An Energy-efficient Scheduling based on Q-learning for Energy Harvesting Embedded System

Jiayuan Wei[1], Xingyu Miao[2], Kehong Xu[3], Rui Liu[1], and Yongqi Ge*
[1]School of Information Engineering, Ningxia University, Yinchuan, China
[2]Durham University, Durham, England, U.K
[3]Xinhua College of Ningxia University, Yinchuan, China
jiayuan_wei@163.com, xingyu.miao@durham.ac.uk, keh_nxu@163.com, liurui2004_nxu@163.com, geyongqi@nxu.edu.cn
*corresponding author

Abstract—**In the embedded system, the energy level of the device can satisfy the energy consumption of the minimum task at all times, which is a necessary condition to maintain the sustainable operation of the energy harvesting embedded system. Aiming at the problem that the energy-harvesting embedded system is unable to ensure the schedulability of tasks in the system due to energy shortage and energy waste, we propose an energy-saving scheduling strategy based on Q-learning. On the premise of meeting the real-time requirements of tasks, the agent can reduce the energy consumption in the process of task execution by reasonably arranging the task scheduling sequence, and ensuring that the remaining energy of the system can maintain the normal execution of the scheduling task. The experimental results show that the q-learning algorithm saves 61.8% and 55.32% of the average energy compared with the as late as possible (ALAP) and as soon as possible (ASAP) algorithms, respectively, and the time of system energy is maintained within a reasonable range is on average 21.01% more stable than that of the ASAP.**

Keywords- reinforcement learning; embedded systems; energy harvesting

I. Introduction

With the application of embedded systems in various fields, the operating environment of the system has gradually become complicated [1], such as remote uninhabited areas or harsh ecological locations, posing its energy utilization requirements increasingly stringent [2]. Traditional embedded devices are mainly powered by batteries with limited available energy. Batteries run out of power and face the need to replace batteries, which is impractical in those areas where the deployment environment is harsh, so energy harvesting technology is introduced into embedded systems, which uses energy harvested from the environment to continuously supply power [3]. However, although energy harvesting technology provides a new energy supply solution for embedded systems, it also brings new challenges.

Due to the instability and dynamic variability of energy harvesting [4], the remaining energy of the equipment is unable to guarantee the schedulability of tasks at all times. In this work, we take solar energy into account for follow-up research. As shown in Fig. 1, it shows the energy fluctuation of the energy harvesting embedded system. When the energy harvesting rate is larger, the harvested energy exceeds the energy demand of the computing unit, the excess energy is stored in the storage unit, however, when the stored energy reaches the upper limit of the storage capacity E_{max}, the harvested excess energy will cause energy waste. On the contrary, when the energy harvesting rate is slower, the available energy is less than the energy demand of the computing unit, and the task of the computing unit will be interrupted due to energy shotrage. The traditional scheduling algorithm of the energy harvesting embedded system proposed many energy-saving methods to solve the energy shortage problem, and stabilized the energy supply of equipment through energy saving technology but failed to provide a good solution to the energy waste problem.

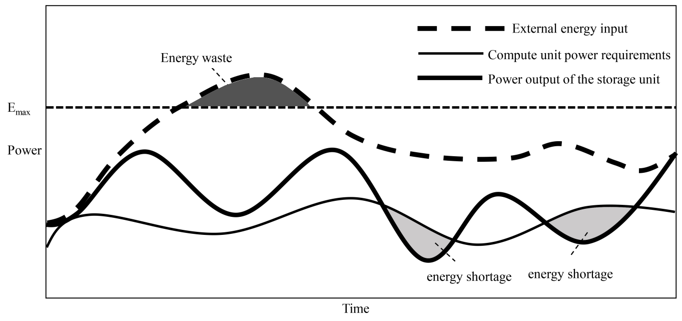


Fig. 1. Profile of energy required and energy available from external sources. The unsustainable operation can be caused by the imbalance of available and required power, in which case additional energy needs to be extracted from the battery.

ALAP and ASAP are the classic scheduling strategies in scheduling algorithms, but both of them have shortcomings. The ALAP scheduling algorithm aims to focus on the available energy for scheduling in the system and delay the execution of the task as much as possible [5], so it is unable to guarantee that the task will be completed before the deadline, resulting in scheduling failure. The ASAP scheduling algorithm aims to ensure the execution of tasks in time [6]. Once the energy in the system can meet the needs of the task, the scheduling task will be executed, but it is unable to guarantee that the system has the minimum energy to maintain sustainable operation after the execution of the task, resulting in insufficient system energy.

