC MANAGEMENT, RESERVOIR MANAGEMENT and BANDWIDTH OPTIMIZATION. We also compare our SDP approach with a discretized approximation showing that the SDP approach is more efficient and scalable without the error inherently induced by discretization.

2 Discrete and Continuous State MDPs

In a DC-MDP, vectors of variables $(\vec{b}, \vec{x}) = (b_1, \ldots, b_n, x_1, \ldots, x_m)$ represent a state. Each state variable b_i $(1 \le i \le n)$ is boolean s.t. $b_i \in \{0, 1\}$ and each x_j $(1 \le j \le m)$ is continuous s.t. $x_j \in [L_j, U_j]$ for L_j, $U_j \in \mathbb{R}$. We also assume a finite set of actions $A = \{a_1, \ldots, a_{|A|}\}$.

We define a DC-MDP by the following: (1) a transition model $P(\vec{b}', \vec{x}' | \vec{b}, \vec{x}, a)$ which specifies the probability of the next state $(\vec{b}', \vec{x}')$ conditioned on the current state and action a; (2) a reward function $R(\vec{b}, \vec{x}, a)$ which specifies the immediate reward obtained by taking action a in state $(\vec{b}, \vec{x})$; and (3) a discount factor $\gamma, 0 \le \gamma \le 1$. A policy π specifies the action $\pi(\vec{b}, \vec{x})$ to take in each state $(\vec{b}, \vec{x})$. Then, our goal is to find an optimal sequence of horizon-dependent policies $\Pi^* = (\pi^{*,1}, \ldots, \pi^{*,H})$ that maximizes $V^{\Pi^*}(s_0)$, the expected sum of discounted rewards over a horizon h; $H \ge 0$:

$$V^{\Pi^*}(s_0) = E_{\Pi^*}\left[\sum_{h=0}^{H} \gamma^h \cdot r^h \middle| \vec{b}_0, \vec{x}_0\right], \tag{1}$$

Here r^h is the reward obtained at horizon h while following Π^*, and we assume a starting state $s_0 = (\vec{b}_0, \vec{x}_0)$ at $h = 0$.

DC-MDPs as defined above are naturally factored in terms of state variables $(\vec{b}, \vec{x})$ [Boutilier *et al.*, 1999]; as such a transition structure can be exploited in the form of a dynamic Bayes net (DBN) [Dean and Kanazawa, 1989] where the individual conditional probabilities $P(b_i' | \cdots, a)$ and $P(x_j' | \cdots, a)$ condition on a subset of the current and next state variables. We disallow *synchronic arcs* (variables conditioning on each other in the same time step) within the binary $\vec{b}$ and continuous variables $\vec{x}$, but we allow synchronic arcs from $\vec{b}$ to $\vec{x}$ (note: these conditions enforce the directed acyclic graph requirements of DBNs). We write the joint transition model as

$$P(\vec{b}', \vec{x}' | \vec{b}, \vec{x}, a) = \prod_{i=1}^{n} P(b_i' | \vec{b}, \vec{x}, a) \prod_{j=1}^{m} P(x_j' | \vec{b}, \vec{b}', \vec{x}, a) \tag{2}$$

where $P(b_i' | \vec{b}, \vec{x}, a)$ may condition on a subset of $\vec{b}$ and $\vec{x}$, and $P(x_j' | \vec{b}, \vec{b}', \vec{x}, a)$ may condition on a subset of $\vec{b}, \vec{b}'$ and $\vec{x}$.

We refer to the conditional probabilities $P(b_i' | \vec{b}, \vec{x}, a)$ for *binary* variables b_i $(1 \le i \le n)$ as conditional probability functions (CPFs) — not tabular enumerations because these functions can condition on both discrete and continuous states. For *continuous* variables x_j $(1 \le j \le m)$, we represent the CPFs $P(x_j' | \vec{b}, \vec{b}', \vec{x}, a)$ as *piecewise linear equations* (PLEs) with three properties: (1) PLEs can only condition on the action, current state, and previous state variables; (2) PLEs are deterministic meaning that to be represented by probabilities, they must be encoded using Dirac $\delta[\cdot]$ functions; and (3) PLEs are piecewise linear, where the piecewise conditions may be arbitrary logical combinations of $\vec{b}$, $\vec{b}'$ and linear inequalities over $\vec{x}$. Example PLEs are in Figure 1 where the use of the $\delta[\cdot]$ function ensures that this is a conditional probability function that integrates to 1 over x_j'. PLEs allow modeling of continuous variable transitions as a mixture of δ functions used in continuous state MDP solutions [Meuleau *et al.*, 2009].

2.1 Dynamic Programming Solution

A continuous state generalization of *value iteration* [Bellman, 1957] is a dynamic programming algorithm for computing the optimal value function V^{Π^*} in Eq.1. It proceeds by constructing a series of h-stage-to-go value functions $V^h(\vec{b}, \vec{x})$. Specifically, initializing $V^0(\vec{b}, \vec{x})$ (e.g., to $V^0(\vec{b}, \vec{x}) = 0$) we repeatedly compute the following for $h \in \{0, \ldots, H-1\}$:

$$Q_a^{h+1}(\vec{b}, \vec{x}) = R(\vec{b}, \vec{x}, a) + \gamma \cdot \sum_{\vec{b}'} \int_{\vec{x}'} \left(\prod_{i=1}^{n} P(b_i' | \vec{b}, \vec{x}, a) \prod_{j=1}^{m} P(x_j' | \vec{b}, \vec{b}', \vec{x}, a) \right) V^h(\vec{b}', \vec{x}') d\vec{x}' \tag{3}$$

$$V^{h+1}(\vec{b}, \vec{x}) = \max_{a \in A} \left\{ Q_a^{h+1}(\vec{b}, \vec{x}) \right\} \tag{4}$$

