Journal of
Low Power Electronics and Applications

MDPI

Article

Highly Adaptive Linear Actor-Critic for Lightweight Energy-Harvesting IoT Applications

Sota Sawaguchi [1], Jean-Frédéric Christmann [1,*] and Suzanne Lesecq [2]

[1] University Grenoble Alpes, CEA, List, F-38000 Grenoble, France; sota.sawaguchi@cea.fr
[2] University Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France; suzanne.lesecq@cea.fr
* Correspondence: jean-frederic.christmann@cea.fr

Abstract: Reinforcement learning (RL) has received much attention in recent years due to its adaptability to unpredictable events such as harvested energy and workload, especially in the context of edge computing for Internet-of-Things (IoT) nodes. Due to limited resources in IoT nodes, it is difficult to achieve self-adaptability. This paper studies online reactivity issues of fixed learning rate in the linear actor-critic (LAC) algorithm for transmission duty-cycle control. We propose the LAC-AB algorithm that introduces into the LAC algorithm an adaptive learning rate called Adam for actor update to achieve better adaptability. We introduce a definition of "convergence" when quantitative analysis of convergence is performed. Simulation results using real-life one-year solar irradiance data indicate that, unlike the conventional setups of two decay rate β_1, β_2 of Adam, smaller β_1 such as 0.2–0.4 are suitable for power-failure-sensitive applications and 0.5–0.7 for latency-sensitive applications with $\beta_2 \in [0.1, 0.3]$. LAC-AB improves the time of reactivity by 68.5–88.1% in our application; it also fine-tunes the initial learning rate for the initial state and improves the time of fine-tuning by 78.2–84.3%, compared to the LAC. Besides, the number of power failures is drastically reduced to zero or a few occurrences over 300 simulations.

Keywords: reinforcement learning; energy-harvesting; Internet-of-Things; low power; machine learning; edge computing

Citation: Sawaguchi, S.; Christmann, J.-F.; Lesecq, S. Highly Adaptive Linear Actor-Critic for Lightweight Energy-Harvesting IoT Applications. *J. Low Power Electron. Appl.* **2021**, *11*, 17. https://doi.org/10.3390/jlpea11020017

Academic Editor: Alex Yakovlev

Received: 25 February 2021
Accepted: 30 March 2021
Published: 8 April 2021

1. Introduction

Energy-harvesting Internet-of-Things (EH-IoT) in wireless communications is a hot topic today, as it potentially enables perpetual operations of nodes in wireless sensor networks. Nonetheless, myriad uncertainties exist not only in the nature but also in the applications. Under such uncertainties, the power-performance trade-offs, i.e., striking the balance between energy saving and quality of service (QoS) should be addressed in an automatic way.

One popular solution is to use reinforcement learning (RL) that adapts itself to the environment at run-time with no a priori information about their changes (i.e., state transition probabilities) to produce the optimal actions. Nonetheless, the hyperparameter setting is sensitive to the environment, and, therefore, the system often fails self-adaptations. Many researchers resort to neural network-based RLs to ensure more scalability in such a situation. However, extra training data may be required while they are more compute- and memory-intensive compared to linear approximation-based RLs. Ideally, we would rather find a low-cost solution, i.e., μW-range, RL method for the upcoming μW-range IoT end devices [1]. The current trend is moving more processing from the cloud to nodes, leading to edge computing. Thus, the power manager of a node itself should be lightweight to be consistent with the low power requirement induced by the low energy budget of the node.

Taking into account the potential continuous action and state spaces (e.g., energy storage device), linear function approximations can be exploited [2,3]. The use of continuous spaces that can scale to discrete problems is also an advantage [4]. In a previous

work, we also investigated the linear actor-critic (LAC) method and employed a data buffer and an energy buffer to represent the system state. However, with such a state representation that induces power-performance trade-offs, the fixed learning rate hinders the adaptation to new situations, e.g., workload changes, since they directly impact the amplitude of the oscillation of the rewards or the variance of the gradients of the learned parameters. As such, to gain robustness to drastic changes in the gradients, we incorporate into the LAC an adaptive learning rate such as the Adam optimizer [5] (hereafter, Adam). Adam is originally designed to tackle the sparse gradient issue in training neural networks with large decay rates (e.g., $\beta_1 = 0.9$ and $\beta_2 = 0.999$). In this paper, we show that it can provide fast reactivity to the environment with smaller decay rates. We define *reactivity* as the adaptation from a previous near-optimality to a new one induced especially by non-trivial environmental changes. Further, we attempt to evaluate convergence speed in a quantitative manner, which is application-dependent and hardly conducted previously. Our contributions are summarized as follows:

1. To provide better adaptability, we combined and evaluated the LAC algorithm with Adam (LAC-A) using smaller decay factors for transmission (TX) duty-cycle optimization in an application of sensor data TX in a point-to-point network.
2. With the use of smaller decay rates in Adam, we can exclude the initialization bias correction terms to reduce the calculations, and we call the algorithm LAC-AB (LAC with Adam Biased).
3. We defined the time of convergence quantitatively based on the mean and variance information for evaluating the speed of convergence of different approaches.
4. Simulation results show that, for our application use case, smaller decay rates β_1 such as 0.2–0.4 are better for power-failure-sensitive applications and 0.5–0.7 for latency-sensitive applications with $\beta_2 \in [0.1, 0.3]$ in the LAC-AB algorithm.
5. We show that LAC-AB with such decay rates helps achieve more reactivity and stability to drastic gradient changes, such as doubled workload, and enables fine-tuning the actions to the initial state more quickly.

The rest of the paper is organized as follows. We present the state-of-the-art and clarify the differences from our approach in Section 2. Section 3 summarizes the system model used in our study. In Section 4, the detailed settings for the simulations of TX duty-cycle modulation are listed. Section 5 concisely explains the basics of RL algorithm. In Section 6, we introduce the use of Adam to deal with the reactivity issue along with the table of the whole LAC-AB algorithm. We then give the definition of time of convergence in Section 7. With respect to simulation studies, we firstly show the reactivity problem of the LAC algorithm using fixed learning rate in Section 8.1. Next, Section 8.2 demonstrates how effective the LAC-AB algorithm is compared to LAC and how we can evaluate its convergence adequately, as presented in the following subsection. Further simulation results are shown in Section 8.3 to study the combination of the two decay rates in terms of convergence speed, power failure and latency. Finally, Section 9 concludes this paper with some future work directions.

2. Related Work

Many research works in the literature of energy management in both lower-power IoT and EH-IoT exist. Roose et al. [6] proposed a hardware solution for a sense and compress application. The method proposed loads tuning parameters obtained by offline evolutionary algorithm onto statistics (sample variance)-based runtime control. It reduces the online resources, but the runtime adaptability might not be enough in the case of unexpected events that were not in the training dataset. A stepwise online adjustment is one solution presented in [7] that works with a certain discrete action space, which can be a limit for applications that require continuous spaces. The use of prediction of harvested energy is also a downside, as it is likely to produce errors that can make decision-making worse.

Energy management can be addressed through the solution of an optimization problem. For instance, the authors of [8,9] formulated the optimization problems with some

constraints and solve it using Lyapunov optimization method. However, a non-convex problem is difficult and costly to solve, with possibly multiple optima. Bhat et al. [10,11] introduced hardware solutions for their optimization problems with the help of relaxed Karush–Kuhn–Tucker (KKT) conditions or of simplex algorithm for solving their linear programming. They needed to prepare design points [11] or energy harvesting model for predictions offline [10] in their methods. Note that our approach is independent of such offline analysis. Vigorito et al. [12] formulated a linear-quadratic problem by defining the ENO-Max condition in the TX duty-cycle modulation. This method is model-free and agnostic of prior knowledge on energy harvesting. Another model-free algorithm, tabular Q-learning [13], is designed as a dedicated hardware to address adaptive power management. However, it cannot scale to large state–action spaces and continuous states and actions. Note that all the aforementioned methods do not take data queues into account.

Masadeh et al. [14] presented an actor–critic RL method for TX output power control in energy-harvesting communications system. Their actor learns the parameters for the mean and standard deviation of a normal distribution, while the critic is constructed by a two-layer neural network, which can be costly for resource-constrained devices. Likewise, in [15–17], three layer neural networks are employed, which is again computation- and memory-intensive. For the purpose of low-cost implementation, linear function approximation is adopted in [2,18]. However, these papers do not discuss the reactivity of their algorithm to environmental changes, or covariate shift, i.e., re-optimization to a new state. All of the above works use fixed learning rate, which cannot guarantee enough reactivity. By contrast, we aim at devising a lightweight reactive control algorithm.

The authors of [19–21] adopted deep learning-based RL (DRL) solutions with Adam. The effect of this adaptive algorithm, however, is to overcome the sparse gradient issue that frequents neural networks. To this end, larger decay rates in Adam are employed, which attains stable and fast convergence in training neural networks. Nonetheless, such setups of decay rates essentially lead to slow adaptation to new situations since they put less weight on recent changes and retain longer past gradient information. Besides, initialization bias correction is required.

