Power Flow Management in Electric Vehicles Charging Station Using Reinforcement Learning

Arwa O. Erick
Department of Electrical Engineering
University of Cape Town
South Africa
arweri001@myuct.ac.za

Komla A. Folly
Department of Electrical Engineering
University of Cape Town
South Africa
komla.folly@uct.ac.za

***Abstract*—This paper investigates optimal power flow management problem in an electric vehicle charging station. The charging station is powered by solar PV and is tied to the grid and a battery storage system through necessary power conversion interfaces for DC fast charging. The optimal power management problem for EV charging is solved via reinforcement learning (RL). Unlike classical optimization methods such as dynamic programming, linear programming (LP) and mixed-integer linear programming which are limited in handling stochastic problems adequately and are slow due to the curse of dimensionality when used for large dynamic problems, RL does not have to iterate for every time step as learning can be done completely offline and optimal solutions saved in a lookup table, from which optimal control actions can be retrieved almost instantaneously. The optimization problem in this paper is defined as a Markov Decision Process (MDP) and a modified Q-learning algorithm that indexes both states and control actions in a hash-table (dictionary) fashion is used to solve it. The algorithm is tested with a typical load curve over a 24-hour horizon. The simulations results demonstrate that the modified Q-learning algorithm achieves higher total rewards and returns a 14% lower global cost than the conventional Q-learning formulation.**

Keywords—Electric vehicles, charging station, reinforcement learning, renewable energy, Q-Learning.

I. Introduction

The substitution of internal combustion engine vehicles (ICEVs) with electric vehicles (EVs) has been proposed to reduce greenhouse gas emissions [1]. Current grids are still predominantly fossil fuel-powered, there is a need to shift from conventional fossil fuel-based power system to a clean fuel made of more renewable energy sources (RES) [2], [3]. Only under such conditions that supplying EVs with electricity from the grid will be sustainable. Solar and wind power have been used to charge EVs, but the cities are space-constrained, hence RES will be insufficient on their own. Grid-tied RES with Battery Behind Meter (BBM) architecture has successfully been applied instead [2]. Solar PVs are more adaptive to the spatial limitation in cities. There is a need to develop optimal power management algorithms that will ensure EV loads are supplied at competitive charging rates while minimizing the cost of imported grid electricity and other operational costs such as battery degradation due to temperature and depth of discharge [3].

Optimal power management algorithms for BBM systems have been developed around the battery scheduling approach to control EV charging. In [4] the authors used linear programming with Model Predictive Control, classical (synchronous) dynamic programming (DP) has been employed in [5] and [6], and mixed-integer linear programming in [7]. These algorithms are limited in the size and complexity of problems they can solve and cannot handle stochastic problems. Global search methods like swarm intelligence-based methods and genetic algorithm-based techniques face problems of slow convergence and inability to operate online [8], [9]. The main problem with the battery scheduling approach is that it assumed that all power from RES must first pass through the battery and this is expensive when battery degradation cost is considered [10]. The battery scheduling approach has been used in [12]. However, the main issue with this approach is that it puts much stress on the battery and cannot handle additional energy sources. In [13], the authors proposed a priority list approach to solve the problem of energy management in the EV charging station. However, this approach suffers fundamental drawbacks in that it requires an optimal initial state of charge of the battery to be determined before the solution can be optimal, otherwise, the algorithm may give wrong priorities. The method only works with a static grid tariff and cannot handle the complexity that comes with a dynamic grid tariff.

In the past, several algorithms have been used to solve Markov Decision Process (MDP). The oldest of them is DP which has traditionally been used to find optimal policies through value iteration provided the environment is Markovian in nature. A value function is initialized and successively evaluated and improved recursively using the Bellman equation until an optimal value function is achieved which satisfies the *Bellman optimality* criterion given in [11], from which an optimal policy can be obtained [12]. However, classical DP which is synchronous suffers from serious limitations related to the size and complexity of problems it can solve [14], [13].

Reinforcement learning (RL) is one of the most popular solutions for sequential decision-making problems. In RL, a learning environment is developed with states occurring sequentially, such that the current state contains all information required to get to the next state. Such an environment is said to be Markovian. A learning agent is designed to interact with the environment in stages and iteratively in order to obtain a policy that recommends the best action to take in each of the possible states. The environment has states which the agent observes, takes actions and receives feedback as to the goodness of the actions taken in the individual states [14]. RL algorithms that borrow from DP methodology include Monte Carlo learning, temporal difference (TD) learning and Q-learning [14][12]. Q-learning is similar to TD, however, unlike TD and Monte Carlo learning which learn on-policy, it is an off-policy method and imposes more relaxed computational demands [15]. Unlike the methods used in [5], [9] and [10], that run quasi-real-time, suffer computational speed problems and cannot adapt to the stochasticity of the power management environment, RL does not have to iterate for every time step as learning can be done completely offline and optimal solutions saved in a lookup table, from which optimal control actions can be retrieved almost instantaneously [16].

In this paper, a power scheduling task for a solar PV-based EV charging station tied to the utility grid and with the integration of a battery energy storage system (BSS) is performed using reinforcement learning (RL). The charging station's daily operation has been expressed as a Markov Decision Process (MDP) problem and an asynchronous Q-learning method that indexes both states and control actions in a hash-table (dictionary) fashion is proposed. The state space is accessed asynchronously, i.e., state-actions-set pairs are added as they occur, and the agent does not sweep through the entire state space in each episode. The algorithm is tested with a typical forecasted electric vehicle charging load curve and a PV generation profile over a 24-hour horizon. The simulation results show that the agent achieves higher total rewards using this method and returns about 14% lower global cost than the conventional Q-learning algorithm.

The rest of this paper is organized as follows: Section II deals with the charging station model, section III outlines the mathematical formulation of the optimization problem, section IV deals with the MDP model of the problem, section V describes the RL solution, section VI deals with results and discussions and section VI presents the conclusions.

