Proceedings of the 2021 IISE Annual Conference
A. Ghate, K. Krishnaiyer, K. Paynabar, eds.

Maximum-posterior Evaluation for Partially Observable Multistage Stochastic Programming

Murwan Siddig, Yongjia Song, and Amin Khademi
Clemson University
Clemson, SC, USA

Abstract

This paper introduces a computationally practical approach for solving a class of partially observable multistage stochastic programming problems. In this class of problems, the underlying stochastic process is assumed to follow a hidden Markov chain. Recent advances in the literature introduce a derivative of the stochastic dual dynamic programming (SDDP) method, known as the partially observable SDDP algorithm. This approach, however, introduces a belief-state vector to the stochastic programming formulation, which increases the computation significantly. Instead, we solve the problem assuming that this Markov chain is fully observable. However, when evaluating the resulting policy, we use a Bayesian update of the belief vector and use the state's decisions that have the highest posterior. Our numerical experiments on a hydrothermal planning problem show the practicality of this approach.

Keywords
Stochastic dual dynamic programming, Multistage stochastic linear programs, Partially observable Markov chain.

1. Introduction
Multistage stochastic programming (MSP) is a class of problems for *sequential* decision-making under *uncertainty*. In an MSP problem, the goal of the decision-maker (DM) is to minimize the total cost over a planning horizon consisting of T periods (decision epochs). To do this, at each point in time $t = 1, \ldots, T$, the DM observes the state of a system and influences its behavior by choosing the actions which minimize the expected total cost, which consists of the immediate cost and the expected future cost. MSP models have a wide range of applications in many areas such as energy [3], transportation [6], and finance [5], among others. For instance, in a hydrothermal power generation system, a large-scale network of facilities can be deployed over a long horizon to produce energy by circulating H_2O fluids (water) through the network. To meet the target demand, at each point in time, the DM observes the amount of water in the network and then decides on an operational decision for distributing the available water through the network. This produces two results: (i) the DM pays an immediate cost for operating the network, and (ii) the system evolves to a new state at a subsequent point in time, where there may be a different set of operational decisions to choose from. The goal of the DM is, then, to define an operation strategy (decision policy) that minimizes the overall production cost in expectation. This problem can be modeled as an MSP problem because of the uncertainty in the problem data, such as future water inflows (e.g., amount of rainfall), demand, production costs, and equipment availability.

A typical approach to tackle these problems is to approximate the underlying stochastic process using a scenario tree that is associated with a probability distribution $\hat{D}$ known as the *nominal* distribution. One major drawback to this approach, however, is that this nominal distribution is typically constructed using statistical predictive methods, which often inherit a considerable amount of estimation errors and forecast misspecifications. Such inaccuracies might lead to inconsistency issues and poor out-of-sample performances. This ultimately weakens the interventions that ought to be carried out by the DM, as they are no longer based on reliable data but rather presumptions.

Many approaches have been proposed in the literature to hedge against such inconsistency issues and poor out-of-sample performance. One approach is the so-called distributionally robust optimization (DRO) (see, e.g., [2, 8]). In the DRO setting, the idea is, instead of using a single nominal distribution $\hat{D}$, the DM uses an *ambiguity* set $\mathcal{D}$ of probability distributions and optimizes with respect to the *worst-case* distribution over this set. One critique to this approach in the MSP setting is that the ambiguity set does not evolve as new information arrives. Another recent advancement in the literature is the *partially observable* MSP model introduced in [4]. In this framework, the idea is

to define an *ambiguity* set using a *partially* observable Markov chain (MC), where each state in this MC represents a candidate distribution. Initially, the DM is unsure which of the candidate distributions (states) is most similar to the *true* underlying distribution. However, as time evolves, the DM observes new realizations of the stochastic process and uses a Bayesian update to improve their *belief* about which one of the candidates most accurately reflects the underlying distribution. This approach is analogous to the one developed in the Markov decision process (MDP) literature, where it is referred to as *contextual* MDP (CMDP) [7].