Differently from those existing methods, we propose applying Adam to a low-cost LAC algorithm for fast adaptability. That is, thanks to the use of linear approximations, we can perform Adam with smaller decay rates such as $\beta_1, \beta_2 \in [0.1, 0.7]$, which can make our RL more reactive. Such small decay rates are also suitable, in that LAC is fully dependent on the latest action to learn, i.e., it is an on-policy RL. These choices of values are further elucidated by the use of exponentially weighted moving average (EWMA) in the literature of online workload change detection [22–24]. Comparisons among the above state-of-the-art solutions and our approach are summarized in Table 1.

Table 1. Comparison of each algorithmic representations of the actor-critic method.

Paper	Method	SoB	SoC	Harvester	Action	Neural?	LR
[6]	Online statistics & offline optimization	-	Finite	-	Sense & Compress setups	No	NA
[7]	Prediction & online stepwise adjustment	-	Finite	Solar	Duty-cycle, TX power	No	Fixed
[8]	Lyapunov optimization	-	Finite	Solar	TX modulation	No	NA
[9]	Lyapunov optimization	-	Finite	Solar	TX power	No	NA
[10]	Online KKT & prediction	-	Finite	Solar	Duty-cycle	No	Fixed
[11]	Simplex algorithm with prediction and offline data	-	Finite	Solar	Active time (Duty-cycle), accuracy	No	NA
[13]	Q-learning	-	Finite	-	Suspension mode selection	No	Fixed
[14]	Actor-Critic	Infinite	Finite	Solar	TX power	3 layers (3-10-5-1)	Fixed

Table 1. *Cont.*

Paper	Method	SoB	SoC	Harvester	Action	Neural?	LR
[12]	Linear-Quadratic Regulator	-	Finite	Solar	Duty-cycle	No	Fixed
[15]	DRL	-	Finite	Solar	Duty-cycle	3 layers (4-64-64-2)	Fixed
[16]	Asynchronous Advantage Actor-Critic	Finite	Finite	Uniform distribution	Duty-cycle, TX power (source and relay)	3 layers (5-300-200-3(1))	Fixed
[17]	Deep RL	Infinite	Finite	Uniform distribution	TX modulation (=TX power)	3 layers (3-10-10-1)	Fixed or decaying
[2]	Linear Actor-Critic	-	Finite	Solar, wind	Packet rate	No (Linear function)	Fixed
[18]	Multi-Agent Actor-Critic	Finite	Finite	Solar	Duty-cycle, TX power	No (Linear function)	Fixed
[19]	Double Deep Q-Network	-	Finite	Two-state Markov process	Uplink scheduling policy	Yes (No details)	Adam (less adaptive)
[20]	Multi-Agent Double Deep Q-Network	-	-	-	Base station and channel selections	4-layers (50-64-32-32-26(30))	Adam (less adaptive)
[21]	Deep Deterministic Policy Gradient	-	Finite	Solar	Energy budget	Actor: 3(4)-60-30-1, Critic: 3(4)-60-60-60-60-60-1	Adam (less adaptive)
This paper	Linear Actor-Critic	Finite	Finite	Solar	Duty-cycle	Linear function	Adam (highly adaptive)

3. System Model

Adaptive control in EH-IoT has already been addressed in previous works (see, e.g., [18]). In the current paper, we study the exact same system. However, the system is now described to make the paper self-content.

3.1. Energy Harvesting and State-of-Charge Model

Several types of energy harvesting sources exist, e.g., solar, wind, vibration, thermal and piezoelectric. Here, we focus on solar energy-harvesting. The scavenged power P_t^{harv} is formally calculated by:

$$P_t^{harv} = \eta \cdot A \cdot TF \cdot I_t \quad (1)$$

with the solar irradiance $I_t[\mathrm{W/m^2}]$, the size of the photovoltaic (PV) cell $A[\mathrm{m^2}]$, the conversion efficiency (rate) η, and the tracking factor (TF) of maximum power point tracking. In the present work, we employ the harvest–use–store scheme [25,26].

A supercapacitor is considered an optimal solution as an energy storage in energy-harvesting applications because of the well-balanced trade-offs among physical lifetime, energy density, maximum charge cycles, and self-discharging rate [27]. Let E^{max} and E^{fail} denote the maximum and minimum (i.e., failing-threshold) energy levels. The state-of-charge (SoC) of the residual energy E_t is represented as:

$$\phi_t^{SoC} = \frac{E_t - E^{fail}}{E^{max} - E^{fail}} \quad (2)$$

Note that a severe self-discharging is a known issue in supercapacitors. Indeed, the self-discharge rate τ during time Δt can be up to 20% per day [27]. With its capacity C and voltage level V_t^{sc}, the leak power P^{leak} due to self-discharging is given by:

$$P^{leak} = \frac{1}{2\Delta t} C(1-\tau^2) V_{sc}^2 \quad (3)$$

In our simulations, Δt is equal to 1 min, and the discharging rate is simplified as $\tau = 0.8^{\frac{1}{1440}}$, which stands for 20% per day.

3.2. Application Data and State-of-Buffer model

In this paper, an embedded sensor is assumed to generate data that are stored into the TX buffer. The sensor type is not considered here, even if the sensor technology and type of information sensed have a strong influence on the power consumption, and thus on the required harvested energy. Data acquisition can be periodic, aperiodic, or completely random. We assume here that the arrival of data to the TX buffer is generated based on the Poisson distribution with the average of λ [pkts/min] [28].

We consider a TX buffer with maximum capacity B^{max}. The buffer stores data that are newly generated or that require retransmission because of sending issue. The State-of-Buffer (SoB) ϕ_t^{SoB} of the current buffer level B_t is defined as:

$$\phi_t^{SoB} = \frac{B_t}{B^{max}} \tag{4}$$

3.3. Power Consumption and Transmission Model

Depending on the wireless protocol, different controllable variables exist such as the duty-cycle, the output power, the modulation, the spreading factor, the bandwidth, and the cyclic redundancy check. In this paper, we assume a continuous duty-cycle control of a CC2500 transceiver module whose output power is fixed to +1 dBm that consumes 21.5 mA according to the data sheet [29]. Considering the nA- and mA-order of current in deep-sleep mode and in active mode, the impact of TX duty-cycle control is much larger than that of TX output power control.

We assume the cycle period T^{cycle} = 1 min that consists of the active time T^{active} and the sleep time T^{sleep}. The TX duty-cycle D_t is defined as the ratio of T^{active} to T^{cycle}. Therefore, we have $T^{active} = D_t \cdot T^{cycle}$. Taking into account η^{act} and η^{slp} that are the efficiency of the DC-DC regulator in active and sleep mode [7], respectively, the TX power consumption during a cycle P^{cycle} is expressed as:

$$P^{cycle} = \frac{P^{active}}{\eta^{actice}} \cdot D + \frac{P^{sleep}}{\eta^{sleep}} \cdot (1 - D) \tag{5}$$

where P^{active} is the power consumption in the active mode. T^{active} is comprised of the sum of time-on-air of frame TX and acknowledgements (RX) [30]. Note that the transceiver wake-up time overhead is therefore ignored in this paper. T^{ack} denotes the time required to receive an acknowledgement packet: it is assumed to be equal to the acknowledgement frame's time-on-air. Under these assumptions, P^{active} is obtained by:

$$P_t^{active} = P^{tx} \cdot (1 - \frac{T^{ack}}{T^{active}}) + P^{rx} \cdot \frac{T^{ack}}{T^{active}} \tag{6}$$

where P^{tx} and P^{rx} are the power consumption during packet transmission and acknowledgement reception, respectively. While the overall power consumption of an IoT node typically comprises sensing, processing and communication power, only the last one was considered in our simulations. In addition, this work neglects the power consumption overhead of the proposed actor–critic-based controller. Note that a combined path-loss and shadowing model [31] in outdoor environment is assumed, and the packet error ratio was constantly zero throughout the whole simulations.

4. Application Scenario

We consider the case of TX duty-cycle optimization in an energy-harvesting IoT sensor end-node communicating with a sink node. In such a situation, the space of application

scenarios is quite large. The common features of the application scenario assumed in our study are described as follows:

- The control update interval (CUI) is $T^{cui} = 30$ min.
- For the PV cell model, we set the cell area A, conversion efficiency η, and tracking factor TF as 2.5 cm^2, 10%, and 96.3%, respectively [7]. This choice is consistent with an off-the-shelf solar cell that can harvest power from μW to mW per cm^2, depending on the lighting condition [32]. Note that we use the real-life solar irradiance data provided by Oak Ridge National Laboratory [33].
- The self-discharge of a supercapacitor whose capacitor size is 1F is considered 20% per day (detailed in Section 3.1). The harvest–use–store scheme is adopted [25,26] to provide high energy efficiency.
- The wireless link quality is under the influence of path-loss and shadowing.
- The workload follows the Poisson distribution. The average rate doubles after the first six months (where the algorithm is put through the test of fast adaptability/reactivity). More precisely, the system receives the average of 1.0 pkt/min for the first six months, and it impulsively becomes twice (2.0 pkt/min) afterwards.

