Physics-Informed Stochastic Post-Decision State Reinforcement Learning for Computationally-Efficient IoT Sensor Scheduling

Alireza Mohammadhosseini*, Andrew Corra†, Ming Shi†, Nicholas Mastronarde†, and Jacob Chakareski*
*College of Computing, New Jersey Institute of Technology; †Dept. of Electrical Engineering, University at Buffalo

***Abstract*—Delay-sensitive Internet of Things (IoT) applications are rapidly growing with the adoption of new wireless technologies. We formulate the transmission scheduling problem for an IoT sensor as a Constrained Markov Decision Process (CMDP) that aims to minimize the transmission power consumption subject to a buffer cost constraint, and propose to solve it using reinforement learning (RL). However, conventional RL techniques ignore prior knowledge of system dynamics, leading to sample inefficiency. Post-decision state (PDS) learning mitigates this issue by leveraging known system information to improve learning performance, but its high computational complexity makes it impractical for resource-constrained devices. To address this challenge, we propose *stochastic PDS learning*, a novel RL algorithm combining PDS principles with stochastic sampling. This approach produces a physics-informed RL agent that can exploit known system information even while operating under limited computational resources. We further extend this approach to Virtual Experience (VE) learning, which exploits the basic "physics" of the system to learn about multiple states in each time step, demonstrating that stochastic sampling enables practical implementation of the otherwise computationally intensive VE method. Simulation results show that sampling-based VE learning achieves 5-10% weighted percent error (WPE) between the optimal value function and learned value function and sampling-based PDS learning achieves 25–30% WPE, both outperforming Q-learning (with 45–50% WPE) and achieving performance comparable to standard PDS and VE methods without sampling. Notably, the sampling-based algorithms are effective even when using a single sample for approximation, validating their efficiency and suitability for resource-constrained IoT devices.**

***Index Terms*—Physics-informed learning, Reinforcement learning, Constrainted Markov decision process, Post-decision states, Stochastic sampling, IoT sensors, transmission scheduling**

I. Introduction

As 5G/6G networks and Internet of Things (IoT) systems evolve, devices are increasingly deployed in unknown and dynamic environments requiring adaptive control strategies. Reinforcement Learning (RL) has emerged as powerful tool for these scenarios, enabling agents to optimize their decision *policies* and *value functions* purely through interaction with their operating environment, where the value function quantifies the "desirability" of being in each state and the policy determines the action to take in each state.

However, traditional RL algorithms—whether model-free or model-based, tabular or approximated—are fundamentally data-driven. They treat the system as a "black box," ignoring prior knowledge of system dynamics (e.g., packet losses, buffer evolution), available actions (e.g., scheduling decisions), or cost structures (e.g., energy, delay).

By failing to leverage this prior knowledge, conventional data-driven agents are forced to "learn" the basic physics of the environment, leading to significant sample inefficiency and slower convergence. To address this, *physics-based* or *physics-informed* machine learning [1], [2] is proposed, which integrates models of systems into the learning process, enabling algorithms to generalize better from fewer training samples.

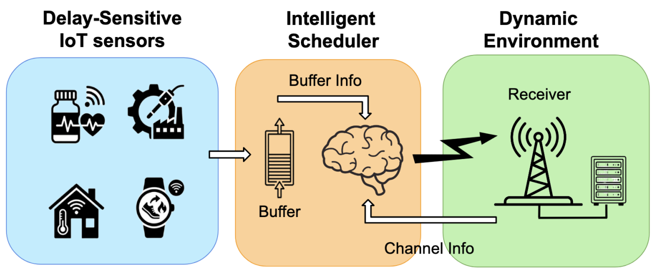


Fig. 1: Intelligent low-complexity scheduler for delay-sensitive wireless IoT sensors and applications over wireless networks.

In this paper, we embed the prior knowledge of the system into the RL algorithm rather than utilizing a neural network to extract them. This can be achieved with so-called PDSs (also known as afterstates [3]), which alter the learning structure by factoring the state transition into *known* components controlled by system physics and *unknown* components driven by environment stochasticity.

Extensive work has demonstrated the efficacy of PDS-based learning and its ability to improve the empirical convergence rate of RL algorithms [4]–[10]. Early applications, such as [4] and [6], demonstrated that PDS formulations could optimize energy-efficient scheduling over fading channels with remarkable efficiency. These studies defined a deterministic PDS, where the intermediate state (e.g., the buffer level post-transmission) is a deterministic function of the current state and action. Under this assumption, PDS learning has complexity comparable to tabular model-free RL algorithms such as Q-learning.

In [5], Mastronarde et al. addressed a variation of the energy-efficient delay constrained scheduling problem by generalizing the PDS framework to allow the PDS to be a (known) stochastic function of the current state and action. In [7], Sharma et al. minimized the queuing delay of an energy harvesting wireless sensor, utilizing PDS-based RL and value function approximation. Subsequently, Zhang et al. [8], proposed *blind* PDS learning, a method that eliminated the need for prior knowledge of the "known" component of the transition probabilities, by estimating them directly from interactions with the environment. He et al. [9] and Yang et al. [10] augmented deep RL (DRL) algorithms with PDSs to optimize offloading in mobile-edge computing and beamforming for intelligent reflective surface (IRS)-aided secure communication. However, when the PDS is a stochastic function of the state and action, the convergence gains of PDS-based RL come at the expense of increased computational complexity due to the need to compute expectations over the known transition probability function (or its estimate in [8]). This added complexity can be prohibitive for resource-constrained devices or in highly complex DRL environments with continuous state variables or multiple users.

In [11], Sun et al. proposed a hardware accelerator based on stochastic computing to speed up the aforementioned expectation calculation. They showed that extremely short stochastic representations could be used to approximate the expectation without significantly sacrificing learning performance.

In our prior work [12], inspired by the results in [11], we introduced a PDS-based RL framework that utilized stochastic sampling to estimate the expectation over the PDS value function with respect to the known component of the transition probability function, demonstrating that stochastic sampling can achieve performance similar to traditional learning algorithms at significantly lower complexity. Compared to [12], this paper makes the following contributions:

- **First**, we formulate delay-sensitive point-to-point IoT sensor transmission scheduling as a constrained Markov decision process (CMDP) and apply an online Lagrangian multiplier method to dynamically adjust the cost weight to minimize transmission power under a buffer delay constraint.
- **Second**, we apply stochastic sampling to PDS Learning and extend it to Virtual Experience (VE) Learning – first introduced in [5], [13] – thereby enabling multiple low-cost virtual updates of the value function from each real transition.
- **Third**, we conduct extensive simulations and provide a detailed evaluation, demonstrating that the proposed methods achieve performance comparable to standard PDS Learning and VE Learning baselines with substantially lower computational complexity.

The rest of the paper is structured as follows. Section II details the system model for point-to-point transmission scheduling. Section III formally presents the CMDP formulation and reviews the standard PDS and Virtual Experience (VE) algorithms. In Section IV, we introduce our core contribution: the stochastic sampling-based learning algorithm. Section V provides a comprehensive performance evaluation against standard baselines. Finally, Section VI concludes the paper.