To solve the resulting problem, the authors in [4] present a modeling framework and an algorithmic extension to a popular approach for solving MSP problems known as stochastic dual dynamic programming (SDDP) [9]. This analogous algorithmic procedure is referred to as *partially observable* SDDP (POSDDP). This modeling framework, however, introduces a belief-state vector in the formulation, which makes the computational complexity scale quite poorly – especially with the number of candidate distributions within the ambiguity set. To mitigate these computation burdens, we propose a practical approach that defines a separate policy for every state in the MC and then chooses the state which has the highest posterior probability to implement the decisions. Given that this approach does not leverage all information provided by the belief state (but rather uses the state with the highest posterior probability), the resulting policy is typically suboptimal. Nevertheless, our numerical results show that this suboptimality gap is small relative to the significant reduction in the computational time. The rest of this paper is organized as follows. In Section 2, we introduce the problem formulation. In Section 3, we describe our proposed method. In Section 4, we provide some preliminary results for the proposed approach and compare it to the POSDDP algorithm. Finally, in Section 5, we conclude with some final remarks.

2. Markovian Multistage Stochastic Programming

In this section, we discuss some necessary assumptions, introduce the mathematical notation, the fully observable MSP formulation, and compare it to its partially observable counterpart.

Consider solving the following MSP problem,

$$\min_{x_1 \in \mathcal{X}_1(x_0,\xi_1)} f_1(x_1,\xi_1) + \mathbb{E}_{|\xi_{[1]}}\left[\min_{x_2 \in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \mathbb{E}_{|\xi_{[2]}}\left[\cdots + \mathbb{E}_{|\xi_{[T-1]}}\left[\min_{x_T \in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)\right]\right]\right]. \tag{1}$$

Here, $\xi_{[t]} := (\xi_1,\ldots,\xi_t)$ is a random vector, with a known probability distribution (probability measure) D_t supported on a set $\Xi_t \subset \mathbb{R}^n$, and the set Ξ_t is equipped with its Borel sigma-algebra $\mathcal{F}_t$. To facilitate a computationally tractable formulation, we make the following assumptions:

Assumption 1. *$f_t(x_t,\xi_t)$ is a linear function in x_t given ξ_t, $\forall t = 1,\ldots,T$.*

Assumption 2. *$\mathcal{X}_t(x_{t-1},\xi_t) := \{x_t \in \mathbb{R}^n \mid \tilde{A}_t x_t + \tilde{B}_t x_{t-1} = \tilde{b}_t\}$, i.e., $\xi_t = (\tilde{A}_t,\tilde{B}_t,\tilde{b}_t)$, $\forall t = 1,\ldots,T$*

Assumption 3. *$\forall\, x_{t-1} \in \mathcal{X}_{t-1}$ and $\xi_t \in \Xi_t$, $\mathcal{X}_t(x_{t-1},\xi_t) \neq \emptyset$, $\forall t = 1,\ldots,T$.*

Assumption 4. *The candidate distributions $D_t \in \mathcal{D}$ follow a MC with a finite set of Markovian states $\{k\}_{k\in\mathcal{D}}$ and transition probabilities $p_j(k) := \mathbb{P}(D_{t+1} = k | D_t = j)$. Now the probability of realization ξ_{t+1} happening, given the Markovian state $D_{t+1} = k$, is given by $q_k(\xi_{t+1}) = \mathbb{P}(\xi_{t+1} \in \Xi_{t+1} | D_{t+1} = k)$.*

Assumptions 1 and 2 imply that problem (1) is a multistage stochastic linear program (MSLP). Assumption 3 is the standard relative complete recourse assumption. Assumption 4 models the evolution of the stochastic process, and it implies *conditional* stage-wise independence; meaning that ξ_t depends only on t and the current state of the MC $\{D_t : t = 1,\ldots,T\}$. In other words, we have that: $\mathbb{P}(\xi_t \in \Xi_t, \xi_{t+1} \in \Xi_{t+1} | D_{t+1} = k, D_t = j) = \mathbb{P}(\xi_t \in \Xi_t | D_t = j) \cdot \mathbb{P}(\xi_{t+1} \in \Xi_{t+1} | D_{t+1} = k)$.