The scheduling algorithm needs to ensure the real-time performance of tasks and sufficient available energy to avoid task interruption. The reinforcement learning algorithm has a significant effect on solving the problem of balance between the two [7]. Therefore, we propose a reinforcement learning task scheduling algorithm, which reduces the energy waste in the scheduling process and balances the energy and the real-time requirements of the task. The actions that can satisfy the requirements will receive a reward from the enviroment, and the actions that cause insufficient energy or the real-time performance is unsatisfied are given punishment measures, and the optimal action execution sequence is found by continuously interacting with the environment. The main contributions of this work are as follows:

1) We designed an energy-saving scheduling method based on Q-learning to overcome the problem of insufficient energy in the energy harvesting embedded system.
2) Our proposed method reduces the wasted energy on average by 55.32% and 61.8% owing to limited storage capacity when compared to the ASAP and ALAP algorithms.
3) Our proposed method takes an average of 21.01% more time than the ASAP method to keep the energy state stable.

This paper is split into well-regulated systematic sections as follow: Section II provides a brief literature overview on EHES and RL. Our experimental model is presented in section III. Then, in section IV of the article, we introduce the related information of RL and apply Q-learning to our model. Section V discusses the system's simulation results as well as a comparison to other classic algorithms. Finally, Section VI describes the conclusion of the paper.

II. Related Work

Many scheduling algorithms focus on energy saving in energy harvesting embedded systems. Moser et al. [8] proposed an algorithm called the Lazy Scheduling Algorithm (LSA), which was based on the hypothesis that energy consumption correlated with worst-case response time, but this assumption later proved to be infeasible [9]. Abdeddaim et al. proposed classical ALAP and ASAP scheduling algorithms, in which the goal of the ALAP algorithm is to harvest as much energy as possible to delay the execution of tasks, and compress slack time [10] as much as possible so that the system can supplement battery energy to the maximum extent. ASAP algorithm judges the current energy level before each time unit task execution. If the current energy level can support the task execution for one time unit, the task will be executed at this time unit, otherwise, the system will suspend for one time unit to replenish energy, and then repeat the previous judgment. However, the application of these two algorithms may cause unnecessary waste of resources in some scenarios. Later, a clairvoyant algorithm called EDeg has been proposed in [11], [12], which relied on a generalizable meta policy, as long as the system can perform without energy failure, however, as soon as future energy failure is detected, the system is suspended as long as timing constraints are met or until the energy storage units is full.

In recent years reinforcement learning(RL) methods have been used to solve the problem of energy depletion in embedded devices. In RL, agents make intelligent decisions through interactive learning with the environment and constantly learn from the rewards they receive [13]. Islam et al. [14] proposed a new Q-learning-based approach that combined multiple Dynamic Vlotage and Frequency Scaling (DVFS) technologies and explored which DVFS technology is most suitable for different situations. In [15], a q-learning-based energy saving scheduling method for periodic tasks in real-time systems is proposed, which is combined with the energy saving technology DVFS to realize energy saving research. In addition, Ding et al. [16] proposed a two-stage q-learning adaptive scheduling algorithm to solve the problem of task allocation and energy consumption in task scheduling. On this basis, we consider applying q-learning method to energy acquisition embedded system to achieve the balance between energy consumption and task scheduling.

III. System model

In this section, we introduced the energy harvesting embeded system(EHES) model with solar energy as an example. EHES is composed of an energy management unit and task execution unit, we investigate the energy management model and tasks model in this work, as shown in Fig. 2.

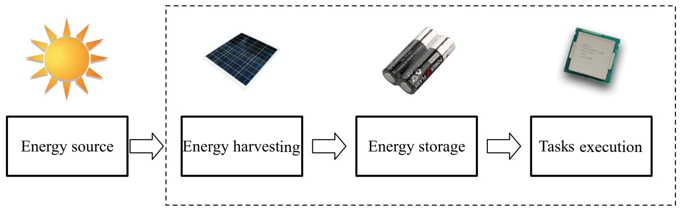


Fig. 2. An Energy-Harvesting Embedded System Architecture.