II. System Model

As depicted in Fig. 2, we consider a simple time-slotted point-to-point wireless communication system in which data is transmitted from a packet buffer of size N_b to a receiver over a block fading channel. We consider slots of equal length ΔT and let t denote the current time slot index. System states are observed at the beginning of each time slot and are denoted by $s_t = (b_t, h_t) \in \mathcal{S} = \mathcal{B} \times \mathcal{H}$, where $b_t \in \mathcal{B} = \{0, 1, ..., N_b\}$ denotes the buffer state (number of buffered packets) and $h_t \in \mathcal{H}$ denotes the channel state (fading channel coefficient).

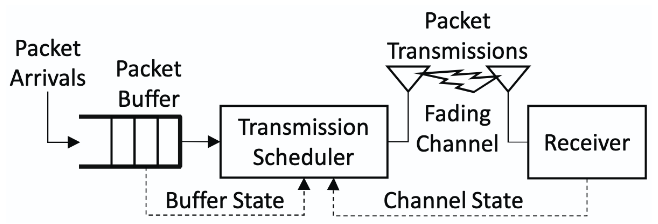


Fig. 2: Point-to-point wireless communication system model.

In each time slot, based on the current buffer and channel states, the transmission scheduler selects a scheduling action a_t from the set $\mathcal{A}(s_t) = \{0, 1, ..., \min(N_a, b_t)\}$, where a_t denotes the number of transmitted packets and N_a denotes the maximum number of packets that can be transmitted based on the available communication bandwidth and modulation schemes. Since it is not possible to transmit more packets than are in the buffer, we also have $a_t \leq b_t$. The goal of the scheduler is to learn a decision policy π in which actions are selected that minimize power consumption subject to a buffer delay constraint. While the considered model is chosen for its clarity, the principles of our framework are extensible to more complex scenarios, such as age of information [14] and adaptive video streaming [15].

A. Channel Model

Similar to prior work [4]–[6], we consider a frequency non-selective block-fading channel, where the channel state h_t denotes the fading channel coefficient between the transmitter and the receiver during time slot t, the set of possible channel states is discrete and finite, the channel state h_t is constant for the entirety of each time slot, and h_t evolves after each time slot according to a stationary but unknown Markov chain with transition probability function $P^h(h'|h)$.

B. Physical Layer Model

We assume a single-input single-output physical layer with data rate β_t/T_s bits/second, where T_s is the symbol duration and β_t denotes the number of bits per symbol, which depends on the chosen modulation scheme. To transmit a_t packets of size M bits, a modulation scheme is selected such that $\beta_t = \lceil a_t M T_s / \Delta T \rceil$ is satisfied, where ΔT is the time slot duration and $\lceil \cdot \rceil$ denotes the ceiling operator.

We set a fixed target bit error probability BEP_{target} for all transmissions. We let $\rho(h_t, a_t; BEP_{target})$ watts denote the required transmission power to transmit a_t packets in channel state h_t with maximum bit-error probability BEP_{target}. In this paper, we assume Quadrature Amplitude Modulation (M-QAM) such that the transmission power is defined as in [11]; however, other modulation schemes can also be used, e.g., phase-shift keying (M-PSK) as in [7].

C. Traffic and Buffer Models

As noted earlier, $b_t \in \mathcal{B} = \{0, 1, ..., N_b\}$ denotes the IoT sensor's finite buffer state in time slot t, where N_b is the maximum number of packets that can be held in the buffer. At the end of time slot t, after transmitting a_t packets, l_t new packets arrive. We assume that packet arrivals are independent and identically distributed (IID) with unknown distribution $P^l(l)$. If more packets arrive than can fit in the finite buffer, then they are dropped. The buffer state evolves according to the following recursion:

$$b_{t+1} = \min(b_t - f_t(a_t; BEP_{target}) + l_t, N_b), \tag{1}$$

where $f_t(a_t; BEP_{target})$ denotes the number of packets correctly received at the receiver in time slot t, which we refer to as the *goodput*. Since the number of received packets cannot exceed the number of transmitted packets, we have $0 \leq f_t(a_t; BEP_{target}) \leq a_t$ by definition. For brevity, we will write the goodput as $f_t(a_t)$ in the remainder of the paper. Note that arrivals at the end of time slot t cannot be scheduled in time slot t and untransmitted packets remain in the buffer to be transmitted in a future time slot.

Under the assumption of IID bit errors and no error correction, we can model the goodput $f_t(a_t)$ as a binomial random variable with probability mass function $P^f(f|a;q) = \binom{a}{f}(1-q)^f q^{a-f}, f = 0, 1, ..., a$, where the packet loss rate (PLR) q can be expressed as $q = 1 - (1 - BEP_{target})^M$.

Given the scheduling action, goodput distribution, and packet arrival distribution, the buffer state transition probability function can be expressed as:

$$P^b(b_{t+1}|b_t, a_t) = \mathbb{E}_{f,l}\left[\mathbb{I}_{\{b_{t+1}=\min(b_t-f+l,N_b)\}}\right], \tag{2}$$

where $\mathbb{E}_{f,l}$ denotes an expectation over $P^f(\cdot|a;q)$ and $P^l(\cdot)$; and $\mathbb{I}_{\{\cdot\}}$ denotes an indicator function that is set to 1 when the statement in $\{\cdot\}$ is true and set to 0 otherwise.

III. PDS-based Reinforcement Learning

A. CMDP Formulation

We formulate the delay-sensitive energy-efficient transmission scheduling problem as a Constrained Markov Decision Process (CMDP) based on the system model defined in Section II. We define the CMDP's finite state set as $\mathcal{S} = \mathcal{B} \times \mathcal{H}$ and its finite action set as $\mathcal{A}$. The sequence of states $s_t = (b_t, h_t)$, $t = 0, 1, \ldots$ evolves according to a controlled Markov chain with transition probability function $P(s'|s,a) = P^b(b'|b,a)P^h(h'|h)$. Note that we cannot calculate the transition probability function directly because we do not know the channel state transition probabilities $P^h(h'|h)$ or the packet arrival distribution $P^l(l)$.

The goal of the scheduler is to minimize the transmission power consumption while satisfying a holding cost constraint. We explicitly separate the buffer-related costs into the overflow penalty c^{ovf}, which penalizes the number of dropped packets due to the finite buffer size, and the holding cost c^{hold}, representing the number of packets that will be held in the buffer into the next time slot:

$$c^{\text{hold}}(b,a) = b - \mathbb{E}_f[f(a)], \tag{3}$$

$$c^{\text{ovf}}(b,a) = \eta \cdot \mathbb{E}_{f,l}[\max(b - f(a) + l - N_b, 0)], \tag{4}$$

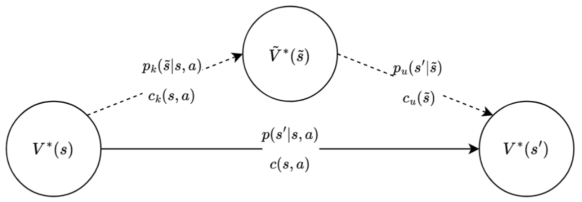


Fig. 3: Relationships among the current state value $V(s)$, next state value $V(s')$, and PDS value $\tilde{V}(\tilde{s})$. Figure inspired by [8].

where η is static penalty weight for packet drops. Now, we formulate the following constrained optimization problem:

$$\min_{\pi} \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \left(\rho(h_t, a_t) + c^{\text{ovf}}(b_t, a_t)\right)\right] \tag{5}$$

$$\text{s.t.} \quad \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t c^{\text{hold}}(b_t, a_t)\right] \leq \frac{\mathrm{C}_b}{1-\gamma}. \tag{6}$$

where γ is the discount factor, $\rho(h_t, a_t)$ is the transmission power cost defined in Section II-B, C_b is the target holding cost (buffer level), and $a_t = \pi(s_t)$.

