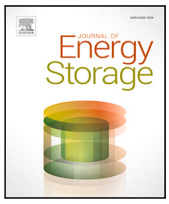
Enhanced Wolpertinger architecture for solving optimal control problems of reconfigurable battery systems

Journal of Energy Storage 179 (2026) 123881

Contents lists available at ScienceDirect

Journal of Energy Storage

journal homepage: www.elsevier.com/locate/est



Research Papers

Enhanced Wolpertinger architecture for solving optimal control problems of reconfigurable battery systems

Changyou Geng [a], Rui Li [b], Dezhi Ren [a], Enkai Mao [a], Xinyi Zheng [a], Changfu Zou [c], Weiji Han [a,b,c,*]

[a] China-UK Low Carbon College, Shanghai Jiao Tong University, Shanghai, 201306, China
[b] School of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[c] Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, 41296, Sweden

ARTICLE INFO

Keywords:
Reconfigurable battery systems
Optimal control
Deep reinforcement learning
Discrete action space
Wolpertinger architecture

ABSTRACT

Reconfigurable battery systems (RBSs) offer flexibility in battery connections to address several critical issues in battery safety, balancing, degradation, energy delivery, etc. However, the optimal control of RBSs is complicated due to the nonlinear characteristics of batteries and the discrete-value states of switches. To solve RBS optimal control problems, while some traditional methods, such as rule-based methods, graph theory-based methods, dynamic programming, and heuristic algorithms, can be deployed, some assumptions are commonly needed to simplify the system modeling or problem formulation. Deep reinforcement learning (DRL) has shown the potential to solve RBS optimal control problems without such assumptions, but it still faces critical difficulties in dealing with large-scale discrete action spaces. To address this challenge, we propose to introduce and enhance the Wolpertinger architecture (WA) when using DRL for solving RBS optimal control problems. Specifically, three action embedding methods are designed to enable the training of WA. Then, the original WA is enhanced by deploying a higher-performance agent and a nearest neighbor search algorithm to improve the convergence optimality and computation efficiency, respectively. Case studies based on an experimentally verified RBS model demonstrate that the enhanced WA manages to effectively solve RBS optimal control problems and outperforms the original WA and other DRL methods in terms of the convergence speed and converged reward. Such advantages become increasingly pronounced as the system scales up. For even larger-scale practical battery systems (e.g., 50 cells), a multi-agent architecture is introduced to successfully solve the optimal control problem. Experimental tests on an embedded RBS control platform further confirm the superior real-time performance and potential applicability of the enhanced WA in practical RBS control scenarios.

1. Introduction

The intrinsic electrochemical nature of batteries necessitates their series or parallel connection to meet high voltage, current, or power demands. However, heterogeneity among connected batteries can lead to imbalances in key states such as state of charge (SoC), temperature, and state of health (SoH), resulting in significant issues like reduced energy delivery, accelerated degradation, and increased safety risks. Conventionally, passive or active balancing technologies have been widely employed to mitigate these imbalances. Passive balancing dissipates excess energy as heat through resistors, leading to poor system energy efficiency [1]. While traditional active balancing improves efficiency, it heavily relies on auxiliary energy transfer components, e.g., capacitors, inductors, or transformers. This reliance not only increases the system's cost and volume but also introduces parasitic power losses during charge transfer. Furthermore, the mandatory charge-shuttling process imposes additional micro-charge/discharge cycles on the cells, inadvertently accelerating battery aging [2].

To overcome these fundamental limitations, reconfigurable battery systems (RBSs) have emerged as a superior option. Compared with traditional balancing schemes, the core advantage of an RBS lies in its ability to achieve highly efficient balancing by dynamically altering the battery topology to directly route current during operation [3]. By eliminating the need for dedicated, single-function equalization hardware, RBS transforms the balancing task into a pure software-defined control problem. Although the introduction of a switch array incurs an initial hardware cost, it offers multiplexing functional advantages. Thanks to the introduced flexibility in battery connections, the system performance can be enhanced from multiple perspectives, such as isolating faulty batteries for continuous and safe operation [4], improving

* Corresponding author.
E-mail addresses: w.han@sjtu.edu.cn, weijih@chalmers.se (W. Han).

https://doi.org/10.1016/j.est.2026.123881
Received 31 July 2025; Received in revised form 20 July 2026; Accepted 24 July 2026
Available online 30 July 2026

Nomenclature

A2C	Advantage Actor-Critic
ANN	Approximate Nearest Neighbor
BP	Bypassed
CCV	Cell Current Vector
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FLANN	Fast Library for Approximate Nearest Neighbors
HNSW	Hierarchical Navigable Small World
LN	Last Non-Bypassed
NCV	Neighboring Connection Pattern Vector
NMSLIB	Non-Metric Space Library
PL	Parallel Connection
PPO	Proximal Policy Optimization
RBS	Reconfigurable Battery System
SoC	State of Charge
SoH	State of Health
SR	Series Connection
SSV	Switch State Vector
TD3	Twin Delayed Deep Deterministic Policy Gradient
WA	Wolpertinger Architecture

energy conversion [5], balancing battery SoCs [6], optimizing thermal management [7], customizing the terminal voltage within a wide range [8,9], enhancing lifetime [10], and facilitating the management of batteries with varying degradation, types, and chemistries [11].

To fully harness the benefits of RBSs, an optimal control problem needs to be formulated, incorporating the objective function, system model, and necessary constraints. However, due to the nonlinearity of battery characteristics and the discrete nature of switch states, the formulated RBS optimal control problem becomes a complex, high-dimensional, nonlinear, 0–1 integer programming problem, resulting in critical challenges in solving the problem. Additionally, the problem's dimensionality increases as the number of batteries and switches in the RBS grows, further complicating the problem solving. To solve RBS optimal control problems, several methods have been proposed in recent studies.

Kim et al. [12] established an offline lookup table to determine the optimal configuration for maximizing converter efficiency. This rule-based method is effective and easy to implement, but it is only applicable to small search spaces. He et al. developed graph theory-based algorithms to minimize the discharging current of individual cells [13], and maximize the deliverable discharge capacity [14] and charge capacity [15]. In such graph theory-based methods, the RBS model was simplified to a graph representation, and the constraints and objectives were expressed accordingly. For example, minimizing the discharging current was transformed into finding the maximum number of disjoint simple paths [13]. Although model simplification facilitates the problem solving, it inevitably results in certain loss of accuracy and reduced applicability.

Some studies also explored the application of widely used general optimization methods to solve RBS optimal control problems, such as dynamic programming (DP) [16,17] and heuristic algorithms [18]. DP is inherently designed for discrete states, making it less suitable for the continuous battery state space of RBSs. On the other hand, heuristic algorithms struggle to guarantee both optimality and convergence. More importantly, as the complexity of sequential decision problems increases with the number of actions and states, the computational time for both methods grows at a rate far exceeding linear scaling.

To further reduce the complexities associated with the sequential decision-making in RBS optimal control problems, Deep Reinforcement Learning (DRL), built on Markov decision processes, offers a control framework that transforms the originally combinatorial policy search problem into one scaling linearly with the number of decision steps, states, and actions [19]. Although DRL requires substantial training data and time to develop a policy or value network, its computational demands during real-time operation are minimal compared to the above DP and heuristic methods. After training, the DRL model no longer needs to compute the entire optimization trajectory in advance. Instead, it generates a suitable action in real time based solely on the current system state using a compact neural network with high computational efficiency. By repeatedly executing this policy network, multi-step optimization can be achieved. This high efficiency makes the DRL-based approach well-suited for real-time applications on embedded devices. Jeon et al. [20] utilized proximal policy optimization (PPO) method to enhance the discharge efficiency of RBS considering the cell balancing and rate capacity effect, demonstrating better performance than rule-based methods. Yang et al. [21] employed a model-free deep Q-network (DQN) to dynamically improve cell balancing in an RBS. While the DRL agent interacts directly with the real-world environment to collect necessary data, the training speed is significantly limited by data collection, and operational safety is not guaranteed. Silvio et al. [22] also optimized cell balancing in an RBS using DQN, showing that DRL approaches outperform rule-based and heuristic methods while exhibiting strong generalization capabilities.

While the above methods have been applied to RBS optimal control problems, solution search must be conducted within sufficiently small search spaces. This is typically achieved by limiting the system size or control horizon length, simplifying system modeling based on certain assumptions, or restricting the number of switch operations per decision step. However, when binary switch states are treated as control variables, the size of the search space grows exponentially with the number of switches and decision steps. For example, a 20-switch RBS over a three-step horizon results in a search space of $2^{60} \approx 1.15 \times 10^{18}$. Faced with such vast discrete action spaces, DP requires too long computation time to achieve an optimal solution. Similarly, heuristic methods also demand substantial time while often yielding only low-quality solutions. DRL, despite its advantages, still struggles in large discrete action spaces. In value-based methods like DQN, the complexity of the value network scales linearly with the action space size, making training impractical for large-scale problems. Meanwhile, policy-based methods like PPO struggle to generalize effectively across actions, often leading to suboptimal performance [23].

To address the critical challenges caused by the large-scale discrete action space in this study, Wolpertinger architecture (WA) [23] based on DRL is introduced to solve RBS optimal control problems. In WA, the discrete actions are first embedded into the continuous space of Deep Deterministic Policy Gradient (DDPG) to enable action generalization. The actions generated by the DDPG actor in the continuous space are then mapped back to the original discrete actions through nearest neighbor search algorithms and inverse embedding. Through this design, WA harnesses the action generalization capabilities of value-based methods while avoiding the computational burden of evaluating the entire action space. Moreover, WA is a single-agent approach more computationally efficient than many MA learning methods [24,25].

When applying WA to solving RBS optimization problems, the first step is to design specific embedding methods to map discrete actions into a continuous space. Thus, three action embedding methods are proposed and tested in this study. Further, to improve the solution optimality, the original agent DDPG is updated by Twin Delayed Deep Deterministic Policy Gradient (TD3) in this study since TD3 has demonstrated superior performance over DDPG in various training scenarios, offering improved stability and reducing overestimation bias [26].

Additionally, to reduce the computational burden for searching the nearest neighbor of the generated action, in this study Hierarchical Navigable Small World (HNSW) is selected for showing significantly higher computational speed in benchmark tests than the originally deployed K-means and K-d tree methods [27]. As will be demonstrated in Section 3, through the proposed embedding design and modifications to WA, the convergence optimality and computational efficiency can be enhanced as compared with DRL methods using original WA, PPO, or DQN. These advantages become even more significant as the action space expands. In addition, experimental results demonstrate that, under identical training conditions, the proposed enhanced WA achieves faster SoC balancing, longer discharge duration, and higher energy output compared to the traditional PPO algorithm, confirming its superior real-time performance and practical applicability.

When deploying the enhanced WA for RBS control, handling exponentially growing discrete action spaces remains one critical challenge [28]. In practical energy storage applications, as the system scales up, the centralized discrete action space explodes exponentially. This rapid expansion easily exceeds physical memory limits, rendering centralized action evaluation computationally intractable [29]. To address this challenge, the latest paradigm in DRL has shifted towards Multi-Agent Reinforcement Learning (MARL) relying on a divide-and-conquer strategy [30,31]. Recent studies have also demonstrated the promising application of MARL to decentralized control in multi-module battery systems [32]. More specifically, the integration of MARL with action space embedding (i.e., the MA+WA framework) has been proven highly feasible in tackling massive discrete control problems, such as multi-user beamforming in massive MIMO systems [33,34]. However, these application scenarios of existing MA+WA methods fundamentally differ from the RBS control. In communication systems, an invalid or sub-optimal action merely results in a performance penalty (e.g., signal degradation or interference). In contrast, an invalid topological action in RBS can trigger catastrophic safety issues, such as short-circuits or dangerous voltage surges.

To bridge this critical safety gap, a safety-guaranteed MA+WA framework tailored for large-scale RBS control is further proposed in this study. First, building upon the feasible solution space principles established in our previous work, we allocate to each decentralized agent only the valid switch configurations corresponding to its local position. This fundamentally eliminates the vast majority of impossible or dangerous actions at the local level. Secondly, to address the dangerous configurations that may still emerge when local actions are stitched together, we introduce a robust fail-safe mechanism. Utilizing high-speed topology detection algorithm, any dangerous action is instantaneously identified and forcefully replaced by its nearest safe action within the global feasible space. This dual-layer protection guarantees the absolute physical safety of the DRL outputs. Thus, this paper not only fully develops the enhanced WA framework for a small size RBS control, but also successfully propose this highly-scalable and safety-guaranteed MA+WA framework for industrial-level RBSs.

This paper is organized as follows. Section 2 provides a detailed description of the enhanced WA, including the essential virtual environment and embedding methods as well as the proposed modifications. Then, Section 3 presents various case studies along with experimental tests to demonstrate the effectiveness and advantages of the enhanced WA in addressing the critical challenge of large-scale discrete action spaces. Finally, conclusions are drawn in Section 4.

2. Enhanced Wolpertinger architecture for solving RBS optimal control problems

2.1. Wolpertinger architecture (WA)

The RBS optimal control problem is formulated as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ [35]. In this tuple, the definitions of the state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ depend on the specific RBS optimal control problem. Therefore, they will be explicitly defined for the case study in Section 3. Moreover, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition probability distribution determined by the system dynamics. $\gamma \in [0, 1]$ is the discount factor that accounts for the importance of future rewards. As discussed in Section 1, WA is selected to solve the formulated problem.