A. Energy management unit

The energy management unit is mainly composed of an energy harvesting unit and an energy storage unit.

a) Energy harvesting unit: In this paper, we consider the solar panel as the stored unit of renewable energy. The harvested energy with a large fluctuation charging rate $P_r(t)$ from the surrounding environment is uncontrollable and then converted into electrical power. The process of energy production is a time integral function. For the convenience of calculation, we assume that the charging rate is assumed a constant, based on this, the energy production during any time interval $[t_1, t_2]$ is denoted as $E_r(t_1, t_2)$ the following is shown in equation (1).

$$E_r(t_1, t_2) = \int_{t_1}^{t_2} P_r(t)dt \tag{1}$$

Where the $P_r(t)$ is a constant function, so it can be represented as the following formula (2).

$$P_r(t) = P_r \tag{2}$$

Therefore, in the following the energy harvest during the time interval $[t_1, t_2]$ is given by equation (3).

$$E_r(t_1, t_2) = P_r \times (t_2 - t_1) \tag{3}$$

b) Energy storage unit: There are various energy storage devices are available in EHES, in this work, we use a battery as an energy storage device. Without loss of generality, we suppose there is no loss during energy storage. The energy storage of the battery has two extreme values, i.e. E_{max} and E_{min}, where E_{max} is the maximum capacity that the battery can store, and E_{min} is the minimum capacity that the battery can store, we assume that initial energy $E_{min} = 0$, the energy stored at time t is $E_s(t)$, and the energy stored during the time $[t_1, t_2]$ is denoted as $E_s(t_1, t_2)$. When $E_s(t_1, t_2)$ is positive, it means energy consumption is less than energy storage, however, when $E_s(t_1, t_2)$ is negative, it means energy consumption is more than energy storage. In the process of converting solar energy into electrical energy and storing it in the energy storage unit, a certain amount of energy conversion loss will occur, in this paper, we assume that the energy storage is lossless.

B. Tasks model

We suppose the task set consists of a series of independent and periodic real-time tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$, and task consists of many real-time sub-jobs, the different task has different real-time jobs, each task has a 4-tuple (C_i, D_i, E_i, T_i), in which C_i is the worst-case execution time, D_i is a relative deadline of task $(D_i \leq T_i)$, E_i the worst-case energy consumption, and T_i is the period of task. In this work, we randomly generated a large number of periodic task sets and use YARTISS [17], [18] as the simulation tool, which utilizes an adapted version of the UUniFast-Discard algorithm [19] coupled with a limitation of the hyperperiod technique to generate task sets.

IV. RL PROBLEM FORMULATION FOR EHES

Reinforcement algorithm methods are essential to solve the optimal control problem and environment interaction. The goal is to maximize the reward of the agent through a series of responses to the dynamic environment. Q-learning can be applied to the scheduling tasks with time and energy constraints problem because it can effectively capture the dynamic of the task scheduling and environmental conditions efficiently, and it is the most popular value-based reinforcement learning algorithm, in which the agent builds its Q-table to estimate the discounted future reward and takes the action with the largest Q value at each step. After the agent takes an action to act on the environment, the environment state changes, and a reward is given. The environment state moves to the next new state until the policy converges. Fig. 3. shows an overview of task scheduling using Q-learning. The agent selects the action that needs to be performed, then the environment feeds back the effect of the current action, and the agent gets the reward while reaching the new state.

The general goal of the agent is to maximize its total reward by learning which action is optimal for a particular state, the state value function is used to represent the agent's reward expectation at state s. When the agent explores the optimal execution sequence, the reward will converge to a steady state. Therefore, the concept of discount factor γ is introduced, and

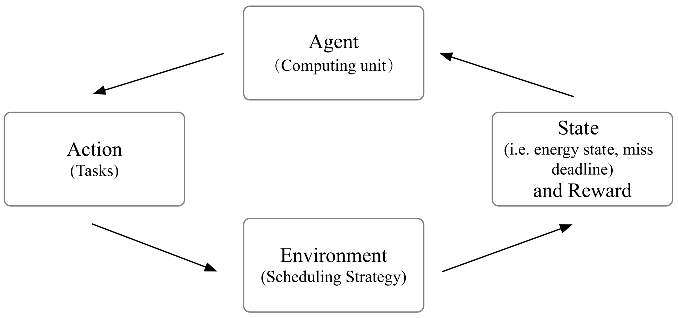


Fig. 3. RL Communication Process for EHES.