2.2 Linear Program (LP) Transitions

In this paper, we consider *DC-MDPs with LP transitions*. In this set of problems, the state transition function over continuous state variables and the reward function depend on a p-dimensional intermediate vector $\vec{y}^*$, that is, $P(\vec{x}'|\vec{b},\vec{b}',\vec{x},a,\vec{y}^*)$ and $R(\vec{b},\vec{x},a,\vec{y}^*)$; whereas the transition over binary state variables remains as in Eq.2. Here, $\vec{y}^*$ is determined by the following LP given a fixed state and action $\vec{b},\vec{x},a$:

$$\vec{y}^* = \arg\max_{\vec{y}\in\mathcal{D}_y} f(\vec{b},\vec{x},a,\vec{y}) \text{ s.t. } \phi_i(\vec{b},\vec{x},a,\vec{y}),\ \forall i\in\{1,\ldots,k\} \tag{5}$$

where $\mathcal{D}_y \in \mathbb{R}^p$ is the domain of the continuous vector $\vec{y}$; ϕ_i $(1 \le i \le k)$ and f are respectively the linear constraints and the linear objective of the LP.

3 Symbolic Dynamic Programming

In this section, we review and extend the *symbolic dynamic programming* (SDP) framework of value iteration (VI) for DC-MDPs [Sanner *et al.*, 2011] to accommodate LP transitions. Continuous action [Zamani *et al.*, 2012; Zamani *et al.*, 2013] extensions are straightforward but omitted for clarity.

3.1 Case Operators and SDP

In SDP, we assume that all symbolic functions can be represented in *case form* [Boutilier *et al.*, 2001]:

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases} \tag{6}$$

Here ϕ_i are logical formulae defined over the state $s = (\vec{b},\vec{x})$ and the intermediate variable $\vec{y}$ from Eq.5, which can include arbitrary logical $(\wedge,\vee,\neg)$ combinations of (1) boolean variables and (2) *linear* inequalities $(\ge,>,\le,<)$ over continuous variables. Each ϕ_i will be disjoint from the other ϕ_j $(j \ne i)$; however, ϕ_i may not exhaustively cover the entire domain, so f may be undefined for some variable assignments. In this work, we assume the f_i are also linear in the continuous variables. Furthermore, we require f to be continuous (including no discontinuities at partition boundaries); case operations will then preserve this property.

Unary operations such as scalar multiplication $c \cdot f$ $(c \in \mathbb{R})$ or negation $-f$ on case statements are simply applied to each f_i $(1 \le i \le k)$. To perform a *binary operation* on two case statements, we take the cross-product of the logical partitions of each case statement and perform the operation on the resulting paired partitions. Letting each ϕ_i and ψ_j denote generic first-order formulae, we can perform the "cross-sum" $\oplus$ of two (unnamed) cases in the following manner:

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \begin{cases} \phi_1 \wedge \psi_1 : & f_1 + g_1 \\ \phi_1 \wedge \psi_2 : & f_1 + g_2 \\ \phi_2 \wedge \psi_1 : & f_2 + g_1 \\ \phi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

Likewise, we perform $\ominus$ and $\otimes$ by, respectively, subtracting or multiplying partition values to obtain the result. Some partitions resulting from case operators may be infeasible and removed.

Next, we define *symbolic case maximization* whose result is still piecewise linear:

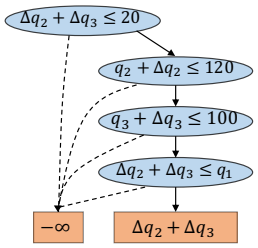

Figure 2: The LP in SIMPLE TRAFFIC MANAGEMENT when $a = 0$ (see Figure 1). Satisfying all the constraints leads to the objective function of the LP; otherwise it leads to the -∞ leaf node.

$$\text{casemax}\left(\begin{cases}\phi_1 : f_1 \\ \phi_2 : f_2\end{cases}, \begin{cases}\psi_1 : g_1 \\ \psi_2 : g_2\end{cases}\right) = \begin{cases} \phi_1 \wedge \psi_1 \wedge f_1 > g_1 : f_1 \\ \phi_1 \wedge \psi_1 \wedge f_1 \le g_1 : g_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 > g_2 : f_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 \le g_2 : g_2 \\ \vdots \qquad\qquad\qquad\quad \vdots \end{cases}$$

Another important operation is *symbolic substitution*. The operation takes a set σ of variables and their substitutions, e.g., $\sigma = \{x_1'/(x_1 + x_2), x_2'/(x_1 - x_2)\}$ where the LHS of $/$ represents the substitution variable and the RHS of $/$ is the expression that should be substituted in. Then, we write the substitution of a non-case function f_i with σ as $f_i\sigma$. We can also substitute into case partitions ϕ_j by applying σ to each inequality operand. Then, we can define the substitution operation for case statements in general:

$$f = \begin{cases} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{cases}, \quad f\sigma = \begin{cases} \phi_1\sigma : & f_1\sigma \\ \vdots & \vdots \\ \phi_k\sigma : & f_k\sigma \end{cases}$$

With the insight that integration over PLEs reduces to symbolic substitution [Sanner *et al.*, 2011], SDP simply executes value iteration in Eqs. 3 and 4 using the DC-MDP definition defined previously and the case operators defined above to yield a *closed-form, exact symbolic* derivation of the value function [Sanner *et al.*, 2011].