2.1 Fully Observable Multistage Stochastic Programming

In its present formulation, problem (1) has a nested form. A common approach to proceed with the computation is to use a dynamic programming (DP) formulation [1]. Under the conditional stage-wise independence assertion implied by Assumption 4, the following *fully* observable DP formulation can be defined for each state $j \in \mathcal{D}$ in the MC, represented by a *cost-to-go function*:

$$Q_t^j(x_{t-1},\xi_t) := \begin{cases} \min\limits_{x_t} & f_t(x_t,\xi_t) + \sum\limits_{k\in\mathcal{D}} p_j(k)\cdot \mathfrak{Q}_{t+1}^k(x_t) \\ \text{s.t.} & \tilde{A}_t x_{t-1} + \tilde{B}_t x_t = \tilde{b}_t. \end{cases} \tag{2}$$

Whereby, $\mathfrak{Q}^k_{t+1}(x_t)$ is the so-called *value function*, and it corresponds to the expected cost-to-go function given by: $\mathfrak{Q}^k_{t+1}(x_t) := \mathbb{E}_{\xi_{t+1}}[Q^k_{t+1}(x_t, \xi_{t+1})]$, $\forall t \neq T$, and $\mathfrak{Q}^k_{T+1}(x_T) := 0$, $\forall k \in \mathcal{D}$. This is known as a *fully* observable MC because all the information about the stochastic process is available to the DM. Note that in situations where ξ_t has a continuous distribution, a typical approach is to proceed by means of discretization and/or approximate the (discretized) distribution of ξ_t by a sample N_t. This is, for instance, the case in which (2) is a *Sample Average Approximation*. We refer the reader to [11] for a detailed discussion on the topic. As such, under Assumptions 1 and 2, the value function $\mathfrak{Q}^k_t(x_{t-1}) := \mathbb{E}[Q^k_t(x_{t-1}, \xi_t)]$ is piecewise linear convex with respect to x_{t-1}, $\forall t = 1, \ldots, T$ and $\forall k \in \mathcal{D}$. Therefore, it can be approximated from below by an *outer* cutting-plane linear approximation. This cutting-plane outer approximation is represented by the maximum of a collection L^k of hyperplanes where, $\mathfrak{Q}^k_t(x_{t-1}) \geq \max_{\ell \in L} \left\{ \beta^\top_{t,\ell} x_{t-1} + \alpha_{t,\ell} \right\}$ $\forall k \in \mathcal{D}$.

A common approach for assembling the collection of hyperplanes L^k_t is the SDDP algorithm. Drawing influence from the backward recursion technique developed in DP, the SDDP algorithm alternates between the following two main steps. (i) *Forward pass:* a forward simulation which evaluates the current policy obtained by the current approximation for the expected value functions $\mathbb{E}[\check{Q}_t(x_{t-1}, \xi_t)]$ and provides a sequence of decisions $\left(\check{x}_t := \check{x}_t(\xi_t)\right), \forall\, t = 2, \ldots, T$; and (ii) *Backward pass:* a backward recursion to improve the current approximation for $\mathbb{E}[\check{Q}_t(x_{t-1}, \xi_t)]$ by adding new cutting-planes to the collection of hyperplanes L^k_t for $t = T, \ldots 2$. After the forward step, a statistical upper bound for the optimal value of problem (1) can be computed, and after the backward step, an improved lower bound is obtained. We refer the reader to [9] for a further discussion on this topic.

