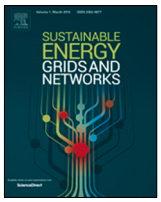
Online EV charging controlled by reinforcement learning with experience replay

Sustainable Energy, Grids and Networks 36 (2023) 101162

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

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



Online EV charging controlled by reinforcement learning with experience replay

Andrey Poddubnyy [a,*], Phuong Nguyen [a], Han Slootweg [a,b]

[a] Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[b] Enexis Netbeheer, 's-Hertogenbosch, The Netherlands

ARTICLE INFO

Article history:
Received 8 February 2023
Received in revised form 18 August 2023
Accepted 22 August 2023
Available online 5 September 2023

Keywords:
Congestion management
Distribution network
Electric vehicles
Reinforcement learning

ABSTRACT

The extensive penetration of distributed energy resources (DERs), particularly electric vehicles (EVs), creates a huge challenge for the distribution grids due to the limited capacity. An approach for smart charging might alleviate this issue, but most of the optimization algorithms has been developed so far under an assumption of knowing the future, or combining it with complicated forecasting models. In this paper we propose to use reinforcement learning (RL) with replaying past experience to optimally operate an EV charger. We also introduce explorative rewards for better adjusting to environment changes. The reinforcement learning agent controls the charger's power of consumption to optimize expenses and prevent lines and transformers from being overloaded. The simulations were carried out in the IEEE 13 bus test feeder with the load profile data coming from the residential area. To simulate the real availability of data, an agent is trained with only the transformer current and the local charger's state, like state of the charge (SOC) and timestamp. Several algorithms, namely Q-learning, SARSA, Dyna-Q and Dyna-Q+ are tested to select the best one to utilize in the stochastic environment and low frequency of data streaming.

1. Introduction

A rapid pace of using electrical vehicles (EVs) leads to big changes in both the scale and the shape of electricity consumption. Typical residential load shape already has its daily (as well as seasonal) peaks. Utilization of electric vehicles along with other distributed energy resources (e.g. heat pumps) can make the situation even worse. Thus, system operators will be forced to postpone the EV chargers installation approval till the grid is reinforced adequately.

The price-based smart charging has been studied extensively, as discussed in the literature section, and mainly with help of optimization methods [1–3]. Such models rarely include the EVs number or consumption pattern change into account. Most of optimization algorithms has been developed under the assumption of having perfect future knowledge which is not quite applicable for the high uncertainty from a massive number of EVs.

Reinforcement learning is considered as a promising approach to handle the uncertainty, especially with on-line applications [4, 5], since it allows the agent (EV charger) to learn from experience and make decisions from his current perspective maximizing the expected long-term return. Such a statement of the problem requires only to correctly describe the environment to the agent and select relevant algorithms to learn. It does not require knowing a physical model of a grid or EV batteries, nor the statistical properties of the environment, but only the data from these objects.

In this paper the reinforcement learning algorithms, controlling an EV charging station's consumption power in real time are utilized. The objective here will be the optimization of the charging expenses and prevention of the possible overload. This will allow to loosen the capacity limits of the power grid, preventing unnecessary investments and accelerating the deployment of EVs' infrastructure.

The data for the model training will include samples from the power flow problem-based environment. Residential daily load profiles, along with their placement in distribution grid, equipment capacities and EV chargers locations are acquired from an industrial partner.

This paper is an improved version of the conference publication [6]. It introduces an extended version of Dyna-Q algorithm, i.e. Dyna-Q+, to tackle the problem of concept drift in reinforcement learning. It was tested on the load and price data and was proven to better adjust to conceptually drifting environment. The seasonality of the load dataset was removed with help of time series decomposition to make the rewarding of the algorithm consistent with its actions. In addition, the environment was

* Corresponding author.
E-mail addresses: a.poddubnyy@tue.nl (A. Poddubnyy), p.nguyen.hong@tue.nl (P. Nguyen), han.slootweg@enexis.nl (H. Slootweg).

https://doi.org/10.1016/j.segan.2023.101162

Nomenclature

ϵ	probability to make a random action in the ϵ-greedy policy
γ	discounting factor
κ	small number, scaling factor for τ
λ	electricity price
λ_{med}	median price before arrival or after departure
ω_{cap}	reward factor for the transformer load (capacity) objective term
ω_{cost}	reward factor for the electricity cost objective term
ω_{SOC}	reward factor for the state-of-charge objective term
π	policy of an agent
π_{opt}	optimal policy of an agent
τ	accumulated reward, increased by 1 for every step it's state is not visited
A	set of actions
P	probabilistic state transition model
$P_{load,disc}$	discretized P_{load}
P_{load}	consumption of the whole grid
$P_{thresh,disc}$	discretized P_{thresh}
P_{thresh}	transformer load threshold, used to divide rewards into positive and negative
Q_π	action value function, given the policy π
R	reward function
r_p	reward during the planning step of the Dyna-Q+ algorithm
s'	state after taking an action
S	set of states
s	state before taking an action
SOC	state-of-charge of the EV battery

adjusted by adding the departure and arrival times and their distribution to make the model more realistic. Also the general agent's behavior was improved in a way, that the clearly disadvantageous and infeasible state–action pairs, such as the action to charge while having a full battery, were excluded from the search domain. It all enhanced algorithms' learning speed and made the difference between them more expressed and visible.

The contribution of this paper is in the following:

- The planning algorithm (i.e. Dyna-Q), which learns the transition model from the interaction with the environment, was introduced for replaying past experience and, thus, making more use of the interaction with the environment.
- This algorithm was extended to a more explorative version, which proved itself better under changing environment, such as prices drift. The price change remained almost unnoticed by Q-learning and SARSA but was detected by the planning algorithms.
- The planning algorithms were benchmarked against the other value-learning ones (i.e. Q-learning, SARSA) to validate their performance.
- Smart-charging algorithms used are aware of not only electricity prices and time but also the state of the network, in particular the load of the transformer.

2. Background

There have been many studies in recent years, dedicated to the large-scale deployment of charging stations. Some of them were mainly focused on the best allocation of EV stations in the grid [7,8], others on the best battery load shape to meet the sustainable energy supply or off-peak hours [9,10]. In this paper the EV charging problem is discussed from both perspectives, considering the personal needs of an EV owner, as well as the requirements of the grid.

2.1. Traditional approaches

2.1.1. Optimization

The most widespread way to optimize EV charge is to utilize conventional optimization algorithms. Generally, optimization under the power flow constraint is considered non-convex but can be relaxed using various techniques, e.g. second-order conic programming [1]. EV charging is often considered in coupling with another problem like photovoltaic (PV) generation [2]. Since the optimization task in a highly dimensional space is extremely time-consuming, there were proposed some algorithms to tackle this issue [11]. Optimization models can also contain modeling of EVs movement statistical properties, as with Monte Carlo simulations [3]. Although traditional optimization models present a solid approach to handle the issue in terms of the accuracy of the solution, they have some limitations:

- They require the detailed and precise modeling of the environment, including the EV battery model, and power flows, which can pose a challenge, especially in complex distribution networks.
- They demand a forecast of the future data, used in the model, to be able to observe the whole time scope, for which the actions are taken. The performance of the forecast model severely affects the overall performance of EV charge control
- In optimization-based models there should be a consideration of non-stationarity of the variables (prices, loads), either in the forecasting or optimization parts, depending on the architecture. This can be tricky since both types of models typically rely on the predetermined or stochastic data, sampled from the stationary distribution.
- Conventional optimization models are poorly scalable to large vehicle fleets, since the growth of computational complexity with the increase of dimensionality.