3.2 Multivariate Symbolic max Operator

Before we proceed to extend SDP to DC-MDPs with LP transitions, we first have to understand how to represent and manipulate an LP in the symbolic case notation. To this end, note that an LP (Eq. 5) has a natural *case* structure and can be represented even more compactly in XADD form [Sanner *et al.*, 2011] shown in Figure 2: *every leaf* shows a value f_i and *every path from root to leaf* is uniquely associated with one partition ϕ_i (a conjunction of decisions) and the value f_i. A solid line from a decision node to another node indicates the *true* branch of the parent node, while a dotted line shows the *false* branch. Each decision node corresponds to a constraint of the LP, and *false* branches lead to the $-\infty$ node. This is an intuitive way to specify LP infeasibility as we are solving a maximization problem. We denote the LP as $f(\vec{x},\vec{b},\vec{y})$.

Symbolically solving an LP amounts to symbolically maximizing out continuous variables from the case statement. Thus, we adopt the approach in [Zamani *et al.*, 2012] to make three observations: (1) multivariate maximization can be decomposed into a series of univariate maximizations; (2) to

maximize a case statement w.r.t. a variable y_l ($1 \le l \le p$), we need to perform a *symbolic* $\max_{y_l}$ on each case partition then combine results via casemax; (3) a partition ϕ_i defines lower (LB_i) and/or upper (UB_i) bounds over y_l. Since f_i is linear, the max of f_i has to occur at one of the bounds. I.e.,

$$\max_{y_l} \begin{cases} \phi_1: & f_1 \\ \vdots & \vdots \\ \phi_k: & f_k \end{cases} = \max_{y_l} \operatorname*{casemax}_{i=1,\ldots,k} \begin{cases} \phi_i: & f_i \\ \neg\phi_i: & -\infty \end{cases} \tag{7}$$

$$= \operatorname*{casemax}_{i=1,\ldots,k} \max_{y_l} \begin{cases} \phi_i: & f_i \\ \neg\phi_i: & -\infty \end{cases}$$

$$= \operatorname*{casemax}_{i=1,\ldots,k} \begin{cases} \phi_{i,ind}: \operatorname{casemax}\left(f_i\sigma_i^{lb}, f_i\sigma_i^{ub}\right) \\ \neg\phi_{i,ind}: -\infty \end{cases}$$

where $\phi_{i,ind}$ includes $LB_i \le UB_i$ and decisions in ϕ_i that are independent of y_l. $\sigma_i^{lb} = \{y_l / LB_i\}$ and $\sigma_i^{ub} = \{y_l / UB_i\}$ represent substitution of the lower and upper bounds to y_l in f_i, respectively. Note that LB_i and UB_i are also symbolic case functions (see the example in the following section).

3.3 Multivariate Symbolic argmax Operator

We now introduce the novel symbolic $\arg\max$ operation that will allow us to analytically solve for the solution of our LP transitions in Eq.5, substitute this symbolic LP solution to reduce our transitions to a standard DC-MDP, and then apply the known SDP solution for this reduced DC-MDP.

For each (ϕ_i, f_i), either $y_l = LB_i$ or UB_i will output the max in Eq.7. So, we annotate f_i accordingly to keep track of which y_l value has resulted in the max for the particular logical partition. Let $f(\vec{x}, \vec{b}, \vec{y}_{-l}) = \max_{y_l} f(\vec{x}, \vec{b}, \vec{y})$. If we assume — for the sake of simplicity — the values of $\operatorname{casemax}(f_i\sigma_i^{lb}, f_i\sigma_i^{ub})$ are different for all i ($1 \le i \le k$)[2], then every function value of $f(\vec{x}, \vec{b}, \vec{y}_{-l})$ can be uniquely traced back to an annotated y_l as follows:

$$f(\vec{x}, \vec{b}, \vec{y}_{-l}) = \begin{cases} \psi_1: & f_{(1,-y_l)}, o_1 \\ \vdots & \vdots \\ \psi_{k'}: & f_{(k',-y_l)}, o_{k'} \end{cases} \tag{8}$$

Here, each partition ψ_j ($1 \le j \le k'$) has an associated function value $f_{(j,-y_l)}$ and annotation o_j.

The trick is to split $\arg\max$ into arg and max. Once we obtain $f(\vec{x}, \vec{b}, \vec{y}_{-l})$ from max, we perform arg on it to retrieve the annotations and to build a symbolic case statement representing $\arg\max_{y_l} f(\vec{x}, \vec{b}, \vec{y})$ as in Eq.9.

$$y_l^* = \arg_{y_l} f(\vec{x}, \vec{b}, \vec{y}_{-l}) = \begin{cases} \psi_1: & o_1 \\ \vdots & \vdots \\ \psi_{k'}: & o_{k'} \end{cases} \tag{9}$$

The arg operation removes $f_{(j,-y_l)}$ from ψ_j and leaves o_j only. Note that o_j can be a case function, in which case the partitions in o_j are combined with ψ_j and simplified.