II. EV Charging Station

The charging station consists of a solar PV generator with a grid-tie supply, all interconnected through the necessary power conversion equipment. In EV charging stations, a common bus is necessary to maintain the same power level amongst all the plugged-in chargers [17]. The DC bus system has a common DC link with a common DC voltage so that all other supplies are connected to it through an appropriate power conversion device. The grid, the PV generator and the battery storage are linked to the DC bus through appropriate power conversion equipment, and the solar PV generator is linked to the DC bus by an MPPT enabled boost convertor, and the BSS is also linked through a bidirectional DC to DC converter. Fast charging is done at DC voltages, thus, the DC link enables connection of the power sources the electric vehicle charging equipment through only one boost converter [17]. The optimization task assumes that the PV, the station's load profile and the day-ahead tariff have been forecasted and are fed into the algorithm for scheduling purposes.

III. Mathematical Formulation

The optimization problem can be viewed as a stochastic battery scheduling or unit commitment problem [18]. In the battery scheduling formulation, the solver is designed to consider the load, the RES output and the grid tariff at every time step and find the optimal schedule of BSS state of charge (SoC) from an initial SoC to the final value at the end of the optimization. The SoC schedule determines the amount of power to be supplied to or taken from the battery at every time step. The solver then considers the load and commits the remaining RES output to supply the load. The excess RES generator output is then sent to the grid whereas if there is a deficit, power is purchased from the grid. This formulation can be found in [10], [19] and [20]. A major challenge with this formulation is that it causes much strain on the battery and results in higher cost of degradation, thus, the resultant schedule may not be optimal. The unit commitment formulation considers each source as a separate unit with the battery as a special unit that can absorb or generate power (prosumer). The optimal schedule is determined for all the units simultaneously as described in [5] and [31]. In this paper, the unit commitment formulation is adopted.

The key purpose of the optimal scheduling task is to minimize the operative cost of supplying a given charging station's load profile within system constraints. The algorithm will take into account the cost incurred in purchasing from the utility grid and the BSS degradation cost The instantaneous amount of energy in the BSS is given by $E(t) = SoC(k)E_b$ where $SoC(t)$ is the measured or estimated state of charge and E_b is the battery energy at full charge. At every time step, the power balance equation is given by:

$$P_{cl}(t) = P_{bss}(t) + P_g(t) + P_{pv}(t), \tag{1}$$

where $P_{cl}(t)$, $P_{bss}(t)$, $P_g(t)$ and $P_{pv}(t)$ are the instantaneous charging station's load, battery power, grid power and the PV power. $P_{cl}(t)$ and $P_{pv}(t)$ are forecasted values, while $P_{bss}(t)$ and $P_g(t)$ are a result of the algorithm's decision every time step. Therefore, the instantaneous cost of the decision, $C(t)$, is given by:

$$C(t) = C_{P_g}(t) + C_{P_{bss}}(t), \tag{2}$$

where $C_{P_g}(t)$ is the cost of power purchase from the grid and $C_{P_{bss}}(t)$ is the cost of battery degradation as a result of charge or discharge operations. The battery power is taken to be positive when discharging and negative when charging. The cost of degradation is assumed to be independent of the direction of power flow in the battery. The grid power is taken to be positive when it is supplying power and negative when it is absorbing power from the charging station. For simplicity, the revenue from power export to the grid has been ignored.

The objective function of the optimization problem is given by:

$$Min\,(C_{tot}) = Min \sum_{t=0}^{T} \left[C_{P_g}(t) + C_{P_{bss}}(t)\right], \tag{3}$$

in which T is the total number of time-steps in the optimization period, in this case, 24-hours. The above equation is limited by power equilibrium at the DC bus of the electric vehicle supply equipment, battery state of energy boundaries, $SoE_{min} \le SoE(t) \le SoE_{max}$, battery power limits, $P_{bss}^{min} \le P_{bss}(t) \le P_{bss}^{max}$ and the value of power from the utility constraint, $P_g^{min} \le P_g(t) \le P_g^{max}$. The grid power constraint is governed by a contract between the charging station (CS) owner and the distribution system operator [23].

If a dynamic tariffing regime is considered, the cost of energy import from the utility grid scheduled for every time step is given by $C_{Pg}(t) = G_t(t)\, P_g(t)\Delta t$, where $G_t(t)$ is the cost per kWh of power from the grid and Δt is the time-step size, in this case, 1-hour. The cost of drawing power from or storing power in the BSS is given by $C_{P_{bss}}(t) = P_{bss}(t)C_{bd}(t)\Delta t$, where $C_{bd}(t)$ is the cost of degradation and has been derived in [10] and [24]. This is given in (4).

$$C_{bd} = C_{bt} max \left\{ \left(\int_{t_o}^{t_f} \frac{dt}{Y_h L_t(T)} \right), \left(\left[\left(\frac{1}{L(DoD_2)} \right) - \left(\frac{1}{L(DoD_1)} \right) \right] \right), \left(\frac{mSoC_{av} - d}{Q_{fade} n Y_h} \right) \right\} \quad (4)$$

where C_{bt} is the battery purchase price per kWh, t_o and t_f are initial and final battery operation time for charge or discharge operations, Y_h is the number of hours in a year, $L(DoD_j)$ is the number of cycle lives of the battery at DoD_j, $L_t(T)$ is the battery life as a function of battery ambient temperature [25], Q_{fade} is the capacity fade at battery end of life, SoC_{av} is the average SoC and m, n and d are curve fitting constants. The degradation equation has been derived in [10].

IV. Markov Decision Process

For the scheduling task to be performed using reinforcement learning, it has to be defined as a Markov Decision Process (MDP). With a Markov property for the states, the learning agent does not need the memory of previous events, but a single set of state variables for the current state is assumed to contain all the information the agent needs to make the right decision. Thus, MDPs are an accepted essential construction for reinforcement learning [26]. An MDP consists of a set of states within a predefined state space and a set of actions defined for each state and is normally used to formalize chronological decision making. For every state, there is a known stationary state transition function (or probability) that leads the agent to the next state once it takes an action in the current state [27]. Also, for every action, there is a defined reward (reinforcement) function that measures the immediate value of the action taken. The reward function holds the objective of the agent at every state. In this case, the reward is related to the cost minimization function of the EV charging operation.