2.1.2. Heuristic algorithms

A number of heuristic and meta-heuristic algorithms were also introduced for the EV charging problem [12]. The main motivation of such methods is in their ability to achieve the solution faster, than with the regular optimization techniques, at the cost of its sub-optimality. Since heuristic algorithms also have to simulate the EV charging in the grid environment, It inherits the same drawbacks from the optimization models, except it has better scalability. Apart from that it introduces new issues, connected to the sub-optimality of the solution.

2.2. RL for power grid balancing

When compared to the traditional optimization methods, Reinforcement Learning has several potential advantages:

- RL is a model-free approach for the optimization of the agent's reward function. Thus, it does not require any knowledge of the environment but only a set of interactions with it.

- RL does not require a forecasting model either, since it optimizes expected returns at the end of the episode from the current state's perspective.
- As it is shown in this study, RL agent's online learning allows policy adjustment, starting immediately after the change in the environment.
- The scalability of RL algorithms varies depending on the architecture, but for the independently learning agents without any interaction and coordination between them, the computational complexity grows linearly.

Reinforcement learning was applied to different aspects of consumption optimization problems with various goals and environment settings.

The problem of voltage security was discussed in [13]. In this paper DDPG (Deep Deterministic Policy Gradient) algorithm was used at the same time with a plain optimization problem. The resulting agent mainly transfers the consumption from peak hours to off-peak hours, as a consequence of lower prices.

Research, conducted in [14] sets the problem in a similar way, as it is proposed in this work, but the solution serves the purpose of the PV panels overvoltage mitigation by means of reinforcement learning. The algorithm minimizes the curtailment of PV power with regard to the voltage level and treats the state of the network as known, running the power flow problem.

In RL-based energy consumption optimization, [15] can be considered as one of the milestones in terms of architecture and the results achieved. The authors analyzed two different deep neural networks (DNN) based architectures of RL: Deep Q-learning (DQL) and Deep Policy Gradient (DPG). Both methods approximate a function with the help of DNN. In the first one, the function is the action-value, in the second it is a directly parametrized policy. Both algorithms managed to minimize the cost, as well as reduce the peak load.

2.3. RL for EV charging

The application of RL for EV charging was considered already in some studies. They vary significantly in terms of problem formulation. The studies mainly focus on: minimizing charging costs [16], transformer loadings [17] and waiting times [18], maximizing welfare [19], renewables utilization [20], profits of individual EV owner [21], charging station or distribution system [22]. In many cases, the objectives are combined into a single multi-objective reward function to chase several goals simultaneously. Constraints, whether it is the grid capacity or state of the charge, are imposed indirectly via the reward function as penalties for their violation.

In [5] the problem is to optimize EV charge bids taking the network constraints into account. The constraints are implemented as a cap power consumption for the aggregated area. The Q-learning algorithm is used for optimization. In [23] similar problem was also considered but with the help of SARSA (State–action–reward–state–action) algorithm

Another EV charger optimization study was described in [24]. In this case, the EV charger is working at the same time with the building and has two points of supply: PV panel and power grid. The objectives in this model are PV usage and state of charge at the moment of departure maximization. The authors use the algorithms of DDQN (Double Deep Q-networks), DDPG and P-DQN (Parametrized Deep Q-networks). Contrary to the algorithms, used in this paper, the aforementioned algorithms utilize deep neural networks, which utilize an artificial neural network as a policy. The main advantage of these algorithms is that they can solve highly-dimensional tasks.

An extensive review on the EV charging RL algorithms was done in [4]. From among the main conclusions is the fact, that Q-learning algorithm, one of the basic ones in reinforcement learning, was found one of the most effective, as well as its deep extension DQN. The other conclusion is that multi agent reinforcement learning solutions (MARL) are more probable to converge to the optimal behavior of agent, yet they are more difficult to do it and more computationally expensive due to the time-consuming power flows simulations. The last issue was addressed in various studies (e.g. in [25]).

2.4. RL under non-stationary conditions

RL agent can face certain issues, when the environment, where it is deployed, is a subject of a change. That implies, that developed policies might no longer be relevant or can provide completely wrong solution. Even though RL might be adaptive to some changes, without modifications it prefers following the existing, previously learned policies, since locally they provide the best solution. For the small changes near the optimal it can be enough to have an ϵ-greedy policy for some exploration, but it may not help in case the optimal solution lies in a distance from the currently preferred one.

RL under non-stationary conditions has been the topic of interest for a long time. In principle, there are two different sources of non-stationarity in RL problems:

- Environment-induced non-stationarity, caused by the inherent change of the environment
- Agent-induced non-stationarity, which is mainly an issue for multi-agent models [26] and Deep RL [27].

Non-stationarity in the environment's distributions can also be divided into predictable and random. Predictable changes can be taken into account by explicit modeling of this change, using, for example, time-series analysis techniques [26], or context-depending learning [28]. These methods specifically focus on the changes, that are assumed to be known in advance or predicted.

It is generally difficult to track the changes in the distribution network since it is the most branched and stochastic part of the power grid. Thus, particularly in this part of the grid, it is desirable to avoid the perfect knowledge requirements. With this said, it is supposed in this study, that we want to make non-stationarity adaptation as automatic as possible, relying only on the experience, acquired by the RL agent. This is also supported by one of the main ideas of RL, which is a model-free learning, that does not require the physical modeling of the environment.

In this paper we use the extended version of the Q-learning algorithm by adding the experience replay for general better performance (Dyna-Q). These algorithms was already applied to EV charge problem in [29] for an additional learning in the simulated experience, which gives better overall learning results.

In this paper we also utilize the further extended Dyna-Q+ algorithm, which is an explorative version of Dyna-Q, that better handles conceptually drifting data. In Dyna-Q+ exploration has a different nature, than in the ϵ-greedy policy. It rewards agent for once in a while visiting the long abandoned states. The reward is assigned to all the states, no matter how remote they are from the current optimal path.

Most of the algorithms, used in this paper, are described in [30], which is a comprehensive handbook for reinforcement learning.

3. Methodology

Here we discuss the theoretical background of reinforcement learning environment representation as well as introduce agent behavioral algorithms to be tested. In the paper two experience replay algorithms (i.e. Dyna-Q and Dyna-Q+) were benchmarked with two other tabular algorithms, which only make a single update per data point.

