Delay-Sensitive Energy-Harvesting Wireless Sensors: Optimal Scheduling, Structural Properties, and Approximation Analysis

Nikhilesh Sharma, Nicholas Mastronarde, *Senior Member, IEEE*, and Jacob Chakareski, *Senior Member, IEEE*

***Abstract*— We consider an energy harvesting sensor transmitting latency-sensitive data over a fading channel. We aim to find the optimal transmission scheduling policy that minimizes the packet queuing delay given the available harvested energy. We formulate the problem as a Markov decision process (MDP) over a state-space spanned by the transmitter's buffer, battery, and channel states, and analyze the structural properties of the resulting optimal value function, which quantifies the long-run performance of the optimal scheduling policy. We show that the optimal value function (i) is non-decreasing and has increasing differences in the queue backlog; (ii) is non-increasing and has increasing differences in the battery state; and (iii) is submodular in the buffer and battery states. Taking advantage of these structural properties, we derive an approximate value iteration algorithm that provides a controllable tradeoff between approximation accuracy, computational complexity, and memory, and we prove that it converges to a near-optimal value function and policy. Our numerical results confirm these properties and demonstrate that the resulting scheduling policies outperform a greedy policy in terms of queuing delay, buffer overflows, energy efficiency, and sensor outages.**

***Index Terms*— Markov decision processes, energy harvesting, latency-sensitive wireless sensing, approximate dynamic programming, structural properties.**

I. Introduction

ENERGY-CONSTRAINED wireless sensors often operate in challenging environments featuring dynamic channel conditions and latency-sensitive data sources. Increasingly, such sensors also include energy harvesting capabilities, allowing them to operate autonomously using energy in the environment (e.g., ambient light or RF energy [2]). Emerging applications of this nature include real-time remote visual sensing, the Internet of Things (IoT), body sensor networks, and mobile virtual and augmented reality [3]–[7]. Critical to their success is the ability to deliver the captured data in a timely manner.

A key step towards successful deployment of such systems is to understand their fundamental performance limits under different operating conditions, the characteristics of the optimal decision policies that will achieve these limits, and how to efficiently compute the policies. We set out in this paper to make progress towards these foundational objectives. In particular, we consider an energy-harvesting sensor (EHS) transmitting delay-sensitive data over a fading channel. We aim to understand the structure of the optimal value function, which quantifies the long-run performance of the optimal scheduling policy that minimizes the packet queuing delay given the available harvested energy. We then leverage this knowledge to formulate an approximate value iteration algorithm that provides a controllable tradeoff between approximation accuracy, computational complexity, and memory. Our contributions are as follows:

- We formulate the delay-sensitive energy harvesting scheduling (DSEHS) problem as a Markov Decision Process (MDP) that takes into account the stochastic traffic load, harvested energy, and channel conditions experienced by the EHS.
- We show that the optimal value function (i) is non-decreasing and has increasing differences in the queue backlog; (ii) is non-increasing and has increasing differences in the battery state; and (iii) is submodular in the buffer and battery states.
- We show that, owing to its structure, the optimal value function can be approximated with a bounded error as a piece-wise planar function, which upper bounds the optimal value.
- We derive a low-complexity value iteration algorithm that operates on the approximated value function and prove that it converges to a near optimal solution.
- Our numerical results confirm the value function's structural properties, demonstrate the efficacy of our proposed value iteration algorithm, and show that the resulting scheduling policies outperform a so-called greedy policy in terms of queuing delay, buffer overflows, energy efficiency, and sensor outages.

The rest of the paper is organized as follows. We review related work in Section II. We introduce our system model in Section III and formulate the DSEHS problem in Section IV.

Manuscript received July 5, 2019; revised October 8, 2019; accepted November 15, 2019. Date of publication November 28, 2019; date of current version April 16, 2020. The work of N. Sharma and N. Mastronarde was supported in part by the National Science Foundation (NSF) under Award ECCS-1711335. The work of J. Chakareski was supported in part by the NSF under Award CCF-1528030, Award ECCS-1711592, Award CNS-1836909, and Award CNS-1821875, and by research gifts and an Adobe Data Science Award from Adobe Systems. This article was presented in part at the 2018 IEEE International Conference on Communications [1]. The associate editor coordinating the review of this article and approving it for publication was G. Iosifidis. *(Corresponding author: Nikhilesh Sharma.)*

N. Sharma and N. Mastronarde are with the Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260 USA (e-mail: nsharma9@buffalo.edu; nmastron@buffalo.edu).

J. Chakareski is with the Ying Wu College of Computing, New Jersey Institute of Technology, Newark, NJ 07103 USA (e-mail: jakov@jakov.org).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2019.2956510

Next, we analyze the structural properties of the DSEHS problem in Section V, formulate our approximate value iteration algorithm in Section VI, and present our numerical results in Section VII. Finally, we conclude the paper in Section VIII, while rigorous proofs of the lemmas and propositions introduced throughout the paper are provided in the Appendix.

II. Related Work

Although EHSs can operate autonomously without the need to change their batteries, the stochastic nature of harvested energy sources poses new challenges in sensor power management, transmission power allocation, and transmission scheduling. This has driven a growing body of literature to address these challenges.

One important body of work focuses on computing optimal transmission policies for EHSs [8]–[10]. Gurakan and Ulukus [8] consider a multi-access channel with two EHSs and derive the optimal offline transmission power and rate allocations that maximize the sum rate, given known energy and traffic arrival processes. Lu *et al.* [9] formulate a throughput-optimal channel selection policy for EHSs operating as secondary users in a cognitive radio network. Gunduz *et al.* [10] identify MDPs [11] as a useful tool for optimizing EHSs in unpredictable environments with only causal information about the past and present. Though these studies identify numerous techniques for calculating optimal policies, they do not provide general insights into their structures.

Another body of work focuses on characterizing the structure of optimal transmission policies for EHSs [4], [12]–[16]. Numerous studies have shown that optimal power allocation policies for EHSs have various water-filling structures [12]–[14]. Other types of structural results are derived in [15], [16]. Michelusi *et al.* [17] formulate the problem of maximizing the average importance of transmitted data as an MDP, and show that the EHS should only transmit data having importance above a threshold that is a decreasing function of the energy level. Aprem *et al.* [16] formulate outage-optimal power control policies for EHSs, showing that the optimal policy for the underlying MDP is a threshold function in the battery state for the special case of binary transmission power levels.

In our recent work [1], we formulated the DSEHS problem as an MDP and analyzed its structural properties. Unlike [4], [12]–[16], which focus on the policy's structure, [1] focuses on the value function's structure. The present paper extends [1] to include (i) a more general system model that allows multiple packets to be transmitted in each time slot (in [1], we only considered binary scheduling actions); (ii) full derivations of the value function's structural properties; (iii) a novel low-complexity value iteration algorithm that converges to a near-optimal (piece-wise planar) approximation of the value function; and (iv) more extensive simulation results.

Aside from our prior work, the most closely related work to the present paper is [18], in which the authors formulate a delay-optimal energy management problem (similar to the DSEHS) as an MDP. However, they determine that computing the delay-optimal policy using value iteration is too computationally intensive. In this paper, as noted above, we propose a low-complexity approximate value iteration algorithm to compute near-optimal policies by exploiting new structural properties of the optimal value function, which were not presented in [18].

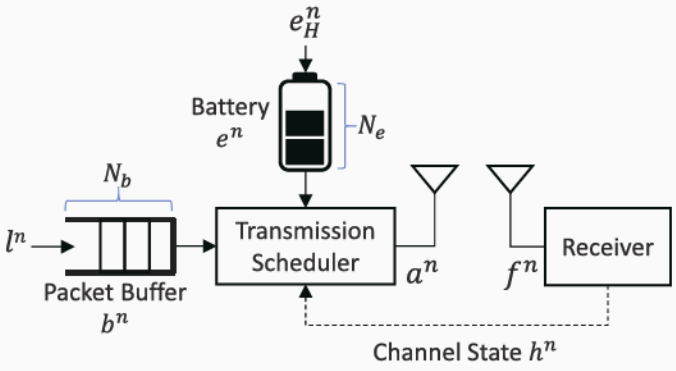


Fig. 1. System block diagram.

III. Wireless Sensor Model

We consider a time-slotted single-input single-output energy harvesting sensor that transmits latency-sensitive data over a fading channel. The system model is depicted in Fig. 1. The system comprises two buffers: a packet buffer with size $N_b > 0$ and an energy buffer (battery) with size $N_e > 0$. We assume that time is divided into slots with length ΔT (s) and that the system's state in the n-th time slot is denoted by $s^n \triangleq (b^n, e^n, h^n) \in \mathcal{S}$, where $b^n \in \mathcal{S}_b = \{0, 1, \ldots, N_b\}$ is the packet buffer state (i.e., the number of backlogged data packets), $e^n \in \mathcal{S}_e = \{0, 1, \ldots, N_e\}$ is the battery state (i.e., the number of available energy packets), and $h^n \in \mathcal{S}_h$ is the channel fading state. At the start of the nth time slot, the transmission scheduler observes the system's state, s^n, and takes the scheduling action $a^n \in \mathcal{A} = \{0, 1, \ldots, N_a\}$, where $N_a < \infty$ denotes the maximum number of packets that can be transmitted. Please note that, at times when the context is clear, we will omit the superscript n of the involved variables, to make the notation less cumbersome.

Channel model: We assume a block-fading channel, such that the channel is constant during each time slot but may change from slot-to-slot. Similar to prior work [12], [18]–[21], we assume that the channel state $h^n \in \mathcal{S}_h$ is known to the transmitter at the start of each time slot, that $\mathcal{S}_h$ denotes a finite set of N_h channel states, and that the evolution of the channel state can be modeled as a Markov chain with transition probability function $P^h(h'|h)$.

Physical layer model: We assume that the physical layer transmits at a data rate β^n/T_s (bits/s), where β^n is the number of bits per symbol determined by the modulation scheme and T_s (s) is the symbol duration. Therefore, in order to transmit a^n packets of size L (bits) in ΔT (s), the modulation scheme must be selected such that

$$\beta^n = \lceil a^n L T_s / \Delta T \rceil \quad \text{(bits/symbol)}, \tag{1}$$

where $\lceil \cdot \rceil$ denotes the ceiling operator.

Similar to [19], [20], [22], we set a target bit error probability (BEP), BEP_{target}, for all transmissions. We assume that,

given the channel state and target BEP, the transmission power is a non-decreasing function of the scheduling action a^n, i.e.,

$$P_{\text{TX}}^n = P_{\text{TX}}(h^n, a^n; BEP_{\text{target}}) \quad \text{(W)}. \tag{2}$$

This assumption holds for typical modulation schemes, e.g., M-ary PSK and M-ary QAM [23, Table 6.1]. Also similar to [19], [20], [22], our formulation does not include coding; however, it can be integrated by modifying (1) and (2).

Energy harvesting model: We assume that battery energy is stored in the form of energy packets as in, e.g., [16], [19]. Let $e_H^n \in \mathcal{E} = \{0, 1, \ldots, M_e\}$ denote the number of energy packets that are available for harvesting in the nth time slot and let $P^{e_H}(e_H)$ denote the energy packet arrival distribution. Note that $P^{e_H}(e_H)$ allows us to account for the fact that it may take multiple slots to harvest one energy packet worth of energy. Energy packets that arrive in time slot n can be used in future time slots. Thus, the battery state at the start of time slot $n+1$ can be found through the following recursion:

$$e^{n+1} = \min(e^n - e_{\text{TX}}^n + e_H^n, N_e), \tag{3}$$

where $e_{\text{TX}}^n = e_{\text{TX}}(h^n, a^n; BEP_{\text{target}})$ denotes the number of energy packets consumed in time slot n given the channel state h^n, scheduling action a^n, and target BEP. For simplicity, we assume that the transmission energy e_{TX}^n is an integer multiple of energy packets, such that.

$$\begin{aligned} e_{\text{TX}}^n &= e_{\text{TX}}(h^n, a^n; BEP_{\text{target}}) \\ &= \lceil P_{\text{TX}}(h^n, a^n; BEP_{\text{target}}) \Delta T \rceil \text{ (energy packets)}. \end{aligned} \tag{4}$$

Note that the transmission action a^n in time slot n cannot use more energy than is available in the battery, i.e., $e_{\text{TX}}(h^n, a^n; BEP_{\text{target}}) \leq e^n$. For notational simplicity, we will omit the transmission energy's dependence on the target BEP in the remainder of the paper.

Given the current state $s = (b, e, h)$ and action a, the probability of observing battery state e' in the next slot is:

$$P^e(e'|[e,h],a) = \mathbb{E}_{e_H}[\mathbb{I}_{\{e' = \min(e - e_{\text{TX}}(h,a) + e_H, N_e)\}}], \tag{5}$$

where $\mathbb{E}_{e_H}[\cdot]$ denotes an expectation over the energy packet arrival distribution and $\mathbb{I}_{\{\cdot\}}$ is an indicator variable that is set to 1 when $\{\cdot\}$ is true and is set to 0 otherwise.

