The Journal of Supercomputing
https://doi.org/10.1007/s11227-022-04643-9

Q-learning-based algorithms for dynamic transmission control in IoT equipment

Hanieh Malekijou[1] · Vesal Hakami[1] · Nastooh Taheri Javan[2] · Amirhossein Malekijoo[3]

Accepted: 1 June 2022

Abstract

We investigate an energy-harvesting IoT device transmitting (delay/jitter)-sensitive data over a wireless fading channel. The sensory module on the device injects captured event packets into its transmission buffer and relies on the random supply of the energy harvested from the environment to transmit them. Given the limited harvested energy, our goal is to compute optimal transmission control policies that decide on how many packets of data should be transmitted from the buffer's head-of-line at each discrete timeslot such that a long-run criterion involving the average delay/jitter is either minimized or never exceeds a pre-specified threshold. We realistically assume that no advance knowledge is available regarding the random processes underlying the variations in the channel, captured events, or harvested energy dynamics. Instead, we utilize a suite of Q-learning-based techniques (from the reinforcement learning theory) to optimize the transmission policy in a model-free fashion. In particular, we come up with three Q-learning algorithms: a constrained Markov decision process (CMDP)-based algorithm for optimizing energy consumption under a delay constraint, an MDP-based algorithm for minimizing the average delay under the limitations imposed by the energy harvesting process, and finally, a variance-penalized MDP-based algorithm to minimize a linearly combined cost function consisting of both delay and delay variation. Extensive numerical results are presented for performance evaluation.

Keywords Delay · Energy harvesting · Jitter · Transmission control · Markov decision process · Reinforcement learning

✉ Vesal Hakami
vhakami@iust.ac.ir

Extended author information available on the last page of the article

Published online: 03 July 2022

1 Introduction

1.1 Research background

IOT is a novel inter-networking notion envisioning cyber-physical sensory devices interconnected with each other and ideally making actuating decisions independently without human intervention. This emerging computing/networking technology is expected to eventually support billions of wireless device interconnections, but at the same time trigger a dramatic rise in battery demand for powering the sensors and radio transceivers. In fact, given the limited energy storage installed on a typical IoT device, there is a growing interest in adopting energy harvesting techniques to supply the nodes and recharge their batteries through natural energy sources (e.g., solar power, piezoelectric energy, etc.) [1–3]. The main benefit of energy harvesting is in enabling the autonomous operation of IoT devices in remote places without access to grid power and the need for battery replacement. However, the random nature of the harvested energy leads to uncertainty in the node's power supply, thereby making it very challenging to come up with efficient transmission control policies. Top this all off with the fact that wireless transmission is subject to time-varying channel qualities. Also, the sensory modules mounted on the IoT devices typically generate random traffic loads—arising due to the nature of the events stochastically occurring in the environment. Therein arises a requirement to investigate a new set of optimization problems that explicitly cater to the uncertainty and the time-varying randomness faced by the IoT nodes to support their applications. Of particular interest is the study of energy-constrained wireless IoT devices that are deployed for (delay/jitter)-sensitive applications (e.g., wireless body area networks, remote visual monitoring, surveillance, etc.).

Related work has studied the computation of optimal transmission policies for energy harvesting IoT devices by proposing various deterministic [2–4], stochastic [5–7] as well as model-free optimization algorithms [8–10]. Before stating our motivations and contributions, we give a thorough overview of the previous schemes in Sect. 1.2.

1.2 Literature review

We categorize the previous schemes on transmission control for IoT equipment into three major categories: *Deterministic*, *Stochastic*, and *Learning-based* model-free schemes:

- In a deterministic optimization scheme, it is assumed that exact information about the time-varying quality of the wireless channel, amount of data, and energy arrival is available ahead of decision-making at the transmitter side.
- In a stochastic optimization scheme, optimal policies are derived only based on the available statistical information about the environment such as data and energy arrival and the wireless channel.

- In many practical applications, complete non-causal knowledge or even statistical knowledge of the system dynamics might not be available [11, 12]. This is especially the case with energy harvesting processes when they are non-stationary or from sources with unknown distributions. For example, in a wireless network where nodes are distributed randomly over a geographical area, the characteristics of the harvested energy (e.g., solar power) at each node depend on the node location which changes over time in a non-stationary fashion [13, 14]. In such cases, prior knowledge about the dynamics of energy sources is very difficult to obtain. As such, a learning-based model-free optimization scheme is required to overcome such challenges.

In Table 1, we have summarized the comparison of related work. In the sequel, we review some representative related work from each category.

1.2.1 Deterministic schemes

Yang et al. [3] have assumed that both the data and energy buffering capacity of each node are infinite and that all the harvested energy is to be allocated for transmission data over a static channel. In addition, to minimize transmission completion time in a single-user communication system, the authors have presented an optimal scheduling policy that adjusts the transmission rate according to the harvested energy and channel traffic. He et al. [2] have investigated rechargeable sensor nodes assuming the availability of deterministic information. Authors have argued that the optimal power management policy is based on a fixed transmission power at each time interval until the amount of input energy or channel status is changed. So, in this case, the total energy consumed is equal to the total energy harvested at the specified time. Moreover, in an optimal transmission control policy, the transmission rate should increase steadily over time. Therefore, the authors have assumed both the battery capacity and the data buffer in infinite length (which is not a practical assumption).

1.2.2 Stochastic schemes

As for the stochastic schemes, Shaviv et al. [5] have suggested a random-energy power harvesting system with a limited-space battery. They have also developed a simple stochastic power optimal transmission control policy that requires to know the statistical distribution of the received energy. Arafa et al. [6] have proposed a power scheduling policy, intended for single-user energy harvesting systems to identify stochastic policies that maximize the average long-term performance. The authors have uniformly increased productivity of the system and have employed a transmitter that is equipped with a limited battery space to store its harvested energy. The simulation results have shown that the policies that maintain a certain constant amount of battery life at any given time are optimal in terms of energy consumption. Wenwei et al. [22] have formulated the optimization problem that minimizes packet delays in the Timely Data Delivery (TDD) scheme for the IoT devices to determine when to harvest energy and deliver data. The TDD scheme reduces the packet delay

Table 1 Comparison of related work

Refs#	Scheme type	Objective	Constraints	Solution approach
]	Deterministic	Minimizing average data buffer length	A, B, C	Off-line Scheduling Policy
[5]	Stochastic	Maximizing average throughput	A	Approximation Scheduling Policy
[6]	Stochastic	Maximizing average long-term performance	A, B	Approximation Scheduling Policy
[7, 14–16]	Stochastic	Minimizing energy consumption	A	Bernoulli model, Uniform process, Poisson process
[10]	Learning-Based	Minimizing average data buffer length	C	RL
[17]	Learning-Based	Minimizing average data buffer length	A, C	Adaptive transmission rates were combined in the physical layer
[18]	Learning-Based	Minimizing average latency	B, C	Mathematical formulation
[19]	Learning-Based	Minimizing energy consumption	B	MDP model
[20]	Learning-Based	Minimizing average data buffer length	B, C	Multi-dimensional CMDP model
[21]	Learning-Based	Minimizing energy, high-fidelity reconstruction at the IoT gateway, low packet drop ratio, and timeliness of data	B	RL, PDS, VE
Proposed Framework	Learning-Based	Minimizing energy, Minimizing average data buffer length, Minimizing Average Jitter	A, B, C	RL (MDP, CMDP, Var-Penalized MDP)

A: Battery Capacity B: Data Buffer length C: Wireless channel status.

by letting each IoT device adjust a pair of parameters, i.e., the lower bound and the upper bound of the energy storage, according to the packet arrival rate, packet delivery time, and energy harvesting rate. Aprem et al. [7] have examined the energy generation process by implementing a Bernoulli model and a fixed harvesting rate under the assumption that the harvested energy in each timeslot is identically and independently distributed (i.i.d.). Other uncorrelated energy harvesting models reported in the literature include the uniform process [16], Poisson process [15, 23], and exponential process [24]. Although these models are all easy to implement, they are inadequate to benefit from the temporal correlation advantages of the harvested energy from most energy sources. To get around this problem, a correlated time process followed by a first-order discrete-time Markov model has been adopted by Blasco et al. [24] to model the energy packet arrivals.

1.2.3 Learning-based model-free schemes

Model-free schemes can further be divided into two subcategories: those that are based on Lyapunov optimization [8] and those that utilize learning-theoretic schemes, e.g., reinforcement learning (RL) techniques. In a model-free scheme, the sensor nodes, rather than relying on a pre-built model, use the experience gained from interaction with the actual environment over time to discover the optimal transmission control policy. The application of these techniques is particularly important because oftentimes the precise modeling of the probabilistic structure of the system is not possible in many practical scenarios. Moreover, the solution based on a particular model loses its validity when environmental conditions change.

Sharma et al. [9] have investigated an IoT sensor node that transmits delay-sensitive data over a wireless channel. This sensor places packets in a queue and sends them over the channel through the energy it receives from the environment. The goal is to minimize the packet delay in the queue for each node, while respecting the energy constraints. In another study [10], the transmissions of delay-sensitive image data over time-varying wireless channels have been studied. RL techniques have been adopted to reduce the average delay. In this study, the proposed learning algorithm has incorporated only a portion of the system states and has thus significantly reduced the time complexity of the updates. Moreover, the objective function is the average data buffer length with a constraint on the wireless channel status. To balance energy and delay, Mastronarde et al. [17] have proposed a model-free scheme for multifunctional energy-aware scheduling for delay-sensitive data sent over a wireless channel. In this scheme, the authors have employed the Markov decision process (MDP) formalism [25] to formulate the transmission control problem. Xiao et al. [26] have proposed a SARSA (state-action-reward-state-action) RL algorithm [27] to choose the transmission action based on the observed state (the queue length of the buffer, the channel gain, the previous bit error rate, and the previous packet loss rate) without prior knowledge of the transmission channel model at the transmitter and the receiver. The authors' intention has been to guarantee the video quality and to reduce interference in 6G multimedia services. Hakami et al. [18] have proposed a model-free RL-based algorithm for the delay-constrained optimization

of joint compression and transmission control for IoT equipment. Masadeh et al. [19] have used the notion of MDPs to maximize the sensor's performance (in terms of average throughput) while minimizing its energy consumption. The authors have proposed a new RL-based method and have compared it against a greedy method to demonstrate its superiority. Hu et al. [20] have presented a cross-layer approach to adapt the transmission power, rate, and compression ratio based on the instantaneous buffer and channel states to minimize the average delay subject to average power and distortion constraints. They have proposed a multi-dimensional constrained MDP (CMDP) formulation which is subsequently decomposed into a deterministic hierarchical optimization consisting of linear programming and a convex optimization problem. Namjoonia et al. [21] have proposed three RL procedures based on Q-learning, post-decision state (PDS), and virtual experience (VE) learning, respectively. They have addressed the delay-constrained joint lossy compression and transmission control problem for an IoT device with rechargeable energy storage.