Figure 1 illustrates this application scenario. For the solar irradiance data, we use three kinds of datasets:

1. EHD1: Non-processed real-life one-year data from 1 June 2018 to 31 May 2019
2. EHD2: Real-life one-year data made by stacking 365 one day (1 December 2018) worth of solar irradiance data
3. EHD3: Real-life one-year data made by stacking 365 one day (1 June 2018) worth of solar irradiance data

In each simulation study presented below, we specify which dataset was used by the notation, EHD1, EHD2, or EHD3. The uses of EHD2 and EHD3 is justified in Sections 7 and 8.2.

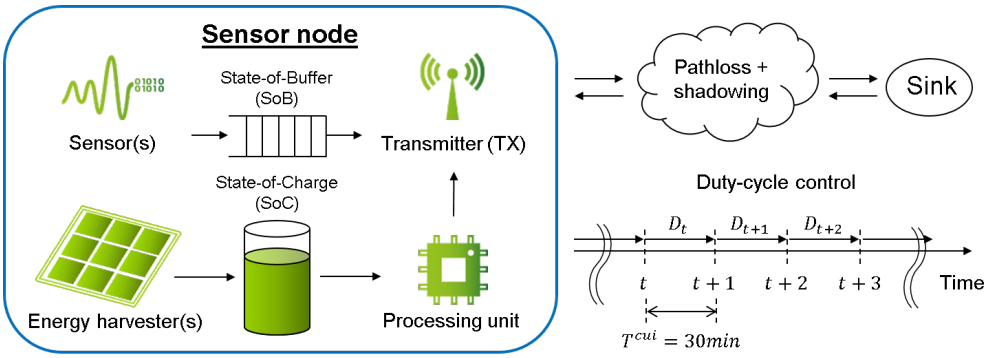

Figure 1. Overview of the application scenario.

5. Reinforcement Learning

In reinforcement learning (RL) [34], a decision maker and its surroundings exist. The former is called the agent, while the latter the environment. The agent learns to optimize its actions by interacting with the environment. As opposed to dynamic programming where the agent perfectly knows the environment's behavior as a model, no a priori information of the environment is given in RL, which makes it practically useful in real-life, as the environment is often too complex and transient to accurately build its model.

We denote the agent's action and the environment's state by a_t and s_t, respectively. With the premise of Markov decision process, the agent decides, based on its policy $\pi(a_t|s_t)$, the action a_t that influences the environment and its state changes to s_{t+1}. As a result, the agent receives from it a reward $r_{t+1} = f(s_t, a_t, s_{t+1})$ that represents how good the action a_t was at the state s_t. The goal of RL is therefore defined as maximizing the expected total rewards from the present to the future. The future discount factor γ is introduced

in general to put more weight in the near future. The expected total future-discounted reward, more frequently called the value function V_π at state s_t, is then defined as:

$$V_\pi(s_t) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} | s_t] \quad (7)$$

Nonetheless, this value is practically impossible to calculate, since the future is unforeseeable. Hence, we need to accurately estimate this value.

At this point, the RL's goal can be re-defined and is comprised of two purposes:

1. Find the optimal policy π^*.
2. Determine the estimate of the value function v_π under a certain policy;

Note that the value function v_* under the optimal policy π^* is defined as $\forall s \in \mathcal{S}, v_* = \max_\pi v_\pi(s)$ where $\mathcal{S}$ is the state space. We must seek these two under the following two conditions: (1) The reward for each state–action pair is unknown. (2) The state transition probability is unknown. Getting samples from the interactions with the environment helps the agent capture information about the reward and state transition probability to learn the policy and the value function. In actor–critic RL, the actor is in charge of the first purpose and the critic the second one. Each goal can be achieved by policy gradient theorem and TD(λ) algorithm (refer to [2]), which we opt for in this paper, or by the use of neural networks.

6. Algorithm: LAC-AB

The LAC-AB algorithm is based on the LAC that was originally devised by Ait Aoudia et al. [2]. One difference is that we apply Adam to the LAC to address the reactivity problem caused by fixed learning rate, which we discuss in Section 8.1. Thus far, Adam is typically used for sparse gradient issue in use of neural networks. The decay factors β_1 and β_2 are therefore set as 0.9 and 0.999 (which we call PA (prior art) setting below), respectively. The use of linear functions breaks free from such an issue, enabling us to set the decay factors smaller, such as 0.1–0.7 as generally adopted in the literature of workload change detection [22–24], so as to achieve faster adaptation. The reason is two-fold:

- Smaller values lead to more weight on recent changes, i.e., faster online adaptation.
- The gradient variance can become too large and yet carries an important information for parameter updates that can be lost with larger values of β_1 and β_2.

Further, with such smaller rates, we can reduce some computations by excluding the bias correction terms (refer to [5]) that are accounted in Adam. Hence, the LAC-AB algorithm is a LAC algorithm using Adam with no initialization Bias correction terms.

Figure 2 illustrates the structure of the LAC-AB and Algorithm 1 shows its whole algorithm concerning our application example. However, it is trivial to transfer this algorithm to other application use cases. The state is composed of the State-of-Buffer (SoB) and State-of-Charge (SoC), which take into account the "quantity" of the in and out of data and energy. The value function is assumed to be linearly proportional to the multiplication of $1 - \phi_{SoB}$ and ϕ_{SoC} (Line 6), which indicates that the value of the state is higher when less SoB and more SoC are confirmed. Adam is employed with no initialization bias correction terms in actor update (Lines 11–13) with adaptation-aware setups. The linear relationship is also assumed between (mean) action value and the multiplication of ϕ_{SoB} and ϕ_{SoC} (Line 14), which means that smaller action values (i.e., less performance) is enough when the SoB level is less, and higher values can be provided when the SoC level is higher. The final action is generated based on the Gaussian distribution (Line 15) to guarantee explorations and to find an optimal action. Note that the algorithm mostly contains multiply and add operations; only two divisions, one squared root operations and a Gaussian random number generator are used.

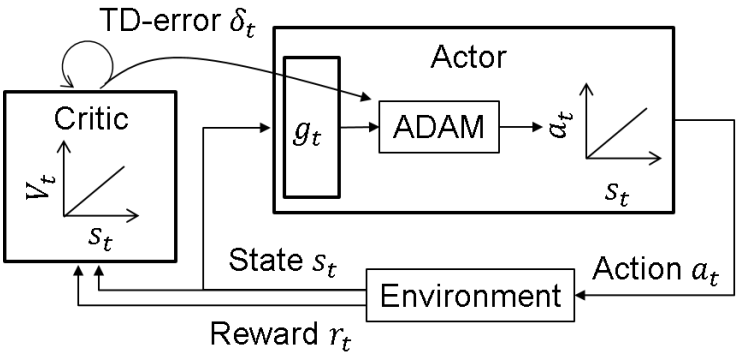

Figure 2. Overview of the actor and critic in LAC-AB algorithm.

Algorithm 1 LAC-AB: LAC Algorithm Using Adam with No Initialization Bias Correction

Require:

/* **Inputs** */

- State-of-Buffer ψ_{t+1}^{SoB} and State-of-Charge ψ_{t+1}^{SoC}

/* **Hyper-parameters for Actor-Critic** */

- Learning rates β and α for Actor and Critic, respectively
- Discount factor $\gamma \in [0, 1]$ for past reward R_{t+1}
- Recency weight $\lambda \in [0, 1]$ in the TD(λ) algorithm
- Exploration space σ (standard deviation for the Gaussian policy)
- Decay rates $\beta_1 \in [0, 1]$ and $\beta_2 \in [0, 1]$ for EWMA in Adam
- ϵ to avoid division by infinitesimally small values in Adam

Ensure:

- Action $a_t \in [a^{min}, a^{max}]$
- Actor and Critic parameter ψ_t and θ_t

Initialize at time $t = 0$:

- Empty data buffer $\psi_0^{SoB} = 0$ and fully-charged energy buffer $\psi_0^{SoC} = 1$
- ψ_0 and θ_0 are random numbers ranging $[0, 1]$

for each $t \in [0, \infty]$ **do**

/* **Observe the current state** */

$R_{t+1} = (1 - \phi_{t+1}^{SoB}) \cdot \phi_{t+1}^{SoC}$ ▷ For minimizing SoB and maximizing SoC

$V_t = \theta_t \cdot (1 - \phi_t^{SoB}) \cdot \phi_t^{SoC}$ ▷ Less SoB and more SoC are better states (better values)

