Near-optimal energy management for plug-in hybrid fuel cell and battery propulsion using deep reinforcement learning

Peng Wu*, Julius Partridge, Enrico Anderlini, Yuanchang Liu, Richard Bucknall

Marine Research Group, Department of Mechanical Engineering, University College London, London WC1E 7JE, UK

Abstract

Plug-in hybrid fuel cell and battery propulsion systems appear promising for decarbonising transportation applications such as road vehicles and coastal ships. However, it is challenging to develop optimal or near-optimal energy management for these systems without exact knowledge of future load profiles. Although efforts have been made to develop strategies in a stochastic environment with discrete state space using Q-learning and Double Q-learning, such tabular reinforcement learning agents' effectiveness is limited due to the state space resolution. This article aims to develop an improved energy management system using deep reinforcement learning to achieve enhanced cost-saving by extending discrete state parameters to be continuous. The improved energy management system is based upon the Double Deep Q-Network. Real-world collected stochastic load profiles are applied to train the Double Deep Q-Network for a coastal ferry. The results suggest that the Double Deep Q-Network acquired energy management strategy has achieved a further 5.5% cost reduction with a 93.8% decrease in training time, compared to that produced by the Double Q-learning agent in discrete state space without function approximations. In addition, this article also proposes an adaptive deep reinforcement learning energy management scheme for practical hybrid-electric propulsion systems operating in changing environments.

Keywords:

Hybrid fuel cell and battery propulsion, Coastal ferry, Continuous monitoring, Deep reinforcement learning, Energy Management System

1. Introduction

1.1. Background and motivation

Plug-in hybrid fuel cell and battery propulsion systems are promising to revolutionise emission from transportation applications such as coastal ships [1] and road vehicles [2], provided that the fuel cells operate on sustainable fuels (e.g. H_2) [3] and the decarbonisation of grid electricity is successful [4]. However, it remains challenging to adopt such systems due to high costs from equipment degradation and energy consumption [5, 6]. Minimising the costs of such systems is essential [7].

This article focuses on reducing the costs of such hybrid systems from the operational perspective, i.e. by optimal energy management of the power sources. Efforts have been made by the authors to optimise the cost and emission performance of the plug-in hybrid systems concurrently in the system design phase [6]. However, developing a cost-effective Energy Management System (EMS) that can provide long-term optimal control references for the hybrid system remains challenging due to the uncertainty of read-world power demands [8]. The optimal energy management can be modelled as a sequential decision-making problem, i.e. what actions to be taken with observed system states to achieve objectives such as minimum mission costs and Greenhouse Gas (GHG) emissions, such that the system can 'plan' to make overall optimal decisions to guide the control of the power sources [4].

1.2. Previous work

The authors have attempted to solve the energy management problem in their previous work [9], using Double Q-learning Reinforcement Learning (RL) with large scale real-world stochastic continuous monitoring data from [10]. However, the cost-effectiveness of the energy management strategies generated by tabular reinforcement learning approaches without function approximations is limited due to the 'curse of dimensionality', as the computational requirements grow exponentially with the number of state parameters and their resolutions

*Corresponding author
Email address: peng.wu.14@ucl.ac.uk (Peng Wu)

[11]. In addition, the results in [9] suggest that overestimates of action values can be problematic in real-world stochastic environments.

1.3. Aim

This article aims to further enhance the cost-saving performance of the EMS using Deep Reinforcement Learning (DRL) that could potentially mitigate function overestimates in real-world stochastic environments. Previously, the energy management problem was solved in discrete space, which limits the resolution of EMS and its performance [9]. The introduction of deep neural networks extends the state space to be continuous, enabling better perceiving of system states. In addition, this article also provides a practical adaptive EMS updating scheme based on DRL, to achieve long-term near-optimal operations for hybrid-electric propulsion systems.

1.4. Literature review

Conventional approaches such as rule-based, equivalent minimisation strategies and Model Predictive Control (MPC) have been widely adopted to advance the research of EMS for hybrid-electric propulsion systems [12]. Banvait et al. [13] developed a rule-based EMS for a plug-in hybrid electric road vehicle, which improved the gas mileage by 16% for standard driving cycles. Peng et al. [14] proposed a rule-based EMS for a plug-in hybrid electric bus optimised by dynamic programming, which reduced both fuel and electricity consumptions for a fixed load profile. Wang et al. [15] developed a rule-based EMS with power prediction for a hybrid-electric propulsion system, which was evaluated by two driving cycles. Kalikatzarakis et al. [16] applied an equivalent consumption minimisation strategy to a plug-in hybrid ship propulsion system. Their simulation results showed a 6% fuel saving could be realised under several load profiles. Another example of equivalent factor optimisation strategy can be found in [17]. Ebrahim et al. [18] implemented a self-adaptive Harris Hawks Optimisation-based scheme for a hybrid fuel cell and battery power system with improved efficiency and system performance. Bassam et al. [19] developed an EMS with multi schemes for a hybrid passenger ship powered by fuel cell and battery, which

comprised several sub-strategies for different load conditions. Wang et al. [20] conducted a comparative study between the Proportional–Integral–Derivative and rule-based EMS, and applied dynamic programming to evaluate the performance of the EMS under investigation. Another comparative study on EMS can be found in [21]. Hou et al. [22] proposed an MPC algorithm to control the hybrid electric propulsion systems using additional penalties on the rapid change of energy storage system State of Charge (SOC) to overcome the limitations caused by short predictive horizon of MPC. More comprehensive reviews on EMS can be found from [23] for road vehicles, and [24] for ships.

It should be noted that deterministic dynamic programming requires complete knowledge of the load profile before generating an energy management strategy. Therefore, the strategy generated by such an approach is typically applied to calculate an optimum off-line strategy to evaluate other online strategies' effectiveness. Moreover, real-world load profiles could deviate significantly from the standard cycles used to calibrate the EMS, making the EMS perform well only in the specific calibration cycles. It remains a challenge to develop EMS which can perform optimally in unseen load cycles over long terms [8]. Additionally, as both fuel cells and batteries can degrade rapidly under certain operating conditions, it is necessary to consider power degradation characteristics in the EMS [9, 25, 26, 27].

Recently, novel intelligent approaches such as RL and DRL have been applied to advance EMS for hybrid-electric propulsion systems. Liu et al. [28] adopted Q-learning to solve the optimal energy management problem for a tracked hybrid-electric vehicle with a specific load profile, stochastic dynamic programming generated strategy was applied to evaluate the performance of the developed EMS. Their Q-learning approach has reduced computation time. Xiong et al. [29] adopted RL to develop the hybrid system control strategy, and applied Kullback–Leibler divergence rate to determine whether the strategy needs an update. However, the additional parameter can be avoided as one can easily calculate optimal energy management strategies for past load profiles and their corresponding objective values which can be used as direct EMS updating indicators. Xu et al. [30] proposed an ensemble RL-based energy management strategy, using Q-learning with thermostatic and equivalent consumption minimization strategies to improve the fuel economy. The authors Wu et al.

[9] applied Double Q-learning to generate generic energy management strategies for unseen load profiles using large scale stochastic load profiles and suggested that the stochasticity of the load profiles can lead to diverged agent training. Although the RL-based EMS can significantly improve system efficiency, the resolution of the discrete state space prohibits further improvements due to the 'curse of dimensionality' [11]. The work of Wu et al. [31] suggests that DRL with deep neural networks as function approximators can further improve the RL-based EMS. Further efforts of applying DRL to develop intelligent EMS for hybrid-electric propulsion systems have been observed in studies such as [32], [33] and [34]. However, none of those studies mentioned above has applied DRL agents with large-scale stochastic load profiles.

1.5. Research gap and contributions

From the analysis above, it is evident that various approaches have been applied to develop EMS for hybrid-electric propulsion systems, including conventional (e.g. rule-based) and novel intelligent ones. Although RL and DRL have been applied to generate energy management strategies for the hybrid systems, there is a lack of EMS in continuous state space and capable of dealing with unseen load profiles, especially the highly stochastic ones in marine applications. Existing approaches are mostly based on certain standard load profiles or very limited number of load profiles. The EMS performance can be achieved in long-term real-world operations is yet to be better investigated.

This article narrows the research gaps in the literature by:

- Training the Double Deep Q-Network (Double DQN) [35], i.e. a variant of Deep Q-Network (DQN) [36] developed to mitigate action value overestimations, with large-scale real-world stochastic load profiles with continuous state parameters, to generate near-optimal energy management strategies for unseen future load profiles.

- The concept of Double DQN is applied with Huber loss function to further reduce the impact of overestimation due to real-world stochasticity.

- This article also discusses an adaptive EMS updating scheme that can guide the practical applications of the DRL-based EMS to achieve long-term near-optimal energy management of hybrid-electric systems.

1.6. Organisation

The remainder of this article is arranged as follows. Section 2 formulates the optimal energy management problem. Section 3 details the DQN and Double DQN agents. Section 4 details the environment formulation. Section 5 presents the training of the agent. Section 6 details the results by applying the Double DQN EMS to sample validation voyages. Section 7 discusses the adaptive EMS updating scheme for long-term operations of hybrid-electric propulsion systems. Section 8 details the conclusions.

2. Problem formulation

Figure 1 illustrates the schematic the agent-environment interaction framework proposed to generate the EMS using Double DQN and the procedure of applying the generated EMS. The environment comprises the plug-in hybrid PEMFC and battery propulsion system model and historical load profiles. The Double DQN interacts with the environment by controlling the fuel cell power and observing the reward signal and resulting system state returned by the environment. The EMS is designed to minimise the voyage costs. In other words, the energy management strategy or policy of the EMS maximises the cumulative rewards that represent the cost-effectiveness of the system. With a finite horizon T, for an episodic task, the action-value function (Q function), is the expected return of taking action a following a policy $\pi(s|a)$ in state s [9]:

$$Q(s,a) = \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi\right] \tag{1}$$

Solving the optimal energy management problem is to find an optimal strategy π^*:

$$\pi^*(s) = \arg\max_a \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k r_{t+k} | s_t = s, a_t = a\right] \tag{2}$$

which corresponds to the optimal Q function [11]:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k r_{t+k} | s_t = s, a_t = a\right] \tag{3}$$

where the subscripts denote time steps, $\gamma \in [0,1]$ is the discount rate, reward r_t is a measurement of the cost-effectiveness of taking action a_t in state s_t that results in the next system state s_{t+1}.

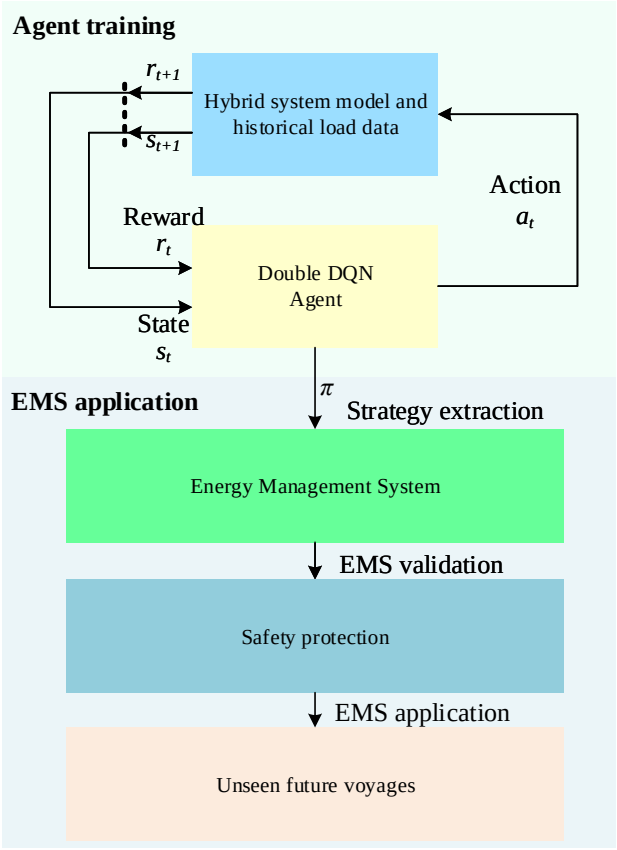

Figure 1: Schematic of the agent-environment interaction and EMS application. The environment consists of historical load profiles and the hybrid system model. The Double DQN agent is trained by the historical data to generate the energy management strategy parametrised by the Q-network, which is subsequently validated by a dataset unseen by the agent.