1.3 Motivations

According to our review of the related work in Sect. 1.2, at least two major shortcomings can be identified in association with the existing research:

- Firstly, while the previous studies have made significant progress toward optimizing IoT communication systems, they have mostly considered simplifying assumptions regarding their system model. In particular, a majority of the available schemes have been basically assuming that the underlying processes behind the channel variations, energy charging, and event generation are all i.i.d. [2, 9, 28–30]. Such assumptions are not suitable for more realistic stochastic dynamics with temporal dependency between correlated fading channel conditions, energy arrivals, and sensory event generation. This warrants a formulation based on a more generalized formalism of MDPs that can exploit (for instance) the temporal correlation/channel memory inherent in the Markovian channels to further improve the network performance.
- Secondly, in event-driven IoT applications, the incoming traffic load on a device can momentarily exceed the wireless channel capacity. In such cases, data packets are temporarily queued in a transmission buffer the length of which can be controlled by dynamically tuning the transmission behavior of the node. To strike a good balance between the delay requirements of the applications and energy consumption, the node's transmission rate should be increased when the buffer grows too much and decreased when the buffer recedes. Such dynamic control of transmission rates results in oscillating buffering behavior. Heavy fluctuations can increase buffer overflow (i.e., packet loss), buffer underflow (i.e., wastage of the available channel bandwidth), and jitter or delay variance. It is therefore imperative that we minimize buffer fluctuations (or equivalently buffer length variance) to minimize the above-mentioned undesirable effects. However, as we review in the sequel, the majority of prior work has solely focused on throughput

and delay as conventional performance metrics to judge the quality of service (QoS) [9, 15, 17, 29–31]. To support real-time IoT scenarios, the time-averaged variance in packet delivery times is also an important metric. Hence, the network algorithm design should not only efficiently optimize the delay and packet delivery ratio but also account for jitter or delay variation as the higher-order statistics of the arrival and transmission processes. Achieving an optimal "mean–variance trade-off" in packet delivery times is a key issue absent from the related work on transmission control for energy harvesting IoT devices.

1.4 Contributions

So motivated, in this paper, we consider an IoT device that is supposed to transmit real-time generated event packets to a gateway over a fading wireless channel. We are interested in designing efficient transmission control policies that can achieve optimal mean–variance trade-off in packets delivery delay, by judiciously consuming the ambient energy harvested from the environment. Our specific contributions can be summarized as follows:

- We come up with three problem formulations all concerning the delay-sensitive transmission of event data. All problems are based on the MDP formalism [25] to take into account the random nature of the captured events, harvested energy, as well as channel variations. Our first formulation is a CMDP [32] and aims at the minimization of the long-run average energy consumption subject to a prespecified constraint over the average length of the transmission buffer. The second formulation concerns the minimization of the long-run average buffer length (as the delay measure) while heeding the restrictions imposed by the energy harvesting process. The third formulation is an instance of the so-called variance-penalized MDP [33] with a system cost composed of the average buffer length (as the delay measure) linearly combined with the variance of the buffer length (as an indicator of jitter or inter-packet delay variation). A penalty weight parameter is introduced to balance these two aspects of the system cost.
- We propose a suite of Q-learning-based algorithms (from the RL theory) [27] to solve the above-mentioned problems in a model-free fashion in the absence of prior statistical information about the underlying processes that govern the channel, event occurrences, or harvested energy dynamics. In particular, to solve the delay-constrained problem, we employ the Lagrangian technique [34] to re-express the Bellman equations underlying the CMDP formulation in non-constrained form. This technique simplifies solving the original CMDP problem by finding the solution of two coupled optimizations over two distinct spaces of the control policies and the Lagrangian multiplier. The control policy is estimated iteratively via a Q-learning-based procedure on a fast timescale and the Lagrange multiplier is estimated using stochastic gradient descent on a slower timescale. The separation of timescales is a well-established technique from the stochastic approximation theory [35], which avoids divergence in coupled updating procedures. As for our variance-penalized formulation, we also deploy a two-timescale

learning procedure along the lines of [33]. In particular, to compute the variance of the IoT node's transmission buffer length, an estimate of the mean buffer length should already be available. This estimate can be obtained by running a slow-timescale moving average over the instantaneous buffer lengths. Q-learning can then be iterated on a fast timescale to estimate the state-action values of the underlying MDP based on samples of costs as well as their squared deviations from the estimated mean (as samples of variance).

- We present extensive numerical experiments to exhibit the convergence properties of all the learning algorithms. We also investigate the impact of various system parameters such as data arrival rate, energy harvesting rate, and renewable battery capacity on the performance of the three transmission policies in terms of energy consumption, delay, and delay variation.

1.5 Outline

The rest of the paper is organized as follows: In Sect. 2, our system model is described. Next, in Sect. 3, problem formulations are given accompanied later by their corresponding learning-theoretic solutions. In Sect. 4, we present and discuss our simulation results. The paper concludes in section.

2 System model

In this section, we first introduce our system model for all the three methods and then elaborate on the assumptions made about the dynamics of the energy charging process, sensory data generation, and variations in the wireless channel state. The list of notations is given in Table 2. In our proposed system, wireless IoT nodes, which are responsible for sensing and reporting from the environment, are equipped with multiple sensors, as illustrated in Fig. 1.

In a real operating environment, where there is no external power supply for each node, we assume that the IoT sensor nodes subsist on rechargeable batteries and energy harvesting circuits. These nodes decide intelligently on the number of units data (i.e., packets) that should be sent out over the wireless channel to the IoT gateway.

2.1 Channel model

We assume that time is slotted (indexed by n) with normalized slot durations. Data are transmitted over a wireless channel with potentially varying channel conditions from one slot to the next. We assume block-fading channels that remain constant within each timeslot. To model the time-varying quality of the wireless channel, we assume that the channel quality gain, h_n, is known by the transmitter at the start of each timeslot.

Table 2 Summary of the notations use in the system model

Symbol	Description
τ	Time slot duration
n	Time slot index
e_n^{in}	Amount of harvested solar energy in slot n
e_n^{out}	The number of consumed energy quanta in time slot n
e_n	The energy buffer state in time slot n
E	The energy buffer capacity
EU	Energy quantum size (J)
d_n	The data buffer input size in time slot n
b_n^{out}	The transmission data size
b_n^{in}	The data buffer input size in time slot n
b_n	The data buffer state in slot n
L	Length of a data packet
B	Data buffer capacity
h_n	Wireless channel state in time slot n
W	Channel bandwidth
P	The transmission power
E_{TX}	The transmission energy

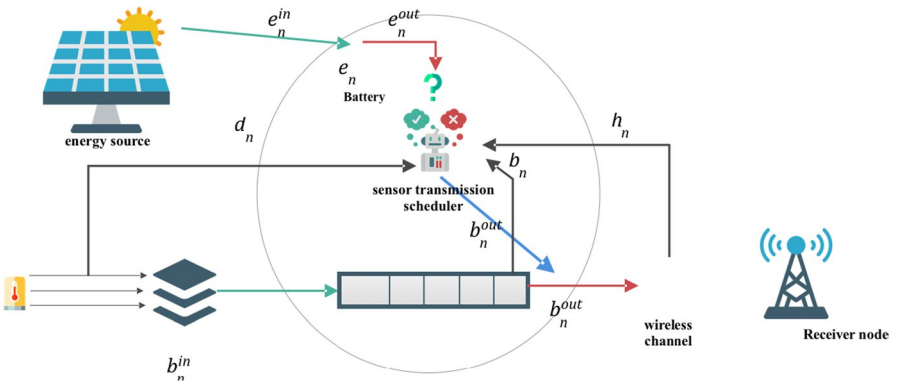

Fig. 1 Schematic of the IoT device

Assumption 1 (*Channel model*) For simplicity, suppose that $h_n \in \mathrm{H}$ and that H denotes a finite state space. In fact, similar to [36], we assume a quantized model for the channel's random status whose dynamics follow a Markovian process. We further assume that the wireless link quality remains unchanged throughout a timeslot, but may change randomly from one slot to another. According to [31], for reliable and error-free transmission of b_n^{out} Packets (each being L bits in size) over a link with bandwidth W, when the channel state is h_n, we need the power determined by Eq. 1:

$$p\left(h_n, b_n^{\text{out}}\right) = \frac{\text{WN}_0}{h_n}\left(2^{\frac{b_n^{\text{out}} L}{W}} - 1\right), \tag{1}$$

where N_0 represents white Gaussian noise with mean zero and variance N_0^2, and the product of WN_0 is normalized to 1. In this model, the transmission power is a strictly increasing function of b. Note that, Eq. 1 assumes h_n is expressed in dB. Equivalently, we may use Eq. 2 to express things in Watts:

$$p\left(h_n, b_n^{\text{out}}\right) = \frac{\text{WN}_0}{10^{\frac{h_n}{10}}}\left(2^{\frac{b_n^{\text{out}} L}{W}} - 1\right). \tag{2}$$

2.2 Energy model

2.2.1 Energy source model

It is assumed that the required power for the IoT nodes is supplied through harvesting energy from the environment, as illustrated in Fig. 2. The harvested energy is modeled as a stochastic process in which fixed-length energy quanta (packets) of size EU are received from the environment. According to the environmental conditions of the energy source (e.g., solar energy depends on the time of day, weather conditions, etc.), the received energy quanta vary over timeslots.

Assumption 2 (*Energy source model*) $\left\{e_n^{in}\right\}_{n\in\mathbb{N}}$ denotes the random number of harvested energy packets, measured in packets of size EU. In addition, the energy source model evolves as a finite-state Markov chain (FSMC), which takes values from the finite space $\chi = \{0, 1, 2 \ldots X\}$.

2.2.2 Energy harvesting model

We assume that the battery energy is stored in the form of energy packets. Let $\varepsilon = \{0, 1, 2 \ldots E\}$ denote the number of energy packets that are available for harvesting in the n-th timeslot. The "energy state information" (ESI) dynamics can be expressed by Eq. 3:

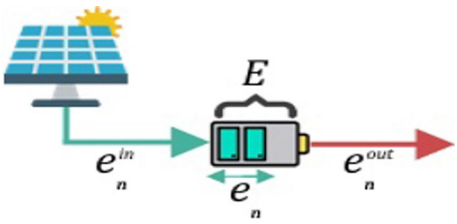

Fig. 2 Energy harvesting storage model in IoT node

$$e_{n+1} = \min\left(\max\left(e_n - e_n^{\text{out}}, 0\right) + e_n^{\text{in}}, E\right). \tag{3}$$

The required energy to send b_n^{out} data packets is calculated as the product of power and time which is formulated in Eq. 4:

$$E_{\text{TX}}\left(h_n, b_n^{\text{out}}\right) = P\left(h_n, b_n^{\text{out}}\right) \times \tau. \tag{4}$$

For simplicity, we assume that the transmission energy $E_{TX}(h_n, b_n^{\text{out}})$ is an integer multiple of energy packets. Note that, we only allow the transmission of b_n^{out} packets such that $E_{\text{TX}}\left(h_n, b_n^{\text{out}}\right) \leq e_n$.