$\hat{V_{t+1}} = \theta_t \cdot (1.0 - \phi_{t+1}^{SoB}) \cdot \phi_{t+1}^{SoC}$

/* **TD-error for Actor-Critic** */

$\delta_{t+1}^{TD} = R(t+1) + \gamma \hat{V_{t+1}} - V_t$ ▷ Advantage function: $A(s,a) = Q(s,a) - V(s)$ ($Q(s,a)$: state-action value function)

/* **Critic: TD(λ) algorithm** */

$z_{t+1} = \gamma \lambda z_t + (1 - \phi_{t+1}^{SoB}) \cdot \phi_{t+1}^{SoC}$ ▷ Calculate the eligibility trace z_{t+1}

$\theta_{t+1} = \theta_t + \alpha \delta_{t+1} z_{t+1}$ ▷ Update the critic parameter

/* **Actor: Policy gradient theorem using Adam with no initialization bias corrections** */

$g_{t+1} = \delta_{t+1} \cdot \frac{a_t - \mu_t}{\sigma^2} \cdot \phi_t^{SoB} \cdot \phi_t^{SoC}$

$m_{t+1} = \beta_1 \cdot m_t + (1 - \beta_1) \cdot g_{t+1}$ ▷ Estimate the first-order moment

$v_{t+1} = \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_{t+1}^2$ ▷ Estimate the second-order moment

$\psi_{t+1} = \psi_t + \beta_{t+1} \cdot \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon}$ ▷ Update the actor parameter

/* **Next TX duty-cycle selection** */

$\mu_{t+1} = \psi_{t+1} \cdot \phi_{t+1}^{SoB} \cdot \phi_{t+1}^{SoC}$ ▷ Less SoB, smaller action values; More SoC, higher action values

$\mu_{t+1} \leftarrow$ Clamp μ_{t+1} to $[a^{min}, a^{max}]$

$a_{t+1} \sim \mathcal{N}(\mu_{t+1}, \sigma)$ ▷ Gaussian policy for action generation

$a_{t+1} \leftarrow$ Clamp a_{t+1} to $[a^{min}, a^{max}]$

Return a_{t+1}

end for each

7. Definition of Convergence

The convergences of TD(λ) algorithm in linear function approximations and policy gradient theorem are proven, but it is tricky and often application-specific to determine when they have converged. For instance, in [35], the authors defined the time of convergence in an episodic task as when the returns of the first 10 consecutive episodes are all within 5% of the average of the final 150 episodes. Meanwhile, other work have compared several methods and/or setups and analyzed the convergence only by the visual qualitative measurements [17,20].

In this study, we analyzed two kinds of convergence: the convergence for the initial state (i.e., for the first six months) and the convergence for a new state (i.e., for the last six months). We call the time of those convergence the time of fine-tuning (ToF) and the time of reactivity (ToR), respectively. Since the optimization process may greatly differ for each simulation due to the Gaussian policy as well as other stochastic factors, such as workload, scavenged energy, and wireless conditions, and the variance of the trace also appears to converge, as shown in Figure 3, the convergence analysis was conducted for *the average trace* of a concerned variable. Figure 4 shows the idea of how to analyze the convergence using two time-windows. We augment the approach used in [35] and define them as follows:

1. All the mean values (e.g., actor parameter values ψ_t) taken over all the simulations at the same time points in a x-day sweeping window are all within 5% error band of the average of all the mean values in the last x-day window under almost the same state (e.g., under the same workload scenario in our test study here).
2. The variances of the mean values taken over all the simulations at the same time points are confirmed to be not different, i.e., the homogeneity of variance is tested and confirmed by means of Levene's test [36], more precisely Brown–Forsythe [37] test, with the confidence interval of y% (note that we cannot say "the same" mathematically).

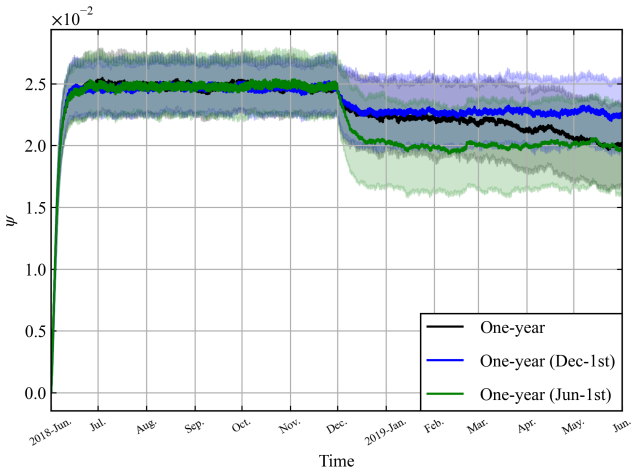

Figure 3. Transitions of actor parameter ψ using LAC-AB ($\beta_1, \beta_2 = 0.4$) for EHD1, EHD2, and EHD3.

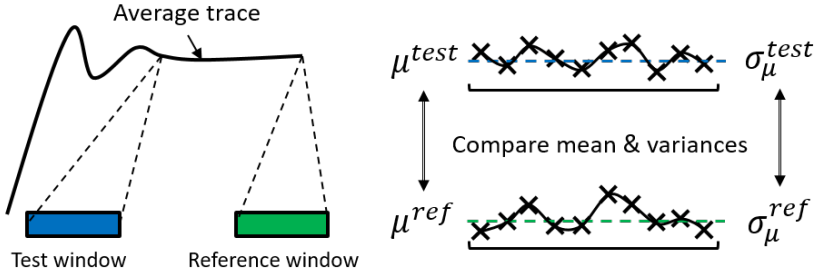

Figure 4. Overview of convergence analysis.

Note that we run the sweeping window from the first day towards the end with step size of a time point, which is equal to T^{cui} = 30 min, until the convergence is admitted. The reason for evaluating the homogeneity of variance is that the sequences of the mean values may be different in terms of variance between the two windows, which can defy the convergence. The use of Brown–Forsythe test is because we cannot expect the mean values to follow normal or symmetric distribution, and it is not as sensitive to violations of the normality assumption as alternative tests such as Hartley's F_{max} test [38]. Further, we observe the daily (e.g., weather) and seasonal effects that also heavily impact the optimization (see Section 8.2); therefore, we crafted an artificial one-year trace data by stacking one-day trace data upon one another 365 times. The use of such data trace is considered acceptable because the RL algorithm adopted is on-policy and is incapable of learning features or correlations between the current day and any past days. In other words, the algorithm is agnostic of time-independent information.

We set $x = 5$ throughout this paper, which corresponds to, for example, 240 data samples in a window in the case of $T^{cui} = 30$ min, to ensure that the convergence is not merely temporal, and $y = 5$. Note that this quantitative way of evaluating the convergence is application-specific and applicable to our case. Although this is an attempt to provide the convergence time and its comparison, the performance itself may be sufficient at an earlier point for the practical use.

8. Simulation Results

The models shown in Section 3 and adaptive decision-making algorithms proposed in Section 6 were all coded in C++ and different simulation studies were conducted. Note that we used the real-life solar irradiance datasets provided by Oak Ridge National Laboratory [33]. We now present the results of these simulations in the following sub-sections.

8.1. Divergence and Reactivity Problem

Our application scenario involves the TX duty-cycle optimization, which is in essence the same as the one addressed by the LAC algorithm in [2]. Their reward function represents the multiplication of SoC and packet rate that is equivalent to the duty-cycle as a performance factor. The evolutions of ψ and TD-errors over a year obtained by simply applying their approach are illustrated in Figure 5. Such a representation of the reward function may entice the agent to greedily increase the duty-cycle for more rewards, which ends up causing power failures. More precisely, a sufficient energy reserve during the daytime helps such an increase, while, at some point, the blown-up duty-cycle cannot quickly be reduced anymore by the time the whole energy runs out due to a constant exploration range. In this case, the upper bound of the TX duty-cycle is way too large for the system as well as the application; in other words, no proper upper bound for braking duty-cycle explosion exists in the existing method. Note that, with the use of either Adam or a much smaller learning rate for the case of five-year dataset, the same divergence was observed. To resolve this issue, we suggest the use of SoB both as a performance and an upper bound index.