The objective of the DRL agent training is to find a policy π^*, i.e. an energy management strategy with near-optimal cost-effectiveness which can be applied as the core of the EMS to provide control references to the hybrid propulsion system, such that the hybrid system can

have the capability of 'planning' to maximise the overall cost-effectiveness of the system. Once the DRL agent has been trained and validated, the EMS can instantaneously output an action by observing the current system state. Additional safety functions, such as battery over-discharge protection, can also be added to the application procedure. However, such functions are beyond the DRL training scope.

3. Deep reinforcement learning agent

3.1. Deep Q-Network

For RL problems with large or continuous state spaces, function approximators are typically needed to generalise from previously encountered states which are similar in some sense to current ones [11]. A function approximator can be linear or non-linear [37, 38]. However, the training process of RL agents can be unstable or even diverge when a non-linear function approximator such as a neural network is used [39]. Lin [40] developed the concept of 'experience replay' to store the agent experience into a memory pool to train a RL agent with a neural network. Later work of Mnih et al. [41] proposed deep Q-learning using a deep neural network with convolution layers to approximate high dimensional raw pixel state inputs. Mnih et al. [36] further improved the deep Q-learning agents by adding target networks to improve training stability. Mnih et al. [36]'s Deep Q-Network (DQN) achieved performance levels comparable to professional human game testers in 49 Atari 2600 games.

The DQN is a model-free, off-policy reinforcement learning algorithm [36]. The agent maintains an experience memory pool with capacity M, storing the most recent M transition sequences. A transition sequence, collected via agent-environment interaction, at time step t, is denoted by:

$$\phi = (s_t, a_t, s_{t+1}, r_{t+1}) \tag{4}$$

i.e. in state s_t, action a_t is performed by the agent (following ε-greedy policy) and observes next environment state s_{t+1} and a reward signal r_{t+1} is returned from the environment. In each agent training step, a mini-batch with capacity D is randomly sampled from the experience memory pool such that previous experiences can be used effectively. In addi-

tion, the random sampling breaks the correlations of consecutive samples which can lead to unstable neural network training. The DQN agent includes two deep neural networks with identical structure, i.e. the Q-network $Q(s,a;\theta)$ parametrised by θ, and the Q-target network $\hat{Q}(s,a;\theta^{-})$ parametrised by θ^{-}. These neural networks approximate the action-value function with state (s) inputs for all actions (a) in the action space A.

As an improvement to the work of Mnih et al. [41], the additional Q-target network enhances the agent training stability by providing fixed target action value y_j for non-terminal states:

$$y_j = r_{j+1} + \gamma \max_{a'} \hat{Q}\left(s_{j+1}, a'; \theta^{-}\right) \tag{5}$$

where j denotes $j-th$ sample in the mini-batch. In the original DQN algorithm of [36], the Q-target network is updated periodically, while in this work it is soft-updated at each training step to further improve training stability:

$$\theta^{-} \leftarrow \tau\theta + (1-\tau)\,\theta^{-} \tag{6}$$

where $\tau \ll 1$ [42].

Considering the overestimations of the action-value function can affect the agent training [9], the Huber loss [43] is employed in this study to improve agent training stability, and to generate the energy management strategies. In addition, to investigate the influences of the loss function over training stability and EMS quality in a stochastic environment, experiments using Mean Squared Error (MSE) loss function have also been conducted to investigate the influences of the loss function over training stability and EMS quality in a stochastic environment.

The MSE loss is defined as the mean squared error of the temporal difference (denoted by δ) between the action values given by the Q-network and the targets y_j ($j \in [1, D]$) over a mini-batch:

$$L(\theta) = \frac{1}{D}\sum_{j=1}^{D} \delta_j^2 \tag{7}$$

where the temporal difference δ_j of $j-th$ sample in the mini-batch is:

$$\delta_j = y_j - Q\left(s_j, a_j; \theta\right) \tag{8}$$

In the work of Mnih et al. [36], δ_j was clipped to between -1 and +1 to improve the DQN algorithm stability. Such a technique corresponds to using an absolute value loss function for temporal differences outside $(-1, 1)$. Note that the clipping reduces the chances of overestimations for the action-value function when values given by the networks are noisy over large ranges. In [9], the Q-learning agent failed due to overestimations caused by the maximisation operation which approximated the expected action value. The concept of error clipping may provide a new approach to dealing with overestimations in the stochastic environment.

Instead of clipping the error term, the Huber loss, which performs similar function has been employed in this study. The Huber loss is calculated by [43]:

$$L(\theta) = \frac{1}{D}\sum_{j=1}^{D}\sigma_j \tag{9}$$

where:

$$\sigma_j = \begin{cases} \frac{1}{2}\delta_j^2, & \text{if } |\delta_j| < 1 \\ |\delta_j| - \frac{1}{2}, & \text{otherwise} \end{cases} \tag{10}$$

The Huber loss is the mean squared error when the temporal difference δ_j is small ($|\delta_j| < 1$) but acts like the mean absolute error ($|\delta| - \frac{1}{2}$) when the difference is large, which makes it more robust when overestimations of action-value function may degrade the agent training.

The Q-network and Q-target network share an identical network structure. The Q-network is trained by minimising the loss function $L(\theta)$ with respect to its parameters θ. The optimiser adopted in this study is the Adam optimiser [44]. The neural networks output action-value function values for each possible action with given state inputs. Note that the continuous signals (battery SOC, power demand and fuel cell per unit power) are not discretised as in [9]. Instead, these states are used as direct inputs to the Q-network such that the environment states can be accurately represented by continuous actual values. The state inputs are forward propagated sequentially from the input layer via hidden layers to output Q-values for all actions. Note that each neuron of the output layer corresponds to an action in the action space.

The DQN agent training starts with randomly initialised network parameters θ and θ^{-}. The exploration probability ε of the ε-greedy policy decreases linearly from a large initial value with the increase of training episode number, and is fixed at a small final value in the later stage of the training (i.e. training episode $n > N_d$). Note that completely random explorations initially fill the experience memory pool before the neural network training starts. The Q-network is trained every Z steps to gain sufficient experience.

3.2. Double Deep Q-Network

The results of Wu et al. [9] suggest that maximisation biases introduced during the construction of the action-value function can lead to poor learning performance if such biases are not addressed properly. The Double Q-learning agent achieved satisfactory performance using two Q-functions, while the Q-learning agent diverged with identical hyperparameters. It is not clear whether the DQN agent (as a deep variant of Q-learning) can succeed in the highly stochastic environment based on recorded historical power profiles. Therefore, the authors have explored solving the energy management problem with Double DQN.

The Double DQN (Algorithm 1) is proposed by [35] based on the concept of Double Q-learning [45] and DQN [36]. In Double Q-learning, two Q-functions are used to reduce the overestimations by decomposing the maximisation in the target into action selection and action evaluation [35, 45]. Without introducing additional neural networks to DQN, the Double DQN utilises the Q-target network to evaluate the maximising action (i.e. $\arg\max_a \left(Q\left(s_{j+1}, a; \theta\right)\right)$) given by the Q-network (see Figure 2) such that, the target value is calculated by:

$$y_j = r_j + \gamma Q\left(s_{j+1}, \arg\max_a \left(Q\left(s_{j+1}, a; \theta\right)\right); \theta^{-}\right) \tag{11}$$

4. Environment

4.1. Candidate ship and the data

The candidate ship is a typical coastal ferry operating between two fixed ports [6, 10]. It is intended that the plug-in hybrid fuel cell and battery propulsion system will replace the

Algorithm 1 Double Deep Q Network RL agent, adapted from [35] and [36].

Initialise replay memory D to capacity M
Initialise action-value function Q with random weights θ
Initialise target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
while $n < N_{max}$ **do**
    Initialise initial state s_1
    for $t = 1 : T$ **do**
        if $rand < \varepsilon$ **then**
            Select action a_t randomly from A
        else
            $a_t \leftarrow \arg\max_a \left(Q\left(s_t, a; \theta\right)\right)$
        end if
        Take action a_t, observe r_t, s_{t+1} and Termination Flag
        Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in D
        Every Z steps sample random mini-batch of transitions $(s_j, a_j, r_{j+1}, s_{j+1})$ from D
        Set $y_{j+1} = \begin{cases} r_{j+1}, & \text{if episode terminates at step } j+1 \\ r_{j+1} + \gamma Q\left(s_{j+1}, \arg\max_a Q\left(s_{j+1}, a; \theta\right); \theta^{-}\right), & \text{otherwise} \end{cases}$
        Perform a gradient decent on $\left(y_{j+1} - Q\left(s_{j+1}, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
        Soft update the target network: $\theta^{-} \leftarrow \tau\theta + (1-\tau)\,\theta^{-}$
        Terminate if Termination Flag is **true**
    end for
    if $n \leq N_d$ **then**
        $\alpha \leftarrow \alpha - \Delta\alpha \times n$
        $\varepsilon \leftarrow \varepsilon - \Delta\varepsilon \times n$
    end if
end while

original diesel-based propulsion system, which has a power capacity of 4370 kW (five diesel generator sets, with each prime mover rated at 874 kW). The annual operating duty is 300