For the multivariate $\arg\max$, let's assume we maximize $y_1, \ldots, y_p$ in this order, namely $\max_{\vec{y}} f(\vec{x}, \vec{b}, \vec{y}) = \max_{y_p} \cdots \max_{y_1} f(\vec{x}, \vec{b}, \vec{y})$. Then, we repeat the following steps p times:

1. Compute the innermost maximization $f(\vec{x}, \vec{b}, \vec{y}_{-(1:l)}) = \max_{y_l} f(\vec{x}, \vec{b}, \vec{y}_{-(1:l-1)})$;
2. Perform arg operation on $f(\vec{x}, \vec{b}, \vec{y}_{-(1:l)})$ to get y_l^*.

This way, we obtain y_l^* for all $l = 1, \ldots, p$ in case form. Note that y_l^* will still contain 'outer' variables $y_{l'}$ ($l+1 \le l' \le p$) in its case statement. However, in order to get a closed-form symbolic solution to MDPs with LP transitions, we need to be able to express each y_l^* only with $\vec{b}, \vec{b'}, \vec{x}$.

Removing outer variables from the case statement of y_l^* involves repeated substitution of $y_{l'}^*$. Observe that y_p^* contains only $\vec{b}, \vec{b'}, \vec{x}$, while y_{p-1}^* additionally has y_p. Therefore by substituting y_p^* into y_{p-1}^*, it now contains $\vec{b}, \vec{b'}, \vec{x}$ only. For y_{p-2}^*, we need to substitute both y_p^* and the modified y_{p-1}^*. This process should be repeated for all y_l ($1 \le l \le p-1$).

However, we *cannot* substitute y_{l+1}^* into y_l^* via the symbolic substitution since y_{l+1}^* is also a case statement. Instead, we merge two case statements as described in Example 2.

Example 2. *Define $g(x, y)$ and $y = h(x)$ as follows:*

$$g(x,y) = \begin{cases} \nu_1: & x+y \\ \neg\nu_1: & x-y \end{cases}, \quad y = h(x) = \begin{cases} \nu_2: & 3x \\ \neg\nu_2: & 2x \end{cases}$$

When we substitute $y = h(x)$ into $g(x, y)$, we get

$$g(x, h(x)) = \begin{cases} \nu_2 \wedge \nu_1: & x + 3x = 4x \\ \nu_2 \wedge \neg\nu_1: & x - 3x = -2x \\ \neg\nu_2 \wedge \nu_1: & x + 2x = 3x \\ \neg\nu_2 \wedge \neg\nu_1: & x - 2x = -x \end{cases}$$

So, we take the cross-product of the logical partitions and substitute an appropriate value of y into $g(x, y)$.

We summarize in four major steps how to get a closed-form $\arg\max$ solution to an LP using the $a = 0$ case in Example 1. We denote the LP in case form as $f(q_1, q_2, q_3, \Delta q_2, \Delta q_3)$.

1. *Univariate maximization with annotation*. In Figure 2, the feasible partition consists of $\{(\Delta q_2 + \Delta q_3 \le 20) \wedge (q_2 + \Delta q_2 \le 120) \wedge (q_3 + \Delta q_3 \le 100) \wedge (\Delta q_2 + \Delta q_3 \le q_1)\}$. Call this partition ϕ_i, then we have $\phi_i : f_i = \Delta q_2 + \Delta q_3$ and $\phi_j : -\infty$ ($j \ne i$). When we maximize out Δq_2, decisions in ϕ_i and a domain bound $\Delta q_2 \in [0, 20]$ define UB and LB of Δq_2. For example, $UB = \operatorname{casemin}(20, 20 - \Delta q_3, 120 - q_2, q_1 - \Delta q_3)$, which becomes the following case statement:

$$\begin{cases} (q_2 - \Delta q_3 \ge 100) \wedge (q_1 + q_2 - \Delta q_3 \ge 120) : 120 - q_2 \\ (q_2 - \Delta q_3 \ge 100) \wedge (q_1 + q_2 - \Delta q_3 < 120) : q_1 - \Delta q_3 \\ (q_2 - \Delta q_3 < 100) \wedge (q_1 \le 20) : q_1 - \Delta q_3 \\ (q_2 - \Delta q_3 < 100) \wedge (q_1 > 20) : 20 - \Delta q_3 \end{cases}$$

The maximum of f_i occurs either at $\Delta q_2 = UB$ or LB. Hence, we take casemax to determine the max, that is,

$$\max_{\Delta q_2} f_i = \operatorname{casemax}(f_i\{\Delta q_2 / LB\}, f_i\{\Delta q_2 / UB\})$$

Then, we incorporate independent decisions: $LB \le UB$ and $q_3 + \Delta q_3 \le 100$. We refer readers to [Zamani *et al.*, 2012] for the step-by-step walk-through. The difference is that we annotate values with either LB or UB.

2. *Annotation preserving* casemax. Once continuous max is done, we take casemax and obtain $f(q_1, q_2, q_3, \Delta q_3)$. This step preserves the annotations from Step 1.

[2] When the two arguments are the same for some i, we can have multiple optimal solutions. As often done by off-the-shelf LP solvers, we simply select one optimal solution in this case.

3. *Application of* arg *operator*. We then compute $\Delta q_2^* = \arg_{\Delta q_2} f(q_1, q_2, q_3, \Delta q_3)$ (Eq.9).
   We repeat Step 1-3 on $f(q_1, q_2, q_3, \Delta q_3)$ to maximize over Δq_3, producing $f(q_1, q_2, q_3)$. Then, Δq_3^* is a case function of q_1, q_2, q_3, while Δq_2^* is a case function of $q_1, q_2, q_3, \Delta q_3$.
4. *Case substitutions*. Finally, we substitute the case statement Δq_3^* into Δq_2^*. Both are now functions of q_1, q_2, q_3 and are used in defining the state transition function.