To solve this constrained problem, we employ the Lagrange multiplier method [16]. This converts the CMDP problem into an unconstrained problem by introducing a Lagrange multiplier $\lambda \geq 0$. The resulting cost function $c(s,a;\lambda)$ becomes a sum of the power, overflow, and weighted holding costs:

$$c(s,a;\lambda) = \rho(h,a) + c^{\text{ovf}}(b,a) + \lambda c^{\text{hold}}(b,a). \tag{7}$$

Now, the optimal value $V^*(s)$ that satisfies the Bellman equation [17] can be written as:

$$V^*(s) = \min_{a\in\mathcal{A}(s)} \left\{c(s,a;\lambda) + \gamma \sum\nolimits_{s'\in\mathcal{S}} P(s'|s,a)V^*(s')\right\}. \tag{8}$$

However, since the transition probabilities and the arrival distribution are unknown, we cannot solve (8) via dynamic programming. Furthermore, the optimal λ^* that satisfies the constraint is unknown. This motivates the use of a reinforcement learning approach that can handle unknown dynamics and also adapt λ online.

B. Post-Decision States

A PDS, denoted by $\tilde{s} \in \mathcal{S}$, represents an intermediate state after the known effects of an action take place but before the unknown dynamics take place. For the system model defined in Section II, we define the PDS in time step t as $\tilde{s}_t = (\tilde{b}_t, \tilde{h}_t)$, where the post-decision buffer state $\tilde{b}_t = b_t - f_t(a_t)$ denotes the buffer state after packets are successfully transmitted but before new packets arrive and the post-decision channel state $\tilde{h}_t = h_t$ because the action has no impact on the channel state. We can then express the state in time step $t+1$ in terms of the PDS in time step t as $s_{t+1} = (b_{t+1}, h_{t+1})$, where $b_{t+1} = \min(\tilde{b}_t + l_t, N_b)$ with l_t drawn from the unknown

Algorithm 1 PDS Learning Algorithm

```
1: Initialize $\tilde{V}_0(\tilde{s})$ arbitrarily
2: Initialize $P^k(\tilde{s}|s,a)$ and $c^k(s,a)$
3: Initialize $n(\tilde{s}) = 0$ for all $\tilde{s} \in \tilde{\mathcal{S}}$
4: for $t$ in $0, 1, \ldots$ do
5:     Observe the state $s_t = (b_t, h_t)$
6:     Select the action $a_t$ using (11)
7:     Observe the PDS experience tuple $\sigma_t = (\tilde{s}_t, c_t^u, s_{t+1})$
8:     Evaluate the state value $V_t(s_{t+1})$ using (13)
9:     Update the learning step size $\alpha_t$ based on $n(\tilde{s}_t)$
10:    Update the PDS value $\tilde{V}_{t+1}(\tilde{s}_t)$ using (12)
11:    $n(\tilde{s}_t) \leftarrow n(\tilde{s}_t) + 1$
12:    Dynamic cost weight adjustment using (14)
13: end for
```

arrival distribution $P^l(l)$ and h_{t+1} drawn from the unknown channel transition probabilities $P^h(h'|\tilde{h}_t)$.

More generally, as depicted in Fig. 3, the PDS factors the transition from the current state s to the next state s' into two parts: 1) a *known* component from the state s to the PDS $\tilde{s}$ with expected cost $c^k(s,a)$ and transition probabilities $P^k(\tilde{s}|s,a)$; and 2) an *unknown* component from the PDS $\tilde{s}$ to the next state s' with expected cost $c^u(\tilde{s})$ and transition probabilities $P^u(s'|\tilde{s})$. Based on this factorization, we can decompose the Bellman equation in (8) as follows:

$$\tilde{V}^*(\tilde{s}) = c^u(\tilde{s}) + \gamma \sum\nolimits_{s' \in \mathcal{S}} P^u(s'|\tilde{s}) V^*(s'), \tag{9}$$

$$V^*(s) = \min_{a \in \mathcal{A}} \left\{ c^k(s,a) + \sum\nolimits_{\tilde{s} \in \mathcal{S}} P^k(\tilde{s}|s,a) \tilde{V}^*(\tilde{s}) \right\}. \tag{10}$$

Importantly, if we know the optimal PDS value function $\tilde{V}^*(\tilde{s})$, then we can find the optimal action $\pi^*(s)$ in state s by taking the action that optimizes the right-hand side of (10).

For the system model defined in Section II and our PDS definition $\tilde{s}_t = (b_t - f_t(a_t), h_t)$, the known cost function $c^k(s,a)$ includes the transmission power and the weighted holding cost (3) and the unknown cost function $c^u(\tilde{s})$ is the weighted overflow cost defined in (4); the known transition probabilities $P^k(\tilde{s}|s,a)$ depend on the known goodput distribution; and the unknown transition probabilities $P^u(s'|\tilde{s})$ depend on the unknown arrival distribution and channel transition probabilities.

C. The PDS Learning Algorithm

We mentioned earlier that if we can learn the optimal PDS value function $\tilde{V}^*(\tilde{s})$, then we can determine the optimal policy $\pi^*(s)$. To this end, we use the PDS learning algorithm presented in Algorithm 1 to learn $\tilde{V}^*(\tilde{s})$. First, we initialize the PDS value function $\tilde{V}_0(\tilde{s})$ arbitrarily (line 1). Next, we initialize the known transition probabilities $P^k(\tilde{s}|s,a)$ and the known cost function $c^k(s,a)$ based on our knowledge of the system (line 2). Lastly, we initialize $n(\tilde{s}) = 0$ for all post-decision states to track update counts and to set the step size (line 3). In each time step t, we observe the current state $s_t = (b_t, h_t)$ (line 5) and then select an action as (line 6):

$$a_t = \arg\min_{a \in \mathcal{A}(s_t)} \left\{ c^k(s_t,a) + \sum\nolimits_{\tilde{s} \in \mathcal{S}} P^k(\tilde{s}|s_t,a) \tilde{V}_t(\tilde{s}) \right\}. \tag{11}$$

After taking the action, we observe the PDS experience tuple $\sigma_t = (\tilde{s}_t, c_t^u, s_{t+1})$, which comprises the resulting PDS $\tilde{s}_t \sim P^k(\cdot|s,a)$, the resulting unknown cost c_t^u such that $\mathbb{E}[c_t^u] = c^u(\tilde{s}_t)$, and the resulting next state $s_{t+1} \sim P^u(\cdot|\tilde{s}_t)$ (line 7). We then update the PDS value function based on the experience tuple using the following update rule (line 10):

$$\tilde{V}_{t+1}(\tilde{s}_t) \leftarrow (1-\alpha_t)\tilde{V}_t(\tilde{s}_t) + \alpha_t[c_t^u + \gamma V_t(s_{t+1})], \tag{12}$$

where the decaying learning step size α_t is calculated using $n(\tilde{s}_t)$ (line 9) and

$$V_t(s_{t+1}) = \min_{a \in \mathcal{A}(s_{t+1})} \left\{ c^k(s_{t+1},a) + \sum\nolimits_{\tilde{s} \in \mathcal{S}} P^k(\tilde{s}|s_{t+1},a) \tilde{V}_t(\tilde{s}) \right\} \tag{13}$$

is calculated in line 8. Then, $n(s_t, a_t)$ is incremented by 1 (line 11). Note that the cost of updating $n(\tilde{s}_t)$ and calculating α_t is negligible compared to calculating the expectations in (12) and (13).