A. *State and State Space*

The system state, x_k defined as the set $\{k, P_{cl}^k, P_{pv}^k, G_t^k, E_b^k\}$, where k is the state index, also representing the time-step index: $k = 0,1,\ldots,T-1$, in which T is the number of states in one episode of scheduling, P_{cl}^k is the charging station's instantaneous load at k, P_{pv}^k is power from the PV generator at k, G_t^k is the cost per kWh of power from the grid at k, while E_b^k is the BSS energy at k. The state-space is thus defined as a union of all the set of states in *T*: $\chi = x_0 \cup x_1 \cup, \ldots, \cup\, x_{T-1}$ [18], [23].

B. *Action and Action Space*

An action represents the decision that the agent is supposed to make at any given state. Every state has a list of allowed decisions that define the action space for that state. In this scheduling problem, the action is the decision on what amount of power is to be drawn from or absorbed by the stationary battery, the amount of power to be imported from the grid. Thus, the action at a time k is consequently defined as $a_k = \{P_{bss}^k, P_g^k\}$. The action space is defined here as a dependent on the state variables: $\mathcal{A}_k = f(x_k)$. Thus, possible actions are limited by the grid power and battery energy level according to: $P_g^{min} \le P_g^k \le P_g^{max}$ as well as $E_b^{min} \le E_b^k - P_{bss}^k \Delta t \le E_b^{max}$. Therefore, to define the action space for every state, the power deficit, given by $\Delta P_k = P_{cl}(k) - P_{pv}(k)$ is calculated. Given this value, the system computes all possible combinations of $P_{bss}^k\ and\ P_g^k$ such that $\Delta P_k = P_{bss}^k + P_g^k$.

The limitation of the action space to a set of feasible solutions within the boundaries helps the agent to avoid exploring decisions which the designer already knows are not candidate solutions according to the system constraints. This reduces the problem to that of looking for optimal actions within an action space that is within the boundaries and improves the performance of the algorithm as will be seen in section VI. Consequently, the set of all state-dependent action spaces defined by, $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup, \ldots, \cup\, \mathcal{A}_{T-1}$, gives the full set of the agent's action space.

C. *State Transition Function*

After the learning agent takes an action, there is a need for a state transition function that takes the agent to the next state. As stated before, in an MDP with Markovian property, the next state is dependent on the current state and not the sequence of actions that led to the current state. In partially observable MDPs, the agent does not have full access to states but just observations that relay limited information of the states, and the agent's experience of these observations form the agent state [28]. The next agent state may then be derived from the experience of the next observations. In this case, the MDP is fully observable. The state transition is defined as: $x_{k+1} = f(x_k, a_k, x_{k+1})$ [14], where x_{k+1} is the vector containing the system inputs for the following state with the values of load P_{cl}^{k+1}, the PV generator output P_{pv}^{k+1}, and the grid tariff G_t^{k+1} as well as updated energy level of the BSS given by $E_b^{k+1} = E_b^k \pm P_{bss}^k \Delta t$. The state transition is thus deterministic such that the next state x_{k+1} is defined as follows (5).

$$x_{k+1} = \{k+1, P_{cl}^{k+1}, P_{pv}^{k+1}, G_t^{k+1}, E_b^{k+1}\} \quad (5)$$

D. *Reward Function*

A reward is any scalar quantity that is meant to relay the purpose of the learning algorithm to the agent. Appropriate "reward engineering" is essential to link the agent's actions with the objective of the algorithm [29]. The reward is defined as: $r(k) = g(x_k, a_k, x_{k+1})$. The intention of the learning process is to minimize the cost of power purchase from the utility grid and lessen strain on the BSS by reducing the degradation. Consequently, the reward function for this purpose is given by (6) as:

$$r(k) = \frac{1}{C_{P_g}(t) + C_{P_{bss}}(t) + 1} \quad (6)$$

In equation (6) above, the reward is defined as the inverse of the cost so that while the agent learns to maximize this reward, it minimizes the cost. The addition of 1 helps to avoid division by zero in situations where the instantaneous cost is zero.

V. Q-Learning Solution of the MDP

A. *Introduction to Reinforcement Learning*

In reinforcement learning, an agent is modelled to make optimal decisions by interacting with the environment in sequentially and iteratively. In each environment's state, the learning agent witnesses the state of the environment, takes an action, gets to the following state, and receives a scalar quantity as a reward. The goal of this software agent is to optimize its cumulative expected discounted returns. In the process, the agent obtains the knowledge of a policy that maps every state to the best action [15] [14]. Generally, if an agent visits a state x, performs an action endorsed by a policy, π, and thus transits to the next stage y, the value the new state may be computed by (7).

$$V^{\pi}(x) = r(\pi(x)) + \gamma \sum_{y} P_{xy}[\pi(x)] V^{\pi}(y) \qquad (7)$$

where $r(\pi(x))$ is the instantaneous return, P_{xy} is the state transition probability, $V^{\pi}(y)$ is the value of state y and $\gamma \in (0,1)$ is the discount factor, which controls the value of future returns in the current state . Equation (7) means that the value of an action in a state is the immediate reward plus the total expected discounted reward in upcoming states that the action leads to. *R. Bellman* [11] established that there is at least one optimal policy, π^*, for which the state value given by (8) below is the best the agent can achieve in the state x:

$$V^*(x) = V^{\pi^*}(x) = \max_{a} \left\{ r(a) + \gamma \sum_{y} P_{xy}[\pi(x)] V^{\pi^*}(y) \right\} \qquad (8)$$