3.1. Markov decision process

Reinforcement learning model can be applied to the problem, formulated in terms of Markov Decision Process (MDP). MDP is usually represented by the tuple: (S, A, P, R, γ). Here $s \in S$ is a set of states, $a \in A$ is a set of actions, P is a transition model, returning the probability of transition to the state s' given the action a taken in the state s. To evaluate the actions' immediate reward from taking an action a, that leads to a state s' from the state s the reward function $R = f(s, a, s')$ must be introduced. The discounting factor γ is used for the long-term reward calculation within the action-value function Q_π.

$Q_\pi(s, a)$ is a function, that estimates the expected long-term return from taking an action a in the state s in accordance with the policy π. Policy $\pi(s) = a$ is a function, that returns the action to be taken in s. Thus, the ultimate goal of the reinforcement learning algorithm running is to find the optimal policy π, which can be represented as the action, leading to the highest long-term return:

$$\pi_{opt} = argmax_a Q_\pi(s, a) \tag{1}$$

In order to find the optimal policy, the action-value function has to be approximated also. There are various methods to do this, using parametric and non-parametric functions. These methods form a variety of RL algorithms, that have their advantages and bottlenecks and fit different types of problems. For example in [15,24] the artificial neural network was used, while in [5] more conventional SARSA algorithm. In this paper we consider algorithms, which utilize a table for state–action couples, where cells are updated after visiting the corresponding state–actions in accordance to the rules, varying for every algorithms. These rules and algorithms are explained in the following sections.

3.2. Temporal difference learning

Intuitively, it can be thought about finding the action-value function $Q_\pi(s, a)$ and corresponding optimal policy π_{opt} as a set of Monte Carlo (MC) experiments: starting at a particular state and making transitions till the end of an episode for an episodic task or a time horizon for a continuous task in accordance with the greedy policy. In this case the averaged returns for each state–action pair will be an estimation for $Q_\pi(s, a)$.

Because of end episode update, with such an approach online learning is not possible. Family of methods, allowing online learning by utilizing *bootstrapping* (i.e making a guess from another guess), is called Temporal Difference (TD) learning. The difference between MC and TD algorithms is that in TD the value function estimation is made each step via the update rule, which includes the estimated value of the following state. Both MC and TD algorithms utilize sample updates, contrary to the model-based approaches, such as Dynamic Programming, where it is necessary to know the model of the environment. This difference is illustrated on Fig. 1

The two common algorithms from the family are SARSA and Q-learning.

3.2.1. SARSA algorithm

SARSA is the on-policy algorithm which utilizes the information of the current state, current action, reward from taking that action, next state and the action to be taken in the next state in accordance with the policy $(S_t, A_t, R, S_{t+1}, A_{t+1})$ for each update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \cdot Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2}$$

Each episode ends with the *terminal* state, at which the environment resets to its initial condition. Using the update rule, the whole algorithm can be formulated as a nested loop over the episodes and over the states:

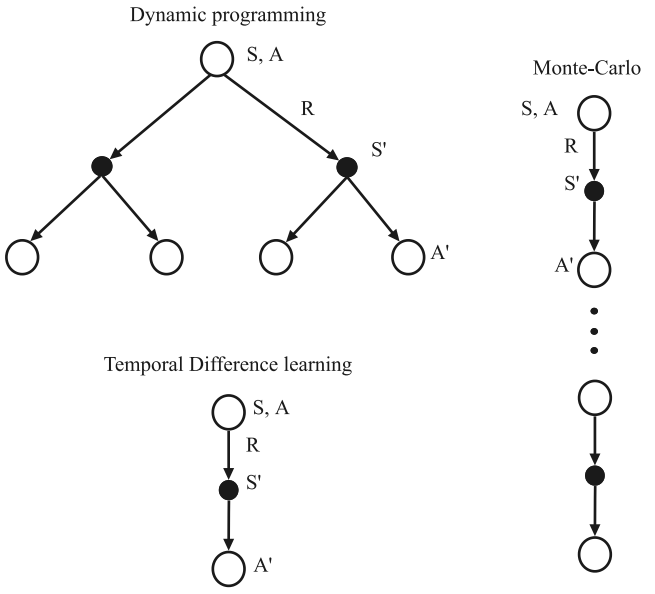


Fig. 1. Information needed for different types of update rules.

Algorithm 1 SARSA algorithm

Initialize $Q(S, A)$ arbitrarily for all $s \in S^+, a \in A(s)$, except terminal states, where Q = 0
for each episode **do**
  Initialize S_t;
  Choose A_t from S_t based on the policy and current Q estimation;
  while $S_t \neq terminal$ **do**
    Take A_t, observe R, S_{t+1};
    Choose A_{t+1} from S_{t+1} based on the policy and current Q estimation;
    Make an update by the rule (2);
    $S_t \leftarrow S_{t+1}, A_t \leftarrow A_{t+1}$;
  end while
end for

If it is desired for the agent to not get stack in sub-optimal solution and explore the environment more, it would be necessary to utilize the exploration type of policy, e.g. $\epsilon-greedy$ policy. This policy takes a greedy action with the probability $1 - \epsilon$ and takes an exploration step with the probability ϵ.

3.2.2. Q-learning algorithm

Q-learning is another algorithm, utilizing bootstrapping, but the main difference with SARSA is that it allows to learn the target policy and explore by behavior policy, which is different from the target. Typically such quality is used to simplify exploration. Its update rule looks like this:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \cdot \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3}$$

The formalized algorithm looks similar to the one for SARSA, with the main difference in the update rule, which is now defined by (3):

3.3. Dyna-Q algorithm

All the aforementioned models are related to the *model-free* class of RL algorithms. That means, that learning (value function

Algorithm 2 Q-learning algorithm

Initialize $Q(S, A)$ arbitrarily for all $s \in S^+, a \in A(s)$, except terminal states, where $Q = 0$
for each episode **do**
  Initialize S_t;
  while $S_t \neq$ *terminal* **do**
    Take A_t, observe R, S_{t+1};
    Choose A_t from S_t based on the behavior policy and current Q estimation;
    Make an update by the rule (3);
    $S_t \leftarrow S_{t+1}$;
  end while
end for

update) happens only through the experiments and real actions in the environment. But in case of the real data learning the data availability is quite limited, i.e. the amount of experiments needed to get a satisfactory results may not always be produced. In this case it can be addressed by the *model-based* RL algorithms. This class of algorithms reconstructs the state transition depending on the current state and the action taken. The updates of the value function based on the modeled, rather than real, environment are called *planning* steps.

Modeling of the environment can generally be done in two ways: *sample model* and *distribution model*. Distribution models are assumed in dynamic programming, which is the basis for the reinforcement learning. But in case of the distribution based model it would be necessary to calculate the probabilities of all the possible outcomes, which is unrealistic in case of stochastic power systems with a plenty of elements. Sample models produce a single result according to the distribution behind the environment, but the distribution itself remains unknown. Thus, distribution models are better in terms of results, but they are sometimes impractical and the sample based ones are much easier to acquire.

For the computationally expensive environments, like power systems, model-based algorithms give one more advantage. The planning updates do not require re-computation of the new states, but rather sample them.