γ value is between 0 and 1. When it approaching 1 means that the expected reward in the future is valued, otherwise, which means that the expected reward in the present is valued. The derivation process of the state value function of the agent is as follows:

$$\begin{aligned} Q_\pi(s,a) &= r_0 + \gamma Q_1 \\ &= r_0 + \gamma(r_1 + \gamma Q_2) \\ &= r_0 + \gamma r_1 + \gamma^2(r_2 + \gamma Q_3) \\ &= \ldots \\ &= r_0 + \gamma(r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots) \\ &= r_0 + \gamma max_a Q\pi(s', a) \end{aligned} \tag{4}$$

The Q value of the state-action pair is updated by the following formula (5).

$$\begin{aligned} &Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) \\ &+ \alpha \cdot \left(R_{t+1} + \gamma \cdot \max_{(\forall a \in A)} \left[Q_t(s_{t+1}, a_t) \right] - Q_t(s_t, a_t) \right) \end{aligned} \tag{5}$$

A. Agent

The agent is a selector of the actions that a task can perform. The agent selects tasks according to the strategy at each unit of time. In the Q-learning algorithm, the agent follows the ε-greedy strategy as the action selection strategy, which means the agent selects the action with the highest Q value with epsilon probability, otherwise, it will randomly select another action with the probability of 1 - ε, continuously interacting with the environment until the Q-table reaches converge state.

B. Environment

Tasks need to satisfy several conditions during execution, which are deployed in the environment. The following situations may occur during task execution.

- During the execution of a task, energy will be consumed. Therefore, it is necessary to consider that the remaining energy of the battery is less than the energy consumed by the task. If the agent finds that the current energy in the battery is insufficient to support the execution of the next task, the system will automatically enter the hibernation state, that is, no task will be executed to replenish energy.
- Through iterative training, the agent can train the task execution sequence to ensure that the task is completed within the deadline $C_i + t < D_i$.

C. State

The least common multiple of the period of each task is defined as a hyperperiod [20], and the hyperperiod is divided into multiple time units of 1 second. In this hyperperiod, each task executes a time unit, and then if the system has enough energy, the agent decides the task to be executed in the next time unit according to the policy, otherwise the system is in sleep state, so we summarize what may happen in the task execution as the state space including the task missed deadline, the system energy is insufficient, the system executes the task, the system is idle state, the remaining time in the super cycle cannot meet the task execution time, and the task is not completed within the hypercycle.

D. Action

In this paper, we define a task as an action space. When the system energy is insufficient, it cannot continue to execute the task. In order to keep the available energy in the system, we also define an additional action, that is, the system is in a sleep state and does not execute the task.

E. Reward function

In a time unit, if the deadline is missed after the task is executed, it will get a negative reward of -50, otherwise it will get a positive reward of 0.1, after a round is over, tasks that meet the real-time and energy requirements will be Get a reward with a value of 100. We define that if the remaining energy of the system is between 20% and 80% of the maximum energy, the system has a longer life [21], so for tasks that satisfy this condition, the reward obtained is as shown in the formula (6) .

$$\sum_{t=0}^{n} \frac{|E_{curr} - E_{max} \times 50\%|}{E_{max}} + E_{ave} \tag{6}$$

Where t denotes time unit t, E_{curr} denotes the current energy of system at a certain moment, E_{max} denotes the max energy that battery can store, E_{ave} denotes the average energy, it can be expressed by equation (7), which means the average that total E_{curr} of at n time points.

$$\sum_{t=0}^{n} \frac{E_{curr}}{n} \tag{7}$$

F. Q-learning algorithm

Algorithm 1 introduces the application of Q-learning in EHES in detail. The agent receives the initial system state and action index after initializing various parameters (lines 1-4). The algorithm starts the episode and chooses an action following the ε-greedy policy (lines 6-10). The environment will feedback the next system state according to the selection of actions (line 12). If the scheduling is successful, the timeline will be added with one (line 11). After that, we calculate the reward value according to the current action, current state, and next state (line 13), and then update the Q table (lines 14-16). We gradually improved our policy training strategy to shorten the convergence time.