RL algorithms are supposedly self-adaptive to environmental changes. Such changes can be characterized by their periodicity and speed. To counteract them, the system can modulate its action(s) and/or the CUI. We fix the CUI as 30 min and consider an impulsive workload change (from 1.0 pkt/min to 2.0 after the first six months) to shed light on the reactivity issue of fixed learning rate. The set of hyperparameters for each algorithm is listed in Table 2. Note that these values are used for all simulation study throughout the paper. The traces of ψ_t values and TD-errors averaged over 90 successful runs are illustrated in Figure 6. The power failures (in total 10 out of 10 unsuccessful cases, since one power failure was observed in each of those cases) are highlighted as red crosses at each time point, just showing when they occurred. As the CUI is fixed, the gradient (or its variance) becomes larger after 1 December, i.e., the workload doubles up as the SoB term is included in the reward, which leads to larger actor updates for ψ due to fixed learning

rate. TD-errors also fluctuate in a certain range without divergence. This is because the SoB index provides the upper bound for how much performance is necessary. Note that we observed ψ values since the convergence speed is slower than TD(λ) algorithm and the TD-error follows the same trend as ψ values. Hence, in the case of the fixed CUI, the learning rate needs to be adapted at run-time. In the next subsection, we introduce a widely-used adaptive learning method, i.e., Adam, to the LAC algorithm by arranging the decay rates and show its effectiveness for reactivity alongside with fine-tuning adaptation to the initial state.

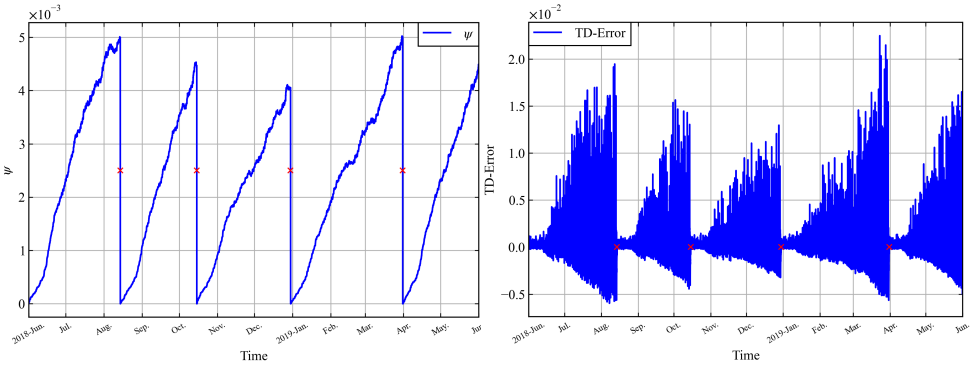

Figure 5. Divergences of ψ (**left**) and TD-errors (**right**) over one year with the state-of-the-art LAC method (red crosses indicate the power failure points).

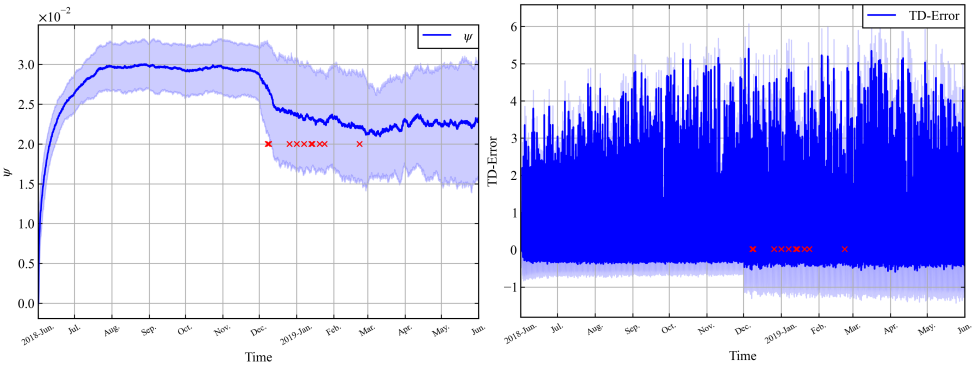

Figure 6. Transitions of ψ (**left**) and TD-errors (**right**) in case of the LAC algorithm using fixed learning rate with the SoB as a performance upper bound (red crosses represent when power failure(s) occurred).

8.2. Effectiveness and Convergence of LAC-AB

As observed in the previous section, the existing LAC method is limited in adaptability due to fixed learning rate. Thus, we introduce Adam to LAC to deal with this problem and call it LAC-A algorithm. In addition, as explained in Section 6, it would be possible to exclude the use of initialization bias correction terms in Adam, which leads to the LAC-AB algorithm. In this section, we use 300 simulation results and compare these approaches with different decay rate setups in terms of convergence, latency, and power failures. Note that the latency in this work is defined as the time from when a packet arrives in the data buffer until when it is successfully transmitted to the sink node.

First, we use the EHD1 dataset and make a comparison among LAC-A with PA setting, LAC-A, and LAC-AB with different decay rates, through which we show more suitable setups for LAC-A/LAC-AB and the improvement in terms of latency, reactivity, and the number of power failures. The hyperparameters for this analysis are listed in Table 2. The transitions of ψ_t for the three modes (LAC with PA setting, LAC-A, and LAC-AB) with $\beta_1 = \beta_2 = 0.4$ are depicted in Figure 7. The use of Adam rescales the gradient between before and after the workload change; however, this does not necessarily help avoid the power failures. While the number of failures increased to 28 times in PA setting compared

to 10 in the case of LAC (fixed learning rate) (Figure 6), the system yielded no failures with 0.4 for both decay factors. This suggests that too large decay rates may cancel the gradient direction that is calculated as a result of Gaussian policy. Moreover, we observe that smaller decay rates in Adam help achieve fine-tuning of the learning rate, faster reactivity and even less gradient variation, since they allow for faster online tracking of changes in gradients, the rewards, or even the SoB and the SoC. The decay rate 0.4, for example, is relatively small and may permit no initialization bias corrections. Table 3 summarizes the latency and the number of power failures for each case above. The latency is evaluated before and after the workload change. As expected, the latency values of both LAC-A and LAC-AB with $\beta_1 = \beta_2 = 0.4$ are the same in the order of 10^{-2} (e.g., 3.52 min), except for the standard deviation of the latency in the first six months, whose error is only 0.03 min (0.3%). As such, for the rest of the paper, we leverage and focus on the LAC-AB algorithm for achieving faster adaptations while reducing the computation and memory footprint.

Table 2. Hyperparameters for actor–critic + Adam.

Algorithm	α	β	γ	σ	λ	ϵ
LAC	0.1	2.0×10^{-6}	0.9	5.0×10^{-4}	0.9	1.0×10^{-6}
LAC-A	0.1	3.0×10^{-4}	0.9	5.0×10^{-4}	0.9	1.0×10^{-6}
LAC-AB	0.1	3.0×10^{-4}	0.9	5.0×10^{-4}	0.9	1.0×10^{-6}

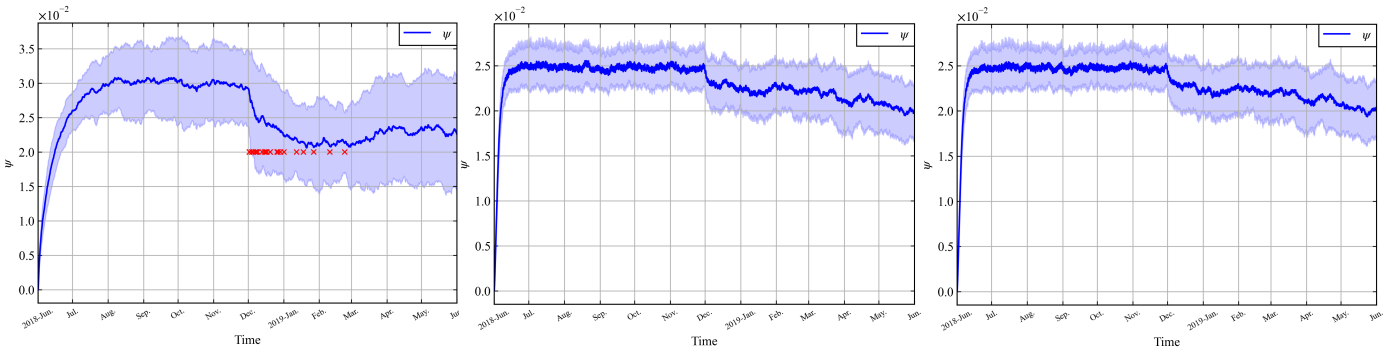

Figure 7. Transition of actor parameter ψ using Adam and EHD1 dataset: (**left**) LAC-A ($\beta_1 = 0.9$, $\beta_2 = 0.999$); (**middle**) LAC-A ($\beta_1 = \beta_2 = 0.4$); and (**right**) LAC-AB ($\beta_1 = \beta_2 = 0.4$).

Table 3. Latency (min) and power failures for the three different algorithms and setups.