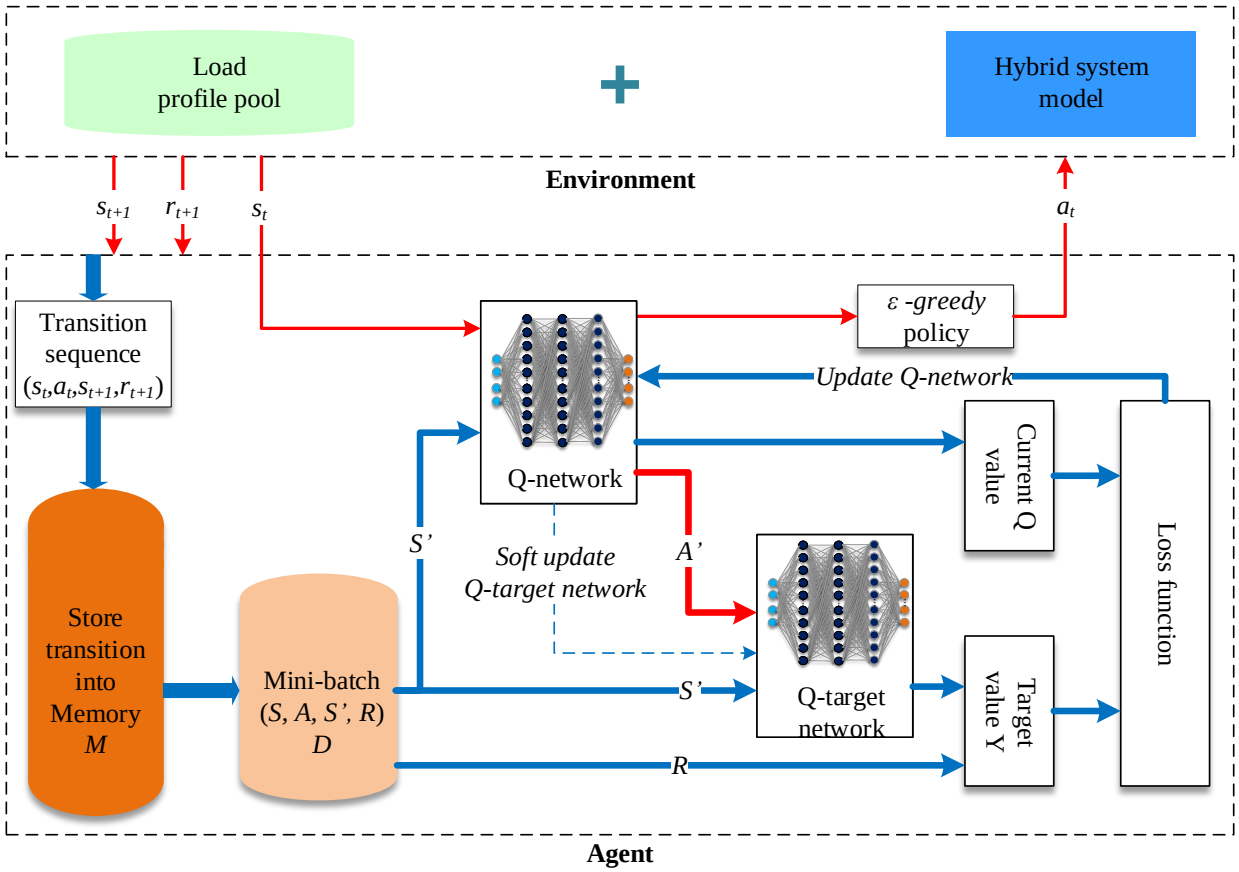

Figure 2: Double Deep Q-Network agent and environment schematic.

days, and the ship operates between two fixed ports accomplishing 16 voyages per day, with each voyage being of 60 min duration [6]. It is assumed that battery charging can be carried out in both ports of the defined route, and hydrogen will be replenished overnight and never during the operational period [9].

The load profiles used to train and validate the EMS were from [10]. There are 1081 voyages training in total, collected from 01/07/2018 to 31/08/2018. Another dataset with 392 voyages collected from 01/09/2018 to 30/09/2018 will be used for EMS validation [9]. Figure 3 shows 6 randomly selected sample power profiles in the training dataset. To reduce measurement noise, the raw data has been processed by a Gaussian-weighted moving average filter. The filter's moving average window is 4. The standard error is acquired from 20% of the whole window length. Although the power profiles follow a specific pattern in general,

each of them varies from the others.

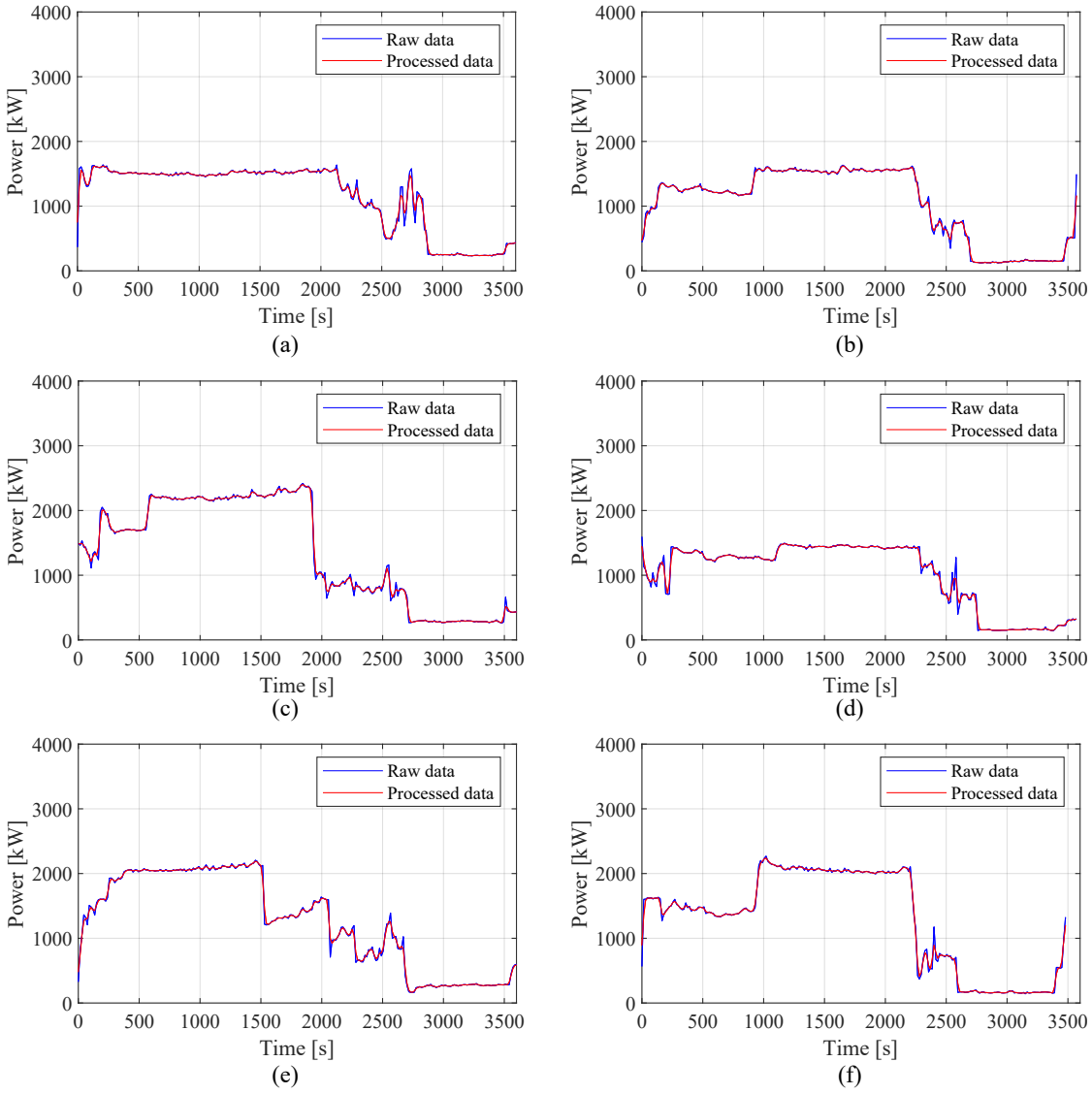

Figure 3: Sample load profiles from the training dataset. Measurement noise in the raw data has been reduced by a Gaussian-weighted moving average filter.

Figure 4 illustrates the power transition probability map with a grid length of 50 kW (a power index is assigned every 50 kW). The vertical and horizontal axes are about current and next power demand indices, respectively. The colour of the plot represents the transition probability from the current power demand index to the next power demand index. The diagonal line from lower left to the upper right of the figure corresponds to the situations

those with current and next power demand indies are identical. In general, the next power demand is more likely to have the same power demand index (see the highlighted diagonal line). However, the power transition pattern varies in different power regions. For example, in the low power regions (0–300 kW, 0–6 power demand indices), the probability of having the same power demand index is close to 1 (colour close to red). In the power regions from 350 to 1250 kW (7–25 power indices), the probability of having the same power index in the next time step is around 0.3. In the power regions from 1300 to 1750 kW (26–35 power indices), the probability of having the same power index in the next time step is approximately 0.5. More scattered transition probability pattern can be observed in the high power regions (3000–3500 kW, 60–75 power demand indices).

Note that these transition probabilities are not explicitly used in the EMS training since all RL agents adopted in this study are model-free. Instead, the agents are trained continuously by experiencing different power profiles from the training dataset in each training episode.

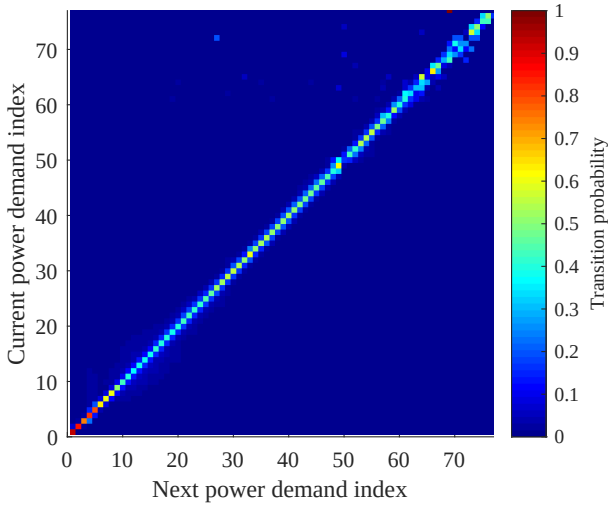

Figure 4: Case ship power transition probability map with grid length of 50 kW.

4.2. System model

Figure 5 provides an overview of the plug-in hybrid PEMFC and battery propulsion system model, which has been developed and optimised using the methodologies proposed

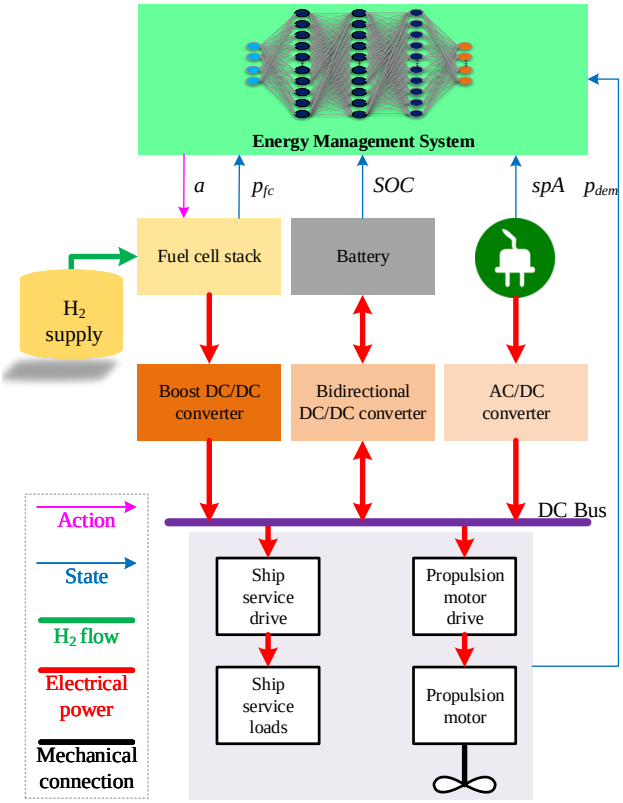

Figure 5: Overview of the plug-in hybrid PEMFC and battery propulsion system model.

in [46] and [6]. This model has also been used in the authors' previous work for EMS development [9]. The model consists of a PEMFC operating on H_2, a Lithium battery, a shore electricity supply, an EMS, power converters and a total system power demand including both the propulsion and service loads. The shore electricity supply can provide power when the ship is in the port. Note that the PEMFC power (2940 kW), battery capacity (581 kW h) and other model settings are identical as in [9]. Readers are referred to [9] for more details. The EMS controls the PEMFC power by seeing current system states, including shore power availability, power demand, battery SOC and current PEMFC power.

