Learning Optimal Edge Processing with Offloading and Energy Harvesting

Andrea Fox, Francesco De Pellegrini, Eitan Altman

▶ **To cite this version:**

Andrea Fox, Francesco De Pellegrini, Eitan Altman. Learning Optimal Edge Processing with Offloading and Energy Harvesting. MSWiM '23: Int'l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, Oct 2023, Montreal Quebec Canada, Canada. pp.83-92, 10.1145/3616388.3617516 . hal-04022507v2

HAL Id: hal-04022507
https://hal.science/hal-04022507v2

Submitted on 23 Mar 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Learning Optimal Edge Processing with Offloading and Energy Harvesting

Andrea Fox*, Francesco De Pellegrini* and Eitan Altman*†

ABSTRACT

Modern portable devices can execute increasingly sophisticated AI models on sensed data. The complexity of such processing tasks is data-dependent and has relevant energy cost. This work develops an Age of Information markovian model for a system where multiple battery-operated devices perform data processing and energy harvesting in parallel. Part of their computational burden is offloaded to an edge server which polls devices at given rate. The structural properties of an optimal policy for a single device-server system are derived. They permit to define a new model-free reinforcement learning method specialized for monotone policies, namely Ordered Q-Learning, providing a fast procedure to learn the optimal policy. The method is oblivious to the devices' battery capacities, the cost and the value of data batch processing and to the dynamics of the energy harvesting process. Finally, the polling strategy of the server is optimized by combining this policy improvement technique with stochastic approximation methods. Extensive numerical results provide insight into the system properties and demonstrate that the proposed learning algorithms outperform existing baselines.

KEYWORDS

AoI, Energy-Harvesting, Offloading, MDP, Reinforcement Learning

ACM Reference Format:

Andrea Fox*, Francesco De Pellegrini* and Eitan Altman*†. 2023. Learning Optimal Edge Processing with Offloading and Energy Harvesting. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

New generations of mobile access networks promise low delay and high-speed throughput data connections paired with in-network processing capabilities [1]. They will support mobile computing services able to integrate AI-intensive processing tasks in their workloads such as, e.g., smart city services or virtual and augmented reality applications. In the next future, the vast majority of enterprise IoT projects are expected to have an AI component, up from less than 10% in 2018 [2]. To support these applications, edge networks must evolve to solve the key challenges of such scenario [3]. In fact, the energy consumption of AI applications is critical for battery operated devices. On the other hand, they may require periodic updates based on fresh data, settling specific requirements on the rate at which data are fetched and processed. The standard metric to this respect is the Age of Information (AoI), denoting the freshness of information when received at the destination [4]. In the literature, AoI performance has been studied for several queuing disciplines determining the average system time of data reads. The standard objective is to maximize some long term reward for the AoI of retrieved information [5, 6]. In the context of this paper, it is the long term reward for processing data, also called age of processing [7].

Energy harvesting from renewable sources, such as solar cells or piezoelectric generators, has becoming available on devices used, e.g., for Internet of things (IoT) applications. In the literature, optimal policies for AoI minimization under energy harvesting have been studied using Markov decision theory both in continuous and in discrete time [8–12]. Most such studies focused on optimal policies to optimize AoI subject to energy causality constraints. The problem is to reserve the battery charge for moments when it is most needed, e.g., for alarm generation events [12].

The scenario studied in this paper considers the uncertainty of battery consumption and the uncertainty about the amount of energy harvested over time. In fact, the energy cost of AI tasks depends inherently on the statistical distribution of input data and on the preference for lightweight or heavyweight models. Thus, when such a computing task is launched on a batch of data, the energy required to finish processing may exceed the available battery charge. In this event, a further delay for data processing is needed in order to harvest enough energy to complete the ongoing task.

Finally, in edge computing, task offloading to edge-servers mitigates the problem of energy consumption to run AI tasks on mobile devices [7, 13–19]. Thus, a device can delegate computing tasks to edge servers. However, in a realistic scenario, offloading is constrained by the availability of the edge-server, which may be supporting multiple devices. Moreover, not always task offloading proves convenient, depending also on the delays it introduces.

Main contribution. This work develops a markovian modeling framework where multiple battery-operated devices process data in parallel and perform energy harvesting. Furthermore, an intermittent edge-server supports task offloading. Events of battery depletion may occur due to the unknown complexity of data processing and to the randomness of the harvesting process. Hence, an optimal stationary policy prescribes whether or not a device should process a new data batch based on AoI and battery status. By proving that such policy is monotone it is possible to design a lightweight reinforcement learning (RL) algorithm, namely Ordered Q-learning. It applies to the wide class of problems with monotone structure of the optimal value function w.r.t. the state-action partial order. In this setting, it outperforms standard baselines, including policy gradient methods.

*Laboratoire informatique d'Avignon (LIA), Avignon University, France,†INRIA, Sophia Antipolis, France. This work has been partially supported by the French National Research Agency (ANR) within the framework of the PARFAIT project (ANR-21-CE25-0013), see http://parfait.univ-avignon.fr.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2023 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: List of symbols used in the paper.

Symbol	Meaning
t	time step $t = 1, 2, \dots$
x_t	age of information (AoI) $x_t \in \{1, \dots, M\}$
e_t	buffer energy level $e_t \in \{0, 1, \dots, B\}$
z_t	server availability $z_t \in \{0, 1\}$
$s_t = (x_t, e_t, z_t)$	device state
γ	discount factor
$\mathcal{S}$	state space
$\mathcal{A} = \{0, 1\}$	action set, action set of state s, $\mathcal{A}(s) \subset \mathcal{A}$
$a_t = \{0, 1\}$	action taken at time t
B	battery capacity
M	saturation bound on AoI
C_t	batch processing cost at time t
H_t	harvested energy at time t
δ	offloading roundtrip time

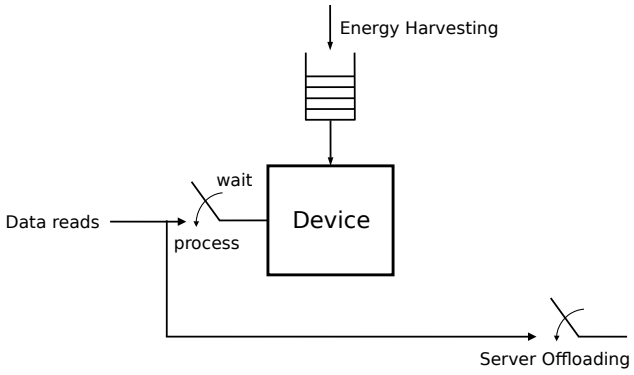

Figure 1: Device performing local processing of data batches with energy harvesting and intermittent edge-server offloading.

Finally, the polling probability vector of the server is optimized by stochastic learning. To the best of the authors' knowledge, this is the first work to study the problem of AoI minimization in a system where multiple battery operated devices perform energy harvesting and rely on an edge-server for task offloading, incurring possibly in battery depletion depending on data and battery status.

The paper is organized as follows. Sec. 2 describes the state of the art. In Sec. 3 the basic server-device Markov decision model is introduced. The monotone structure of an optimal policy is analyzed in Sec. 4. Sec. 5 describes RL algorithms converging the optimal solution. Sec. 6 extends analysis and algorithmic solutions to the case of multiple devices. Numerical results are provided in Sec. 7 and a concluding section ends the paper.

2 RELATED WORKS

The term AoI was first used in [4] to denote freshness of data received at the destination. Several later works on AoI focused on the minimization of such metric [5]. The AoI minimization has been studied in combination with energy harvesting to charge IoT devices [8–12]. The energy-aware scheduling policies of the type studied in this paper have been introduced to vary the sensing rate based on the instantaneous battery charge [10]. Conversely, the effect of error prone channels of the type studied in [8] and [9] is left for future works. In the proposed model, as in [10, 11], measurements are possible only when the battery level is sufficiently high. As in [12], the proposed model accounts for data-dependent energy consumption. Moreover, it considers in detail the effect of battery depletion events.

Several works combine AoI minimization and task offloading to external devices. Authors of [7], [17] and [18] studied the performance of a device-server system where a local processor device is supported by a remote server. As the present work, [13] and [14] study the trade-off between energy consumption and execution delays to offload applications to an edge server. In [15] and [16] deep reinforcement learning (DRL) is used to compute optimal offloading policies. As showed later, knowing the properties of the optimal value function permits much less expensive solutions. With very few exceptions, e.g., [19] and [6], AoI in multi-device systems of the type studied in this paper are not addressed in the literature. In [19] the average AoI is minimized by choosing whether to use the local processor or an edge server according to the network status. [6] provides lower bounds on the average AoI performance achievable for several queuing disciplines on a single-hop network as the one studied in this work. Note that the metric addressed in this work is not the average AoI but rather the peak AoI [20].

The first Markovian model for the control of AoI appeared in [21]. MDP models were later used in [12, 16, 19, 22]. Works [7, 8] use constrained MDPs, while [11, 23] use partially observable MDPs. Including constraints and partial observations are interesting extensions but out of the scope of the present work. Threshold policies for the control of the AoI have been obtained before [7, 9, 11]. In [9] the threshold is one-dimensional while the policy structure described in [7] is only characterized numerically. In [11] the threshold regards two incorrelated control variables: in the proposed model, state transitions for AoI and energy both depend on the chosen action.

3 SYSTEM MODEL

The device-server scheme is represented in Fig. 1: the device can read data batches and process them, while the edge server can poll the device to offload the computation. Thus, data batches are either read and processed locally on the device or read and offloaded. Time steps are discrete with index $t = 1, 2, \dots$. Processing data at time t on the device requires C_t energy units, which form a sequence of i.i.d. random variables $\{C_t\}$ with probability distribution $p_C(c) = \mathbb{P}\{C = c\}$. When the device is polled by the edge server, the data processing task is offloaded and it has zero energy cost for the device. However, sending and retrieving processed data from the server incurs in a constant delay of $\delta \geq 0$ time units. The device has a battery of capacity $B > 0$ energy units and it can harvest a number of energy units H_t per timestep t. The energy fetched per timestep, namely the *harvesting rate*, forms a sequence of i.i.d. random variables $\{H_t\}$ whose corresponding probability distribution is $p_H(h) = \mathbb{P}\{H = h\}$.

Let introduce the Semi Markov Decision Process (SMDP) which models the system. The state of the device at time t is denoted as $s_t = (x_t, e_t, z_t)$, where x_t represents the AoI, e_t is the device battery level and z_t is the server state. Binary variable z_t indicates whether the server has polled the device for offloading, $z_t = 1$, or not, $z_t = 0$.

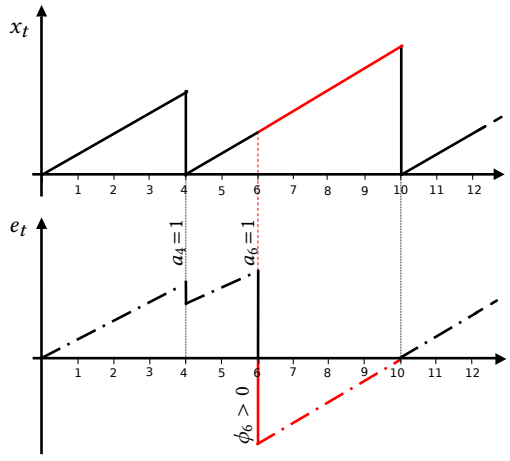

Figure 2: A sample path of the process $s_t = (x_t, e_t, 0)$ for $H \equiv 1$: in red the vacation periods where the battery is empty. The duration of timeslot starting at time $t = 6$ is 4 timesteps.

The corresponding per slot transition probability is $p_Z(z'|z)$, e.g., $p_Z(1|0)$ is the probability for the server to poll the device at $t+1$ given that it did not at t. Note that x_t is the AoI of the last data batch processed by a tagged device and the AoI is measured at the end of each time slot. It holds $x_t = 1$ when the device has just processed fresh data. Afterwards, the AoI increases of one timestep at every time slot t until either the device fetches a new data batch and performs the computation or the server polls the device and offloading occurs. In both cases, at the end of the data processing the AoI is reset to 1.

Freshness function $u(x)$ is the utility for processing a data batch x time units after the last one was processed; $u(\cdot)$ is bounded and non-increasing. It is assumed that there exists $M > 0$ such that $u(M+k) = u(M)$ for all $k > 0$, so that the AoI takes values in $\{1, \dots, M\}$. The state space is denoted $\mathcal{S} = \{1, \dots, M\} \times \{0, \dots, B\} \times \{0, 1\}$.