The original WA was proposed based on the DDPG algorithm [23]. DDPG inherently operates in a continuous action space, rendering it incompatible with discrete action spaces in RBS optimal control problems. To bridge this gap, an embedding mechanism needs to be developed to map the discrete action space $\mathcal{A}$ into a continuous space $\mathcal{C}$, yielding the embedded action space $\hat{\mathcal{A}}$. The structure of WA based on DDPG is illustrated in Fig. 1. For each decision step, given the initial system state, the actor network in DDPG first generates a proto-action within the continuous action space $\mathcal{C}$. Subsequently, Approximate Nearest Neighbors (ANN) search is deployed to identify K nearest neighbors of this proto-action within the embedded action space $\hat{\mathcal{A}}$. These K candidate actions are evaluated by the critic network, and the one with the highest Q-value is selected as the best embedded action. This embedded action is then inversely embedded into the original discrete action space $\mathcal{A}$ and executed in the environment, yielding a corresponding reward and the next state, i.e., the initial system state for the next decision step. Finally, the transition tuple, comprising the state, proto-action, reward, and the next state, is stored in the replay buffer for further training. The actor and critic networks are then updated following the DDPG training framework.

In the WA structure shown in Fig. 1, the essential environment and embedding methods for constructing WA will be developed in Sections 2.2 and 2.3, respectively. To further enhance the optimization performance of WA, the original DDPG-based agent and ANN methods will be improved in Section 2.4.

2.2. RBS simulation environment

In the WA structure shown in Fig. 1, an environment needs to be constructed to generate the system response to a specific action for each decision step. For the optimal control of RBSs, such as the 10-cell RBS illustrated in Fig. 2, the on or off switch states are regarded as decision variables, and, hence, the action is characterized by a vector consisting of all switch states (SSV). For each decision step, given the initial system state, the state update and reward in response to the given action can be obtained by an RBS experiment or a high-fidelity simulation. Since a huge number of combinations of initial states and potential actions need to be tested and evaluated during the training process, it becomes too time- and energy-consuming to accomplish this task through RBS experiments. Moreover, the majority of SSVs in the search space are infeasible, which can lead to battery faults such as short circuits or undesired open circuits. Testing such infeasible actions by experiments can incur safety issues. Thus, a simulation-based virtual environment is developed using the unified equivalent circuit model for RBSs proposed in our recent work [36]. The high simulation accuracy of this model has also been validated by experimental data. Thanks to this unified RBS model, there is no need to reconstruct the system model due to dynamic system reconfiguration.

2.3. Action embedding methods

As illustrated in Fig. 1, the DDPG method, receiving inputs from both the environment and the replay buffer, is designed to operate within a continuous action space. While the state information of battery cells, such as voltages or currents of modeling components, consists of continuous values, the discrete-valued SSVs form a discrete action space, which is unsuitable for training the DDPG-based agent. Thus, an embedding method needs to be developed to map the discrete action space into a continuous space. In the absence of a universal embedding method for all situations, three action embedding methods are proposed as follows.

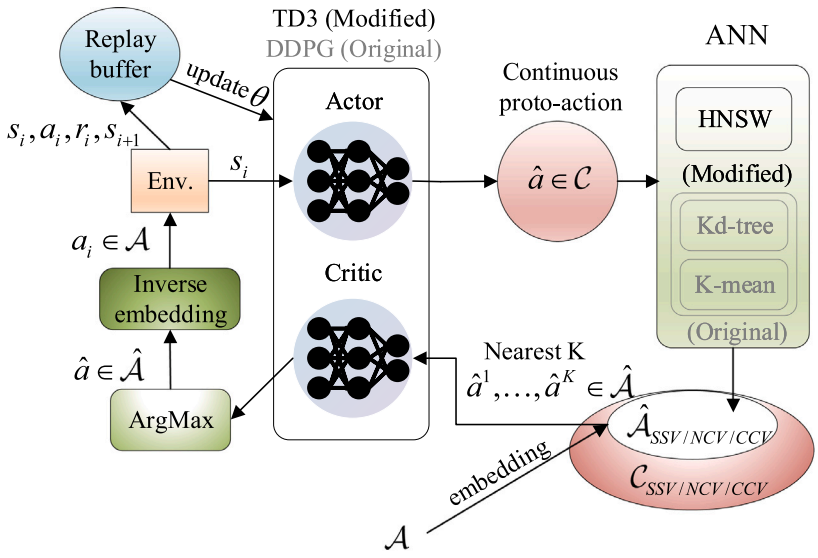


Fig. 1. Structure of the enhanced Wolpertinger architecture.

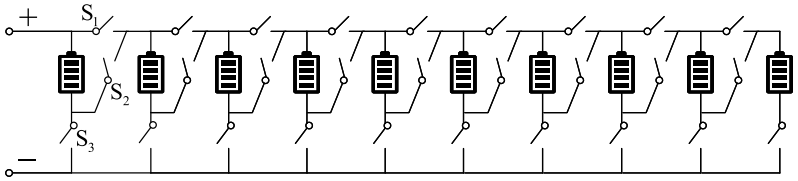


Fig. 2. Diagram of a 10-cell RBS with 3 switches associated with each cell except the last one.

2.3.1. Switch state vector method

Each SSV in the original discrete action space $\mathcal{A}$ can be directly mapped to the same SSV in a continuous action space $\mathcal{C}_{SSV}$ as defined below. The SSVs in $\mathcal{C}_{SSV}$ corresponding to all SSVs in $\mathcal{A}$ constitute an embedded action space denoted by $\hat{\mathcal{A}}_{SSV}$.

$$\mathcal{C}_{SSV} = [0,1]^{N_S}, \tag{1}$$

$$\hat{\mathcal{A}}_{SSV} = \mathcal{A}, \tag{2}$$

where N_S denotes the total number of switches in the RBS.

2.3.2. Cell current vector method

In an RBS, the SSV in the original discrete action space $\mathcal{A}$ specifies the connection structure of all cells, i.e., the system configuration, which influences the current distribution among cells, described by a cell current vector (CCV). The CCV determines the operating performance of each battery cell and the entire battery system. Motivated by this, the discrete space $\mathcal{A}$ for SSVs can be embedded into a continuous space for CCVs, denoted by $\mathcal{C}_{CCV}$, as defined below. The CCVs in $\mathcal{C}_{CCV}$ corresponding to all SSVs in $\mathcal{A}$ constitute an embedded action space denoted by $\hat{\mathcal{A}}_{CCV}$.

$$\mathcal{C}_{CCV} = [i_{\text{low}}, i_{\text{high}}]^{N_B}, \tag{3}$$

$$\hat{\mathcal{A}}_{CCV} = \left\{ CCV \mid CCV = \Phi_I(x, SSV, u_{\text{sys}}),\ SSV \in \mathcal{A} \right\}. \tag{4}$$

where i_{low} and i_{high} denote the minimum and maximum allowable cell currents, respectively, N_B the number of battery cells, Φ_I the cell current output function based on the unified RBS model introduced in Section 2.2, x the battery state vector, and u_{sys} the system's input current or power.

It can be seen that the defined continuous space $\mathcal{C}_{CCV}$ has a dimension of N_B, much smaller than that of the original discrete action space $\mathcal{A}$, i.e., N_S, especially for RBSs with multiple switches associated with each cell. For instance, $N_B = \frac{1}{3}(N_S + 2)$ for the RBS illustrated in Fig. 2. Thus, in addition to generating a continuous action space for the DDPG, this embedding method can significantly reduce the dimension of the action space. This will enhance the computational efficiency, especially for the ANN search process.

Moreover, as shown in (4), the battery cell current distribution is also dependent on the battery state vector x and the system input. However, it is too computationally expensive to rebuild the embedded set $\hat{\mathcal{A}}$ and to update the ANN search index for all possible battery state vectors and system inputs. Alternatively, a fixed battery state vector and a constant system input are deployed during the training period so that the ANN search index only needs to be constructed and precomputed once. Two options for the fixed battery state vector are considered. One is a vector composed of all cells' initial SoCs, and the other is a vector of the same size with each entry equal to the mean of all cells' SoCs. These two embedded methods, referred to as CCV1 and CCV2, respectively, will be evaluated and compared in Section 3.4.

2.3.3. Neighboring connection vector method

The SSV of an RBS determines the connection patterns between neighboring battery cells, described by a vector consisting of each cell's neighboring connection pattern (NCV). Specifically, as shown in Fig. 2, each cell can get connected with its next neighboring cell in series (SR) or parallel (PL). The last non-bypassed (LN) cell is directly connected with the system's negative terminal. Moreover, a bypassed (BP) cell has no connection with others. Taking the RBS design in Fig. 2 with three switches per cell as an example, these four types of neighboring connection patterns between battery cells are indexed from 0 to 3 in Table 1, where their corresponding states of switches around the cell (referred to as the cell SSV) are also specified. Thus, as defined below, the original SSV space $\mathcal{A}$ can be mapped to a NCV space, denoted by $\hat{\mathcal{A}}_{NCV}$, and $\hat{\mathcal{A}}_{NCV}$ is further expanded to a continuous-value space $\mathcal{C}_{NCV}$ to facilitate the training of DDPG-based agent.

$$\hat{\mathcal{A}}_{NCV} = \{ NCV \mid NCV = \text{map}(SSV),\ SSV \in \mathcal{A} \}, \tag{5}$$

Table 1
Cell neighboring connection patterns and the corresponding cell SSVs for the RBS design in Fig. 2.

Index	Cell neighboring connection pattern	Cell SSV
0	BP: The cell is bypassed.	[1, 0, 0]
1	SR: The cell is series-connected with the next one.	[0, 1, 0]
2	PL: The cell is parallel-connected with the next one.	[1, 0, 1]
3	LN: The cell is the last non-bypassed one.	[0, 0, 1]

$$C_{NCV} = [0, 3]^{N_B}. \tag{6}$$

Since the neighboring connection pattern is defined for each battery cell, $\hat{\mathcal{A}}_{NCV}$ has a dimension of N_B, which is the same as that of $\hat{\mathcal{A}}_{CCV}$ and about one third of the original SSV space's dimension. This dimension reduction can enhance the search efficiency of ANN.

The performance of these three embedding methods will be assessed and compared in Section 3.7. When the critic scores the embedded actions in Fig. 3, the embedded action closest to the proto-action may not yield the highest Q-value. To address this, the K nearest neighbors in the embedded action space are first identified as candidates, and the one with the highest Q-value is selected as the optimal action. In theory, increasing K can improve the Q-value of the selected action. However, this also increases the search time for ANN and the critic's evaluation time. Therefore, in the experimental setup, K will be treated as a variable when evaluating the three proposed embedding methods, and its impact on training speed and performance will be analyzed in Section 3.5.

2.4. Modifications to WA

After developing the essential simulation environment and embedding methods for WA, the DRL agent and ANN algorithm in the original WA are further improved to pursue enhanced training performance and computational efficiency.

To enhance the training robustness and performance, the DDPG-based agent in the original WA is replaced with one based on the Twin Delayed Deep Deterministic Policy Gradient (TD3). TD3 is an improved version of DDPG, designed to mitigate overestimation bias and enhance learning efficiency [26]. TD3 significantly outperforms DDPG in continuous control tasks, e.g., within OpenAI Gym, due to introducing three key innovations: clipped double Q-learning to reduce Q-value overestimation, delayed target network updates to improve policy stability, and policy smoothing regularization to prevent overfitting to Q-function noise. Since TD3 still follows the Deterministic Policy Gradient framework, just like DDPG, it is fully compatible with WA. The network update strategy of the enhanced WA is introduced in Appendix. Details on the TD3 implementation and hyperparameter settings are provided in Section 3.2.4, and its performance in WA is compared with that of the DDPG in Section 3.4.

Moreover, when searching for the nearest discrete actions of the continuous proto-action in the WA illustrated in Fig. 1, the original ANN search method is updated to enhance the search efficiency. Since exact k-nearest neighbors search is highly time-consuming for large-scale and high-dimensional data, ANN algorithms [37] have become a common approach in both industry and academia, offering a substantial improvement in retrieval speed at the cost of a slight loss in accuracy. The original WA utilizes the Fast Library for Approximate Nearest Neighbors (FLANN) [38], which employs a so-called "auto-tuned" algorithm to automatically select the appropriate ANN algorithms, such as K-means and K-d tree, along with the optimal parameters based on the target recall accuracy. However, FLANN's slow search speed may impact the training efficiency, particularly when handling large values of K. To enhance the computational efficiency without compromising accuracy, the original FLANN is replaced with the Hierarchical Navigable Small World (HNSW) algorithm. HNSW is an advanced ANN search method that uses a hierarchical multilayer graph structure to allow efficient traversal across different distance scales, heuristic neighbor selection to minimize unnecessary distance computations, and an optimized search process to significantly accelerate nearest neighbor retrieval, particularly in high-dimensional large-scale datasets [39]. Compared to the K-d tree in FLANN, HNSW demonstrates superior efficiency and accuracy, as confirmed by benchmarking studies [27]. Detailed performance comparison of FLANN and HNSW will be presented in Section 3.3.

3. Case studies for performance evaluation

3.1. General settings for case studies

To evaluate the performance of using the proposed enhanced WA for solving RBS optimal control problems, a series of case studies aimed at maximizing the discharge energy are investigated in this section. For the RBS design illustrated in Fig. 2, both 10-cell and 15-cell systems are tested using the high-fidelity simulation environment introduced in Section 2.2 on a desktop computer with an Intel Xeon W-2245 CPU.