2.2 Partially Observable Multistage Stochastic Programming

Unlike the *fully* observable setting presented in Section 2.1, the methods presented in [4] proceed by assuming that the MC is *partially* observable; meaning that the DM is unable to observe the state of the MC D_t and can only observe the realization of the random variable ξ_t in each stage t. Another interpretation is that this is an MSP defined on a *hidden* MC [10]. For instance, in the hydrothermal power generation planning problem used in our numerical experiments in Section 4, ξ_t represents the random amounts of rainfall (inflows) in each stage t. In this context, the (hidden) unobservable states D_t can represent the different types of rain seasons (e.g., wet, dry, etc.)

Though the DM is unable to observe the state of the MC, he/she maintains a probability distribution θ_t over the set of possible states in the MC, which is referred to as the *belief vector*. This belief vector is then used to reflect the possibility of being in each of the different states, contingent on the observed realizations of the random vector ξ_t at time t. Specifically, upon observing the realization of the random vector ξ_t, the DM updates the *prior* belief vector θ_t according to the Bayesian update formula, such that the *posterior* probability of being in a state $D_t = j$ is given by

$$\mathbb{P}(D_t = j | \xi_t, \xi_{t-1}, \ldots, \xi_1) = \theta_t(j) := \frac{\mathbb{P}(D_t = j, \xi_t, \xi_{t-1}, \ldots, \xi_1)}{\sum_{k \in \mathcal{D}} \mathbb{P}(D_t = k, \xi_t, \xi_{t-1}, \ldots, \xi_1)} \tag{3}$$

Consequently, instead of defining a cost-to-go function for every state $j \in \mathcal{D}$ in the DP formulation (2), we define a single cost-to-go function $Q_t(x_{t-1}, \xi_t, \theta_t)$ for every $t = 1, \ldots, T$ by incorporating the belief vector θ_t as a state variable:

$$Q_t(x_{t-1}, \xi_t, \theta_t) := \begin{cases} \min\limits_{x_t} & f_t(x_t, \xi_t) + \sum\limits_{j \in \mathcal{D}} \theta_t(j) \sum\limits_{k \in \mathcal{D}} p_j(k) \cdot \mathfrak{Q}^k_{t+1}(x_t, \theta_t) \\ \text{s.t.} & \tilde{A}_t x_{t-1} + \tilde{B}_t x_t = \tilde{b}_t, \end{cases} \tag{4}$$

where the value function $\mathfrak{Q}^k_{t+1}(x_t, \theta_t) := \mathbb{E}_{\xi_{t+1}}[Q_{t+1}(x_t, \xi_{t+1}, \theta_{t+1}) | D_{t+1} = k]$, $\forall t \neq T$, and $\mathfrak{Q}^k_{T+1}(x_T, \theta_T) := 0$.

Under the linearity Assumptions 1 and 2, a backward induction argument can be used to show that the cost-to-go function $Q_t(x_{t-1}, \xi_t, \theta_t)$ is piecewise linear *convex* with respect x_{t-1} for any given $\xi_t \in \Xi_t$ and $\theta_t \in \Theta_t$; and piecewise linear *concave* with respect to θ_t for any given $x_{t-1} \in \mathcal{X}_{t-1}$ and $\xi_t \in \Xi_t$. Since the cost-to-go function $Q_t(x_{t-1}, \xi_t, \cdot)$ is piecewise linear concave, the value function $\mathfrak{Q}_t(x_{t-1}, \cdot)$ can be approximated from below by an *inner* cutting-plane linear approximation. This cutting-plane inner approximation is represented by the minimum of a collection of L hyperplanes: $\mathfrak{Q}_t(x_{t-1}, \cdot) \leq \min_{\ell \in L} \left\{ \mu_t^\intercal \theta_{t-1,\ell} + \nu_{t,\ell} \right\}$ and the cost-to-go function (4) is approximated by