Algorithm		LAC-A		LAC-AB
β_1/β_2		0.9/0.999	0.4/0.4	0.4/0.4
Latency (Mean/Std)	First 6 months	3.40/10.54	3.52/11.11	3.52/11.14
	Last 6 months	6.21/11.07	6.06/10.76	6.06/10.76
# of power failures/# of failed simulations		28/28	0/0	0/0

Nonetheless, in the case of the latter two modes, the ψ traces of the last half a year constantly decrease, which makes it harder to judge the convergence, whereas that of the first half remains almost constant. We use the EHD2 and EHD3 datasets where the seasonal and weather changes are removed and the randomness of real-life solar irradiance is still retained. The analysis with the use of those datasets can be supported by the fact that the LAC algorithm is an on-policy RL and no other algorithm for capturing correlations of any two different days is used. We obtained the traces of ψ for EHD2 and EHD3 depicted in Figure 3 as well as that for EHD1. We can claim *qualitatively* that the ψ of LAC-A using $\beta_1 = \beta_2 = 0.4$ for EHD1 dataset converges after the workload change despite its constant decrease, because the value converges to 2.25×10^{-2} for EHD2 (corresponding to

December) and to 2.0×10^{-2} for EHD3 (corresponding to June) and the trace in-between can be explained by interpolation. This observation explains that the difference in the ratio of variations of the SoB and the SoC at each control interval gives rise to different optimization process. It can also be seen that the seasonal and weather-induced fluctuations in energy-harvesting may blur the optimization process. Hence, the analysis of ToF and ToR was conducted using EHD2 and EHD3. This way, we can also infer the range of convergence time for whenever the state changes.

8.3. Decay Rates Study for LAC-AB

For β_1 and β_2, various combinations can be made, and, therefore, we need to grasp the tendency of which ones work better. To this end, we used EHD1 to evaluate the number of power failures and EHD3 to assess the convergence speed. We conducted 300 simulations for each set based on $\beta_1, \beta_2 \in [0.1, 0.7]$ with a 0.1 step, because no use of bias correction terms implies the use of lower values. With respect to ToF/ToR analysis, we used the average values over 300 simulations, since the randomness in each uncertainty such as Gaussian policy, workload, harvested energy, and wireless link quality still makes the optimization process unclear and yet can be mitigated by taking the average. Afterwards, we applied the ToF and ToR definition in Section 7 to these results. Note that the ToF and ToR can be measured also with EHD2 dataset, but the outcome is quite similar to EHD3's, and, therefore, removed for brevity. The results are depicted in Figure 8. The number of power failures (Figure 8, left) tends to decrease as β_1 value goes down to $[0.1, 0.4]$ with at most one failure. With $\beta_1 \in [0.1, 0.4]$, both ToF and ToR become faster when using lower $\beta_2 \in [0.1, 0.3]$. This can be explained by the fact that the sudden change in gradients, i.e., SoB and/or SoC will be mitigated by the quick rescaling of variance, i.e., smaller decay rate such as $\beta_2 = 0.1$. By contrast, opting for $\beta_2 = 0.1$ and even 0.2 severely deteriorates the performance of energy management when using larger β_1 such as 0.6–0.7. For this range of β_1, we obtain better outcome in both power failure and convergence speed with $\beta_2 \in [0.3, 0.4]$. All the above results considered, the choices must be made by considering the trade-offs between power failure and convergence speed.

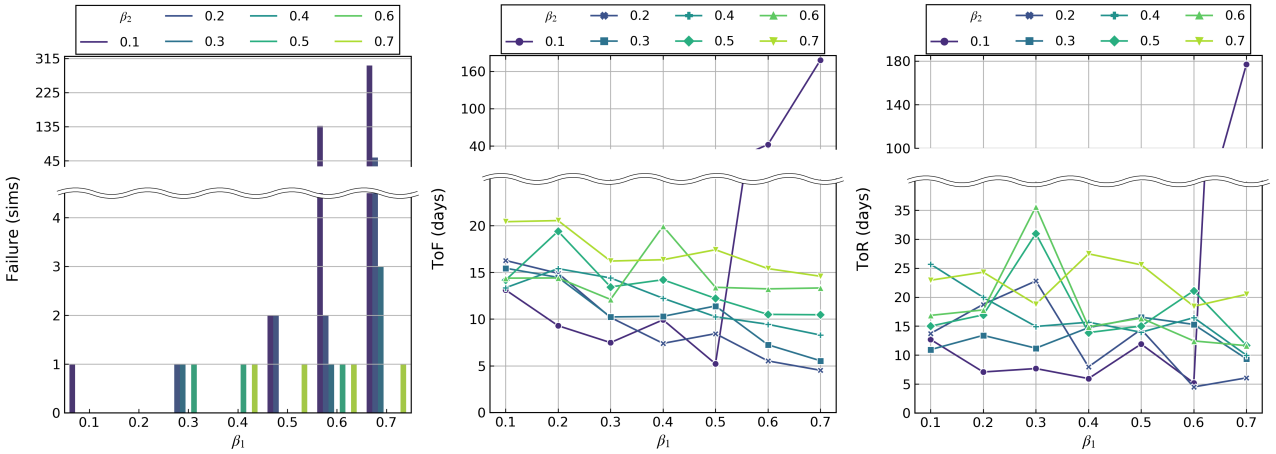

Figure 8. Power failure and Convergence speed analysis of LAC-AB for different sets of (β_1, β_2) using EHD1 and EHD3 datasets, respectively: (**left**) number of power failures; (**middle**) ToF; and (**right**) ToR.

The analysis on the latency in the two different workload periods was also conducted, as illustrated in Figures 9 and 10. As can be seen, the results show the opposite trends. For the first six months, the latency has a tendency to decline as β_1 increases, ranging from 3.4 to 3.6 min and from 10 to 12 min for the mean and standard deviation, respectively, except for $\beta_1 \in [0.5, 0.7]$ with $\beta_2 \in [0.1, 0.2]$. By contrast, the rising trends are observed during the period of doubled workload, where the mean and standard deviation of latency fall between 6.0 and 6.2 min and between 10.65 and 11 min, respectively. Again, the exceptions occur with $\beta_1 \in [0.5, 0.7]$ with $\beta_2 \in [0.1, 0.2]$. These phenomena are explained by the

effectiveness of high adaptability to each situation; higher adaptability by smaller decay rates is more suitable when the state such as SoB varies greatly (e.g., during the last six month), and less when the state is relatively constant (e.g., during the first six month).

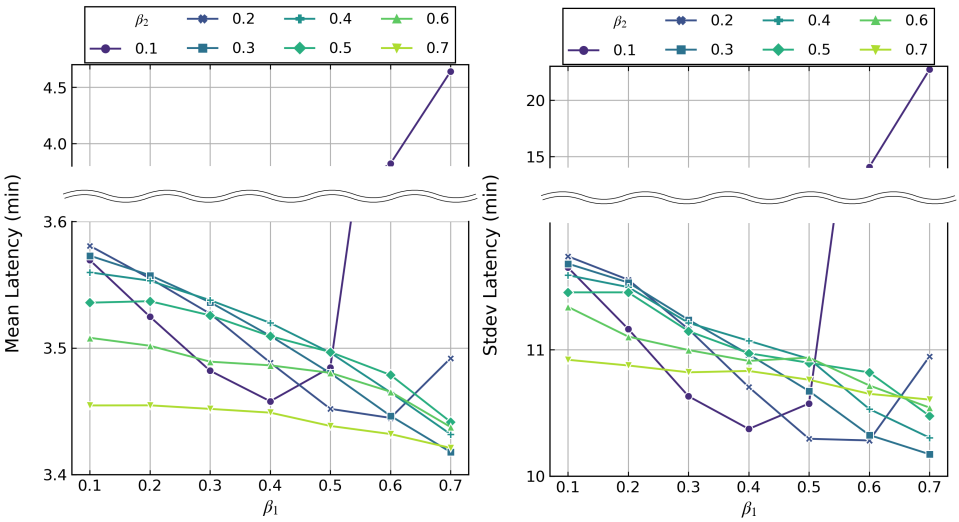

Figure 9. Latency analysis of LAC-AB during the first sxi months for different sets of (β_1, β_2) using EHD1 dataset: (**left**) mean; and (**right**) standard deviation.

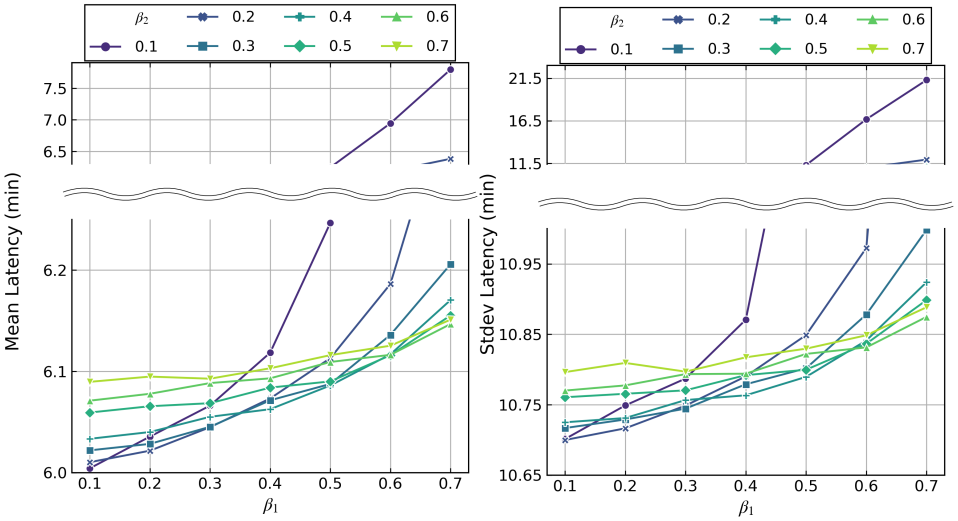

Figure 10. Latency analysis of LAC-AB during the last six months for different sets of (β_1, β_2) using EHD1 dataset: (**left**) mean; and (**right**) standard deviation.