In each case study, each lithium-ion battery cell has a nominal charge capacity of 2 Ah. Battery cells with inconsistent SoCs and model parameter values are discharged with a constant power of 70 W until any cell's SoC falls below 5%. During this process, i.e., the duration of each decision step, denoted by T_{ds}, is set to 20 s, the system is reconfigured every 20 s according to the obtained optimization result. The total number of decision steps involved in the discharge process is denoted by N_{ds}. Then, the total discharge time is $N_{ds}T_{ds}$. Since the discharge power is constant, maximizing the discharge energy is equivalent to maximizing the total discharge time. Because the initial total available energy of the battery pack is fixed, maximizing the total discharge time mathematically guarantees maximizing the total delivered energy, which directly translates to maximizing the system's overall capacity utilization rate (i.e., energy efficiency). Furthermore, the discharge process of a battery system is constrained by the "bucket effect"—the entire system must terminate discharge the moment the weakest cell hits the predefined lower SoC boundary. Therefore, an extended total discharge time inherently drives the agent to actively balance the cells, ensuring that the SoC levels of all cells approach the depletion limit as uniformly as possible before termination. Thus, maximizing $N_{ds}T_{ds}$ serves as an effective macroscopic reflection of both energy efficiency and balancing performance. Thus, the optimization problem is formulated as follows.

$$\max_{SSV_1,\ldots,SSV_{N_{ds}}} \quad J = N_{ds}T_{ds} \tag{7}$$

$$x_k = \Phi_x(x_{k-1}, SSV_k, P_k), \tag{8}$$

$$SSV_k \in \{0, 1\}^{N_B}, \tag{9}$$

$$v_t^{\mathrm{lb}} \le v_{t,k} \le v_t^{\mathrm{ub}}, \tag{10}$$

$$I_B^{\mathrm{lb}} \le I_{n,k} \le I_B^{\mathrm{ub}}, \tag{11}$$

$$SoC^{\mathrm{lb}} \le SoC_{n,k} \le SoC^{\mathrm{ub}}, \tag{12}$$

$$k = 1, 2, \ldots, N_{ds},\ n = 1, 2, \ldots, N_B. \tag{13}$$

In this formulation, the decision steps are indexed by k, $1 \le k \le N_{ds}$. At the kth decision step, x_k, SSV_k, and P_k denote the switch state vector, system state vector, and system load power, respectively. $\Phi_x(x_{k-1}, SSV_k, P_k)$ denotes the system's state transition function. Moreover, at the kth decision step, the system's terminal voltage, the nth cell's current, and the nth cell's SoC are denoted by $v_{t,k}$, $I_{n,k}$, and $SoC_{n,k}$, respectively. They must fall within their own lower and upper bounds, indicated by lb and ub in the superscripts, respectively. These specific bounds are strictly determined by the physical hardware specifications. For instance, the upper bound of the cell current ($I_B^{\mathrm{ub}} = 9$ A) is explicitly selected based on the maximum continuous discharge current

rating provided in commercial lithium-ion cell datasheets to prevent accelerated battery degradation and safety issues. The upper bound of the cell SoC ($SoC^{\mathrm{ub}} = 1$) corresponds to the absolute maximum cell voltage limit (e.g., 4.2 V) to prevent overcharging. The system voltage is rigidly bounded on both ends: the lower bound v_t^{lb} is dictated by the minimum input voltage requirement of the downstream DC load or inverter, while the upper bound v_t^{ub} is specified by the maximum design voltage of the battery system and the maximum allowable load voltage.

To clarify the control logic during the discharge process, it is crucial to distinguish between safety constraint violations and the normal termination signal. Constraints on system voltage (10) and cell current (11) are strict safety constraints. If a poor topological decision causes a violation of these limits, the system fails to safely support the load, triggering an early termination with a severe penalty to the DRL agent. Conversely, reaching the lower SoC limit serves as the normal End-of-Discharge (EoD) signal. If the episode ends simply due to reaching the cell SoC limit without violating other voltage and current constraints, it is considered a normal termination of the discharge episode, and no penalty is needed.

3.2. Specific settings for the enhanced WA

To solve the formulated RBS optimal control problem using DRL, the enhanced WA is adopted as the learning framework. The specific settings of WA are detailed in this subsection, including the state space, action space, reward function, embedded constraint-handling mechanism, and DRL agents.

It is important to note that in the proposed enhanced WA, the network is trained on specified ranges of cell SoCs and load power values. When deploying the trained network for optimization, it can generate results for any SoC and load power within those trained ranges. In this study, since the primary objective of the case studies is to validate the effectiveness of the enhanced WA, the power profile is kept constant during both training and deployment for simplification. Additionally, operational constraints, such as voltage and current limits, are enforced by incorporating penalty terms in the reward design, as detailed in Section 3.2.3.

3.2.1. State space

The state s is defined as the SoCs of all battery cells. Cell SoCs can be estimated using commonly used methods, such as Coulomb counting or Kalman Filter-based methods. Thus, the estimation of cell SoCs is detailed here.

3.2.2. Action space

The system configuration of an RBS is determined by the switch state vector (SSV). As aforementioned in Section 2.2, among all possible actions, many SSVs correspond to infeasible system configurations incurring local battery faults or failing to meet the requirement of system voltage. Moreover, different SSVs might lead to the same system configuration for certain RBSs. Therefore, the complete action space is narrowed down to the feasible action space $\mathcal{F}$ corresponding to only feasible configurations through the automatic algorithm proposed in our recent work [29]. The feasible action space $\mathcal{F}$ is defined as the set of $SSV \in \{0,1\}^n$ that ensures the SSV prevents cell short circuits, system open circuits, and undesired terminal voltages. The action space $\mathcal{A}$ is now a discrete space composed of these feasible SSVs from $\mathcal{F}$, defined as follows

$$\mathcal{A} = \{a \mid a \in \mathcal{F}\} \tag{14}$$

The action space size can be reduced from 2^{28} to 2116 for the 10-cell RBS, and from 2^{43} to 147,594 for the 15-cell RBS. Despite this substantial reduction, the discrete action space remains large enough to pose significant challenges for standard DRL methods.

3.2.3. Reward function

As shown in the objective function in (7), the RBS optimal control is aimed at extending the total discharge time. Thus, the reward function of each decision step is defined by the discharge time during this step, i.e., $r_k = T_{ds}$. Then, during the training process, the rewards of all steps involved in the episode add up to the episode reward, i.e., the total discharge time.

In addition to excluding any actions leading to cell short circuits, system open circuits, or undesired system terminal voltage levels by constructing the feasible action space, the safety constraints in (10), (11), and (12) must also be rigorously monitored. As established in Section 3.1, although the lower SoC limit in (12) is fundamentally a safety boundary to prevent over-discharge, it also serves as the standard End-of-Discharge (EoD) indicator. Thus, the SoC limit in (12) is enforced during the operation simulation in the environment. Once any cell SoC reaching the lower limit, the discharge episode is terminated, and the "done" flag is activated. Conversely, the voltage and current limits in (10) and (11) are action-dependent safety constraints. To ensure the learned policy strictly penalizes unsafe topological decisions that violate these physical boundaries, a constraint-aware modification to the original reward function at each decision step is proposed as follows.

$$r_k = \begin{cases} T_{ds} - \lambda, & \text{if any action-dependent safety constraint} \\ & \text{in (10) or (11) is violated,} \\ T_{ds}, & \text{otherwise,} \end{cases} \tag{15}$$

where λ is a positive constant much larger than T_{ds}.

As seen from this reward function design, a normal reward of T_{ds} is assigned when the system operates safely. When a cell eventually depletes (i.e., reaches the normal EoD threshold $SoC^{\mathrm{lb}} = 0.05$), the episode concludes naturally, and no penalty is assigned since the discharge goal has been successfully achieved. In contrast, if the DRL agent makes a poor topological decision that causes a system voltage falling below the required load level (10) or causes a cell current exceeding its upper limit (11), the battery system fails to safely support the load and an abnormal halt is triggered. Consequently, a severe penalty ($-\lambda$) is actively imposed in such cases to strongly discourage such unsafe actions.

Since the agent tries to maximize the following discounted return for each decision step,

$$G_k = \sum_{j=k}^{N_{ds}} \gamma^{j-k} r_j, \tag{16}$$

any state–action trajectory including such a negatively rewarded action yields a low return. Therefore, through this constraint-aware reward function design, the learning process can be effectively steered toward policies that satisfy all operational constraints and extend the total energy delivery.

3.2.4. Agent implementation and hyperparameters

To compare the performance of the TD3 agent deployed in the enhanced WA with other DRL agents, several agents are tested and compared in the case studies. TD3 and DDPG are implemented in strict accordance with the original TD3 proposed in [26], while PPO follows OpenAI's open-source implementation [40]. Given the wide range of DQN variants, the Rainbow DQN [41] is selected to ensure a robust and credible comparison. Due to integrating multiple enhancements, the Rainbow DQN has demonstrated the capability of achieving state-of-the-art performance in terms of data efficiency and final results on the Atari 2600 benchmark.

The default training hyperparameters for each agent are provided in Table 2, while any unspecified hyperparameters remain consistent with their original implementations. To mitigate dependence on initial policy parameters, all agents follow a purely exploratory policy for the first 2000 training steps, as outlined in [26].

Table 2
Main hyperparameters.

	Hidden layer	learning rate	Batch size	Discount factor	Total steps	Reference
TD3, DDPG, PPO	256 × 256 for Actor and Critic	3×10^{-4}	256	0.99	10^6	[26,40]
DQN	256 × 256	3×10^{-4}	256	0.99	–	[41]

Table 3
HNSW parameters and test performance.

M	efConstruction	Number of threads	Batch size	efSearch	K	Total query time (ms)	Recall accuracy
20	600	8	256	5	5	2.99	0.934
				50	50	6.98	0.979
				100	100	9.97	0.988
				1000	1000	79.79	0.998

Table 4
Performance comparison of HNSW and K-means.

	Total query time (ms)				Recall accuracy			
K	5	50	100	1000	5	50	100	1000
HNSW	4	12	20	151	0.995	0.999	0.999	0.999
K-means	146	196	204	472	0.991	0.991	0.991	0.992

3.3. Computational efficiency comparison between HNSW and FLANN

HNSW, implemented using the Non-Metric Space Library (NMSLIB) [39], is introduced into the enhanced WA to improve the ANN search performance. Table 3 presents the HNSW index construction and search parameters along with the query time and recall accuracy for retrieving K nearest neighbors within the embedded action space $\hat{\mathcal{A}}_{CCV}$ of a 15-cell RBS. HNSW supports multi-threaded search to boost computational efficiency, and, hence, the number of threads is set to 8 in the case studies. Parameters "M" and "efConstruction" are essential for building the HNSW index, while parameters "Batch size" and "efSearch" primarily influence the search phase. Specifically, M specifies the maximum number of neighbors in the base and higher layers of the hierarchical graph. A higher value of efConstruction can enhance the graph quality during index construction, resulting in improved search accuracy while increasing the indexing time. Likewise, a higher efSearch value leads to longer query time but enhanced accuracy, as demonstrated in Table 3, and efSearch needs to reach or exceed K due to HNSW's constraints. Therefore, in HNSW case studies, efSearch is set equal to K, and the batch size is set to 256, corresponding to the batch size for training the agent.

To quantitatively evaluate the advantages of HNSW over FLANN, their query time and recall accuracy under different K values are compared in Table 4. In FLANN, K-means is selected by the "auto-tuned" algorithm. To ensure a fair comparison, HNSW uses a single-thread execution to match FLANN's limitation, while other parameters are set to the same values presented in Table 3. As compared with K-means, HNSW can achieve higher recall accuracy while using much shorter query time, especially for small K values such as 5. Thus, substituting the FLANN in the original WA with HNSW can boost the computational efficiency of both the ANN process and the overall training pipeline. Moreover, as shown in Tables 3 and 4, when increasing the number of threads from 1 to 8 in HNSW, the query time can be significantly reduced while the recall accuracy is only slightly compromised. Then, multiple threads are preferred in the setting of HNSW.

3.4. Optimality comparison between original and enhanced WAs

After introducing the above settings of the agent and ANN search method, the total discharge times obtained by the enhanced and original WAs over up to 1 million training steps are compared in Fig. 3 based on a virtual environment for a 10-cell RBS. In all the tests, the episode reward characterized by the total discharge time is evaluated every 5000 training steps. For the comparative evaluation, the initial SoC values are fixed to a specific deterministic array generated from $\mathcal{U}(0.94, 1.0)$ to ensure a fair and reproducible baseline. In both WAs, K is set to 5, and the four embedding methods proposed in Section 2.3, referred to as SSV, CCV1, CCV2, and NCV methods, respectively, are all examined. To focus on comparing the optimality of DDPG and TD3, the accuracy of FLANN in the original WA is set to the same as the HNSW in the enhanced WA.

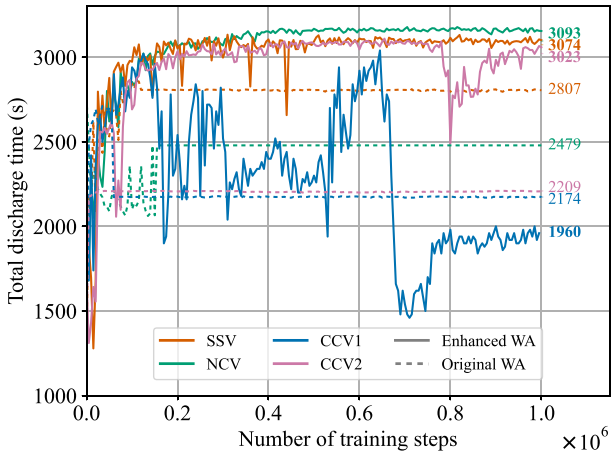


Fig. 3. Training performance comparison between original and enhanced WAs with different embedding methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As shown in Fig. 3, for all embedding methods (indicated by different curve colors) except CCV1, the enhanced WA (described by the solid curve) generally outperforms the original WA (described by the dashed curve) in terms of the resulting total discharge time. Since the ANN search accuracy is set to the same in both WAs, the higher optimality of the enhanced WA can be primarily attributed to replacing the DDPG agent with a TD3 agent, as discussed in Section 2.4. Furthermore, as compared with the CCV1 embedding method based on the initial cell SoCs, the CCV2 method based on the mean of initial cell SoCs yields longer discharge times in both WAs and demonstrates better stability in the enhanced WA. The major reason is that battery cell SoCs gradually get more balanced during the optimal operation, which is closer to the identical cell SoCs assumed in CCV2. Thus, CCV2 is selected in all the remaining analyses of the enhanced WA.

As demonstrated in Table 4 and Fig. 3, the WA enhanced by TD3 and HNSW offers superior agent performance and ANN search efficiency than the original WA. Therefore, the following case studies will be focused on the enhanced WA.