$$\check{Q}^L_t(x_{t-1}, \xi_t, \theta_t) := \begin{cases} \min\limits_{x_t} \max\limits_{\gamma_t \in \mathbb{R}^N_+} & f_t(x_t, \xi_t) + \sum_{\ell=1}^L \gamma_{t,n} \mathcal{V}'_{t+1} & \\ \text{s.t.} & \tilde{A}_t x_{t-1} + \tilde{B}_t x_t = \tilde{b}_t, & \\ & \mathcal{V}_{t+1} \geq \alpha_{t+1,\ell} + \beta^\intercal_{t+1,\ell} x_t, & \forall \ell = 1, \ldots, L \\ & \mathcal{V}'_{t+1} \geq -M & \\ & \sum_{\ell=1}^L \gamma_{t,\ell} \theta_{t,\ell} = \theta_t & (\mu_{t+1}) \\ & \sum_{\ell=1}^L \gamma_{t,n} = 1 & (\nu_{t+1}) \end{cases} \tag{5}$$

Here L is the number of belief vectors encountered so far and the inner approximation is achieved by a column

generation approach, in the same spirit as the Dantzig-Wolfe decomposition. Here, $\theta_{t,\ell}$ is the belief vector assembled in the ℓ-th iteration, $-M$ is a deterministic lower bound for the value function obtained a priori, and μ_{t+1}, ν_{t+1} are the dual multipliers. Taking the dual of the inner maximization problem gives

$$\check{Q}_t^L(x_{t-1},\xi_t,\theta_t) := \begin{cases} \min\limits_{x_t,\mu_{t+1},\nu_{t+1}} & f_t(x_t,\xi_t)+\mu_{t+1}^\intercal\theta_t+\nu_{t+1} \\ \text{s.t.} & \tilde{A}_t x_{t-1}+\tilde{B}_t x_t=\tilde{b}_t, \\ & \mu_{t+1}^\intercal\theta_{t,\ell}+\nu_{t+1}\geq\alpha_{t+1,\ell}+\beta_{t+1,\ell}^\intercal x_t, \quad \forall\ell=1,\ldots,L \\ & \mu_{t+1}^\intercal\theta_{t,\ell}+\nu_{t+1}\geq -M, \qquad\qquad \forall\ell=1,\ldots,L. \end{cases} \tag{6}$$

Finally, a variant of the SDDP algorithm, namely the POSDDP algorithm [4], can be performed to solve the problem (6), which converges almost surely to an optimal policy in a finite number of iterations. Here, similarly to SDDP, the POSDDP algorithm moves *forward* in time, evaluating the current approximations of the value functions and assembling a sequence of candidate solutions $\left(\check{x}_t := \check{x}_t(\xi_t)\right)$ for $t=1,\ldots,T$. Moreover, in the backward pass, in each point in time $t=T,\ldots,2$ and given a candidate solution $\check{x}_{t-1}$, problem (6) is solved for each $\xi_t\in\Xi_t$. However, the only difference here is that, instead of adding cuts with certainty to the stage $(t-1)$ problem, the cuts are weighed by the DM's belief at that stage. We refer the reader to [4] for a detailed description of the algorithm and further discussions.

3. Maximum Posterior Out-of-sample Evaluation

While the decision policy provided by the partially observable reformulation (6) provides optimal actions for every belief state, it is important to note that the value function $\mathfrak{Q}_t(x_{t-1},\theta_t)$ in this formulation is now also a function of the belief vector θ_t. This increases the dimensionality of the problem, which might scale the computation very poorly. To overcome this, we propose a *maximum posterior* out-of-sample evaluation approach which we discuss in this section.

The high-level idea of our proposed maximum posterior out-of-sample evaluation approach can be summarized into two main steps: (i) a *fully observable training* step where the states of the Markov chain are ostensibly assumed to be observable, and the policy is henceforth trained for *every* state in the MC, and (ii) a *partially observable out-of-sample evaluation* step, where the belief vector is updated sequentially in a Bayesian fashion (same as the POSDDP setting), while in each stage, we apply the policy corresponding to the value function associated with the state $k\in\mathcal{D}$ that has the maximum posterior probability. That is, $k\in\arg\max_{j\in\mathcal{D}}\theta_t(j)$. We describe these two steps in more detail next.