Finally, it is possible that the battery available at timeslot t is not sufficient to terminate the computation immediately, namely, $\phi_t := c_t - (h_t + e_t) > 0$. In this case, a delay is incurred in order to harvest a sufficient amount of energy and complete the processing of the current batch. The corresponding timeslot has a random duration, due to the stochastic nature of energy harvesting.

The action set is $\mathcal{A} = \{0, 1\}$, where 0 means *wait* and 1 *process*. The action taken at time t is denoted a_t. If $a_t = 1$, the device fetches a new data batch, which is processed at energy cost $c_t > 0$. For every state s_t where $z_t = 0$ the action set available at each device is $\mathcal{A}(s_t) = \{0, 1\}$, i.e. it can process or not the information. On the other hand, when $z_t = 1$ the action set is $\mathcal{A}(s_t) = \{1\}$: thus if the device is polled by the server, by default the data unit is processed. Furthermore, the energy to transmit data to the server is assumed to be negligible compared to local data processing. Finally, the dynamics of the AoI for data batches is

$$x_{t+1} = \begin{cases} 1 & \text{if } a_t = 1 \\ [x_t + 1]_M & \text{if } a_t = 0 \end{cases}$$

where $[y]_A := \max\{0, \min\{y, A\}\}$. The AoI at the renewal instants is also called *peak AoI* [20] in the literature: it corresponds to the processing rate.

It now possible to characterize the transition probabilities for a given action $a \in \mathcal{A}(s)$. Hereafter H is considered to be deterministic with $H \equiv 1$ and the time index is omitted for notation's sake. The general case for stochastic harvesting rate is derived in (22) and (23).

Let $s = (x, e, z)$ be the current state of the device and $s' = (x', e', z')$ the following state under action a.

The transition probability from s to s' is written, for $a = 0$, as

$$p(s'|s, 0) = \begin{cases} p_Z(z' \mid 0) & s = (x, e, 0), \\ & s' = ([x+1]_M, [e+1]_B, z'), \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Instead, when $a = 1$, the transition probability from s to s' is

$$p(s' \mid s, 1) = \begin{cases} p_Z(z' \mid 0) p_C(e + 1 - e') & s = (x, e, 0), \\ & s' = (1, e', z'), \\ & 0 < e' < B \\ p_Z(z \mid 0) \sum_{c=e+1}^{\infty} p_C(c) & s = (x, e, 0), \\ & s' = (1, 0, z') \\ p_Z(z \mid 0) \sum_{c=1}^{e+1-B} p_C(c) & s = (x, e, 0), \\ & s' = (1, B, z') \\ p_Z(z' \mid 1) & s = (x, e, 1), \\ & s' = (1, [e+1]_B, z'), \\ & e + h < B \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The sojourn times $\tau(s, s')$ of the SMDP are associated to the transition from an origin state $s = (x, e, z)$ to a destination $s' = (x', e', z')$. They are unitary except possibly those corresponding to the transition to a renewal state, i.e., where $x' = 1$. For that transition, if $z = 1$ the sojourn time is $\tau(s, s') = 1 + \delta$, due to the round-trip delay for the edge-server offloading. If $z = 0$ it is $\tau(s, s') = 1$ when $\phi = C - e - H \le 0$. Otherwise, when $C - e - H > 0$ it writes

$$\tau(s, s') = \min \left\{ k \mid \sum_{r=1}^{k} H_r > C - e - H \right\} \tag{3}$$

that is the number of recharges required in order to complete the computation. Note that during such sojourn time, the device cannot be interrupted until the end of processing. In the case of deterministic energy harvesting with $H \equiv 1$, it holds $\tau(s, s') = (1 - z) \max\{1, C - e - H\} + z(1 + \delta)$.

In order to simplify the analysis, let consider the equivalent MDP obtained by sampling the decision process at the time when actions are taken. Its reward under the state action pair (s_t, a_t) at time t is proportional to the freshness of information at the end of process, as well as on the initial level of energy and the server availability:

$$r_{t+1}(s_t, a_t) = \begin{cases} u(x_t) - d(c_t - e_t - h_t) \cdot a_t & \text{if } z_t = 0 \\ u([x_t + \delta]_M) & \text{if } z_t = 1 \end{cases} \tag{4}$$

Function $d : \mathbb{R} \to \mathbb{R}^+$ is a penalty that depends on the amount of energy units required to complete the local processing if $a = 1$ is selected. It is further assumed that $d(x) = 0$ when $x \le 0$ and that d is a bounded non-decreasing function.

3.1 Discounted Model

The processing policy π for a tagged device is a probability distribution over the state-action space. The value function for policy π is $v_\pi(s) := \mathrm{E}_\pi[G_t|s_t = s]$ where $G_t := \sum_{t=0}^{\infty} \gamma^k r_{t+1+k}(s_t, \pi(s_t))$; the Q-function $q_\pi(s,a) := \mathrm{E}_\pi[G_t|s_t = s, a_t = a]$ is used later in Section 5.2. In the discounted model presented next, the objective is to maximize $v_\pi(s)$ in the set of stationary policies, where $\pi = \pi(s)$ is the probability that the device performs action $a = 1$ in state s. As showed in Sec. 4.2, the corresponding ergodic state occupancy probability permits to calculate the sampling rate based on the average peak AoI. The optimal value function $v_*(s)$ solves the the Bellman optimality equation associated to the discounted reward problem

$$v_*(x,e,0) = \max\Bigg\{ u(x) - \sum_{c=1}^{\infty} p_C(c) \sum_{h=1}^{\infty} p_H(h) d(e+h-c) + \\ + \gamma \sum_{c=1}^{\infty} p_C(c) \sum_{h=1}^{\infty} p_H(h) \mathbb{E}_Z[v_*(1, [e+h-c]_B, z)], \\ u(x) + \gamma \sum_{h=1}^{\infty} p_H(h) \mathbb{E}_Z[v_*([x+1]_M, [e+h]_B, z)] \Bigg\}$$

in the case $z = 0$. When $z = 1$ the action set is a singleton, then

$$v_*(x,e,1) = u([x+\delta]_M) + \sum_{h=1}^{\infty} p_H(h) \mathbb{E}_Z[v_*(1, [e+h]_B, z)] \quad (5)$$

Structural properties. The optimal policy and the corresponding value functions have several properties that are exploited in the next section. First, from (4) it is immediate that, if the harvesting process $\{H'_t\}$ is stochastically larger than $\{H_t\}$, also $v'(s) \geq v(s)$ for all $s \in \mathcal{S}$; similar observations hold w.r.t. process $\{C_t\}$.

Before proving the next result, a few definitions are introduced. First, a policy is monotone if the action is either increasing or decreasing in the state, given an assigned partial order on the state space $\mathcal{S}$ and on the action space A. In particular, the following partial order is defined on the state space

$$\begin{aligned} (x, e+1, z) &\geq (x, e, z') \\ (x-1, e, z) &\geq (x, e, z') \end{aligned} \quad (6)$$

whereas the obvious order is imposed on the action space $A = \{0, 1\}$. Let s^+ be *larger* than s^- if $s^+ \geq s^-$. The real function $\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is *supermodular* if $\psi(x^+, y^+) + \psi(x^-, y^-) \geq \psi(x^+, y^-) + \psi(x^-, y^+)$, also equivalent to say that $\psi(x^+, y) - \psi(x^-, y)$ is decreasing in y. The *tail transition probability* $w(s'|s,a) := \sum_{\zeta \geq s'} p(\zeta|s,a)$ defines the probability of making a transition to states larger than s taking action a.

LEMMA 1. *i. $v_*(x,e,z)$ is non increasing in x*
ii. $v_*(x,e,z)$ is non decreasing in e.

PROOF. The proof is made by verifying that the system model adheres to the following three assumptions [24].
i. Non decreasing rewards. Let consider $s^+ = (x,e,z)$, $s^- = (x+1,e,z)$ and a fixed value of c. For $a = 1$, $r(s^+,1) = u(x) - d(c-e-h) \geq u(x+1) - d(c-e-h) = r(s^-,1)$, as u is nonincreasing. For $a = 0$ the same property holds since $r(s^+,0) = u(x) \geq u(x+1) = r(s^-,0)$. Now assume $s^+ = (x,e+1,z)$ and $s^- = (x,e,z)$. For $a = 0$ the reward is equal for both states. For $a = 1$, $r(s^+,1) = u(x) - d(c-e-1-h) \geq u(x) - d(c-e-h) = r(s^-,1)$ since d is by assumption nondecreasing.
ii. Supermodular rewards. This property can be easily verified for both cases considered before;
iii. Increasing tail transition probability. First, let consider the case $s^+ = (x,e,z)$ and $s^- = (x+1,e,z)$. When $a = 1$,

$$w(s'|s^+,1) = \sum_{\zeta \geq s'} p(\zeta|s^+,1) = \sum_{\zeta \geq s'} p(\zeta|s^-,1) = w(s'|s^-,1)$$

as the final value of the AoI is always 1 after action $a = 1$, while the final energy level only depends on the initial energy e. If $a = 0$, then $w(s'|s^+,0) = \mathbb{1}\{(x+1,e+h,z) \geq s'\} \geq \mathbb{1}\{(x+2,e+h,z) \geq s'\} = w(s'|s^-,0)$. The inequality is induced by the partial order of states. Now, consider $s^+ = (x,e+1,z)$ and $s^- = (x,e,z)$. When $a = 1$, the inequality is verified because the final energy obtained with a transition that starts in s^+ is larger or equal than the one obtained when starting in s^- w.p.1. Conversely, when $a = 0$, $w(s' \mid s^+,0) = \mathbb{1}\{(x+1,e+h+1,z) \geq s'\} \geq \mathbb{1}\{(x+1,e+h,z) \geq s'\} = w(s' \mid s^-,0)$ and the inequality follows from (6). □

From Lemma 1 it follows easily a property of the optimal Q-function $q_*(s,a)$ which will prove useful in order to develop suitable learning algorithms.

COROLLARY 1. *Fixed $a \in \{0,1\}$, then for $s = (x,e,z) \in \mathcal{S}$*
i. $q_*((x,e,z),a) \leq q_*((x,[e+1]_B,z),a)$
ii. $q_*(([x+1]_M,e,z),a) \leq q_*((x,e,z),a)$

In all numerical tests, function Q appears indeed superadditive in $\mathcal{S} \times \mathcal{A}$, a property that would be sufficient to prove the monotonicity of the optimal policy. Unfortunately, the classic assumption on the supermodularity of the tail transition probability (see Prop. 4.7.3 in [24]) which grants the existence of a monotone policy can be easily disproved.

THEOREM 1 (OPTIMAL POLICY STRUCTURE). *A stationary deterministic policy π is optimal if and only if it is monotone. In particular, there exist thresholds $T(e)$, $0 \leq T(e) \leq M$ such that :*
i. $\pi(e,x,0) = 1$ if and only if $x \geq T(e)$;
ii. $T(e) \geq T(e+1) + 1$;
iii. if $\gamma = 0$, then $T(e) \geq T(e+1)$

PROOF. i. At first it is showed the existence of a threshold $T(e)$ for every level of energy e. Consider the function $\Delta q_*(x,e,z) = q_*((x,e,z),1) - q_*((x,e,z),0)$: it holds

$$\Delta q_*(x,e,0) = -\mathbb{E}_C[d(e+h-c)] + \gamma \Big\{ \sum_{c=1}^{\infty} p_C(c) \mathbb{E}_Z[v_*(1,[e+h-c]_B,z)] \\ - \mathbb{E}_Z[v_*([x+1]_M,[e+h]_B,z)] \Big\} \quad (7)$$

The increment w.r.t. x is nonnegative, due to Lemma 1:

$$\Delta q_*(x+1,e,0) - \Delta q_*(x,e,0) = -\gamma \Big(\mathbb{E}_Z[v_*([x+2]_M,[e+h]_B,z] - \\ - \mathbb{E}_Z[v_*([x+1]_M,[e+h]_B,z)] \Big)$$

ii. Observe that all states $([T(e)+k]_M,[e+1]_B,z)$ are transient for any integer $k > 1$. Hence, one could replace the policy for those states with no change in the state occupancy probability.
iii. Follows from the structure of the instantaneous reward. □

From Thm. 1 it follows that an optimal policy is monotone with a threshold structure.