2.3 Traffic model

The sensors on the IoT device generate a random number of data packets during every timeslot. The state space of the number of generated data packets is represented as a discrete and finite space $D = \{0, 1, 2, \ldots, D\}$. The number of input data packets at time n is denoted by $d_n \in \mathcal{D}$.

The data for transmission are queued (with FIFO discipline) in a buffer. The data "queue state information" (QSI) are represented by a finite discrete state space $\mathcal{B} = \{0, 1, 2, ..., B\}$. Also, the queue state at the start of timeslot $n + 1$ can be computed through Eq. 5:

$$b_{n+1} = \min\left(\max\left(b_n - b_n^{\text{out}}, 0\right) + b_n^{\text{in}}, B\right), \tag{5}$$

where $b_n \in \mathcal{B}$ denotes the number of packets currently in the queue. The number of packets sent out from the queue (during the timeslot n) is denoted by b_n^{out}. As mentioned before, b_n^{in} represents the received packet from IoT sensors, while b_n^{out} is determined by "transmission control."

3 Problem definition and proposed solution

In the absence of the statistical knowledge of the underlying system dynamics, making myopic (i.e., instantaneously greedy) decisions for the transmission policy cannot lead to a long-term optimal transmission control policy. In our proposed approach, we are not looking for greedy policies that transmit packets only with enough energy. When the channel conditions are not favorable, it is best not to send further packets to the channel to avoid wasting the harvested energy. Conversely, when the battery is fully charged, sending a packet can be helpful to release space and make room for more energy to be harvested in the future; otherwise, some energy will be lost due to the finite battery size. Therefore, the control policy needs to be based on the dynamic state of the system, i.e., the Data State Information (DSI), the Channel State Information (CSI), the Energy State Information (ESI), the Queue State Information (QSI), and the Battery State Information (BSI). In particular, adaptation to CSI is essential both for optimal utilization of the channel dynamic and for achieving more value for the

power invested. DSI and QSI-adaptability are required to obtain a delay-aware policy under the conditions of unsaturated traffic and finite-length queue at the node. Also, to increase the chances of having a sufficient amount of energy for transmission purposes, an ESI- and a BSI-adaptive policy is required to prevent unwanted usage of harvested energy. In fact, "transmission control" is a sequential decision problem because the node sometimes has to make instantaneous decisions, though it can obtain better long-term performance by inducing the system states to transition to more desirable states.

Optimal transmission control policies in sequential decision problems can be obtained systematically by casting the problem in a stochastic optimization framework. Given the Markovian nature of our setting (c.f., Assumptions 1 and 2), we resort to the notion of MDPs from control theory. MDP is a discrete-time stochastic control process that can be used for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. At each time step, the process is in some state s, and the decision-maker may choose any action a that is available in state s. The process responds at the next time step by randomly moving into a new state s', and giving the decision-maker a corresponding cost (as function of both the current state and the current action). The probability that the process moves into its new state s' is influenced by the chosen action. Specifically, it is given by a state transition function. Thus, the next state s' depends on the current state s and the decision-maker's action a. But given s and a, it is conditionally independent of all previous states and actions; in other words, the state transitions of an MDP satisfy the Markov property. In the sequel, we come up with three formulations based on the notion of MDP:

1. In the *Delay-Constrained* method, our objective is to design a transmission control policy that minimizes the expected cumulative power expenditure of the device while satisfying a certain average delay constraint for the sensory events queued in the transmission buffer. We formulate the problem of minimizing energy usage by IoT devices as a CMDP [37]. To do this, we transform the expected long-term cumulative energy usage into a medium-term constraint over the transmission queue.
2. In the *Delay-Centric* method, we propose to minimize the average buffer length that will ultimately minimize the packet delay perceived by the receiver. In this approach, we formulate the buffer length minimization problem as an MDP [38]. It is worth mentioning that only a small number of related papers have dealt with packet delay as perceived by the receiver.
3. In the *Variance-Penalized* method, we aim to minimize the data buffer delay variance that will ultimately minimize the packet delay variation perceived by the receiver. We formulate this problem as a Variance-Penalize MDP [33]. To the best of our knowledge, the consideration of delay variation has been largely neglected in the previous work.

3.1 Delay-constrained transmission control

In this method, we attempt to optimize average energy usage subject to a delay constraint using the CMDP.

3.1.1 CMDP-based formulation

Using the system model presented in Sect. 2 and illustrated in Fig. 3, we formulate the transmission control problem using the CMDP formalism. The CMDP can be described by a tuple $< S, A, P, C_E, C_B >$, where S, A, P, C_E and C_B denote a set of states, a set of control actions, transition probabilities, energy cost, and buffering cost, respectively. More specifically, the system state space is a discrete and finite space that is defined as the Cartesian product of CSI, DSI, QSI, ESI, and BSI spaces; i.e., $S = H \times D \times B \times \varepsilon \times X$. Also, $s_n = \left(h_n, d_n, b_n, e_n^{in}, e_n\right)$ represents the system state at time n. In the following, a CMDP-based formulation can be characterized by five elements: actions, transition probabilities, immediate cost/constraint functions, and the optimization objective:

- *Actions* According to the current system state s_n, the decision-maker determines a set of feasible control actions in each timeslot. The space of possible actions in s_n is a discrete and finite space $A_{S_n} = B_{S_n}$ in which $B_{S_n} \subset B$ is a set of data packets that can be transmitted out of the buffer (which depends on the CSI h_n, BSI e_n, as well as QSI b_n). We have to determine whether a control action b^{out} can be included in the feasible set B_{S_n}:

$$b^{\text{out}} \in B_{s_n} \begin{cases} 1 & if\ E_{\text{TX}}\left(h_n, b^{out}\right) \leq e_n * EU\ and\ b^{\text{out}} \leq b_n \\ 0 & otherwise \end{cases} \tag{6}$$

According to Eq. 6, the required transmission energy is compared with the available energy level in the battery to identify feasible actions. In case there is enough energy, the node can take an action in the state s_n. The current action of the decision-maker is denoted as a_n which is defined by $a_n = b_n^{\text{out}} \in A_{s_n}$.

- *System transition law* When an action a_n is executed in the state s_n, the system state transits to the next state s_{n+1}. The transition probability $P(s_{n+1}|s_n, a_n)$ is determined by Eq. 7:

$$P\left(s_{n+1}|s_n, a_n\right) = P\left(h_{n+1}|h_n\right).P\left(b_{n+1}|b_n, d_n, b_n^{\text{out}}\right).P\left(e_{n+1}|e_n, e_n^{\text{in}}, a_n\right).P\left(e_{n+1}^{\text{in}}|e_n^{\text{in}}\right) \tag{7}$$

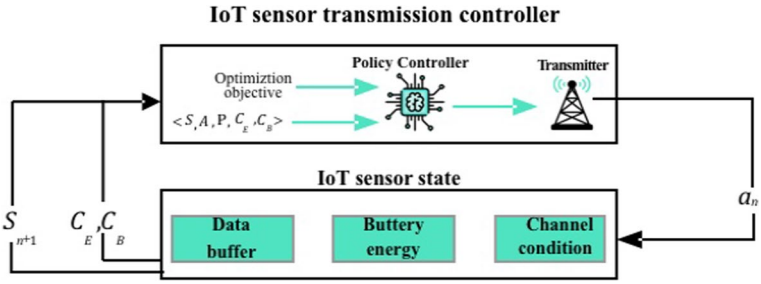

Fig. 3 IoT sensor controller

According to Eq. 7, h_n affects the next data buffer state implicitly. In addition, the energy source state e_n^{in} impacts e_{n+1} by affecting a_n. Note that, in the above transition law, the feasible action $a_n = b_n^{\text{out}}$ is based on the impact of the CSI h_n on the next QSI b_{n+1} as well as on the next BSI e_{n+1}.

- *Immediate cost and constraint function* The immediate cost of taking action a_n in state s_n is defined as the total energy consumption cost denoted by $C_E(s_n, a_{s_n})$ and calculated by Eq. 8:

$$C_E\left(s_n, a_{s_n}\right) = E_{\text{TX}}\left(h_n, b_n^{\text{out}}\right). \tag{8}$$

In the CMDP formulation [37], the immediate constraint function $C_B(s_n, a_n, s_{n+1})$ needs to be defined in a way that could keep the system from violating a certain reporting delay threshold for the sensory events. To achieve this purpose, Little's law [39] is employed to compute a measure of the mean delay experienced by the reported events. Therefore, the average buffer length $\bar{C}_B$ is equal to the product of the average arrival rate of sensory data $\bar{a}$ and the average delay experienced by the packets in the buffer $\bar{D}$, as shown in Eq. 9:

$$\bar{C}_B = \bar{a}\bar{D} \tag{9}$$

In Eq. 9, the constant $\bar{a}$ can be ignored. Hence, $\overline{C}_B$ can be used as a good measure of the average delay $\overline{D}$. We consider buffer occupancy as an immediate constraint. The average data buffer delay should not exceed a certain threshold; thus, we need to keep track of the instantaneous buffer occupancy as an immediate constraint as defined in Eq. 10:

$$C_B(s_n, a_n s_{n+1}) \triangleq b_{n+1} \tag{10}$$

- *Optimization objective* In our proposed CMDP formulation, the objective is to minimize the long-term average cost of the energy usage $\overline{C}_E$, while maintaining the average buffer length $\overline{C}_B$ under a given application-specific threshold δ. This parameter can have a variable value depending on the amount of delay tolerance threshold in practical applications. More formally, to control the transmission rate, the IoT node should learn an optimal policy π^*, so that for all states, we have

$$\begin{aligned} &\pi^*(s) \arg\min_{\pi} \bar{C}_E^{\pi} \triangleq \lim_{n\to\infty} \frac{1}{n}\mathbb{E}^{\pi}\left[\sum_{n=1}^{\infty} C_E\left(s_n, a_{s_n} | s1 = s\right)\right] \\ &\text{s.t. } \bar{C}_B^{\pi} \triangleq \lim_{n\to\infty} \frac{1}{n}\mathbb{E}^{\pi}\left[\sum_{n=1}^{\infty} C_B\left(s_n, a_{s_n}, s_{n+1} | s1 = s\right)\right] \leq \delta \end{aligned} \tag{11}$$

where the functions $\overline{C}_E^{\pi}$ and $\overline{C}_B^{\pi}$ are time averages over an infinite horizon. The threshold δ is assumed to be specified by the system designer to reflect the amount of delay tolerance in practical applications.