Lastly, we deploy a dynamic weight adaptation mechanism (line 12). This method updates λ online using a sub-gradient descent approach [18] to find a $\lambda^* \geq 0$ such that the resulting policy satisfies the holding-cost budget C_b. We treat λ as a learnable parameter that increases when the constraint is violated and decreases when the constraint is satisfied with slack. The Lagrange multiplier λ is updated using the difference between the estimated cost and the target constraint C_b:

$$\lambda_{t+1} = \left| \lambda_t + \alpha_\lambda \left(c_t^{\text{hold}} - \mathrm{C}_b \right) \right|_{\lambda_{\min}}^{\lambda_{\max}}, \tag{14}$$

where we use an L1 norm projection operator $|\cdot|_a^b$ to restrict the Lagrange multiplier values to a range $[\lambda_{\min}, \lambda_{\max}]$ and ensure numerical stability of the algorithm, and α_λ is the dual learning step size.

This dynamic λ is injected into the cost function $c(s, a; \lambda)$ in (7). This mechanism allows the framework to automatically converge to the optimal trade-off point that satisfies the user-defined holding cost constraint.

Significant improvements are made over data-driven techniques by leveraging known information in the PDS learning algorithm. Only information about the transition from the PDS to the next state must be learned, and learned information is applicable to all possible state-action pairs that precede the PDS. Additionally, we eliminate the need for action exploration since all effects of the selected action are captured in the PDS. In other words, we can always take the greedy action in each time slot (line 6 of Algorithm 1) and we do not need to explore sub-optimal actions.

It should be noted, however, that the benefits of PDS learning come at the cost of added computational complexity compared to a purely data-driven RL algorithm like Q-learning. Specifically, both the *action selection* and *value function evaluation* steps of the PDS learning algorithm (lines 6 and 8 in Algorithm 1, respectively), require calculating an expectation over the PDS value function using the known transition probabilities. This adds significant complexity, which can make it infeasible to implement PDS learning on resource-constrained devices. We address this problem in Section IV.

D. Virtual Experience Learning

While PDS learning exploits known components of the system model to avoid exploration, its convergence rate is still limited by the fact that only one PDS is updated in each time slot. Virtual Experience (VE), first introduced in [5], [13], accelerates learning by generating multiple valid "virtual" experience tuples from a single observed time slot and performing multiple PDS updates in parallel.

In our system, the packet arrival process and the channel-state evolution are independent of the buffer state. Therefore, the realizations observed in slot t (e.g., packet arrivals and the next channel state) is valid not only for the current post-decision buffer state, but also for any post-decision buffer state. VE leverages this property by reusing the same observed outcomes to create a set of virtual experience tuples across a collection of buffer states.

Let $s_t = (b_t, h_t)$ denote the pre-decision state and $\tilde{s}_t = (\tilde{b}_t, h_t)$ denote the PDS, where $\tilde{b}_t$ is the buffer state after transmission but before arrivals. Standard PDS learning (Algorithm 1) updates $\tilde{V}(\tilde{s}_t)$ only for the realized $\tilde{b}_t$. Under VE (Algorithm 2), we choose a virtual set of post-decision buffer states $\mathcal{B}_{\mathrm{VE}} \subseteq \mathcal{B}$ (e.g., $\mathcal{B}_{\mathrm{VE}} = \mathcal{B}$) and update all post-decision states $\{(\tilde{b}, h_t) : \tilde{b} \in \mathcal{B}_{\mathrm{VE}}\}$ using same observed realizations of the arrivals l_t and the next channel state h_{t+1}. Specifically, given l_t and h_{t+1}, for each $\tilde{b} \in \mathcal{B}_{\mathrm{VE}}$ we define the virtual next state $s_{t+1}^{(\tilde{b})} = (b_{t+1}^{(\tilde{b})}, h_{t+1})$, where $b_{t+1}^{(\tilde{b})} = \min(\tilde{b}+l_t, N_b)$, and the corresponding unknown cost $c_t^{u,(\tilde{b})}$. Using these, VE updates the PDS value function as

$$\tilde{V}_{t+1}\left(\tilde{s}_t^{(\tilde{b})}\right) \leftarrow (1-\alpha_t^{(\tilde{b})})\tilde{V}_t\left(\tilde{s}_t^{(\tilde{b})}\right) + \alpha_t^{(\tilde{b})}\left[c_t^{u,(\tilde{b})} + \gamma V_t\left(s_{t+1}^{(\tilde{b})}\right)\right], \tag{15}$$

where $V_t(\cdot)$ is evaluated as in (13), and $\alpha_t^{(\tilde{b})}$ may be chosen based on the visit count of the corresponding PDS.

VE increases sample efficiency by performing $|\mathcal{B}_{\mathrm{VE}}|$ value updates per slot, significantly accelerating convergence compared to updating only the realized $\tilde{s}_t$. However, this further increases computational cost. In the next section, we introduce our proposed sample-based method to reduce the complexity of PDS learning and VE learning with only marginal impact on the learning convergence rate.

IV. Stochastic Sampling Method

To mitigate the computational complexity of PDS and VE learning, we propose to replace the expectation computations in (11) and (13) with a sample mean of PDS values, i.e.,

$$\sum\nolimits_{\tilde{s}\in\mathcal{S}} P^k(\tilde{s}|s,a)\tilde{V}(\tilde{s}) \approx \frac{1}{N}\sum\nolimits_{n=1}^{N} \tilde{V}(\tilde{s}_n), \tag{16}$$

where $\tilde{s}_n \sim P^k(\cdot|s,a)$ for $n = 1, 2, \ldots, N$. By the weak law of large numbers, the sample mean on the right-hand side of (16) converges in probability to the expected PDS value on the left-hand side of (16) in the limit as the number of samples $N \to \infty$. We hypothesize that for finite N, if (16) is substituted into (13), then the stochastic iterative PDS value function updates in (12) still converge to the optimal PDS value function $\tilde{V}^*(\tilde{s})$ because the expectation will be accurately estimated as the number of steps $t \to \infty$. A formal proof of this is left for future work. Replacing the expectation with the sample mean reduces the computational complexity from being proportional to $|\mathcal{S}|$ to being proportional to N (with $N \ll |\mathcal{S}|$), where $|\mathcal{S}|$ denotes the cardinality of the set $\mathcal{S}$ and N denotes the number of samples. N is a parameter that trades estimation accuracy with computational complexity, allowing application-specific tuning of N for both the action-selection step and the value function evaluation step. Thus, with N tuned appropriately, even systems with few computational resources can execute sampling-based PDS learning.