One of the most widely known model-based algorithms is Dyna-Q [30]. In fact, it utilizes the Q-learning update rule, but also incorporates the modeling part:

Algorithm 3 Dyna-Q algorithm

Initialize $Q(S, A)$, $Model(S, A)$ for all $s \in S^+, a \in A(s)$,
for each episode **do**
  Initialize $S_t \leftarrow$ current state;
  Initialize $A_t \leftarrow \epsilon$-greedy policy(S_t, Q);
  Take action A_t observe reward R and state S_{t+1};
  Make an update from the Q-learning update rule (3);
  $Model(S_t, A_t) \leftarrow R, S_{t+1}$ assuming the deterministic environment
  for each planning step **do**
    $S_t \leftarrow$ random previous state;
    $A_t \leftarrow$ random action, taken in the previous state;
    $R_t, S_{t+1} \leftarrow Model(S_t, A_t)$;
    Make an update by the rule (3);
  end for
end for

3.4. Dyna-Q+ algorithm

All above mentioned algorithms perform relatively good in the stochastic environment with the same underlying distribution for the state–action–reward dependency all the time. But real environments often are far from static. In an environment of a distribution grid, prices and consumption patterns may change drastically due to a variety of reasons, such as weather seasonality, new electrically powered equipment installations, solar panels integration, demand response due to the market incentives and others. All these may severely deteriorate the quality of the models (e.g. q-values become outdated, learned transition models become wrong). Above mentioned reasons lead to the necessity of the concept drift resistant approaches, that may quickly adjust to the changes in the environment. Presence of the ϵ parameter may save the model in case of minor drifts, but such important changes, as shifting of the optimal charge time to a different part of the day can easily stay unnoticed by the agent. The modification of the Dyna-Q algorithm, called Dyna-Q+ makes the agent more leaning towards exploration, during the planning stage. By introducing the additional reward for the states, not visited for a while, proportionally to the time they were not visited, the necessary exploration can be achieved:

$$r_p = r + \kappa\sqrt{\tau}$$

where r_p is a reward during a planning step, τ is an accumulative reward, which is zeroed in case the state is visited, and increased by 1 when not, κ is a small number.

With this, Dyna-Q+ algorithm can be written as:

Algorithm 4 Dyna-Q+ algorithm

Initialize $Q(S, A)$, $Model(S, A)$, $\tau(S, A)$ for all $s \in S^+, a \in A(s)$,
for each episode **do**
  Initialize $S_t \leftarrow$ current state;
  Initialize $A_t \leftarrow \epsilon$-greedy policy(S_t, Q);
  Take action A_t observe reward R and state S_{t+1};
  Make an update from the Q-learning update rule (3);
  $\tau(S, A) = \tau(S, A) + 1,\ \tau(S_t, A_t) = 0$
  $Model(S_t, A_t) \leftarrow R, S_{t+1}$ assuming the deterministic environment
  for each planning step **do**
    $S_t \leftarrow$ random previous state;
    $A_t \leftarrow$ random action, taken in the previous state;
    $R_t, S_{t+1} \leftarrow Model(S_t, A_t)$;
    $R_t \leftarrow R_t + \kappa\sqrt{\tau(S_t, A_t)}$;
    Make an update by the rule (3);
  end for
end for

3.5. Naive rule-based control

As an benchmark intuitive algorithm in this paper the rule-based control is considered. It follows easily comprehensible rules, which are to consume at earliest opportunity until fully charged under specific limitations. The limitations are that the agent has to consume only when the price or grid load are lower than a threshold. In our case for price we choose the median value one and the load is a value within first $P_{thresh} = 1$ slot in the discretized load. These values are based on the estimation of ability to fully charge the vehicle before the departure and consider other objectives at the same time. Mathematically the decision making process can be described as:

$$a(\lambda_t, P_{load,disc}) = \begin{cases} 1 \text{ if } (\lambda_t \leq \lambda_{med,t} \\ \text{and } P_{load,disc} \leq P_{thresh,disc}) \\ \text{and } SOC_t < 100 \\ 0 \text{ else} \end{cases} \quad (4)$$

where λ_t is the electricity price, $\lambda_{med,t}$ is a median price before departure or after arrival.

4. Study case

4.1. Environment

At first, the environment for the problem has to be properly defined in the form of MDP. Set of states is represented by the power flows problem results, in particular the currents of elements to be protected:

$$s_t = (P_{load,t}, SOC_t, t) \tag{5}$$

where $P_{load,t}$ is the active power consumption of the whole grid, SOC is a state of the charge. Power consumption is assumed to be acquired from the transformer in the particular model setting.

Set of actions is be given by the discretized power consumption of an EV battery, as a share of maximal:

$$a_t = \{0, \delta, 2\delta, \ldots, 1\} \tag{6}$$

where δ is a discretization step. The model choice in the paper consists of two discrete actions: charge 1 and stay still 0. Small amount of actions allow to decrease dimensionality of the problem and thus increase computational efficiency.

The reward function will represent the necessity to minimize cost as well as to avoid consumption in the high load of the elements and ensure the charging of the battery. Also the reward function features the rescaling of load and price terms in relation to the threshold values for them. It is needed to emphasize the goal and to avoid RL agent preferring not charging over charging everywhere.

$$R_t = a_t(\omega_{cost}(\lambda_{med,t} - \lambda_t)P_{max}\Delta t + \omega_{cap}\frac{P_{thresh} - P_{load,t}}{P_{thresh}}) + \omega_{SOC}(SOC_t - 100), \quad 0 < SOC_t < 100 \tag{7}$$

where P_{max} is a maximum consumption of EV charging, ω_{cost}, ω_{cap}, ω_{SOC} are reward factors for the cost, capacity and SOC reward terms, $P_{load,t}$ is the load of the transformer at time t, P_{thresh} is the transformer load threshold, used to divide rewards into positive and negative (in this case 25% of the transformer capacity).

The problem considered does not have clear episode division, since there is no any real resetting time point. That means one can consider time periods as episodes and assumptions for them. In this paper the time domain is split into days, as they are natural boundaries for the electrical energy consumption cycle. Because of this, it would be a valid assumption that the results of one day modeling will not affect the following day, which is a mandatory condition for the RL episodes.

States in the problem have to also be discretized to satisfy the requirements of the algorithms. $P_{trans,t}$ is divided into 4 intervals — less than 25, 50, 75 and 100% of the maximum consumption in the dataset. SOC is divided into 5 20% intervals.

Environment implemented in python as three separate objects: environment, grid and battery. The grid topology, used for the test is IEEE 13 bus test feeder [31], with substation transformer 0.25 MVA 10/0.4 kV (Fig. 4), thus, without step-down transformer between nodes 633 and 634. The main goal of the model is to show, if it is possible for RL algorithm, dispatching the EV charger output power and assuming the EV connected, to avoid overloading the transformer and to optimize cost of the consumption.