In Q-learning, though, the agent is ignorant of these values in advance, but still acquires the knowledge of the optimal policy, π^*, mapping every state to the optimal action by using the state-action value function. The state-action value function, $Q(x,a)$ is the state-action value, such that [30]:

$$Q(x,a) = r_x(a) + \gamma \sum_{y} P_{xy}[\pi(x)] V^{\pi}(y). \qquad (9)$$

The optimal Q-value for the optimal policy is such that: $Q^*(x,a) = Q^{\pi^*}(x,a), \forall\, x \in \chi, \forall a \in \mathcal{A}$. If we can have all the Q-values of all permissible actions in any state x, performed enough times, then the action that maximizes the Q-value in a given state is the optimal action. The optimal state-action value for a minimization problem is given by:

$$Q^*(x,a) = \min_{\pi} Q^{\pi}(x,a), \forall\, x \in \chi, \forall a \in \mathcal{A}. \qquad (10)$$

so that the optimal policy in state x is defined as: $\pi^*(x) = argmin_{a \in \mathcal{A}} Q^*(x,a)$. That means the optimal action in that state is such that $Q^*(x,a^*) > Q(x,a_i), \forall\, a_i \neq a^*$. The optimal action, a^*, for the state x, is referred to as the greedy action, a_g.

During each episode, the agent gets to each of the states x_k, action space, $\mathcal{A}_k$, and using a given selection strategy, chooses an action a_k and as a result, transits to the next state x_{k+1}, receiving an instant reward, $r = g(x_k, a_k, x_{k+1})$. Then the Q-values are adjusted as given in (11) below [30].

$$Q^{n+1}(x,a) = Q^n(x,a) + \alpha\big[g(x_k, a_k, x_{k+1}) + \gamma max_{a_{k+1}} Q^n(x_{k+1}, a_{k+1}) - Q^n(x,a)\big] \qquad (11)$$

where $\alpha \in (0,1)$ is the learning rate which controls the extent of modification of Q-values, $Q^n(x,a)$ is the current Q-value, $Q^{n+1}(x,a)$ is the next Q-value while $\gamma \in (0,1)$ is the discount factor. If α is sufficiently small, Q^n converges to Q^* after enough iterations. If the present state is a final state, then there is no following state in the episode, therefore, the Q-value is updated according to equation (12).

$$Q^{n+1}(x,a) = Q^n(x,a) + \alpha[g(x_k, a_k, x_{k+1}) - Q^n(x,a)] \qquad (12)$$

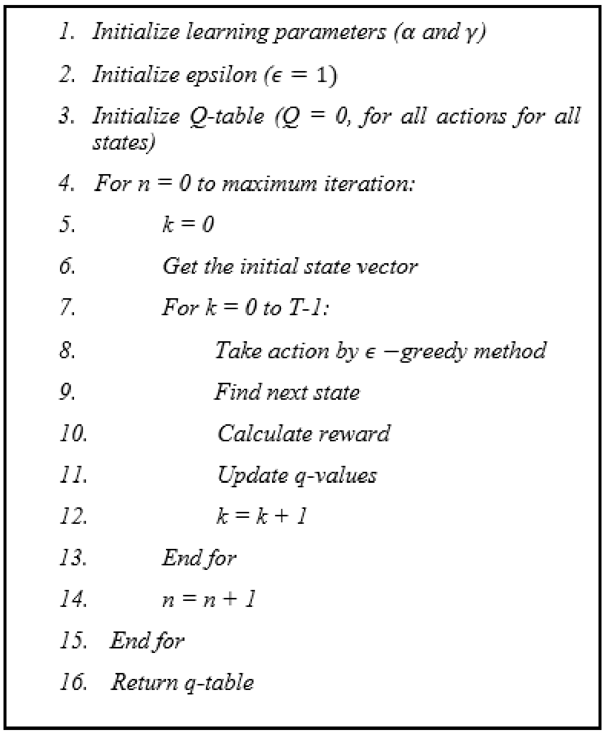
As the agent selects an action from the action space, there is a necessity to balance exploration with exploitation. Exploration helps evade being stuck in a local extremum while exploitation enables convergence in later stages. The common strategies for balancing exploration with exploitation are to use the *ϵ-greedy*, the *pursuit method* or the Boltzmann's SoftMax method [18], [31]. We consider the *ϵ-greedy* method due to its ease of implementation. In the ϵ-greedy algorithm, the greedy action has the probability 1- ϵ, for $\epsilon \in (0,1)$, of being chosen in every given episode, while the rest of the actions in the action space, $\mathcal{A}_k$, have the probability ϵ of being chosen by the agent. [32]. We also used exponential decay function this gradual cut so that in every episode, the value of ϵ is updated thus: $\epsilon = min\_\epsilon - (max\_\epsilon - min\_\epsilon) exp\{-\mathbb{C} \times n\}$, where min_ϵ is the minimum value of ϵ, max_ϵ is the maximum value of ϵ, $\mathbb{C}$ is the cooling constant and *n* is the sum of all episodes.

B. Algorithm 1: Conventional Q-Learning

In the conventional Q-learning, a Q-table is defined for the entire state-action space before learning begins as a state by possible actions matrix. The action space is fixed to a set of possible actions for all the states. The agent, thus, accesses the states sequentially and synchronously, each time selecting an action and the Q-value for the state-action pair is updated. The Q-table for the conventional Q-learning is illustrated in Table 1 below. The Q-learning algorithm has been described in [12], [14] and used for energy scheduling in [20]. The flow of events in the conventional Q-learning is shown in Fig.1 below.