Algorithm 2 Virtual Experience Learning (VE)

```
1:  Initialize $\tilde{V}_0(\tilde{s})$ for all $\tilde{s}$
2:  Choose VE buffer set $\mathcal{B}_{\mathrm{VE}} \subseteq \mathcal{B}$ and VE update period $T$
3:  for $t = 0, 1, 2, \ldots$ do
4:      Observe current state $s_t = (b_t, h_t)$
5:      Select the action $a_t$ using (11)
6:      Execute $a_t$ and observe arrivals $l_t$ and next channel state $h_{t+1}$
7:      if $t \bmod T = 0$ then
8:          for each $\tilde{b} \in \mathcal{B}_{\mathrm{VE}}$ do
9:              Form virtual PDS $\tilde{s}_t^{(\tilde{b})} \leftarrow (\tilde{b}, h_t)$
10:             Obtain $s_{t+1}^{(\tilde{b})}$ and $c_t^{u,(\tilde{b})}$
11:             Evaluate $V_t(s_{t+1}^{(\tilde{b})})$ using (13)
12:             Update $\tilde{V}_{t+1}(\tilde{s}_t^{(\tilde{b})})$ using (15)
13:             $n(\tilde{s}_t^{(\tilde{b})}) \leftarrow n(\tilde{s}_t^{(\tilde{b})}) + 1$
14:         end for
15:     else
16:         Only update the PDS value $\tilde{V}_{t+1}(\tilde{s}_t)$ using (12)
17:         $n(\tilde{s}_t) \leftarrow n(\tilde{s}_t) + 1$
18:     end if
19:     Dynamic cost weight adjustment using (14)
20: end for
```

The proposed sampling-based PDS learning algorithm can be obtained from Algorithm 1 with two simple updates. First, instead of using (11) to select the action on line 6, we use:

$$a_t = \arg\min_{a\in\mathcal{A}(s_t)} \left\{ c^k(s_t, a) + \frac{1}{N_1}\sum\nolimits_{n=1}^{N_1} \tilde{V}_t(\tilde{s}_n) \right\}, \tag{17}$$

where $\tilde{s}_n \sim P^k(\cdot|s_t, a)$ for $n = 1, 2, \ldots, N_1$. Second, instead of using (13) to evaluate the value function on line 8, we use:

$$V_t(s_{t+1}) = \min_{a\in\mathcal{A}(s_{t+1})} \left\{ c^k(s_{t+1}, a) + \frac{1}{N_2}\sum_{n=1}^{N_2} \tilde{V}_t(\tilde{s}_n) \right\}, \tag{18}$$

where $\tilde{s}_n \sim P^k(\cdot|s_{t+1}, a)$ for $n = 1, 2, \ldots, N_2$. In our simulation results, we explore the impact of the sample numbers

TABLE I: Simulation Parameters

Parameter	Notation	Value	Units
Discount Factor	γ	0.95	-
Overflow Penalty (fixed)	η	50	-
Holding Cost (initial)	λ	0.1	-
Buffer Capacity	N_b	40	Packets
Max Action	N_a	8	Packets
Packet Size	M	5000	Bits
Noise Power Spectral Density	N_0	10^{-7}	Watts/Hz
Channel Bandwidth	W	10	MHz
Time Slot Duration	ΔT	1	Sec
Avg. Arrival Rate	μ	3	Packets/ΔT
Packet Loss Rate	q	10%	-

N_1 and N_2 in the action selection and state value function evaluation steps, respectively, on the learning performance.

Note that VE learning (Algorithm 2) can also be implemented using sampling by selecting the action on line 5 using (17) and updating the value function on line 11 using (18) with s_{t+1} replaced with $s_{t+1}^{(\tilde{b})}$.

V. Evaluation

We implement the system model described in Section II as a custom Gymnasium environment [19] with the key simulation parameters summarized in Table I. We decay the learning step size as $\alpha_t = (1/(n(\tilde{s}_t)+1))^{3/4}$, where $n(\tilde{s})$ denotes the total number of visits to PDS $\tilde{s}$. We assume a Poisson packet arrival process with mean $\mu = 3$ packets per time slot. We define the set of channel gains as $\mathcal{H}$ ={-18.82, -13.79, -11.23, -9.37, -7.80, -6.30, -4.68, -2.08} dB and define define $P^h(h'|h)$ with equal probability of staying in the same state or moving to an adjacent channel state. We compare the performance of the proposed sampling-based PDS learning algorithm (Stoch. PDS) against multiple benchmarks: the optimal policy (Ideal Policy) computed using value iteration, Q-learning [3], and standard PDS learning (Algorithm 1). All results are averaged over 10 simulations of 50,000 time slots.

A. Evaluation Results

1) Performance with Dynamic Weight Adaptation: Here, we report the performance of Stochastic PDS learning against benchmark algorithms Q-learning, PDS, and Virtual Experience (VE) under the dynamic weight controller that enforces a buffer (holding) budget C$_b$ = 5. In contrast to the fixed λ setting, here the Lagrange multiplier λ is updated online based on the observed holding cost. To provide an “ideal” reference, we additionally plot Ideal (with λ^*), where λ^* is found via bisection so that the resulting policy attains the same holding target C$_b$ in the environment.

Fig. 4 shows the evolution of λ. All learning-based methods increase the holding cost weight sharply once the queue builds up, then reduce it as the queue is driven back toward the budget. However, the evolution of λ differs substantially: Q-learning requires a much larger multiplier, which does not converge within the simulation duration, while VE quickly stabilizes near the optimal λ with notably smaller variation. This behavior indicates that VE's parallel value updates react faster to the changing Lagrangian cost than Q-learning, which requires more time slots and larger penalty signals to correct its trajectory.

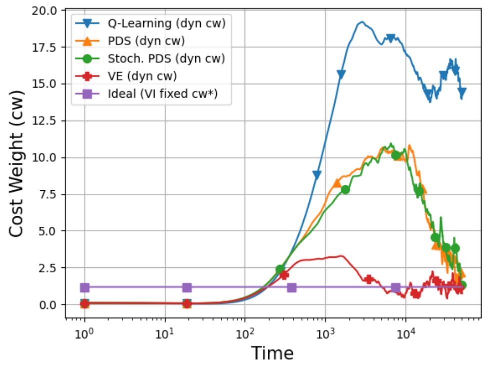


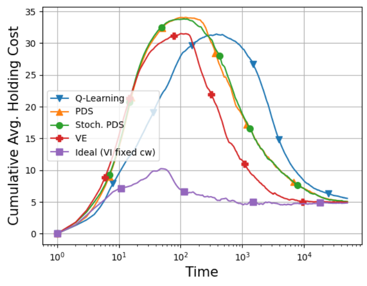
Fig. 4: Evolution of the dynamic cost weight (Lagrange multiplier) λ under the holding-cost budget $C_b = 5$.

Fig. 5a plots the cumulative-average holding cost. Because the controller starts from a small initial cost weight λ and without knowledge of the arrival distribution $P^l(\cdot)$ and channel state transition probabilities $P^h(\cdot|\tilde{h}_t)$, all RL methods exhibit an early transient where the queue length increases and then is corrected by the rising λ. Importantly, all methods eventually converge close to the target budget C$_b$ = 5, confirming that the dynamic weight mechanism successfully enforces the holding cost constraint in this setting. While VE achieves the fastest convergence, Stochastic PDS demonstrates remarkable efficiency with $N_1 = N_2 = 10$ samples. It significantly outperforms Q-learning and achieves convergence rate comparable to standard PDS learning while being significantly less computationally intensive.