The other important aspect of the modeling is the algorithm that will control the behavior of the agent. There are 4 algorithms tested here: SARSA, Q-learning, Dyna-Q and Dyna-Q+. The main metric to compare their ability to learn is the cumulative reward after an episode. If objective function coefficients are selected correctly, the cumulative reward reflects the priorities of different goals to achieve. It is especially important, considering that the objectives are competitive, i.e. the faster state of charge is increased, the more transformer is loaded and money is spent and vice-versa. The consumer is interested in the faster charge and the distribution system at the lowest price, while DSO in keeping the distribution network not congested. Both targets are directly reflected by the cumulative reward.

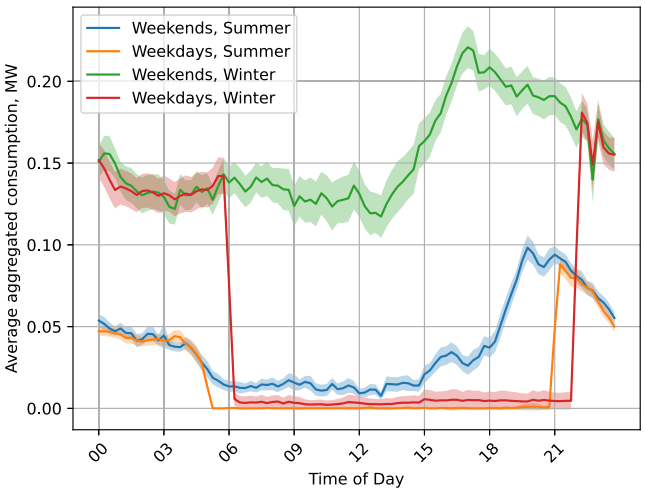


Fig. 2. Average aggregated load profiles for different scenarios, with 95% confidence intervals.

4.2. Data and environment non-stationarity

Data is acquired from industrial partners, it consists of two-year datasets with consumption of residential buildings. These buildings were assigned to the nodes of the network. It has 3 winter peaks since it starts at the beginning of 2016 and ends at the end of 2017. The data was scaled to fit the considered power grid. The aggregated load can go up to 0.1 MW in the summer and 0.2 MW in the winter time (Fig. 2). Only the weekdays were considered for the training process, as it is supposed, that people commute by EVs between home and work in the middle of the day, which is supported by the data.

The yearly seasonal variability poses a significant obstacle to algorithms. One of the reasons is that every season will likely put an agent into that part of the states, that was not visited before, because they have different consumption levels. In this case we would need more data to visit these new states enough to learn a proper policy. Moreover, the algorithm should have a consistently scaled reward to ensure the ability to compare the action values in different seasons and distinguish the good actions from the actions in good circumstances. For example, without removing the seasonal component of the load data a completely random action in a summer episode can easily give a better expected return than the optimal action in the winter episode. In other words, the reward function significantly depends not only on the agent's actions but also on the season, which deteriorates the training process. It is possible because the capacity penalty term depends on the absolute value of the load, which scale varies from season to season. To handle this issue, before training the loading data was deseasonalized using the multiplicative moving average method. The data was decomposed into a product of trend, season and residuals:

$$Trend(t) \cdot Season(t) \cdot Residuals(t) \tag{8}$$

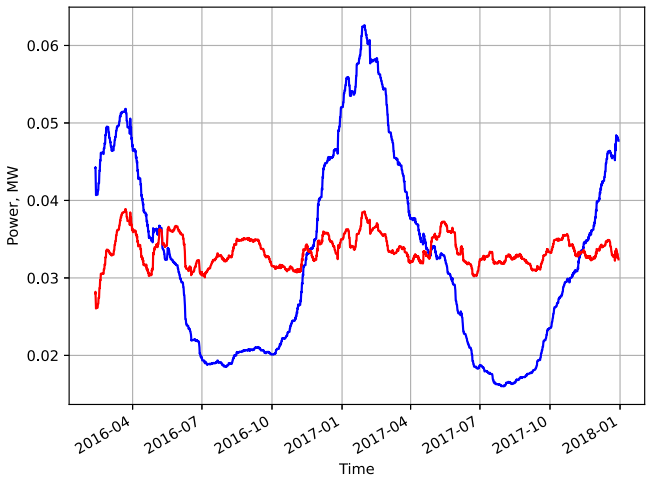


Fig. 3. Moving average of the original and normalized load profiles with the window of 1 month.

The moving average method of decomposition implies first processing the data through a convolution filter, with high enough window to capture the trend accurately. In our case, the window was selected to capture the time of the year deviation. The yearly seasonal deviation was approximated as a trend and removed from the data. The seasonal component represents the day-to-day seasonality, which is retrieved via averaging daily consumption. Residuals are calculated by removing trend and seasonality from the data. In total, the preprocessed load profile consists of two out of three components: daily seasonality and residuals. Daily seasonality is necessary as it contains important features of daily consumption for RL training process. The moving average of the original and deseasonalized load profiles are depicted on Fig. 3

Apart from the removed yearly seasonality, there will be considered an environment change from the side of the prices of electrical energy. The electricity day-ahead prices were assumed to be deterministic in this study. The change of the stationary condition will be represented by the price change: they will be zeroed in the last 2 h of the episode. It is expected to be difficult to notice this change for the agents since the optimal solution before the change is in some distance from this event.

The EV battery was chosen with 100 kWh capacity and the power of the EV charger is 40 kW. EV is assumed to have a stochastic arrival and departure schedule in the household. It departs from the household around 8 a.m. and arrives back around 6 p.m., both sampled from the normal distribution. After arrival, the state of the charge is assumed to reduce to 20% due to the consumption on the way.

ϵ parameter was selected to be 5% or 0.05 as a good balance between exploration and exploitation for the beginning of the training process.

5. Results

The comparison of the algorithms using deseasonalized data is presented on Fig. 5. Due to the high density and stochasticity of values, it is shown as an average for every 20 steps.

As it can be seen, different algorithms have different performances both in the beginning and after the concept drift in the environment data. Dyna-Q and Dyna-Q+ significantly outperformed simple tabular algorithms after the concept drift, while lagging a bit behind before it. The ability to notice the change of environment is connected to several planning updates after the occasional discovery is made, while tabular methods learn only from the experience. The lower performance in the beginning is connected with the fact, that these algorithms make many updates using only a few experience points they have in the model, thus overfitting to the first data points. The rule-based algorithm showed good performance at the beginning of the training process. After the concept drift its cumulated reward started to oscillate. This is due to several factors. First, because now there is a new very beneficial area in the end of the episode. Second, the load data still contains some noise, which is higher in certain periods of the dataset. While the algorithms are resistant to noise, the rule-based approach sometimes fails to fulfill the requirements to charge the vehicle. That means that the rule-based approach has to be manually adjusted to the change in the environment.