3.4 SDP with LP Transitions

Having introduced the symbolic $\arg\max$ operator, we can symbolically solve Eq.5, resulting in a case function $u(\vec{b}, \vec{x}, a)$ in Eq.10. This step only has to occur once. Subsequently, we define a symbolic substitution $\sigma_{\vec{y}^*}$ to substitute the solution into the transition and reward functions (as in Eq.11-13).

$$\vec{y}^* = u(\vec{b}, \vec{x}, a) \tag{10}$$

$$\sigma_{\vec{y}^*} = \{\vec{y}/u(\vec{b}, \vec{x}, a)\} \tag{11}$$

$$P(\vec{x}'|\vec{b}', \vec{b}, \vec{x}, a) = P(\vec{x}'|\vec{b}', \vec{b}, \vec{x}, a, \vec{y})\ \sigma_{\vec{y}^*} \tag{12}$$

$$R(\vec{b}, \vec{x}, a) = R(\vec{b}, \vec{x}, a, \vec{y})\ \sigma_{\vec{y}^*} \tag{13}$$

Overall, we have reduced a DC-MDP with LP transitions to a standard DC-MDP, which can be solved in closed-form by leveraging its SDP solution (Section 3.1).

4 Empirical Results

We now apply our methodology for exact SDP with LP transitions to SIMPLE TRAFFIC MANAGEMENT, RESERVOIR MANAGEMENT, and BANDWIDTH OPTIMIZATION.[3]

We also compare to approaches that discretize the state space of the DC-MDP as discussed in Section 1. While LP transitions may suggest some form of mixed integer linear program (MILP) solution, we note that existing MILP-based solutions for finite horizon DC-MDPs (with no LP transitions) [A. Raghavan *et al.*, 2017] provide no error guarantees and only solve for a known starting state, whereas our SDP approach provides an *exact* value function for *all* states.

SIMPLE TRAFFIC MANAGEMENT: Figure 3 (left) shows the symbolic and approximate value function, $V^{13}(q_1, q_4)$ for $H = 13$. We observe that locations in the state space where all available vehicles in roads r_1 and r_4 can be accommodated within destination lanes show a much higher slope in the value function. These are regions where q_1 and q_4 are closer to their lower domain limits. As q_1 reaches a threshold value, all available spaces in q_2 and q_3 have filled, and there is no consequent gain in value with increase in q_1. A similar observation can also be made for q_4. The value function maxes out when all destination lanes have reached capacity.

The approximate value function can only be obtained for a maximum discretization of 14 before it's no longer feasible to solve for a finer granularity due to a memory explosion as evident in Figure 4. This shows the accuracy and computational limitations of approximate discretization solutions.

RESERVOIR MANAGEMENT: In a Reservoir management problem [Mahootchi, 2009; Yeh, 1985], we examine a 2-reservoir system with one upstream reservoir (r_1) having no electricity generation facility and one downstream reservoir (r_2) with a hydroelectric power generator. The decision is whether to allow or block the flow of water between the two reservoirs. Water is always discharged from r_2 within its operating bounds to produce maximum electricity according to an LP optimization. Water evaporates from a reservoir with the amount proportional to the water level l_i. When the water level is close to its lower limit, we allow a small amount of water to flow into a reservoir to make the LP in Eq.14 feasible. Finally, the stochastic rainfall is described in Eq.15.

The state vector consists of two continuous variables and one binary variable, i.e. $s = (l_1, l_2, r)$. The associated transition LP and the resulting state transitions are as follows:

$$\begin{aligned} \max_{q_1, q_2} \quad & q_2 \\ \text{s.t.} \quad & 1000 \le 0.98 \cdot l_1 - q_1 + 200 \cdot r \le 3000 \\ & 700 \le 0.98 \cdot l_2 + q_1 - q_2 + 200 \cdot r \le 1500 \\ & 0 \le q_1 \le 250 \cdot a,\ \ 0 \le q_2 \le 300 \end{aligned} \tag{14}$$

$$\begin{aligned} & P(r' = 1|r = i) = 0.4,\ i \in \{0, 1\} \text{ (no-rain/rain)} \\ & P(l_1'|l_1, l_2, r) = \delta\Big(l_1' - \big\{0.98 \cdot l_1 - q_1^* + 200 \cdot r\big\}\Big) \\ & P(l_2'|l_1, l_2, r) = \delta\Big(l_2' - \big\{0.98 \cdot l_2 + q_1^* - q_2^* + 200 \cdot r\big\}\Big) \\ & R(l_1, l_2, r, a) = q_2^* \end{aligned} \tag{15}$$

where the water flow is blocked when $a = 0$, and allowed within a limit when $a = 1$. q_1^*, q_2^* are $\arg\max$ of Eq.14, and we can see that q_1^*, q_2^* are case functions of l_1, l_2, r and a.

Figure 3 (middle) shows the value function for the RESERVOIR MANAGEMENT problem for $H = 4$, $V^4(l_1, l_2, r = False)$. We can see that regions where q_1 and q_4 are close to their domain upper limits correspond to areas with the highest values. The symbolic and the approximate solution have similar surface contours but approximation error still creeps in, especially in regions where l_1 is high and l_2 is low.