Algorithm 1 Q-learning Algorithm for EHES

```
1:  Initialize $Q_{table}$ as an empty table
2:  Initialize action $a$, statue $S_{curr}, S_{next}$, and $timeline = 0$
3:  Initialize RL hyperparameters
4:  $S_{curr} \leftarrow$ Environment
5:  while $timeline < hyper - period$ do
6:      if uniform(0,1) $\leq \epsilon$ then
7:          $a = maxQ(S_{curr}, a')$
8:      else
9:          $a =$ take a random action
10:     end if
11:     The environment returns the result. If not done, the
        $timeline + 1$.
12:     $S_{next} \leftarrow$ Environment
13:     r = reward($a, S_{curr}, S_{next}$)
14:     $q_{predit} = Q_{table}[S_{curr}, a]$
15:     $q_{target} = r + \gamma max(Q_{table}[S_{next}, \forall a])$
16:     $Q_{table}[S_{curr}, a] += \alpha(q_{target} - q_{predit})$
17:     $S_{curr} \leftarrow S_{next}$
18: end while
```

TABLE 1

Q-Learning Hyper-parameters Used for Simulations.

Hyper-Parameter	Value
Learning rate	0.01
Epsilon(ε)	0.9
Episode	10000
Reward-decay (γ)	0.99

V. Experiment results

In this section, we compare the algorithm combined with the Q-learning model with the traditional energy harvesting scheduling algorithms, ALAP and ASAP. In order to evaluate the performance of the introducing Q-learning EHES. We developed a simulated environment using python, and we conducted experiments based on the OpenAI gym library.

We adopt the same task set and experiment conditions to ensure that the experiment is fair. We randomly generated 3 task sets, of which there are 4, 5, and 6 independent tasks respectively. We hope that the algorithm, in combination with Q-learning, can assure its operation while maintaining a high battery energy level, assuring the battery's lifespan and therefore extending the life of the EHES. We set various Q-Learning hyper-parameters in the experiment, which are detailed in the Table 1.

We use the YARTISS tool to generate three task sets, the proposed scheduling algorithm with Q-learning was compared with the ALAP algorithm and ASAP algorithm. We compared the remaining energy of the system at the current moment in the task scheduling, the wasted energy and the time the system was in the extended life state, and verified the effectiveness of the algorithm. In Figure 4, we show the average reward convergence while running a simulation and the experimental results are shown in the Figure 5.

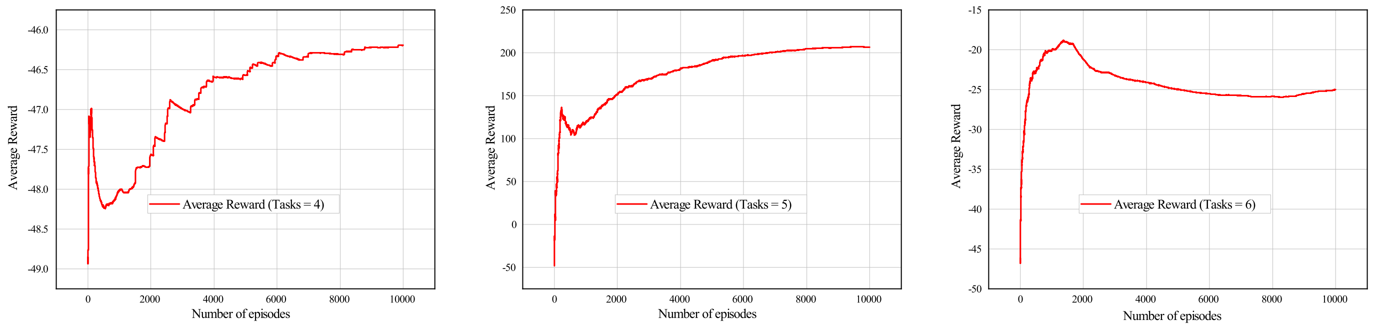
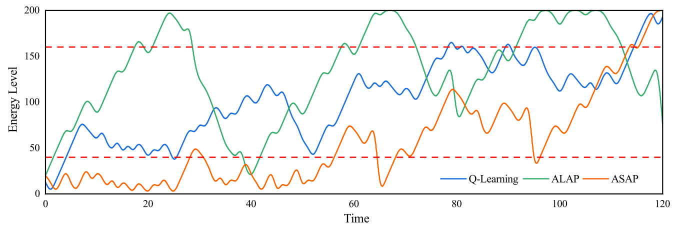
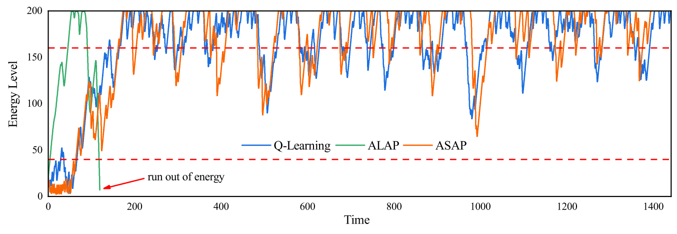