TABLE I. CONVENTIONAL Q-TABLE

State/Action	***a1***	***a2***	***a3***
s1	Q(s1, a1)	Q(s1, a2)	Q(s1, a3)
s2	Q(s2, a1)	Q(s2, a2)	Q(s2, a3)



```
1.  Initialize learning parameters ($\alpha$ and $\gamma$)
2.  Initialize epsilon ($\epsilon = 1$)
3.  Initialize Q-table (Q = 0, for all actions for all
    states)
4.  For n = 0 to maximum iteration:
5.          k = 0
6.          Get the initial state vector
7.          For k = 0 to T-1:
8.                  Take action by $\epsilon$ −greedy method
9.                  Find next state
10.                 Calculate reward
11.                 Update q-values
12.                 k = k + 1
13.         End for
14.         n = n + 1
15.  End for
16.  Return q-table
```

Fig. 1: Conventional Q-learning Algorithm

In this case, the possible actions are the battery states of energy from the minimum value of 10kWh to a maximum value of 100kWh (as shown in Table 2) in steps of 10kWh. At every time-step, the algorithm's agent observes the current state (as described in section IV part A) and decides the next battery energy level. This decision is used to calculate the amount of power to be supplied to the battery to meet the agent's recommendation. Thereafter, given the current value of PV power, the power balance equation (1) is used to determine the amount of power to be purchased from or supplied to the grid. The cost of battery degradation and grid power purchase are then calculated. The system state is then updated according to equation (5). Using the computed cost, the reward is calculated as given in equation (6) and the Q-value of the action taken is then updated as given in (11) or (12) depending on whether the current state is terminating state.

C. *Algorithm 2: Proposed Modified Q-learning algorithm*

In this paper, a hash table approach is used. In this method, the Q-table is initialized as an empty hash table into which states are added as keys, with dictionaries of the allowable actions and their initial Q-values, as values. The Q-table therefore becomes a nested dictionary, with states indices as keys. States are added as they occur during learning, hence avoiding states that are never visited by the agent.

To solve the scheduling problem already defined as an MDP, an empty hash table (or a dictionary data structure in Python programming) is first created. At the beginning of the learning, the algorithm reads the time and load forecast, PV generation forecast and grid tariff values for that time and returns an incomplete state vector. The initial battery energy is appended to the vector to get a complete state vector. The elements of the state vector are then joined to form a single state identity and added to the empty dictionary. If this state had not been in the table, all the possible actions associated with it are computed as described in section IV part B with all Q-values initialized with zeros. An action is selected using the ϵ-greedy policy and implemented. The next state is then found, followed by the reward computation and the Q-value for the state-action pair is updated using equation (11) or (12) depending on whether it is a terminating state. As such, only states that the agent has visited are added to the table with their possible actions. The learning process is asynchronous because the agent does not sweep through the entire state space in each episode. The algorithm is shown in Fig. 2.

```
1.  Initialize learning parameters ($\alpha$ and $\gamma$)
2.  Initialize epsilon ($\epsilon = 1$)
3.  Create an empty q-table
4.  For n = 0 to maximum iteration:
5.      k = 0
6.      Get the initial state vector
7.      For k = 0 to T-1:
8.          If the state is not in q-table:
9.              Get possible actions
10.             Initialize q-values = 0
11.             Add state-action pair to q-table
12.         End if
13.         Take action by $\epsilon -$greedy method
14.         Find next state
15.         Calculate reward
16.         Update q-values
17.         k = k + 1
18.     End for
19.     Update $\epsilon$
20.     n = n + 1
21. End for
22. Return q-table
```

Fig. 2: Proposed Asynchronous Q-learning Algorithm

D. *Algorithm 3: Policy Retrieval*

At the end of the learning progression, the policy retrieval for both algorithms is done using the algorithm in Fig. 3.

```
1.  Read the initial state
2.  k = 0
3.  For k = 0 to T-1:
4.      Get optimal action
5.      k = k + 1
6.      Read the next state
7.  End for
8.  Return list of states and actions
```

Fig. 3: Policy retrieval algorithm

VI. Results and Discussions

A. *Input Data*

An illustrative 80kW, grid-tied, PV-powered, EV fast CS with a maximum PV output of 70kW has been considered as shown in Fig. 4 [23]. Contingent on the instantaneous charging station's load, when the PV generation is insufficient to supply the charging load, it is complemented by the power bought from the grid and when the PV output is in surplus, the additional energy is supplied to the grid. Also, as can be seen in Fig. 4, the grid tariff is dynamic.

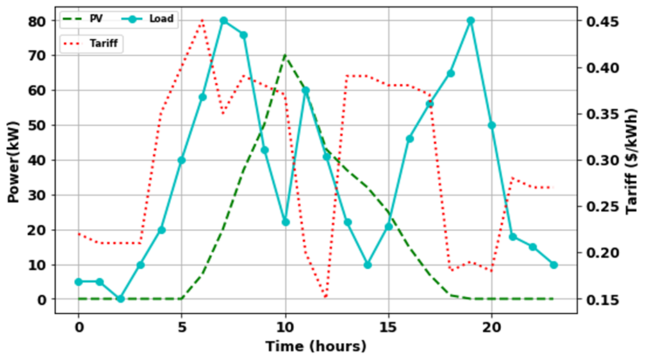


Fig. 4: Input data: load, PV and grid tariff profiles

It is assumed that this grid tariff profile is communicated to the charging station's operators in advance for this scheduling task to be performed.

The learning parameters for both algorithms were selected as shown in Table II below. The original exploration rate has been chosen to be 1.0 to ensure that the search space is maximumly explored. The learning rate was chosen by trial and error and a value of 0.001 was chosen for α. Higher values of α like the normally selected value of 0.01 produced a poor convergence for both algorithms. A discount factor of 1.0 was selected to give the future rewards equal value as the current rewards.

TABLE II. LEARNING HYPERPARAMETERS

Hyperparameter	***Chosen value***
Epsilon (ϵ)	1.0
Learning rate (α)	0.001
Discount factor (γ)	1.0

The simulation parameters are given in Table III below.