4 OPTIMAL POLICY PERFORMANCE

An optimal policy π^* can be obtained using standard dynamic programming methods, under the assumption of having full information on the transition probabilities; in the following p_{π^*} is the stationary distribution given by π^*. Efficient variants of value iteration or policy iteration algorithms for monotone policies do exist. Alternatively, the optimal solution can be found solving a suitable linear program [24]. This section describes first how to evaluate the average peak AoI of the optimal policy, which renders the actual sampling rate of a device. It further introduces some properties of the discounted reward for a given distribution over the initial state for a policy π.

4.1 Average Peak AoI

Once determined an optimal policy, the corresponding average peak AoI can be computed by considering the renewal process $\{N(t), t \geq 0\}$ defined by the instants of visit to a state with AoI $x = 1$ for the corresponding Markov reward process. Let Y_r, $r = 1, 2, \ldots$ be the random variable describing the length of the r-th renewal and the corresponding renewal process [25]. Denote as $\Delta N = \mathbb{E}[N(t+1)] - \mathbb{E}[N(t)]$ the average number of renewal events, i.e., of processing events, per time unit. From Blackwell theorem [26], $\frac{\Delta N}{t} \to \frac{1}{\mathbb{E}[Y]}$. Let denote L the number of transitions during renewal period Y: it is the mean return time to a state with $x = 1$, i.e., the inverse of the stationary probability of being in a state s where $\pi^*(s) = 1$:

$$L = \frac{1}{\sum_{s:\pi^*(s)=1} p_{\pi^*}(s)} \tag{8}$$

By recalling the form of the sojourn times $\tau(s, s')$ of the SMDP, the expected value of the length of the renewal cycle Y is computed as

$$\mathbb{E}[Y] = \sum_{\{s':x'=1\}} \mathbb{E}_{\substack{s\sim p_{\pi^*} \\ \pi^*(s)=1}} \left[\tau(s, s')\right] p_{\pi^*}(s') + \frac{1}{L} - 1$$

obtained using a partition of the possible renewal states. For notation's sake, let

$$\chi(e') := \mathbb{E}_{\substack{s\sim p_{\pi^*}: \\ \pi^*(s)=1, z'}} \left[\tau(s, (1, e', z'))\right]$$

the expected sojourn time when the transition ends in a state $s' = (1, e,', z')$. Computing $\chi(e')$ accounts for the possible sample paths to renew to an energy state e'. Two possible cases are to be considered. If $z = 1$, then $e + h = e'$ with $p_H(\Delta e)$ where $\Delta e = e' - e$. Else, if $z = 0$, the possibility that $\phi > 0$ requires to account for the recharges needed to get to the final level of energy e', which depends on the distribution of both H and C.

$$\begin{aligned}\chi(e') = & p_Z(1)(1+\delta) \sum_{e=0}^{B} p_E(e) p_H(\Delta e) + \\ & + p_Z(0) \left(\sum_{e=0}^{B} p_E(e) \sum_{r=1}^{\infty} \Gamma_{1,r} p_C(h_1 - \Delta e) \right. \\ & \left. + \sum_{k=2}^{\infty} k \sum_{e=0}^{B} p_E(e) \sum_{r=1}^{\infty} \Gamma_{k-1,r} \sum_{c=e+\sum_{j=1}^{k-1} h_{j+1}}^{\infty} p_C(c) p_H\left(\Delta e + c - \sum_{j=1}^{k-1} H_j\right) \right)\end{aligned} \tag{9}$$

where $p_E(e)$ indicates the probability that action $a = 1$ is done when the energy level is equal to e (can be obtained from the stationary distribution under a given policy) and $\Gamma_{k,r}$ is the probability to harvest r energy units in k steps, namely $\Gamma_{k,r} = \mathbb{P}\left(\sum_{j=1}^{k} H_j = r\right)$. Note how it has been assumed that $\forall z_1, z_2\ \mathbb{P}(z_1 \mid z_2) = p_Z(z_1 \mid z_2) = p_Z(z_1)$. For the sake of notation, $p_Z(0) = 1 - p_Z$ and $p_Z(1) = p_Z$, with $p_Z \in [0, 1]$. The same will be assumed throughout the remainder of the work. Without loss of generality, let $k \leq B$: $\Gamma_{k,r}$ is defined by the following iteration

$$\Gamma_{k,r} = \begin{cases} \sum_{j=1}^{B} \Gamma_{k-1,r-j} p_H(j) & k \leq r \leq B \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\Gamma_{1,r} = p_H(r)\mathbb{1}\{1 \leq r \leq B\}$. Finally, the expected value of X is

$$\mathbb{E}[X] = \sum_{z'=0}^{1} \sum_{e'=0}^{B} \frac{p_{\pi^*}(1, e', z')}{\sum_{j=0}^{B} p_{\pi^*}(1, j, z')} \chi(e') + \frac{1}{L} - 1$$

In Section 7 numerical results will explore extensively how the model parameters influence the peak AoI.

4.2 Reward of a policy

The learning procedure for the multi-device model presented later in Sec.6 pivots on the dependence of the discounted reward $R_{\gamma,\pi}$ of a tagged policy π on the server polling probability p_Z. The performance measure for a policy used by a device is the discounted reward given a certain initial state distribution. In particular, sample paths initiate on renewal states with AoI $x = 1$, energy e chosen uniformly at random and expected server availability p_Z.

Fixed π, let P_π be the transition matrix of the corresponding Markov Chain and let vector r_π be the instantaneous reward attained at each state by following the action indicated by the policy. Let further define vector d_0 describing the initial distribution of states. The discounted reward of a device can be computed as

$$R_{\gamma,\pi} = d_0 \left(I + \gamma P_\pi + \gamma^2 P_\pi^2 + \ldots\right) r_\pi \tag{11}$$

Basic facts of MDP theory imply that the geometric series generated by γP_π is always converging so that

$$R_{\gamma,\pi} = d_0 \left(I - \gamma P_\pi\right)^{-1} r_\pi \tag{12}$$

To analyze this quantity it is useful to explicitly express the parametric dependence $d_0 = d_0(p_Z)$ and $P_\pi = P_\pi(p_Z)$. In particular,

$$d_0(p_Z) = \begin{cases} \frac{1}{B+1} p_Z & x = 1, z = 0 \\ \frac{1}{B+1}(1 - p_Z) & x = 1, z = 1 \\ 0 & \text{otherwise} \end{cases}$$

The bijective function $f : \{1, \ldots, M\} \times \{0, \ldots, B\} \to \{1, \ldots, M(B+1)\}$ maps each couple (x, e) to an integer value and two matrixes $P^0, P^1 \in \mathcal{M}^{(B+1)M \times (B+1)M}$ such that

$$\left(P^0\right)_{ij} = \mathbb{P}\left\{f(x', e') = j \mid f(x, e) = i, z = 0, a = \pi(x, e, 0)\right\}$$

and

$$\left(P^1\right)_{ij} = \mathbb{P}\left\{f(x', e') = j \mid f(x, e) = i, z = 1, a = 1\right\}$$

Both matrix will depend on the distribution of the random variables C and H, as well as on the fixed policy π. The explicit dependence of the transition matrix P_π on the server's polling probability writes

$$P_\pi(p_Z) = \begin{pmatrix} (1 - p_Z)P^0 & p_Z P^0 \\ (1 - p_Z)P^1 & p_Z P^1 \end{pmatrix}$$

which provides an explicit form to

$$R_{\gamma,\pi}(p_Z) = d_0(p_Z)\Big(I - \gamma P_\pi(p_Z)\Big)^{-1} r_\pi$$

Remark. For $\delta > 0$ the function $R_{\gamma,\pi}(p_Z)$ appears concave on the interval $[0, 1]$ for all tested stationary policies and parameters of the Markov decision process. The proof of the convergence of the algorithm introduced in Section 6 assumes this fact to be true. A formal proof is provided for a specific choice of parameters ($M = 2, B = 1$) and a particular policy in Section C. Additionally, $R_{\gamma,\pi}(p_Z)$ is indeed continuous and differentiable since each term of the matrix appearing in its explicit form has such property, as showed in C.

Finally, evaluating the optimal reward as a function of p_Z requires to consider different optimal policies and to determine

$$R_\gamma(p_Z) = \max_\pi R_{\gamma,\pi}(p_Z) \tag{13}$$

As it is a point-wise maximum, the continuity of $R_\gamma(\cdot)$ is true. However, even if the conjecture about the concavity of $R_{\gamma,\pi}$ is verified, $R_\gamma(p_Z)$ is not necessarily concave as it is the maximum of a set of concave functions.

5 LEARNING THE OPTIMAL POLICY

In a realistic setting, transition probabilities (1) and (2) may not be available. Furthermore, rewards (4) depend on the model used for the data-batch computation and also on data properties which may be only learned at runtime. A new model-free algorithm, namely *Ordered Q-learning* (OQL) is introduced next. It is a version of Q-learning (QL) [27] adapted for MDPs with monotone value functions under a given partial order imposed on the state space. As showed in Section 7, OQL outperforms other reinforcement learning algorithms such as policy gradient methods and yet retains the convergence guarantees of QL. Two variants of OQL based on the structure of the optimal policy are introduced next; they leverage two different partial orders on the states to find the optimal policy.

5.1 Ordered Q-learning

In the OQL the vector of the estimates of the Q-function are forced to comply to the monotonicity properties of the optimal Q-function. The update rule (14) for each state-action pair $(s, a) \in \mathcal{S} \times A$ hence writes

$$\begin{cases} \overline{Q}_{t+1}(s,a) = (1-\alpha_t)Q_t(s,a) + \alpha_t\Big(r_{t+1}(s,a) + \gamma \max_b Q_t(s',b)\Big) \\ Q_{t+1}(s',a) = \Pi_s\Big(\overline{Q}_{t+1}(s',a)\Big) \qquad \forall s' \in \mathcal{S} \end{cases} \tag{14}$$

where projection $\Pi_s(\cdot)$ at state s is a projection adapted to specific partial order imposed on the state space. Let $f : \mathcal{S} \to \mathbb{R}$: if $s' > s$, then $\Pi_s(f(s')) = \max\{f(s'), f(s)\}$ and if $s' \leq s$, then $\Pi_s(f(s')) = \min\{f(s'), f(s)\}$.

Thus, the basic Q-learning iteration is followed by a state-dependent projection on the set of allowed estimates. Alg. 1 reports on the complete algorithm for the version of the algorithm with a constant stepsize $\alpha_t = \alpha$. The convergence of the OQL can be ensured under the assumption that $\alpha_t = \alpha_t(s, a)$ appearing in (14) is a standard stepsize i.e., $\sum_{t=1}^{+\infty} \alpha_t(s,a) = +\infty$ w.p.1. and $\sum_{t=1}^{+\infty} \alpha_t(s,a)^2 < +\infty$ w.p.1. It holds the following result:

Algorithm 1 Ordered Q-learning

Require: step size $\alpha \in (0, 1]$, discount factor $\gamma \in [0, 1)$
1: $Q(s, a) \in \mathbb{R}, \forall s \in \mathcal{S}$
2: $Q(s', a) \leftarrow 0$ for s' terminal state
3: **for** each episode
4: initialize $s \in \mathcal{S}$
5: **for** each step of the episode
6: choose $a \in \mathcal{A}(s)$ given by Q (e.g., ϵ-greedy)
7: take action a and observe reward R and s'
8: $\overline{Q}(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \alpha\,[R + \gamma \max_a Q(s',a)]$
9: $Q(\tilde{s},a) \leftarrow \Pi_s(\overline{Q}(\tilde{s},a)), \quad \forall \tilde{s} \in \mathcal{S}$
10: $s \leftarrow s'$
11: **end for**
12: **end for**

THEOREM 2. *Consider the Ordered Q-learning algorithm described by the update rule in* (14)*. Let $\gamma < 1$. Let q_* be monotone, i.e., if $s_1 \leq s_2$ according to some order on the states, then $q_*(s_1, a) \leq q_*(s_2, a)$. Then $Q_t(s, a)$ converges to $q_*(s, a)$ w.p.1. for every state $s \in \mathcal{S}$ and for every action $a \in \mathcal{A}(s)$.*

PROOF. A sketch of the proof of Theorem 5 is based on the Policy Improvement theorem's [28]. Let consider a generic step of the OQL algorithm. Let π be the current policy and π' be the policy obtained at the iteration of OQL. Let assume that π' improves the current policy, i.e., $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in \mathcal{S}$. As showed in [28], this implies that $v'_\pi(s) \geq v_\pi(s)\ \forall s \in \mathcal{S}$.