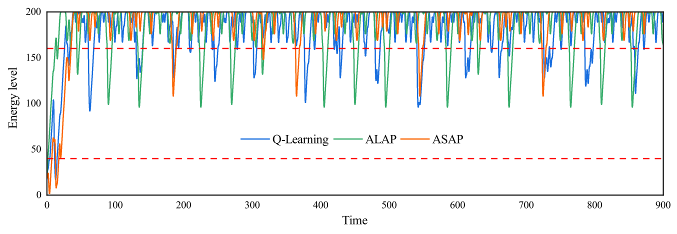
Fig. 4. Average Reward Simulation Results for Three Task Sets.



(a) Task Set with Task Number 4



(b) Task Set with Task Number 5



(c) Task Set with Task Number 6

Fig. 5. Changes in Energy Over Time for Three Task Sets under Three Different Policies.

TABLE 2

Average Energy Level.

Task number	Policy	Value
4	Q-learning	**103.77**
	ALAP	143.32
	ASAP	39.78
5	Q-learning	**165.08**
	ALAP	-
	ASAP	151.74
6	Q-learning	**179.66**
	ALAP	182.20
	ASAP	179.23

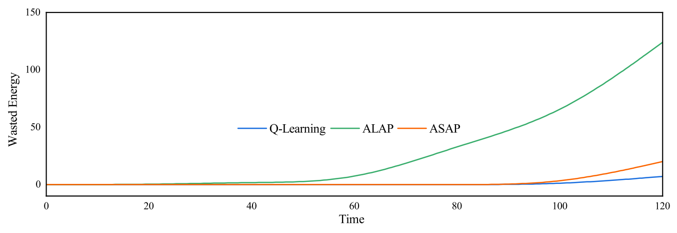
TABLE 3

In a Hyper-period, The Proportion of The Energy Level of The Three Task Sets in The Range of 20%-80% under Three Different Policies.

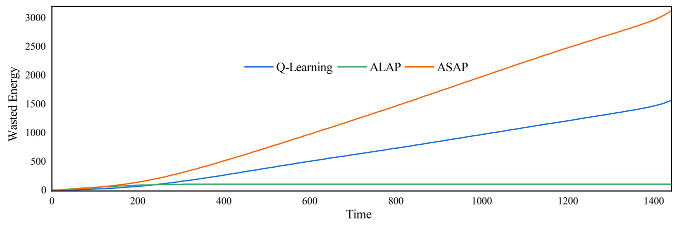
Task number	Policy	Percentage
4	Q-learning	**85.12%**
	ALAP	58.68%
	ASAP	43.80%
5	Q-learning	**29.91%**
	ALAP	-
	ASAP	23.94%
6	Q-learning	**20.64%**
	ALAP	11.88%
	ASAP	4.88%

We observe that ALAP's energy state satisfies its characteristics, and then delay task release as much as possible while ensuring that EHES energy is sufficient and the system's long-term operation is as stable as possible. The ALAP scheduling algorithm guarantees a large energy level in a hyper-period, as shown in the Table 2, the average energy in the task set with four tasks is 143.32, while the average energy in the task set with six tasks is 182.20. It's worth noting that the ALAP scheduling algorithm fails to schedule in the task set of 5 tasks, which is linked to ALAP weaknesses. Because the ALAP is overly concerned with energy, all tasks are postponed until the execution deadline, consuming a great deal of energy and making it difficult to arrange the set of tasks that can be scheduled. On the other hand, ASAP is the opposite. In comparison to ALAP and Q-learning policies, ASAP has a lower energy level in a hyper-period to ensure the priority of task calculation. The average energy level in the tasks set with 4, 5, and 6 is 39.78, 151.74, and 179.23, respectively. Our goal in implementing Q-learning is to take into account as many energy and time attributes as feasible in order to ensure that EHES can run for an extended period of time. During the execution of the task, we must ensure that the energy storage unit's energy is sufficient and at a high level, but that the energy is not exhausted and the EHES stops running. As a result, we established a 20% - 80% energy range to attain this purpose. We believe that in the 20% - 80% energy range, the task can meet the system's minimum execution energy consumption and will not waste the harvested energy due to excessive attention to energy.