TABLE III. SIMULATION PARAMETERS

Parameter	***Symbol***	***Values***
Minimum/Maximum grid power	P_g^{min}/P_g^{max}	-50/90kW
Timestep	Δt	1hour
Battery capital cost	C_{bt}	\$400
Battery capacity	E_b	100kWh
Initial battery energy	SoE_{in}	50kWh
Minimum battery energy	E_b^{min}	10kWh

B. Convergence Analysis: The Cost and Reward Learning Curves

Fig. 5 and Fig. 6 show the episodic cost profile for the proposed asynchronous Q-learning and the conventional Q-learning algorithms respectively for 15000 learning episodes. The moving average values are calculated every 50 episodes. It can be seen in both Fig. 5 and Fig. 6 that the raw and average costs for both algorithms start at high values in earlier episodes between 0 and 2000 and reduce to lower optimized values in the later episodes beyond 2000. This is because, in both algorithms, the initial exploration rate has been set to 1.0 (as shown in table 1), i.e., all the possible actions have equal probability of being selected, thus learning agent begins by exploring the action space because each action's probability of being selected is 1.0. When the policy was retrieved, the optimal episode for the proposed Q-learning method returned 14% lower global cost than the conventional Q-learning method as shown in Table IV.

TABLE IV. GLOBAL COSTS

Algorithm	***Global cost***
Conventional Q-learning	\$209
Proposed method	\$179

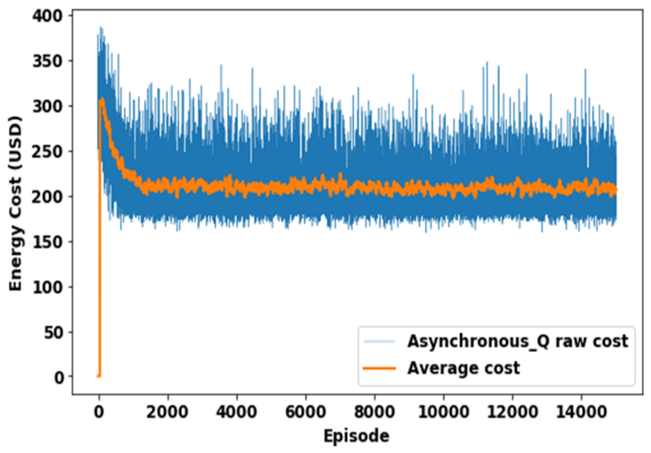


Fig. 5: Episodic cost variations for the proposed asynchronous Q-learning

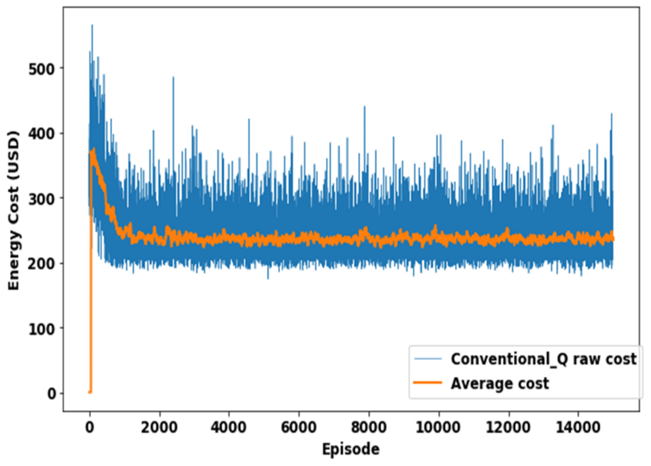


Fig. 6: Episodic cost variations for conventional Q-learning

Fig. 7 shows the comparative plot of the moving average episodic cost profiles for the two algorithms. As can be seen in the figure, the proposed asynchronous method starts at a lower initial average cost of about \$300 and finishes at a value of about \$200, compared to the conventional Q-learning that starts from a higher value of about \$360 and finishes at an average of about \$240. This is because the proposed method has its action space limited to just those set of values of grid power and battery power that meets the load demand. Therefore, the power balance constraint given in (1) is imposed before the actions are taken. As a result, its learning is constrained and well guided, thus preventing it from losing track. That is unlike the conventional Q-learning approach that uses an approach similar to the battery scheduling method in which the power balance is ensured only after the agent takes the action, leading to the collection of bad experiences that causes it to lose track. Although both algorithms converge, the proposed method converges at a lower global cost.

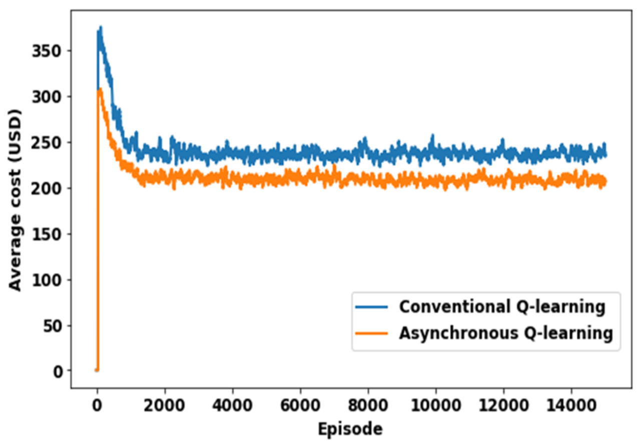


Fig. 7: Episodic average cost variations for both algorithms on one axis

Fig. 8 shows the learning curves for both the conventional Q-learning method and the proposed asynchronous method. It can be seen that the proposed method learns faster and achieves higher average episodic total rewards than the conventional Q-learning method. This difference is also as a result of constraining the action space for each state in the proposed method that causes it to only meet experiences that are within the power balance equation (1).

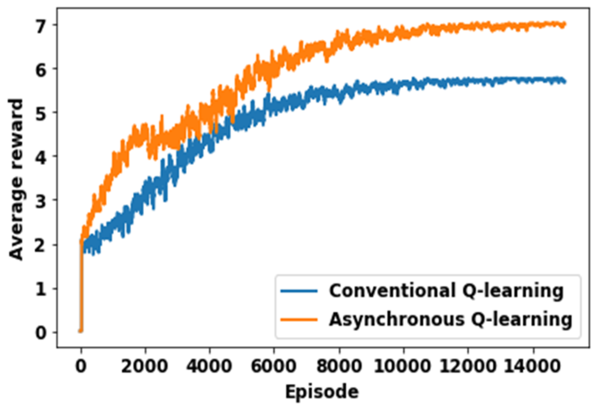


Fig. 8: Learning curves for both algorithms on one axis

C. *Optimized Power Schedule Obtained from the Proposed Asynchronous Q-learning Method*

At the end of the learning period, algorithm 3 described in section V part D is used to extract the optimal episode using the greedy policy. The greedy policy returns the action with the highest Q-value in every state throughout the optimization horizon. In this subsection, the power schedule derived from the optimal episode obtained from the proposed method is discussed. The cost minimization actions recommended by the asynchronous algorithm at various times within the episode are identified and explained.