Now let consider any state $s^+ \geq s$. After applying the projection, it holds $q_\pi(s^+, \pi'(s)) \geq q_\pi(s, \pi'(s))$. Finally, since both $\pi'(s)$ and $\pi'(s^+) \in \mathcal{A}(s^+)$, it is possible to write

$$\begin{aligned} v_{\pi'}(s^+) = q_{\pi'}(s^+, \pi'(s^+)) &= \max\{q_{\pi'}(s^+, \pi'(s)), q_{\pi'}(s^+, \pi(s^+))\} \\ &\geq q_{\pi'}(s^+, \pi'(s)) \geq q_{\pi'}(s, \pi'(s)) = v_{\pi'}(s) \end{aligned}$$

which concludes the proof. □

The above proof sketch is based on the simplifying assumption that the true values of the Q-function are available at each iteration; this assumption permits to use dynamic programming arguments. The complete proof of Theorem 5 is rooted in the original argument of convergence for Q-learning, developed by Tsitsiklis in [29], which is based on stochastic approximations. It is presented in Section D. For OQL, the technical difficulty is to account for the use of state-dependent projection Π_s.

It is further possible to define the n-step version of OQL, where the first equation in (14) is replaced by the one of n-step QL [28].

5.2 Stairway Q-Learning

Stairway Q-learning (SQL) is defined imposing on the state space the natural partial order considered in Cor. 1 so that the assumptions of Thm. 5 automatically hold for the system at hand. For every pair of states $s_1 = (x_1, e_1, z)$ and $s_2 = (x_2, e_2, z)$ and for every vector $v \in \mathbb{R}^{|\mathcal{S}|}$, the explicit form of the projection operator on the set of

vectors that suit the partial order considered is

$$\Pi_{s_1}(v(s_2)) = \begin{cases} \max(v(s_1), v(s_2)) & \text{if } x_1 < x_2 \text{ and } e_1 \geq e_2 \text{ or} \\ & \quad x_1 \leq x_2 \text{ and } e_1 > e_2 \\ v(s_1) & \text{if } x_1 = x_2 \text{ and } e_1 = e_2 \\ \min(v(s_1), v(s_2)) & \text{if } x_1 > x_2 \text{ and } e_1 \leq e_2 \text{ or} \\ & \quad x_1 \geq x_2 \text{ and } e_1 < e_2 \\ v(s_2) & \text{otherwise} \end{cases}$$

In Section 7 it will be showed that the performance of SQL improves significantly compared to the standard QL, both in the 1-step version here introduced and in the n-step version.

5.3 Threshold Q-learning

Threshold Q-learning (TQL) considers a simplified partial order of the type $(x, e, z) \geq (x+1, e, z')$ for $x = 1, \ldots, M-1$: it imposes the partial order only w.r.t. energy. The monotonicity condition writes

$$q_*((x, e, z), a) \leq q_*((\lfloor x+1 \rfloor_M, e, z), a) \quad \forall x, e, z, a \tag{15}$$

The convergence follows from Theorem 5. The projection function w.r.t. a certain state s, $\Pi_s : \mathbb{R} \to \mathbb{R}$, can be written as

$$\Pi_{s_1}(v(s_2)) = \begin{cases} \max(v(s_1), v(s_2)) & \text{if } x_1 < x_2 \\ v(s_1) & \text{if } x_1 = x_2 \\ \min(v(s_1), v(s_2)) & \text{if } x_1 > x_2 \\ v(s_2) & \text{otherwise} \end{cases} \tag{16}$$

for every pair of states $s_1 = (x_1, e, z)$ and $s_2 = (x_2, e, z)$. Note that in this case only states with same energy and server availability are compared: this significantly reduces the number of projection operations compared to SQL, especially when B is large. Despite its reduced complexity, in many cases this algorithm has showed similar convergence speed as SQL. Note how for both SQL and TQL it is not possible to guarantee the corresponding policy to have the threshold structure at each timestep. However, the numerical results show that policy displays the threshold structure after a rather small number of iterations.

5.4 Reinforce

One of the baselines algorithms used for comparison is Reinforce (RF) [30]. RF performs a basic policy-gradient iteration in the form

$$\theta_{t+1} = \theta_t + \alpha_t G_t \frac{\nabla_\theta \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t)} \tag{17}$$

where $0 < \alpha_t < 1$ is a stepsize and G_t is the policy reward estimation. The gradient is operated on the following policy parametrization based on the threshold structure of the optimal policy

$$\pi(x, e, z) = \begin{cases} 0 & x < \lfloor \theta_e \rfloor \\ \frac{1}{1+\exp\{k(\theta_e - x - 0.5)\}} & x = \lfloor \theta_e \rfloor \\ 1 & x > \lceil \theta_e \rceil \end{cases} \tag{18}$$

where $k > 0$ is a suitable hyperparameter.

6 MULTI-DEVICE MODEL

The model is now extended to the case of N devices connected to the edge server. Each device $k \in \{1, \ldots, N\}$ follows the single device-server model described in Section 3. At each timeslot the server polls device k with probability $p_{Z,k}$, where $p_Z \in P_Z = \{p_Z \in [0,1]^N : \sum_{k=0}^N p_{Z,k} = 1\}$ and $k = 0$ denotes the null device, i.e., no polling. All the devices are assumed to be independent of each other: by doing so, the optimal policy of each device is not influenced by the state of the other devices at a given timestep; the interesting case of correlated harvesting or data samples is left as part of future works. Moreover, the polling decision of the server is feedforward, i.e., the state of a device is only known once the device is polled, so that the polling action is independent of the state of devices.

The server seeks the optimal random polling probability $p_{Z,k}$ that maximizes the sum of the rewards of the devices, i.e., $R(p_Z) := \sum_{k=1}^N R_{\gamma_k}(p_{Z,k})$, where $R_{\gamma_k}(p_{Z,k})$ obeys to (13), subject to the polling capacity constraint, i.e.,

$$\text{maximize: } R(p_Z) = \sum_{k=1}^N R_{\gamma_k}(p_{Z,k}) \tag{MD}$$

$$\text{subj. to: } \sum_{k=0}^N p_{Z,k} = 1 \tag{19}$$

$$p_{Z,k} \geq 0 \quad k = 0, \ldots, N$$

In order to solve problem (MD), since the discounted reward (13) cannot be computed (unless all the transition probabilities are known), stochastic approximation might be used to find a solution. Furthermore, as already observed in Section 4.2, $R_\gamma(\cdot)$ cannot be assumed differentiable or concave, ruling out stochastic approximation methods requiring differentiable objective functions, such as SPSA [31]. Other direct methods with less strict requirements, such as Enhanced Localized Random Search do exist [32].

The proposed algorithm, namely the Alternating Polling and Policy Improvement (APPI) algorithm, is summarized in Alg. 2. It alternates two steps: 1) a *policy learning* step (line 4) for a given polling probability vector p_Z, and 2) a *polling optimization* step (line 5) optimizing p_Z for a given policy using stochastic approximation methods. For the policy learning step it is sufficient to use one of the learning methods proposed in Sec. 5, e.g., Stairway Q-learning for the efficiency's sake. Conversely, as observed before, if the policy for each device is fixed, the objective function is $\sum_k R_{\gamma_k,\pi_k}(p_{Z,k})$ with $\pi_k = \pi_k^*(p_{Z,k,old})$, i.e., the optimal policy according to the previous value of the polling probability vector. As showed in Section 7.3, this algorithm is both faster and more accurate than general methods for the problem studied.

In order to discuss the polling optimization step, recall that the

Algorithm 2 Pseudocode of APPI

Require: $\epsilon > 0$
1: initial polling probability $p_{Z,\text{new}}$
2: **while** $|p_{Z,\text{old}} - p_{Z,\text{new}}| > \epsilon$
3: $p_{Z,\text{old}} \leftarrow p_{Z,\text{new}}$
4: **Optimal policy learning:** find optimal policy $\pi_k^*(p_{Z,\text{old}})$ for $\forall k$
5: **Polling optimization:** find $p_{Z,\text{new}} \geq 0$ such that

$$\begin{cases} p_{Z,\text{new}} = \arg\max_{p_Z} \sum_k R_{\gamma_k,\pi_k^*(p_{Z,k,\text{old}})}(p_Z) \\ \sum_k p_{Z,\text{new}} = 1 \end{cases}$$

6: **end while**
7: **return** $p_{Z,\text{new}}$

objective function is concave in p_Z (and so a.e. continuously differentiable) suggesting the use of stochastic gradient ascent methods of the Kiefer-Wolfowitz family [33]. The iteration scheme is

$$p_Z^{n+1} = \Pi_\Theta \left(p_Z^n - \alpha_n \hat{g}_n \right) \tag{20}$$

where p_Z^n is the nth iterate of the parameter, $\hat{g}_n$ represents an estimate of the gradient of the objective function, $\{\alpha_n\}_n$ is a sequence converging to 0 and Π_Θ is a projection on the $N+1$-dimensional simplex $\Theta = \{p_Z \geq 0 | \sum p_{Z,k} = 1\}$, i.e., the space of possible polling probability vectors, including the null action when polling is not active. The projection activates when $p_Z \pm c_n \Delta_n$ lies outside the constraint set and reverts to the nearest point in Θ.

The specific technique to determine $\hat{g}_n$ in (20) is based on simultaneous random increments (SPSA) [31]. This technique estimates two increments, one in the positive and one in the negative direction. Due to the independence between different devices, each component of the gradient depends only on the parameter $p_{Z,k}$ of the device considered. Let $\hat{R}_{\gamma_k,\pi_k}$ denote the approximated reward for device k. The corresponding component of the gradient estimate writes

$$(\hat{g}_n)_k = \frac{\hat{R}_{\gamma_k,\pi_k} \left(p_{Z,k} + c_n (\Delta_n)_k \right) - \hat{R}_{\gamma_k,\pi_k} \left(p_{Z,k} - c_n (\Delta_n)_k \right)}{2 c_n (\Delta_n)_k} \tag{21}$$

where $\{c_n\}$ is a sequence converging to 0 and $\{\Delta_n\}$ is an i.i.d. vector sequence of perturbations of i.i.d. components $\{(\Delta_n)_i, i = 1, \ldots, N\}$ with zero mean and where $\mathbb{E}\left[|(\Delta_n)_i^{-2}|\right]$ is uniformly bounded.
The following proposition estabilishes the conditions on the objective function, on the step-size sequence $\{\alpha_n\}$ and on the gradient estimates $\hat{g}_n$ for which the SPSA iteration converges to the global maximum of the corresponding total discounted reward when a policy is fixed for each device.

PROPOSITION 1. *Let $\{\alpha_n\}$ and $\{c_n\}$ be such that $\sum_n \alpha_n = \infty$, $\sum_n \left(\frac{\alpha_n}{c_n}\right)^2 < \infty$. Moreover, assume that $\mathbb{E}\left[|(\Delta_n)_i|^{-2}\right]$ is uniformly bounded on P_Z and that $R_{\gamma_k,\pi_k}(p_{Z,k})$ is concave. Then the iteration* (20) *converges to the optimal parameter p_Z^* w.p.1.*

PROOF. The desired result is obtained by verifying the assumptions of Proposition 1 in [34], whose proof is a consequence of Theorem 5.3.1 in [33].
i. the objective function is differentiable and either concave or unimodal: its differentiability is proved in Section C, while the other condition is assumed to be verified;
ii. $b_n = \mathbb{E}\left[\hat{g}_n | p_Z, s_n\right] - \nabla J(p_Z) \to 0$ w.p.1: it holds from Lemma 2 in [31] and condition i.;
iii. $\sum_{n=1}^\infty \alpha_n^2 \cdot \mathbb{E}\left[e_n^2\right] < \infty$ w.p.1, where $e_n := \hat{g}_n - \mathbb{E}\left[\hat{g}_n \mid \theta_n, s_n\right]$; this condition holds by Lemma 2 in [31] and by the fact that $R((p_Z)_k, \omega)$, representing the reward of sample path ω has second moment uniformly bounded in p_Z: we can notice that it is independent of p_Z. Indeed, given a path, since $x \leq M$ and $p_Z \in [0, 1]$, the reward is upper bounded by a constant, therefore its second moment is uniformly bounded, which concludes the proof. □

Hence, fixed the policy for each device, the corresponding optimal polling probability vector is attained. Thanks to the convergence properties of the learning methods (see Section 5) for a fixed vector p_Z, and since the number of policies is finite, this proves that APPI converges w.p.1.