Fully observable *training*. In the same spirit as the *fully* observable setting that we describe in Section 2.1, we define a value function $Q_t^j(x_{t-1},\xi_t)$ as given by (2) for every state $j\in\mathcal{D}$ in the MC and every stage $t=1,\ldots,T$. For *every* state $j\in\mathcal{D}$, we start with an initial approximation $\check{Q}_t^j(x_{t-1},\xi_t)$, and then we improve the value function approximation $\check{Q}_t^j(x_{t-1},\xi_t)$, $\forall t=1,\ldots,T$ by using iterative forward/backward passes of the SDDP algorithm.

Partially observable out-of-sample *evaluation*. Consider taking a set of scenarios (sample paths) indexed by $\mathcal{L}$ such that $\{\xi^\ell\}_{\ell\in\mathcal{L}}$, with $|\mathcal{L}|\ll|\Xi_1|\times|\Xi_2|\times\cdots\times|\Xi_T|$ and $\xi^\ell=(\xi_2^\ell,\ldots,\xi_T^\ell)$. Given the trained value functions $\check{\mathfrak{Q}}_t^j(\cdot)$, $\forall j\in\mathcal{D}$ and $t=1,\ldots,T$, the deterministic first-stage realization of the random vector ξ_1, the initial state value x_0, and an initial belief vector θ_1, for every sample-path $\ell\in\mathcal{L}$ do the following.

1. Initialize a list for the candidate solutions $(\check{x}_1,\ldots,\check{x}_T)$ and $z_\ell=0$. Set, $t=1$, $\check{x}_{t-1}=x_0$, $\check{\xi}_t=\xi_1$, $\theta_1^\ell=\theta_1$.
2. If $t=1$, sample a state $k\in\mathcal{D}$ according to the initial belief θ_1, i.e., according to probability distribution $\mathbb{P}(D_1=k)=\theta_1(k)$. Else, given the sampled realization ξ_t^ℓ, update the belief vector θ_t according to the Bayes rule (3) and pick a state k according to a decision policy $k\leftarrow\pi(\theta_t)$ that maps a belief vector to a state in the MC.
3. Solve $\check{Q}_t^k(\check{x}_{t-1},\check{\xi}_t^\ell)$ to obtain x_t^*, set $\check{x}_t=x_t^*$ and let $z_\ell\leftarrow z_\ell+f_t(\check{x}_t,\xi_t^\ell)$.
4. If $t=T$, finish this sample path and go to step 1. Otherwise, let $t\leftarrow t+1$ and go to step 2.

A statistical upper bound for the optimal value of (1) is then calculated by $\bar{z}=\frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}}z_\ell$.

Moreover, in our implementation, we also consider the following policies $\pi(\theta_t)$: (i) a *randomized* policy where, during the evaluation, we choose the state *randomly* according to the posterior belief, (ii) a minimum-posterior policy where we choose the state which has the *minimum* posterior belief $k\in\arg\min_{j\in\mathcal{D}}\theta_t(j)$. Since in the randomized policy, the sampling is weighted by the belief, we expect its performance to be close to the maximum-posterior approach – over the long run. Additionally, we use the minimum-posterior to verify whether the practicality of the approach can indeed be attributed to choosing the state with the highest posterior or not.