3.1.2 The Lagrangian technique

To convert the optimization problem into its unconstrained equivalent, the standard Lagrangian technique offers a viable solution [34]. We introduce a Lagrange multiplier $\lambda > 0$ to linearly combine the objective function in Eq. 11 with its constraint and present $\overline{\mathcal{L}}^{\lambda,\pi}$ as a new combined cost function, which is expressed by Eq. 12:

$$\overline{\mathcal{L}}^{\lambda,\pi}(s) \triangleq \overline{C}_E^{\pi} + \lambda\left(\overline{C}_B^{\pi} - \delta\right) \forall s \in S \tag{12}$$

Intuitively, in Eq. 12, if the average cost buffer-length $\overline{C}_B^{\pi}$ exceeds the threshold value δ, the resultant difference will be added by a positive coefficient to the system cost to further penalize the controller. Given $\overline{C}_E^{\pi}$ and $\overline{C}_B^{\pi}$, the function $\overline{\mathcal{L}}^{\lambda,\pi}(s)$ is also a long-term average, and the immediate Lagrangian $l(s, a, \lambda)$ corresponding to it can be defined by Eq. 13:

$$l(s, a, \lambda) = C_E(s, a) + \lambda(C_B\left(s_n, a_{s_n}, s_{n+1}\right) - \delta) \tag{13}$$

Now, according to Eq. 13, $\overline{\mathcal{L}}^{\lambda,\pi}(s)$ in Eq. 12 can be rewritten as Eq. 14:

$$\overline{\mathcal{L}}^{\lambda,\pi}(s) \triangleq \lim_{n\to\infty}\left(\frac{1}{n}\right)\mathbb{E}^{\pi}\left[\sum_{n=1}^{\infty} l\left(s_n, a_n, \lambda | s1 = s\right)\right] \quad \forall s \in S \tag{14}$$

From Eq. 11, we note that for a feasible policy π, we have Eq. 15:

$$\overline{C}_B^{\pi} \leq 0 \quad (\forall s \in S) \tag{15}$$

As a result,

$$\overline{\mathcal{L}}^{\lambda,\pi}(s) \leq \overline{C}_E^{\pi} \quad \forall (s \in S) \quad \text{for all feasible } \pi \tag{16}$$

Consequently, by minimizing the above inequality over the space of all feasible policies, we have

$$\min_{\pi} \mathcal{L}^{-\lambda,\pi}(s) \leq \min_{\pi} \overline{C}_E^{\pi} \quad \forall (s \in S) \quad \text{for all feasible } \pi \tag{17}$$

Hence $\min_{\pi} \overline{\mathcal{L}}^{\lambda,\pi}(s)$ gives a lower bound for the value of the objective function in Eq. 11 by finding the greatest lower bound, i.e., by maximizing λ in $\min_{\pi} \overline{\mathcal{L}}^{\lambda,\pi}(s)$, the best lower bound can be obtained for the constrained problem defined in Eq. 11. Therefore, instead of solving the problem in Eq. 11, we may now deal with the following new unconstrained optimization problem

$$\max_{\lambda} \min_{\pi} \overline{\mathcal{L}}^{\lambda,\pi}(s), \quad \forall (s \in \mathrm{S}) \tag{18}$$

The pair (π^*, λ^*) is defined as a solution for Eq. 18. According to [40], if the immediate cost function $C_E\left(s_n, a_n\right)$ and the immediate constraint $C_B\left(s_n, a_n, s_{n+1}\right)$

are both convex, there is no difference between the value $\overline{C}_E^{\pi}$, obtained from the constrained problem Eq. 11, and the value $\overline{\mathcal{L}}^{\lambda^*,\pi^*}$ obtained from Eq. 18. In our formulation, functions $C_E(s_n, a_n)$ and $C_B(s_n, a_n, s_{n+1})$ are strictly convex. Consequently, by solving the two-optimization problems given in Eqs. 19 and 20, the optimal policy can be achieved by

$$\pi^* \in \operatorname{argmin}_{\pi} \overline{\mathcal{L}}^{\lambda,\pi}(s) \tag{19}$$

$$\lambda^* \in \operatorname{argmax}_{\lambda} \overline{\mathcal{L}}^{\lambda,\pi^*}(s) \tag{20}$$

In Sect. 4.1.3, we propose a model-free reinforcement learning technique to calculate the optimal solution pair (π^*, λ^*).

3.1.3 Learning the optimal transmission control policy

We apply a learning algorithm consisting of a coupled recursion that iteratively estimates π^* and λ^* for simultaneously solving Eqs. 19 and 20. For this purpose, first, the standard Bellman Eq. 36 and then Q-learning [41] are applied. We denote the so-called Q-function by $Q^{*,\lambda}(s, a)$, which is defined as the sum of the immediate differential cost $l(s, a, \lambda) - {C_B}^*$, obtained by taking action a in the current state s, with the long-term Lagrangian realized from the next state onward (by following the optimal policy π^*). More specifically, we have

$$Q^{*,\lambda}(s, a) = l(s, a, \lambda) - \overline{\mathcal{L}}^{*,\lambda} + \sum_{s_{n+1} \in S} P(s_{n+1}|s, a)\overline{\mathcal{L}}^{\lambda,\pi^*}(s_{n+1}), \tag{21}$$

where $\overline{\mathcal{L}}^{*,\lambda}$ denotes the optimal per-stage Lagrangian. Additionally,

$$\overline{\mathcal{L}}^{\lambda,\pi^*}(s) = \min_{a \in A(s_{n+1})} Q^{*,\lambda}(s, a), \quad \forall s \in S \tag{22}$$

The optimal policy $\pi^{*,\lambda}(s)$ (parameterized with λ) leads to Eq. 23:

$$\pi^{*,\lambda}(s) = \operatorname*{argmin}_{a \in A(s_{n+1})} Q^{*,\lambda}(s, a), \quad \forall s \in S \tag{23}$$

Now, in model-based scheme, the knowledge of the probability distribution P is a requirement to solve the Bellman equation in Eq. 21. However, Q-learning estimates all $Q^{*,\lambda}(s, a)$ values through repeated updates. In particular, the IoT node initializes an estimated table $\widehat{Q}_n(s, a)$ for all (s, a) pairs with arbitrary values in the first step. Then, it observes the current system state and chooses an action from its action space at each iteration. Our proposed Q-learning equation is based on average cost, so, instead of using the discount factor, we need the per-stage cost $\overline{\mathcal{L}}^{*,\lambda}$ to obtain the differential cost at each iteration. To estimate $\overline{\mathcal{L}}^{*,\lambda}$, we need to select some reference

state-action value $Q_n(s^*, a^*)$ to obtain convergence in our simulation [42]. Now the node implements the selected action, computes its immediate cost as $l(s, a, \lambda)$, and then updates its estimate $\widehat{Q}_n(s, a)$ for the current pair (s, a) according to the following rule:

$$\widehat{Q}_n(s,a)\big(1-\beta_n\big)\widehat{Q}_{n-1}(s,a)+\beta_n[l(s,a,\lambda)+\min_{a_{n+1}\in A\left(s_{n+1}\right)}\widehat{Q}_{n-1}\left(s_{n+1},a_{n+1}\right)-Q_n(s^*,a^*)] \tag{24}$$

In Eq. 24, $\widehat{Q}_n(s, a)$ represents the n-th time estimation of $Q^{*,\lambda}(s, a)$ and β_n denotes the step size (learning rate) of Q-learning, which is computed for each (s, a) pair according to Eq. 25:

$$\beta_n=\frac{1}{1+(\text{visit}_n(s,a))^{0.65}}. \tag{25}$$

In Eq. 25, $\text{visit}_n(s, a)$ is the number of times the pair (s, a) has been observed up to iteration n. To guarantee convergence of the above Q-learning algorithm, it is necessary that an action is chosen at each step in the so-called ϵ-greedy fashion (e.g., see [43]). The symbol ϵ denotes a small probability with which an RL agent explores the unknown environment from time to time. In particular, the controller occasionally needs to take random (yet feasible) actions to further explore the quality of an alternative choice in addition to taking greedy actions according to the optimal policy being estimated (i.e., $\pi^{*,\lambda}(s)$ from 23). By implementing such ε-greedy action selection policy, the described Q-learning algorithm is guaranteed not only to converge to $\pi^{*,\lambda}(s)$, but also to the minimum Lagrangian $\overline{\mathcal{L}}^{\lambda,\pi^*}(s)$ for a constant value λ (see [12]).

Having obtained $\overline{\mathcal{L}}^{\lambda,\pi^*}(s)$, we deal with Eq. 20 as a maximization problem. Since we have no knowledge of the transition laws, Eq. 20 cannot be solved by setting to zero the derivative of Eq. 12 w.r.t. λ. Hence, we need to deploy an iterative scheme to estimate λ^* as well. In particular, we define $\widehat{\lambda}_{n+1}$ as the n-th estimate of λ^* and apply a standard stochastic subgradient ascent algorithm to derive $\widehat{\lambda}_n$ toward optimal value; i.e.,

$$\widehat{\lambda}_{n+1}=\Lambda[\widehat{\lambda}_{n+1}+\alpha_n(C_B\left(s_n,a_n,s_{n+1}\right)-\delta)] \tag{26}$$

where α_n is the learning rate (step size) that is computed by Eq. 17:

$$\alpha_n=\frac{1}{n+1} \tag{27}$$

The operator $\Lambda[.]$ is defined to be $\max(., 0)$ and is basically a projection operator meant for keeping $\widehat{\lambda}_n$ from ever becoming negative. The term $\left(C_B\left(s_n,a_n,s_{n+1}\right)-\delta\right)$ is an instantaneous estimate of the gradient direction for the function $\overline{\mathcal{L}}^{\lambda,\pi^*}$. Its form corresponds to the derivative of Eq. 12 w.r.t. λ, although, unlike an exact derivative, it is only a noisy instantaneous estimate. This justifies our application of the "stochastic" subgradient method to guarantee gradual convergence toward optimal

λ^*. Rigorous proofs in terms of generic CMDP formulations concerning the convergence of the estimated pair $(\widehat{Q}_n, \widehat{\lambda}_n)$ to their optimal values (Q^*, λ^*) are given in references [39]. However, making concurrent estimates of λ^* and Q^* by using the learning rules in Eqs. 24 and 26 need further clarification. It should be noted that $\widehat{\lambda}_n$ and $\widehat{Q}_n$ are coupled together, by definition. However, we have allowed for their simultaneous updating, which seemingly works against each other by creating a moving target for one another. The perfect way to with these two updates concurrently is to operate them on two different time scales. More formally, we assume the update rates to be $\beta(n)$ and $\alpha(n)$, such that they satisfy some standard conditions from the theory of stochastic approximation for these two estimates:

$$\sum_n (\beta(n)^2 + \alpha(n)^2) < \infty, \lim_{n\to\infty}\left(\frac{\alpha(n)}{\beta(n)}\right) \to 0 \tag{28}$$

If the conditions in Eq. 28 are satisfied, the estimate $\widehat{\lambda}_n$ will be updated over a shorter time interval compared to $\widehat{Q}_n$. In this way, $\widehat{Q}_n$ appears to be equilibrated (or convergent) to the update rule for $\widehat{\lambda}_n$, while from its viewpoint, the estimate $\widehat{\lambda}_n$ appears to be quasi-static [44]. The pseudo-code of the learning algorithm to find the appropriate action policy is presented in Fig. 4.