The individual battery cell outputs are connected in parallel and series to model the battery module. The open circuit voltage of an individual cell is a function of cell SOC as presented in Figure 6a. The fuel cell model outputs fuel cell specific H_2 consumption (see Figure 6b) which can be used to derive H_2 consumption. Note that the degradation effects

of the fuel cell and battery have been considered, which determine the degradation costs from the two power sources [6].

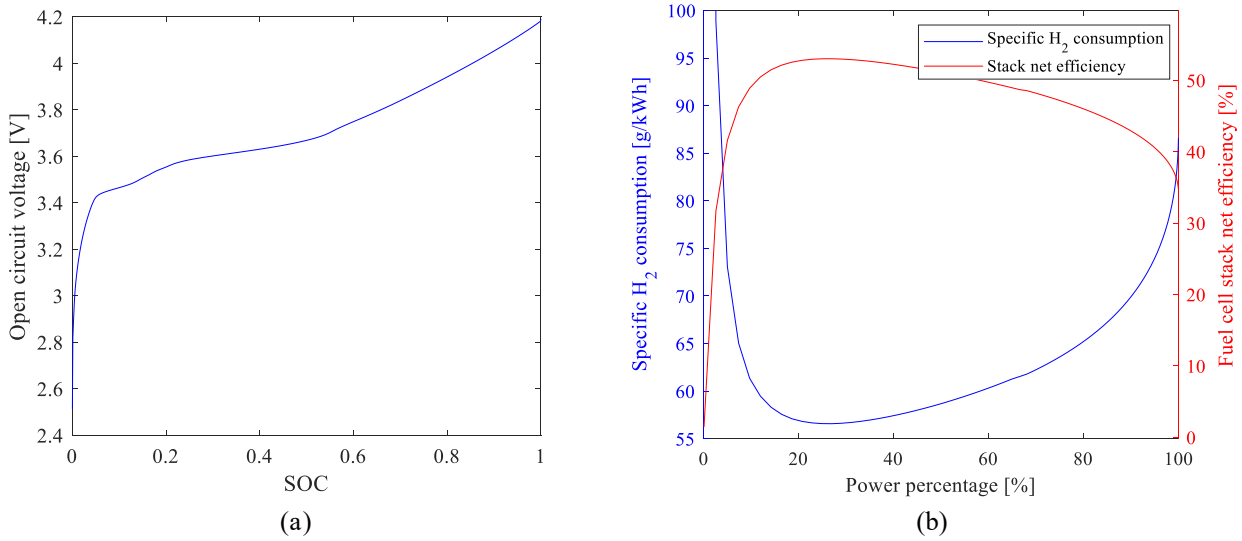

Figure 6: (a) Battery individual cell open circuit voltage-SOC curve, and (b) PEMFC system efficiency and its specific H_2 consumption.

The cost output from the model consists of two parts, i.e. the energy costs of H_2 fuel and shore-generated electricity to charge the battery, as well as the fuel cell and battery degradation costs. The battery is charged to its higher State of Charge (SOC) limit before starting a voyage by shore-generated electricity while at port. In sailing mode, the battery can discharge or be charged by the fuel cell when excessive power is available from the system. For each time step, the model outputs the cost incurred in this time step:

$$c_t = c_b + c_f + c_h + c_e \tag{12}$$

where c_b, c_f, c_h and c_e are the costs incurred by battery degradation, fuel cell degradation, H_2 and shore-generated electricity consumption, respectively. It is worth mentioning that the model parameters would require customisation for different types of power sources to capture unique features, and the model is developed with the flexibility of being calibrated by specific experimental data [4].

4.3. Reward function

In [9], the agent was trained throughout all time steps of the voyages, i.e. from the first moments of departure to last moments in ports. However, the ship was designed to operate purely on shore based electricity when in port mode (i.e. cold ironing). Although the Double Q agent has demonstrated its ability to maintain zero fuel cell power state in port mode, removing the training steps in port mode can simplify the training processes due to the cold ironing logic would only utilise shore provided power. Control of the fuel cell when in port mode appears unnecessary. Also, it has been observed that, the Double Q agent struggled to maintain final battery SOC constraint in some high power profiles. In practice, it would be feasible to increase the port time slightly to get the battery charged to SOC_H. Therefore, the reward function of the environment is reshaped as:

$$r_{t+1} = \begin{cases} -1, & \text{spA=0, if } s_{t+1} \text{ is infeasible} \\ -1, & \text{spA=0, if } p_{fc} + a_t \notin [0, 1] \\ \tanh\left(\dfrac{1}{c_{t+1}}\right), & \text{spA=0, else} \\ \displaystyle\sum_{k=t+1}^{K} \tanh\left(\dfrac{1}{c_k}\right), & \text{spA=1} \end{cases} \tag{13}$$

where when shore power is available ($spA = 1$), the environment returns a summed reward of all the costs incurred in port mode of the current episode. In sailing mode, i.e. $spA = 0$, the reward function is defined identically as in [9]. Note that K is the time step when the entire profile is completed; and $c_k = \infty$ if $k > T$ (i.e. extra time required to charge the battery), otherwise c_k is calculated as described in [9].

4.4. State space

Previously, in [9], the four-dimensional state space was discretised to state indices to store the action-value function into tables indexed by discrete state indices. Such a discretisation process is necessary for tabular RL approaches. However, the discretisation process and its resolution limit the quality of the generated policy [11]. In this work, discretisation of

continuous state parameters has been removed. The actual state space:

$$s(t) = [spA(t), p_{dem}(t), x(t), SOC(t)]^T \tag{14}$$

is directly applied to represent the environment states, where spA denotes the shore power availability ($spA = 0$ for sailing mode, $spA = 1$ for port mode), p_{dem} is normalised system power demand by dividing the actual power demand in kW by 1500 (i.e. $p_{dem} \leftarrow \frac{p_{dem}}{1500}$, such that the power demand input to the Q-network is around 1), $x(t)$ is fuel cell per unit power level at time step t ($x \in [0, 1]$), and $SOC \in [0, 1]$ denotes battery state of charge (SOC).

4.5. Action

The action space is defined identically as in [9], i.e. a tuple of PEMFC power level changes:

$$A = [a_1, a_2, ..., a_m, ..., a_{n-1}, a_n]^T \tag{15}$$

where $a_n > 0$ is the maximum increase and $a_1 < 0$ is the maximum decrease of PEMFC output in a time step. The power change notification of $a_m = 0$ indicates there is no change and the fuel cell output remains the same as in the previous time step.

In sailing mode, the environment overrides any action that would result the PEMFC power output becoming negative or higher than the rated power. With an action $a_t \in A$ chosen at current time step t, the PEMFC power level at the next time step is determined by:

$$x_{t+1} = \begin{cases} 0, & x_t + a_t < 0 \\ 1, & x_t + a_t > 1 \\ x_t + a_t, & \text{else} \end{cases} \tag{16}$$

In port mode, the agent is not required to control the fuel cell. The environment would decrease the PEMFC power to zero if it is not zero. The environment would extend episode length whenever necessary to charge the battery SOC to SOC_H (the power demand would be extrapolated from the last power demand that appears in original power profile). Such settings vary from the ones defined in [9], in which the agents were required to explore actions in port mode to maintain cold ironing.

5. Agent training

The agents were trained on a workstation with two Intel Xeon E5-2683 V3 processors. The environment and the agent were coded in Python. The agent's neural networks were built and trained with PyTorch V1.20. Each agent was trained with 10 different random seeds for reproducibility. During training, the agent policy performance was assessed by calculating the average values and standard deviations across the 10 instances running with different random seeds. Note that as the neural networks are relatively small, only one CPU thread is assigned to each running instance to avoid training speed degradation due to unnecessary parallelisation.

Also, the actual policy performance was periodically tested (every 100 training episodes) with 10 random training voyages during training. Note that in test mode, the $\varepsilon-$ greedy exploration probability was set at 0 with battery over-discharge protection enabled (disabled in training mode). Once the training of all the 10 instances was completed, the agent with the lowest episode cost was chosen to generate detailed EMS results in the following sections.

5.1. Neural network settings

Figure 7 illustrates the neural network configuration for the Q and Q-target networks. The environment state inputs are processed by the input layer with four neurons with Rectified Linear Unit (ReLU) activation function. Two fully-connected hidden layers are configured with 256 neurons each. Note that both hidden layers are applied with an ReLU activation function, while no activation function is applied to the output layer to allow negative action-value outputs. The neural network outputs five Q-values, corresponding to the 5 actions in the action space, respectively.

5.2. Hyperparameter settings

Table 1 details the hyperparameter settings applied. The policy is updated every 32 transition sequences (ϕ). In each training step, a mini-batch with 32 transition sequences is randomly sampled from the experience memory with a capacity of 1×10^6. Such a mini-batch is applied to train the Q-network using an Adam optimiser. The learning rate of the

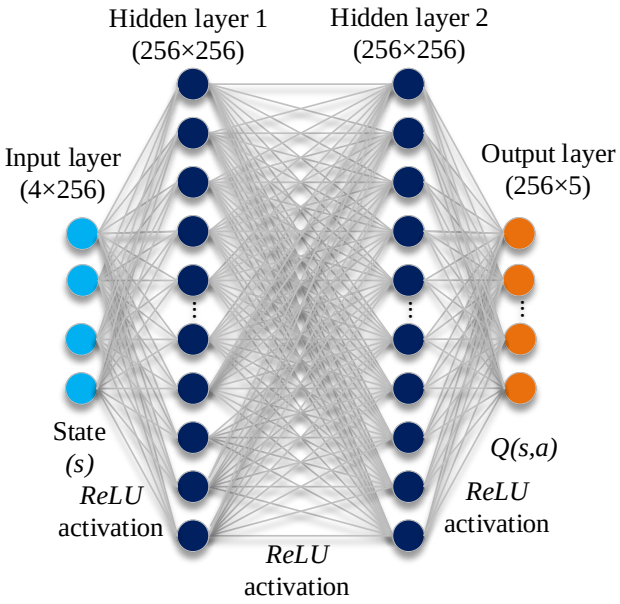

Figure 7: Q-network and Q-target network settings.

Adam optimiser is fixed at 0.0001 throughout the training. The exponential decay rates of first and second moment estimates (β_1 and β_2) are set at 0.9 and 0.999 respectively [44]. Note that the Q-target network is soft-updated with a soft-update weight of $\tau = 0.001$ in each training step. The exploration probability ε of $\varepsilon-$greedy policy starts with 1 and fixes at 0.05 after 5×10^3 episodes of training. Note that these parameters require careful tuning to achieve satisfactory performance.

Table 1: Hyperparameter settings.

Parameter	Description	Value
B	Mini-batch size	32
M	Experience memory size	1×10^6
τ	Target network update weight	0.001
γ	Discount factor	1
Z	Policy update frequency	32
α	Learning rate of Adam optimiser	0.001
β_1	Exponential decay rate for the first moment estimates of Adam optimiser	0.9
β_2	Exponential decay rate for the second moment estimates of Adam optimiser	0.999
ε_0	Initial exploration probability	1
ε_f	Final exploration probability	0.05

5.3. Training

Figure 8 details the training process of the Double DQN agent with the Huber loss function. The strategy is tested (exploration probability ε set to 0) 10 times with random load profiles every 100 training episodes. The training was terminated after 8,000 training episodes. In test mode, the moving average line of the Double DQN strategy converged to a value of around \$780 (Figure 8b). Note that the voyage cost is a sum of the costs incurred in all time steps of a voyage, including the energy (H_2 and electricity) and degradation (fuel cell and battery) costs (see Section 2). The success rate, i.e. the rate of completing voyages, converged to 100%.