Traffic model: Let $l^n \in \mathcal{L} = \{0, 1, \ldots, M_l\}$ denote the number of data packets generated by the sensor in the nth time slot and let $P^l(l)$ denote the data packet arrival distribution. The buffer state in slot $n+1$ can be found through the following recursion:

$$b^{n+1} = \min(b^n - f^n + l^n, N_b), \tag{6}$$

where $f^n = f(a^n; BEP_{\text{target}}) \leq a^n \leq b^n$ is the number of packets transmitted successfully in time slot n. Note that new packet arrivals, and packets that are not successfully received, must be (re)transmitted in a future time slot. Assuming independent and identically distributed (i.i.d.) bit errors, f^n can be modeled as a binomial random variable with conditional probability mass function $P^f(f|a; BEP_{\text{target}}) = \text{Bin}(a, 1-q) = \binom{a}{f}(1-q)^f q^{a-f}$, $f = 0, 1, \ldots, a$, where $q = 1 - (1 - BEP_{\text{target}})^L$ is the packet loss rate (PLR) for a packet of size L (bits). We refer to $P^f(f|a; BEP_{\text{target}})$ as the *goodput distribution* because it denotes the distribution over the number of *correctly received* packets in a time slot. For notational simplicity, hereafter we omit the goodput distribution's dependence on the target BEP.

Given the current state $s = (b, e, h)$ and action a, the probability of observing buffer state b' in the next time slot is:

$$P^b(b'|[b,h],a) = \mathbb{E}_{f,l}[\mathbb{I}_{\{b' = \min(b - f + l, N_b)\}}]. \tag{7}$$

where $\mathbb{E}_{f,l}[\cdot]$ denotes an expectation over the goodput and data packet arrival distributions.

Remark on Markovian data and energy packet arrivals: We note that the structural properties derived in Section V apply under both i.i.d. and Markovian data and energy packet arrivals. Thus, the approximate value iteration algorithm that we formulate in Section VI can be applied to either case.

IV. The Delay-Sensitive Energy-Harvesting Scheduling (DSEHS) Problem

Let $\pi : \mathcal{S} \rightarrow \mathcal{A}$ denote a *policy* that maps states to actions. The objective of the DSEHS problem is to determine the optimal policy π^* that minimizes the average packet queuing delay given the available energy. However, this does not mean that the policy should greedily transmit packets whenever there is enough energy to do so. Instead, it may be beneficial to abstain from transmitting packets in bad channel states and wait to transmit them in good channel states to conserve scarce harvested energy. On the other hand, the policy should not be too conservative. Instead, if the battery is (nearly) full, expending energy by transmitting packets will make room for more harvested energy, which otherwise would be lost due to the finite battery size. To balance these considerations, we formulate the DSEHS problem as an MDP [11].

We define a *buffer cost* to penalize large queue backlogs. Formally, we define the buffer cost as the sum of the *holding cost*[1] and the expected *overflow cost* with respect to the arrival and goodput distributions, i.e.,

$$c([b,h],a) = b + \mathbb{E}_{f,l}[\eta \max\{b - f + l - N_b, 0\}], \tag{8}$$

where the holding cost is equal to the buffer backlog, which is proportional to the queuing delay by Little's theorem [25]. The overflow cost imposes a penalty η for each dropped packet.

A *value function*, $V^\pi(s)$, estimates how good (or bad) it is for the EHS to be in a certain state while following the policy π. We formally define $V^\pi(s)$ as,

$$V^\pi(s) = \mathbb{E}\left[\sum_{n=0}^{\infty} (\gamma)^n c(s^n, \pi(s^n)) | s = s^0\right],$$

where $\gamma \in [0, 1)$ is the discount factor; $(\gamma)^n$ denotes the discount factor to the nth power; $c(s^n, \pi(s^n))$ denotes the buffer cost, as in (8), associated with being in state s^n and taking action $a^n = \pi(s^n)$; and the expectation is taken over the sequence of states, which is governed by the controlled Markov chain with transition probabilities

[1]The term "holding cost" comes from the operations research literature where it is commonly used in inventory control problems [24]. It represents the cost associated with storing inventory that remains unsold. In communication systems, it is interpreted as the cost for keeping a packet queued that has not yet been transmitted [20], [21].

$P(s'|s,a) = P^b(b'|[b,h],a)P^e(e'|[e,h],a)P^h(h'|h)$. We can rewrite $V^\pi(s)$ recursively by taking advantage of the one-step transition probability function to represent the expected future costs:

$$V^\pi(s) = c(s,\pi(s)) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,\pi(s))V^\pi(s')$$

Formally, the DSEHS problem's objective is to determine the scheduling policy that solves the following optimization:

$$\min_{\pi\in\Pi} V^\pi(s), \tag{9}$$

where Π denotes the set of possible policies. The optimal solution to (9) satisfies the following Bellman equation, $\forall s \in \mathcal{S}$:

$$\begin{aligned} V^*(s) &= \min_{a\in\mathcal{A}(s)} \left\{ c(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V^*(s') \right\}, \\ &= \min_{a\in\mathcal{A}(b,e,h)} \{c([b,h],a) + \gamma \mathbb{E}_{l,f,e_H,h'} \\ &\quad [V^*(\min(b-f+l,N_b), \\ &\quad \min(e-e_{\text{TX}}(h,a)+e_H,N_e),h')]\} \\ &\triangleq \min_{a\in\mathcal{A}(s)} Q^*(s,a), \forall s\in\mathcal{S} \end{aligned} \tag{10}$$

where

$$\mathcal{A}(b,e,h) = \left\{ a \in \{0,\ldots,N_a\} : a \le b \text{ and } e_{TX}(h,a) \le e \right\} \tag{11}$$

is the set of feasible actions given the buffer and battery states, $V^*(s)$ is the optimal *state-value function*, and $Q^*(s,a)$ is the optimal *action-value function* denoting the value of taking action a in state s and then following the optimal policy thereafter. The optimal policy $\pi^*(s)$, which gives the optimal action to take in each state, can be determined by taking the action in each state that minimizes the right-hand side (r.h.s.) of (10). Please note that we use the two representations of the cost, $c(s,a)$ and $c([b,h],a)$ interchangeably as they refer to the same quantity.

A. Post-Decision State Based Dynamic Programming

We will find it valuable throughout our analysis to work with so-called *post-decision states* (PDSs [20], [22], [26]) rather than conventional states because computing the value function using PDSs is less complex than computing it using conventional states (see Section VI-C). A PDS, $\widetilde{s} \triangleq (\widetilde{b},\widetilde{e},\widetilde{h}) \in \mathcal{S}$, denotes the state of the system after the controllable/known effects of the action, but before the uncontrollable dynamics occur [20]. In the DSEHS problem, $\widetilde{s}^n = (\widetilde{b}^n,\widetilde{e}^n,\widetilde{h}^n) = ([b^n - f^n],[e^n - e_{\text{TX}}(h^n,a^n)],h^n)$, is the PDS in time slot n. The buffer's PDS, $\widetilde{b}^n = b^n - f^n$, characterizes the buffer state after packets are transmitted, but before any new packets arrive; the battery's PDS, $\widetilde{e}^n = e^n - e_{\text{TX}}(h^n,a^n)$, characterizes the battery state after energy packet(s) are consumed, but before any new energy packets arrive; and the channel's PDS, $\widetilde{h}^n = h^n$, is the same as the channel state at time n. In other words, the PDS incorporates all of the known information about the transition from state s^n to state s^{n+1} after taking action a^n. Note that, although the realization of f^n is not known at the time the action is taken, its distribution P^f is known. As will become clear in (14), this is sufficient to include it in the buffer's PDS. Meanwhile, the uncontrollable dynamics in the transition from state s^n to s^{n+1}, i.e., the channel state transition from h^n to $h^{n+1} \sim P^h(\cdot|h^n)$, the data packet arrivals, $l^n \sim P^l(\cdot)$, and the energy packet arrivals, $e_H^n \sim P^{e_H}(\cdot)$, are not included in the PDS. Importantly, the next state can be expressed in terms of the PDS as follows:

$$\begin{aligned} s^{n+1} &= (b^{n+1},e^{n+1},h^{n+1}) \\ &= (\min(\widetilde{b}^n + l^n, N_b), \min(\widetilde{e}^n + e_H^n, N_e), h^{n+1}). \end{aligned} \tag{12}$$

Just as we defined a value function over the conventional states, we can define a PDS value function over the PDSs. Let $\widetilde{V}^*$ denote the optimal PDS value function. $\widetilde{V}^*$ and V^* are related by the following Bellman equations:

$$\begin{aligned} \widetilde{V}^*(\widetilde{s}) &= \eta \mathbb{E}_l[\max(\widetilde{b}+l-N_b,0)] \\ &\quad + \gamma \mathbb{E}_{l,e_H,h'}[V^*(\min(\widetilde{b}+l,N_b), \\ &\quad \min(\widetilde{e}+e_H,N_e),h')] \end{aligned} \tag{13}$$

$$V^*(s) = \min_{a\in\mathcal{A}(s)} \left\{ b + \mathbb{E}_f[\widetilde{V}^*(b-f, e-e_{TX}(h,a),h)] \right\}, \tag{14}$$

which are obtained by factorizing the Bellman equation in (10) using PDSs. Given $\widetilde{V}^*$, π^* can be found by taking the action in each state that minimizes the r.h.s. of (14). Note that, since the goodput distribution P^f is known, we can take an expectation of the PDS value function over the buffer's PDS $b-f$ in (14).

Algorithm 1 presents a value iteration algorithm for computing the optimal PDS value function. If the data packet arrival, energy harvesting, and channel dynamics are known and stationary, then PDS value iteration can be performed offline. If the dynamics are non-stationary, then the algorithm can be executed online at regular intervals to update the PDS value function as the dynamics change over time.[2] However, if the buffer and battery sizes are sufficiently large, then any tabular approach that requires computing and storing the value function for every single state (e.g., value iteration [27] or PDS value iteration) will become intractable as the size of the state space increases due to the curse of dimensionality [24]. Moreover, such tabular approaches cannot be applied if the buffer and battery sizes are infinite.

Although PDS value iteration is too complex to be implemented on an EHS in general, we leverage its iterative structure to derive several structural properties of the optimal PDS value function $\widetilde{V}^*(\widetilde{s})$ using mathematical induction in Section V. Subsequently, in Section VI, we propose a low-complexity value iteration algorithm that operates on a piece-wise planar approximation of the value function. We then prove that, owing to the derived structural properties, this algorithm converges to a near optimal solution.

V. Structural Properties

In this section, we analyze the structural properties of the optimal PDS value function $\widetilde{V}^*$. Such properties are important because they (i) provide insights into the optimization problem and the system being optimized; (ii) reveal ways in which

[2] In this situation, we also need to make empirical estimates of the data packet arrival, energy harvesting, and channel transition distributions online.

Algorithm 1 Post-Decision State Value Iteration

initialize $\widetilde{V}_0(\widetilde{b},\widetilde{e},\widetilde{h})=0$ for all $(\widetilde{b},\widetilde{e},\widetilde{h})\in\mathcal{S}$ and $m=0$.
repeat
  $\Delta\leftarrow 0$
  for $(b,e,h)\in\mathcal{S}$ **do**
    Update the value function:

$$V_m(b,e,h)\leftarrow\min_{a\in\mathcal{A}(s)}\left\{b+\mathbb{E}_f[\widetilde{V}_m(b-f,e-e_{TX}(h,a),h)]\right\} \tag{15}$$

  end for
  for $(\widetilde{b},\widetilde{e},\widetilde{h})\in\mathcal{S}$ **do**
    Update the PDS value function:

$$\widetilde{V}_{m+1}(\widetilde{b},\widetilde{e},\widetilde{h})\leftarrow\eta\mathbb{E}_l[\max(\widetilde{b}+l-N_b,0)] +\gamma\mathbb{E}_{l,e_H,h'}[V_m(\min(\widetilde{b}+l,N_b),\min(\widetilde{e}+e_H,N_e),h')] \tag{16}$$

    $\Delta\leftarrow\max(\Delta,|\widetilde{V}_m(\widetilde{b},\widetilde{e},\widetilde{h})-\widetilde{V}_{m+1}(\widetilde{b},\widetilde{e},\widetilde{h})|)$
  end for
  $m\leftarrow m+1$
until $\Delta<\theta$ (a small positive constant)

the solution can be represented compactly, with limited memory; and (iii) facilitate efficient computation of near optimal policies. We begin by introducing three important definitions. Then, in Section V-B, we analyze the properties of the cost and transition probability functions and, in Section V-C, we analyze key properties of the conventional value function. These properties are all needed to prove our main results about the PDS value function's structure, presented in Section V-D.

A. Preliminaries

Definition 1 (Integer Convex [21]): An integer convex function $f(n):\mathcal{N}\rightarrow\mathbb{R}$ on a set of integers $\mathcal{N}\in\{0,1,\ldots,N\}$ is a function that has increasing differences in n, i.e.,

$$f(n_1+m)-f(n_1)\leq f(n_2+m)-f(n_2) \tag{17}$$

for $n_1<n_2$, and $n_1,n_2,n_1+m,n_2+m\in\mathcal{N}$.

The second useful definition is that of stochastic dominance.