The experiments thus far led us to choose one of the best values of β_2 for each $\beta_1 \in [0.1, 0.7]$ for further investigations: $\beta_2 = 0.1$ for $\beta_1 \in [0.1, 0.4]$, $\beta_2 = 0.2$ for $\beta_1 \in [0.5, 0.6]$, and $\beta_2 = 0.3$ for $\beta_1 = 0.7$. We ran 300 simulations and obtained ToF and ToR for both EHD2 and EHD3 dataset, since we can expect from the results in Figure 7 that the convergence speed may differ depending on the season. Figure 11 shows the ToF and ToR for the chosen sets of decay rates with the baseline values of those of the LAC algorithm using fixed learning rate. Obviously, the convergence speed improved compared to the case of LAC. With $\beta_1/\beta_2 = 0.3/0.1$, for instance, the ToF and ToR reduced to 7.15 and 0.02 days for EHD2 (or 1 December dataset) and 7.48 and 7.67 for EHD3 (or 1 June dataset), respectively, compared to 83.40 and 47.52 days for EHD2 and 60.17 and 45.77 days for EHD3 in using fixed learning rate in LAC algorithm. Across all the combinations, the worst ToF and ToR are still 13.06 and 5.67 days for EHD2 and 13.12 and 14.44 for EHD3, respectively. In other words, the fine-tuning and reactivity speed have improved by at least 78.2–84.3% and 68.5–88.1%.

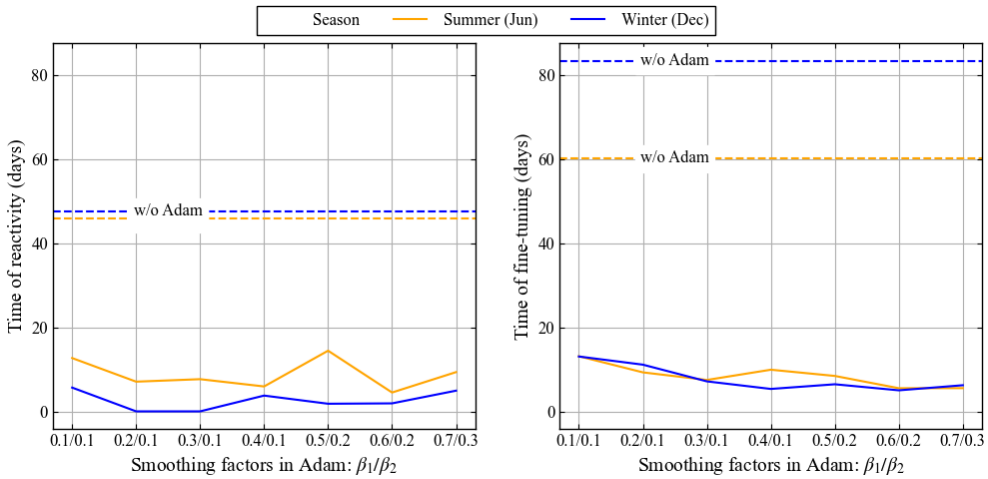

Figure 11. ToF and ToR of LAC-AB algorithm.

With respect to the latency and power failure, Figure 12 depicts these metrics for the chosen decay rate combinations in comparison to the LAC algorithm without Adam. The EHD1 dataset was used to obtain the results consistent with the real-world situations. The differences in the mean values of latency are not outstanding across all cases, but the mean standard deviation for the first six months tends to decrease as the decay rates become larger, especially until β_1 reaches 0.4, while that for the last six months shows the opposite trend. Nonetheless, too large decay rates as well as fixed learning rate are more likely to bring about power failures. With smaller decay rates, the controller reacts more quickly to environmental stochasticity, leading to larger variations in latency, i.e., duty-cycle in case of less drastic changes as in the first six months, and yet with no or a couple of power failures, compared to 38 times in case of using fixed learning rate.

To summarize, we advise setting small decay rates such as $\beta_1 \in [0.2, 0.4]$ and $\beta_2 = 0.1$ for power-failure-sensitive applications and larger $\beta_1 \in [0.5, 0.7]$ with relatively smaller $\beta_2 \in [0.2, 0.4]$ for latency-sensitive ones, although these values may vary according to target applications. With any of these setups, the number of power failures can be drastically reduced to zero or a few, and the reactivity speed falls around within a day up to 15 days and the initial convergence is attainable in about 5–13 days for our application use case where the CUI is 30min. To cope with reactivity to new situations, it would be another solution to reset and re-learn from scratch by detecting the environmental change as in [23] that requires the detection algorithm. However, our solution can achieve faster ToR than ToF, and therefore it is simpler and yet effective without such a mechanism.

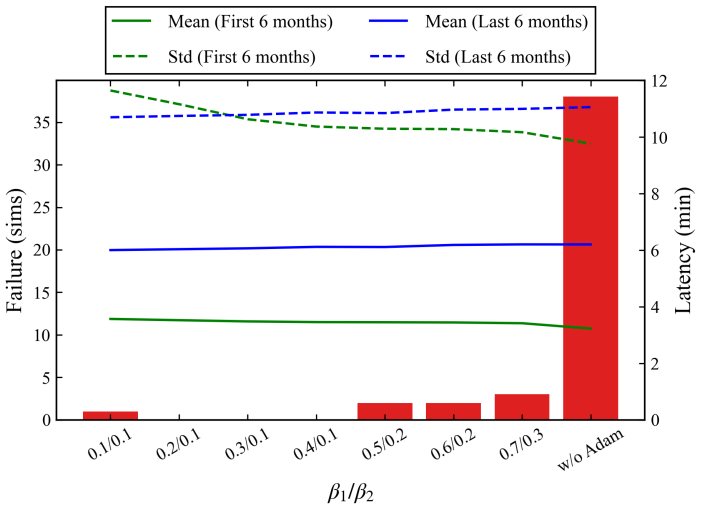

Figure 12. Latency and number of failed simulations of LAC-AB algorithm.

9. Conclusions and Future Direction

In this paper, we propose LAC-AB, the integration of the LAC algorithm and Adam with smaller decay rates and no initialization bias correction terms in order to grapple with drastic environmental changes and avoid power failures. This idea is of importance because the recent trend in EH-IoT requires μW-range low-cost systems with high adaptability to the changing environment. The consideration of the SoB would act as an index for necessary and sufficient performance and help achieve scalability to wide applications. The introduction of Adam overcomes the reactivity issue caused by fixed learning rate in LAC algorithms. For quantitative analysis of convergence, we attempted to define the time of convergence for the ToF and ToR based on the mean and variance of the optimized parameter. First, we proved that, along with the effectiveness of SoB as a part of state, the conventional decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are not suitable and no use of initialization bias correction with smaller rates is acceptable. Then, we performed extensive simulations to obtain the best possible sets of β_1, β_2. Finally, our simulation results show that our proposal enables improving both the fine-tuning and reactivity, yielding 13.06 and 5.67 days for EHD2 and 13.12 and 14.44 for EHD3 for ToF and ToR, respectively, as the worst case among the best possible combinations of β_1, β_2, compared to 83.40 and 47.52 days for EHD2 and 60.17 and 45.77 days for EHD3 in the case of LAC algorithm using fixed learning rate. This corresponds to improvements of fine-tuning and reactivity of at least 78.2–84.3% and 68.5–88.1%, respectively. Importantly, the number of power failures is zero or a few times over 300 simulation cases in the LAC-AB algorithm, compared to 38 times for the LAC algorithm without Adam. Hence, our proposal augments LAC with more adaptability at low cost for lightweight IoT applications.

The proposed algorithm mostly involves multiply and add operations along with only two divisions, one squared root operation and one Gaussian random number generation. Ongoing research is being conducted to turn these computations into only multiply and add operations with the use of fixed-point quantization, and therefore, to make our algorithm more lightweight. Preliminary work suggests that the approximated version of our proposed algorithm exhibits similar behavior and would even allow the design of a dedicated hardware component.

Author Contributions: S.S. conducted the initial formal analysis and he developed the proposed algorithm under the supervision of J.-F.C. and S.L. Then, he proposed the software implementation and performed the tests reported in the present paper. S.S. wrote the initial draft of the paper that has been reviewed and edited by J.-F.C. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study has been partly funded by the OCEAN12 ECSEL project (ECSEL Project ID 783127-2).