In contrast, as shown in Figure 9, the training process of the Double DQN agent with the Mean Squared Error (MSE) diverged, suggesting that the concept of 'Double Deep Q-learning' alone is not sufficient to deal with the highly stochastic load profiles. Also, the

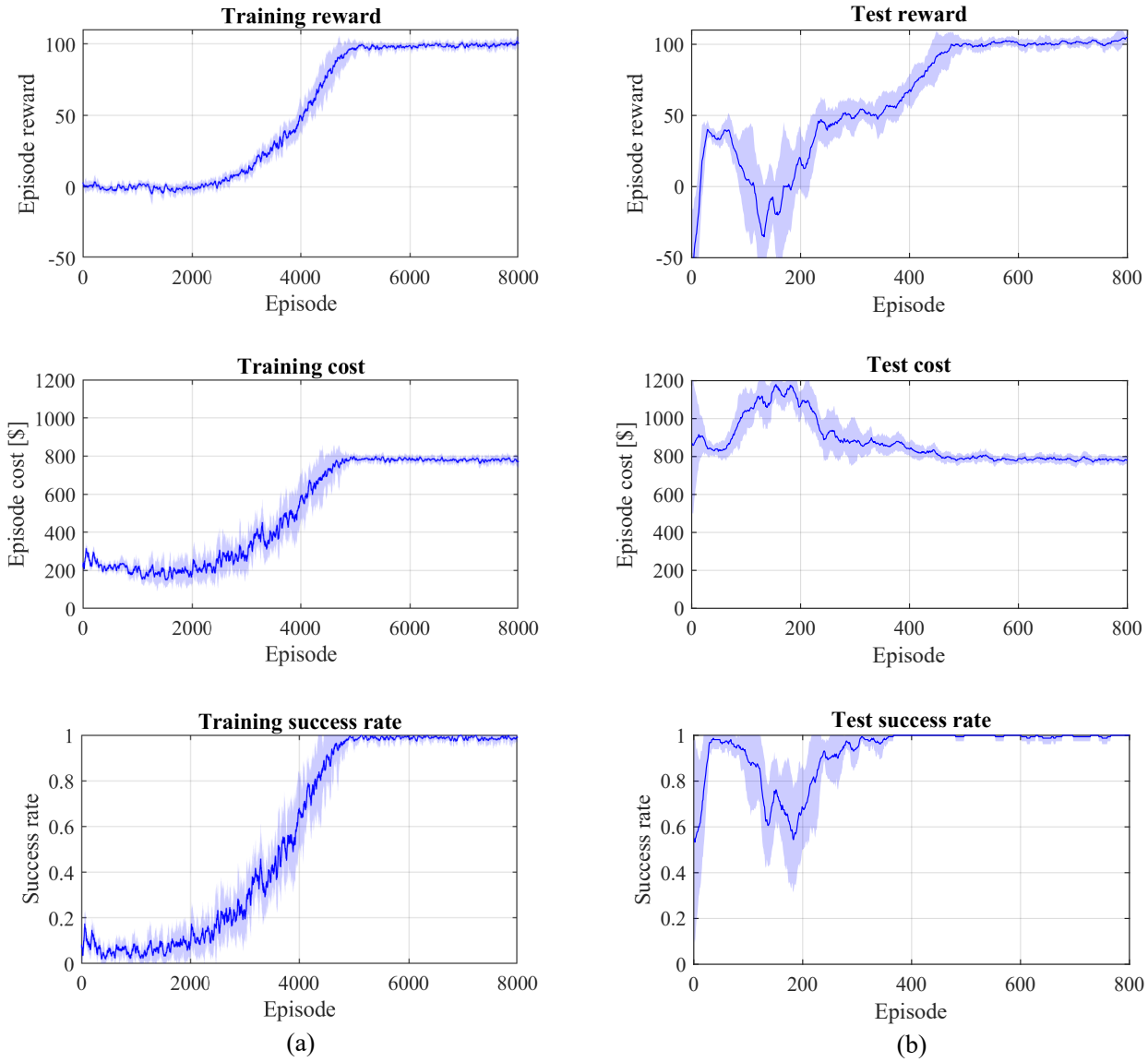

Figure 8: Double DQN agent training and testing with Huber loss function. The deep blue lines are moving average values across 10 instances running with different random seeds. The light blue shadows are the confidence bounds calculated by mean values ± standard deviations across the 10 instances. As shown in (a), after 4,400 episodes of training, all the instances have converged to a policy of completing the training voyages with the maximised reward. The converged training is also confirmed by periodic tests of the agents as in (b), i.e. the voyage cost starts to decrease from test episode of 180 with the success rate converged to 100%.

Huber loss function has improved Double DQN training stability. It is worth mentioning that extensive hyperparameter tuning experiments have been conducted for the combination

of Double DQN and MSE by iterating through predefined hyperparameter grids, without achieving stable agent training.

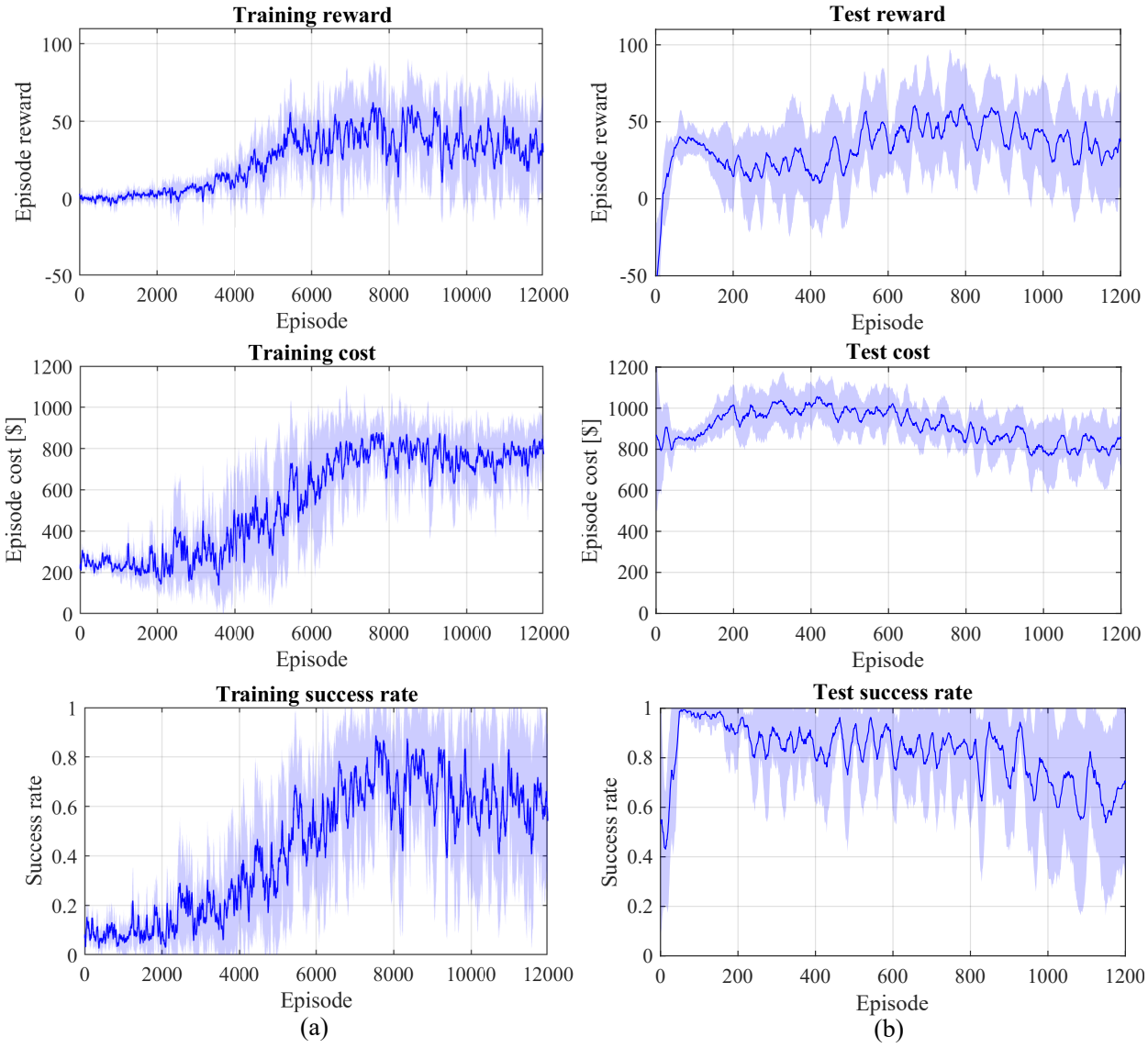

Figure 9: Double DQN agent training and testing with MSE loss function. The deep blue lines are moving average values across 10 instances running with different random seeds. The light blue shadows are the confidence bounds calculated by mean values ± standard deviations across the 10 instances.

6. Validation results

6.1. Overview

The results presented in this section were acquired with the Double DQN agent trained by the Adam optimiser with Huber loss function. Note that the training and validation datasets, sample voyages and model parameters are identical to those used in [9]. Table 2 compares the average voyage cost of the strategies generated by Double DQN of this study with the discrete Double Q-learning and off-line DDP (with a SOC resolution of 0.0125) solutions from [9]. The strategy generated by the Double DQN achieves average costs of \$782.5 and \$768.9 for the training and validation voyages respectively. The off-line (Deterministic Dynamic Programming) DDP strategy average voyage cost is 94.6% and 94.3% of those of the Double DQN strategy, for the training and validation datasets respectively. It is worth mentioning that the DDP strategy is acquired for each voyage independently by providing complete power profiles before solving, representing the best that could theoretically be achieved but requires pre-existing knowledge of power profiles, which is not possible in a real stochastic environment. Therefore, the DDP strategy can only be used as a benchmark to assess other on-line EMS performance. Compared to the Double Q strategy in discrete state space, the Double DQN strategy further reduces the average voyage costs by 5.5% with continuous state space. Note that the Double Q strategy is obtained with a SOC resolution 0.05, while it is continuous for the Double DQN strategy.

Table 2: Double DQN, Double Q and DDP strategy average voyage costs comparison.

	DDP [\$]	Double Q [\$]	Double DQN [\$]	$\frac{\text{DDP}}{\text{Double Q}}$ [%]	$\frac{\text{DDP}}{\text{Double DQN}}$ [%]
SOC resolution	0.0125	0.05	Continuous	-	-
Training voyages	740.0	831.8	782.5	89.0	94.6
Validation voyages	724.9	813.8	768.9	88.9	94.5

The computation time required by the Double DQN agent to generate a strategy is

approximately 27 min using a single thread of an Intel Xeon E5-2683 V3 processor (18 min on an Intel i7-4790 processor). The Double Q agent requires 288 min to generate a strategy using a single thread of an Intel i7-4790 processor. The Double DQN agent managed to reduce the voyage cost by 5.5% with 93.8% less computational resource required, in comparison with the Double Q agent.