Definition 2 (Stochastic Dominance [21]): Let $\theta(x)$ be a random variable parameterized by some $x\in\mathbb{R}$. If $P(\theta(x_1)\geq a)\geq P(\theta(x_2)\geq a)$, for all $x_1\geq x_2$ and for all $a\in\mathbb{R}$, then we say that $\theta(x)$ is first-order stochastically increasing in x. For such $\theta(x)$, it holds:

$$\mathbb{E}[u(\theta(x_1))]\geq\mathbb{E}[u(\theta(x_2))], \tag{18}$$

for all non-decreasing functions $u(x)$. The reverse inequality holds for all non-increasing functions $u(x)$.

Lastly, we define the concept of a submodular function. Note that this definition has some similarities to, but is distinct from, the definition of a submodular set function [28].

Definition 3 (Submodular Function [11, §4.7.2]): A submodular function $f(x,y):\mathcal{X}\times\mathcal{Y}\rightarrow\mathbb{R}$ on sets of integers $\mathcal{X}\in\{0,1,\ldots,X\}$ and $\mathcal{Y}\in\{0,1,\ldots,Y\}$ is a function that has decreasing differences in (x,y), i.e., for $x^+\geq x^-$ and $y^+\geq y^-$

$$f(x^+,y^+)-f(x^+,y^-)\leq f(x^-,y^+)-f(x^-,y^-). \tag{19}$$

B. Properties of the Cost and Transition Probability Functions

We now present key properties of the cost and transition probability functions that we will need for our main results. Recall that the cost function defined in (8) does not directly depend on the battery state e, but that the action a is constrained to be in the set $\mathcal{A}(b,e,h)$ defined in (11). Therefore, when $a\notin\mathcal{A}(b,e,h)$ it is not possible to incur the cost $c([b,h],a)$. To capture this explicitly, we define an auxiliary cost function

$$d([b,e,h],a)=\begin{cases}c([b,h],a), & \text{if } b\geq a \text{ and } e\geq e_{TX}(h,a)\\ c([b,h],0), & \text{otherwise,}\end{cases} \tag{20}$$

where $c([b,h],a)$ is defined in (8). The auxiliary cost allows us to derive key properties of the cost with respect to the buffer and battery states, which are essential for our main results.

Lemma 1: The auxiliary cost $d([b,e,h],a)$ satisfies the following properties:

1) *The auxiliary cost is non-decreasing in b.*
2) *The auxiliary cost is non-increasing in e.*

Proof: The proof is given in Appendix VIII-A. □

Since the auxiliary cost function satisfies Lemma 1, the cost function $c([b,h],a)$, with $a\in\mathcal{A}(b,e,h)$, also satisfies it.

Lemma 2: The next battery state e' is first-order stochastically increasing in the current battery state e, i.e., $\sum_{e'\geq\bar{e}}P^e(e'|[e+1,h],a)\geq\sum_{e'\geq\bar{e}}P^e(e'|[e,h],a)$, $0\leq e<N_e$.

Proof: The proof is given in Appendix VIII-B. □

Lemma 3: The next buffer state b' is first-order stochastically increasing in the current buffer state b, i.e., $\sum_{b'\geq\bar{b}}P^b(b'|[b+1,h],a)\geq\sum_{b'\geq\bar{b}}P^b(b'|[b,h],a)$, $0\leq b<N_b$.

Proof: The proof is similar to that of Lemma 2. We omit it due to space limitations. □

Lemma 2 (resp. Lemma 3) implies that the next battery (resp. buffer) state has a higher probability of exceeding a threshold if the current battery (resp. buffer) state is larger.

C. Properties of the Conventional State Value Function

We now present key properties of the conventional state value function.

Lemma 4: The optimal value function $V^*(b,e,h)$ is non-decreasing in the buffer state b.

Proof: The proof is given in Appendix VIII-C. □

Lemma 5: The optimal value function $V^*(b,e,h)$ is non-increasing in the battery state e.

Proof: The proof is similar to that of Lemma 4. We omit it due to space limitations. □

The following lemma is needed for the inductive steps in our main results (Propositions 3, 4, and 5).

Lemma 6: The following properties are propagated from the PDS value function $\widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h})$ to the conventional value

function $V(b,e,h)$ through the Bellman equation defined in (14):

1) *If $\widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h})$ has increasing differences in $\widetilde{b}$, $V(b,e,h)$ has increasing differences in b.*
2) *If $\widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h})$ has increasing differences in $\widetilde{e}$, $V(b,e,h)$ has increasing differences in e.*
3) *If $\widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h})$ is submodular in $(\widetilde{b},\widetilde{e})$, then $V(b,e,h)$ is submodular in (b,e).*

Proof: The proof is given in Appendix VIII-D. □

D. Properties of the Post-Decision State Value Function

We now prove that the optimal PDS value function is non-decreasing and has increasing differences in the post-decision buffer state $\widetilde{b}$, is non-increasing and has increasing differences in the post-decision battery state $\widetilde{e}$, and has decreasing differences (i.e., is submodular) in $(\widetilde{b},\widetilde{e})$.

Proposition 1: The optimal PDS value function $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ is non-decreasing in $\widetilde{b}$.

Proof: The proof is given in Appendix VIII-E. □

Proposition 2: The optimal PDS value function $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ is non-increasing in $\widetilde{e}$.

Proof: The proof is similar to that of Proposition 1. We omit it due to space limitations. □

Proposition 3: For all PDS buffer states $\widetilde{b} < N_b - M_l$, where M_l is the maximum number of data packet arrivals in one time slot, $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ has increasing differences in $\widetilde{b}$, i.e.,

$$\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b}-1,\widetilde{e},\widetilde{h}) \leq \widetilde{V}^*(\widetilde{b}+1,\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}). \tag{21}$$

If the packet buffer size is infinite (i.e., $N_b = \infty$), then (21) holds for all $\widetilde{b} \in \mathcal{S}_b$.

Proof: The proof is given in Appendix VIII-F. □

Proposition 4: $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ has increasing differences in $\widetilde{e}$, i.e.,

$$\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e}-1,\widetilde{h}) \leq \widetilde{V}^*(\widetilde{b},\widetilde{e}+1,\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}). \tag{22}$$

Proof: The proof is given in Appendix VIII-G. □

Proposition 5: $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ is submodular in $(\widetilde{b},\widetilde{e})$, i.e.,

$$\widetilde{V}^*(\widetilde{b}+1,\widetilde{e}+1,\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e}+1,\widetilde{h}) \leq \widetilde{V}^*(\widetilde{b}+1,\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}). \tag{23}$$

Proof: The proof is given in Appendix VIII-H. □

Together, Propositions 1 and 3 imply that the marginal cost of holding an additional data packet increases with the queue backlog. We have empirically verified that Proposition 3 holds in practice for finite buffer sizes N_b as well. In Appendix VIII-F, we discuss in detail why we are unable to prove that Proposition 3 holds for all $\widetilde{b} \in \mathcal{S}_b$ in the case that $N_b < \infty$. Together, Propositions 2 and 4 imply that the marginal benefit of an additional energy packet decreases with the available battery energy. Finally, Proposition 5 implies that data packets and energy packets are *complementary*. That is, having more energy packets decreases the marginal cost of holding additional data packets (i.e., $\widetilde{V}^*(\widetilde{b}+1,\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h})$ decreases in $\widetilde{e}$). In other words, the marginal cost incurred by an additional data packet is smaller when more energy is available. Additionally, holding more data packets increases the marginal benefit of having additional energy packets (i.e., $\widetilde{V}^*(\widetilde{b},\widetilde{e},\widetilde{h}) - \widetilde{V}^*(\widetilde{b},\widetilde{e}+1,\widetilde{h})$ increases in $\widetilde{b}$). In other words, the marginal benefit of having an additional energy packet is greater when the buffer is more full.

VI. Approximate PDS Value Iteration

In this section, we propose to approximate the PDS value function as a piece-wise planar function. We then prove that, owing to the PDS value function's structural properties (see Section V-D), the resulting approximation error is bounded. Subsequently, we derive a low-complexity value iteration algorithm that operates on the approximated PDS value function and prove that it converges to an ϵ-optimal solution. Finally, we compare the computational and memory complexities of several value iteration algorithms.

A. Piece-Wise Planar Approximation

For each channel state $\widetilde{h} \in \mathcal{S}_h$, we construct a two-dimensional grid of post-decision buffer and battery states, $\mathcal{T}(\widetilde{h}) \subseteq \mathcal{S}_b \times \mathcal{S}_e$, on which we build a PDS value function approximation, $\hat{V}$. To allow for a spatially adaptive approximation on the buffer-battery plane, we use a *quadtree* data structure, such that its leaf node vertices $(\widetilde{b},\widetilde{e}) \in \mathcal{T}(\widetilde{h})$ define the grid. Each leaf of the quadtree is then divided into two triangles, which lie on intersecting planes. Together, the planes of all leaf nodes comprise the proposed piece-wise planar approximation as illustrated in Fig. 2.

Quadtree Construction: Let $\mathcal{T}$ denote a quadtree defined on the set of buffer-battery state pairs $\mathcal{S}_b \times \mathcal{S}_e$ within a bounding box (BB) defined as follows

$$\texttt{BB}(\mathcal{T}) = \{(b_-,e_-),(b_+,e_-),(b_-,e_+),(b_+,e_+)\}, \tag{24}$$

where $0 \leq b_- < b_+ \leq N_b$ and $0 \leq e_- < e_+ \leq N_e$. Note that, as in Fig. 2, we do not require $\mathcal{T}$ to span the entire buffer-battery plane because N_b and N_e may be very large (or infinite) and it is often unnecessary to accurately approximate the values at the extremes of the state space.

If a subtree $\mathcal{T}_{\text{sub}} \subseteq \mathcal{T}$ is a leaf node, then it can be further subdivided into four subtrees (children) spanning its northwest (NW), northeast (NE), southwest (SW), and southeast (SE) quadrants (see Fig. 2). We write $(b,e) \in \mathcal{T}$ if (b,e) is an element of $\mathcal{T}$'s bounding box or one of its children's bounding boxes, recursively down to all of its leaf nodes.

PDS Value Function Approximation: Let $\mathcal{T}(\widetilde{h})$ denote the quadtree used to approximate the PDS value function in channel state $\widetilde{h}$. We approximate the value of any PDS pair $(\widetilde{b},\widetilde{e}) \notin \mathcal{T}(\widetilde{h})$ using the values of PDS pairs $(\widetilde{b},\widetilde{e}) \in \mathcal{T}(\widetilde{h})$. That is, instead of directly using the PDS value function $\widetilde{V}$, we use an approximate PDS value function $\hat{V}$, such that

$$\hat{V}(\widetilde{b},\widetilde{e},\widetilde{h}) = \begin{cases} \widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h}), & \text{if } (\widetilde{b},\widetilde{e}) \in \mathcal{T}(\widetilde{h}) \\ \texttt{approx}\big(\widetilde{V},\mathcal{T}(\widetilde{h}),[\widetilde{b},\widetilde{e},\widetilde{h}]\big), & \text{otherwise,} \end{cases} \tag{25}$$

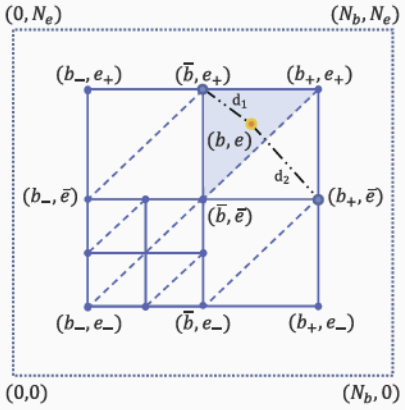


Fig. 2. Piece-wise planar approximation of the PDS value function. The dashed border represents the buffer-battery plane. The solid internal square represents the quadtree's bounding box. Each leaf node is divided into two triangles lying on intersecting planes. The PDS pair (b, e) is associated with the shaded NW triangle of the quadtree's NE leaf node because $d_1 < d_2$.

where $\texttt{approx}\big(\widetilde{V}, \mathcal{T}(\widetilde{h}), [\widetilde{b}, \widetilde{e}, \widetilde{h}]\big)$ approximates the value of PDS pair $(\widetilde{b}, \widetilde{e}) \notin \mathcal{T}(\widetilde{h})$ using the piece-wise planar approximation defined by the quadtree $\mathcal{T}(\widetilde{h})$. If the PDS pair $(\widetilde{b}, \widetilde{e})$ is inside the quadtree's bounding box (as defined in (24) and illustrated in Fig. 2), then the approximation function first associates $(\widetilde{b}, \widetilde{e})$ with the leaf node that contains it using a recursive search from the root. Subsequently, it associates $(\widetilde{b}, \widetilde{e})$ with the triangle that contains it. This triangle forms the approximating plane for our piecewise-planar approximation, as illustrated in Fig. 2. Specifically, let d_1 and d_2 denote the distances between $(\widetilde{b}, \widetilde{e})$ and the leaf node's NW and SE vertices, respectively. If $d_1 < d_2$, then we associate $(\widetilde{b}, \widetilde{e})$ with the NW triangle; otherwise, we associate it with the SE triangle. Finally, we solve the equation of the approximating plane to obtain the approximate value, $\hat{V}(\widetilde{b}, \widetilde{e}, \widetilde{h})$. On the other hand, PDS pairs that are outside the bounding box are not within any leaf node and must be associated with their *nearest* leaf node. Subsequently, they are associated with the leaf node's NW or SE triangle, and their values are approximated by the plane defined by this approximating triangle, similar to the PDS pairs inside the bounding box. Lastly, note that we write $\texttt{approx}\big(V, \mathcal{T}(h), [b, e, h]\big)$ to denote the approximate value of conventional state pair $(b, e) \notin \mathcal{T}(h)$.