Parameter	Subsec. 7.1	Subsec. 7.2 and 7.3
M	15	$\{10, \ldots, 25\}$
B	15	$\{10, \ldots, 25\}$
γ	0.95	$\{0.9, \ldots, 0.99\}$
p_Z	0.05	$\{0.01, \ldots, 0.1\}$
δ	1	1
Reward	$r(x) = M - x$	$r(x) = M - x$
Discharge Penalty	$d(e) = e^4 \cdot \mathbb{1}\{e \geq 0\}$	$d(e) = \mathbb{1}_{e<0}(-e)^k$, with $k \in \{1, 2, 3, 4\}$
Harvesting	Poisson ($\lambda = 1$)	Poisson ($\lambda = 1$)
Processing Cost	uniform, binary and symmetric for $\mu \in \{1, 4, 8.5\}$	symmetric with $\mu \in \{1, \ldots, B/2\}$ $\sigma \in \{1, 2, 3, 4\}$

Table 2: The system parameters used in the numerical experiments.

Unfortunately, this does not imply the convergence to the globally optimal polling probability vector, as the function to optimize is not concave, but rather just the pointwise maximum of a set of concave functions. Due to its particular structure, only convergence to the local optimum of the objective function can be ensured.

7 NUMERICAL RESULTS

The numerical experiments are divided into 3 parts. The first one describes the performance of the single device-server system tested on a range of system parameters. The second one compares the Ordered Q-learning methods introduced in Sec. 5 with baseline RL algorithms. Finally, the performance of the APPI algorithm introduced in Sec. 6 is assessed against alternative general methods. The devices and server parameters, namely $M, B, \gamma, \delta, p_Z, r(\cdot), d(\cdot), p_H$ and p_C, constitute the *environment*. For the data processing cost C three possible probability distributions will be considered: the *uniform* one in $\{1, \ldots, B+1\}$, the *binary* one where $p_C(1) = p_C(B+1) = 1/2$ and finally the *symmetric* one obtained as $p_C(c) = A \cdot \exp\{-\frac{1}{2}((c-\mu)/\sigma)^2\}$ for $c \in \{1, \ldots, B+1\}$: μ and σ shape the distribution, whereas A is an appropriate normalization constant. In Table 2 the first column represents the environment used in Subsec. 7.1. The second column reports the sets from which the system parameters are drawn uniformly at random in the experiments of Subsec. 7.2 and Subsec. 7.3.

7.1 Device-server performance evaluation

Fig. 3a describes the impact of the server availability p_Z onto the average (peak) AoI in a single device system. Excluding the case $\mu = 1$, i.e., for lower values of the average batch processing cost, the average AoI is maximal for $p_Z = 0$ and attains its minimum close to $p_Z = 1$ (the optimal p_Z is highlighted with dots in Fig. 3a). In fact, for higher processing cost, a larger server polling probability permits to select action $a = 1$ more often, as the possible battery discharge due to data processing is compensated by frequent task offloading events. For lower cost figures, instead, it is the roundtrip delay δ that renders the offloading less convenient: when $\delta = 0$, actually, the corresponding curve (not reported for the space's sake) becomes strictly decreasing.

Fig. 3b plots the value of the optimal polling probability p_Z for increasing values of δ. These are obtained by means of an appropriate stochastic optimization. The results show, as expected, that

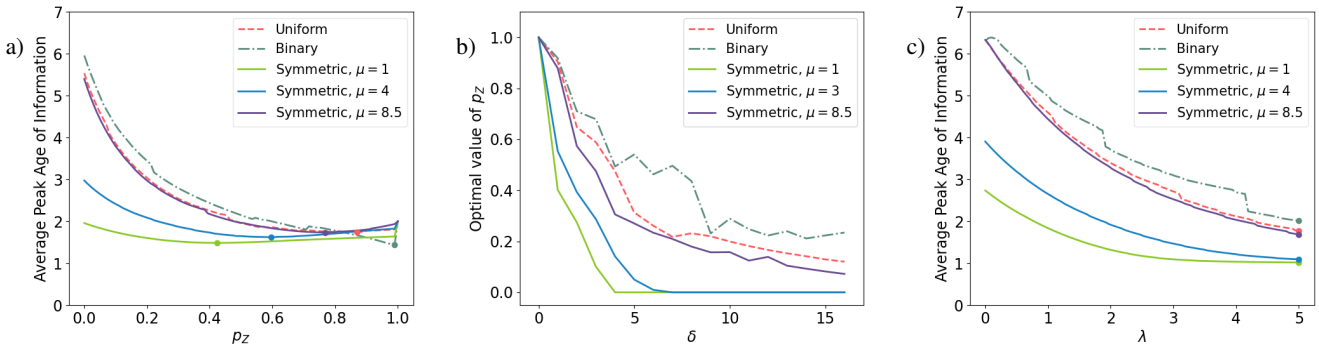

Figure 3: a) Average peak AoI for increasing polling probability p_Z; b) optimal p_Z for increasing round-trip delay δ; c) Average peak AoI for increasing average harvesting rate λ.

the optimal value of p_Z decreases as δ increases, irrespective of the distribution of the processing cost. Fig. 3b shows that offloading can be detrimental for larger values of δ: indeed, for lower batch processing costs ($\mu = 1$ and $\mu = 3$) the value that minimizes the average peak AoI is $p_Z = 0$. In these cases the risk of emptying the battery is sufficiently low to choose action $a = 1$ very frequently. In turn, the presence of the server, and the additional delay required by task offloading, becomes a penalty.

Finally, Fig. 3c depicts the average AoI for increasing average harvesting rates, i.e., for increasing values of λ (note that here $\lambda = 0$ indicates the deterministic case $H \equiv 1$). Irrespective of the processing cost distribution p_C, the average peak AoI is monotone decreasing, as expected and approaches $1 + \delta p_Z$ for large recharging rates.

7.2 Learning the optimal policy

The RL algorithms introduced in Section 5 are compared with two baselines, namely QL and RF. Each test is repeated over 50 different environments where the system's parameters are drawn uniformly at random, as reported in Table 2. Each experiment consists in a sequence of 3000 episodes. At the end of each episode, the discounted reward is calculated using a policy evaluation step. The evaluation is truncated when the weight discount falls below 0.001. The estimation of the optimal reward (12) is collected starting from a state $s_0 = (1, e, z)$, with e and z drawn uniformly at random. In order to compare results of different runs, the rewards have been normalized against the baseline stationary policy which chooses action $a = 1$ if the state is such that either $z = 1$ or $s = (M, B, 0)$.

Finally, for each tested algorithm several sets of hyperparameters have been probed to determine the configuration ensuring the highest discounted reward; in fact, such parameters are observed to influence heavily the performance of the different algorithms. For the Q-learning type of algorithms (QL, TQL and SQL), these consist of n, determining the adopted n-steps variant, and ϕ and β, which determine the learning rate $\alpha_t(s_t) = \phi \cdot (\#\text{visits in state } s_t)^{-\beta}$. For Reinforce the hyperparameter are k, appearing in (18), as well as n and m defining stepsize $\alpha_t = n/(t+1)^m$.

The results of the experiments are summarized in Table 3. In particular, RF appears to converge quickly, but to a suboptimal policy. In order to avoid this, in policy gradient RL it is possible to introduce multiple simultaneous state-action perturbations, e.g., trust-region techniques [35], at the price of increased complexity. Finally, SQL outperforms the other methods attaining a gain of 10%.

	250 episodes	**1000 episodes**	**3000 episodes**
QL	0.723 ± 0.254	0.796 ± 0.220	0.810 ± 0.208
TQL	0.752 ± 0.255	0.857 ± 0.181	0.880 ± 0.117
SQL	0.850 ± 0.209	$\mathbf{0.943 \pm 0.051}$	$\mathbf{0.964 \pm 0.060}$
Reinforce	$\mathbf{0.893 \pm 0.225}$	0.893 ± 0.227	0.893 ± 0.224

Table 3: RL tests: discounted reward for increasing number of episodes; best performance results are marked bold.

7.3 Multi-device system

The two last experiments evaluate the performance of the APPI learning procedure introduced in Sec. 6. To this aim, three alternative algorithms to be compared to APPI have been implemented. They replace the polling probability improvement step with Naive Random Search (NRS), Enhanced Random Search (ENRS) and SPSA, respectively. Details on those procedures are found in [32].

The first test covers a scenario with $N = 3$ devices. For each run of the experiment, a set of parameters is generated at random for each device (i.e., M, B, γ, p_Z, δ $r(\cdot), d(\cdot), p_C$ and p_H) and an initial polling probability distribution is imposed on the edge server. The device parameters are drawn uniformly at random from the sets described in Table 2, second column. For each algorithm it is recorded 1) the total discounted reward and 2) number of policy learning iterations performed. The results are averaged over 50 runs, by considering each time different initial polling distributions and different environments. Also, all the reward values are normalized to fall into an interval $[R_{\min}, R_{\max}]$. $R_{\max}$ is the reward obtained by an ideal APPI implementation which finds the optimal policy using value iteration, while $R_{\min}$ is the reward obtained averaging 10 runs under a random polling probability and the corresponding device's optimal policies. The numerical results are collected in Table 4: APPI consistently outperforms the other algorithms by attaining higher discounted reward and requiring much fewer policy improvement

Algorithm	Discounted reward	Policy learning steps
NRS	0.423 ± 0.237	4.98 ± 1.378
ENRS	0.678 ± 0.184	10.06 ± 3.331
SPSA	0.449 ± 0.276	5.54 ± 1.846
APPI	$\mathbf{0.747 \pm 0.143}$	$\mathbf{2.98 \pm 1.086}$

Table 4: Comparison of different polling improvement algorithms for $N = 3$.

N	Discounted reward	Policy learning steps
1	$\mathbf{0.794 \pm 0.222}$	$\mathbf{2.18 \pm 0.77}$
3	0.747 ± 0.143	2.98 ± 1.086
5	0.699 ± 0.192	3.78 ± 1.221
7	0.724 ± 0.222	4.08 ± 1.074
10	0.714 ± 0.237	5.06 ± 1.302

Table 5: Performance of APPI for increasing number of devices connected to the edge server.

steps as well. The last experiment tests the scalability of APPI by increasing the number of devices N. The outcomes are reported in Table 5. Those are the average of 50 simulations per value of N. As before, in each simulation the environment is drawn at random using the distributions indicated in Table 2. The normalized reward attained by APPI appears marginally affected by the number of devices, whereas the number of policy improvement steps is in the order of a few units and increases linearly.

8 CONCLUSIONS

This work has provided a theoretical framework to model edge devices running AI applications whose energy footprint depends on the data being processed. It factors in energy harvesting and edge-server task offloading. The optimal processing policy on edge devices is hence derived and its key structural features are proved. This is the basis for a model-based, specialized reinforcement learning method, namely Ordered Q-learning. The learning procedure is hence applied to the multidevice setting where the server's offloading rate vector is optimized by alternating stochastic gradient ascent and policy learning. Numerical tests performed against the ground-truth, i.e., an actual optimal policy, show that existing baselines are systematically outperformed both in accuracy *and* convergence speed. This confirms that by rooting RL techniques into the structural properties of the optimal solution, important performance margins against generic RL solutions are attained. Several extension of the proposed models are indeed possible in the multi-device setting. In particular, one could include capacity constraints. This in turn requires to study optimal randomized stationary policies for constrained MDPs. Also, to improve the learning procedure of the polling probability, one could use the history on the device state retrieved at polling instants.

REFERENCES

[1] Y. C. Hu et al. Mobile edge computing: a key technology towards 5G. Technical Report ISBN No. 979-10-92620-08-5, ETSI, Sept. 2015.

[2] J.-D. Lovelock et al. Forecast: The business value of artificial intelligence, worldwide, 2017-2025. Technical report, Gartner Inc., March 2018.

[3] S. Wang et al. Adaptive federated learning in resource constrained edge computing systems. *IEEE JSAC*, 37(6):1205–1221, 2019.