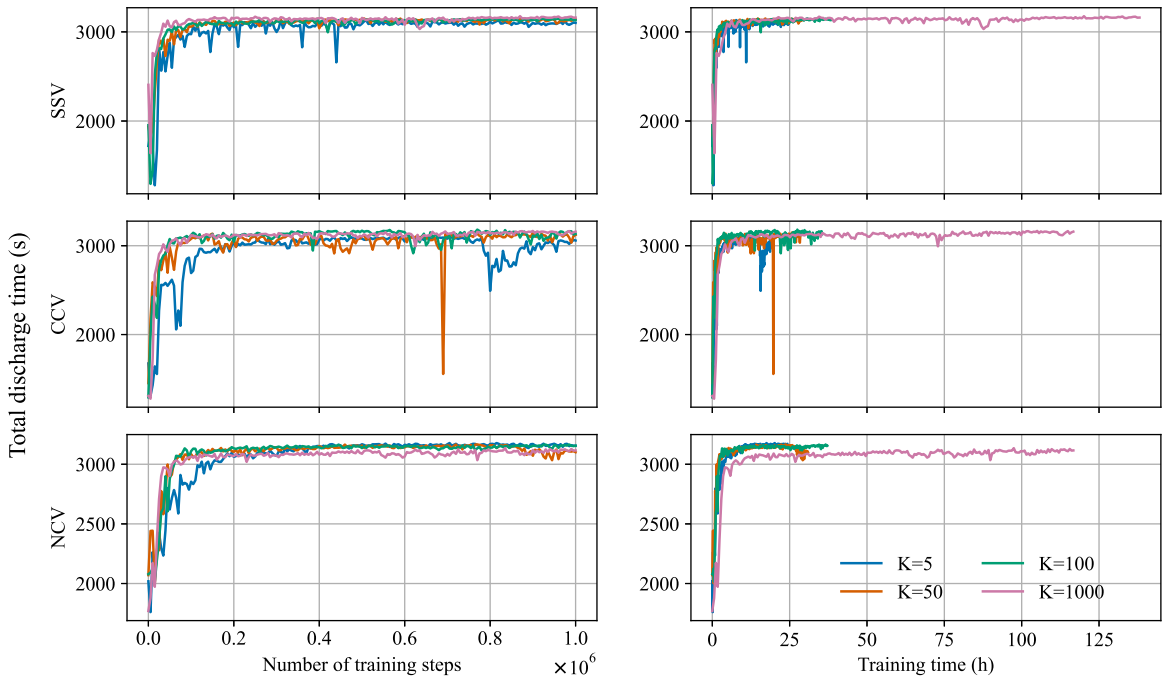


Fig. 4. Comparison of the training performance over training steps and training time for the 10-cell case studies with different embedding methods and various K values.

3.5. Influence of the number of nearest neighbors in HNSW

In the WA shown in Fig. 1, increasing the number of nearest neighbors in HNSW, i.e., the value of K, leads to improved recall accuracy but longer query time, as shown in Tables 3. Moreover, a larger K value expands the action search range for the critic, potentially improving the convergence performance but requiring more computational time for action evaluation. Therefore, the K value needs to be appropriately selected to make a trade-off between the training performance and computational cost. To quantitatively analyze the influence of the number of nearest neighbors in HNSW, different K values are tested across multiple RBS scales. For instance, the test results for the 10-cell RBS using three different embedding methods are presented in Fig. 4.

To facilitate the comparison of convergence performance, several metrics are defined as follows. For each evolution curve of the episode reward (i.e., the total discharge time) over training steps, as illustrated in Fig. 4, the convergence is achieved if the moving average episode reward successfully stabilizes. The moving average over a window of W evaluations, alongside the specific convergence condition (where the absolute differences between consecutive moving averages and their previous W-point windows strictly fall below a tolerance threshold ϵ for N consecutive steps), are mathematically defined as follows.

$$\bar{r}_t(W) = \frac{1}{W} \sum_{\tau=t-W+1}^{t} r_\tau, \tag{17}$$

$$\left| \bar{r}_{t+n}(W) - \bar{r}_{t+n-W}(W) \right| \le \epsilon, \quad \forall\, n = 0, 1, \ldots, N-1, \tag{18}$$

where r_τ denotes the episode reward at the evaluation point τ (evaluated every 5000 training steps, as introduced in Section 3.1), and $\bar{r}_t(W)$ denotes the average of the most recent W episode rewards up to evaluation point t. When the average episode reward meets this stabilization condition, the training step corresponding to the last evaluation point is regarded as the convergence step, the training time taken to reach the convergence step as the convergence time, and the average episode reward at the convergence step as the converged reward.

In these tests, the quantitative convergence parameters are set to $W = 10$, $N = 10$, and $\epsilon = 30$ s. Given that the optimal total discharge time typically spans thousands of seconds, a tolerance of $\epsilon = 30$ s represents a stringent relative error margin of less than 1%. This tight margin ensures that the policy has reached a high-quality optimum rather than experiencing stochastic oscillations. Meanwhile, requiring the variation to remain within this margin for $N = 10$ consecutive evaluations provides high statistical confidence, mathematically confirming that the optimization has settled into a stable state and effectively preventing premature termination. The convergence time and reward for various system sizes, embedding methods, and K values, are compared in Table 5.

The selection of the optimal number of nearest neighbors, K, is examined for increasing RBS sizes, as detailed in Table 5. A larger K expands the exploration boundary for seeking higher rewards but significantly increases the ANN search overhead. To establish a quantitative parameter-selection guideline for various system sizes, a comprehensive evaluation score, denoted by J_K, is introduced as follows, in which the max–min normalization is used to ensure that the metric is strictly bounded within $(0, 1]$.

$$J_K = \omega_R \cdot \left(\frac{R_K}{R_{\max}} \right) + \omega_T \cdot \left(\frac{T_{\min}}{T_K} \right), \tag{19}$$

where R_K and T_K represent the final reward and convergence time for a given K, respectively. $R_{\max}$ and $T_{\min}$ are the highest reward and the shortest convergence time observed for the same system size, e.g., $R_{\max} = 2202.1$ s and $T_{\min} = 3.14$ h for the 7-cell system tests. The weight coefficients are set to $\omega_R = 0.7$ and $\omega_T = 0.3$, prioritizing extending discharge time over saving computational time. Based on this quantitative metric, the preferred K value along with the corresponding embedding method can be suggested for various system scales. By comprehensively considering both the evaluation score J_K and its stability over K, $K \in [50, 100]$ paired with the NCV method is suggested for the 7-cell and 15-cell systems. For the 10-cell system, $K \in [50, 100]$ utilizing SSV provides the best performance, whereas the 12-cell system benefits from a broader search radius with $K \in [100, 1000]$ under the CCV method.

These results indicate that under the action space complexities explored in this study, anchoring K at these suggested values universally

Table 5
Quantitative convergence metrics and comprehensive evaluation scores across various RBS sizes.

Cell (N_B)	Action space size	Embedding method	K	Convergence time T_K (h)	Reward R_K (s)	Score J_K
7	102	NCV	5/50/100	4.00/3.53/3.14	2190.0/2193.5/2185.8	0.93/0.96/0.99
		SSV	5/50/100	8.48/8.26/8.18	2144.6/2197.8/2190.3	0.79/0.81/0.81
		CCV	5/50/100	8.57/8.43/8.15	2170.9/2190.1/2202.1	0.80/0.81/0.82
10	2,116	NCV	5/50/100/1k	6.10/6.00/5.80/19.80	3093/3111/3122/3073	0.94/0.95/0.96/0.76
		SSV	5/50/100/1k	6.80/5.50/5.80/23.50	3074/3103/3096/3147	0.91/0.97/0.96/0.77
		CCV	5/50/100/1k	5.70/8.30/5.20/19.80	3023/3082/3106/3111	0.95/0.87/0.99/0.77
12	12,381	NCV	5/50/100/1k	18.31/22.96/10.30/16.41	3735/3685/3690/3766	0.84/0.80/0.95/0.87
		SSV	5/50/100/1k	21.84/13.64/13.16/13.66	3352/3633/3708/3741	0.75/0.87/0.90/0.89
		CCV	5/50/100/1k	19.04/13.72/10.33/9.07	3596/3613/3649/3655	0.81/0.87/0.94/0.98
15	147,594	NCV	5/50/100/1k	18.10/20.90/23.00/53.20	4512/4515/4566/4622	0.98/0.94/0.93/0.80
		SSV	5/50/100/1k	-/26.20/35.10/63.10	-/4490/4522/4568	-/0.89/0.84/0.78
		CCV	5/50/100/1k	22.70/23.90/30.10/52.90	4532/4610/4601/4618	0.93/0.93/0.88/0.80

Table 6
Quantitative sensitivity analysis of hyperparameters ($N_B = 10$).

Evaluated parameter	Setting	Average reward (s)	Stability (SD) (s)	Peak reward (s)
Adopted baseline	$\tau = 0.005, \sigma = 0.2, lr = 3 \times 10^{-4}$	3125.4	14.0	3158.0
Soft update rate (τ)	0.0005	2015.2	62.4	2696.4
	0.0500	2513.4	6.0	2696.4
Target policy noise (σ)	0.10	3097.4	24.2	3162.4
	0.40	3101.0	20.4	3156.0
Learning rate (lr)	3×10^{-5}	2917.4	84.2	3108.6
	3×10^{-3}	2990.0	72.8	3114.4

provides a sufficient local search radius to achieve high-quality control policies. This effectively decouples the computational search burden from the rapidly expanding action space, indicating the practical feasibility for large-scale RBS deployments.

3.6. Hyperparameter selection

Following the analysis of the nearest neighbor parameter K, a quantitative hyperparameter tuning process is conducted to further optimize the learning performance and ensure the robustness of the proposed DRL-based RBS control framework. Initially, the core hyperparameters are set based on standard empirical values widely recommended in the DRL literature. Subsequently, a localized neighborhood search is performed on the 10-cell RBS to meticulously fine-tune three critical parameters: the soft update rate (τ), the target policy noise (σ), and the learning rate (lr). During these tests, the embedding method and the number of nearest neighbors are fixed to NCV and $K = 100$, respectively.

Table 6 illustrates the localized sensitivity analysis around the finally adopted optimal baseline configuration. To provide a rigorous statistical evaluation, the performance of each deviated configuration is assessed based on the averaged total discharge time and its standard deviation (SD) (representing training stability) over the final 50 training.

As demonstrated in Table 6, the selected baseline configuration ($\tau = 0.005$, $\sigma = 0.2$, and $lr = 3{\times}10^{-4}$) unequivocally achieves the highest steady-state average discharge time (3125.4 s) while maintaining an exceptionally low SD (14.0 s). The results clearly indicate that deviating from this optimal neighborhood inevitably compromises either learning efficiency or training stability. For instance, varying the soft update rate (τ) or learning rate (lr) leads to substantial policy oscillations or severely decelerated convergence. Furthermore, although a lower target policy noise ($\sigma = 0.10$) occasionally yields a marginally higher singular peak reward (3162.4 s), it noticeably deteriorates the overall mean performance and stability. Therefore, this quantitative local sensitivity analysis rigorously justifies the adopted parameter selection, ensuring a robust and well-balanced control policy for the proposed framework.

Furthermore, to ensure a fair and rigorous comparison in the subsequent benchmark analysis, the hyperparameters for the baseline algorithms, i.e., DQN and PPO, are also meticulously tuned to match the RBS control task. For the DQN baseline, the experience replay buffer size and target network update frequency are optimized to 10^5 and 10, respectively, to ensure stable Q-value estimation. For PPO, the clip ratio and target KL divergence are optimized to 0.2 and 0.01 to maintain stable policy updates and prevent catastrophic forgetting. The learning rates for both baselines are optimized at 3×10^{-4}. This consistent optimization ensures that all compared algorithms are evaluated under their respective optimal configurations, providing a solid foundation for the performance benchmark.

3.7. Comparison of the enhanced WA with other DRL methods

To demonstrate the advantage of the proposed enhanced WA over commonly used DRL methods, such as DQN and PPO, the evolution curves of the total discharge time over training time for 7-, 10-, 12-, and 15-cell systems are compared in Fig. 5. The implementation details and hyperparameter values for all methods are provided in Sections 3.6 and 3.2.4. The K values for different embedding methods are selected according to the recommended optimal range established in Section 3.5. Specifically, the K values for the CCV, NCV, and SSV methods are set to 100, 50, and 50 for the 7-cell system; 100, 100, and 50 for the 10-cell system; 100, 100, and 100 for the 12-cell system; and 50, 100, and 50 for the 15-cell system, respectively.

As shown in Fig. 5, the multi-scale evaluation establishes a continuous, four-stage complexity gradient. As the system scales from 7, 10, 12, to 15 cells, the corresponding action space sizes grow exponentially by orders of magnitude, specifically containing 102; 2116; 12,381; and 147,594 discrete actions, respectively. In the small-scale 7-cell scenario with a limited action space, both conventional baselines manage to converge. Specifically, DQN reaches a final reward of 2101 s, while PPO exhibits more acceptable exploration capability (reaching 2181 s), which is only marginally inferior to the proposed enhanced WAs (e.g., 2202 s for CCV). However, as the action space increases by orders of magnitude, the conventional DRL baselines begin to struggle with the increasing complexity, while the three enhanced WAs robustly converge to significantly higher final rewards.