BANDWIDTH OPTIMIZATION: In a bandwidth optimization problem [Adler *et al.*, 2011], a service provider is given a network traffic demand from users, and the company needs to purchase bandwidths from ISPs (Internet service providers). In each time period, the company subscribes to a different set of links (with differing costs) offered by different ISPs to meet the current demand and residual demand from previous time steps while minimizing operating costs.

Now, consider a simple network with 5 links ($L = \{o1, o2, 12, 1e, 2e\}$) and 3 paths ($o \to 1 \to e$, $o \to 1 \to 2 \to e$, $o \to 2 \to e$) from the origin (o) to the edge server (e). The company has 7 buying options ($|A| = 7$), each of which corresponds to purchasing a different combination of paths. We firstly define the associated max flow problem:

$$\max_{x_{ij}, ij \in L} \quad x_{o1} + x_{o2} \tag{16}$$

$$\text{s.t.} \quad x_{o1} = x_{12} + x_{1e}, \tag{17}$$

$$x_{2e} = x_{o2} + x_{12}$$

$$0 \le x_{ij} \le b_{ij} \cdot K_{ij}, \qquad \forall ij \in L \tag{18}$$

where x_{ij} is the max flow on link ij and K_{ij} is its capacity. Eq.17 enforces the balance in flow at every node. Also, given a buying option, $b_{ij} = 1$ for the links on the paths. Otherwise, $b_{ij} = 0$, thereby restricting x_{ij} to 0.

[3] Implementations can be found at `https://github.com/jihwan-jeong/xadd-inference/`

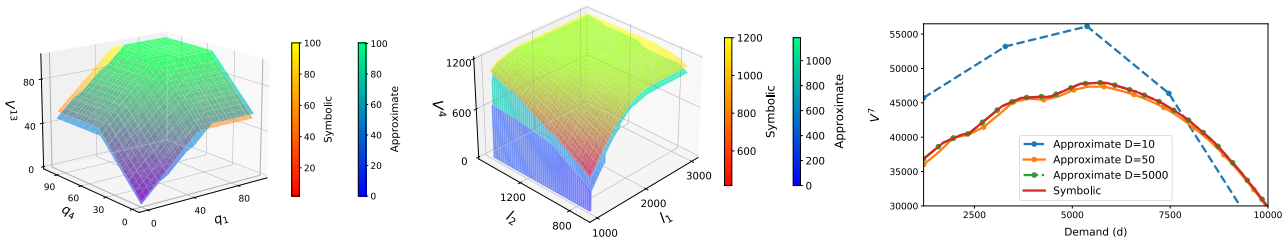

Figure 3: SIMPLE TRAFFIC MANAGEMENT*(left)*: Value function $V^{13}(q_1, q_4)$ when q_2, q_3 and q_5 are 85, 85 and 50 respectively. The approximate solution was obtained with the maximum discretization (14) for all state variables. RESERVOIR MANAGEMENT*(middle)*: Symbolic value function $V^4(l_1, l_2, r = 0)$ alongside approximate value function with the discretization of 500. Both values have similar surfaces except when l_1 nears its lower domain limit due to LP infeasibility. In both examples, differences in the surfaces arise due to the approximation error of the discretized solution. BANDWIDTH OPTIMIZATION*(right)*: Symbolic and approximate value functions $V^{10}(d, l = 0)$ with different discretizations D. The values match with high discretizations, but they differ when using coarser discretization.

We assume the following pricing policy: the provider pays (i) m_{ij} for purchasing the bandwidth on link ij and (ii) a fee as per the actual traffic routed on the link ($c \cdot x_{ij}$ with unit cost c). Then the states, actions and rewards are defined as below:

- State $s = (d, l)$ where $d \in \mathbb{R}^+$ is the remaining demand and $l \in \{0, 1\}$ (low, high) is the level of new demand.
- Action $\{a_1, \ldots, a_7\}$.
- Reward: $R(d, l, a) = r \cdot x_d - p \cdot (d - x_d) - c \cdot x_d - \sum_{i,j \in \mathbb{N}} m_{ij} \cdot b_{ij}$. Here, $x_d = \min(d, x^*_{o1} + x^*_{o2})$ is the actual routed traffic to meet the demand d. r and p are the unit revenue for the traffic delivered and the unit penalty for unmet demands, respectively.

The state transition is defined as the following, where $d_{new} = 2500$ when $l = high$ and $d_{new} = 1200$ otherwise:

$$d' = \begin{cases} (d > x^*_{o1} + x^*_{o2}): & d - (x^*_{o1} + x^*_{o2}) + d_{new} \\ (d \leq x^*_{o1} + x^*_{o2}): & d_{new}. \end{cases}$$

$$P(l' = \alpha | l = \alpha) = 0.7, \quad \forall \alpha \in \{high, low\}$$

In our experiments, we use $r, p, m_{o1}, m_{o2}, m_{12}, m_{1e}, m_{2e}, c$ = 6, 2, 1000, 800, 600, 750, 800, 1.3, respectively. Also, the capacity of each link is $K_{o1}, K_{o2}, K_{12}, K_{1e}, K_{2e}$ = 2100, 1800, 1000, 1500, 1700.

In Figure 3 (right), we can clearly see the value function is piecewise. This suggests that in each interval there is a different optimal policy. However, the approximate solution misses some intervals when the discretization size is small.