As the EMS is intended for use on future voyages for which, of course, there would be no predetermined data, the Double DQN strategy is applied to a set of validation voyages to examine its performance against load profiles that have not been experienced by the agent.

6.2. Validation sample 1 with low power demand

Figure 10 compares the Double DQN strategy (Figure 10b) with the Double Q strategy (Figure 10a) for a validation sample voyage with low power demand. As in Figure 10b, the Double DQN strategy delays the increase of the fuel cell power until the battery SOC has dropped to 0.36 (750 s). During cruising, the fuel cell power is maintained in a narrow band. However, the Double DQN strategy tends to adjust PEMFC power output repeatedly. Nevertheless, unnecessary large adjustments as in Figure 10a (1950–2300 s) have been avoided. Moreover, the minimum battery SOC of the Double DQN strategy is 0.35, while it is 0.4 for the Double Q strategy.

Table 3 details the cost and Global Warming Potential (GWP) emission breakdowns for validation sample voyage 1. The Double DQN strategy reduces the voyage cost by 7.0%, while increasing the GWP emission by 6.9%. This is due to the conflict between voyage cost and GWP emission. PEMFC degradation cost is reduced by 8.3% by avoiding unnecessary PEMFC power adjustments and by making more use of the battery.

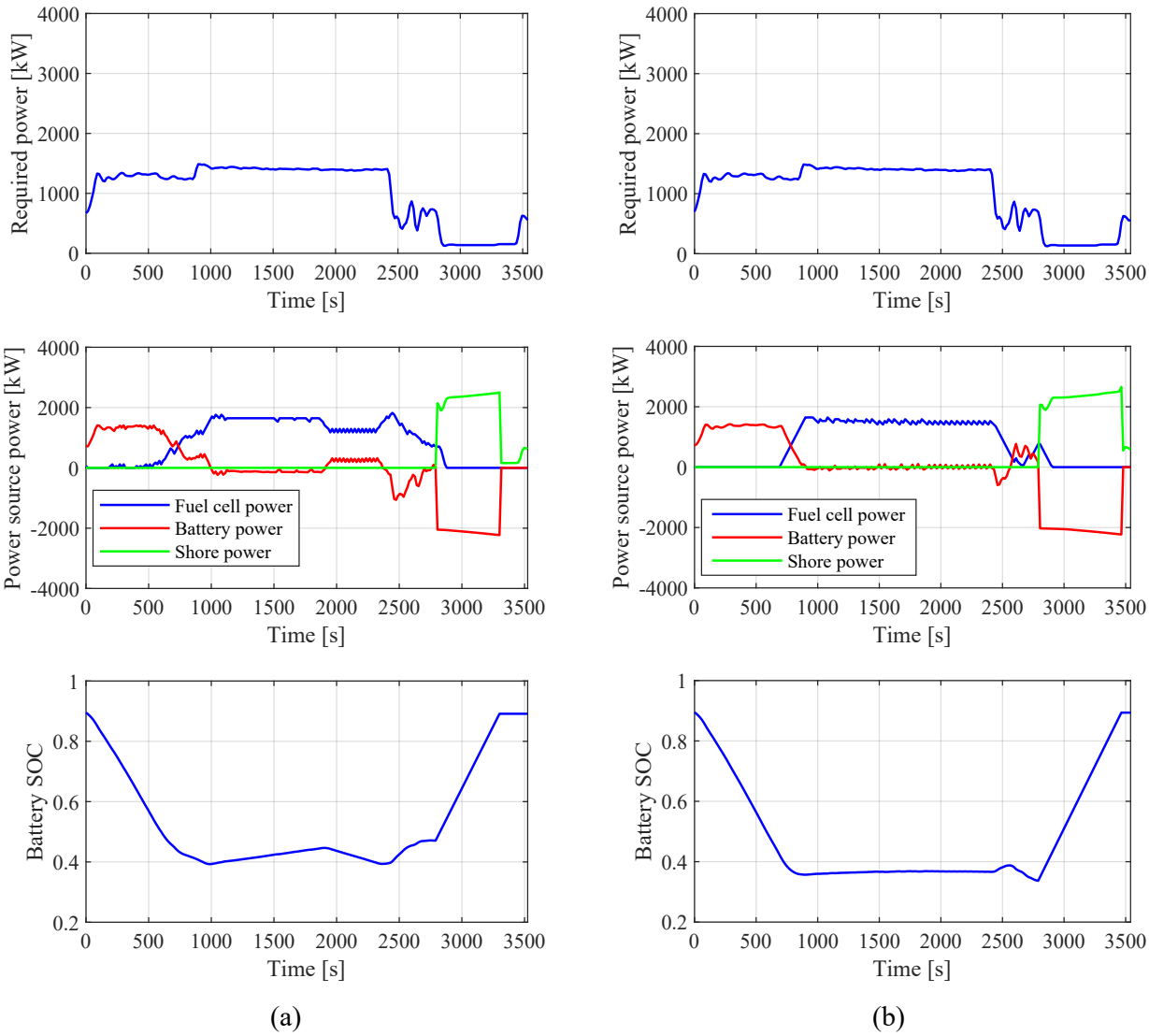

Figure 10: Double Q and Double DQN energy management strategies for validation sample voyage 1 with a low overall power demand: (a) Double Q strategy and (b) Double DQN strategy.

Table 3: Double DQN and Double Q strategy voyage cost and GWP emission breakdowns of validation sample voyage 1.

	Voyage cost			Voyage GWP Emission		
	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$
	[$]	[$]	[%]	[kg]	[kg]	[%]
PEMFC	224.3	244.7	91.7	-	-	-
Battery	63.7	63.7	100.0	-	-	-
Electricity	40.0	31.2	128.2	74.8	58.3	128.2
H_2	366.5	406.8	90.1	66.7	74.1	90.1
Sum	694.5	746.5	93.0	141.5	132.4	106.9

6.3. Validation sample 2 with moderate power demand

Figure 11 illustrates the Double DQN strategy (Figure 11b) in comparison with the Double Q strategy (Figure 11a) for a sample voyage with an overall moderate power demand from the validation voyage load profiles. The Double DQN strategy starts ramping up the PEMFC output at 700 s. As in Figure 11b, the power trajectory of the PEMFC is much smoother compared to that in Figure 11a. The batteries absorb the small power transients by frequent charging and discharging. In addition, when approaching port (2300–2750 s), the Double DQN starts to decrease fuel cell power in advance. Such behaviour has not been observed with the Double Q strategy.

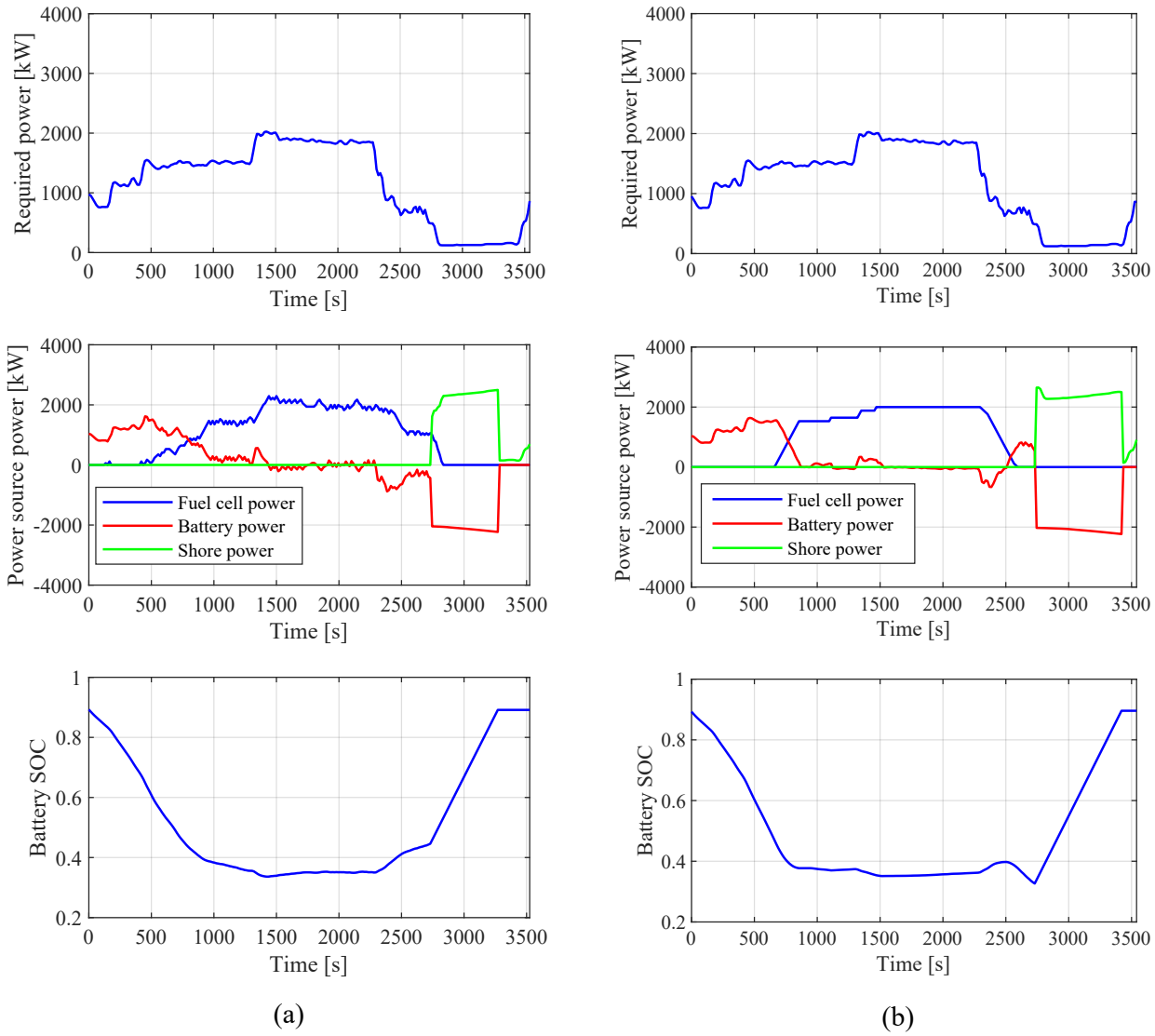

Figure 11: Double Q and Double DQN energy management strategies for validation sample voyage 2 with a moderate overall power demand: (a) Double Q strategy and (b) Double DQN strategy.

Table 4 compares the voyage cost and GWP emission breakdowns of the two strategies for validation sample 2. The Double DQN strategy reduces the voyage cost by 8.4% for this voyage. As the PEMFC power adjustments are less frequent, the PEMFC degradation cost of the Double DQN strategy is reduced by 14.7%. The Double DQN strategy increases the electricity cost by \$9.6 but reduces the H_2 cost by \$43.2. The Double DQN strategy tends to use more shore-generated electricity to achieve lower overall voyage cost. Such a tendency

would increase the electricity cost slightly but would bring greater cost reduction from H_2 consumption. However, the Double DQN strategy increases voyage GWP emission by 7.0%.

Table 4: Double DQN and Double Q strategy voyage cost and GWP emission breakdowns of validation sample voyage 2.