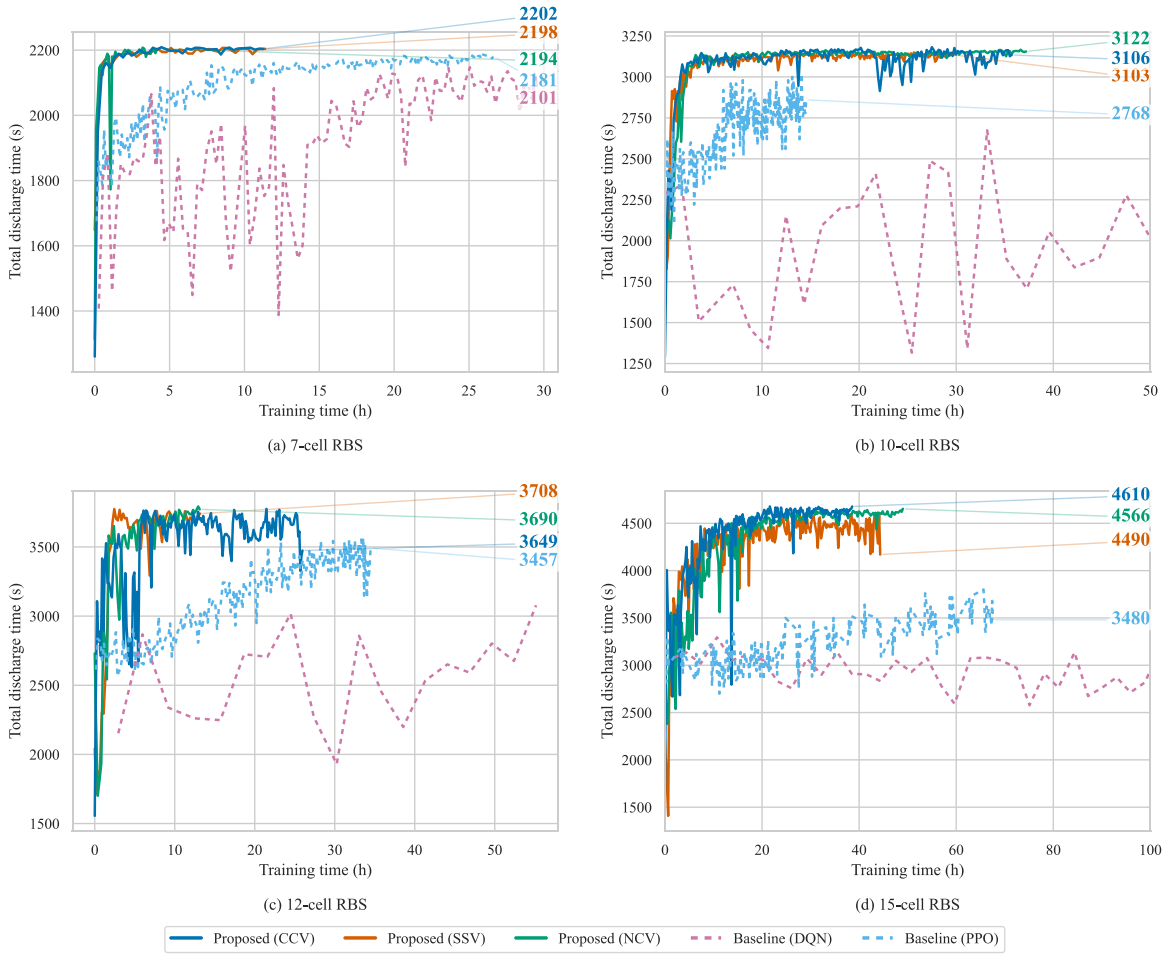


Fig. 5. Comparison of the enhanced WA with DQN and PPO under 7-, 10-, 12-, and 15-cell RBS test environments.

Crucially, such advantages become more pronounced with the system scale. In the 10-cell scenario, PPO plateaus at 2768 s, while the proposed methods securely converge between 3103 s and 3122 s. Similarly, in the 12-cell scenario, PPO struggles to surpass 3457 s, whereas the proposed methods robustly converge to higher values (ranging from 3649 s to 3708 s). In the highly complex 15-cell environment with nearly 150,000 action combinations, PPO plateaus at roughly 3480 s, while the proposed framework securely navigates the high-dimensional space and efficiently isolates the optimal topology, converging to a significantly higher range (4490 s to 4610 s). Notably, although DQN achieves convergence in the initial 7-cell case, it performs significantly worse than other agents across all mid-to-large scale tests. The exponentially growing action space leads to an excessively large output layer for DQN, which renders the network extremely slow and notoriously difficult to train. Ultimately, this results in DQN achieving the lowest final rewards or completely failing to converge according to the criterion specified in Section 3.5. Overall, these multi-scale results reveal a definitive trend: as the size of the action space increases, the performance gap between the proposed enhanced WAs and the conventional baselines widens progressively.

3.8. Algorithm implementation on an embedded RBS control platform

To assess the practical applicability of the proposed algorithm and demonstrate its effectiveness, an embedded RBS control platform was developed, as illustrated in Fig. 6. Specifically, drive circuits for the MOSFET switches were designed and integrated onto printed circuit boards (PCBs). A total of ten battery cells and ten PCBs were interconnected according to the RBS design illustrated in Fig. 2, forming a complete 10-cell RBS system. A power supply provides operating power to the PCBs, while a programmable DC load discharges the battery system under either a constant power of 70 W or a highly fluctuating dynamic power profile. The voltage and current of each cell are acquired by a data acquisition board connected to a controller, which then transmits the measurements to the RK3588 via TCP protocol. The Rockchip RK3588, an industrial-grade embedded computer, runs the pre-trained enhanced WA model and determines the optimal switch states for the upcoming 20-second reconfiguration step based on real-time battery conditions. These switching signals are then sent from the RK3588 to the PCBs, completing one reconfiguration step.

Using the above platform, both the enhanced WA algorithm (with NCV embedding, $K = 100$) and the PPO algorithm are tested in discharge experiments of the RBS. The cutoff conditions, load power, and safety limits are kept consistent with those used during training. To ensure the algorithm does not overfit and to clarify the experimental settings for reproducibility, the initial SoC of the cells during the offline training phase is randomized at the beginning of every episode. However, for the comparative evaluation, the initial SoC values are fixed to a specific deterministic array generated from $\mathcal{U}(0.94, 1.0)$ (as detailed in the first column of Table 7) to ensure a fair and reproducible baseline. Using such initial SoCs, the cell SoC evolution trajectories during the constant power discharge process for the enhanced WA and the PPO algorithms are compared in Fig. 7.

When the enhanced WA algorithm is used to control RBS operation, the battery cell SoCs become balanced at around 200 s and remain generally balanced for the remainder of the test. By rapidly achieving SoC balance across all cells, the algorithm enables simultaneous full discharge of the entire battery system, thus maximizing the total

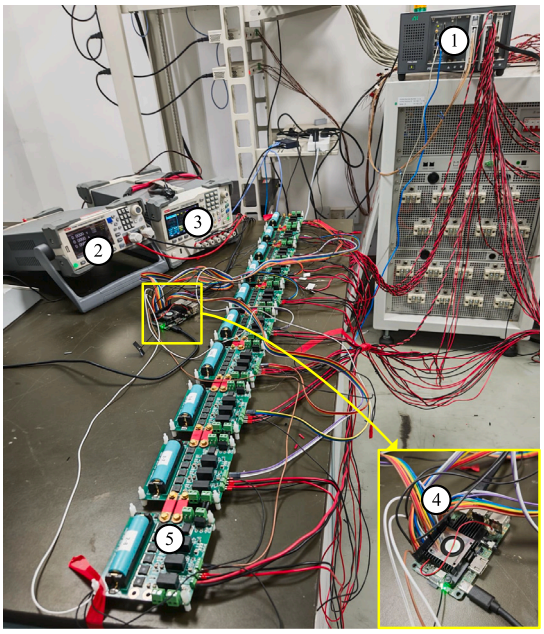


Fig. 6. Embedded control platform for a 10-cell RBS, consisting of ① controller, ② DC load, ③ DC power supply, ④ Rockchip RK3588, and ⑤ battery switching unit.

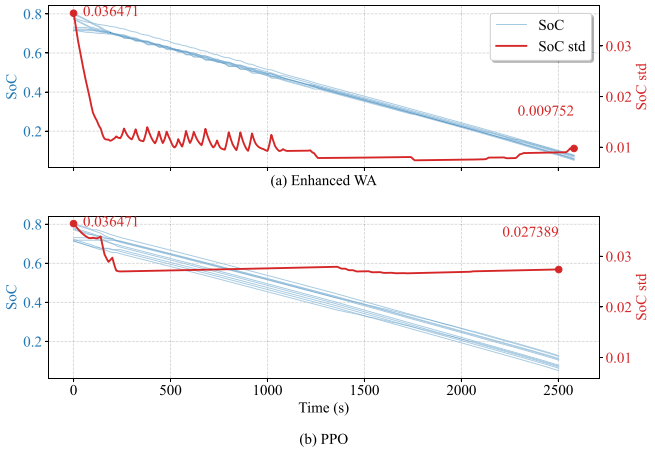


Fig. 7. Comparison of the cell SoC evolution trajectories under different RBS control algorithms during constant power discharge.

Table 7
Initial cell SoC sets and their distributions for embedded control tests.

Cell index	Constant power test $\sim \mathcal{U}(0.94, 1.00)$	Dynamic load test 1 $\sim \mathcal{U}(0.84, 0.95)$	Dynamic load test 2 $\sim \mathcal{U}(0.74, 0.90)$
1	0.9988	0.8875	0.8868
2	0.9717	0.8961	0.8480
3	0.9645	0.8526	0.7474
4	0.9555	0.9493	0.8513
5	0.9743	0.8873	0.8623
6	0.9679	0.8409	0.8029
7	0.9851	0.9263	0.8686
8	0.9782	0.9144	0.7663
9	0.9681	0.8751	0.8756
10	0.9999	0.8899	0.8369

discharge energy. In contrast, the PPO algorithm fails to promote SoC balance, resulting in a shorter discharge duration and reduced total energy output, since some cells reach their lower SoC limits earlier than

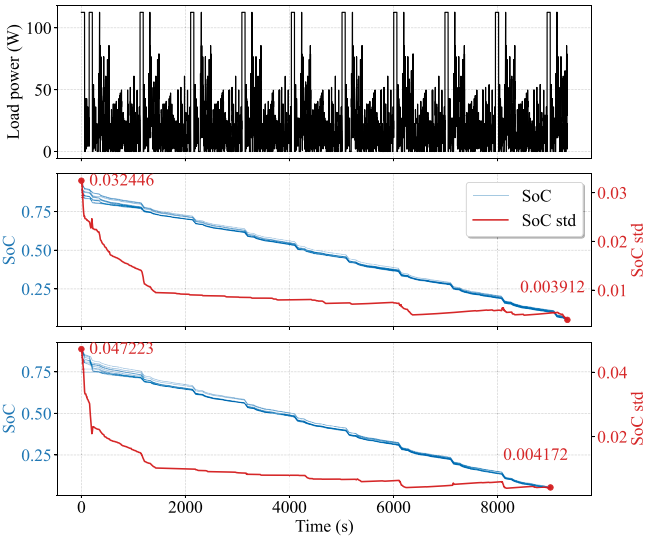


Fig. 8. Performance of the proposed enhanced WA framework under a highly fluctuating dynamic load profile with two distinct, imbalanced initial SoC configurations.

others. Specifically, the enhanced WA achieves approximately 3.2% higher energy delivery compared to PPO. These results demonstrate that the proposed enhanced WA can be embedded to control real battery systems, and it outperforms the traditional DRL method PPO under the same training conditions and duration.

To further evaluate the robustness of the proposed framework against realistic operational scenarios, additional experiments are conducted utilizing a highly fluctuating dynamic load profile derived from Urban Dynamometer Driving Schedule (UDDS), as shown in the top subfigure of Fig. 8. To verify that a single pre-trained model can handle entirely different initial conditions without requiring retraining, two distinct, highly imbalanced initial SoC sets with SDs of 0.032 and 0.047, are tested. For these two tests, the initial cell SoCs are detailed in the second and third columns of Table 7, and the testing results are presented in the middle and bottom subfigures in Fig. 8. Despite the intensely varying discharge currents and the random cell SoC imbalances, the proposed enhanced WA successfully balance the cell SoCs gradually in both test scenarios. The SDs of cell SoCs eventually drop to 0.0039 and 0.0042, respectively. These results conclusively prove that the proposed DRL method is highly robust to dynamic power fluctuations and maintains exceptional real-time control efficacy across various initial cell SoC conditions.

Having verified the closed-loop control performance of the proposed framework under dynamic operating conditions, we further evaluate whether its embedded implementation satisfies the real-time computational and resource requirements. First, 100 decisions are executed to ensure stable program operation, and these results are excluded from the statistics. The following 1000 decisions are then used to measure the decision time, CPU utilization, and memory usage. Each decision covers the computation from the cell SoCs to a feasible SSV output, excluding data acquisition, communication, and physical switch actuation. The measured results are summarized in Table 8.

The longest decision accounts for only 0.040% of the 20-second control interval. During continuous inference, the algorithm utilizes 12.50% of the available eight-core CPU capacity, while its cycle-averaged CPU utilization and peak memory consumption are only 0.0043% and 14.26 MiB, respectively. These results demonstrate that the embedded implementation delivers low-latency performance while requiring only modest computational resources, confirming its suitability for real-time control.

Table 8
Computational cost summary for the embedded implementation.

Average time ± SD (ms)	P99 time (ms)	Longest time (ms)	Continuous algorithm CPU utilization (%)	CPU per control interval (%)	Algorithm peak memory (MiB)
6.96 ± 0.17	7.75	8.03	12.50	0.0043	14.26

Notes: Average time denotes the mean time required for a single decision, where SD represents the standard deviation. P99 time denotes the time within which 99% of all decisions are completed, while the longest time corresponds to the slowest recorded decision. Continuous algorithm CPU utilization represents the percentage of the available eight-core CPU capacity used when decisions are executed continuously without interruption. CPU per 20-second control interval denotes the estimated average CPU utilization when one decision is executed every 20 s. Algorithm peak memory refers to the maximum additional process memory consumed by the algorithm after the required runtime libraries have been loaded, excluding memory used by the operating system and other processes.