4. Implementation Details and Numerical Results

In this section, we present some preliminary numerical results for the performance of our proposed heuristic approaches compared to the standard POSDDP algorithm. We consider the multistage hydrothermal power generation planning problem described in [12]. We implemented all algorithms in C++ with commercial solver *CPLEX*, version 12.8. All the tests are conducted on an iMac desktop with four 4.00GHz processors and 16Gb memory. The number of threads is set to be one. To test the performance of each algorithm, first, we implement the *fully* observable and the POSDDP algorithm and train each policy with a time limit of three hours. After the training step, we generate a set of scenarios indexed by $\mathcal{L}$ which samples, both, a set of states $\{D^\ell\}_{\ell\in\mathcal{L}}$ and set of realizations of the random vector $\{\xi^\ell\}_{\ell\in\mathcal{L}}$, where $D^\ell = (D_1^\ell,\ldots,D_T^\ell)$ and $\xi^\ell = (\xi_2^\ell,\ldots,\xi_T^\ell)$, $\forall \ell \in \mathcal{L}$. To sample the states D^ℓ, at time $t=1$ a state D_1^ℓ according to an initial belief where $\theta_t(j) = \frac{1}{|\mathcal{D}|}, \forall j \in D$. Then, for $t = 2,\ldots,T$, state D_t^ℓ is sampled with probability $p_i(j) = \mathbb{P}(D_t^\ell = j | D_{t-1}^\ell = i)$. Moreover, to sample the realizations ξ^ℓ, for $t = 2,\ldots,T$, we sample ξ_t^ℓ with a probability $q_i(\xi_t^\ell) = \mathbb{P}(\xi_t^\ell \in \Xi_t | D_t^\ell = i)$. Each algorithm is then evaluated on the same set of out-of-sample sample paths, and the statistical upper bound $\bar{z}$ for each algorithm is reported as the performance metric. Furthermore, we evaluate the policy obtained by the *fully* observable training step in a *fully* observable setting. That is, during the evaluation step, the DM observes both the state of the MC D_t^k and the realization ξ_t^k, $\forall k \in \mathcal{K}$ and $t = 1,\ldots,T$. This policy is considered as the "ground truth" optimal policy, and all other policies are compared to this one as the benchmark.

In Table 1, we report: (i) the optimality gap pertaining to the different policies (POSDDP, maximum-posterior, minimum-posterior, and random) relative to the fully-observable benchmark (which is denoted by FOSDDP), where each value is given by $(\bar{z}_{\text{policy}} - \bar{z}_{\text{FOSDDP}})/\bar{z}_{\text{FOSDDP}}\%$, and (ii) the time it took to do the *fully observable* training and to train the POSDDP algorithm for each test instance. From Table 1, we see that the POSDDP policy has the best

Instances			Optimality gap				Training-time (seconds)	
$\tilde{d}_t$	$\lvert\mathcal{D}\rvert$	T	Max-Posterior	Min-Posterior	Random	POSDDP	MCSDDP	POSDDP
7000	3	7	5.73%	26.31%	5.99%	2.61%	259.0	240.7
		13	1.16%	3.92%	1.62%	0.84%	1785.7	5723.2
		25	0.44%	1.15%	0.57%	0.34%	4347.1	-
		49	0.34%	1.09%	0.50%	0.27%	10276.1	-
	8	7	7.70%	135.12%	12.39%	5.68%	552.5	141.7
		13	2.50%	27.84%	4.36%	1.82%	1631.9	2711.4
		25	0.69%	5.07%	1.23%	0.70%	4457.9	-
		49	0.53%	4.65%	1.02%	0.67%	-	-
8000	3	7	1.94%	2.62%	1.07%	0.48%	314.6	276.6
		13	0.36%	1.52%	0.55%	0.30%	405.8	1683.4
		25	0.18%	0.81%	0.30%	0.14%	969.6	3047.2
		49	0.17%	0.69%	0.25%	0.13%	2363.0	8141.7
	8	7	0.94%	10.98%	2.66%	0.55%	366.0	137.8
		13	0.60%	6.85%	1.26%	0.66%	588.1	932.6
		25	0.26%	5.73%	0.76%	0.39%	1383.2	5935.2
		49	0.20%	3.22%	0.47%	0.64%	4004.8	-