[4] S. Kaul, R. Yates, and M. Gruteser. Real-time status: How often should one update? In *Proc. of IEEE INFOCOM*, pages 2731–2735. IEEE, 2012.

[5] R. Yates, Y. Sun, D.R. Brown, S. Kaul, E. Modiano, and S. Ulukus. Age of information: An introduction and survey. *IEEE SAC*, 39(5), 2021.

[6] I. Kadota and E. Modiano. Minimizing the age of information in wireless networks with stochastic arrivals. In *Proc. of ACM Mobihoc*, page 221–230, 2019.

[7] R. Li et al. Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time IoT applications. *IEEE IoT Journal*, 8(19), 2021.

[8] Q. Wang, H. Chen, Y. Li, Z. Pang, and B. Vucetic. Minimizing aoi for real-time monitoring in resource-constrained industrial IoT networks. In *Proc. of IEEE INDIN*, 2019.

[9] Elif Tugce Ceran, Deniz Gunduz, and Andras Gyorgy. Learning to minimize age of information over an unreliable channel with energy harvesting. *arXiv preprint arXiv:2106.16037*, 2021.

[10] J. Yang, X. Wu, and J. Wu. Optimal online sensing scheduling for energy harvesting sensors with infinite and finite batteries. *IEEE JSAC*, 34(5), 2016.

[11] Mohamed A Abd-Elmagid, Harpreet S Dhillon, and Nikolaos Pappas. Aoi-optimal joint sampling and updating for wireless powered communication systems. *IEEE Transactions on Vehicular Technology*, 69(11):14110–14115, 2020.

[12] G. Stamatakis, N. Pappas, and A. Traganitis. Control of status updates for energy harvesting devices that monitor processes with alarms. In *Proc. of IEEE Globecom*, 2019.

[13] M.-H. Chen, B. Liang, and M. Dong. Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point. In *Proc. of IEEE INFOCOM*, pages 1–9, 2017.

[14] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM TON*, 24(5), 2015.

[15] J. Yan, S. Bi, and Y.J.A. Zhang. Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach. *IEEE Trans. on Wireless Communications*, 19(8), 2020.

[16] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis. Performance optimization in mobile-edge computing via deep reinforcement learning. In *Proc. of IEEE VTC (Fall)*, 2018.

[17] A. Arafa, R.D. Yates, and H.V. Poor. Timely cloud computing: Preemption and waiting. In *Proc. of the Annual Allerton Conference*, 2019.

[18] X. Song, X. Qin, Y. Tao, B. Liu, and P. Zhang. Age based task scheduling and computation offloading in mobile-edge computing systems. In *Proc. of IEEE WCNCW*. IEEE, 2019.

[19] J. Huang, H. Gao, S. Wan, and Y. Chen. AoI-aware energy control and computation offloading for industrial IoT. *Future Gen. Computer Systems*, 139:29–37, 2023.

[20] B. Barakat, H. Yassine, S. Keates, I. Wassell, and K. Arshad. How to measure the average and peak aoi in real networks? In *Proc. of EWC*, 2019.

[21] E. Altman, R. El Azouzi, D. Menasché, and Y. Xu. Poster: aging control for smartphones in hybrid networks. *ACM SIGMETRICS Perf. Evaluation Review*, 39(2):68–68, 2011.

[22] J.P. Champati, R. Avula, T.J. Oechtering, and J. Gross. On the minimum achievable age of information for general service-time distributions. In *Proc. of IEEE INFOCOM*, 2020.

[23] G. Yao, A. Bedewy, and N.B. Shroff. Age-optimal low-power status update over time-correlated fading channel. *IEEE Trans. on Mobile Computing*, 2022. doi: 10.1109/TMC.2022.3160050.

[24] M. Puterman. *Markov Decision Processes*. Wiley, 2014.

[25] P. Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer, 2001.

[26] S. M. Ross. *Stochastic processes*. Wiley, 1996.

[27] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

[28] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[29] J.N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

[30] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Proc. of NIPS*, 1999.

[31] M.C. Fu and S.D. Hill. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE transactions*, 29(3):233–243, 1997.

[32] J.C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Wiley, 2005.

[33] H.J. Kushner and D.S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Applied Mathematical Sciences. Springer, 1978.

[34] P. L'Ecuyer and P.W. Glynn. Stochastic optimization by simulation: Convergence proofs for the GI/G/1 queue in steady-state. *Management Science*, 40(11):1562–1578, 1994.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proc. of ICML*, pages 1889–1897. PMLR, 2015.

A TRANSITION PROBABILITIES: STOCHASTIC HARVESTING RATE

This section derives the transition probabilities (1) to the case of a stochastic harvesting sequence of i.i.d. random variables $\{H_t\}$. Here, both both H_t and C_t may have unbounded support $[0, +\infty)$. When the action is $a = 0$ they write

$$p(s'|s,0) = \begin{cases} p_Z(z' \mid 0) p_H(h) & \text{if } s = (x,e,0),\, s' = ([x+1]_M, e+h, z'), \text{and } e+h < B \\ p_Z(z' \mid 0) \sum_{h=B-e}^{\infty} p_H(h) & \text{if } s = (x,e,0), \text{and } s' = ([x+1]_M, B, z') \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

The generalization of (2), for the case when $a = 1$, writes:

$$p(s' \mid s,1) = \begin{cases} p_Z(z' \mid 0)\left(\sum_{r=0}^{\infty} \Gamma_{1,r} p_C(e+r-e') + \sum_{k=2}^{\infty} \sum_{r=k-1}^{\infty} \Gamma_{k-1,r} \sum_{t=e+r+1}^{\infty} p_C(c) p_H(e'+c-r-e) \right) & \begin{array}{r} s = (x,e,0), \\ s' = (1,e',z'), \\ 0 \le e' < B \end{array} \\ p_Z(z \mid 0) \sum_{c=1}^{\infty} p_C(c) \sum_{h=B-e+c}^{\infty} p_H(h) & \begin{array}{r} s = (x,e,0), \\ s' = (1,B,z') \end{array} \\ p_Z(z' \mid 1) p_H(h) & \begin{array}{r} s = (x,e,1), \\ s' = (1,e+h,z'), \\ e' < B \end{array} \\ p_Z(z' \mid 1) \sum_{h=B-e}^{\infty} p_H(h) & \begin{array}{r} s = (x,e,1), \\ s' = (1,B,z') \end{array} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

where $\Gamma_{k,r}$ has been defined in (10). For what concerns the instantaneous reward, it depends on the difference a $H_t - C_t$, that is

$$r_{t+1}(s_t, a_t) = \begin{cases} u(x_t) - \sum_{h_t=1}^{+\infty} \sum_{c_t=1}^{+\infty} d(e_t + h_t - c_t) \cdot a_t & \text{if } z_t = 0 \\ u([x_t + \delta]_M) & \text{if } z_t = 1 \end{cases} \tag{24}$$

B DERIVATION OF THE AVERAGE SOJOURN TIME

This section shows how to derive the expression (9) for $\chi(e')$, i.e., the expected sojourn time when the transition leads to a state $s' = (1, e,', z')$. In particular from the law of total probability it follows

$$\begin{aligned}
\chi(e') &= (1+\delta) p_Z(1) \mathbb{P}\big(e + h_1 = e'\big) + p_Z(0) \left(\mathbb{P}\big(e + h_1 - c = e'\big) + 2\mathbb{P}(e + h_1 - c < 0)\, \mathbb{P}\big(e + h_1 + h_2 - c = e'\big) + \ldots \right) \\
&= (1+\delta) p_Z(1) \mathbb{P}\big(e + h_1 = e'\big) + p_Z(0) \left(\mathbb{P}\big(e + h - c = e'\big) + \sum_{k=2}^{\infty} k \mathbb{P}\left(e + \sum_{j=2}^{k-1} h_j - c < 0 \right) \mathbb{P}\left(h_k = e' + c - \sum_{j=2}^{k-1} h_j - e \right) \right) \\
&= (1+\delta) p_Z(1) \sum_{e=0}^{B} p_E(e) p_H(e'-e) + p_Z(0) \Bigg(\sum_{e=0}^{B} p_E(e) \sum_{h_1=1}^{\infty} p_H(h_1) p_C(e + h_1 - e') + \\
&\quad + \sum_{k=2}^{\infty} k \sum_{e=0}^{B} p_E(e) \sum_{h_1=1}^{\infty} p_H(h_1) \cdot \cdots \cdot \sum_{h_{k-1}=0}^{\infty} p_H(h_k) \sum_{c=e+\sum_{j=1}^{k-1} h_j + 1}^{\infty} p_C(c) p_H(e' + c - \sum_{j=1}^{k-1} h_j - e) \Bigg) \\
&= (1+\delta) p_Z(1) \sum_{e=0}^{B} p_E(e) p_H(e'-e) + p_Z(0) \Bigg(\sum_{e=0}^{B} p_E(e) \sum_{r=1}^{\infty} \Gamma_{1,r} p_C(e + h_1 - e') + \\
&\quad + \sum_{k=2}^{\infty} k \sum_{e=0}^{B} p_E(e) \sum_{r=1}^{\infty} \Gamma_{k-1,r} \sum_{c=e+\sum_{j=1}^{k-1} h_j + 1}^{\infty} p_C(c) p_H(e' + c - \sum_{j=1}^{k-1} h_j - e) \Bigg)
\end{aligned}$$

C PROPERTIES OF THE DISCOUNTED REWARD

The APPI learning procedure is based on the assumptions that function $R_{\gamma,\pi}(p_Z)$ which describes the discounted reward for a fixed policy $\boldsymbol{\pi}$ is smooth and concave.

Continuity and differentiability

As recalled in Section 4.2, the function $R_{\gamma,\pi}(p_Z)$ which describes the discounted reward for a fixed policy $\boldsymbol{\pi}$ and a given initial state distribution is continuous in $[0,1]$ and differentiable in $(0,1)$. The following theorem formally proves the result.

THEOREM 3. *$R_{\gamma,\pi}(p_Z)$ is continuous in $[0,1]$ and differentiable in $(0,1)$ w.r.t. p_Z for every fixed policy π.*

PROOF. To show the continuity of $R_{\gamma,\pi}(p_Z)$ we consider the equivalent formulation (11). First of all notice that $d_0(p_Z)$ and $P^{\pi}(p_Z)$ are componentwise continuous and differentiable in p_Z. By recalling the expression for the average reward

$$R_{\gamma,\pi}(p_Z) = d_0(p_Z)\left(I + \gamma P_{\pi}(p_Z) + \gamma^2 P_{\pi}^2(p_Z) + \ldots\right) r_{\pi}$$

and recalling that r_{π} is a vector in $\mathbb{R}^{|\mathcal{S}|}$ and whose terms are independent on p_Z, the statement follows. □

Concavity

Extensive numerical experiments have showed that $R_{\gamma,\pi}(p_Z)$ is concave for a fixed the discount factor γ and and the policy π when $\delta > 0$. The case with $\gamma = 0$ gives that $R_{0,\pi}(p_Z) = \langle d_0(p_Z), r_{\pi}\rangle$, which is a linear combination of concave functions and will therefore not be considered in the following. The result is demonstrated for a specific example. Backed by the variety of experiments tested, it is likely that a similar conclusion can be reached also in general.