Time and Space: Note that our solution presolves for the closed-form $\vec{y}^*$ in Eq.5 and substitutes the analytical solution into transition and reward functions before starting VI. This way, we only need to solve the symbolic LP argmax once. Figure 4 shows the time and space complexity of approximate solutions along with that of the symbolic solutions. As before, we see an exponential blowup for the approximate discretized solution as the discretization granularity increases. We also note that as the number of state variables vary between different problems, the discretization-based approach suffers from the *curse of dimensionality* with respect to memory requirements, while in stark contrast, *the SDP solution size is independent of the number of state variables*.

For problems with a long horizon, the SDP approach may need to compute larger and more complex symbolic value functions (cf. Figure 6 in Appendix), depending on the characteristics of the problem. While approximate discretization methods demonstrate some computational advantages in this setting, their limited accuracy may make it harder to use the approximate solution when quality guarantees are required.

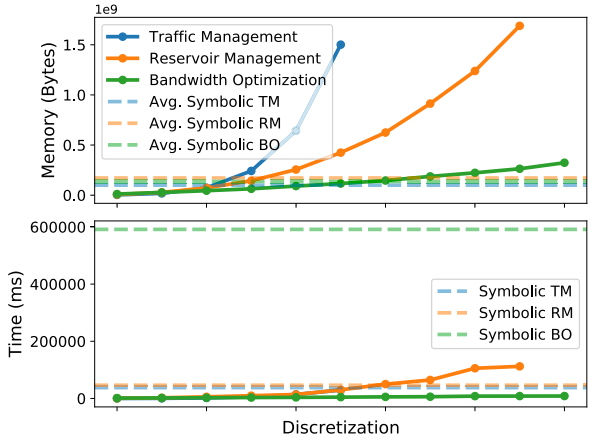

Figure 4: Memory and time vs. discretization. We observe exponential increases in memory as discretization gets finer, with the rates of increase being determined by the number of variables. Horizontal lines show average memory (over iterations) and total time required by the symbolic solutions, which are generally found to be much lower in memory use than their discretized counterpart.

5 Related Work and Concluding Remarks

Building on a long line of work in solving continuous state MDPs [Boyan and Littman, 2001; Feng *et al.*, 2004; Li and Littman, 2005] culminating in expressive symbolic dynamic programming (SDP) approaches [Sanner *et al.*, 2011], we proposed a novel SDP method for exactly solving DC-MDPs with LP transitions. Specifically, we extended the symbolic $\max$ operator [Zamani *et al.*, 2012] to an $\arg\max$ operator via annotation augmentations to the case notation and its operations. Overall, this work provides the *first exact solution method for DC-MDPs with LP transitions*, and hence opens up a new class of challenging MDPs for research exploration.

For future work, one could leverage initial state focused techniques [Meuleau *et al.*, 2009; Vianna *et al.*, 2015] to reduce solution scope and size. Recent advances from Weighted Model Integration (WMI) [Kolb *et al.*, 2018] may speed up computation of the symbolic $\max$ and $\arg\max$. Finally, methods for approximately bounding optimal value function structure [St-Aubin *et al.*, 2000; Vianna *et al.*, 2013] or further afield [Remi Munos, 2002; Kveton *et al.*, 2006; Marecki *et al.*, 2007] may improve scalability.

References

[A. Raghavan *et al.*, 2017] A. Raghavan, S. Sanner, P. Tadepalli, A. Fern, and R. Khardon. Hindsight optimization for hybrid state and action mdps. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, USA, 2017.

[Adler *et al.*, 2011] Micah Adler, Ramesh Sitaraman, and Harish Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55:4007–4020, 12 2011.

[Bellman, 1957] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Boutilier *et al.*, 1999] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[Boutilier *et al.*, 2001] Craig Boutilier, Ray Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *IJCAI-01*, pages 690–697, Seattle, 2001.

[Boyan and Littman, 2001] Justin Boyan and Michael Littman. Exact solutions to time-dependent MDPs. In *Advances in Neural Information Processing Systems NIPS-00*, pages 1026–1032, 2001.

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

[Dean and Kanazawa, 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Feng *et al.*, 2004] Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, and Richard Washington. Dynamic programming for structured continuous markov decision problems. In *Uncertainty in Artificial Intelligence (UAI-04)*, pages 154–161, 2004.

[Kolb *et al.*, 2018] Samuel Kolb, Martin Mladenov, Scott Sanner, Vaishak Belle, and Kristian Kersting. Efficient symbolic integration for probabilistic inference. In *IJCAI*, pages 5031–5037, 2018.

[Kveton *et al.*, 2006] Branislav Kveton, Milos Hauskrecht, and Carlos Guestrin. Solving factored mdps with hybrid state and action variables. *Journal Artificial Intelligence Research (JAIR)*, 27:153–201, 2006.

[Li and Littman, 2005] Lihong Li and Michael L. Littman. Lazy approximation for solving continuous finite-horizon mdps. In *National Conference on Artificial Intelligence AAAI-05*, pages 1175–1180, 2005.

[Lin and Wang, 2005] Wei-Hua Lin and Chenghong Wang. An enhanced 0–1 mixed-integer lp formulation for traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 5:238 – 245, 01 2005.

[Mahootchi, 2009] Masoud Mahootchi. *Storage System Management Using Reinforcement Learning Techniques and Nonlinear Models*. PhD thesis, University of Waterloo,Canada, 2009.

[Marecki *et al.*, 2007] Janusz Marecki, Sven Koenig, and Milind Tambe. A fast analytical algorithm for solving markov decision processes with real-valued resources. In *International Conference on Uncertainty in Artificial Intelligence IJCAI*, pages 2536–2541, 2007.