3.9. Robustness of the frozen DRL controller under progressive battery aging

Battery aging changes the available capacity and internal resistance of each cell, while cell-to-cell differences may also increase because the cells do not necessarily age at the same rate [42–44]. Consequently, the electrical plant can gradually deviate from the environment used to train a DRL agent, and a policy that remains frozen over the entire battery lifetime may lose its original control and constraint-handling performance. This subsection therefore investigates how progressive parameter drift affects the optimization performance of the pretrained enhanced WA and whether the agent must be updated continuously or only after a finite change in SoH. The purpose of this simulation is to test the robustness of the controller at representative lifetime snapshots, rather than to reproduce the complete electrochemical aging process through physical cycle-aging tests.

The simulations use the experimentally calibrated 10-cell RBS model. The actor and critic networks and the HNSW setting are the same as those used in the embedded-platform tests in Section 3.8. All network parameters are frozen, and neither retraining nor online adaptation is performed during the aging tests. Only the cell capacity Q_i and ohmic resistance $R_{0,i}$ are varied with aging; the polarization parameters, the OCV–SoC relationship, and temperature are held constant so that the influence of these two dominant parameter drifts can be isolated.

Sixteen population-mean SoH levels are evaluated, ranging from 100% to 85% in increments of one percentage point. First, 20 virtual 10-cell packs are generated. To represent cell-to-cell variation, the capacities of the Fresh cells follow a normal distribution with a coefficient of variation (CV, which quantifies the degree of cell-to-cell dispersion) of 0.8%, while their internal resistances follow a lognormal distribution with a CV of 1.94%. At 85% SoH, the capacities follow a Weibull distribution with a shape parameter of 51.68 and a population mean equal to 85% of the nominal Fresh-cell capacity. The internal resistances remain lognormally distributed, with their CV increasing to 3.19%. These distribution types and CV values are based on the cell-population measurements reported by Schuster et al. [42]. Regarding the population-mean internal resistance, the cycle-aging experiments conducted by Devie et al. showed that the internal resistance increased by approximately 70% when the capacity loss reached approximately 25%–30% [43]. Under the linear approximation adopted over the SoH range examined in this study, a capacity loss of 15% corresponds to an estimated internal-resistance increase of approximately 35%–42%. Therefore, the population-mean internal resistance at 85% SoH is set to 40% above its Fresh value. For an intermediate population-mean SoH of $h\%$, the normalized aging progress is defined as $\lambda = (100-h)/15$. The capacity and internal resistance of cell i are then linearly interpolated as $Q_i(h) = (1-\lambda)Q_i(100) + \lambda Q_i(85)$ and $R_{0,i}(h) = (1-\lambda)R_{0,i}(100) + \lambda R_{0,i}(85)$, respectively. Thus, $\lambda = 0$ at $h = 100$ and $\lambda = 1$ at $h = 85$. During sampling, each cell is further constrained to exhibit non-increasing capacity and non-decreasing internal resistance with aging, ensuring that its parameters vary continuously between its own Fresh and 85%-SoH endpoints.

Each virtual pack is tested with ten initial-SoC arrangements. The first arrangement is Dynamic Load Test 1 in Table 7, with a mean SoC of 0.8919 and a standard deviation of 0.0324. The other nine arrangements contain exactly the same ten SoC values but assign them to different cells. This changes which aged cell starts with a high or low SoC without changing the overall initial-SoC distribution. The same ten arrangements are used at every SoH level. Thus, each level contains $20 \times 10 = 200$ discharge episodes, and the complete test contains 3200 episodes. All episodes use the same UDDS-derived load profile, a 20-second topology-reconfiguration interval, and the end-of-discharge condition $\mathrm{SoC}_{\mathrm{cut}} = 0.05$.

The primary balancing metric is the residual SoC above the cutoff, averaged over the initial-SoC arrangements,

$$\bar{R}_{\mathrm{SoC}}(h) = \frac{1}{N_{\mathrm{arr}}} \sum_{j=1}^{N_{\mathrm{arr}}} \left[\sum_{i=1}^{N_{\mathrm{B}}} \mathrm{SoC}_{i,j}^{\mathrm{final}}(h) - N_{\mathrm{B}} \mathrm{SoC}_{\mathrm{cut}} \right]. \tag{20}$$

Here, $i = 1, \ldots, N_{\mathrm{B}}$ indexes the cells in a pack, and $j = 1, \ldots, N_{\mathrm{arr}}$ indexes the initial-SoC arrangements. $\mathrm{SoC}_{\mathrm{cut}} = 0.05$ is the end-of-discharge cutoff. In this section, $N_{\mathrm{B}} = 10$ and $N_{\mathrm{arr}} = 10$. Thus, $\bar{R}_{\mathrm{SoC}}(h)$ represents the average final residual SoC of a virtual pack at SoH h over its ten initial-SoC arrangements. A smaller value indicates less unused charge at the end of discharge. At each SoH level, the 20 pack-level $\bar{R}_{\mathrm{SoC}}$ values are treated as the statistical units ($n = 20$), and their mean and two-sided 95% percentile bootstrap confidence interval are estimated using 10,000 resamples. In addition to $\bar{R}_{\mathrm{SoC}}$, violations of the 9-A cell-current limit were observed during the simulations. Therefore, for every SoH level, the number of affected episodes, cumulative current-violation time, and corrected maximum cell current are recorded directly from all 200 episodes; no bootstrap resampling is applied to these three safety diagnostics.

Fig. 9(a) shows no systematic deterioration in the SoC-balancing metric as SoH decreases. Across the complete 100%–85% SoH range, the mean $\bar{R}_{\mathrm{SoC}}$ remains between 0.03144 and 0.05179. Since $\bar{R}_{\mathrm{SoC}}$ is summed over ten cells, this range corresponds to an average residual of only 0.00314–0.00518 per cell, or approximately 0.314–0.518 SoC percentage points above the 5% cutoff. Moreover, the full minimum-to-maximum variation is 0.02034 in the pack-level metric, equivalent to only 0.00203 per cell, or 0.203 SoC percentage points. The mean values in the Fresh and 85%-SoH conditions are 0.03927 (95% bootstrap confidence interval: 0.03613–0.04308) and 0.03756 (0.03554–0.03979), respectively. Since a smaller residual SoC indicates less unused battery charge, these results confirm that the controller continues to achieve effective SoC balancing despite progressive battery aging.

The current-margin results reveal a different effect. No cell current limit violation occurs from zero to five percentage points of SoH loss. The first violation appears at a six-percentage-point loss, where 1 of 200 episodes exceeds 9.0 A for a total of 3 s and reaches 9.030 A. As the plant drifts further, the number and cumulative duration of violations generally increase, although they are not strictly monotonic because the controller selects discrete switching topologies. For example, at a ten-percentage-point loss, 9 of 200 episodes are affected for a cumulative 92 s, while the largest cumulative duration is 124 s at a twelve-percentage-point loss. More importantly, once violations emerge, the peak-current envelope increases continuously from 9.030 A at a six-percentage-point loss to 9.619 A at a fifteen-percentage-point loss, as shown in Fig. 9(b)–(d).

Taken together, although battery aging has only a limited impact on SoC-balancing performance, the associated parameter variations can

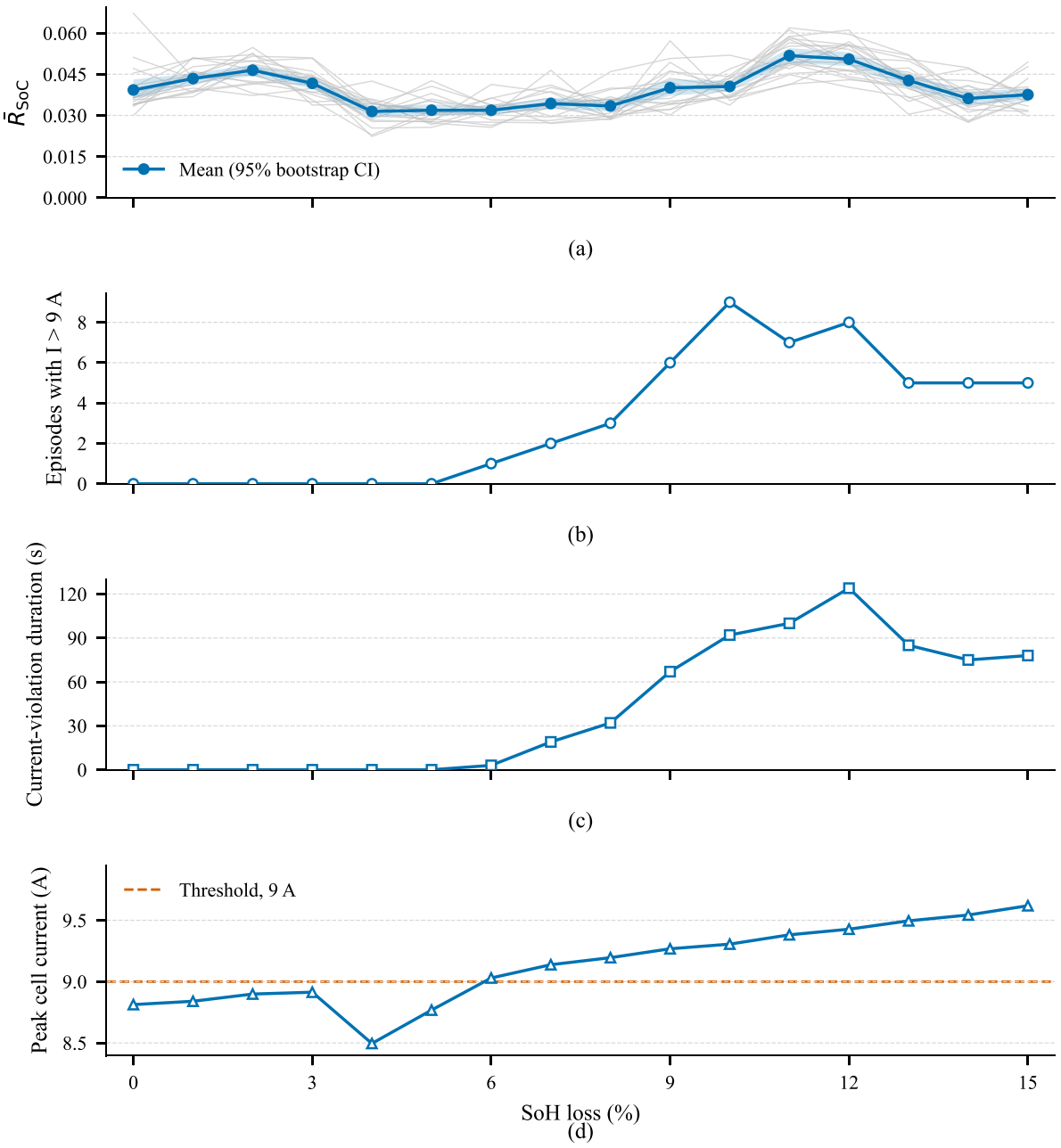


Fig. 9. Performance and current-margin diagnostics of the frozen enhanced WA under progressive battery aging. (a) Average final residual SoC $\bar{R}_{SoC}$. Each thin gray line represents one of the 20 virtual packs after averaging its ten initial-SoC arrangements; the blue line and shaded band show the mean and 95% bootstrap confidence interval across the 20 packs. (b) Number of episodes with a cell current above 9.0 A. (c) Cumulative time above 9.0 A. (d) Peak cell current. Each mean-SoH level contains 200 episodes, and the dashed line in (d) denotes the adopted 9.0-A cell current threshold.

alter the current distribution corresponding to a given state–action pair. When the deviation from the parameters used in the training environment becomes sufficiently large, the pretrained agent may incorrectly determine whether a selected action satisfies the current constraint, thereby reducing the reliability and applicability of the original policy. Nevertheless, the results also show that, when the battery-parameter variations remain limited, the DRL agent can still accurately identify whether the selected actions violate the constraints. Therefore, to maintain the applicability of the policy, it is sufficient to update the virtual environment and retrain or fine-tune the agent before the SoH variation exceeds a predefined threshold.

Under the battery model, load profile, and aging distributions investigated in this study, a change of up to five SoH percentage points causes neither a systematic deterioration in SoC-balancing performance nor a significant cell-current violation. Accordingly, five SoH percentage points can be adopted as a conservative model-review interval for the present controller. Once the cumulative SoH variation reaches this level, or earlier if the measured cell-current margin decreases, the cell parameters in the virtual environment should be re-identified, followed by retraining or fine-tuning of the agent. It should be emphasized that this five-percentage-point interval is an empirical result obtained for the present system and test conditions, rather than a universal battery-aging threshold. Alternatively, operational data collected from deployed systems could be used to continuously update the plant model and adapt the control policy online. However, such an adaptive framework would require explicit constraint handling, safe exploration, and stability guarantees during online updating, and is therefore left as an important direction for future research.

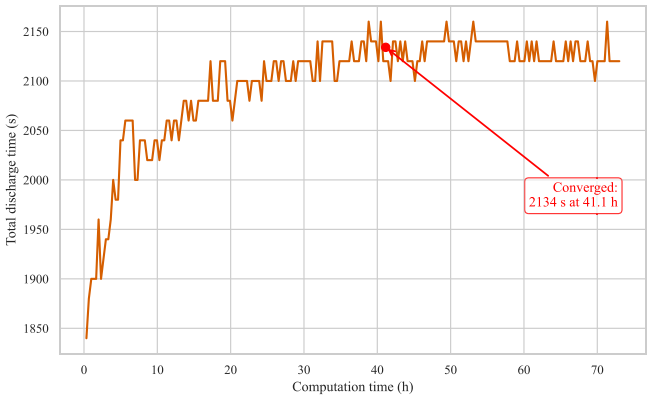


Fig. 10. Genetic Algorithm optimization for the 10-cell RBS.

3.10. Comparison with traditional optimization methods

To comprehensively evaluate the performance of the proposed DRL framework, it is compared with traditional optimization methods. Then, Dynamic Programming (DP) and Genetic Algorithm (GA) are selected as representative baselines for evaluation tests on the 10-cell RBS.