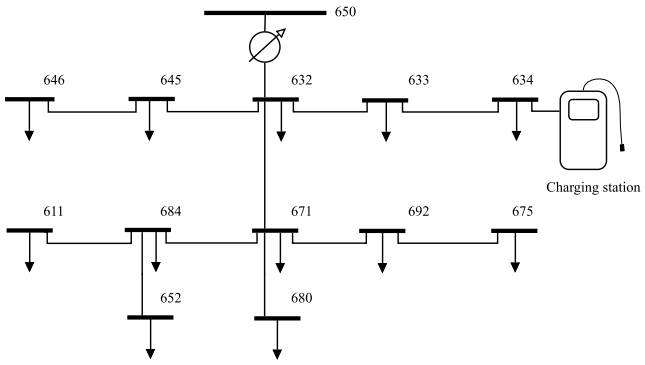


Fig. 4. Test distribution network based on IEEE 13 Node Test Feeder.

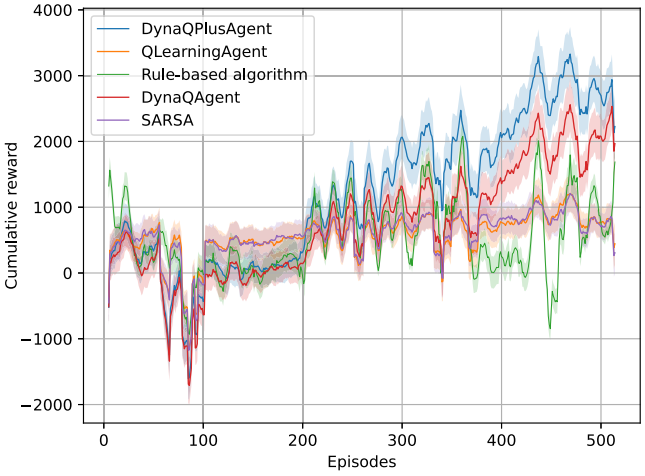


Fig. 5. Comparison of the training process with 50 independent runs for different algorithms with environment change in episode 200.

Interesting conclusions can be done from the behavior of different components of the multi-objective function. As it is seen in Fig. 6, the costs and transformer load rewards go up, while the SOC reward is going down (Fig. 7) before the concept drift. This is due to postponing the battery charge to more beneficial time points later. After the drift the capacity term starts to drop because at the end of the episode, the prices become much more important and the capacity is ignored at this time. Dyna-Q+ algorithm shows a better ability to learn the prices and sacrifice capacity reward for that.

The trained Dyna-Q+ policy was utilized during the roll-out experiment against the rule-based approach. The results are shown on Fig. 8. As it can be seen, the significant reduction

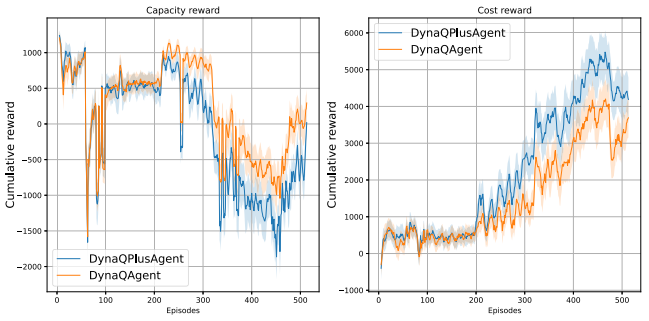


Fig. 6. Components behavior during the training process of the Q-learning algorithm.

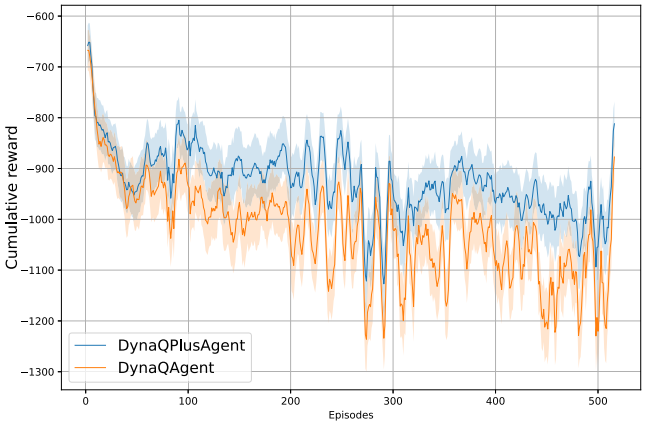


Fig. 7. SOC component of the reward function during the change of environment.

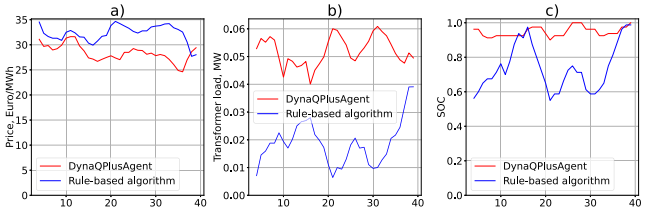


Fig. 8. Policy roll-out experiment. The trained policy was utilized on the last 40 days of the dataset and its performance metrics were traced: (a) average cost of the charge, (b) average transformer load during the charge, (c) SOC before departure or at the end of the day.

of the charging cost was achieved, maintaining the SOC at the appropriate level. This came with the cost of consuming in a more loaded hours. Considering, that the typical highly loaded times were avoided, it is not important, what is the transformer load during the charge. Also, the rule-based algorithm failed to charge to appropriate battery levels, since sometimes SOC does not exceed 60%.

A comparative analysis of different γ influence on the training process is provided in the paper. Discount factor γ represents, how valuable the algorithm perceives the information, achieved at later stages of an episode to estimate the value function at a current state. Generally, the lower the value of γ, the more important the agent considers the immediate reward in comparison to the future rewards, and vice-versa.

Another hyperparameter is the step size α, which defines, how much of the action-value is to be updated at each iteration. This parameter was selected as α = 0.5 as a trade-off between slow learning and overfitting to the most recent timesteps.

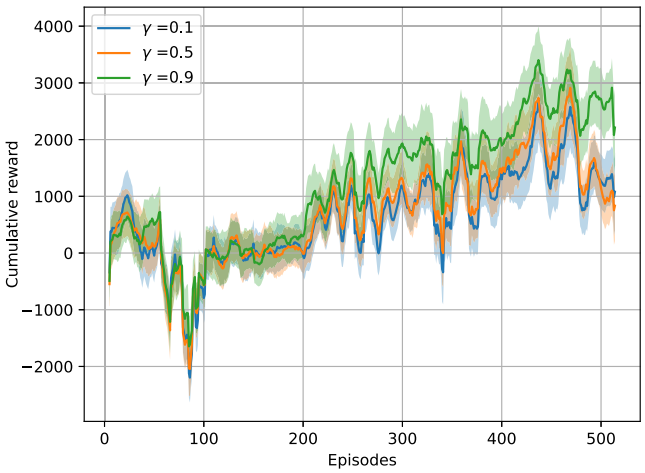


Fig. 9. Influence of γ on the training process for Dyna-Q+ algorithm with 20 independent runs.