[Meuleau *et al.*, 2009] Nicolas Meuleau, Emmanuel Benazera, Ronen I. Brafman, Eric A. Hansen, and Mausam. A heuristic search approach to planning with continuous resources in stochastic domains. *Journal Artificial Intelligence Research (JAIR)*, 34:27–59, 2009.

[Nicol *et al.*, 2013] Sam Nicol, Olivier Buffet, Takuya Iwamura, and Iadine Chadès. Adaptive management of migratory birds under sea level rise. In *International Conference on Uncertainty in Artificial Intelligence IJCAI*, pages 2955–2957, 08 2013.

[Remi Munos, 2002] Andrew Moore Remi Munos. Variable resolution discretization in optimal control. *Machine Learning*, 49, 2–3:291–323, 2002.

[Sanner *et al.*, 2011] Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes de Barros. Symbolic dynamic programming for discrete and continuous state mdps. In *Proceedings of the 27th Conference on Uncertainty in AI (UAI-2011)*, Barcelona, 2011.

[St-Aubin *et al.*, 2000] Robert St-Aubin, Jesse Hoey, and Craig Boutilier. APRICODD: Approximate policy construction using decision diagrams. In *NIPS-2000*, pages 1089–1095, Denver, 2000.

[Vianna *et al.*, 2013] L. G. Rocha Vianna, S. Sanner, and L. N. de Barros. Bounded approximate symbolic dynamic programming for hybrid MDPs. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13)*, Bellevue, USA, 2013.

[Vianna *et al.*, 2015] L. G. Rocha Vianna, L. N. de Barros, and S. Sanner. Real-time symbolic dynamic programming for hybrid MDPs. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, Austin, USA, 2015.

[Yeh, 1985] William G Yeh. Reservoir management and operations models: A state-of-the-art review. *Water Resources research*, 21,12:1797–1818, 1985.

[Zamani *et al.*, 2012] Z. Zamani, S. Sanner, and C. Fang. Symbolic dynamic programming for continuous state and action mdps. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada, 2012.

[Zamani *et al.*, 2013] Z. Zamani, S. Sanner, K. Delgado, and L. Barros. Robust optimization for hybrid mdps with state-dependent noise. In *IJCAI*, 2013.

Appendix

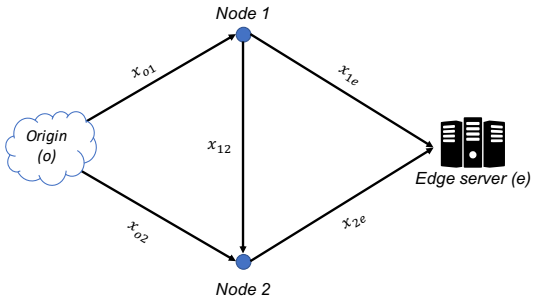

Figure 5: The network used in the BANDWIDTH OPTIMIZATION problem.

Details of BANDWIDTH OPTIMIZATION Figure 5 represents the network of the BANDWIDTH OPTIMIZATION problem, depicting how the origin, edge server and intermediate nodes are connected through links. There are 3 viable paths (path1: $o1 \rightarrow 1e$, path2: $o1 \rightarrow 12 \rightarrow 2e$, path3: $o2 \rightarrow 2e$) that the service provider can purchase and the number of buying options (7) relates to the different ways in which these paths can be purchased. That is, the action space is $A = \Big\{\{\text{path1}\}, \{\text{path2}\}, \{\text{path3}\}, \{\text{path1}, \text{path2}\}, \{\text{path2}, \text{path3}\}, \{\text{path1}, \text{path3}\}, \{\text{path1}, \text{path2}, \text{path3}\}\Big\}$.

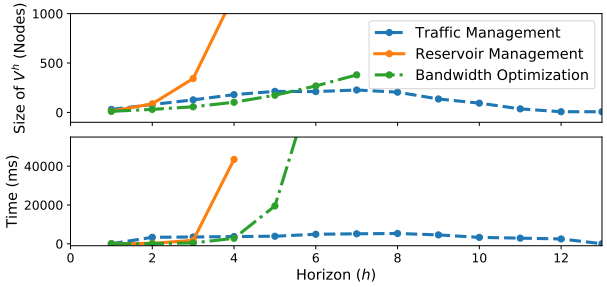

Figure 6: Space and elapsed time vs. horizon for SIMPLE TRAFFIC MANAGEMENT, RESERVOIR MANAGEMENT and BANDWIDTH OPTIMIZATION. RESERVOIR MANAGEMENT solution sees large increases in the number of nodes with increasing horizons, while BANDWIDTH OPTIMIZATION solution has the highest values of time elapsed in each iteration. SIMPLE TRAFFIC MANAGEMENT needs much fewer resources and time, all the while reaching convergence in the given iterations.

Space and time vs. horizon for the symbolic approach In Figure 6, we show how the symbolic solution scales with horizon for different classes of problems. We can see that the number of *horizons* over which the problem can be solved is dependent on the problem formulation itself. The SIMPLE TRAFFIC MANAGEMENT problem has more continuous variables, but it is the RESERVOIR MANAGEMENT problem that shows a comparative blow up in space and time. The fact that the latter contains a stochastic binary variable may complicate computations. BANDWIDTH OPTIMIZATION problem on the other hand, only experiences blowups for the time elapsed per iteration all the while staying steady in terms of memory usage. This figure in conjunction with Figure 4 shows us the relative efficiency of the symbolic solution in terms of memory usage.