At this point, we find it appropriate to introduce an operator $\mathcal{A}_{\mathcal{T}}$ that operates on the PDS value function, $\widetilde{V}$, to give a piece-wise planar approximation, $\hat{V}$, using the quadtree $\mathcal{T}$. Eq. (25) can then be represented succinctly in terms of this operator as $\hat{V} = \mathcal{A}_{\mathcal{T}}\widetilde{V}$ (when approximating the conventional value function V, we can write $J = \mathcal{A}_{\mathcal{T}}V$, where J denotes the approximate value function). The following proposition shows that the resulting approximation error is bounded.

Proposition 6: Let $\widetilde{V}$ denote a PDS value function and let $\hat{V} = \mathcal{A}_{\mathcal{T}}\widetilde{V}$ denote its piece-wise planar approximation (25). Let $(\widetilde{b}, \widetilde{e})$ lie inside the triangle with vertices $\mathrm{x}_i = (b_i, e_i, \widetilde{V}(b_i, e_i, \widetilde{h}))$, for $i = 1, 2, 3$ in the quadtree $\mathcal{T}$. The error $\mathcal{A}_{\mathcal{T}}\widetilde{V} - \widetilde{V}$ is bounded as follows:

$$(\mathcal{A}_{\mathcal{T}}\widetilde{V})(\widetilde{b}, \widetilde{e}, \widetilde{h}) - \widetilde{V}(\widetilde{b}, \widetilde{e}, \widetilde{h}) \leq \delta, \tag{26}$$

Algorithm 2 Approximate Value Iteration

initialize $\mathcal{T}^0(\widetilde{h}) = \texttt{BB}(\mathcal{T})$, $\forall \widetilde{h} \in \mathcal{S}_h$; Approx. value $J(b, e, h) = 0$, $\forall (b, e) \in \mathcal{T}^0(\widetilde{h}), \widetilde{h} \in \mathcal{S}_h$; Approx. PDS value $\hat{V}(\widetilde{b}, \widetilde{e}, \widetilde{h}) = 0$, $\forall (\widetilde{b}, \widetilde{e}) \in \mathcal{T}^0(\widetilde{h}), \widetilde{h} \in \mathcal{S}_h$; and $m = 0$.
repeat
  $\Delta \leftarrow 0$
  for $(b, e) \in \mathcal{T}^m(\widetilde{h})$ **do**
    Update the approximate value function:

$$J_m(b, e, h) \leftarrow \min_{a \in \mathcal{A}(s)} \left\{ b + \mathbb{E}_f[\hat{V}_m(b - f, e - e_{TX}(h, a), h)] \right\} \tag{27}$$

  end for
  for $(\widetilde{b}, \widetilde{e}) \in \mathcal{T}^m(\widetilde{h})$ **do**
    Update the approximate PDS value function:

$$\hat{V}_{m+1}(\widetilde{b}, \widetilde{e}, \widetilde{h}) \leftarrow \eta \mathbb{E}_l[\max(\widetilde{b} + l - N_b, 0)] + \gamma \mathbb{E}_{l, e_H, h'}[J_m(\min(\widetilde{b} + l, N_b), \min(\widetilde{e} + e_H, N_e), h')] \tag{28}$$

    $\Delta \leftarrow \max(\Delta, |\hat{V}_m(\widetilde{b}, \widetilde{e}, \widetilde{h}) - \hat{V}_{m+1}(\widetilde{b}, \widetilde{e}, \widetilde{h})|)$
  end for
  for each leaf $\ell \in \mathcal{T}^m(\widetilde{h})$ and each triangle $i \in \ell$ **do**
    Calculate approximation error $\delta_{\ell i}$ as in (26)
  end for
  $\delta_{\max} \leftarrow \max_{\ell i} \delta_{\ell i}$ and $\ell_{\max} \leftarrow \arg\max_{\ell i} \delta_{\ell i}$
  if $\delta_{\max} > \delta_{target}$ **then**
    Subdivide leaf $\ell_{\max} \in \mathcal{T}^m(\widetilde{h})$ to get $\mathcal{T}^{m+1}(\widetilde{h})$
  end if
  $m \leftarrow m + 1$
until $\Delta < \theta$ (a small positive constant)

where $\delta = \max_{i \in \{1,2,3\}} \widetilde{V}(b_i, e_i, \widetilde{h}) - \min_{i \in \{1,2,3\}} \widetilde{V}(b_i, e_i, \widetilde{h})$ depends on the PDS $\widetilde{s} = (\widetilde{b}, \widetilde{e}, \widetilde{h})$ and is equal for all PDSs that lie inside the same approximating triangle. Hereafter, we refer to δ as the single-step approximation error.

Proof: The result follows from Propositions 3 and 4. In particular, since $\widetilde{V}$ has increasing differences in $\widetilde{b}$ and $\widetilde{e}$, the plane defined by the approximating triangle provides an upper bound on the true value function. Additionally, since $\widetilde{V}$ and $\hat{V}$ are non-decreasing in $\widetilde{b}$ and non-increasing in $\widetilde{e}$, they are bounded below by $\min_{i \in \{1,2,3\}} \widetilde{V}(b_i, e_i, \widetilde{h})$ and above by $\max_{i \in \{1,2,3\}} \widetilde{V}(b_i, e_i, \widetilde{h})$ for all $(\widetilde{b}, \widetilde{e})$ that lie in the approximating triangle. Eq. (26) immediately follows. □

B. Approximate PDS Value Iteration (AVI) Algorithm

The proposed approximate value iteration (AVI) algorithm is presented in Algorithm 2. Let $\mathcal{T}^m(\widetilde{h})$ denote the quadtree used to approximate the PDS value function in channel state $\widetilde{h}$ during iteration m of the AVI algorithm. At the start of the AVI algorithm ($m = 0$), we can initialize $\mathcal{T}^0(\widetilde{h})$ with $\texttt{BB}(\mathcal{T}^0(\widetilde{h}))$ defined as in (24) or using any arbitrary quadtree. Thus, $\mathcal{T}^0(\widetilde{h})$ serves as the initial set of grid points for estimating the values of all $(\widetilde{b}, \widetilde{e}) \in \mathcal{S}_b \times \mathcal{S}_e$ using the proposed piece-wise planar approximation. After initialization,

TABLE I
COMPUTATIONAL AND MEMORY COMPLEXITY OF SEVERAL VALUE ITERATION ALGORITHMS

Algorithm	Iteration Complexity	Memory Complexity
Value Iteration	$\mathcal{O}(\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert)$	$\mathcal{O}(\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert+\lvert\mathcal{S}\rvert)$
Factored Value Iteration	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\Pi_1\rvert\lvert\mathcal{A}\rvert)$	$\mathcal{O}(\lvert\mathcal{S}\rvert+\lvert\Sigma\rvert)$
PDS Value Iteration	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert^2+\lvert\mathcal{S}\rvert\lvert\Pi_2\rvert)$	$\mathcal{O}(\lvert\mathcal{S}\rvert+\lvert\Sigma\rvert)$
Approximate PDS Value Iteration	$\mathcal{O}(k\lvert\mathcal{T}\rvert\lvert\mathcal{A}\rvert^2+k\lvert\mathcal{T}\rvert\lvert\Pi_2\rvert)$	$\mathcal{O}(\lvert\mathcal{T}\rvert+\lvert\Sigma\rvert)$

the AVI algorithm proceeds similarly to PDS value iteration (Algorithm 1) but with two key differences. First, instead of acting on the full value functions $V_m(b,e,h)$ and $\widetilde{V}_m(b,e,h)$, $\forall(b,e,h)\in\mathcal{S}$, it acts on the corresponding approximate value functions $J_m(b,e,h)$ and $\hat{V}_m(b,e,h)$, respectively, defined on $(b,e)\in\mathcal{T}^m(h),\forall h\in\mathcal{S}_h$.[3] Second, we can (optionally) refine the quadtree after each iteration to meet a target error tolerance. This can be achieved by refining every leaf node of the quadtree $\mathcal{T}^m$ for which $\delta>\delta_{target}$, where δ is defined in Proposition 6, to the coarsest level such that $\delta\leq\delta_{target}$. We define the operator $\mathcal{A}_{\mathcal{T}}^{\delta_{target}}$ to denote this operation.

The following proposition shows that the AVI algorithm converges to an ϵ-optimal PDS value function rather than the optimal PDS value function. The proof is in Appendix VIII-I.

Proposition 7: Given a target single-step approximation error δ_{target}, the AVI algorithm converges to within a bound ϵ of the optimal PDS value function $\widetilde{V}^*$, i.e., $\lim_{m\to\infty}||\hat{V}^m-\widetilde{V}^*||<\epsilon$, where $\hat{V}^m$ denotes the approximate PDS value function after m iterations, $||\cdot||$ denotes the L_∞ norm, and $\epsilon=\frac{\gamma\delta_{target}}{1-\gamma}$.

Remark on Quadtree Refinement: It is interesting to consider whether it is possible that the quadtree will be refined too much in the intermediate steps of the AVI algorithm, while the required granularity to meet the target approximation error after convergence is in fact coarser. We argue that this will not happen because: 1) value iteration converges monotonically toward the optimal value function [11, Proposition 6.3.2]; 2) the integer convexity of the value function combined with the monotonicity of value iteration imply that the value function's slope increases with each iteration; and 3) the approximation error bound defined in Proposition 6 is increasing with the value function's slope. Since we refine the value function only when the error bound exceeds a target error threshold, δ_{target}, it follows that we will not refine the value function too much in the intermediate steps of value iteration.

C. Computational and Memory Complexity Analysis

In Table I, we compare the computational and memory complexities of conventional value iteration [29], factored value iteration (by factorizing the expectation over the next state as in the second line of (10)), PDS value iteration (Algorithm 1), and approximate PDS value iteration (Section VI-B). In the following discussion, let $|\mathcal{S}|$ and $|\mathcal{A}|$ denote the number of states and actions, respectively; let $|\mathcal{S}_b|$, $|\mathcal{S}_e|$, and $|\mathcal{S}_h|$ denote the number of data buffer states, battery states, and channel states, respectively; let $|\mathcal{L}|$ and $|\mathcal{E}|$ denote the size of the supports of the data packet and energy packet arrival distributions P^l and P^{e_H}, respectively; and note that the goodput distribution P^f has a support of size $|\mathcal{A}|$.

Value iteration's per-iteration complexity is $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ as it needs to iterate over all current and next state combinations, and compute a maximum over the actions, in each iteration. Additionally, it requires storing the transition probability function $P(s'|s,a)$ and the value function $V(s)$. Thus, its memory requirements are $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|+|\mathcal{S}|)$.

For factored value iteration, we only need to compute the expectation over the support $\Pi_1=\mathcal{L}\times\mathcal{A}\times\mathcal{E}\times\mathcal{S}_h$ instead of the full state space $\mathcal{S}$ as in conventional value iteration. Thus, factored value iteration's per-iteration complexity is $\mathcal{O}(|\mathcal{S}\times\Pi_1\times\mathcal{A}|)$. The algorithm requires storing the value function, the goodput distribution $P^f(f|a)$, the data arrival distribution $P^l(l)$, the energy arrival distribution $P^{e_H}(e_H)$, and the channel transition probabilities $P^h(h'|h)$. Thus, its memory requirements are $\mathcal{O}(|\mathcal{S}|+|\Sigma|)$, where we define $|\Sigma|=|\mathcal{A}|^2+|\mathcal{L}|+|\mathcal{E}|+|\mathcal{S}_h|^2$ for succinctness.

PDS value iteration's per-iteration complexity can be broken into two parts. First, the value function update step in (15) has complexity $\mathcal{O}(|\mathcal{S}||\mathcal{A}|^2)$ as it calculates an expectation over the goodput distribution and computes a maximum over the actions in every state. Second, the PDS value function update step in (28) calculates expectations over data arrivals, energy arrivals, and next channel states for every PDS. Consequently, it has complexity $\mathcal{O}(|\mathcal{S}||\Pi_2|)$, where $\Pi_2=\mathcal{L}\times\mathcal{E}\times\mathcal{S}_h$. It is reasonable to assume that $|\mathcal{L}|\ll|\mathcal{S}_b|$ and $|\mathcal{E}|\ll|\mathcal{S}_e|$ because the buffer and battery capacities should be much larger than the data and energy packet arrivals in any time slot; therefore, $|\Pi_2|=|\mathcal{L}\times\mathcal{E}\times\mathcal{S}_h|\ll|\mathcal{S}|=|\mathcal{S}_b\times\mathcal{S}_e\times\mathcal{S}_h|$, so PDS value iteration's total per-iteration complexity, $\mathcal{O}(|\mathcal{S}||\mathcal{A}|^2+|\mathcal{S}||\Pi_2|)$, is less than that of value iteration. Finally, the PDS value iteration's memory complexity is the same as that of the factored value iteration, as it needs to store the same value functions and distributions.