The example is represented by a system with $B = 1$ and $M = 2$. This corresponds to the case when battery of the device is either full or empty while the information can just be new, i.e., $x = 1$, or old, i.e., $x = 2$. Moreover, assume a deterministic harvesting rate $H \equiv 1$ and cost distribution

$$p_C(c) = \begin{cases} \frac{1}{4} & c = 1 \\ \frac{3}{4} & c = 2 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Finally, consider $\delta = 1$ and $\gamma \in (0,1)$. The policy studied is the trivial one, π, such that

$$\pi(x,e,z) = \begin{cases} 1 & z = 1 \text{ or } s = (M,B,0) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Consider now an By assuming an initial distribution where $z = 0$, the transition towards a state $s' = (x,e,z)$ having AoI x, energy e and any value if z is described by

$$P^0 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ \frac{3}{4} & 0 & \frac{1}{4} & 0 \end{pmatrix} \tag{27}$$

where states are indexed using index function $\text{index}(1,0) = 1, \text{index}(2,0) = 2, \text{index}(1,1) = 3, \text{index}(2,1) = 4$. The transitions from states $s = (x,e,1)$ are described by the matrix

$$P^1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{28}$$

Now, let the states be ordered according to function *index* and the server availability, i.e., by having states with $z = 0$ first. The resulting transition probability of the Markov chain given by policy π

$$P^{\pi}(p_Z) = \begin{pmatrix} (1-p_Z)P^0 & p_Z P^0 \\ (1-p_Z)P^1 & p_Z P^1 \end{pmatrix} \tag{29}$$

As seen in (12), in order to compute the discounted reward it has to be defined the inverse of the matrix $I - \gamma P_\pi$, which will not be reported here, but can be computed by standard means. The product between the initial distribution $d_0(p_Z)$ and the latter matrix gives

$$d_0(p_Z)\left(I-\gamma P^\pi\right)^{-1} = \left(\tfrac{1-p_Z}{2} \quad 0 \quad \tfrac{1-p_Z}{2} \quad 0 \quad \tfrac{p_Z}{2} \quad 0 \quad \tfrac{p_Z}{2} \quad 0\right)\left(I-\gamma P^\pi\right)^{-1} \tag{30}$$

$$= \frac{1}{4\gamma^2 - 4\gamma^2 p_Z + 4\gamma p_Z - 4} \begin{pmatrix} -\gamma^2 + 3\gamma^2 p_Z^3 - 7\gamma^2 p_Z^2 - 2\gamma p_Z^2 + 5\gamma^2 p_Z + 2\gamma p_Z + 2p_Z - 2 \\ 0 \\ \gamma^2 - 3\gamma^2 p_Z^3 + 7\gamma^2 p_Z^2 + 2\gamma p_Z^2 - 5\gamma^2 p_Z - 2\gamma p_Z + 2p_Z - 2 \\ 4(-\gamma - \gamma p_Z^2 + 2\gamma p_Z \\ -3\gamma^2 p_Z^3 + 4\gamma^2 p_Z^2 + 2\gamma p_Z^2 - \gamma^2 p_Z - 2p_Z) \\ 0 \\ 3\gamma^2 p_Z^3 - 4\gamma^2 p_Z^2 - 2\gamma p_Z^2 + \gamma^2 p_Z - 2p_Z \\ 4(\gamma p_Z^2 - \gamma p_Z) \end{pmatrix}^T$$

For the system at hand, the instantaneous reward can be written as $r((x,e,z),a) = u(x)$. Finally, direct computation show that

$$R_{\gamma,\pi}(p_Z) = \frac{x\left(u(1) + \gamma u(2) - u(2)\right) - \left(u(1) + \gamma u(2)\right)}{(\gamma - \gamma^2)x + \gamma^2 - 1} \tag{31}$$

This function is concave for $\gamma \in (0,1)$ as its second derivative can be explicitly computed and it is always non positive for $u(1) \geq u(2)$:

$$\begin{aligned} \frac{d^2}{dp_Z^2} R_{\gamma,\pi}(p_Z) &= \frac{2(\gamma-\gamma^2)\left((\gamma-\gamma^2)(-u(1)-\gamma u(2)) - (u(1)+\gamma u(2)-u(2))(\gamma^2-1)\right)}{(\gamma^2 - \gamma^2 p_Z + \gamma p_Z - 1)^3} \\ &= \frac{2(\gamma-\gamma^2)\left(u(1)(1-\gamma) - u(2)(1-\gamma)\right)}{(\gamma^2 - \gamma^2 p_Z + \gamma p_Z - 1)^3} \\ &= \frac{2(\gamma-\gamma^2)(1-\gamma)(u(1)-u(2))}{(\gamma^2 - \gamma^2 p_Z + \gamma p_Z - 1)^3} \end{aligned}$$

For $\gamma \in [0,1)$, $u(1) \geq u(2)$ and $p_Z \in [0,1]$ the numerator is clearly nonnegative, while the denominator is always negative, thus implying that the second derivative of $R_{\gamma,\pi}(p_Z)$ is nonThen-positive and the function itself is concave in the case studied.

Note how this example also proves that $\delta > 0$ is a necessary condition to have the concavity of the discounted reward $R_{\gamma,\pi}$. Indeed, if $\delta = 0$ it can be showed that

$$R_{\gamma,\pi}(p_Z) = \frac{\gamma u(2) p_Z - \left(u(1) + \gamma u(2)\right)}{(\gamma - \gamma^2)p_Z + \gamma^2 - 1}$$

which is not concave for $p_Z \in (0,1)$.

D CONVERGENCE OF ORDERED Q-LEARNING

The proof of the convergence of the Ordered Q-learning algorithm introduced in Section 5.1 follows the same path as the one for Q-learning introduced in [29], of which this is a mere adaptation to the particular structure of the functions involved in the case studied. This section is divided into two parts: in the first one, it will be proved a results for generic sequences, while in the latter it will be showed how Ordered Q-learning's convergence is a consequence of the first result.

The general algorithm consists in an alternation of noisy updates of one single component of a vector $x \in \mathbb{R}^n$ and an update of other elements of the vector, with the goal of maintaining a certain (possibly partial) order on the values of all the components of x. Let $x(t)$ be the vector at time t and denote as $x_i(t)$ its i-th component. The update equation is

$$\begin{aligned} \overline{x}_i(t+1) &= x_i(t) + \alpha_i(t)\left(F_i\left(x(t)\right) - x_i(t) + w_i(t)\right) \\ x_j(t) &= \Pi_i\left(\overline{x}_i(t)\right) \quad \forall j \in \mathcal{S} \end{aligned} \tag{32}$$

with the function $\Pi_i(\cdot)$ indicating the projection on the space of ordered vectors.

Before stating the convergence result for (32), some assumption required in its proof will be described.

The first one refers to the statistics of the random variable w_i involved in the algorithm.

ASSUMPTION 1. *(i): $x(0)$ is $\mathcal{F}(0)$-measurable*
(ii): for every i and t, $w_i(t)$ is $\mathcal{F}(t)$-measurable
(iii): for every i and t, $\alpha_i(t)$ is $\mathcal{F}(t)$-measurable
(iv): for every i and t, $\mathbb{E}\left[w_i(t)|\mathcal{F}(t)\right] = 0$
(v): there exist deterministic constants A and B such that

$$\mathbb{E}\left[w_i^2(t)|\mathcal{F}(t)\right] \leq A + B \max_j \max_{\tau \leq t} \left|x_j(\tau)\right| \quad \forall i,t$$

It follows an assumption on the values of the sequence of learning rates $\{\alpha_i\}_i$:

ASSUMPTION 2. *(i): for every i,*

$$\sum_{t=0}^{\infty} \alpha_i(t) = \infty \quad w.p.1$$

(ii): there exists some deterministic constant C such that for every i,

$$\sum_{t=0}^{\infty} \alpha_i^2(t) \leq C \quad w.p.1$$

Finally, some assumptions on the structure of the iteration mapping F are required.

ASSUMPTION 3. *(i): F is monotone*
(ii): the mapping F is continuous
(iii): the mapping F has an unique fixed point x^*
(iv): if $e \in \mathbb{R}^n$ is the vector with all components equal to 1 and r is positive and scalar, then

$$F(x) - re \leq F(x - re) \leq F(x + re) \leq F(x) + re$$

(v): for every value of $i \in \mathcal{S}$ and for every vector $v \in \mathbb{R}^n$

$$F(\Pi_i(v)) = \Pi_i(F(\Pi_i(v)) \tag{33}$$

The last assumption requires that the mapping F maintains the order on the values when applied to an already ordered vector $v = \Pi_i(v)$. We can now state and prove the general result that will be later used to find the convergence of Ordered Q-learning.

THEOREM 4. *Assume that Assumptions 1, 2 and 3 hold, and suppose that $x(t)$ is bounded with probability 1. Then $x(t)$ computed according to* (32) *converges to $x^\star$ with probability 1.*

The proof of Theorem 4 requires the use of additional lemmas that will be stated when needed.

LEMMA 2. *Let $\{\mathcal{F}(t)\}$ be an increasing sequence of σ-fields. For each t, let $\alpha(t), w(t)$ and $B(t)$ be $\mathcal{F}(t)$-measurable scalar random variables. Let C be a deterministic constant. Suppose that the following hold with probability 1:*

(1) $\mathbb{E}\left[w(t) \mid \mathcal{F}(t)\right] = 0$
(2) $\mathbb{E}\left[w^2(t) \mid \mathcal{F}(t)\right] \leq B(t)$
(3) $\alpha(t) \in [0, 1]$
(4) $\sum_{t=0}^{\infty} \alpha(t) = \infty$
(5) $\sum_{t=0}^{\infty} \alpha^2(t) \leq C$

Suppose that the sequence $\{B(t)\}$ is bounded with probability 1. Let $W(t)$ satisfy the recursion

$$W(t+1) = (1 - \alpha(t))W(t) + \alpha(t)w(t)$$

Then $\lim_{t\to\infty} W(t) = 0$ *with probability 1.*

The proof is found in [29].

It is then proved that the sequence $x(t)$ is bounded:

PROPOSITION 2. *The sequence $x(t)$ given by* (32) *is bounded with probability 1.*

PROOF. This can proved by induction. First of all, observe that to have $x(0)$ bounded, it is sufficient to choose a bounded initial vector and thus it is trivially proved.
Then, assume $x(t)$ bounded. To prove the boundedness of $x(t+1)$ it is sufficient to observe that $\overline{x}(t+1)$ is bounded, as the update rule is the same as in Q-learning (and it is known, for Theorem 1 in [29] that Q-learning gives a bounded sequence). The desired result is then a consequence of the definition of the projection Π_i for a certain i. □

From the bounded sequence $\{x(t)\}$ one can further define two additional sequences, $\{U^k\}$ and $\{L^k\}$. Let r be a scalar such that $x^\star - re \leq x(t)x^\star + re$ for every t, with e being the vector with unitary value in each component. The first term of each sequence is defined, respectively, as $U^0 = x^\star + re$ and $L^0 = x^\star - re$, while the other terms are defined in a recursive way:

$$U^{k+1} = \frac{U^k + F(U^k)}{2} \quad k \geq 0$$

and

$$L^{k+1} = \frac{L^k + F(L^k)}{2} \quad k \geq 0$$

Note how, due to (33) and the vectors U^0 and L^0 being ordered, for every $k \geq 0$ and for every i, the following equalities are verified:

$$\Pi_i(U^k) = U^k, \quad \Pi_i(L^k) = L^k$$

Lemma 3, Lemma 4 and Lemma 6 will be simply stated. Their proof can be found in [29].

LEMMA 3. *For every $k \geq 0$, the inequalities*

$$F(U^k) \leq U^{k+1} \leq U^k$$

and

$$F(L^k) \geq L^{k+1} \geq L^k$$

are verified.

LEMMA 4. *The sequences of ordered elements $\{U^k\}$ and $\{L^k\}$ converge to $x^\star$.*

To conclude the proof of theorem 4, it will be sufficient to show that for every k there exists some t_k such that

$$L^k \leq x(t) \leq U^k \quad \forall t \geq t_k \tag{34}$$

Once again, a proof by means of induction will be build.
For $k = 0$, equation (34) is always verified with $t_0 = 0$.
Now, fix a certain k for which (34) is true. Let $W_i(0) = 0$ and

$$W_i(t+1) = (1 - \alpha_i(t))W_i(t) + \alpha_i(t)w_i(t)$$

Due to Lemma 2, $\lim_{t\to\infty} W_i(t) = 0$. Moreover, for every time t_0, can be further defined $W_i(t_0; t_0) = 0$ and

$$W_i(t+1; t_0) = (1 - \alpha_i(t))W_i(t; t_0) + \alpha_i(t)w_i(t) \quad t \geq t_0$$

Following the same argument as above, it is proved that $\lim_{t\to\infty} W_i(t; t_0) = 0$.
Then, one can design a sequence $X_i(t), t \geq t'_k$, by letting $X_i(t'_k) = U_i^k$ and

$$X_i(t+1) = (1 - \alpha_i(t))X_i(t) + \alpha_i(t)F_i(U^k) \quad t \geq t'_k$$

With these definitions, the following lemma can be proved.