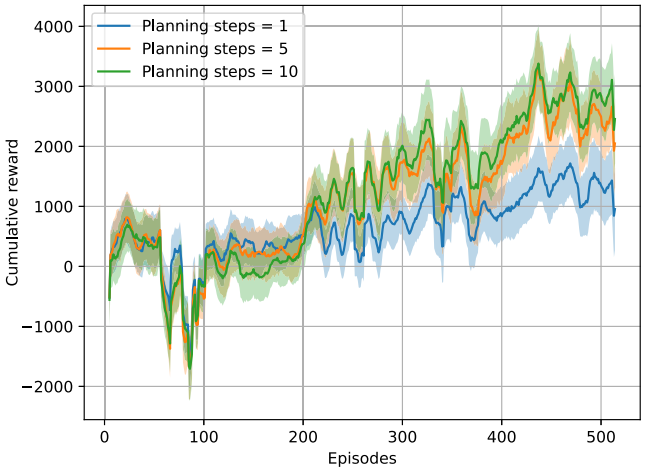


Fig. 10. Influence of the number of planning steps on the training process for Dyna-Q+ algorithm with 20 independent runs.

Some demonstrative tests with Dyna-Q+ algorithm and different γ with a fixed value of α = 0.5 are shown in Fig. 9.

As it can be seen, higher γs = 0.5 and 0.9 lead to the better learning after the environment change. This is due to the fact, that in order to distribute the delayed rewards back to the beginning of the episode it is necessary to take the future rewards as more important. Therefore higher discount rate leads to a larger share of the last timesteps' rewards will be accounted for in the first timesteps after arrival.

Other important parameters, that are part of Dyna-Q+ algorithm, are the number of planning steps during the experience replay and parameter κ, which is responsible for the scaling of the rewards from θ. Optimal κ depends on the scales of the multi-objective function factors and had to be coordinated with them. Generally, too high κ leads to a lower average reward even with the optimal policy. Too small kappa makes the exploration happen rarely and negates the effect of additional explorative reward, making the Dyna-Q+ to get closer to the regular Dyna-Q. The sensitivity analysis of planning steps with the fixed $\kappa = 0.3$ is presented on Fig. 10

A few planning steps would turn the algorithm into a simple Q-learning, which is seen in the figure. Higher value 10 leads a more stable result, than for 1 step. Number of planning steps

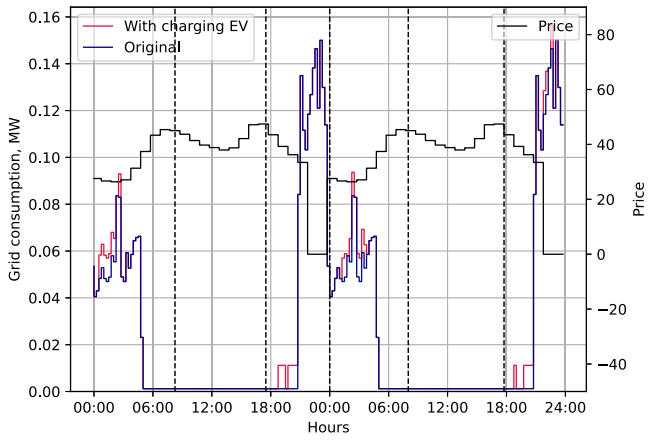


Fig. 11. Behavior of the agent in the distribution grid. Left: Q-leaning, right: Dyna-Q+.

was selected to be 10 as the higher number would not give any improvement, but decreased computation speed.

The trained agent manages to shift the charge to the points, where there is a trade-off between the load and the price rewards. The example behavior of one day with two algorithms (Q-learning and Dyna-Q+) is presented on Fig. 11. In this case, the simulation happens after the environment change, with zero price time slots in the evening. Times of departure and arrival are close, but a bit different due to the randomness of every simulation. In this plot, the EV departs in the morning around 8 a.m. (dashed line in the morning) and arrives around 6 p.m (dashed line in the evening). As it can be seen, in the case of Dyna-Q+ algorithm the agent prefers now the area with zero price, contrary to Q-learning, which still stays away from the evening time. This happens because Q-learning algorithm visits the evening hours rarely and cannot exploit that experience to the full extent, while Dyna-Q+ has the incentive to explore the state space once in a while and to uncover new rewarding trajectories.

6. Conclusions

In this work the possibility of using a reinforcement learning model to ease the burden of growing number of electrical vehicles was shown. Several algorithms, including completely model-free Q-learning and SARSA and planning Dyna-Q and Dyna-Q+ were considered. It was proven, that replaying experience can boost the learning process in the changing environment, by applying more iterative updates with help of the generated transition model. It was also discussed, that the system peaks can be a threat to the reinforcement learning algorithms, but the data deseasonalization can ease the issue and make rewards more consistent. It was proven, that the exploration bonus to the Dyna-Q model can have a positive effect and help to faster discover the more optimal solutions in changing environments. Although the tabular methods are generally known for their limited scalability, the same principles could be applied to other techniques, e.g. deep reinforcement learning algorithms.

There were also observed some limitations of the proposed approach. First, the learning process would be much more difficult and results unclear without the data preprocessing. In this study, we focused on the deseasonalization of the data, but the general idea is to make the rewards proportional only to how good the agent action is and not connected to the specific features of the dataset in a particular timepoint. Second, the learning process depends very much on the multi-objective reward function shapes and their factors, especially on their mutual coordination, which requires some effort to tune.

CRediT authorship contribution statement

Andrey Poddubnyy: Conceptualization, Methodology, Visualization, Software, Writing – original draft. **Phuong Nguyen:** Conceptualization, Writing – review & editing, Project administration, Supervision. **Han Slootweg:** Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andrey Poddubnyy reports financial support and administrative support were provided by Dutch Research Council.

Data availability

The authors do not have permission to share data.

Acknowledgment

This publication is part of the research program 'MegaMind - Enabling distributed operation of energy infrastructures through Measuring, Gathering, Mining and Integrating grid-edge Data', (partly) financed by the Dutch Research Council (NWO), through the Perspectief funding instrument under number P19-25.

References

[1] R. Zhang, Z. Li, C. Wei, Y. Li, J. Yang, S. Su, Optimization and solution method for electric vehicle charging and discharging load, in: 2021 IEEE 4th International Electrical and Energy Conference, CIEEC, 2021, pp. 1–6, http://dx.doi.org/10.1109/CIEEC50170.2021.9510947.

[2] N.A. El-Taweel, H. Farag, M.F. Shaaban, M.E. AlSharidah, Optimization model for EV charging stations with PV farm transactive energy, IEEE Trans. Ind. Inform. 18 (7) (2022) 4608–4621, http://dx.doi.org/10.1109/TII.2021.3114276.

[3] S. Ayyadi, H. Bilil, M. Maaroufi, Optimal charging of electric vehicles in residential area, Sustain. Energy Grids Netw. 19 (2019) 100240, http://dx.doi.org/10.1016/j.segan.2019.100240.

[4] H.M. Abdullah, A. Gastli, L. Ben-Brahim, Reinforcement learning based EV charging management systems–A review, IEEE Access 9 (2021) 41506–41531, http://dx.doi.org/10.1109/ACCESS.2021.3064354.

[5] M.G. Vayá, L.B. Roselló, G. Andersson, Optimal bidding of plug-in electric vehicles in a market-based control setup, in: 2014 Power Systems Computation Conference, 2014, pp. 1–8, http://dx.doi.org/10.1109/PSCC.2014.7038108.