The proposed approximate PDS value iteration algorithm features similar complexity to PDS value iteration, save for two key differences. First, the evaluated states span the quadtree $\mathcal{T}$ instead of the full state space of $\mathcal{S}$. Second, approximating the value of a state $(\widetilde{b},\widetilde{e})\notin\mathcal{T}$ that is associated with a leaf (subtree) at maximum quadtree depth k has complexity $\mathcal{O}(k)$. Thus, a sparser approximation will be more computationally and memory efficient.

VII. NUMERICAL RESULTS

We now present our numerical results. In Section VII-A, we illustrate the optimal PDS value function's structural properties. In Section VII-B, we compare the performance of

[3]This has three important consequences: 1) the `for`-loops in Algorithm 2 loop over all $(b,e)\in\mathcal{T}^m(h),\forall h\in\mathcal{S}_h$ rather than all $(b,e,h)\in\mathcal{S}$; 2) the approximate value functions must be stored in memory rather than the full value functions; and 3) the values of states $(b,e)\notin\mathcal{T}^m(h)$ must be calculated based on the piece-wise planar approximation in (25).

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Modulations	BPSK, 4-PSK, 8-PSK	Packet Size, L	127 bytes
Packet Buffer Size, N_b	25 (data packets)	Discount Factor, γ	0.98
Energy Buffer Size, N_e	15 (energy packets)	Simulation Duration (slots)	50,000
Transmission Action, $a \in \mathcal{A}$	$\{0, \ldots, 3\}$ (data packets)	Packet Arrival PMF, P^l	Bern(x) or Pois(x) with variable x
Time Slot Duration, ΔT	5 ms	Energy Arrival PMF, P^{e_H}	Bern(x) with $x = 0.35, 0.7$
Bandwidth, W	250 kHz	Overflow Penalty, η	50
Noise Spectral Density, N_0	-174 dBm	Energy Packet Size	9.143 nJ
Channel States, $h \in \mathcal{S}_h$	{-18.82,-13.79,-11.23,-9.37, -7.80,-6.30,-4.68,-2.08} (dB)		

our approximate solution against the optimal scheduling policy and a greedy policy, which transmits as many backlogged packets as possible given the available energy (i.e., $a^n = \min\{b^n, \max\{a : e^n \geq e_{TX}(h^n, a)\}\}$), as considered in [18]. The parameters used in our MATLAB-based simulator are given in Table II. We consider M-PSK modulation. In channel state h, the energy required to transmit a packets with target bit-error probability, BEP_{target}, is given by (4), and the transmission power P_{TX} is approximated by [23, Table 6.1]:

$$P_{\text{TX}}(h, a; BEP_{\text{target}}) = \begin{cases} \dfrac{N_0}{2hT_s}[Q^{-1}(BEP_{\text{target}})]^2, & \text{if } \beta = 1, \\ \dfrac{N_0}{2h\beta T_s \sin^2(\pi/M)}[Q^{-1}(\dfrac{\beta}{2}BEP_{\text{target}})]^2, & \text{if } \beta > 1, \end{cases} \tag{29}$$

where $Q^{-1}(\cdot)$ is the inverse of the Q-function, N_0 is the noise spectral density, and β is the number of bits per symbol. The energy packet size was selected to be the amount of energy required to transmit a single packet in the best channel state.

A. Structural Properties

In this section, for illustration, we assume that the data and energy packet arrivals in each time slot are Bernoulli random variables with parameters 0.2 and 0.7, respectively. We consider two channel states $h = -4.68$ dB and $h = -11.23$ dB. Figs. 4a, 4g, and Figs. 4b, 4h, respectively, illustrate the optimal PDS value functions and policies. It is clear that the optimal PDS value functions (i) are non-decreasing and have increasing differences in the queue backlog (Propositions 1 and 3) and (ii) are non-increasing and have increasing differences in the battery state (Propositions 2 and 4). We can observe the difference in the shape of the optimal value functions for different channel conditions. In particular, the optimal value function has a smaller magnitude in the better channel state (-4.68 dB) because the long-term expected cost from that state is lower.

Figs. 4c and 4d show the greedy value function and policy, respectively. We observe that the optimal policy is more conservative than the greedy policy as it does not transmit packets in low battery states. That is because the optimal policy weighs the impact of its present scheduling action on its future performance given the data arrival, energy arrival, and channel dynamics.

Figs. 4e, 4i, and Figs. 4f and 4j show the approximate PDS value functions and policies, respectively, generated using a dynamically refined quadtree (resulting in 20 planes with finer refinement at higher buffer states[4]). The approximate PDS value functions preserve the optimal PDS value function's structure. Moreover, the policies derived from the approximate PDS value function (i.e., the *AVI policy*) have a similar structure to the optimal policy and are more aggressive than the optimal policy, but less aggressive than the greedy policy.

Finally, Fig. 3 shows that $\widetilde{V}(\widetilde{b}+1, \widetilde{e}, \widetilde{h}) - \widetilde{V}(\widetilde{b}, \widetilde{e}, \widetilde{h})$ is non-increasing in the battery state $\widetilde{e}$, i.e., the optimal PDS value function is submodular in $(\widetilde{b}, \widetilde{e})$ (Proposition 5).

B. Performance Evaluation

We now compare the performance of the AVI, optimal, and greedy policies in two scenarios considering both Bernoulli and Poisson distributed packet arrivals:

- **Abundant energy:** The energy packet arrival distribution $P^{e_H}(e_H) = \text{Bern}(0.7)$ and the data packet arrival distribution $P^l(l) = \text{Bern}(x)$ or $\text{Poisson}(x)$, where $x \in \{0.1, 0.113, \ldots, 0.6\}$. Since data packets often take multiple energy packets to transmit, the EHS ranges from lightly loaded to overloaded.
- **Scarce energy:** The energy and data packet arrival rates are halved compared to the abundant energy case, while maintaining the same data-to-energy packet ratios.

Fig. 5 illustrates how the system states and scheduling actions evolve over 400 time slots under the optimal policy when $P^l(l) = \text{Bern}(0.2)$ and $P^{e_H}(e_H) = \text{Bern}(0.7)$. We observe that the optimal policy tends to schedule transmissions if the buffer or battery occupancy is high, if the channel state is good, or if some judicious combination of these three conditions holds. It is particularly interesting to see how, under the optimal policy, the channel dynamics in Figure 5c influence the buffer and battery state evolution in Figures 5a and 5b, respectively. As the channel state worsens, the buffer state tends to increase because transmissions become too costly and it becomes difficult to serve the buffer as quickly as new data packets arrive. Due to the high transmission costs, the battery state also tends to decrease as the channel state worsens and we see a large drop in the battery state whenever more than one packet is scheduled for transmission. Despite this, the optimal policy manages to keep the buffer from overflowing and to maintain some energy reserves most of the time.

[4]The approximate value function is refined more at high buffer states because the cost directly depends on the buffer state b, and only indirectly depends on the battery state e (because e restricts the set of feasible actions). However, it is still possible to see finer refinement at lower battery states by, for example, using a lower approximation error threshold.

TABLE III
ALGORITHM COMPLEXITY AND APPROXIMATION ERROR COMPARISON

Algorithm	**Iteration Complexity** (FLOPs)	**Memory Complexity** (floats)	**Approx. Error**
Conventional Value Iteration	44302336	44305664	0
Factored Value Iteration	1703936	3412	0
PDS Value Iteration	159744	3412	0
AVI-3	93312	732	15.3
AVI-1	3456	153	277.5

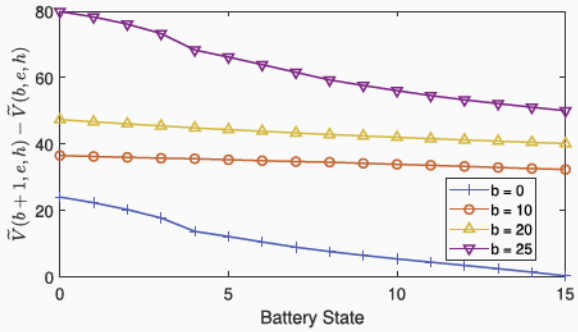


Fig. 3. Submodularity in (b, e) ($h = -6.30$ dB, $P^l = \text{Bern}(0.2)$, $P^{eH} = \text{Bern}(0.7)$).

In Fig. 6, we show how the average queuing delay,[5] battery occupancy, buffer overflows, and battery outages[6] vary with respect to the packet arrival rate in both the abundant and scarce energy scenarios, under the assumption of Bernoulli arrivals. Each measurement is taken by averaging across twelve 50,000 time slot simulations of the corresponding policy. The approximate policies labeled AVI-ℓ are calculated using the proposed approximate PDS value iteration algorithm with ℓ quadtree divisions (cf. Section VI-B). Fig. 6a shows the average queuing delay (left axis) and average battery occupancy (right axis) in the abundant energy scenario, and Fig. 6b shows the corresponding average overflows (left axis) and outages (right axis). We observe that AVI-ℓ's performance gradually improves in terms of all four performance metrics as the quadtree undergoes more subdivisions (i.e., as ℓ increases), and that its performance lies between that of the greedy and optimal policies. This can be attributed to the fact that a coarser piecewise-planar approximation results in a larger error between the approximate and optimal PDS value functions, yet acting on such an approximation is better than acting greedily.

Figs. 6a and 6b show that AVI-1 and AVI-3, respectively, achieve on average 10.61% and 25.79% lower average delays than the greedy policy across all packet arrival rates; 52.81% and 86.07% higher average battery occupancies; 14.94% and 33.14% fewer overflows; and 53.76% and 74.56% fewer outages. Taken together, these results demonstrate that the proposed solution not only *serves more sensor data* than the greedy policy (because less sensor data is lost when there are fewer overflows), but also serves it *with lower delay while using less energy*. We believe that simultaneous improvement of these metrics is an intrinsic feature of the problem, and while we are optimizing for delay metrics (queuing delay and overflows), improvement in terms of battery metrics is necessary to serve the incoming traffic effectively.

Figs. 6c and 6d show results for the scarce energy scenario with Bernoulli data packet arrivals. These results closely mirror the abundant energy scenario's trends in Figs. 6a and 6b, but exhibit a few key differences. First, the queuing delay is higher in the scarce energy scenario compared to the abundant energy scenario, while the overflows are lower. These are direct consequences of the data and energy packet arrival rates being halved in the scarce energy scenario while the data-to-energy packet ratio is kept fixed. Second, although the average battery occupancy is similar in both cases, we observe more outages. This is a direct consequence of scarce available energy. Due to lower average energy arrivals, the sensor has to wait longer to harvest equivalent energy to the abundant energy case, thereby causing more outages.

In Fig. 7, we show how the considered metrics perform with respect to the packet arrival rate in both the abundant and scarce energy scenarios, under the assumption of Poisson arrivals. These results closely mirror the trends for Bernoulli packet arrivals in Fig. 6, but exhibit slightly lower average queuing delays, fewer overflows, and fewer outages across all policies at the highest arrival rates. Since the Poisson distribution allows for more than 1 packet arrival in a time slot, these trends can be attributed to the policy being more aggressive to compensate for the extra arrivals. Due to space limitations, we are not able to show the optimal policies here.

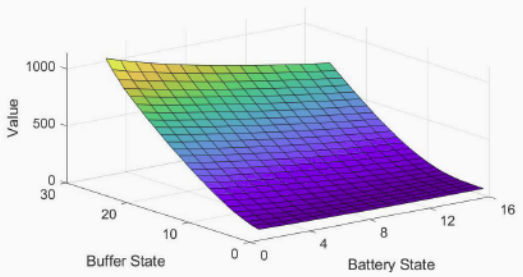
Finally, in Table III, we present an estimate of the iteration and memory complexities of value iteration, factored value iteration, PDS value iteration, AVI-3, and AVI-1 in units of floating point operations (FLOPs) and floating point numbers, respectively. These estimates are determined by plugging in the appropriate simulation model parameters in Table II into the iteration and memory complexity expressions in Table I. We consider only Bernoulli data and energy arrivals from Table II for this computation. AVI-3 requires approximately 40% lower iteration complexity and 75% lower memory complexity than PDS value iteration, which itself is orders of magnitude more efficient than conventional value iteration. AVI-1 reduces both iteration and memory complexity by over an order of magnitude compared to PDS value iteration.

VIII. CONCLUSION

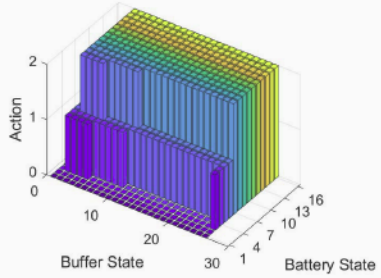
We formulated the DSEHS problem as an MDP and analyzed its structural properties. Our analysis does not make any

[5]The average queuing delay is calculated by dividing the average buffer backlog by the average number of packets admitted into the buffer in each time slot (i.e., packet arrivals minus buffer overflows). It is measured in units of time slots.

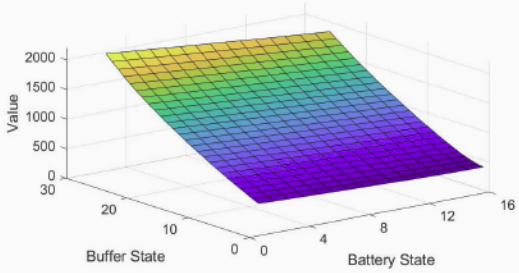
[6]A battery outage occurs when there is not enough energy to transmit a single packet in the current channel state.