Fig. 9 shows the power schedule for both the battery and the utility grid plotted alongside the input PV, grid tariff and the charging station's load profiles. Fig. 10 is a plot of the optimized battery energy profile for the same optimization period. To minimize the cost of supplying the charging station's load, the algorithm is designed to use the following strategies:

1. maximize the self-consumption of the PV generated power,
2. utilize less of grid power when the tariff is high and more of it when the tariff is low.

The first strategy is implemented by charging the battery when the solar PV generation is high and using the stored energy when there is low PV and the grid tariff is high. The battery and grid power are positive when supplying power to meet the charging station's load and negative when absorbing power from the station's common DC bus. It should also be noted that the battery energy changes according to a charge or discharge action in a previous hour, thus, there is a time difference of 1 hour between the battery power changes and the corresponding battery energy changes.

The second strategy is executed by scheduling high grid power intake when the tariff is low and reducing the purchase of grid power when the grid tariff is high.

It can be seen from Fig. 9 that the grid power intake is high between 10th and 12th hours and between 17th and 21st hours when the grid tariff is cheap. However, when there is low PV generation and the battery is energy is low between the 5th and the 7th hour (see in Fig. 10), the algorithm schedules the purchase of the grid power to meet the station's instantaneous load, even though the grid tariff is high at these times.

Fig. 10 displays a general discharge from the beginning up to the 6th hour when a charge decision is occasioned by the load dropping to zero at a time when the grid tariff is low. From 7th hour, the battery power goes to negative when the PV is high up to the 12th hour and the battery energy starts to rise as shown in Fig. 10. Before the 10th hour, the battery charging is slow as most of the generated PV is being used to supply the load due to the high grid tariff. This is followed by a sharp rise in the charging rate from 10th to 12th hour. At other times, the battery may only be charged if the grid tariff is very low to take advantage of the cheap grid power. The stored energy of the battery is used when the grid tariff rises, and the PV power drops. This can be observed from the 16th hour when the battery energy drops sharply. Some instabilities may be noticed due to the dynamic nature of the tariff as seen between the 18th and 20th hour.

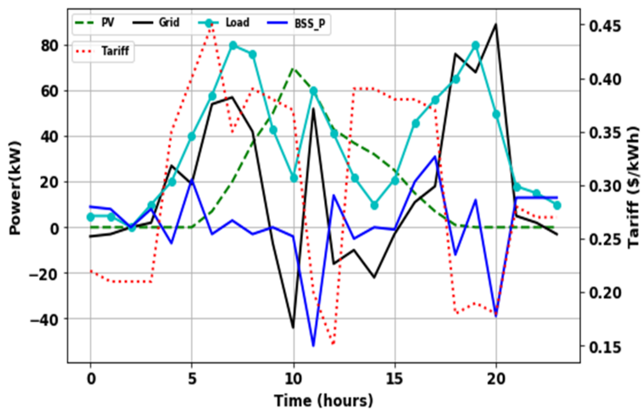


Fig. 9: Optimized power schedule alongside the input PV, load and grid tariff profiles

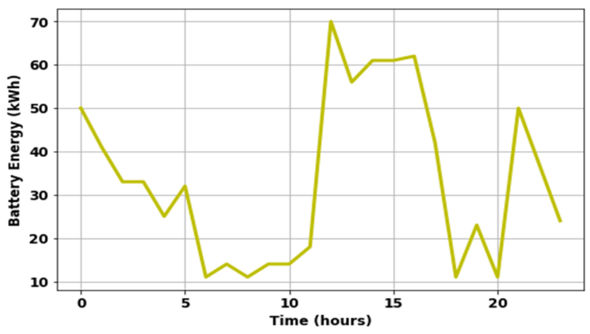


Fig. 10: Optimized battery energy schedule

VII. Conclusion

In this paper, a modified Q-learning algorithm has been used for power management in a grid-tied PV/battery electric vehicles fast-charging station. The costs considered comprise the cost of power purchased from the grid and the cost of battery degradation due to temperature, depth of discharge and average state of charge. The algorithm is compared with the conventional Q-learning method. The simulation results show that the agent of proposed Q-learning method learns to achieve higher total reward and returns about 14% lower global cost than the conventional Q-learning algorithm. Future research on this problem will focus on the use of deep reinforcement learning techniques to overcome the curse of dimensionality and lack of policy generalization associated with Q-table based learning methods.

Acknowledgment

Our sincere gratitude also goes to the Mandela Rhodes Foundation for funding the first author.

References

[1] World Energy Council, “World Energy Resources 2016,” World Energy Counc. 2016, pp. 6–46.

[2] T. N. Gucin, K. Ince, and F. Karaosmanoglu, “Design and power management of a grid-connected Dc charging station for electric vehicles using solar and wind power,” 2015 3rd Int. Istanbul Smart Grid Congr. Fair, ICSG 2015, May 2015.

[3] S. K. Hoke Anderson, Brissette Alexander and P. Annabelle, “Accounting for Lithium-Ion Battery Degradation in Electric Vehicle Charging Optimization,” IEEE J. Emerg. Sel. Top. Power Electron., vol. 2, no. 3, pp. 691–700, 2014.

[4] C. J. Boo and H. C. Kim, “Photovoltaic based electric vehicle charging optimization,” Int. J. Softw. Eng. its Appl., vol. 9, no. 12, pp. 285–292, 2015.