Table 1: Optimality gap pertaining to each policy (POSDDP, maximum-posterior, minimum-posterior, and randomized policy) relative to the $\bar{z}_{\text{FOSDDP}}$ value obtained by the FOSDDP policy.

performance in most of the instances. Specifically, in instances where $|\mathcal{D}| = 3$ and/or $T = 7$, POSDDP consistently outperforms all the other approaches in terms of the optimality gap. These instances can be seen as small-size instances. In larger instances, i.e., instances where $|\mathcal{D}| = 8$ and $T \in \{25, 61\}$, we can see that the POSDDP algorithm is outperformed by the max-posterior, although the performance difference is small. This, however, can be attributed to the fact that POSDDP reaches the time limit in 3 out of 4 of those instances, and hence the policy is not trained well enough. On the other hand, the difference in the optimality gap between the POSDDP algorithm and the maximum-posterior evaluation is relatively small compared to the difference in the computational time. Specifically, averaging across all test instances, the difference in their optimality gap has an average of 0.47% across the different instances, whereas the computational time is 46.36% (2404.14 seconds) higher for the POSDDP algorithm within the given three hours time limit. In addition, we can see that the randomized policy performs similarly to the maximum-posterior policy. We can also see that the difference in the optimality gap between the different policies tends to shrink as the size of the instance increases (in both $|\mathcal{D}|$ and T). Finally, we can see that overall (and especially in the instances where $\tilde{d}_t = 8000$), the difference in optimality gaps between the different maximum-posterior, the randomized policy and POSDDP is not very large. We suspect that this is due to the specific structure of the problem studied.

5. Conclusion

This work proposes a computationally efficient heuristic approach for solving a class of *partially* observable MSP problems. A recently developed algorithm in the literature, known as the POSDDP algorithm [4], introduces an SDDP variant for solving this class of problems. Because the POSDDP algorithm introduces a belief vector to the value functions, the computation scales poorly as the number of states in the MC increases. Instead, we propose a more practical approach where a value function is defined and trained for each state in the partially observable MC in the sample-path. Then, during the evaluation step, we only choose the state which has the highest posterior. While the POSDDP algorithm defines a policy for every belief, our numerical results show that when comparing our maximum-posterior evaluation approach to the POSDDP algorithm, the reduction in the optimality gap is significantly small compared to the increase in compute time. This observation is more apparent in instances where the MC has a large state space.

Acknowledgments

The authors acknowledge partial support by the National Science Foundation [Grant CMMI 1854960]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Richard Bellman. Dynamic programming. *Princeton University Press*, 1957.

[2] Dimitris Bertsimas, Shimrit Shtern, and Bradley Sturt. A data-driven approach for multi-stage linear optimization. *Available at Optimization Online*, 2018.

[3] Vitor L de Matos, David P Morton, and Erlon C Finardi. Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling. *Annals of Operations Research*, 253(2):713–731, 2017.

[4] Oscar Dowson, David P Morton, and Bernardo K Pagnoncelli. Partially observable multistage stochastic programming. *Operations Research Letters*, 48(4):505–512, 2020.

[5] Jitka Dupačová and Jan Polívka. Asset-liability management for Czech pension funds using stochastic programming. *Annals of Operations Research*, 165(1):5–28, 2009.

[6] Boutheina Fhoula, Adnene Hajji, and Monia Rekik. Stochastic dual dynamic programming for transportation planning under demand uncertainty. In *2013 International Conference on Advanced Logistics and Transport*, pages 550–555. IEEE, 2013.

[7] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[8] Jianqiu Huang, Kezhuo Zhou, and Yongpei Guan. A study of distributionally robust multistage stochastic optimization. *arXiv preprint arXiv:1708.07930*, 2017.

[9] Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991.

[10] Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[11] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[12] Wim Van Ackooij, Welington de Oliveira, and Yongjia Song. On level regularization with normal solutions in decomposition methods for multistage stochastic programming problems. *Computational Optimization and Applications*, 74(1):1–42, 2019.