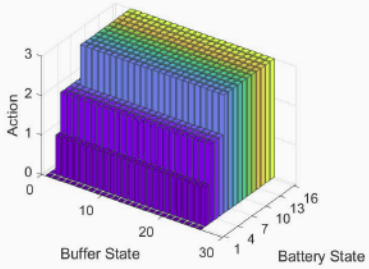
(a) Optimal PDS value function ($h = -4.68$ dB)



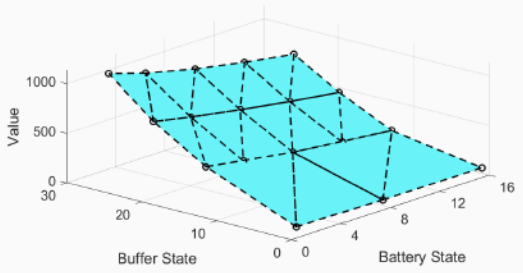
(b) Optimal policy ($h = -4.68$ dB)



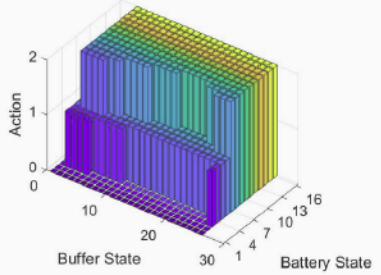
(c) Greedy PDS value function ($h = -4.68$ dB)



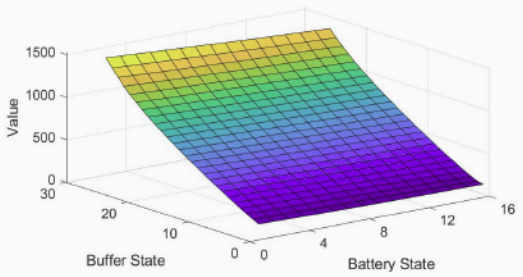
(d) Greedy policy ($h = -4.68$ dB)



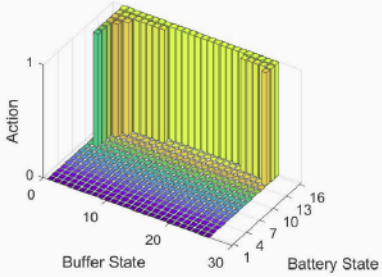
(e) Approximate PDS value function ($h = -4.68$ dB)



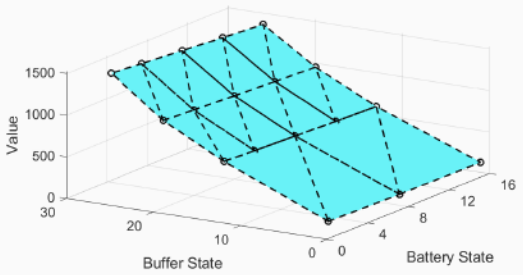
(f) AVI policy ($h = -4.68$ dB)



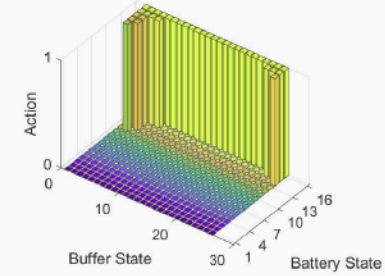
(g) Optimal value function ($h = -11.23$ dB)



(h) Optimal policy ($h = -11.23$ dB)



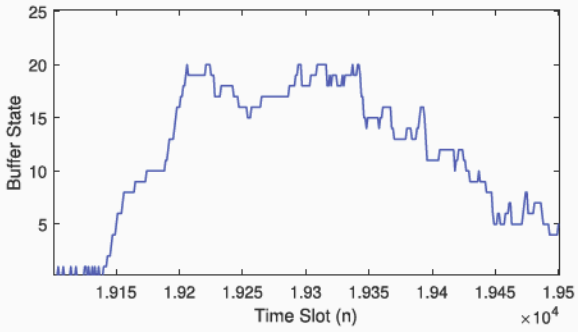
(i) Approximate PDS value function ($h = -11.23$ dB)



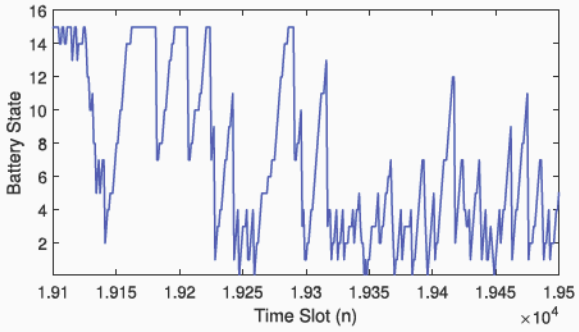
(j) AVI policy ($h = -11.23$ dB)

Fig. 4. PDS value functions and policies ($P^l = \text{Bern}(0.2)$, $P^{eH} = \text{Bern}(0.7)$, $\delta_{target} = 20$).

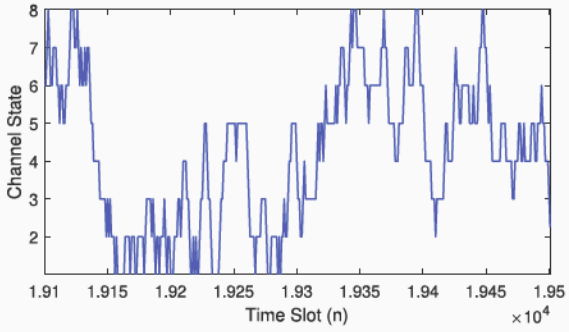
assumptions about the specific data and energy arrival distributions, save for they are i.i.d. or Markovian, and does not make any assumptions about the channel transition probabilities, other than that they are Markovian. We leverage the derived structural properties to design a low-complexity value iteration algorithm that operates on a piece-wise planar approximation



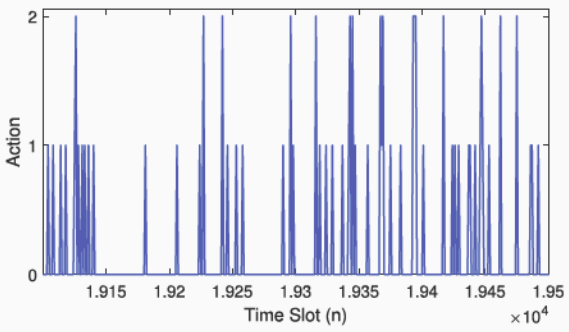
(a) Buffer state evolution over 400 time slots



(b) Battery state evolution over 400 time slots

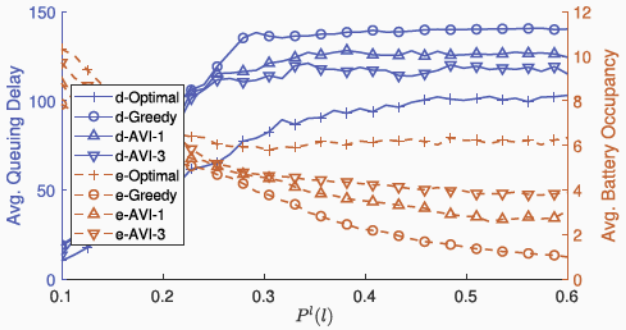


(c) Channel state evolution over 400 time slots

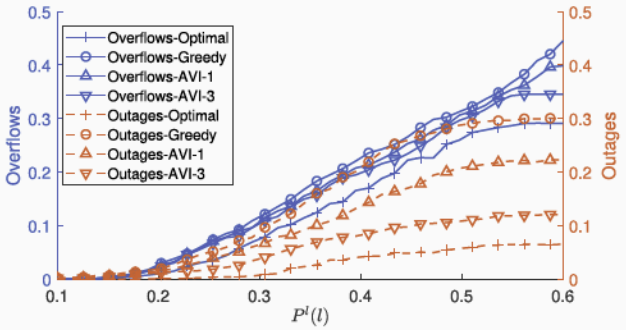


(d) Scheduling actions over 400 time slots

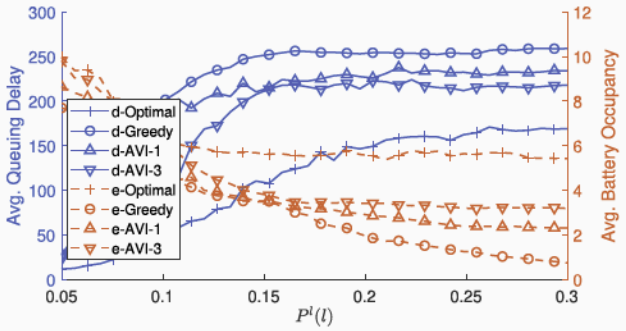
Fig. 5. System states and actions over 400 time slots (P^l = Bern(0.2) and P^{e_H} = Bern(0.7)).



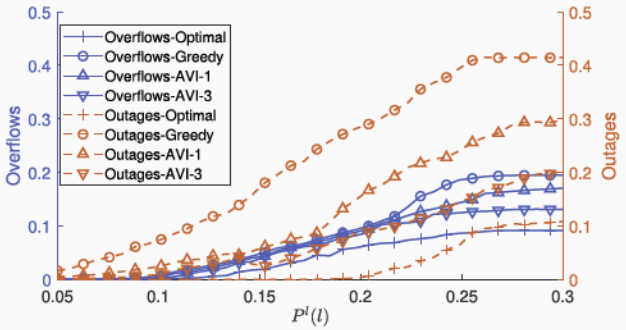
(a) Avg. delay and battery state vs. packet arrival rate



(b) Avg. outages and overflows vs. packet arrival rate



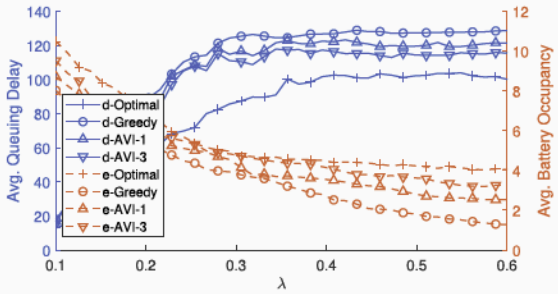
(c) Avg. delay and battery state vs. packet arrival rate



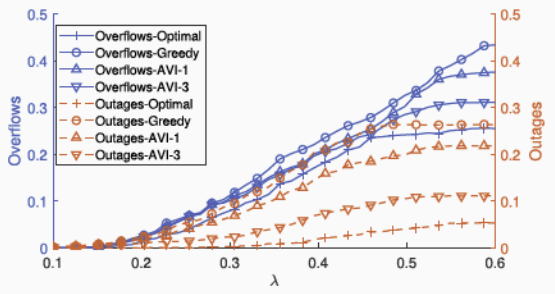
(d) Avg. outages and overflows vs. packet arrival rate

Fig. 6. Performance comparison under Bernoulli packet arrivals with different arrival rates. (a,b) Abundant energy with $P^{e_H}(e_H)$ = Bern(0.7). (c,d) Scarce energy with $P^{e_H}(e_H)$ = Bern(0.35).

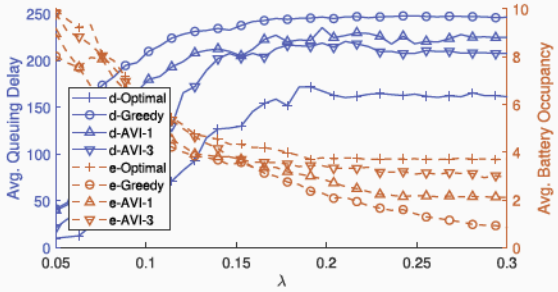
of the value function, and we prove that the algorithm's resulting approximation error is bounded. We demonstrate through comprehensive simulations that the policies computed by the proposed algorithm outperform a so-called greedy policy in terms of average queuing delay, battery occupancy, buffer overflows, and battery outages, while approaching the optimal policy's performance as finer approximations are used.



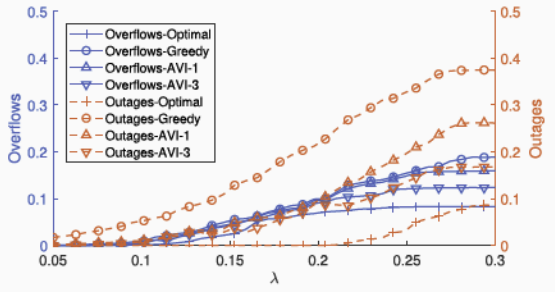
(a) Avg. delay and battery state vs. packet arrival rate



(b) Avg. outages and overflows vs. packet arrival rate



(c) Avg. delay and battery state vs. packet arrival rate



(d) Avg outages and overflows vs. packet arrival rate

Fig. 7. Performance comparison under Poisson arrivals with different arrival rates. (a,b) Abundant energy with $P^{e_H}(e_H) = \text{Poiss}(0.7)$. (c,d) Scarce energy with $P^{e_H}(e_H) = \text{Poiss}(0.35)$.

APPENDIX

A. Proof of Lemma 1

Let us first consider the case when $a = 0$. The first condition follows from the fact that $c([b,h],0)$ is a non-negative weighted sum of the holding and overflow costs, which are non-decreasing in b. The second condition is trivially satisfied because $d([b,e,h],0) = c([b,h],0)$ does not depend on e.