LEMMA 5. $\overline{x}_i(t) \leq X_i(t) + W_i(t; t'_k) \quad t \geq t'_k$

PROOF. This lemma is analogous to Lemma 6 in [29], where the inequality was proved for $x_i(t)$ with the corresponding definition. It can be equally proved in our case following the same proof, with the care of considering $\overline{x}_i(t)$ at each step in spite of $x(t)$. □

Now, let δ_k be equal to the minimum of $\left(U_i^k - F_i(U^k)\right)/4$, where the minimum is taken over all i for which $U_i^k - F_i(U^k)$ is positive. Clearly δ_k is well-defined and positive, unless $U^k = F(U^k)$, but this would imply that convergence has been reached.
Let $t''_k \geq t'_k$ be such that

$$\prod_{\tau=t'_k}^{t''_k-1} (1 - \alpha_i(\tau)) \leq \frac{1}{4}$$

and

$$W_i(t; t'_k) \leq \delta_k$$

for all $t \geq t''_k$. Such value of t''_k exists due to the first assumption on the sequence of αs and the convergence of $W_i(t; t'_k)$ to 0.

LEMMA 6. $\overline{x}_i(t) \leq U_i^{k+1}$ *for all i and $t \geq t''_k$.*

The proof of this lemma is the same as the one of Lemma 7 in [29], considering $\overline{x}_i(t)$ instead of $x_i(t)$. Finally, thanks to the knowledge on the structure of the solution, it can further be proved the following lemma.

LEMMA 7. $x_i(t) \leq U_i^{k+1}$ *for all i and $t \geq t''_k$.*

PROOF. Lemma 6 gives that $\overline{x}_i(t) \leq U_i^{k+1}\ \forall t \geq t''_k$
Now consider $x(t) = \Pi_i(\overline{x}(t))$, where i represents the state visited at time t for which the updated estimate has been computed in the first equation.

$$x_j(t) = \begin{cases} \max\left(\overline{x}_j(t), \overline{x}_i(t)\right) & j > i \\ \overline{x}_i(t) & i = j \\ \min\left(\overline{x}_j(t), \overline{x}_i(t)\right) & j < i \\ \overline{x}_j(t) & \text{otherwise} \end{cases} \tag{35}$$

Now, to prove the desired result each of the cases described in (35) has to be studied: clearly when $i = j$ the result is immediately proved. When $j > i$, if $\overline{x}_j(t) < \overline{x}_i(t) \leq U_i^{k+1} \leq U_i^{k+1}$ then $x_j(t) = \overline{x}_i(t) \leq U_j^{k+1}$. Clearly the result is also proved when $x_j(t) = \overline{x}_j(t)$.
On the other hand, when $j < i$, if $\overline{x}_j(t) > \overline{x}_i(t)$ then one gets

$$x_j(t) = \overline{x}_i(t) < \overline{x}_j(t) \leq U_j^{k+1}$$

which proves the desired result. To conclude the prove it is sufficient to observe that the result is clearly still valid when $x_j(t) = \overline{x}_j(t)$. □

By symmetrical argument one can also establish that $x_i(t) \geq L_i^{k+1}$ for all values of $t \geq t_k'''$ for a certain t_k'''.
This proves equation (34) for $k + 1$, thus completing the proof of the theorem.

Note how in the case studied for every pair of states s^+, s^- such that $s^+ \geq s^-$, one has:

(1) non decreasing reward function: $r(s^+, a) \geq r(s^-, a)\ \forall a$
(2) superaddittive reward function: $r(s^+, a^+) + r(s^-, a^-) \geq r(s^+, a^-) + r(s^-, a^+)$
(3) increasing tail trandition probability: $w(s'|s^+, a) = \sum_{\zeta \geq s'} p(\zeta|s^+, a) \geq \sum_{\zeta \geq s'} p(\zeta|s^-, a) = w(s'|s^-, a)$

Using the result in Theorem 4, it is now possible to prove the convergence of Ordered Q-learning:

THEOREM 5. *Consider the Ordered Q-learning algorithm described by the update rule in* (14)*. Let* $\gamma < 1$*. Assume that, for every pair of states* s^+, s^- *such that* $s^+ \geq s^-$*, one has:*

(1) non decreasing reward function: $r(s^+, a) \geq r(s^-, a)\ \forall a$
(2) superaddittive reward function: $r(s^+, a^+) + r(s^-, a^-) \geq r(s^+, a^-) + r(s^-, a^+)$
(3) increasing tail trandition probability: $w(s'|s^+, a) = \sum_{\zeta \geq s'} p(\zeta|s^+, a) \geq \sum_{\zeta \geq s'} p(\zeta|s^-, a) = w(s'|s^-, a)$

Then q_* *is a monotone function, i.e., if* $s_1 \leq s_2$ *according to some order on the states, then* $q_*(s_1, a) \leq q_*(s_2, a)$*. Moreover,* $Q_t(s, a)$ *converges to* $q_*(s, a)$ *w.p.1. for every state* $s \in \mathcal{S}$ *and for every action* $a \in \mathcal{A}(s)$*.*

PROOF. As already observed in the proof of Lemma 1, the conditions on the reward and on the transition probabilities imply the monotonicity of the optimal value function v_*. Moreover, applying Lemma 4.7.2 in [24] (with the sequence x_j given by the tail transition probabilities and the non decreasing optimal value function) yields the following condition on the transition probability for every action a:

$$\mathbb{E}_{s' \sim p_{s^+\cdot}(a)} \left[v(s') \right] \geq \mathbb{E}_{s' \sim p_{s^-\cdot}(a)} \left[v(s') \right] \tag{36}$$

with v denoting any non decreasing function, such as the optimal value function v_*. This easily implies that even the optimal Q-function q_* is non decreasing for every action a.

To prove the convergence of the Ordered Q-learning algorithm it is then sufficient to show that all the assumptions of Theorem 4 are verified.
Consider a problem described by a Markov Decision process defined on a finite space state $\mathcal{S}$. For every states $s \in \mathcal{S}$ there is a finite set $\mathcal{A}(s)$ of possible actions and a set of transition probabilities $p_{ij}(a) \in [0, 1], i, j \in \mathcal{S}, a \in \mathcal{A}(i)$ such that $\sum_{j \in \mathcal{S}} p_{ij}(s) = 1$ for all $a \in \mathcal{A}(i)$ and $\forall i \in \mathcal{S}$.
For every state $s \in \mathcal{S}$ and every action $a \in \mathcal{A}(s)$ can be defined a random variable r_{sa} which represents the reward obtained when applying action a at state s. It can further be assumed that the variance of r_{sa} is finite for every state s and action $a \in \mathcal{A}(s)$.
Recall that the goal is to evaluate the optimal function $v_* = \max_\pi v_\pi$, defined for each state $s \in \mathcal{S}$ as

$$\begin{aligned} v_*(s) &= \max_\pi v_\pi(s) = \max_\pi \max_{a \in \mathcal{A}(s)} \mathbb{E}_{s' \sim \pi} \left[r_{sa} + \gamma v_\pi(s') \right] \\ &= \max_\pi \max_{a \in \mathcal{A}(s)} q_\pi(s, a) \\ &= \max_\pi \max_{a \in \mathcal{A}(s)} \mathbb{E}_{s' \sim p_{s\cdot}(a)} \left[r_{sa} + \gamma \max_{b \in \mathcal{A}(s')} q_\pi(s', b) \right] \end{aligned}$$

Recall that the positive constant value $\gamma \in \mathbb{R}$ is assumed to be strictly smaller than 1.
Now define the dynamic programming operator $T : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ with components T_i by letting

$$T_i(V) = \max_{a \in \mathcal{A}(i)} \left[\mathbb{E}\left[r_{ia} \right] + \gamma \sum_{j \in \mathcal{S}} p_{ij}(a) V_j \right]$$

It is well known that if $\gamma < 1$ the operator T is a contraction with respect to the norm $\|\cdot\|_\infty$ and v_* is its unique fixed point.
The Ordered Q-learning algorithm previously described is a method for computing v_* based on a reformulation of the Bellman equation $v_* = T(v_*)$.
In order to prove the convergence of the method it is sufficient to show that equation (14) has the form of (32) and it satisfies all the assumptions

of Theorem 4.
Let $F : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ be the mapping whose components F_{sa} is defined as

$$F_{sa}(Q) = r_{sa} + \gamma \mathbb{E}\left[\max_{b \in \mathcal{A}(s')} Q(s', b)\right] \tag{37}$$

where the expected value is expressed in terms of the state s' reached from the initial state s with a probability equal to $p_{s,s'}$. Note how, differently to the case studied in [29], this time the reward is deterministic given state and action.
Clearly if Q is a fixed point of F, then the vector with components $V_s = \max_{a \in \mathcal{A}(s)} Q_{sa}$ is also a fixed point of T. The definition of the mapping F allows to rewrite equation (14) as

$$\begin{cases} \overline{Q}_{t+1}(s,a) & = Q_t(s,a) + \alpha_t(s,a)\left(F_{sa}(Q_t) - Q_t(s,a) + w_t(s,a)\right) \\ Q_{t+1}(s,a) & = \Pi_s(\overline{Q}_{t+1}(s',a)) \quad \forall s' \end{cases} \tag{38}$$

where

$$w_t(s,a) = \gamma\left(\max_{b \in \mathcal{A}(s')} Q_t(s',b) - \mathbb{E}\left[\max_{b \in \mathcal{A}(s')} Q_t(s',b) \mid \mathcal{F}(t)\right]\right) \tag{39}$$

with the expected value expressed in terms of the reached state s'.
It remains to show that the Assumptions 1, 2 and 3 previously described are verified in the framework considered.
Let the filtration $\mathcal{F}(t)$ represent the history of the algorithm up to time t. This implies that (1.i), (1.ii) and (1.iii) are naturally verified. Moreover, (1.iv) is a natural consequence of the definition of the random variable w_i in (39). Finally, assumption (1.v) is verified as the conditional variance of w_i can always be bounded by $\max_{s' \in \mathcal{S}} \max_{a \in \mathcal{A}(s)} Q_t^2(s,a)$, which is itself a bounded quantity.
Assumption 2 can be imposed by choosing an appropriate definition of the random variable α describing the stepsizes in the Ordered Q-learning algorithm.
Finally, it remains to prove that Assumption 3 is verified. Proving (3.i) and (3.ii) is trivial, as both are a natural consequence of the definition of the mapping F. If $\gamma < 1$ it can also be observed how

$$|F_{sa}(Q_1) - F_{sa}(Q_2)| \leq \gamma \max_{s' \in \mathcal{S}, b \in \mathcal{A}(s')} |Q_1(s',b) - Q_2(s',b)| \quad \forall Q_1, Q_2 \in \mathbb{R}^{|\mathcal{S}|}$$

which yields that the mapping F is a contraction and verifies (3.iii).
The validity of (3.iv) is a consequence of the linearity of F and of the expected value, while to prove (3.v) it can be directly showed that $\forall v \in \mathbb{R}^n$ one has

$$v = \Pi_j(v) \Rightarrow \Pi_j(F(\Pi_j(v))) = F(\Pi_j(v))$$

$$\begin{aligned} \Pi_{ja}(F(v)) &= \left[\Pi_{ja}\left(r_{sa} + \gamma \mathbb{E}_{s' \sim p_{s\cdot}(a)}\left[\max_b v(s',b)\right]\right)\right]_{sa} \\ &= \left[\begin{cases} \max\left(r_{sa} + \mathbb{E}_{s' \sim p_{s\cdot}(a)}\left[\max_b v(s',b)\right], r_{ja} + \gamma \mathbb{E}_{s' \sim p_{j\cdot}(a)}\left[\max_b v(s',b)\right]\right) & \text{s > j} \\ r_{sa} + \gamma \mathbb{E}_{s' \sim p_{s\cdot}(a)}\left[\max_b v(s',b)\right] & \text{s = j} \\ \min\left(r_{sa} + \gamma \mathbb{E}_{s' \sim p_{s\cdot}(a)}\left[\max_b v(s',b)\right], r_{ja} + \gamma \mathbb{E}_{s' \sim p_{j\cdot}(a)}\left[\max_b v(s',b)\right]\right) & \text{s < j} \end{cases}\right]_{sa} \\ &\overset{\star}{=} \left[r_{sa} + \gamma \mathbb{E}_{s' \sim p_{s\cdot}(a)}\left[\max_b v(s',b)\right]\right]_{sa} \\ &= F(v) \end{aligned}$$

where $\star$ is a consequence of (36). Further note how the components different to s, a remain unchanged and are not reported for space's sake.
This concludes the proof. □