	Voyage cost			Voyage GWP Emission		
	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$
	[$]	[$]	[%]	[kg]	[kg]	[%]
PEMFC	204.5	239.6	85.3	-	-	-
Battery	63.7	63.7	100.0	-	-	-
Electricity	42.0	32.4	129.9	78.6	60.5	129.9
H_2	434.4	477.6	91.0	79.1	86.9	91.0
Sum	744.7	813.2	91.6	157.7	147.4	107.0

6.4. Validation sample 3 with high power demand

Figure 12 details the Double DQN and Double Q strategies for validation sample 3. This sample has a relatively high power demand (with an average power requirement of 1597.8 kW). Although the PEMFC power trajectories of the two strategies follow similar trends in general, the Double DQN strategy maintains the PEMFC power more consistently and only makes adjustments when significant power transients have been observed (e.g. at 1450 s). Also, the Double DQN discharges the battery to a SOC of around 0.26 (close to the lower SOC limit). In addition, the Double DQN decreases the PEMFC output in advance of reaching the port and reduces fuel cell power output to zero when shore power is available.

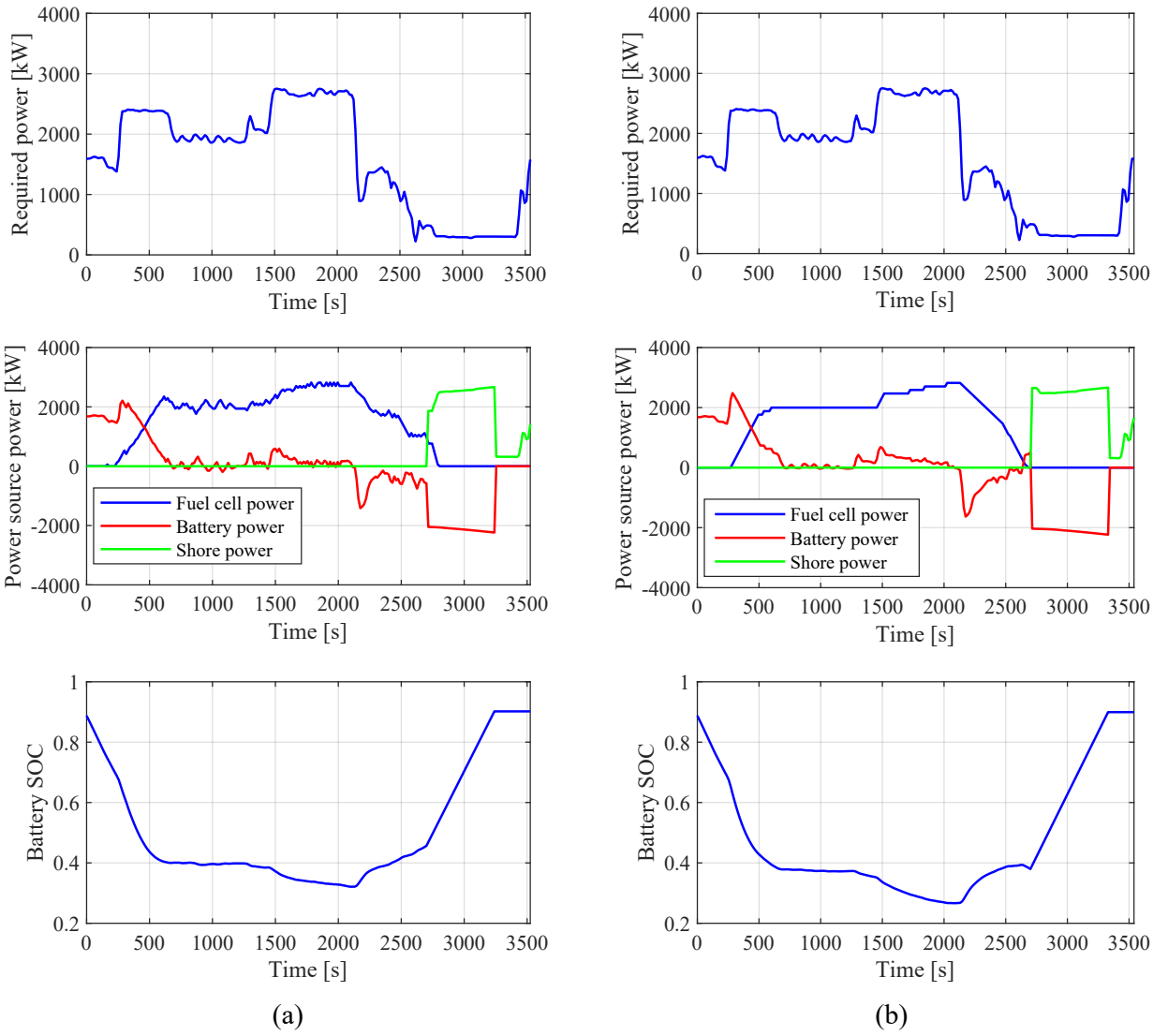

Figure 12: Double Q and Double DQN energy management strategies for validation sample voyage 3 with a high overall power demand: (a) Double Q strategy and (b) Double DQN strategy.

Detailed voyage cost and GWP emission breakdowns of the two strategies for this voyage are detailed in Table 5. The voyage costs of the Double DQN and Double Q strategies are \$1056.7 and \$1093.0, respectively, corresponding to a 3.3% voyage cost difference. The cost saving of 3.3% is lower compared to those voyages discussed in Section 6.2 and 6.3. The reason for the reduced cost-saving is that the Double DQN strategy only adjusts fuel cell power when necessary.

Table 5: Double DQN and Double Q strategy voyage cost and GWP emission breakdowns of validation sample voyage 3.

	Voyage cost			Voyage GWP Emission		
	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$
	[$]	[$]	[%]	[kg]	[kg]	[%]
PEMFC	245.7	257.4	95.5	-	-	-
Battery	63.7	63.7	100.0	-	-	-
Electricity	43.5	36.9	117.9	81.4	69.1	117.9
H_2	703.7	734.9	95.8	128.1	133.8	95.8
Sum	1056.7	1093.0	96.7	209.5	202.8	103.3

6.5. Summary of results

Table 6 summaries the Double DQN strategy performance in comparison with that of the Double Q strategy. The Double DQN strategy further reduces the average cost for the validation dataset by and 5.5%. Compared to the off-line optimum acquired by DDP (see Table 2), the Double DQN strategy cost is only 6.0% higher than that of the DDP strategy in the validation dataset. It appears that the Double DQN gives greatest cost savings for lower power demand profiles. The results are much closer for the high power demand ones, possibly because there is less flexibility in how the system can operate to meet the performance criteria. Additionally, as the agent aims to achieve long-term near-optimum cost performance, consequently the learned policy is better suited for profiles with moderate power demands which are much more common in both training and validation datasets.

It is worth noting that the reduced voyage cost has led to increased GWP emission, owing to the increased usage of shore-generated electricity which is more carbon-intensive in the current simulation settings, from a life-cycle perspective. Readers are referred to the authors' previous work [46, 6] for the trade-offs between costs and GWP emissions which

have been determined in the system design phase. Although the EMS developed in this study has led to a slight increase of in GWP emission, it is acceptable as the plug-in hybrid fuel cell and battery propulsion system is mainly constrained by high costs, and the GWP emission reduction is significant compared to a conventional diesel-based system [4].

Table 6: Comparison of Double DQN and Double Q strategy average voyage costs and GWP emissions.

Category	Profile	Voyage cost			Voyage GWP emission		
		Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$	Double DQN	Double Q	$\frac{\text{Double DQN}}{\text{Double Q}}$
		[$]	[$]	[%]	[kg]	[kg]	[%]
Validation	Sample 1	694.5	746.2	93.1	141.5	132.4	106.9
	Sample 2	744.7	813.0	91.6	157.7	147.4	107.0
	Sample 3	1056.7	1092.7	96.7	209.5	202.8	103.3
Average	all profiles	768.9	813.8	94.5	157.5	149.2	105.5

7. Discussion

As power transition patterns may change over time, a self-adaptive EMS updating procedure can be applied with minimum human intervention. Figure 13 shows the training process of an adaptive EMS. The ship starts with an EMS trained by an initial set of power profiles. A dynamic profile pool is maintained throughout the ship's operation by replacing the oldest profile with the most recent profile. Periodically, the EMS performance is evaluated by comparing the actual EMS performance against those solved via DDP. The agent would need training if the deviation between the on-line and DDP strategies exceeds the performance threshold; otherwise, existing EMS would be applied until the next performance evaluation point. Note that the EMS update indicator can be changed. Nevertheless, the on-line and DDP strategies' cost deviation can be directly interpreted by the vessel operators.

However, this adaptive EMS update scheme is not implemented in this study as the developed Double DQN strategy has shown consistent performance throughout the total

382 validation load profiles in the available dataset. This work has further improved the RL-based EMS developed in [9], with improved energy management performance and significantly reduced training time, making it more realistic for practical applications that would require frequent updates throughout the vessel's deployments.

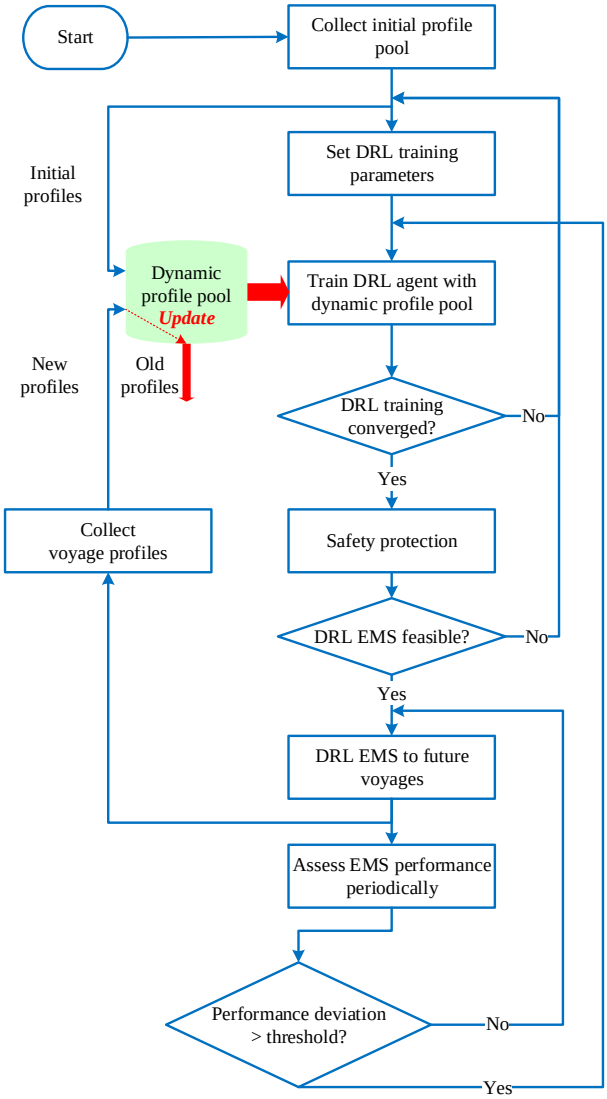

Figure 13: Adaptive EMS update procedure.

8. Conclusions

This work aimed to further improve the cost-effectiveness of reinforcement learning-based energy management strategies by extending discrete state space to be continuous. A

novel approach using deep reinforcement learning has been proposed to solve the optimal energy management problem of the plug-in hybrid Proton Exchange Membrane Fuel Cell and battery propulsion system with continuous state parameters. This novel approach is effective in dealing with large-scale real-world stochastic load profiles, by reducing the impact of function overestimation due to real-world stochasticity.