Now, we consider the case when $a > 0$. (i) Since $d([0,e,h],a) \leq d([1,e,h],a)$ and $d([b,e,h],a) \leq d([b+1,e,h],a)$ for all $b >= 1$, the first condition holds. (ii) Since $d([b,e,h],a) = d([b,e+1,h],a)$ if $e+1 < e_{TX}(h,a)$, $d([b,e,h],a) \geq d([b,e+1,h],a)$ if $e+1 = e_{TX}(h,a)$, and $d([b,e,h],a) = d([b,e+1,h],a)$ if $e+1 > e_{TX}(h,a)$, the second condition holds. □

B. Proof of Lemma 2

Assuming that $e \geq e_{TX}(h,a)$, we have

$$\begin{aligned}
&\sum_{e' \geq \bar{e}} P^e(e'|[e+1,h],a) \\
&\quad = \sum_{e' \geq \bar{e}} \sum_{e_H \in \mathcal{E}} \mathbb{I}_{\{e' = \min(e+1-e_{TX}(h,a)+e_H, N_e)\}} P^{e_H}(e_H) \\
&\quad = \sum_{\{e_H \in \mathcal{E} \,:\, \min(e+1-e_{TX}(h,a)+e_H, N_e) \geq \bar{e}\}} P^{e_H}(e_H) \\
&\quad \geq \sum_{\{e_H \in \mathcal{E} \,:\, \min(e-e_{TX}(h,a)+e_H, N_e) \geq \bar{e}\}} P^{e_H}(e_H) \\
&\quad = \sum_{e' \geq \bar{e}} \sum_{e_H \in \mathcal{E}} \mathbb{I}_{\{e' = \min(e-e_{TX}(h,a)+e_H, N_e)\}} P^{e_H}(e_H) \\
&\quad = \sum_{e' \geq \bar{e}} P^e(e'|[e,h],a),
\end{aligned}$$

where the first equality follows from the definition of P^e in (5); the second equality is obtained by moving the indicator function into the summation bounds; and the subsequent inequality follows from the fact that $\min(x+e_H, N_e) \geq \bar{e}$ for more values of e_H when x is larger. □

C. Proof of Lemma 4

The proof follows by induction. Since value iteration converges for any initialization, select $V_0([b,e,h])$ to be non-decreasing in b. Assume that $V_t([b,e,h])$ is non-decreasing in b. We prove that $V_{t+1}([b,e,h])$ is also non-decreasing in b. By definition,

$$\begin{aligned}
&V_{t+1}([b,e,h]) \\
&\quad = \min_{a \in \mathcal{A}(s)} \left\{ c([b,h],a) + \gamma \mathbb{E}_{b',e',h'}[V_t([b',e',h'])] \right\} \\
&\quad = \min_{a \in \mathcal{A}(b,e,h)} Q_{t+1}([b,e,h],a), \qquad (30)
\end{aligned}$$

where the expectation is over the transition probability function $P(s'|s,a)$ defined in Section IV. In Lemma 1.1, we established that the cost function $c([b,h],a)$ is non-decreasing in b. Additionally, since $V_t([b,e,h])$ is non-decreasing in b by the induction hypothesis and b' is stochastically increasing in b by Lemma 3, the expected future value is non-decreasing in b. It follows that $Q_{t+1}([b,e,h],a)$ is also non-decreasing in b.

Let a^* be the optimal action in state $(b+1,e,h)$. Assuming that $a^* \in \mathcal{A}(b,e,h)$, we have

$$\begin{aligned}
V_{t+1}([b+1,e,h]) &= Q_{t+1}([b+1,e,h],a^*) \\
&\geq Q_{t+1}([b,e,h],a^*) \\
&\geq \min_{a \in \mathcal{A}(b,e,h)} Q_{t+1}([b,e,h],a) \\
&= V_{t+1}([b,e,h])
\end{aligned}$$

where the first inequality follows from the fact that $Q_{t+1}([b,e,h],a)$ is non-decreasing in b and the second inequality follows from optimality. □

D. Proof of Lemma 6

We may express the value function defined in (14) as

$$V(b,e,h) = \min_{a\in\mathcal{A}(b,e,h)} \left\{ b + \mathbb{E}_f[\widetilde{V}(b-f, e-e_{TX}(h,a), h)] \right\} = b + \sum\nolimits_f P^f(f|a^*)\widetilde{V}([b-f, e-e_{TX}(h,a^*), h]),$$

where $a^* \in \{0,1,\ldots,N_a\}$ denotes the optimal action in state (b,e,h). If $\widetilde{V}(\widetilde{b},\widetilde{e},\widetilde{h})$ (i) has increasing differences in $\widetilde{b}$, (ii) has increasing differences in $\widetilde{e}$, or (iii) is submodular in $(\widetilde{b},\widetilde{e})$, then the results follow from the fact that a non-negative weighted sum of functions with increasing (decreasing) differences has increasing (decreasing) differences. □

E. Proof of Proposition 1

Recall from (13) that the optimal PDS value function, $\widetilde{V}^*([\widetilde{b},\widetilde{e},\widetilde{h}])$, can be written in terms of the optimal value function, $V^*(b,e,h)$. It is clear that the first term in the sum in (13) is non-decreasing in $\widetilde{b}$. Also, from Lemma 4, we note that the second term in the sum is non-decreasing in $\widetilde{b}$. The proof then follows from the fact that a non-negative weighted sum of non-decreasing functions is non-decreasing. □

F. Proof of Proposition 3

Consider the value iteration algorithm, which converges for any initial condition. Initialize the PDS value function $\widetilde{V}_0(\widetilde{b},\widetilde{e},\widetilde{h})$ to satisfy (21). Assume that (21) holds for $\widetilde{V}_t(\widetilde{b},\widetilde{e},\widetilde{h})$, for some $t>0$. We aim to show that (21) holds for $\widetilde{V}_{t+1}(\widetilde{b},\widetilde{e},\widetilde{h})$. Recall from (13) that the PDS value function can be expressed as a function of the conventional value function. The first term on the r.h.s. of (13) has increasing differences in $\widetilde{b}$. Thus, we only need to show that the second term on the r.h.s. of (13) has increasing differences in $\widetilde{b}$. This is implied if the following holds:

$$V_t([\widetilde{b}+l]^{N_b}, e', h') - V_t([\widetilde{b}-1+l]^{N_b}, e', h') \le V_t([\widetilde{b}+1+l]^{N_b}, e', h') - V_t([\widetilde{b}+l]^{N_b}, e', h'), \quad (31)$$

where $[x]^N = \min(x,N)$ and $e' = \min(\widetilde{e}+e_H, N_e)$. If we let $N_b = \infty$, then (31) reduces to

$$V_t(\widetilde{b}+l, e', h') - V_t(\widetilde{b}-1+l, e', h') \le V_t(\widetilde{b}+1+l, e', h') - V_t(\widetilde{b}+l, e', h'),$$

which holds by Lemma 6.1. This concludes the proof. □

We have empirically verified that Proposition 3 holds for the finite-buffer case, as well, throughout our extensive empirical evaluations. Moreover, we can analytically show its validity for N_b finite, when $\widetilde{b}+l \neq N_b$. However, completing the proof, to include this boundary case, turns out to be extremely challenging. We summarize in the following the reasons why. First, we note that (31) is sufficient, but not necessary for the desired result. To derive the necessary condition, we must expand each term in (21) using (13). As stated above, we can show that (21) holds in the finite-buffer case (i.e., $N_b < \infty$) except when $\widetilde{b}+l = N_b$ in (13). In this case, (21) reduces to the following condition:

$$P^l(N_b-\widetilde{b})\mathbb{E}_{e_H,h'}[V^*(N_b, \min(\widetilde{e}+e_H, N_e), h') - V^*(N_b-1, \min(\widetilde{e}+e_H, N_e), h')] \le \eta, \quad (32)$$

where P^l is the data packet arrival distribution and η is the overflow cost. Unfortunately, since the value function is recursively defined, a closed form expression for it does not exist. Additionally, we are not able to derive any meaningful upper bound for the left-hand side of (32) that we can compare to the overflow cost. Consequently, we are not able to prove the validity of (21), only for this single case of $\widetilde{b}+l = N_b$, when N_b is finite. However, if $\widetilde{b} < N_b - M_l$, where M_l is the maximum number of data packets that can arrive in one time slot, then we do not have to worry about this single case, and we can conclude that (21) holds even when N_b is finite.

G. Proof of Proposition 4

Consider the value iteration algorithm, which converges for any initial condition. Initialize the PDS value function $\widetilde{V}_0(\widetilde{b},\widetilde{e},\widetilde{h})$ to have increasing differences in the PDS battery state $\widetilde{e}$, i.e., to satisfy (22). Assume that (22) holds for $\widetilde{V}_t(\widetilde{b},\widetilde{e},\widetilde{h})$, for some $t>0$. We aim to show that (22) holds for $\widetilde{V}_{t+1}(\widetilde{b},\widetilde{e},\widetilde{h})$. Recall from (13) that the PDS value function can be expressed as a function of the conventional value function. The first term on the r.h.s. of (13), i.e., $\eta\sum_{l\in\mathcal{L}} P^l(l)\max(\widetilde{b}+l-N_b, 0)$ does not depend on $\widetilde{e}$; therefore, it has increasing differences in $\widetilde{e}$. Thus, we only need to show that the expected future value (i.e., the second term on the r.h.s. of (13)) has increasing differences in $\widetilde{e}$. This is implied if the following condition holds:

$$\begin{aligned} &V_t(b', \min(\widetilde{e}+e_H, N_e), h') - V_t(b', \min(\widetilde{e}-1+e_H, N_e), h') \\ &\le V_t(b', \min(\widetilde{e}+1+e_H, N_e), h') \\ &\quad - V_t(b', \min(\widetilde{e}+e_H, N_e), h'), \end{aligned} \quad (33)$$

where $b' = \min(\widetilde{b}+l, N_b)$. To verify that (33) holds, we consider the following three cases.

Case 1 ($\widetilde{e}+1+e_H \le N_e$): Assuming that $\widetilde{e}+1+e_H \le N_e$, we may rewrite (33) as follows:

$$V_t(b', \widetilde{e}+e_H, h') - V_t(b', \widetilde{e}-1+e_H, h') \le V_t(b', \widetilde{e}+1+e_H, h') - V_t(b', \widetilde{e}+e_H, h'),$$

which holds by Lemma 6.2.

Case 2 ($\widetilde{e}+e_H = N_e$): Assuming that $\widetilde{e}+e_H = N_e$, we may rewrite (33) as follows:

$$V_t(b', N_e, h') - V_t(b', N_e-1, h') \le V_t(b', N_e, h') - V_t(b', N_e, h'),$$

which holds by Lemma 5.

Case 3 ($\widetilde{e}+1+e_H > N_e$): Assuming that $\widetilde{e}+1+e_H > N_e$, we may rewrite (33) as follows:

$$V_t(b', N_e, h') - V_t(b', N_e, h') \le V_t(b', N_e, h') - V_t(b', N_e, h'),$$

which holds because both sides are equal to 0. □

H. Proof of Proposition 5

Consider the value iteration algorithm, which converges for any initial condition. Initialize $\widetilde{V}_0(\widetilde{b},\widetilde{e},\widetilde{h})$ to satisfy (23). Assume that (23) holds for $\widetilde{V}_t(\widetilde{b},\widetilde{e},\widetilde{h})$, for some $t>0$. We aim to show that (23) holds for $\widetilde{V}_{t+1}(\widetilde{b},\widetilde{e},\widetilde{h})$. Recall from (13) that the PDS value function can be expressed as a function of the conventional value function. The first term on the r.h.s. of (13) is submodular in $(\widetilde{b},\widetilde{e})$. Thus, we only need to show that the expected future value (i.e., the second term on the r.h.s. of (13)) is submodular in $(\widetilde{b},\widetilde{e})$, i.e.,

$$\begin{aligned}&V_t([b''+1]^{N_b},[e''+1]^{N_e},h')-V_t([b'']^{N_b},[e''+1]^{N_e},h')\\&\quad\le V_t([b''+1]^{N_b},[e'']^{N_e},h')-V_t([b'']^{N_b},[e'']^{N_e},h'),\end{aligned}\tag{34}$$

where we use $[x]^N \triangleq \min(x,N)$, $b''\triangleq \widetilde{b}+l$ and $e''\triangleq \widetilde{e}+e_H$ to keep the equations compact. To verify that (34) holds, we consider the following two cases.

Case 1 ($b''+1\le N_b$): Assuming that $b''+1\le N_b$, we may rewrite (34) as follows:

$$\begin{aligned}&V_t(b''+1,[e''+1]^{N_e},h')-V_t(b'',[e''+1]^{N_e},h')\\&\qquad\le V_t(b''+1,[e'']^{N_e},h')-V_t(b'',[e'']^{N_e},h').\end{aligned}$$

If $e''+1\le N_e$, then the condition holds by Lemma 6.3 and, if $e''+1>N_e$, then both sides are equal; thus, Case 1 holds.