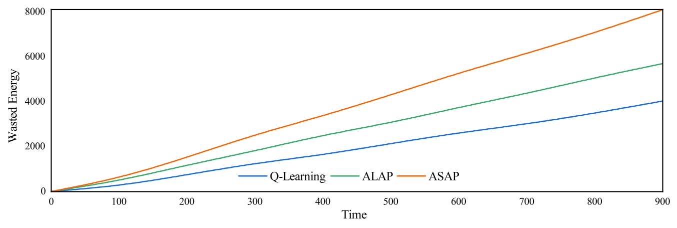
In the test of three sets of task sets, Q-learning has the largest proportion in the range of 20% to 80% when compared to ALAP and ASAP, as shown in Table 3. We observe that this value appears to be influenced by the change of task attributes and energy harvesting power in the task set. While the results of ALAP, ASAP, and Q-learning may be similar in some extreme instances, Q-learning outperforms ALAP and ASAP in terms of time and energy coordination. In addition, as shown in Table 3, Q-learning has a relatively high energy level, indicating that Q-learning can keep the system energy steady and healthy while allowing EHES to operate for a long time.



(a) Task Set with Task Number 4



(b) Task Set with Task Number 5



(c) Task Set with Task Number 6

Fig. 6. Change of Wasted Energy of Three Task Sets with Time under Three Different Policies.

TABLE 4

Wasted Energy Value.

Task number	Policy	Value
4	Q-learning	**7**
	ALAP	124
	ASAP	20
5	Q-learning	**1445**
	ALAP	110
	ASAP	2931
6	Q-learning	**4026**
	ALAP	5689
	ASAP	8094

The comparison of ALAP, ASAP, and Q-learning algorithms for battery energy waste is presented in Figure 6. In task sets of 4, 5, and 6, we can see that the Q-learning scheduling method wastes the least amount of energy when compared to ALAP and ASAP algorithms. In the hyper period with a task set of four, the Q-learning algorithm saves 94.35% more energy than the ALAP algorithm and 65% more energy than the ASAP strategy, as demonstrated in Table 4. In the hyper-period with the task set of 5, its waste energy is 50.70% less than the ASAP algorithm. In the hyper-period with the task set of 6 (Figure 6(c)), its waste energy is 29.23% less than the ALAP algorithm and 50.26% less than the ASAP algorithm. We can see that the wasted energy algorithm of the ASAP method is almost identical to the Q-learning algorithm in Figure 6(a). This is because the characterizer of the ASAP algorithm keeps the battery energy at a lower level at the start of the scheduling. However, if the battery can store enough energy, the amount of energy wasted over time may grow. The amount of energy wasted by ALAP is constant in Figure 6(b). The system stops functioning because the ALAP algorithm fails to schedule at roughly 150 time units.

VI. Conclusion

In this study, an energy harvesting task scheduling strategy based on the Q learning was proposed, which aims to satisfy the characteristic of real-time and energy requirements in the process of task scheduling, reduce energy waste during energy harvesting, maintain the system energy level in the state that can maintain the normal operation of task scheduling. Moreover, compared with the typical energy harvesting scheduling algorithms, the proposed Q-learning scheduling algorithm can utilize the harvested energy more effectively, to achieve the purpose of prolonging the operation life of the system. In the future, we will improve its scalability and focus on applying it to scheduling strategies of other renewable energy.

VII. acknowledgment

This work was supported in part by Natural Science Foundation of Ningxia under grant no. 2022AAC03004 and 2021AAC03041, in part by the National Natural Science foundation of China under grant no. 61862049 and 62162052, in part by the Key Research and Development Project of Ningxia Province (special talent) under grant no. 2021BEB04016, and in part by the Young Scholar in Western China of Chinese Academy of Sciences under grant no.XAB2018AW12.