3.2 Delay-centric transmission control

In this method, our objective is to optimize the average delay while respecting the energy availability constraint.

Algorithm 1: Pseudo code of the *Delay-Constraint* method

Initialization:
$\hat{Q}(s,a) = 0 \quad \forall s, a$
$\hat{\lambda} = 0$
$count = 0$ (state-action counter)
$n = 1$ (Initial iteration number)
λ= Some (time) average buffer threshold
Initialize $s_1 = (h_1, b_1, d_1, e_1)$ arbitrarily
Identify feasible action $A(s_1)$ using Eq. 7
Procedure: Main Learning Loop
  While $n <$ Max_iteration_number **do**
    Select an Action from the set $A(s_n)$ in ε-greedy fashion
    $a_{sn} = (b_n^{out}) \leftarrow \begin{cases} \arg\min Q & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
    Update *counter*
    Compute $C_B(s_n, a_n, s_{n+1})$ cost using Eq.8
    Compute $C_E(s_n, a_n, s_{n+1})$ cost using Eq.10
    Compute Lagrangian as $l(s,a,\lambda)$ using Eq.13
    Observe the next system state s_{n+1}
    Identify feasible action $A(s_{n+1})$ using Eq.6
    Compute step size β_n using Eq.25
    Update $\hat{Q}_n(s,a)$ using Eq.24
    Compute step size α_n using Eq.27
    Update $\hat{\lambda}_{n+1}$ using Eq.26
    Update n

Fig. 4 Algorithm 1

3.2.1 MDP-based formulation

In the Delay-Centric method, we formulate the optimization problem by using the MDP formalism in terms of the tuple $< S, A, P, C_B >$. The space of the system state is $S = H \times D \times B \times X$. Likewise, $S_n = \left(h_n, d_n, b_n, e_n\right) \in S$ represents the system state at time n. The action and system transition laws are similar to the "Delay-Constrained" method. In the sequel, we elaborate on the immediate cost and optimization objective:

- Immediate *cost* The immediate cost of taking action a_n in the state s_n is defined to be the data buffer length denoted by $C_B(s_n, a_n, s_{n+1})$ and calculated by Eq. 29:

$$C_B(s_n, a_n, s_{n+1}) = b_{n+1} \tag{29}$$

- *Optimization objective* The goal of our MDP formulation is to minimize the long-run average data buffer length. Hence, in this method, we seek an optimal policy π^* to control the transmission rate for all $s \in S$ such that

$$\pi^*(s) \in \underset{\pi}{\operatorname{argmin}}\, \bar{C}_B^{\pi} \triangleq \lim_{n\to\infty} \frac{1}{n} \mathbb{E}^{\pi}\left[\sum_{n=1}^{\infty} C_B\left(s_n, a_{s_n} | s1 = s\right)\right] \tag{30}$$

3.2.2 Bellman equation

Similarly to the previous scheme, we need to come up with the Bellman equations associated with the optimization problem in (30) in order to determine the necessary condition for the optimal transmission control policy. The Q-factor version of the Delay-Centric Bellman equation for all (s, a) pairs is as expressed by Eq. 31:

$$Q(s,a) = \left[C_B\left(s_n, a_{s_n}, s_{n+1}\right) - Q(s^*, a^*)\right] + \min_{a \in A(s)} \sum_{s \in S} P\left(s, a, s\right) Q\left(s, a\right) \tag{31}$$

3.2.3 Learning the optimal transmission control policy

We present an algorithm to minimize the average data buffer length. Unlike the previous method, here, the algorithm will only be based on immediate costs and there is no constraint to be respected by the optimal transmission control policy. According to Eq. 32, $\widehat{Q}_n(s,a)$ for the current (s,a) pair is estimated in each iteration as follows:

$$\widehat{Q}_n(s,a) \leftarrow \left(1-\beta_n\right)\widehat{Q}_{n-1}(s,a) + \beta_n\left[C_B\left(s_n, a_n, s_{n+1}\right) + \min_{a_{n+1} \in A\left(s_{n+1}\right)} \widehat{Q}_{n-1}\left(s_{n+1}, a_{n+1}\right) - Q_n(s^*, a^*)\right], \tag{32}$$

In this algorithm, the only subtlety in updating the Q tables is to restrict the minimization (within the square brackets) to only the energy-feasible actions of the newly realized state.

3.3 Variance-penalized method

In this method, we aim to optimize the delay variation by using the formalism of Variance-Penalized MDP. As mentioned earlier in Sect. 1.3, the importance of such control is in avoiding the oscillating buffering behavior (i.e., overflow and underflow events).

3.3.1 Variance-penalized MDP-based formulation

Here, we use the formalism of variance-penalized MDPs [33] to describe the transmission control problem. In particular, this MDP can be described by the tuple $< S, A, P, C_B >$. The space of the system state is $S = H \times D \times B \times X$ and $S_n = \left(h_n, d_n, b_n, e_n\right) \in S$ represents the system state at time n as in the previous method. Action and system transition laws are similar to the first method (c.f., Sect. 3.4). The immediate cost and optimization objective are defined as follows:

Immediate cost The immediate cost of taking action a_n in state s_n is defined as the total buffer length cost denoted by $C_B(s_n, a_n, s_{n+1})$ and calculated by Eq. 33:

$$C_B\left(s_n, a_n, s_{n+1}\right) = b(n) + \theta(b_n - \mathscr{b})^2, \tag{33}$$

where $\mathscr{b}$ is the mean buffer length and $(b_n - \mathscr{b})^2$ denotes the immediate buffer length variance. The parameter θ is a positive scalar and represents a penalty factor to penalize high variance behavior.

Optimization objective In this method, our goal is to minimize a linear combination of the long-term average data buffer length and its variance. The penalty factor θ serves to quantify the degree of risk aversion, and the resulting metric (average cost plus θ times the variance) serves to identify policies that seek a compromise: a low (but not necessarily the lowest) average cost with low variance. More formally, we seek to minimize (over all $\pi \in P$):

$$\pi^*(s) \in \min_{\pi} \varphi_\pi = \mathscr{b}^\pi + \theta(\delta_B^\pi)^2, \tag{34}$$

where φ is equal to the variance-penalized score associated with the policy π, and $\mathscr{b}^\pi$ and is the mean data buffer length (under policy π). Also, θ takes small values (e.g., 0,1). The symbol δ_B^π denotes the long-term data buffer length variance (defined according to Eq. 36):

$$(\delta_B^\pi)^2 = \lim_{n\to\infty} \left(\frac{1}{n}\right) \mathbb{E}^\pi[\sum_{n=1}^{\infty} (b_n - \mathscr{b})^2 | s1 = s] \tag{35}$$

3.3.2 Bellman equation

In this method, a novel Bellman equation is developed that incorporates data buffer length variance and a contraction factor allowing the value function to remain bounded. Similarly to [33], we use the one-step data buffer length variance as a risk measure for infinite horizon and propose a policy iteration algorithm.

Assumption 3 Suppose that a scalar φ^* and function $J : S \rightarrow R$ exist that solve Eq. 36 for every $s \in S$:

$$\varphi^* + J(s) = \min_{n \in \mathcal{B}} \left[\min_{a \in A(s)} \left[\sum_{s \in S} P\big(s, a, s\big)[b + \theta(b - \not{a})^2 + J(s)] \right] \right] \tag{36}$$

so that there exists a minimizer $\not{a}$, solving Eq. 36. Moreover, $\not{a}$ equals the average reward of the policy π^* defined by Eq. 37:

$$\pi^*(s) \in \underset{a \in A(s)}{\operatorname{argmin}} [\sum_{s' \in S} P\Big(s, a, s'\Big)[b + \theta(b - \not{a})^2 + J(s')]] \tag{37}$$

The Q-factor version of variance-penalized Bellman equation for all (s, a) pairs is as expressed by Eq. 38:

$$Q(s, a) = \sum_{s \in S} P\big(s, a, s\big)[b + \theta(b - \not{a})^2 - \varphi^* + J(ss)]], \tag{38}$$

where J, π^*, and $\not{a}$ are as defined as in Assumption 3. Using the above definition, the Bellman equation, i.e., Eq. 36, can be rewritten as Eq. 39:

$$J(s) = \min_{a \in A(s)} Q(s, a) \tag{39}$$

When we combine Eq. 39 with the definition in Eq. 38, the Q-factor version of the Bellman equation can be written as in Eq. 40:

$$Q(s, a) = \sum_{s \in S} P\big(s, a, s\big)[b + \theta(b - \not{a})^2 - \varphi^* + \min_{b \in A(s)} Q(s, b)]] \tag{40}$$

Such that $\not{a}$ equals the average cost of the policy defined as follows:

$$\pi^*(s) \in \arg\min_{a \in A(s)} Q(s, a) \tag{41}$$

3.3.3 Learning the optimal transmission control policy

In this section, we present an RL algorithm, which is based on Q-learning, to solve the formulated variance-penalized MDP with unknown transition probabilities. The steps in our RL algorithm are as follows:

- Step 1 The IoT node initializes at any arbitrary state and also initializes the Q-table $\widehat{Q}_n(s,a)$ with zero or arbitrary values for all (s,a) pairs. The node then makes a transmission decision and the system transitions into a new state. At the new state s_{n+1}, $\widehat{Q}_n\left(s_n,a_n\right)$ for the current pair $\left(s_n,a_n\right)$ is updated as follows:

$$\widehat{Q}_n\left(s_n,a_n\right) \leftarrow \left(1-\beta_n\right)\widehat{Q}_{n-1}\left(s_n,a_n\right)+\beta_n \left[b_n+\theta(b_n-\widehat{\ell}_n)^2+\min_{a_{n+1}\in A\left(s_{n+1}\right)}\widehat{Q}_{n-1}\left(s_{n+1},a_{n+1}\right)-Q_n(s^*,a^*)\right], \tag{42}$$

where β_n is a learning rate that is calculated by Eq. 43:

$$\beta_n=\frac{\log(n+1)}{n+1}. \tag{43}$$

Note that, φ^* in Eq. 42 has been replaced by $Q_n(s^*,a^*)$ for some fixed (s^*,a^*) pair. In [42], it has been argued that this estimate of the Q-value can replace φ^* in the sense that once Q-learning converges, φ^* can be approximated as $\lim_{n\to\infty} Q_n(s^*,a^*)$. The symbol $\widehat{\ell}_n$ denotes the estimate of the mean buffer length that is be updated in Step 2.