3.10.1. Quantitative analysis of DP intractability

Theoretically, DP guarantees a globally optimal solution by evaluating every possible future trajectory. However, its application to RBS control is fundamentally prohibited by the "curse of dimensionality". To demonstrate this computational bottleneck, consider a 10-cell RBS. If the cell SoC is coarsely discretized with a 5% resolution and the maximum cell SoC difference cannot exceed 20%, all cell SoCs are confined to a sliding window of just 5 contiguous levels. Then, the cell SoC grid consists of $|S_{\text{pruned}}| = 15 \times 5^{10} \approx 1.46 \times 10^8$ states, where 15 is the total number of sliding windows from [5%, 25%] to [75%, 95%]. At each decision step, the DP needs to evaluate $|A| = 2116$ valid actions, as listed in Table 5. For a typical discharge horizon of $H = 100$ steps, computing the DP value function across this pruned grid requires the following number of evaluations: $N_{\text{eval}} = H \times |S_{\text{pruned}}| \times |A| = 100 \times (1.46 \times 10^8) \times 2116 \approx 3.23 \times 10^{13}$. Executing over 32 trillion complex state-transition evaluations and massive memory-indexing operations is prohibitively expensive. As the RBS scales towards industrial sizes, e.g., 50-cell systems, the state space explodes exponentially to astronomical figures (e.g., $15 \times 5^{50} \approx 1.3 \times 10^{36}$). This renders DP intractable for large-scale applications.

3.10.2. Comparison with GA

Unlike DP, heuristic algorithms such as GA bypass strict state–space traversal, making them computable for optimizing the discharge trajectory for the 10-cell RBS under the same initial cell SoCs set for DRL evaluation. As illustrated in Fig. 10, the GA converges at the 114th generation, consuming a massive computation time of 41.15 h. More importantly, the final converged total discharge time is only 2134.0 s, which is significantly inferior to the over 3000 s performance achieved by the proposed enhanced WA framework in Table 5.

Traditional heuristic algorithms like GA feature critical drawbacks such as lacking generalization and requiring long computational time. The 41.15 h GA computation yields a fixed action sequence strictly bound to one specific initial condition. Any change of initial conditions requires re-execution from scratch, making it too time-consuming for real-time control. In contrast, the offline-trained DRL agent functions as a generalized closed-loop policy, inferring optimal topological actions for arbitrary, unseen system states only in milliseconds with almost negligible computational cost. As rigorously verified in the previous experiments in Section 3.8, the trained DRL agent exhibits robust adaptability and consistent optimality under varying initial SoC distributions and dynamic load profiles, successfully addressing the issues of lacking generalization and long latency in traditional GA.

3.11. Scalable and safety-guaranteed MA+WA framework for large-scale RBSs

As the RBS scales up to industrial sizes, such as 50 cells, the centralized discrete action space explodes exponentially. This rapid expansion easily exceeds physical memory limits and renders the standard single-agent WA framework computationally intractable. To enhance the scalability, we propose a decentralized multi-agent (MA) WA framework combined with action embedding. By employing a divide-and-conquer strategy, the massive global action space is factorized into localized, computationally tractable sub-spaces, enabling scalable cooperative optimization for large-scale RBSs.

The proposed safety-guaranteed MA+WA framework, as illustrated in Fig. 11, is built upon the standard centralized-training, decentralized-execution MA architecture [25]. While this baseline formulation effectively resolves the dimensionality explosion by factorizing the 50-cell global observation into localized agent states, deploying it directly on an RBS is highly risky, as unconstrained exploratory actions easily trigger short-circuit faults. To bridge this critical safety gap, our framework introduces a dual-layer protection mechanism. The first layer focuses on local feasible space allocation during the WA action embedding phase. Rather than allowing agents to explore the full combinatorial switch state space, the continuous proto-actions generated by the shared actors are directly mapped into an exclusive, pre-allocated feasible space dictionary (adapted for localized sub-modules from the global formulation established in our previous work [29]) via an ANN search. This fundamental constraint ensures that the discrete sub-actions are strictly physically safe within their assigned sub-modules. However, independently safe local sub-actions might still form dangerous global loops when stitched together. Therefore, a robust fail-safe mechanism is deployed as the second safety layer. After the localized candidates are assembled and the centralized Critic network identifies the optimal joint action, this combined topology is instantaneously evaluated by a high-speed topology solver. This solver is developed as a zero-order derivation of the analytical model introduced in our other prior study [36]. If a fatal configuration is detected, the framework forcefully overrides it with the nearest safe default topology, guaranteeing absolute physical safety before the global switch topology a_i is executed. Finally, the safe transition data is stored in a prioritized replay buffer to drive the standard centralized training process, effectively guiding the decentralized actors towards global optimal SoC balancing without breaking physical safety boundaries.

To demonstrate the necessity and effectiveness of the proposed physical knowledge-injected architecture, comprehensive simulations are conducted on the 50-cell RBS environment utilizing the CCV embedding method with $K = 50$. The system operates under a power load of 210 W to accommodate the expanded module size. For a fair and consistent evaluation, the initial SoCs for the evaluation episodes are fixed, generated from a uniform distribution between 0.74 and 1.0 across the 50 cells. Furthermore, to effectively drive the decentralized MA framework, the reward function is reformulated. Each agent receives an individualized reward comprising the original global system reward augmented by a localized penalty term. This local term calculates the change in the local SoC SD, multiplied by a scaling factor of 500, which explicitly encourages each agent to prioritize the balancing of its assigned sub-module.

The proposed safety-guaranteed framework is compared with a standard MA baseline, which utilizes identical MA structures but relies entirely on reward penalties to avoid dangerous actions, lacking both the local feasible space embedding and the global fail-safe mechanism. The comparative results are illustrated in Fig. 12. As shown

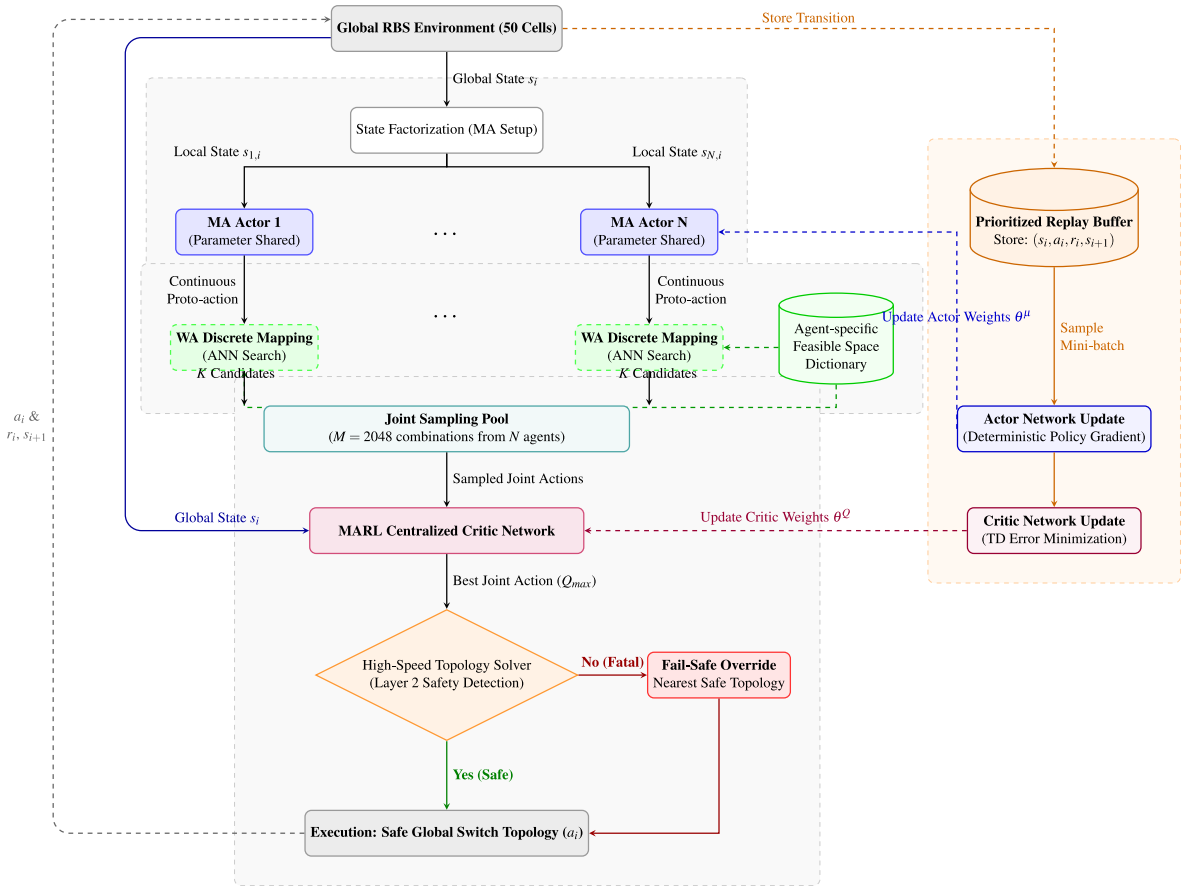


Fig. 11. Structure of the proposed Safety-Guaranteed MA+WA framework.

in Fig. 12(a), the standard baseline exhibits an immediate and total failure, consistently yielding a total discharge time of 0 s throughout the training process. This catastrophic failure visually indicates that without the dual-layer safety mapping, the baseline generates massive short-circuit faults at the very first timestep, triggering an emergency system shutdown to prevent physical destruction. In contrast, the proposed MA+WA method demonstrates highly stable learning capabilities, progressively extending the operational duration and eventually converging to a maximum total discharge time of 4672 s.

Furthermore, Fig. 12(b) tracks the real-time SoC balancing performance under the fully trained MA+WA framework. The framework successfully coordinates the 50-cell system, dynamically balancing the localized SoCs into a cohesive, narrow band until the end of the discharge cycle. The exceptional balancing capability is quantitatively reflected by the SoC SD trajectory, which continuously drops from a highly unbalanced initial value of 0.078987 to 0.020605 at the end of the discharge. This demonstrates that the proposed MA+WA framework does not only guarantee absolute physical safety but also achieves highly effective optimization for large-scale battery systems.

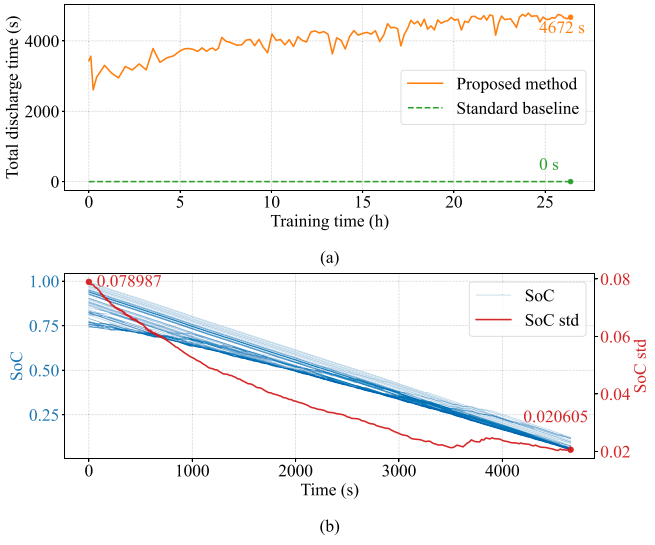


Fig. 12. Performance comparison between the proposed safety-guaranteed MA+WA framework and the standard baseline on a 50-cell RBS. (a) Convergence of the evaluated total discharge time during training. (b) Real-time SoC balancing trajectories and the evolution of SoC SD under the proposed MA+WA framework.

4. Conclusions and future work

Optimal control problems of RBSs involve high-dimensional and discrete-value search space, and the space expands exponentially as the system scales. To solve such challenging problems, an enhanced WA is developed in this study. Three embedding methods are first proposed to map the discrete-value actions into a continuous-value space to facilitate the WA training. Then, TD3 and HNSW search are integrated for enhanced optimality and efficiency. Case studies aimed at maximizing the discharge energy delivery demonstrate that the enhanced WA significantly improves the discharge energy compared to the original WA across various scenarios. To balance convergence performance and computational cost, an appropriate range for the number of nearest neighbors in HNSW is recommended based on these case studies. Furthermore, compared to other DRL methods such

as DQN and PPO, the enhanced WA achieves superior convergence performance, particularly in larger action spaces. Experimental tests on embedded circuit platform confirm the enhanced WA's applicability and effectiveness in practical RBS control scenarios.

The enhanced WA developed in this study offers a scalable and high-performance DRL framework for solving general RBS optimal control problems, with potential applications in electric vehicles and energy storage systems. Since the large action space size of practical battery systems is a primary factor limiting the training speed of DRL, future studies will be focused on further restricting the action space size for improved training efficiency. Moreover, the enhanced WA will also be investigated to improve other critical performance metrics of RBSs in our future work.

CRediT authorship contribution statement

Changyou Geng: Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Rui Li:** Writing – review & editing, Validation, Supervision, Resources, Conceptualization. **Dezhi Ren:** Validation, Resources, Data curation. **Enkai Mao:** Visualization, Validation. **Xinyi Zheng:** Investigation, Data curation. **Changfu Zou:** Writing – review & editing, Funding acquisition. **Weiji Han:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Ethics statement

This study does not involve human participants or animal subjects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Science and Technology Commission of Shanghai Municipality under Project 23160713400, the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie Actions grant agreement No. 101209044, the National Natural Science Foundation of China under Grant 52577208, and the Science and Technology Innovation program of Hunan Province under Grant 2024RC4004.

Appendix. Network update of the enhanced WA

To train the policy within the enhanced Wolpertinger Architecture (WA), the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [26] is employed as the underlying learning framework. During the training process, as shown in Fig. 1, a batch of experience tuples (s_k, a_k, r_k, s_{k+1}) are sampled from the replay buffer, and they are used to update the critic and actor networks as described below.