Data Availability Statement: Data used in this study can be found on [33].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lallement, G.; Abouzeid, F.; Cochet, M.; Daveau, J.; Roche, P.; Autran, J. A 2.7 pJ/cycle 16 MHz SoC with 4.3 nW power-off ARM Cortex-M0+ core in 28 nm FD-SOI. In Proceedings of the ESSCIRC 2017—43rd IEEE European Solid State Circuits Conference, Leuven, Belgium, 11–14 September 2017; pp. 153–162. [CrossRef]
2. Ait Aoudia, F.; Gautier, M.; Berder, O. RLMan: An Energy Manager Based on Reinforcement Learning for Energy Harvesting Wireless Sensor Networks. *IEEE Trans. Green Commun. Netw.* **2018**, 2, 408–417. [CrossRef]
3. Ortiz, A.; Al-Shatri, H.; Li, X.; Weber, T.; Klein, A. Reinforcement learning for energy harvesting point-to-point communications. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
4. Van Hasselt, H.; Wiering, M.A. Using continuous action spaces to solve discrete problems. In Proceedings of the 2009 International Joint Conference on Neural Networks, Atlanta, GA, USA, 14–19 June 2009; pp. 1149–1156. [CrossRef]
5. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.

6. De Roose, J.; Xin, H.; Andraud, M.; Harpe, P.J.A.; Verhelst, M. Flexible and Self-Adaptive Sense-and-Compress for Sub-MicroWatt Always-on Sensory Recording. In Proceedings of the ESSCIRC 2018—IEEE 44th European Solid State Circuits Conference (ESSCIRC), Dresden, Germany, 3–6 September 2018; pp. 282–285. [CrossRef]
7. Ju, Q.; Zhang, Y. Predictive Power Management for Internet of Battery-Less Things. *IEEE Trans. Power Electron.* **2018**, *33*, 299–312. [CrossRef]
8. Qiu, C.; Hu, Y.; Chen, Y.; Zeng, B. Lyapunov Optimization for Energy Harvesting Wireless Sensor Communications. *IEEE Internet Things J.* **2018**, *5*, 1947–1956. [CrossRef]
9. Hu, Y.; Qiu, C.; Chen, Y. Lyapunov-Optimized Two-Way Relay Networks With Stochastic Energy Harvesting. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 6280–6292. [CrossRef]
10. Bhat, G.; Park, J.; Ogras, U.Y. Near-optimal energy allocation for self-powered wearable systems. In Proceedings of the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, USA, 13-16 November 2017; pp. 368–375. [CrossRef]
11. Bhat, G.; Bagewadi, K.; Lee, H.G.; Ogras, U.Y. REAP: Runtime Energy-Accuracy Optimization for Energy Harvesting IoT Devices. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6.
12. Vigorito, C.M.; Ganesan, D.; Barto, A.G. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In Proceedings of the 2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, San Diego, CA, USA, 18–21 June 2007; pp. 21–30. [CrossRef]
13. Debizet, Y.; Lallement, G.; Abouzeid, F.; Roche, P.; Autran, J. Q-Learning-based Adaptive Power Management for IoT System-on-Chips with Embedded Power States. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5. [CrossRef]
14. Masadeh, A.; Wang, Z.; Kamal, A.E. An Actor-Critic Reinforcement Learning Approach for Energy Harvesting Communications Systems. In Proceedings of the 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019; pp. 1–6. [CrossRef]
15. Murad, A.; Kraemer, F.A.; Bach, K.; Taylor, G. Autonomous Management of Energy-Harvesting IoT Nodes Using Deep Reinforcement Learning. In Proceedings of the 2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Umea, Sweden, 16–20 June 2019; pp. 43–51. [CrossRef]
16. Qian, L.P.; Feng, A.; Feng, X.; Wu, Y. Deep RL-Based Time Scheduling and Power Allocation in EH Relay Communication Networks. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–7. [CrossRef]
17. Li, M.; Zhao, X.; Liang, H.; Hu, F. Deep Reinforcement Learning Optimal Transmission Policy for Communication Systems With Energy Harvesting and Adaptive MQAM. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5782–5793. [CrossRef]
18. Sawaguchi, S.; Christmann, J.F.; Molnos, A.; Bernier, C.; Lesecq, S. Multi-Agent Actor-Critic Method for Joint Duty-Cycle and Transmission Power Control. In Proceedings of the 2020 Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1015–1018.
19. Li, D.; Xu, S.; Zhao, J. Partially Observable Double DQN Based IoT Scheduling for Energy Harvesting. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
20. Zhao, N.; Liang, Y.; Niyato, D.; Pei, Y.; Jiang, Y. Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [CrossRef]
21. Qiu, C.; Hu, Y.; Chen, Y.; Zeng, B. Deep Deterministic Policy Gradient (DDPG)-Based Energy Harvesting Wireless Communications. *IEEE Internet Things J.* **2019**, *6*, 8577–8588. [CrossRef]
22. Biswas, D.; Balagopal, V.; Shafik, R.; Al-Hashimi, B.M.; Merrett, G.V. Machine learning for run-time energy optimisation in many-core systems. In Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 1588–1592. [CrossRef]
23. Das, A.; Merrett, G.V.; Tribastone, M.; Al-Hashimi, B.M. Workload Change Point Detection for Runtime Thermal Management of Embedded Systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *35*, 1358–1371. [CrossRef]
24. Shafik, R.A.; Yang, S.; Das, A.; Maeda-Nunez, L.A.; Merrett, G.V.; Al-Hashimi, B.M. Learning Transfer-Based Adaptive Energy Minimization in Embedded Systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *35*, 877–890. [CrossRef]
25. Yuan, F.; Zhang, Q.T.; Jin, S.; Zhu, H. Optimal Harvest-Use-Store Strategy for Energy Harvesting Wireless Systems. *IEEE Trans. Wirel. Commun.* **2015**, *14*, 698–710. [CrossRef]
26. Christmann, J.F.; Beigne, E.; Condemine, C.; Vivet, P.; Willemin, J.; Leblond, N.; Piguet, C. Bringing Robustness and Power Efficiency to Autonomous Energy-Harvesting Microsystems. *IEEE Des. Test Comput.* **2011**, *28*, 84–94. [CrossRef]
27. Dekimpe, R.; Xu, P.; Schramme, M.; Flandre, D.; Bol, D. A Battery-Less BLE IoT Motion Detector Supplied by 2.45-GHz Wireless Power Transfer. In Proceedings of the International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS) 2018, Platja d'Aro, Spain, 2–4 July 2018; pp. 68–75. [CrossRef]
28. Sangare, F.; Xiao, Y.; Niyato, D.; Han, Z. Mobile Charging in Wireless-Powered Sensor Networks: Optimal Scheduling and Experimental Implementation. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7400–7410. [CrossRef]

29. Texas Instruments. CC2500 Low-Cost Low-Power 2.4 GHz RF Transceiver. 2019. Available online: http://www.ti.com/lit/ds/swrs040c/swrs040c.pdf (accessed on 6 April 2021).
30. Urard, P.; Romagnello, G.; Banciu, A.; Grasset, J.C.; Heinrich, V.; Boulemnakher, M.; Todeschni, F.; Damon, L.; Guizzetti, R.; Andre, L.; et al. A self-powered IPv6 bidirectional wireless sensor amp; actuator network for indoor conditions. In Proceedings of the 2015 Symposium on VLSI Circuits (VLSI Circuits), Kyoto, Japan, 17–19 June 2015; pp. C100–C101. [CrossRef]
31. Goldsmith, A. *Wireless Communications*; Cambridge University Press: Cambridge, UK, 2005. [CrossRef]
32. Panasonic Industry. Available online: https://www.panasonic-electric-works.com/cps/rde/xbcr/pew_eu_en/ca_amorton_solar_cells_en.pdf (accessed on 6 April 2021).
33. Oak Ridge National Laboratory (RSR) Daily Plots and Raw Data Files. Available online: https://midcdmz.nrel.gov/apps/sitehome.pl?site=ORNL (accessed on 6 April 2021).
34. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
35. Devlin, S.; Yliniemi, L.; Kudenko, D.; Tumer, K. Potential-based Difference Rewards for Multiagent Reinforcement Learning. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'14), Paris, France, 5–9 May 2014; pp. 165–172.
36. Gleser, L.J.; Perlman, M.D.; Press, S.J.; Sampson, A.R. *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*; Springer: Berlin/Heidelberg, Germany, 1960; pp. 278–292.
37. Brown, M.B.; Forsythe, A.B. Robust Tests for the Equality of Variances. *J. Am. Stat. Assoc.* **1974**, *69*, 364–367. [CrossRef]
38. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*, 5th ed.; CRC Press: Boca Raton, FL, USA, 2011.