- Step 2 The IoT node chooses the so-called greedy action that minimizes the Q-factor (i.e., belongs to the set $\arg\min_{a\in A(s_n)}\widehat{Q}_n\left(s_n,a\right)$) with probability $1-\frac{\overline{\epsilon}}{|A(s)-1|}$ and any one of the remaining $|A(s)-1|$ actions with probability $\frac{\overline{\epsilon}}{|A(s)-1|}$. Set $\overline{\epsilon}=\frac{C}{\overline{\gamma}(s)}$, where C is a selected value in the interval (0, 1) and $\overline{\gamma}(s)$ represents the number of times state s has been visited thus far. Moreover, $\widehat{\ell}_n$ is updated according to Eq. 44:

$$\text{if a greedy action was chosen, set } \widehat{\ell}_n=\widehat{\ell}_{n-1}+\alpha(n)\left[b_n-\widehat{\ell}_{n-1}\right]. \text{otherwise, set } \widehat{\ell}_n=\widehat{\ell}_{n-1} \tag{44}$$

where α_n is a learning rate calculated by Eq. 45:

$$\alpha_n=\frac{C_1}{C_2+n} \tag{45}$$

- Step 3 The node implements the selected action and computes its immediate buffer cost b_n.
- Step 4 Update the estimate $\widehat{Q}_n(s,a)$ for the current (s,a)pair by using Eq. 42.

Based on these explanations, the pseudo-code of the proposed learning algorithm for computing the optimal transmission control policy is given in Fig. 6.

4 Simulation results

In this section, we experiment with our proposed methods in an in-house simulation environment, where we simulate the point-to-point scenario depicted in Fig. 1. To model the system, we assume that the transmitter at the beginning of each timeslot knows the channel fading state and that the channel is divided into eight levels with boundaries as below:

$$\left\{\begin{array}{l}(-\infty,\ -13\text{dB}),[-13\text{dB},\ -8.47\text{dB}),[-8.47\text{dB},\ -5.41\text{dB}),\\ [-5.41\ \text{dB},\ -3.28\text{dB}),[-3.28\text{dB},\ -1.59\text{dB}),[-1.59\text{dB},\ -0.08\text{dB}),\\ [-0.08\ \text{dB}, 1.42\text{dB}),[1.42\text{dB}, 3.18\text{dB}),[3.18\text{dB}, \infty).\end{array}\right\}$$

The quantized levels are as follows:

$$\text{H} = \left\{\begin{array}{l}\text{h1} = -13\text{dB},\ \text{h2} = -8.47\text{dB},\ \text{h3} = -5.41\text{dB},\\ \text{h4} = -3.28\text{dB},\ \text{h5} = -1.59\text{dB},\ \text{h6} = -0.08\text{dB},\ \text{h7} = 1.42\text{dB},\ \text{h8} = 3.18\text{dB}\end{array}\right\}$$

We also simulate the energy harvesting scenario where the active-to-inactive or inactive-to-active transition occurs (on average) once within every 12 h, respectively. This scenario is meant to showcase a sunny day when (on average) the system stays in the active and the inactive states for around 12 h each. The rest of the simulation parameters are given in Table 3.

We make comparisons with two algorithms for dynamic transmission control presented in [9]. Similar to our schemes, the approaches in [9] are specifically proposed for energy-aware transmission control, but they only aim at minimizing the expected packet queuing delay given the available harvested energy. The RL algorithms proposed in [9] use a batch update technique and greedy action selection to achieve faster convergence, but as evidenced by our simulation results, these algorithms are not suited for minimizing delay variations.

4.1 Convergence properties

In this section, we explore the convergence properties of our three Q-learning-based algorithms: the "Delay-Constrained" method in Fig. 4, the "Delay-Centric" method in Fig. 5, and the "Variance-Penalized" method in Fig. 6. The plots in Fig. 7a, b and c represent the convergence trend of the proposed algorithms, which is a key property in reinforcement learning algorithms.

In Fig. 7a, the estimates of $\widehat{\lambda}_n$ are obtained from the iterative updates in Eq. 26 for the "Delay-Constrained" method. As can be seen, the Lagrange multiplier converges after approximately 9000 iterations. It is also clear that the constraint on the average buffer length is tightly satisfied.

In Fig. 7b, the sequence $\left\{\widehat{\mathscr{C}}_n\right\}_{n\geq 0}$ as the moving average cost of the policy converges to the actual mean value after approximately 15,000 iterations. The next plot in Fig. 7c verifies the convergence of the average energy usage of all five methods.

Table 3 Simulation Parameters

Parameter	Value	Description
$\boldsymbol{\tau}$	1msec	Time slot duration
n	$2e6$	Number of iterations
$\boldsymbol{X}$	42 pkts(Only Fig. 7) 6 pkts	Maximum number of harvestable energy packets in each iteration
EU	2000 μJ	Size of energy packets
$\boldsymbol{E}$	8 Energy pkts (Only Fig. 7) 9 Energy pkts	Energy buffer capacity
$\boldsymbol{D}$	44 pkts (Only Fig. 7) 5 pkts	Maximum number of data Packets sensed in each timeslot
$\boldsymbol{L}$	512 bits	Length of a data packet
B	8 data pkts (Only Fig. 7) 9 data pkts	Data buffer capacity
H	[– 13, – 8.47, – 5.41, – 3.28, – 1.59, – 0.08, 1.42, 3.18]	Wireless channel state space [12]
$\boldsymbol{W}$	5 MHZ	Channel bandwidth
$\boldsymbol{\delta}$	6.8 pkts	Data Buffer threshold
$\boldsymbol{\varepsilon}$	0.1	The probability of random selection in the ε-greedy method
$\boldsymbol{\theta}$	0.15	Penalty factor
$\overline{\boldsymbol{\gamma}(\mathbf{s})}$	1	The number of the times state s has been visited
$(\boldsymbol{s}^*, \boldsymbol{a}^*)$	(3, 8, 8, 10, 1)	Reference state-action pair
$\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a})$	MAXINT	The initial value of Q

Algorithm 2: Pseudo code of the *Delay-Centric* method

Procedure: Main Learning Loop
  While $n <$ Max_iteration_number **do**
    Select an Action from the set $A(s_n)$ in ε-greedy fashion
    $a_{sn} = (b_n^{out}) \leftarrow \begin{cases} \arg\min Q & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
    Update *counter*
    Compute $C_B(s_n, a_n, s_{n+1})$ cost using Eq.29
    Observe the next system state s_{n+1}
    Identify feasible action $A(s_{n+1})$ using Eq.6
    Compute step size β_n using Eq.25
    Update $\hat{Q}_n(s,a)$ using Eq.32
    Update n

Fig. 5 Algorithm 2

Algorithm 3: Pseudo code of the *Variance-Penalized* method

Initialization:
$\hat{Q}(s,a) = 0 \quad \forall s,a$
$n = 1$ (Initial iteration number)
$\overline{\gamma} = 1 \quad \forall s,a$
Initialize $s_1 = (h_1, b_1, d_1, e_1)$ arbitrarily
Identify feasible action $A(s_n)$ using Eq. 7
Procedure: Main Learning Loop
  While $n <$ Max_iteration_number **do**
    Select an Action from the set $A(s_n)$ in ε-greedy fashion
    $a_{sn} = (b_n^{out}) \leftarrow \begin{cases} \arg\min Q & \text{with probability } 1-\frac{\overline{\epsilon}}{|A(s)-1|} \\ \text{remaining} & \text{with probability } |A(s)| \end{cases}$
    $\overline{\epsilon} = \frac{C}{\overline{\gamma}(s)}$
    C is a selected value in interval (0,1)
    Compute $C_B(s_n, a_n, s_{n+1})$ average length buffer using Eq.44
    Observe the next system state s_{n+1}
    Identify feasible action $A(s_{n+1})$ using Eq.6
    Compute step size $\beta_n$using Eq.44
    Update $Q_n(s,a)$ using Eq.40
    Compute step size α_n using Eq.46
    Update $\hat{Q}_n(s,a)$ using Eq.43
    Update $\hat{\mathscr{V}}_n$ using Eq. 45
    Update n

Fig. 6 Algorithm 3

As can be seen, the "Delay-Centric" and "Variance-Penalized" methods perform almost similarly in terms of convergence, while the "Delay-Constrained" method converges to a lower average energy usage. The reason is that "Delay-Centric" and "Variance-Penalized" methods do not explicitly take into account the energy usage. Their objective is to come up with a transmission policy that either directly minimizes delay, or the linear combination of delay with its variation. This would cause the IoT node to consume more energy to serve its data queue. Also, with a limited iteration budget, both PDS- and VE-learning algorithms from [9] achieve a higher performance level (i.e., lower energy consumption).

In Fig. 7d, the average buffer length across 90,000 iterations is plotted. As can be seen, all five methods converge after 15,000 iterations (with VE-learning algorithm being the fastest). Note that, in the case of the "Delay-Constrained" method, the

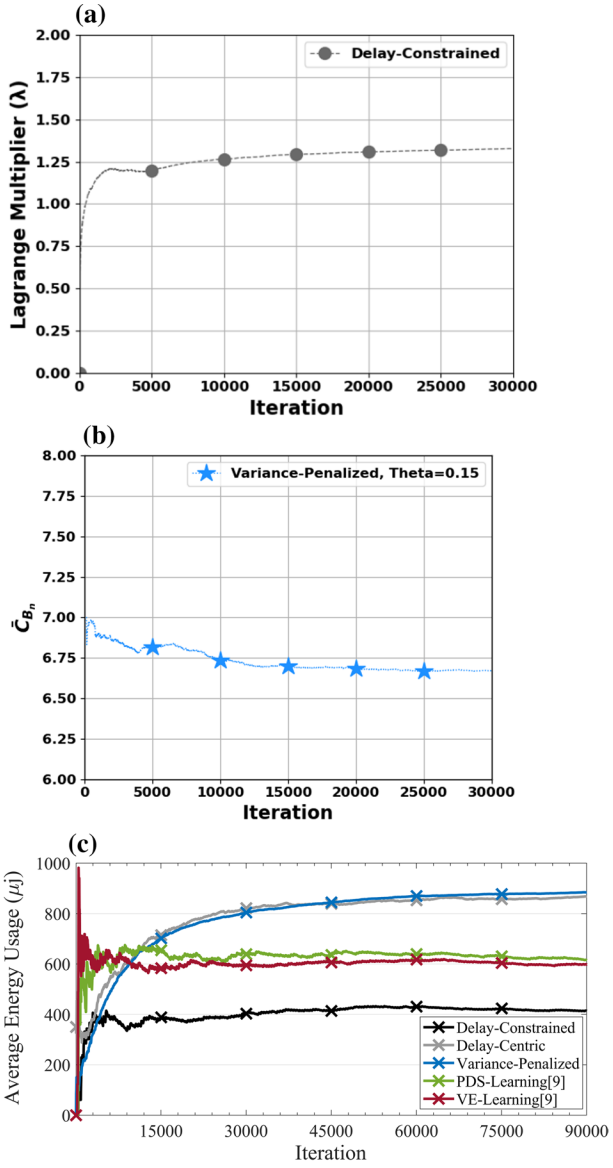

Fig. 7 **a** Convergence of Lagrange multiplier. **b** Convergence of average buffer length as the cost of a policy. **c** Convergence of average energy usage **d** Convergence of average data buffer length **e** Convergence of data buffer length variance

average buffer length completely satisfies the buffer constraint threshold (shown by a red line). Moreover, the convergence speed of this method is higher compared to that of our two other methods; however, these latter methods achieve a smaller