Case 2 ($b''\ge N_b$): Assuming that $b''\ge N_b$, we may rewrite (34) as follows:

$$\begin{aligned}&V_t(N_b,[e''+1]^{N_e},h')-V_t(N_b,[e''+1]^{N_e},h')\\&\qquad\le V_t(N_b,[e'']^{N_e},h')-V_t(N_b,[e'']^{N_e},h'),\end{aligned}$$

where both sides are equal to 0; thus, Case 2 holds. □

I. Proof of Proposition 7

To simplify the proof, we introduce some new notation. Using the PDS, we can factor the transition probabilities into known and unknown components, where the known component accounts for the transition from the current state s to the PDS $\widetilde{s}$ and the unknown component accounts for the transition from the PDS $\widetilde{s}$ to the next state s' [22]. Formally,

$$P(s'|s,a)=\sum\nolimits_{\widetilde{s}\in\mathcal{S}} p_u(s'|\widetilde{s})p_k(\widetilde{s}|s,a),\tag{35}$$

where the subscripts k and u denote the known and unknown components, respectively. We can factor the cost function similarly:

$$c(s,a)=c_k(s,a)+\sum\nolimits_{\widetilde{s}\in\mathcal{S}} p_k(\widetilde{s}|s,a)c_u(\widetilde{s}).\tag{36}$$

In our problem, the known and unknown costs and transition probabilities are defined as:

$$c_k(s,a)=b,\tag{37}$$

$$c_u(\widetilde{s})=\eta\sum\nolimits_{l\in\mathcal{L}} P^l(l)\max(\widetilde{b}+l-N_b,0),\tag{38}$$

$$P_k(\widetilde{s}|s,a)=P^f(b-\widetilde{b}|a)\mathbb{I}_{\{\widetilde{e}=e-e_TX(h,a)\}}\mathbb{I}_{\{\widetilde{h}=h\}},\tag{39}$$

$$\begin{aligned}P_u(s'|\widetilde{s})=&\sum\nolimits_{l\in\mathcal{L}}\sum\nolimits_{e_H\in\mathcal{E}}\mathbb{I}_{\{b'=\min(\widetilde{b}+l,N_b)\}}\\&\mathbb{I}_{\{e'=\min(\widetilde{e}+e_H,N_e)\}}P_l(l)P_{e_H}(e_H)P_h(h'|h),\end{aligned}\tag{40}$$

where $\mathbb{I}_{\{\cdot\}}$ is the indicator function. Using this new notation, we may rewrite (13) and (14) as follows:

$$\widetilde{V}^*(\widetilde{s})=c_u(\widetilde{s})+\gamma\sum\nolimits_{s'\in\mathcal{S}} P_u(s'|\widetilde{s})V^*(s')\tag{41}$$

$$V^*(s)=\min_{a\in\mathcal{A}}\left\{c_k(s,a)+\sum\nolimits_{\widetilde{s}\in\mathcal{S}} P_k(\widetilde{s}|s,a)\widetilde{V}^*(\widetilde{s})\right\}\tag{42}$$

Plugging (42) into (41), we define a mapping H_{PDS} that maps a $\widetilde{V}$-vector to a new $\widetilde{V}$-vector $H_{PDS}\widetilde{V}$, as follows

$$\begin{aligned}(H_{PDS}\widetilde{V})(\widetilde{s})=c_u(\widetilde{s})+\gamma\sum_{s'\in\mathcal{S}}P_u(s'|\widetilde{s})\min_{a\in\mathcal{A}}\Bigg\{&c_k(s',a)\\&+\sum_{\widetilde{s}'\in\mathcal{S}}P_k(\widetilde{s}'|s',a)\widetilde{V}(\widetilde{s}')\Bigg\},\forall\widetilde{s}\end{aligned}\tag{43}$$

where c_k, c_u, P_k, P_u denote the known cost, unknown cost, known transition probabilities, and unknown transition probabilities, respectively. We can show that H_{PDS} is a contraction mapping (i.e., $||(H_{PDS}\widetilde{V})-\widetilde{V}^*||\le\gamma||\widetilde{V}-\widetilde{V}^*||$, where $||\cdot||$ denotes the L_∞ norm) and that PDS value iteration (Algorithm 1) converges to the optimal PDS value function $\widetilde{V}^*$.

Based on (43), applying H_{PDS} to the approximate value function, $\hat{V}=A_T^\delta\widetilde{V}$, we get

$$\begin{aligned}(H_{PDS}A_T^\delta\widetilde{V})(\widetilde{s})=c_u(\widetilde{s})+\gamma\sum_{s'\in\mathcal{S}}P_u(s'|\widetilde{s})\min_{a\in\mathcal{A}}\Bigg\{&c_k(s',a)\\&+\sum_{\widetilde{s}'\in\mathcal{S}}P_k(\widetilde{s}'|s',a)A_T^\delta\widetilde{V}(\widetilde{s}')\Bigg\}.\end{aligned}$$

Since A_T^δ introduces a single-step error of δ, the previous equation can be rewritten as,

$$\begin{aligned}(H_{PDS}A_T^\delta\widetilde{V})(\widetilde{s})\le c_u(\widetilde{s})+\gamma\sum_{s'\in\mathcal{S}}P_u(s'|\widetilde{s})\min_{a\in\mathcal{A}}\Bigg\{&c_k(s',a)\\&+\sum_{\widetilde{s}'\in\mathcal{S}}P_k(\widetilde{s}'|s',a)(\widetilde{V}(\widetilde{s}')+\delta)\Bigg\}\le(H_{PDS}\widetilde{V})(\widetilde{s})+\gamma\delta.\end{aligned}$$

Thus, applying H_{PDS} to the approximate value function, $A_T^\delta\widetilde{V}$, gives

$$\begin{aligned}||(H_{PDS}A_T^\delta\widetilde{V})-\widetilde{V}^*||&\le||(H_{PDS}\widetilde{V})+\gamma\delta-\widetilde{V}^*||\\&\le||(H_{PDS}\widetilde{V})-\widetilde{V}^*||+\gamma\delta\le\gamma||\widetilde{V}-\widetilde{V}^*||+\gamma\delta\end{aligned}\tag{44}$$

Let $(H_{PDS}A_T^\delta)^m$ denote the case when the contraction mapping H_{PDS} and the approximation operator A_T^δ are applied m times consecutively. Then, we have

$$\begin{aligned}&||(H_{PDS}A_T^\delta)^m\widetilde{V}-\widetilde{V}^*||\\&\quad=||(H_{PDS}A_T^\delta)(H_{PDS}A_T^\delta)^{m-1}\widetilde{V}-\widetilde{V}^*||\\&\quad\le\gamma\delta+\gamma||(H_{PDS}A_T^\delta)^{m-1}\widetilde{V}-\widetilde{V}^*||\\&\quad\le\gamma\delta+\gamma^2\delta+\gamma^2||(H_{PDS}A_T^\delta)^{m-2}\widetilde{V}-\widetilde{V}^*||\\&\qquad\vdots\\&\quad=\gamma\delta\sum\nolimits_{i=0}^{m-1}\gamma^i\quad+\quad\gamma^m||\widetilde{V}-\widetilde{V}^*||,\end{aligned}\tag{45}$$

where all the inequalities follow from (44). Taking the limit of (45) as $m \to \infty$, we get

$$\lim_{m\to\infty} ||(H_{PDS}A_T^{\delta})^m \widetilde{V} - \widetilde{V}^*|| = \lim_{m\to\infty} \left\{ \gamma\delta \sum_{i=0}^{m-1} \gamma^i \quad + \quad \gamma^m ||\widetilde{V} - \widetilde{V}^*|| \right\} = \frac{\gamma\delta}{1-\gamma}$$

Thus, the AVI algorithm converges to the optimal value function within a bound $\epsilon = \frac{\gamma\delta}{1-\gamma}$.

REFERENCES

[1] N. Sharma, N. Mastronarde, and J. Chakareski, "Structural properties of optimal transmission policies for delay-sensitive energy harvesting wireless sensors," in *Proc. IEEE Int. Conf. Commun.*, May 2018, pp. 1–7.

[2] R. J. M. Vullers, R. V. Schaijk, H. J. Visser, J. Penders, and C. V. Hoof, "Energy harvesting for autonomous wireless sensor networks," *IEEE Solid State Circuits Mag.*, vol. 2, no. 2, pp. 29–38, Feb. 2010.

[3] J. Chakareski, "Uplink scheduling of visual sensors: When view popularity matters," *IEEE Trans. Commun.*, vol. 63, no. 2, pp. 510–519, Feb. 2015.

[4] A. Seyedi and B. Sikdar, "Energy efficient transmission strategies for body sensor networks with energy harvesting," *IEEE Trans. Commun.*, vol. 58, no. 7, pp. 2116–2126, Jul. 2010.

[5] J. Chakareski, "Aerial UAV-IoT sensing for ubiquitous immersive communication and virtual human teleportation," in *Proc. IEEE INFOCOM Workshops*, Atlanta, GA, USA, May 2017, pp. 718–723.

[6] J. Chakareski, "Drone networks for virtual human teleportation," in *Proc. 3rd ACM MobiSys Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet)*, Niagara Falls, NY, USA, Jun. 2017, pp. 21–26.

[7] J. Chakareski, "Informative state-based video communication," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2115–2127, Jun. 2013.

[8] B. Gurakan and S. Ulukus, "Energy harvesting multiple access channel with data arrivals," in *Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[9] X. Lu, P. Wang, D. Niyato, and E. Hossain, "Dynamic spectrum access in cognitive radio networks with RF energy harvesting," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 102–110, Jun. 2014.

[10] D. Gunduz, K. Stamatiou, N. Michelusi, and M. Zorzi, "Designing intelligent energy harvesting communication systems," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 210–216, Jan. 2014.

[11] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.

[12] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 8, pp. 1732–1743, Sep. 2011.

[13] C. K. Ho and R. Zhang, "Optimal energy allocation for wireless communications with energy harvesting constraints," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4808–4818, Sep. 2012.

[14] J. Yang and S. Ulukus, "Optimal packet scheduling in a multiple access channel with energy harvesting transmitters," *J. Commun. Netw.*, vol. 14, no. 2, pp. 140–150, Apr. 2012.

[15] J. Yang and S. Ulukus, "Optimal packet scheduling in an energy harvesting communication system," *IEEE Trans. Commun.*, vol. 60, no. 1, pp. 220–230, Jan. 2012.

[16] A. Aprem, C. R. Murthy, and N. B. Mehta, "Transmit power control policies for energy harvesting sensors with retransmissions," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 5, pp. 895–906, Oct. 2013.

[17] N. Michelusi, K. Stamatiou, and M. Zorzi, "On optimal transmission policies for energy harvesting devices," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2012, pp. 249–254.

[18] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *IEEE Trans. Wireless Commun.*, vol. 9, no. 4, pp. 1326–1336, Apr. 2010.

[19] D. Zordan, T. Melodia, and M. Rossi, "On the design of temporal compression strategies for energy harvesting sensor networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1336–1352, Feb. 2016.

[20] 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, Dec. 2011.

[21] D. V. Djonin and V. Krishnamurthy, "MIMO transmission control in fading channels—A constrained Markov decision process formulation with monotone randomized policies," *IEEE Trans. Signal Process.*, vol. 55, no. 10, pp. 5069–5083, Oct. 2007.

[22] 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, Apr. 2013.

[23] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[24] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ, USA: Wiley, 2007.

[25] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.

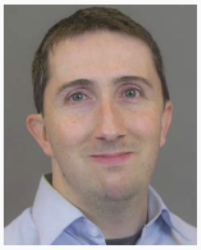
[26] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An online learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, May 2008.

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

[28] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[29] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

Nikhilesh Sharma received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology Srinagar, India, in 2014, and the M.S. degree in electrical engineering from University at Buffalo, NY, USA, in 2017, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include reinforcement learning, deep learning, Markov decision processes, resource allocation in wireless networks and systems, and 4G/5G networks.



Nicholas Mastronarde (S'07–M'11–SM'16) received the B.S. and M.S. degrees from the University of California at Davis, Davis, CA, USA, in 2005 and 2006, respectively, and the Ph.D. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2011, all in electrical engineering. He is currently an Associate Professor with the Department of Electrical Engineering, University at Buffalo, Buffalo, NY, USA. His research interests include reinforcement learning, Markov decision processes, resource allocation and scheduling, UAV networks, and 4G/5G networks.

Jacob Chakareski completed his Ph.D. degree in electrical and computer engineering at Rice and Stanford, held research appointments with Microsoft, HP Labs, and EPFL, and sat on the advisory board of Frame, Inc. He is currently an Associate Professor with the Ying Wu College of Computing, NJIT, where he leads the Laboratory for VR/AR Immersive Communication (LION). His research interests span networked virtual and augmented reality, UAV IoT sensing and networking, real-time reinforcement learning, 5G wireless edge computing/caching, ubiquitous immersive communication, and societal applications. His research was supported by NSF, AFOSR, Adobe, Tencent Research, NVIDIA, and Microsoft. He is the organizer of the first NSF Visioning Workshop on networked VR/AR communications. He received the Adobe Data Science Faculty Research Award in 2017 and 2018, the Swiss NSF Career Award Ambizione in 2009, the AFOSR Faculty Fellowship in 2016 and 2017, and the Best/Fast-Track Paper Awards at IEEE ICC 2017 and IEEE Globecom 2016. For further information, please visit www.jakov.org.