References

[1] M. Malewski, D. M. Cowell, and S. Freear, "Review of battery powered embedded systems design for mission-critical low-power applications," *International Journal of Electronics*, vol. 105, no. 6, pp. 893–909, 2018.
[2] S. Tzilis, P. Trancoso, and I. Sourdis, "Energy-efficient runtime management of heterogeneous multicores using online projection," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 4, pp. 1–26, 2019.
[3] A. Harb, "Energy harvesting: State-of-the-art," *Renewable Energy*, vol. 36, no. 10, pp. 2641–2654, 2011.
[4] R. C. Hsu and T.-H. Lin, "A fuzzy q-learning based power management for energy harvest wireless sensor node," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 957–961, IEEE, 2018.
[5] Y. Chandarli, Y. Abdeddaïm, and D. Masson, "The fixed priority scheduling problem for energy harvesting real-time systems," in *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 415–418, IEEE, 2012.

[6] Y. Abdeddaïm, Y. Chandarli, and D. Masson, “The optimality of pfpasap algorithm for fixed-priority energy-harvesting real-time systems,” in *2013 25th Euromicro Conference on Real-Time Systems*, pp. 47–56, IEEE, 2013.

[7] F. Fraternali, B. Balaji, Y. Agarwal, and R. K. Gupta, “Aces: Automatic configuration of energy harvesting sensors with reinforcement learning,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 4, pp. 1–31, 2020.

[8] C. Moser, D. Brunelli, L. Thiele, and L. Benini, “Lazy scheduling for energy harvesting sensor nodes,” in *IFIP Working Conference on Distributed and Parallel Embedded Systems*, pp. 125–134, Springer, 2006.

[9] R. Jayaseelan, T. Mitra, and X. Li, “Estimating the worst-case energy consumption of embedded software,” in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, pp. 81–90, IEEE, 2006.

[10] J. P. Lehoczky and S. Ramos-Thuel, “An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems.,” in *RTSS*, vol. 92, pp. 110–123, Citeseer, 1992.

[11] H. E. Ghor, M. Chetto, and R. H. Chehade, “A real-time scheduling framework for embedded systems with environmental energy harvesting,” *Computers & Electrical Engineering*, vol. 37, no. 4, pp. 498–510, 2011.

[12] M. Chetto, D. Masson, and S. Midonnet, “Fixed priority scheduling strategies for ambient energy-harvesting embedded systems,” in *2011 IEEE/ACM International Conference on Green Computing and Communications*, pp. 50–55, IEEE, 2011.

[13] R. Sutton and A. Barto, “Reinforcement learning: An introduction. mit press; 2018,” *Google Scholar*, pp. 329–331.

[14] F. M. M. ul Islam and M. Lin, “Hybrid dvfs scheduling for real-time systems based on reinforcement learning,” *IEEE Systems Journal*, vol. 11, no. 2, pp. 931–940, 2015.

[15] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, “Energy-efficient scheduling for real-time systems based on deep q-learning model,” *IEEE transactions on sustainable computing*, vol. 4, no. 1, pp. 132–141, 2017.

[16] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, “Q-learning based dynamic task scheduling for energy-efficient cloud computing,” *Future Generation Computer Systems*, vol. 108, pp. 361–371, 2020.

[17] Y. Chandarli, M. Qamhieh, F. Fauberteau, and D. Masson, *Yartiss: a generic, modular and energy-aware scheduling simulator for real-time multiprocessor systems*. PhD thesis, UPE LIGM ESIEE, 2014.

[18] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, and M. Qamhieh, *Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms*. PhD thesis, UPE LIGM ESIEE, 2012.

[19] P. Emberson, R. Stafford, and R. I. Davis, “Techniques for the synthesis of multiprocessor tasksets,” in *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pp. 6–11, 2010.

[20] R. J. Bril, J. J. Lukkien, and W. F. Verhaegh, “Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited,” in *19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, pp. 269–279, IEEE, 2007.

[21] “How to prolong lithium-based batteries.” https://batteryuniversity.com/learn/article/how_to_prolong_lithium_based_batteries.