Two types of loss functions, i.e. Mean Squared Error and Huber loss functions, have been explored to deal with value overestimations in the stochastic environment. The training processes suggest that Double Deep Q-learning with Mean Squared Error loss function is not sufficient to deal with the highly stochastic load profiles. In contrast, the combination of Huber loss function with Double Deep Q-learning has overcome the training instability issue and has produced an energy management strategy with improved cost-effectiveness. The energy management strategy acquired by the Double Deep Q-Network with Huber loss function was examined in detail in comparison to that obtained by the Double Q agent. The Double Deep Q-Network has achieved a further 5.5% voyage cost reduction with 93.8% computational time reduction. The cost reduction is achieved by more accurate fuel cell control and reduced H_2 consumption. However, the Double Deep Q-Network based energy management strategy leads to a 5.5% increase in voyage Global Warming Potential emission because of the conflicts between voyage emission and cost.

Additionally, an adaptive energy management update procedure has been discussed, which can be further tested and developed in future work. The agent-environment framework, training procedures and the energy management update scheme developed in this study can be extended to other hybrid propulsion and power systems with continuous monitoring, to achieve long-term near-optimal performance in changing environments. The energy management system will be extended to control multiple power sources in continuous action space using advanced Deep Reinforcement Learning algorithms in future work.

Acknowledgements

This work is partially supported by Royal Society (Grant no. IEC\NSFC\191633). The authors thank Jens Christian Bjeldorf and Molslinje A/S for approving using the ship data

in this study. The authors are grateful to Stig Eriksen and his colleagues for collecting the ship data.

References

[1] L. van Biert, M. Godjevac, K. Visser, P. Aravind, A review of fuel cell systems for maritime applications, Journal of Power Sources 327 (2016) 345–364.

[2] H. S. Das, C. W. Tan, A. Yatim, Fuel cell hybrid electric vehicles: A review on power conditioning units and topologies, Renewable and Sustainable Energy Reviews 76 (2017) 268–291.

[3] Y. Bicer, I. Dincer, Clean fuel options with hydrogen for sea transportation: a life cycle approach, International Journal of Hydrogen Energy 43 (2018) 1179–1193. doi:10.1016/j.ijhydene.2017.10.157.

[4] P. Wu, Decarbonising coastal shipping using fuel cells and batteries, Ph.D. thesis, University College London, 2020.

[5] P. Wu, R. Bucknall, Marine propulsion using battery power, Shipping in Changing Climates Conference 2016, 2016.

[6] P. Wu, R. Bucknall, Hybrid fuel cell and battery propulsion system modelling and multi-objective optimisation for a coastal ferry, International Journal of Hydrogen Energy 45 (2020) 3193–3208. doi:10.1016/j.ijhydene.2019.11.152.

[7] S. Ma, M. Lin, T.-E. Lin, T. Lan, X. Liao, F. Maréchal, Y. Yang, C. Dong, L. Wang, et al., Fuel cell-battery hybrid systems for mobility and off-grid applications: A review, Renewable and Sustainable Energy Reviews 135 (2020) 110119. doi:10.1016/j.rser.2020.110119.

[8] C. M. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao, M. Wellers, Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective, IEEE Transactions on Vehicular Technology 66 (2016) 4534–4549. doi:10.1109/TVT.2016.2582721.

[9] P. Wu, J. Partridge, R. Bucknall, Cost-effective reinforcement learning energy management for plug-in hybrid fuel cell and battery ships, Applied Energy 275 (2020) 115258. doi:10.1016/j.apenergy.2020.115258.

[10] S. Eriksen, M. Lützen, J. B. Jensen, J. C. Sørensen, Improving the energy efficiency of ferries by optimizing the operational practices, in: Proceedings of the Full Scale Ship Performance Conference 2018: The Royal Institution of Naval Architects, The Royal Institution of Naval Architects, 2018, pp. 101–111.

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

[12] N. Sulaiman, M. Hannan, A. Mohamed, E. Majlan, W. W. Daud, A review on energy management

system for fuel cell hybrid electric vehicle: Issues and challenges, Renewable and Sustainable Energy Reviews 52 (2015) 802–814. doi:10.1016/j.rser.2015.07.132.

[13] H. Banvait, S. Anwar, Y. Chen, A rule-based energy management strategy for plug-in hybrid electric vehicle (phev), in: 2009 American control conference, IEEE, 2009, pp. 3938–3943. doi:10.1109/ACC.2009.5160242.

[14] J. Peng, H. He, R. Xiong, Rule based energy management strategy for a series–parallel plug-in hybrid electric bus optimized by dynamic programming, Applied Energy 185 (2017) 1633–1643. doi:10.1016/j.apenergy.2015.12.031.

[15] Y. Wang, Z. Sun, Z. Chen, Development of energy management system based on a rule-based power distribution strategy for hybrid power sources, Energy 175 (2019) 1055–1066. doi:10.1016/j.energy.2019.03.155.

[16] M. Kalikatzarakis, R. Geertsma, E. Boonen, K. Visser, R. Negenborn, Ship energy management for hybrid propulsion and power supply with shore charging, Control Engineering Practice 76 (2018) 133–154. doi:10.1016/j.conengprac.2018.04.009.

[17] H. Gao, Z. Wang, S. Yin, J. Lu, Z. Guo, W. Ma, Adaptive real-time optimal energy management strategy based on equivalent factors optimization for hybrid fuel cell system, International Journal of Hydrogen Energy 46 (2021) 4329–4338. doi:10.1016/j.ijhydene.2020.10.205.

[18] M. Ebrahim, B. Talat, E. Saied, Implementation of self-adaptive harris hawks optimization-based energy management scheme of fuel cell-based electric power system, International Journal of Hydrogen Energy 46 (2021) 15268–15287. doi:10.1016/j.ijhydene.2021.02.116.

[19] A. M. Bassam, A. B. Phillips, S. R. Turnock, P. A. Wilson, Development of a multi-scheme energy management strategy for a hybrid fuel cell driven passenger ship, International Journal of Hydrogen Energy 42 (2017) 623–635. doi:10.1016/j.ijhydene.2016.08.209.

[20] Y. Wang, Z. Sun, X. Li, X. Yang, Z. Chen, A comparative study of power allocation strategies used in fuel cell and ultracapacitor hybrid systems, Energy 189 (2019) 116142. doi:10.1016/j.energy.2019.116142.

[21] H. Rezk, A. M. Nassef, M. A. Abdelkareem, A. H. Alami, A. Fathy, Comparison among various energy management strategies for reducing hydrogen consumption in a hybrid fuel cell/supercapacitor/battery system, International Journal of Hydrogen Energy (2019). doi:10.1016/j.ijhydene.2019.11.195.

[22] J. Hou, J. Sun, H. Hofmann, Control development and performance evaluation for battery/flywheel hybrid energy storage solutions to mitigate load fluctuations in all-electric ship propulsion systems, Applied energy 212 (2018) 919–930. doi:10.1016/j.apenergy.2017.12.098.

[23] D.-D. Tran, M. Vafaeipour, M. El Baghdadi, R. Barrero, J. Van Mierlo, O. Hegazy, Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated

energy management strategies, Renewable and Sustainable Energy Reviews 119 (2020) 109596. doi:10.1016/j.rser.2019.109596.

[24] R. Geertsma, R. Negenborn, K. Visser, J. Hopman, Design and control of hybrid power and propulsion systems for smart ships: A review of developments, Applied Energy 194 (2017) 30–54. doi:10.1016/j.apenergy.2017.02.060.

[25] J. Li, Z. Hu, L. Xu, M. Ouyang, C. Fang, J. Hu, S. Cheng, H. Po, W. Zhang, H. Jiang, Fuel cell system degradation analysis of a chinese plug-in hybrid fuel cell city bus, international journal of hydrogen energy 41 (2016) 15295–15310. doi:10.1016/j.ijhydene.2016.06.136.

[26] Y. Zhou, H. Obeid, S. Laghrouche, M. Hilairet, A. Djerdir, A novel second-order sliding mode control of hybrid fuel cell/super capacitors power system considering the degradation of the fuel cell, Energy Conversion and Management 229 (2021) 113766. doi:10.1016/j.enconman.2020.113766.

[27] L. Vichard, N. Y. Steiner, N. Zerhouni, D. Hissel, Hybrid fuel cell system degradation modeling methods: A comprehensive review, Journal of Power Sources 506 (2021) 230071. doi:10.1016/j.jpowsour.2021.230071.

[28] T. Liu, Y. Zou, D. Liu, F. Sun, Reinforcement learning of adaptive energy management with transition probability for a hybrid electric tracked vehicle, IEEE Transactions on Industrial Electronics 62 (2015) 7837–7846. doi:10.1109/TIE.2015.2475419.

[29] R. Xiong, J. Cao, Q. Yu, Reinforcement learning-based real-time power management for hybrid energy storage system in the plug-in hybrid electric vehicle, Applied energy 211 (2018) 538–548. doi:10.1016/j.apenergy.2017.11.072.

[30] B. Xu, X. Hu, X. Tang, X. Lin, H. Li, D. Rathod, Z. Filipi, Ensemble reinforcement learning-based supervisory control of hybrid electric vehicle for fuel economy improvement, IEEE Transactions on Transportation Electrification 6 (2020) 717–727. doi:10.1109/TTE.2020.2991079.

[31] J. Wu, H. He, J. Peng, Y. Li, Z. Li, Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus, Applied energy 222 (2018) 799–811. doi:10.1016/j.apenergy.2018.03.104.

[32] Y. Wu, H. Tan, J. Peng, H. Zhang, H. He, Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus, Applied Energy 247 (2019) 454–466. doi:10.1016/j.apenergy.2019.04.021.

[33] G. Du, Y. Zou, X. Zhang, T. Liu, J. Wu, D. He, Deep reinforcement learning based energy management for a hybrid electric vehicle, Energy (2020) 117591. doi:10.1016/j.energy.2020.117591.

[34] S. Hasanvand, M. Rafiei, M. Gheisarnejad, M.-H. Khooban, Reliable power scheduling of an emission-free ship: Multi-objective deep reinforcement learning, IEEE Transactions on Transportation Electrification (2020). doi:10.1109/TTE.2020.2983247.

[35] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529. doi:10.1038/nature14236.

[37] J. A. Boyan, A. W. Moore, Generalization in reinforcement learning: Safely approximating the value function, in: Advances in neural information processing systems, 1995, pp. 369–376.

[38] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in neural information processing systems, 2000, pp. 1057–1063.

[39] J. Tsitsiklis, B. Van Roy, An analysis of temporal-difference learning with function approximation, IEEE Transactions on Automatic Control 42 (1997) 674–690.

[40] L. Lin, Reinforcement learning for robots using neural networks, Ph.D. thesis, Carnegie-Mellon University Pittsburgh PA School of Computer Science, 1993.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint (2013).

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint (2015).

[43] P. J. Huber, Robust estimation of a location parameter, in: Breakthroughs in statistics, Springer, 1992, pp. 492–518.

[44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint (2014).

[45] H. van Hasselt, Double q-learning, in: J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, A. Culotta (Eds.), Advances in Neural Information Processing Systems 23, Curran Associates, Inc., 2010, pp. 2613–2621.

[46] P. Wu, R. Bucknall, On the design of plug-in hybrid fuel cell and lithium battery propulsion systems for coastal ships, in: P. Kujala, L. Lu (Eds.), 13th International Marine Design Conference (IMDC 2018), volume 2, CRC Press/Balkema, London, 2018, pp. 941–951.