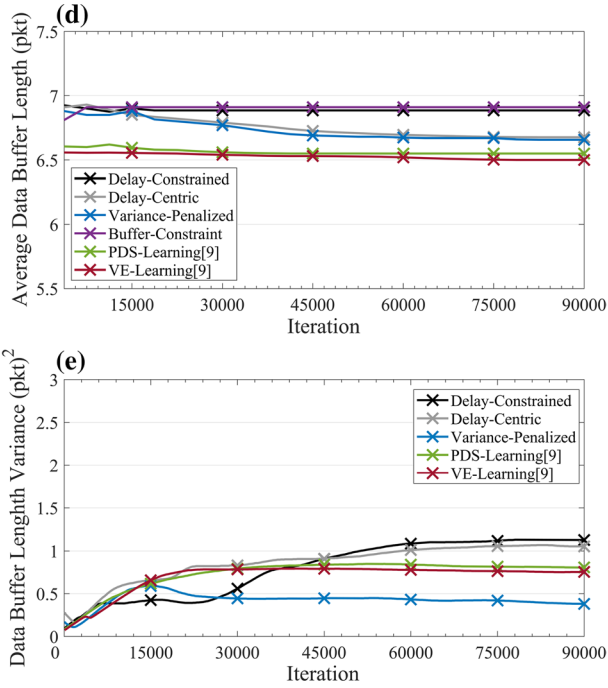

Fig. 7 (continued)

average data buffer length at the time of convergence. In this experiment, the threshold value is set to 6.9 and the average buffer length reaches 6.75.

In Fig. 7e, the impact of data buffer length is investigated. As can be seen, in the "Variance-Penalized" method, the data buffer length variance is much smaller than that of the other methods from the start until reaching a constant value. The PDS- and VE-learning algorithms from [9], despite achieving faster convergence, cannot guarantee low data buffer fluctuation levels.

4.2 Investigating the impact of other parameters

In this section, we compare the performance of the proposed schemes under three different scenarios, namely varying traffic data arrival levels, energy arrival levels, and maximum energy battery space—ranging from 8 to 20 energy packets. Next, the impact of each of these scenarios on average energy usage, average data buffer length, and data buffer length variance is investigated.

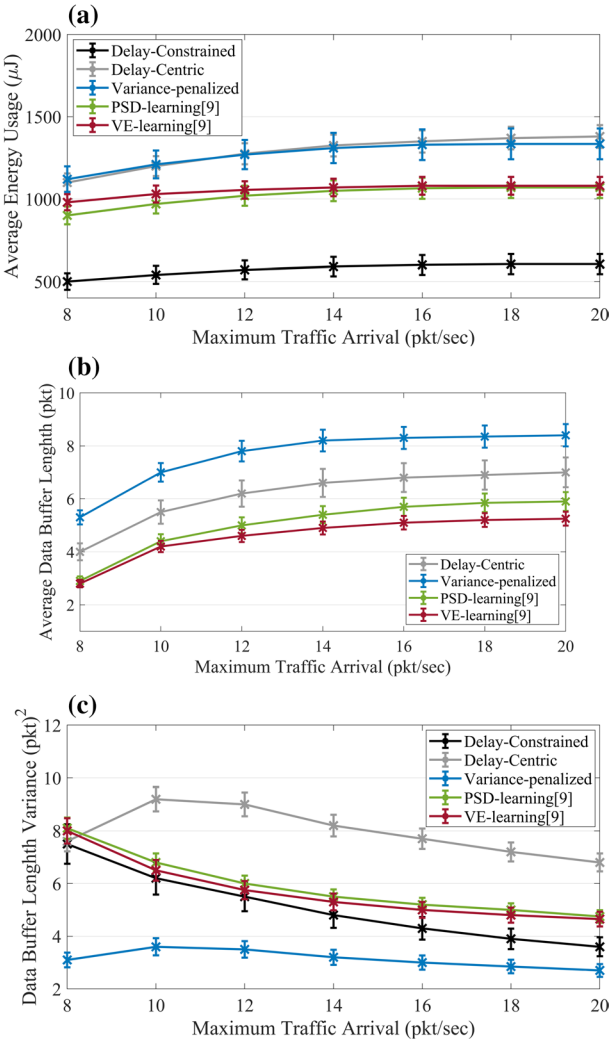

Fig. 8 **a** Impact of the traffic data arrival intensity on average energy usage. **b** Impact of the data arrival on average data buffer length. **c** Impact of the traffic data arrival on data buffer length variance

4.2.1 Impact of traffic arrival intensity

In Fig. 8a, as traffic data arrival level increases, the consumed energy packets by all schemes increase gradually. However, compared to the "Delay-Constrained" method, the consumed energy by all the other methods is much higher. This is due to the fact that in these methods, the primary objective is delay rather than minimizing energy usage.

Figure 8b illustrates how the average data buffer length varies as the level of traffic data arrival rises. In this experiment, we have left out the

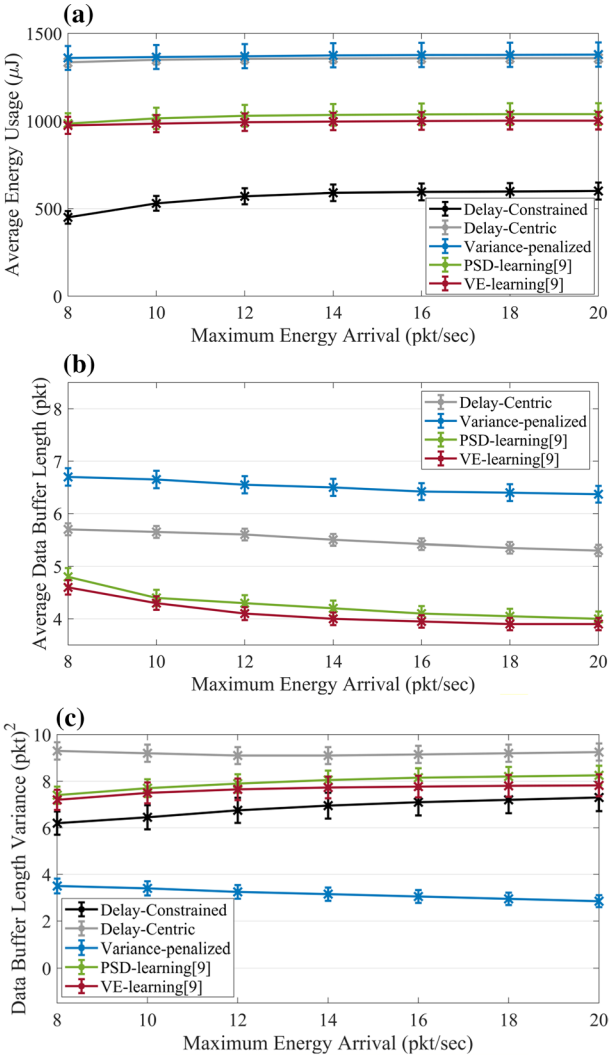

Fig. 9 **a** Impact of the energy arrival level on average energy usage. **b** Impact of the energy arrival level on average data buffer length. **c** Impact of the energy arrival level on data buffer length variance

"Delay-Constrained" method as it must always guarantee a certain buffer length. As shown in Fig. 8b, the "Delay-Centric" method has a smaller average buffer length than its "Variance-Penalized" counterpart, which is a direct result of defining the objective to minimizing data the mean buffer length. As before, both PDS- and VE-learning algorithms outperform our Q-learning-based schemes. This is because these two algorithms have smaller sample complexity and can learn the optimal policy much faster.

As shown in Fig. 8c, the data buffer length variance of the methods decreases almost linearly in response to the increases in traffic data arrival. Evidently, in our proposed "Variance-Penalized" method, we can see the least amount of data buffer length variance. All other methods begin with a large value for buffer length; then, as the traffic intensity increases, the buffer space gets exhausted (the buffer is almost always full), so the variance drops in the end.

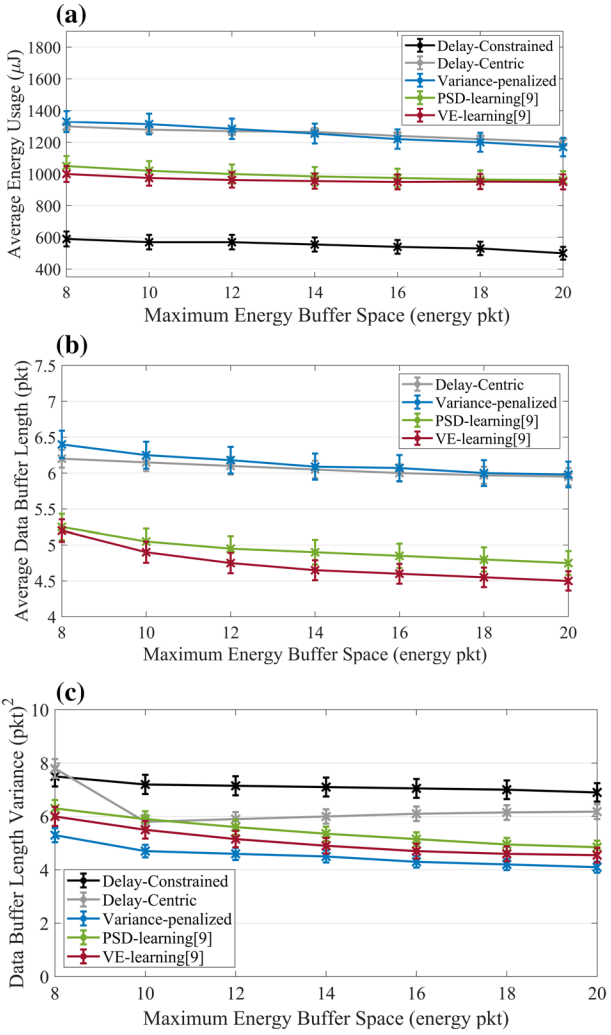

Fig. 10 **a** Impact of the maximum energy buffer space on average energy usage. **b** Impact of the maximum energy buffer space on average data buffer length. **c** Impact of the maximum energy buffer space on average data buffer length variance

4.2.2 Impact of the energy arrival intensity

According to Fig. 9a, as a result of increasing the energy arrival level, the energy usage by the "Delay-Constrained" scheme is the smallest. This is due to the fact that it only needs to satisfy a certain delay bound, while the other methods endeavor to minimize delay as much as possible, hence consuming much more energy in the process.