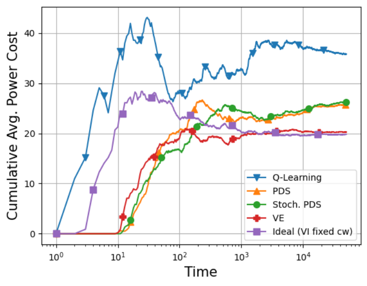
Figs. 5b and 5c illustrate power and overflow performance. Although the dynamic controller aligns all methods to the same holding target, the power efficiency required to satisfy this constraint varies. Stochastic PDS demonstrates its effectiveness by achieving a power trajectory that closely tracks the standard PDS benchmark and remains competitive with the computationally intensive VE method. Crucially, it converges to a substantially lower steady-state power than Q-learning, proving that the sampling approximation captures sufficient structural information to optimize energy efficiency. A similar trend is observed in the cumulative average overflow cost. As the cost weight increases, Stochastic PDS effectively suppresses overflow events, matching standard PDS and approaching the Ideal VI baseline. While, Q-learning reaches the lowest overflow cost with significantly higher power usage.

2) Effect of the Number of Samples: In this section, we explore the robustness of the proposed dynamic Stochastic PDS framework to the number of samples used for action selection (N_1) and value function evaluation (N_2). All experiments reported here utilize the dynamic weight controller with a target holding cost of C$_b$ = 5.

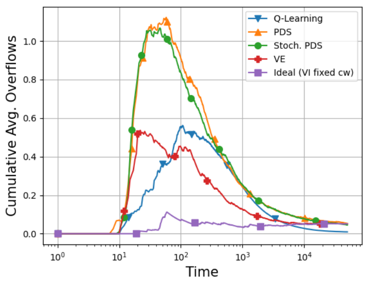
In Fig. 6, we show the cumulative average cost components versus time for $N_1 \in \{1, 5, 50, 200\}$ while keeping N_2 fixed



(a) Holding Cost

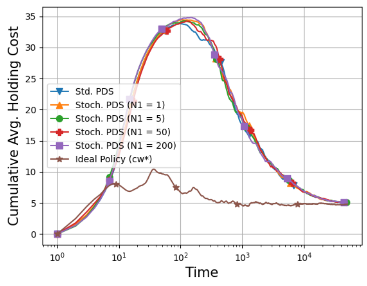


(b) Power cost

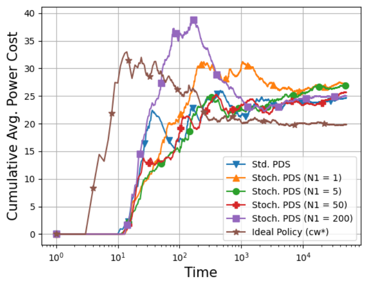


(c) Overflow cost

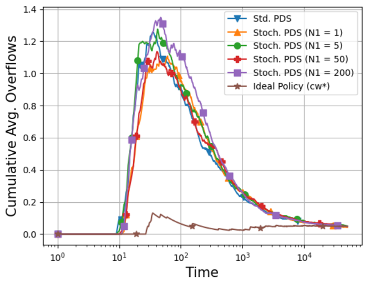
Fig. 5: Cumulative-average cost components under dynamic weight control with $C_b = 5$ ($N_1 = N_2 = 10$ for Stoch. PDS).



(a) Holding Cost (N_1 Varied)

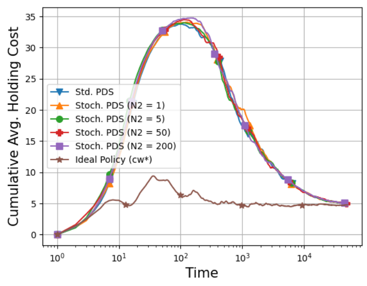


(b) Power Cost (N_1 Varied)

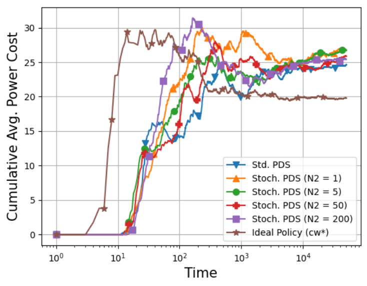


(c) Overflow Cost (N_1 Varied)

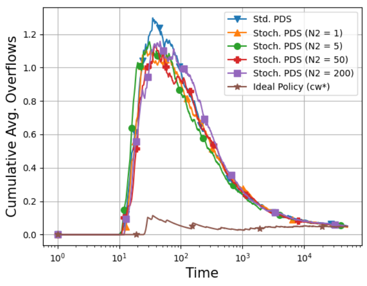
Fig. 6: Component-wise performance breakdown for varying action selection samples (N_1), with $N_2 = 10$.



(a) Holding Cost (N_2 Varied)



(b) Power Cost (N_2 Varied)



(c) Overflow Cost (N_2 Varied)

Fig. 7: Component-wise performance breakdown for varying value evaluation samples (N_2), with $N_1 = 10$.

at 10. While the holding and overflow costs evolve similarly for all values of N_1, there is a notable variation in the transient behavior of the power cost. In the first 1000 time steps, we observe the highest power cost at the extreme values of $N_1 = 1$ and $N_1 = 200$, while the power cost for intermediate values of $N_1 = 5$ and $N_1 = 50$ more closely tracks that of standard PDS learning. This behavior can be explained as follows. Early in the learning process, all learning algorithms have poor estimates of the value function. Therefore, even standard PDS learning experiences a fast rise in the power cost as it improves its value function estimate, hitting a local maximum power cost around 400 time steps. When $N_1 = 200$, stochastic PDS learning is likely to take the same action as standard PDS learning given the value function estimate, but since $N_2 = 10$, it takes longer to improve its value function estimate than standard PDS learning, hitting its maximum power cost around 2000 time steps. On the other hand, stochastic PDS learning with coarse sampling ($N_1 = 1$) introduces noise into the action selection process, so it sometimes selects better actions than standard PDS learning when the value estimate is poor early on. However, once it has a good value function estimate, stochastic PDS learning with $N_1 = 1$ is more likely to take suboptimal actions, so it takes longer to approach the steady-state performance of standard PDS learning. Meanwhile, using intermediate values of N_1 balances the benefits of moderately noisy action selection when the value function estimate is poor with the benefits of good action selection when the value function estimate is good. Importantly, despite the differences in transient behavior, stochastic PDS learning converges to the target constraint and approaches the steady-state performance of standard PDS learning for all N_1 with $N_2 = 10$.

In Fig. 7, we vary N_2 while keeping N_1 fixed at 10. Similar trends in transient behavior and steady-state performance are observed as in Fig. 6; however, the maximum cumulative average power is smaller with roughly 25% lower peaks compared to varying N_1. This suggests that noise in the value function estimate (N_2) is less critical than noise in the action selection process (N_1), as the former is averaged out over iterative updates while the latter directly dictates the system's trajectory. All combinations tested converge to the target constraint and approach the steady-state performance of standard PDS learning.

We extend the proposed sampling idea to Virtual Experience (VE) learning by replacing the exact expectation terms used in VE action selection and value evaluation with sample-mean estimates, similar to (17)–(18). We refer to this variant as *Stoch-VE*. In this experiment, VE performs a full buffer-state sweep each update (i.e., all post-decision buffer states are updated), while the expectation terms are approximated using $N_1 = N_2 = N$ samples.

Figs. 8a–8c compare standard VE with exact expectations against Stoch-VE for $N \in \{1, 5, 20, 50\}$. Overall, Stoch-VE closely tracks the performance of exact VE across holding, power, and overflow costs. As N increases, the transient fluctuations slightly reduce, but the steady-state performance remains similar across all tests. This behavior is expected as VE already improves sample efficiency by updating many PDSs from each real experience sample, which reduces variance in the learned post-decision value function.