For the critic network, TD3 introduces the target policy smoothing to reduce overestimation bias in Q-value estimation. This is achieved by perturbing the target action generated by the target actor network $\pi_{\theta'_\pi}$ with clipped Gaussian noise. The perturbed action is defined as follows.

$$\tilde{a}_k = \pi_{\theta'_\pi}(s_{k+1}) + \mathrm{clip}(\varepsilon, -c, c), \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{A.1}$$

where ε is zero-mean Gaussian noise with SD σ, and the clipping operation ensures that the noise remains within the range $[-c, c]$. Using this smoothed action, the target Q-value is then calculated. To mitigate overestimation, TD3 maintains two critic networks $Q_{\theta_{Q_1}}$ and $Q_{\theta_{Q_2}}$, and computes the target value y_k as follows by taking the minimum of the two target Q-networks $Q_{\theta'_{Q_1}}$ and $Q_{\theta'_{Q_2}}$'s predictions.

$$y_k = r_k + \gamma \min_{j=1,2} Q_{\theta'_{Q_j}}(s_{k+1}, \tilde{a}_k). \tag{A.2}$$

For the jth critic network, $j \in \{1, 2\}$, the parameters in θ_{Q_j} are trained by minimizing the following loss function L_j defined by the mean squared error between its predicted Q-value and the target value y_k.

$$L_j(\theta_{Q_j}) = \frac{1}{N}\sum_{k=1}^{N}\left[Q_{\theta_{Q_j}}(s_k, a_k) - y_k\right]^2, \quad j \in \{1, 2\}, \tag{A.3}$$

where N is the batch size.

For the actor network, the parameters in θ_π are updated less frequently, with the goal of maximizing the expected Q-value under Critic-1. The actor objective is defined by

$$J(\theta_\pi) = -\frac{1}{N}\sum_{k=1}^{N} Q_{\theta_{Q_1}}(s_k, \pi_{\theta_\pi}(s_k)), \tag{A.4}$$

which encourages the policy to choose actions that lead to higher long-term returns.

To improve the training stability, TD3 employs soft update of the target networks' parameters θ'_{Q_j} and θ'_π via Polyak averaging. The update rules are given as follows, in which $\tau \ll 1$ (e.g., $\tau = 0.005$).

$$\theta'_{Q_j} \leftarrow \tau\theta_{Q_j} + (1-\tau)\theta'_{Q_j}, \quad \theta'_\pi \leftarrow \tau\theta_\pi + (1-\tau)\theta'_\pi. \tag{A.5}$$

In the proposed enhanced WA framework, the overall training procedure largely follows TD3, but with a key distinction in action handling. Specifically, while the actions stored in the replay buffer and used for critic updates are selected from the original discrete action space $\mathcal{A}$, the actor operates in a continuous proto-action space $\mathcal{C}$. When updating the actor, the output of actor $\hat{a} = \pi_{\theta_\pi}(s_k)$ is a continuous proto-action, which cannot be directly executed but can be used for gradient computation. The gradient of the actor objective is approximated using the following chain rule.

$$\nabla_{\theta_\pi} J(\theta_\pi) \approx \frac{1}{N}\sum_{k=1}^{N}\left(\nabla_a Q_{\theta_Q}(s, a)\Big|_{\hat{a}=\pi_{\theta_\pi}(s_k)} \nabla_{\theta_\pi}\pi_{\theta_\pi}(s)\Big|_{s=s_k}\right). \tag{A.6}$$

Although the actual action executed is discrete and selected via a nearest-neighbor search in $\mathcal{A}$, the actor network is trained using gradients taken with respect to its continuous proto-action output $\hat{a} = \pi_{\theta_\pi}(s)$. This design allows the policy to be updated via gradient-based optimization, while the critic still learns from the actual discrete actions executed in the environment. The theoretical foundation of this hybrid learning mechanism is detailed in the original WA study [23].

Data availability

Data will be made available on request.

References

[1] Q. Ouyang, J. Chen, J. Zheng, H. Fang, Optimal cell-to-cell balancing topology design for serially connected lithium-ion battery packs, IEEE Trans. Sustain. Energy 9 (1) (2017) 350–360.

[2] F. Feng, X. Hu, J. Liu, X. Lin, B. Liu, A review of equalization strategies for series battery packs: variables, objectives, and algorithms, Renew. Sustain. Energy Rev. 116 (2019) 109464.

[3] L. Komsiyska, T. Buchberger, S. Diehl, M. Ehrensberger, C. Hanzl, C. Hartmann, J. Kleiner M. Hölzle, M. Lewerenz, B. Liebhart, et al., Critical review of intelligent battery systems: Challenges, implementation, and potential for electric vehicles, Energies 14 (18) (2021) 5989.

[4] M. Alahmad, H. Hess, M. Mojarradi, W. West, J. Whitacre, Battery switch array system with application for JPL's rechargeable micro-scale batteries, J. Power Sources 177 (2) (2008) 566–578.

[5] Z. Zhang, Y. Cai, Y. Zhang, D. Gu, Y. Liu, A distributed architecture based on microbank modules with self-reconfiguration control to improve the energy efficiency in the battery energy storage system, IEEE Trans. Power Electron. 31 (1) (2015) 304–317.

[6] W. Han, C. Zou, L. Zhang, Q. Ouyang, T. Wik, Near-fastest battery balancing by cell/module reconfiguration, IEEE Trans. Smart Grid 10 (6) (2019) 6954–6964.

[7] F. Altaf, B. Egardt, L. Johannesson Mårdh, Load management of modular battery using model predictive control: Thermal and state-of-charge balancing, IEEE Trans. Control Syst. Technol. 25 (1) (2017) 47–62, http://dx.doi.org/10.1109/TCST.2016.2547980.

[8] T. Kim, W. Qiao, L. Qu, Power electronics-enabled self-X multicell batteries: A design toward smart batteries, IEEE Trans. Power Electron. 27 (11) (2012) 4723–4733.

[9] J. Kacetl, J. Fang, T. Kacetl, N. Tashakor, S. Goetz, Design and analysis of modular multilevel reconfigurable battery converters for variable bus voltage powertrains, IEEE Trans. Power Electron. 38 (1) (2022) 130–142.

[10] N. Bouchhima, M. Gossen, S. Schulte, K.P. Birke, Lifetime of self-reconfigurable batteries compared with conventional batteries, J. Energy Storage 15 (2018) 400–407.

[11] A. Badam, R. Chandra, J. Dutra, A. Ferrese, S. Hodges, P. Hu, J. Meinershagen, T. Moscibroda, B. Priyantha, E. Skiani, Software defined batteries, Proc. 25th Symp. Operating Syst. Princ., ACM, Monterey California, 2015, pp. 215–229, http://dx.doi.org/10.1145/2815400.2815429.

[12] Y. Kim, S. Park, Y. Wang, Q. Xie, N. Chang, M. Poncino, M. Pedram, Balanced reconfiguration of storage banks in a hybrid electrical energy storage system, in: IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD), 2011, pp. 624–631.

[13] L. He, L. Gu, L. Kong, Y. Gu, C. Liu, T. He, Exploring adaptive reconfiguration to optimize energy efficiency in large-scale battery systems, in: IEEE 34th Real-Time Syst. Symp., 2013, pp. 118–127, http://dx.doi.org/10.1109/RTSS.2013.20.

[14] L. He, Z. Yang, Y. Gu, C. Liu, T. He, K.G. Shin, Soh-aware reconfiguration in battery packs, IEEE Trans. Smart Grid 9 (4) (2018) 3727–3735, http://dx.doi.org/10.1109/TSG.2016.2639445.

[15] L. He, L. Kong, S. Lin, S. Ying, Y. Gu, T. He, C. Liu, Reconfiguration-assisted charging in large-scale lithium-ion battery systems, in: ACM/IEEE Int. Conf. Cyber-Physical Syst. (ICCPS), IEEE, 2014, pp. 60–71.

[16] N. Bouchhima, M. Schnierle, S. Schulte, K.P. Birke, Active model-based balancing strategy for self-reconfigurable batteries, J. Power Sources 322 (2016) 129–137.

[17] S. Ci, J. Zhang, H. Sharif, M. Alahmad, Dynamic reconfigurable multi-cell battery: A novel approach to improve battery performance, in: 27th Annu. IEEE Appl. Power Electron. Conf. Expo. (APEC), 2012, pp. 439–442.

[18] L. He, E. Kim, K.G. Shin, A case study on improving capacity delivery of battery packs via reconfiguration, ACM Trans. Cyber-Phys. Syst. 1 (2) (2017) 1–23, http://dx.doi.org/10.1145/3035539.

[19] C.P. Andriotis, K.G. Papakonstantinou, Managing engineering systems with large state and action spaces through deep reinforcement learning, Reliab. Eng. Syst. Saf. 191 (2019) 106483.

[20] S. Jeon, J. Kim, J. Ahn, H. Cha, Optimizing discharge efficiency of reconfigurable battery with deep reinforcement learning, IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst. 39 (11) (2020) 3893–3905, http://dx.doi.org/10.1109/TCAD.2020.3012230.

[21] F. Yang, F. Gao, B. Liu, S. Ci, An adaptive control framework for dynamically reconfigurable battery systems based on deep reinforcement learning, IEEE Trans. Ind. Electron. 69 (12) (2022) 12980–12987, http://dx.doi.org/10.1109/TIE.2022.3142406.

[22] S. Baccari, M. Tipaldi, V. Mariani, Deep reinforcement learning for cell balancing in electric vehicles with dynamic reconfigurable batteries, 2024,

[23] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, B. Coppin, Deep reinforcement learning in large discrete action spaces, 2016, arXiv:1512.07679 https://arxiv.org/abs/1512.07679.

[24] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, Weighted double deep multiagent reinforcement learning in stochastic cooperative environments, in: PRICAI 2018: Trends Artif. Intell.: 15th Pacific Rim Int. Conf. Artif. Intell., Nanjing, China, August 28–31, 2018, Proc., Part II 15, Springer, 2018, pp. 421–429.

[25] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, 2020, arXiv:1706.02275 https://arxiv.org/abs/1706.02275.

[26] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: Proc. 35th Int. Conf. Mach. Learn., PMLR, 2018, pp. 1587–1596.

[27] E. Bernhardsson M. Aumüller, A. Faithfull, Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms, 2018, arXiv:1807.05614 https://arxiv.org/abs/1807.05614.

[28] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, P.S. Thomas, Learning action representations for reinforcement learning, in: International Conference on Machine Learning, ICML, PMLR, 2019, pp. 941–950.

[29] C. Geng, D. Ren, E. Mao, C. Zou, X. Zheng M. Vašak, W. Han, Fundamental techniques for optimal control of reconfigurable battery systems: System modeling and feasible search space construction, 2025, arXiv:2501.03028 https://arxiv.org/abs/2501.03028.

[30] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. Wu, The surprising effectiveness of ppo in cooperative multi-agent games, Adv. Neural Inf. Process. Syst. 35 (2022) 24611–24624.

[31] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, S. Whiteson, Monotonic value function factorisation for deep multi-agent reinforcement learning, J. Mach. Learn. Res. (JMLR) 21 (178) (2020) 1–51.

[32] A. Mashayekh, S. Pohlmann, J. Estaller, M. Kuder, A. Lesnicar, R. Eckerle, T. Weyh, Multi-agent reinforcement learning-based decentralized controller for battery modular multilevel inverter systems, Designs 7 (4) (2023) 91.

[33] M. Bhuyan, K.K. Sarma, D.D. Misra, K. Guha, J. Iannacci, Adaptive hybrid beamforming codebook design using multi-agent reinforcement learning for multiuser multiple-input–multiple-output systems, Appl. Sci. 14 (16) (2024) 7109.

[34] Y. Zhang, M. Alrabeiah, A. Alkhateeb, Reinforcement learning of beam codebooks in millimeter wave and terahertz mimo systems, IEEE Trans. Commun. 70 (2) (2021) 904–919.

[35] R.S. Sutton, A.G. Barto, et al., Reinforcement learning: An introduction, vol. 1, MIT Press, Cambridge, 1998.

[36] C. Geng, D. Ren, E. Mao, X. Zheng, C. Zou M. Vašak, W. Han, A unified modeling framework of reconfigurable battery systems for optimal control, IEEE Trans. Transp. Electrification (2026).

[37] P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, in: Proc. 30th Annu. ACM Symp. Theory Comput. - STOC'98, ACM Press, Dallas, Texas, United States, 1998, pp. 604–613, http://dx.doi.org/10.1145/276698.276876.

[38] M. Muja, D. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration., in: VISAPP - Proc. 4th Int. Conf. Computer Vis. Theory Appl., Vol. 1, 2009, pp. 331–340.

[39] Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018, arXiv:1603.09320 https://arxiv.org/abs/1603.09320.

[40] J. Achiam, Spinning up in deep reinforcement learning, 2018, https://github.com/openai/spinningup.

[41] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, 2017, arXiv:1710.02298 https://arxiv.org/abs/1710.02298.

[42] S.F. Schuster, M.J. Brand, P. Berg, M. Gleissenberger, A. Jossen, Lithium-ion cell-to-cell variation during battery electric vehicle operation, J. Power Sources 297 (2015) 242–251, http://dx.doi.org/10.1016/j.jpowsour.2015.08.001.

[43] A. Devie, G. Baure, M. Dubarry, Intrinsic variability in the degradation of a batch of commercial 18650 lithium-ion cells, Energies 11 (5) (2018) 1031, http://dx.doi.org/10.3390/en11051031.

[44] D. Oeser, T. Hein, A. Ziegler, M. Seefried, S. Gielinger, D. Montesinos-Miracle, G. Bohn, A. Ackva, Age-related development of cell-to-cell variation under various operating conditions in commercial NMC/graphite lithium-ion cells, J. Energy Storage 101 (2024) 113787, http://dx.doi.org/10.1016/j.est.2024.113787.