As can be seen in Fig. 9b, buffer length in all methods decreases almost linearly. The "Variance-Penalized" scheme has incurred the largest average buffer cost because unlike others its goal is not just minimizing the delay, but a combination of delay and delay variation. Moving on to Fig. 9c, it is seen that as the level of energy arrival rises, all methods (except the "Variance-Penalized" scheme) suffer from large buffer fluctuations. To conclude, our "Variance-Penalized" method shows the lowest data buffer length variance, which results in the smallest delay variation at the receiver.

4.2.3 Effect of maximum energy buffer space

Figure 10a displays average energy usage by all the methods in response to increases in maximum energy buffer space. Similar to the previous plots, average energy usage by the "Delay-Constrained" method is the lowest. It is worth noting that in all cases, as the IoT equipment harvests more energy from the environment, the IoT node is better able to exploit the channel conditions, and as a result, energy usage follows a moderate downward slope. Figure 10b shows that as energy buffer space increases, the average data buffer length for all the schemes decreases. In particular, similarly to the case of increasing the harvesting power, by also increasing the capacity for storing more harvested energy from the environment, the IoT node can better exploit the channel conditions. In other terms, t opportunistically sends a larger number of packets when the channel is good (thanks to the higher available energy), practically emptying its data buffer. Conversely, under poor channel conditions, it rarely transmits, or does not transmit at all. However, as it is noticeable from the plot in Fig. 9b, the reduction in energy usage occurs with a steeper slope when the harvesting power increases. Finally, Fig. 10c illustrates the data buffer length variance for all the methods as we vary the energy buffer space. As can be seen, the "Variance-Penalized" method has the lowest data buffer length variance.

5 Conclusion

In this paper, we have presented three Q-learning-based algorithms for optimal transmission control in energy harvesting IoT equipment, namely Delay-Constrained, Delay-Centric, and Variance-Penalized methods. All three methods operate in the absence of the statistical knowledge of random processes in the environment. The use-case for Delay-Constrained method is in applications where a certain delay

bound should be met by the underlying system. The Delay-Centric method minimizes the data buffer delay subject only to the energy availability constraint. Finally, the Variance-Penalized method is able to strike a trade-off between the achievable mean data buffer length and its variation. This is an important transmission characteristic because data queue fluctuations can increase buffer overflow (i.e., packet loss), buffer underflow (i.e., wastage of the available channel bandwidth), and jitter or delay variance. The simulation experiments show the efficacy of the proposed methods in terms of convergence properties, and in handling high traffic load as well as various energy arrival regimes. As future work, we intend to extend the proposed methods to a multiple access setting where multiple transmitting IoT equipment competes with each other in terms of gaining access to the spectrum.

Funding No funding was received to assist with the preparation of this manuscript.

Declarations

Conflict of interest All authors declare that they have no conflict of interest that are relevant to the content of this article.

Data availability Data sharing is not applicable—no new data generated.

References

1. Lee D, Lee H (2018) IoT service classification and clustering for integration of IoT service platforms. J Supercomput 74:6859–6875
2. He Y, Cheng X, Peng W, Stuber GL (2015) A survey of energy harvesting communications: models and offline optimal policies. IEEE Commun Mag 53(6):79–85. https://doi.org/10.1109/MCOM.2015.7120021
3. Yang J, Ulukus S (2012) Optimal packet scheduling in an energy harvesting communication system. IEEE Trans Commun 60:220–230
4. Sah DK, Amgoth T (2020) A novel efficient clustering protocol for energy harvesting in wireless sensor networks. Wireless Netw 26:4723–4737
5. Shaviv D, Zgur AO (2016) Universally near optimal online power control for energy harvesting nodes. IEEE J Sel Areas Commun 34:3620–3631
6. Arafa A, Baknina A, Ulukus S (2018) Online fixed fraction policies in energy harvesting communication systems. IEEE Trans Wireless Commun 17:2975–2986
7. Aprem A, Murthy CR, Mehta NB (2013) Transmit power control policies for energy harvesting sensors with retransmissions. IEEE J Sel Topics Signal Process 7(5):895–906
8. Neely M (2010) Stochastic network optimization with application to communication and queuing systems. Morgan and Claypool
9. Sharma N, Mastronarde N, Chakareski J (2020) Accelerated structure-aware reinforcement learning for delay-sensitive energy harvesting wireless sensors. IEEE Trans Signal Process 68:1409–1424
10. Toorchi N, Chakareski J, and Mastronarde N. (2016) Fast and low- complexity reinforcement learning for delay-sensitive energy harvesting wireless visual sensing systems. IEEE International Conference on Image Processing (ICIP), 1804–1808.
11. Shahhosseini S, Seo D, Kanduri A, Hu T, Lim S, Donyanavard B, Rahmani AM, Dutt N (2022) Online learning for orchestration of inference in multi-user end-edge-cloud networks. ACM Trans Embed Comput Syst. https://doi.org/10.1145/3520129

12. Aslani R, Hakami V, Dehghan M (2018) A token-based incentive mechanism for video streaming applications in peer- to-peer networks. Multim Tools Appl 77:14625–14653
13. Wang C, Li J, Yang Y, Ye F (2018) Combining solar energy harvesting with wireless charging for hybrid wireless sensor networks. IEEE Trans Mob Comput 17:560–576
14. Malekijoo A, Fadaeieslam MJ, Malekijou H, Homayounfar M, Alizadeh-Shabdiz F, Rawassizadeh R, (2021), FEDZIP: A Compression Framework for Communication-Efficient Federated Learning. https://doi.org/10.48550/arXiv.2102.01593
15. Prabuchandran KJ, Meena SK, Bhatnagar S (2013) Q-learning based energy management policies for a single sensor node with finite buffer. IEEE Wireless Commun Lett 2:82–85
16. Kansal A, Jason H, Zahedi S, Srivastava M (2007) Power management in energy harvesting sensor networks. ACM Trans Embedd Comput Syst 6:32–44
17. Mastronarde N, Modares J, Wu C, and Chakareski J. (2016) Reinforcement learning for energy-efficient delay-sensitive csma/ca scheduling. IEEE Global Communications Conference (GLOBECOM), 1–7.
18. Hakami V, Mostafavi SA, Javan NT, Rashidi Z (2020) An optimal policy for joint compression and transmission control in delay-constrained energy harvesting IoT devices. Comput Commun 160:554–566. https://doi.org/10.1016/j.comcom.2020.07.005
19. Masadeh A, Wang Z, and Kamal AE. (2018) Reinforcement learning exploration algorithms for energy harvesting communications systems. IEEE International Conference on Communications (ICC), 1–6.
20. Hu S, Chen W (2021) Joint lossy compression and power allocation in low latency wireless communications for IIoT: a cross-layer approach. IEEE Trans Commun 69(8):5106–5120. https://doi.org/10.1109/TCOMM.2021.3077948
21. Namjoonia F, Sheikhi M, Hakami V (2022) Fast reinforcement learning algorithms for joint adaptive source coding and transmission control in IoT devices with renewable energy storage. Neural Comput Appl 34:3959–3979. https://doi.org/10.1007/s00521-021-06656-6
22. Wenwei LU, Siliang G, Yihua Z (2021) Timely data delivery for energy-harvesting IoT devices. Chin J Electron 31(2):322–336
23. Lei J, Yates R, Greenstein L (2009) A generic model for optimizing single-hop transmission policy of replenishable sensors. IEEE Trans Wireless Commun 8:547–551
24. Blasco P, Gunduz D, Dohler M (2013) A learning theoretic approach to energy harvesting communication system optimization. IEEE Trans Wireless Commun 12:1872–1882
25. Putterman M. (2014) Markov decision processes.:discrete stochastic dynamic programming.
26. Xiao Y, Niu L, Ding Y, Liu S, Fan Y (2020) Reinforcement learning based energy-efficient internet-of-things video transmission. Intell Converg Netw 3:258–270. https://doi.org/10.23919/ICN.2020.0021
27. Sutton R, Barto AG (2018) Reinforcement learning: an introduction. MIT Press
28. Prakash G, Krishnamoorthy R, Kalaivaani PT (2020) Resource key distribution and allocation based on sensor vehicle nodes for energy harvesting in vehicular ad hoc networks for transport application. J Supercomput 76:5996–6009
29. Chu M, Li H, Liao X, Cui S (2019) Reinforcement learning-based multiaccess control and battery prediction with energy harvesting in IOT systems. IEEE Internet Things J 6:2009–2020
30. Teimourian H, Teimourian A, Dimililer K et al (2021) The potential of wind energy via an intelligent IoT-oriented assessment. J Supercomput. https://doi.org/10.1007/s11227-021-04085-9
31. Berry RA, Gallager RG (2002) Communication over fading channels with delay constraints". IEEE Trans Inf Theory 48(5):1135–1149
32. Altman E (1999) Constrained Markov decision processes. Routledge
33. Gosavi A (2014) "Variance-penalized markov decision processes: dynamic programming and reinforcement learning techniques. Int J Gener Syst 43:871
34. Bertsekas D (1999) Nonlinear programming. Athena Scientific
35. Borkar V, Konda V (1997) The actor-critic algorithm as multi-time-scale stochastic approximation. Sadhana 22:525–543
36. Wang H, Mandayam NB (2004) A simple packet-transmission scheme for wireless data over fading channels. IEEE Trans Commun 52:1055–1059
37. Altman E, Asingleutility I (1999) Constrained markov decision processes. Routledge
38. Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley
39. Little JDC (1961) A proof for the queuing formula: L = (lambda) w. Oper Res 9:383–387

40. Sharma AB, Golubchik L, Govindan R, Neely MJ (2009) Dynamic data compression in multi-hop wireless networks. Sigmetrics Perform Eval Rev 37:145–156
41. Mitchell TM (1997) Machine Learning, 1st edn. McGraw-Hill Inc.
42. Gosavi A (2015) Simulation-based optimization parametric optimization techniques and reinforcement learning. Springer
43. Sakulkar P, Krishnamachari B (2018) Online learning schemes for power allocation in energy harvesting communications. IEEE Trans Inf Theory 64:4610–4628
44. Zordan D, Melodia T, Rossi M (2016) On the design of temporal compression strategies for energy harvesting sensor networks. IEEE Trans Wireless Commun 15:1336–1352

Authors and Affiliations

Hanieh Malekijou[1] · Vesal Hakami[1] · Nastooh Taheri Javan[2] · Amirhossein Malekijoo[3]

Hanieh Malekijou
h_malekijou@cmps2.iust.ac.ir

Nastooh Taheri Javan
nastooh@eng.ikiu.ac.ir

Amirhossein Malekijoo
amirhossein.maleki1990@semnan.ac.ir

[1] School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

[2] Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran

[3] Department of Electrical and Computer Engineering, Semnan University, Semnan, Iran