[5] Y. Riffonneau, S. Bacha, F. Barruel, and S. Ploix, “Optimal power flow management for grid-connected PV systems with batteries,” IEEE Trans. Sustain. Energy, vol. 2, no. 3, pp. 309–320, 2011.

[6] N. A. Luu and Q. T. Tran, “Optimal energy management for grid-connected microgrid by using dynamic programming method,” IEEE Power Energy Soc. Gen. Meet., vol. 2015-Septe, no. 1, pp. 1–5, 2015.

[7] D. Zhang, “Optimal Design and Planning of Energy Microgrids.” Ph.D. Thesis, Department of Chemical Engineering, University College London, 2013.

[8] U. B. Tayab, F. Yang, M. El-Hendawi, and J. Lu, “Energy Management System for a Grid-Connected Microgrid with Photovoltaic and Battery Energy Storage System,” 2018 Aust. New Zeal. Control Conf., pp. 141–144, 2019.

[9] M. El-Hendawi, H. A. Gabbar, G. El-Saady, and E. N. A. Ibrahim, “Enhanced MG with optimum operational cost of pumping water distribution systems,” 2017 5th IEEE Int. Conf. Smart Energy Grid Eng. SEGE 2017, pp. 137–142, 2017.

[10] M. O. Badawy and Y. Sozer, “Power Flow Management of a Grid Tied PV-Battery System for Electric Vehicles Charging,” IEEE Proc. 2017 Winter Simul., vol. 53, no. 2, pp. 1347–1357, 2017.

[11] R. Bellman, “The Theory of Dynamic Programming.” The Rand Corporation, California, 1954.

[12] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. The MIT Press, 2018.

[13] S. Sanner, R. Goetschalckx, K. Driessens, and G. Shani, “Bayesian Real-time Dynamic Programming,” 1994.

[14] E. A. Jasmin, “Reinforcement Learning Approaches to Power System Scheduling.” Ph.D. Thesis, School of Engineering, Cochin University of Technology, 2008.

[15] C. J. C. H. Watkins, “Learning from Delayed Rewards.” University of Cambridge, 1989.

[16] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives,” IFAC, vol. 50, no. 1, pp. 6918–6927, 2017.

[17] S. Bai, Y. Du, and S. Lukic, “Optimum design of an EV/PHEV charging station with DC bus and storage system,” 2010 IEEE Energy Convers. Congr. Expo. ECCE 2010 - Proc., pp. 1178–1184, 2010.

[18] A. O. Erick; K. A. Folly, “Reinforcement learning approaches to power management in grid-tied microgrids : A review,” Clemson Univ. Power Syst. Conf., 2020.

[19] B. V. Mbuwir, F. Spiessens, and G. Deconinck, “Self-learning agent for battery energy management in a residential microgrid,” Proc. - 2018 IEEE PES Innov. Smart Grid Technol. Conf. Eur. ISGT-Europe 2018, pp. 1–6, 2018.

[20] E. Kuznetsova, Y. F. Li, C. Ruiz, E. Zio, G. Ault, and K. Bell, “Reinforcement learning for microgrid energy management,” Energy, vol. 59, pp. 133–146, 2013.

[21] U. Abronzini, C. Attaianese, M. D'Arpino, M. Di Monaco, A. Genovese, G. Pede, G. Tomasso, “Optimal energy control for smart charging infrastructures with ESS and REG,” 2016 Int. Conf. Electr. Syst. Aircraft, Railw. Sh. Propuls. Road Veh. Int. Transp. Electrif. Conf. *ESARS-ITEC 2016*, pp. 1–6, 2016.

[22] U. Abronzini, C. Attaianese, M. D’Arpino, M. D. Monaco, and G. Tomasso, “Power Converters for PV Systems with Energy Storage: Optimal Power Flow Control for EV’s Charging Infrastructures,” PCIM Eur. 2016; Int. Exhib. Conf. Power Electron. Intell. Motion, Renew. Energy Manag., no. May, pp. 1–7, 2016.

[23] A. O. Erick and K. A. Folly, "Energy trading in grid-connected PV-battery electric vehicle charging station," SAUPEC/RobMech/PRASA Conference, Cape Town, South Africa, 2020, pp. 1-6.

[24] C. Zhou, K. Qian, M. Allan, and W. Zhou, “Modeling of the cost of EV battery wear due to V2G application in power systems,” IEEE Trans. Energy Convers. vol. 26, no. 4, pp. 1041–1050, 2011.

[25] K. J. Laidler, “The development of the Arrhenius equation,” J. Chem. Educ., vol. 61, no. 6, pp. 494–498, 1984.

[26] R. S. Sutton, “On the significance of Markov decision processes.” Department of Computer Science, University of Massachusetts, Amherst, MA USA, 2005.

[27] R. Bellman, “A Markovian decision process,” J. Math. Mech., vol. 6, no. 5, pp. 679--684, 1957.

[28] Y. Wan, M. Zaheer, M. White, and R. S. Sutton, “Model-based Reinforcement Learning with Non-linear Expectation Models and Stochastic Environments,” FAIM Work. Predict. Gener. Model. Reinf. Learn. Stock. Sweden, 2018.

[29] D. Dewey, “Reinforcement learning and the reward engineering principle rewards in an uncertain world,” AAAI Spring Symp. Ser., pp. 1–8, 2014.

[30] C. J. C. H. Watkins and P. Dayan, “Technical Note: Q-Learning,” Mach. Learn., vol. 8, no. 3, pp. 279–292, 1992.

[31] M. A. L. Thathachar and P. S. Sastry, Networks of Learning Automata : Techniques for Online Stochastic Optimization. New York: Springer Science+Business Media, 2004.

[32] I. A. T. Parambath, E. A. Jasmin, F. R. Pazheri, and E. A. Al-Ammar, “Reinforcement learning solution to economic dispatch using pursuit algorithm,” 2011 IEEE GCC Conf. Exhib. GCC 2011, pp. 263–266, 2011.