Interestingly, even the minimal configuration $N_1 = N_2 = 1$ performs competitively. This indicates that in our system, the post-decision transition distribution is sufficiently concentrated such that a small number of samples provides an adequate approximation for greedy action ranking.

3) Effect of Packet Loss Rate: We next evaluate the robustness of the proposed method by varying the packet loss rate $q \in \{10\%, 50\%\}$. Fig. 9 compares Stoch-PDS and Stoch-VE against the corresponding optimal policies, using the minimal sampling configuration ($N_1 = N_2 = 1$). Fig. 9a shows that as q increases both stochastic methods exhibit larger transient holding costs; however, Stoch-VE shows faster convergence to the target holding cost constraint than Stoch-PDS. Figs. 9b and 9c highlight the related trade-offs in cost components. The power cost increases with the packet loss rate q because more transmission power is required to achieve a given holding cost constraint using a less reliable channel. At the same time, higher q causes a transient increase in overflow events for Stoch-PDS and Stoch-VE, which then decays as the penalty weight rises and the policy transmits more aggressively. Remarkably, even with a single-sample approximation ($N_1 = N_2 = 1$), both track the performance of the Ideal Policy in steady-state, while Stoch-VE converges faster than Stoch-PDS at the cost of higher computation.

4) Convergence Evaluation: Here, we evaluate learned value functions of each method over time using the following *weighted percent error* (WPE) metric [12], [13]:

$$e_t \triangleq \sum_{s \in \mathcal{S}} p^\star(s) \left| \frac{V^\star(s) - V_t(s)}{V^\star(s)} \right|, \tag{19}$$

where $V^\star(s)$ is the optimal value function with fixed optimal cost weight and $V_t(s)$ is the estimate available at the start of slot t under dynamic weight, and $p^\star(s)$ denotes the stationary probability of being in state s under the optimal policy $\pi^\star$.

Fig. 10 illustrates the WPE evolution across all algorithms with packet loss rate $q = 10\%$. The results demonstrate that the physics-informed methods reduce WPE significantly faster than the data-driven baseline. While VE exhibits the most rapid decrease and the lowest error, descending to approximately 5-10% by the end of the horizon, Stoch-VE with a single-sample approximation achieves nearly identical performance (about 5-10% weighted percent error). Stoch-PDS with single-sample ($N1 = N2 = 1$) achieves performance comparable to PDS; Both converge to nearly 25–30% error with similar speed, showing the efficiency of the proposed method. Finally, Q-learning maintains the highest error (45-50%) over the entire training period.

B. Complexity Analysis

Here, we summarize the computational complexity of the evaluated algorithms. Let $|\mathcal{B}|$ denote the number of buffer states, $|\mathcal{H}|$ the number of channel states, and $|\mathcal{A}|$ the number of actions. Following the notation in [5], let $|\tilde{\mathcal{S}}|$ denote the number of post-decision states involved in the expectation term, and let $|\Sigma|$ denote the number of virtual states updated per slot under Virtual Experience (VE). Therefore, $|\tilde{\mathcal{S}}| = |\mathcal{B}|$ and the number of virtual updates per slot is $|\Sigma| = |\mathcal{B}_{\mathrm{VE}}| = |\mathcal{B}|$.

1) Q-learning: Action selection is greedy over $|\mathcal{A}|$ actions, hence complexity can be described as $\mathcal{O}(|\mathcal{A}|)$ per slot. The update requires evaluating $\min_{a'} Q(s', a')$, which is also $\mathcal{O}(|\mathcal{A}|)$.

2) Standard PDS and VE: Both PDS and VE require evaluating an expected post-decision value for each candidate action. This costs $\mathcal{O}(|\tilde{\mathcal{S}}||\mathcal{A}|)$ for action selection. Standard PDS learning updates only one PDS per slot, so its learning update remains $\mathcal{O}(|\tilde{\mathcal{S}}||\mathcal{A}|)$. VE performs $|\Sigma|$ virtual updates per slot ($|\Sigma| = |\mathcal{B}_{\mathrm{VE}}|$ in our system), which yields a learning update complexity $\mathcal{O}(|\Sigma||\tilde{\mathcal{S}}||\mathcal{A}|)$.

3) Stochastic PDS and VE: Stoch-PDS replaces expectation terms by sample averages. With N_1 samples used for action selection and N_2 samples used for value evaluation, the action-selection complexity becomes $\mathcal{O}(|\mathcal{A}|N_1)$ and the value-evaluation becomes $\mathcal{O}(|\mathcal{A}|N_2)$ where $N_1, N_2 \ll |\mathcal{S}|$. Stoch-VE combines VE with sampling-based evaluation. Action selection at the realized state costs $\mathcal{O}(|\mathcal{A}|N_1)$. In the VE update, $|\Sigma|$ virtual updates are performed per slot; each update requires value evaluation based on N_2 samples, leading to $\mathcal{O}(|\Sigma||\mathcal{A}|N_2)$ for the learning update. It is also important to note that the expectation calculations in standard PDS and VE require multiplication operations while stochastic PDS and VE rely on addition operations (and only one multiplication with

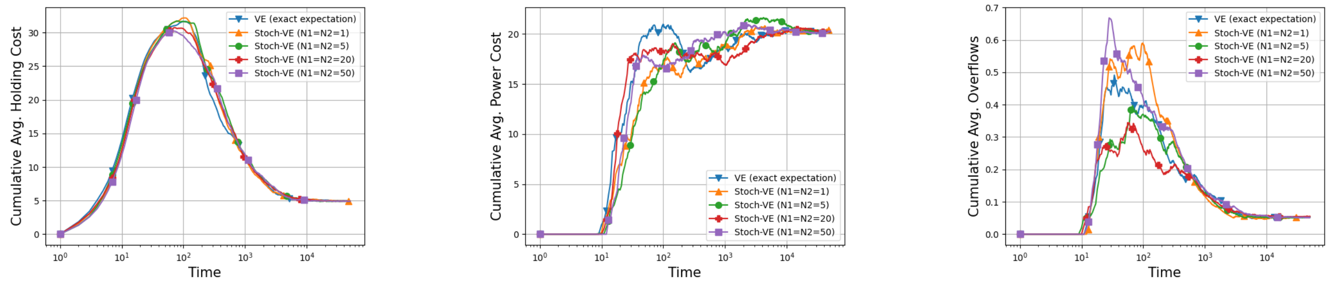


Fig. 8: Component-wise performance breakdown for Standard VE vs. Stoch-VE with $N_1 = N_2 = N$ samples.