[6] A. Poddubnyy, P. Nguyen, H. Slootweg, Network-aware online charge control with reinforcement learning, in: 2022 International Conference on Smart Energy Systems and Technologies, SEST, 2022, pp. 1–6, http://dx.doi.org/10.1109/SEST53650.2022.9898460.

[7] A. Pal, A. Bhattacharya, A.K. Chakraborty, Allocation of electric vehicle charging station considering uncertainties, Sustain. Energy Grids Netw. (2020) 100422.

[8] R. Shabbar, A. Kasasbeh, M.M. Ahmed, Charging station allocation for electric vehicle network using stochastic modeling and grey wolf optimization, Sustainability 13 (6) (2021) http://dx.doi.org/10.3390/su13063314.

[9] P.C. Bons, A. Buatois, F. Schuring, F. Geerts, R. van den Hoed, Flexible charging of electric vehicles: Results of a large-scale smart charging demonstration, World Electr. Veh. J. 12 (2) (2021) http://dx.doi.org/10.3390/wevj12020082.

[10] A. Poddubny, D. Senchuk, A. Gonzalez-Castellanos, D. Pozo, Demand response on the Russian retail market, in: 2021 IEEE Madrid PowerTech, 2021, pp. 1–5, http://dx.doi.org/10.1109/PowerTech46648.2021.9494852.

[11] Y. Dahmane, R. Chenouard, M. Ghanes, M. Alvarado-Ruiz, Optimized time step for electric vehicle charging optimization considering cost and temperature, Sustain. Energy Grids Netw. 26 (2021) 100468, http://dx.doi.org/10.1016/j.segan.2021.100468.

[12] A.K. Vamsi Krishna Reddy, K. Venkata Lakshmi Narayana, Meta-heuristics optimization in electric vehicles -an extensive review, Renew. Sustain. Energy Rev. 160 (2022) 112285, http://dx.doi.org/10.1016/j.rser.2022.112285, [Online]. Available:https://www.sciencedirect.com/science/article/pii/S1364032122002040.

[13] T. Ding, Z. Zeng, J. Bai, B. Qin, Y. Yang, M. Shahidehpour, Optimal electric vehicle charging strategy with Markov decision process and reinforcement learning technique, IEEE Trans. Ind. Appl. 56 (5) (2020) 5811–5823, http://dx.doi.org/10.1109/TIA.2020.2990096.

[14] P.P. Vergara, M. Salazar, J. Giraldo, P. Palensky, Optimal dispatch of PV inverters in unbalanced distribution systems using reinforcement learning, Int. J. Electr. Power Energy Syst. 136 (2022) 107628, http://dx.doi.org/10.1016/j.ijepes.2021.107628.

[15] E. Mocanu, D.C. Mocanu, P.H. Nguyen, A. Liotta, M.E. Webber, M. Gibescu, J.G. Slootweg, On-line building energy optimization using deep reinforcement learning, IEEE Trans. Smart Grid 10 (4) (2019) 3698–3708, http://dx.doi.org/10.1109/TSG.2018.2834219.

[16] A. Chiş, J. Lundén, V. Koivunen, Reinforcement learning-based plug-in electric vehicle charging with forecasted price, IEEE Trans. Veh. Technol. 66 (5) (2017) 3674–3684, http://dx.doi.org/10.1109/TVT.2016.2603536.

[17] F.L. da Silva, C.E.H. Nishida, D.M. Roijers, A.H.R. Costa, Coordination of electric vehicle charging through multiagent reinforcement learning, IEEE Trans. Smart Grid 11 (2020) 2347–2356.

[18] J. Shi, Y. Gao, W. Wang, N. Yu, P. Ioannou, Operating electric vehicle fleet for ride-hailing services with reinforcement learning, IEEE Trans. Intell. Transp. Syst. PP (2019) 1–13, http://dx.doi.org/10.1109/TITS.2019.2947408.

[19] K. Valogianni, W. Ketter, J. Collins, D. Zhdanov, Effective management of electric vehicle storage using smart charging, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28, No. 1, 2014, http://dx.doi.org/10.1609/aaai.v28i1.8760, [Online]. Available:https://ojs.aaai.org/index.php/AAAI/article/view/8760.

[20] I. Dusparic, A. Taylor, A. Marinescu, V. Cahill, S. Clarke, Maximizing renewable energy use with decentralized residential demand response, 2015, pp. 1–6, http://dx.doi.org/10.1109/ISC2.2015.7366212.

[21] F. Zhang, Q. Yang, D. An, CDDPG: A deep reinforcement learning-based approach for electric vehicle charging control, IEEE Internet Things J. PP (2020) 1, http://dx.doi.org/10.1109/JIOT.2020.3015204.

[22] T. Ding, J. Bai, Z. Zeng, B. Qin, Y. Yang, M. Shahidehpour, Optimal electric vehicle charging strategy with Markov decision process and reinforcement learning technique, IEEE Trans. Ind. Appl. PP (2020) http://dx.doi.org/10.1109/TIA.2020.2990096.

[23] A. Chiş, J. Lundén, V. Koivunen, Scheduling of plug-in electric vehicle battery charging with price prediction, in: IEEE PES ISGT Europe 2013, 2013, pp. 1–5, http://dx.doi.org/10.1109/ISGTEurope.2013.6695263.

[24] M. Dorokhova, Y. Martinson, C. Ballif, N. Wyrsch, Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation, Appl. Energy 301 (2021) 117504, http://dx.doi.org/10.1016/j.apenergy.2021.117504.

[25] A. Sangadiev, A. Poddubny, D. Pozo, A. Gonzalez-Castellanos, Quasi-Newton methods for power flow calculation, in: 2020 International Youth Conference on Radio Electronics, Electrical and Power Engineering, REEPE, 2020, pp. 1–6, http://dx.doi.org/10.1109/REEPE49198.2020.9059230.

[26] A. Marinescu, I. Dusparic, S. Clarke, Prediction-based multi-agent reinforcement learning in inherently non-stationary environments, ACM Trans. Auton. Adapt. Syst. 12 (2017) 1–23, http://dx.doi.org/10.1145/3070861.

[27] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, S. Whiteson, Transient non-stationarity and generalisation in deep reinforcement learning, 2021, arXiv:2006.05826.

[28] S. Padakandla, P.K. J., S. Bhatnagar, Reinforcement learning in non-stationary environments, 2019, CoRR abs/1905.03970 [Online]. Available:http://arxiv.org/abs/1905.03970.

[29] F. Wang, J. Gao, M. Li, L. Zhao, Autonomous PEV charging scheduling using dyna-Q reinforcement learning, IEEE Trans. Veh. Technol. PP (2020) 1, http://dx.doi.org/10.1109/TVT.2020.3026004.

[30] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[31] W. Kersting, Radial distribution test feeders, in: 2001 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.01CH37194), Vol. 2, 2001, pp. 908–912, http://dx.doi.org/10.1109/PESW.2001.916993.