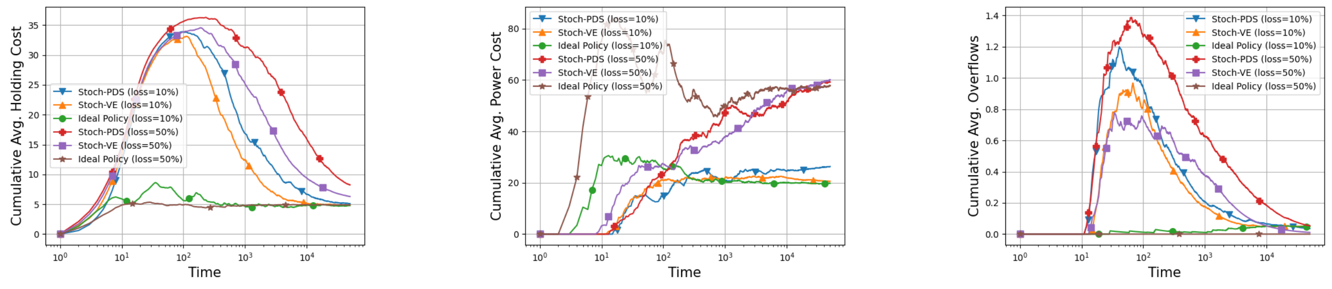


Fig. 9: Component-wise performance of Stoch-PDS vs. Ideal Policy for different packet loss rates with $N_1 = N_2 = 1$ sample.

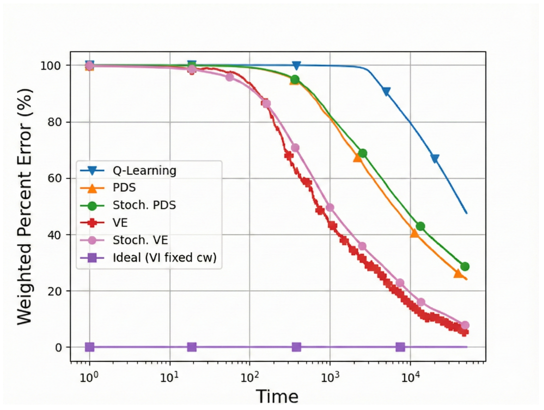


Fig. 10: Evolution of Weighted Percent Error vs. Time for all methods under target budget $C_b = 5$.

TABLE II: Computational complexity of evaluated algorithms.

Method	Action selection	Learning update
Q-learning	$\mathcal{O}(\|\mathcal{A}\|)$	$\mathcal{O}(\|\mathcal{A}\|)$
Standard PDS	$\mathcal{O}(\|\tilde{\mathcal{S}}\|\|\mathcal{A}\|)$	$\mathcal{O}(\|\tilde{\mathcal{S}}\|\|\mathcal{A}\|)$
Virtual Experience (VE)	$\mathcal{O}(\|\tilde{\mathcal{S}}\|\|\mathcal{A}\|)$	$\mathcal{O}(\|\Sigma\|\|\tilde{\mathcal{S}}\|\|\mathcal{A}\|)$
Stoch-PDS (N_1, N_2)	$\mathcal{O}(N_1\|\mathcal{A}\|)$	$\mathcal{O}(N_2\|\mathcal{A}\|)$
Stoch-VE (N_1, N_2)	$\mathcal{O}(N_1\|\mathcal{A}\|)$	$\mathcal{O}(\|\Sigma\|N_2\|\mathcal{A}\|)$

$1/N$). Hence, even if $N_1 = N_2 = |\tilde{S}|$, the stochastic versions are still significantly more efficient to implement.

VI. Conclusion

Reinforcement learning is an effective method to learn decision policies in unknown and dynamic wireless environments. While "physics-informed" Post-Decision State (PDS) learning significantly improves data efficiency by leveraging known system models, its high computational cost makes it infeasible to deploy on resource-constrained devices. In this paper, we addressed this challenge by proposing stochastic PDS learning, an algorithm that replaces expectation calculations with efficient stochastic sampling and yields high sample efficiency with reduced computational overhead. We further demonstrated that this sampling approach extends effectively to Virtual Experience (VE) learning. Simulations confirmed that stochastic PDS and stochastic VE learning consistently outperform benchmarks such as Q-learning. In particular, stochastic VE closely tracking VE achieving 5–10% weighted percent error, while stochastic PDS attains performance comparable to exact PDS (converging to 25–30% error) at substantially lower computational cost. Notably, both stochastic methods remain robust even with a single sample approximation. Surprisingly, the algorithms maintain robust performance even when using a single stochastic sample to approximate the expectations.

In future work, we plan to analytical prove the convergence of the sample-based PDS and VE algorithms, and analytically derive their sample efficiencies and convergence rates. We also plan to explore other, more involved value functions and problem formulations that could capture the end-to-end quality of experience of emerging application settings of key 5G/6G verticals such as mobile extended reality [20], [21].

References

[1] X. Lei, Z. Yang, J. Yu, J. Zhao, Q. Gao, and H. Yu, "Data-driven optimal power flow: A physics-informed machine learning approach," *IEEE Trans. Power Sys.*, vol. 36, no. 1, pp. 346–354, 2020.

[2] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, vol. 1, no. 1, pp. 1–34, 2020.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[4] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, 2008.

[5] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Trans. Signal Process.*, vol. 59, no. 12, pp. 6262–6266, 2011.

[6] F. Fu and M. van der Schaar, "Structure-aware stochastic control for transmission scheduling," *IEEE Trans. Veh. Technol.*, vol. 61, no. 9, pp. 3931–3945, 2012.

[7] N. Sharma, N. Mastronarde, and J. Chakareski, "Accelerated structure-aware reinforcement learning for delay-sensitive energy harvesting wireless sensors," *IEEE Trans. Signal Process.*, vol. 68, 2020.

[8] J. Zhang, X. He, and H. Dai, "Blind post-decision state-based reinforcement learning for intelligent IoT," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10 605–10 620, 2023.

[9] X. He, R. Jin, and H. Dai, "Deep PDS-learning for privacy-aware offloading in MEC-enabled IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4547–4555, 2018.

[10] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 375–388, 2020.

[11] J. Sun, N. Sharma, J. Chakareski, N. Mastronarde, and Y. Lao, "Hardware acceleration for postdecision state reinforcement learning in IoT systems," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9889–9903, 2022.

[12] A. Corra, N. Mastronarde, and J. Chakareski, "Low-complexity physics-informed reinforcement learning using post-decision states with stochastic sampling," in *ICC 2025 - IEEE International Conference on Communications*, 2025, pp. 4559–4564.

[13] N. Mastronarde and M. van der Schaar, "Joint physical-layer and system-level power management for delay-sensitive wireless communications," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, pp. 694–709, 2013.

[14] E. Delfani and N. Pappas, "Optimizing version aoi in energy-harvesting iot: Model-based and learning-based approaches," *arXiv preprint arXiv:2510.00904*, 2025.

[15] W. Yang, X. Chi, L. Zhao, and Z. Xiong, "Qoe-based mec-assisted predictive adaptive video streaming for on-road driving scenarios," *IEEE Wireless Communications Letters*, vol. 10, no. 11, pp. 2552–2556, 2021.

[16] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Nashua, NH: Athena Scientific, Jan. 1996.

[17] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[18] D. Bertsekas, *Convex Optimization Algorithms*, 2nd ed. Belmont, MA: Athena Scientific, Feb. 2015.

[19] M. Towers and et. al., "Gymnasium," Mar. 2023. [Online]. Available: https://zenodo.org/record/8127025

[20] J. Chakareski, M. Khan, and M. Yuksel, "Towards enabling next generation societal virtual reality applications for virtual human teleportation," *IEEE Signal Processing Magazine*, vol. 39, no. 5, pp. 22–41, Sep. 2022.

[21] S. Gupta, J. Chakareski, and P. Popovski, "mmWave networking and edge computing for scalable 360-degree video multi-user virtual reality," *IEEE Trans. Image Processing*, vol. 32, pp. 377–391, 2023.