Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning

Nasrin Sadeghianpourhamami, *Student Member, IEEE*, Johannes Deleu, and Chris Develder, *Senior Member, IEEE*

***Abstract*—Demand response (DR) becomes critical to manage the charging load of a growing electric vehicle (EV) deployment. Initial DR studies mainly adopt model predictive control, but models are largely uncertain for the EV scenario (e.g., customer behavior). Model-free approaches, based on reinforcement learning (RL), are an attractive alternative. We propose a new Markov decision process (MDP) formulation in the RL framework, to jointly coordinate a set of charging stations. State-of-the-art algorithms either focus on a single EV, or control an aggregate of EVs in multiple steps (e.g., (1) make aggregate load decisions, (2) translate the aggregate decision to individual EVs). In contrast, our RL approach jointly controls the whole set of EVs at once. We contribute a new MDP formulation with a scalable state representation independent of the number of charging stations. Using a batch RL algorithm, fitted Q-iteration, we learn an optimal charging policy. With simulations using real-world data, we (i) differentiate settings in training the RL policy (e.g., the time span covered by training data), (ii) compare its performance to an oracle all-knowing benchmark (providing an upper performance bound), (iii) analyze performance fluctuations throughout a full year, and (iv) demonstrate generalization capacity to larger sets of charging stations.**

***Index Terms*—demand response, electric vehicles, batch reinforcement learning.**

NOMENCLATURE

s	State
s'	The next state from s
Δt^{depart}	Time left until departure
Δt^{charge}	Time needed for charging completion
t^{arrival}	Time of arrival
Δt^{flex}	Flexibility (time charging can be delayed)
N_s	Number of connected EVs in state s
$\mathcal{V}_t$	Set of EVs in the system at time t
$\mathbf{x}_s$	Aggregate demand in state s
t	Timeslot
Δt^{slot}	Duration of a decision slot
S_{max}	Maximum number of decision slots
H_{max}	Maximum connection time
N_{max}	Number of charging stations jointly being coordinated
$\mathbf{u}_s$	Action taken in state s
$\mathbf{U}_s$	Set of possible actions from state s
$\mathbf{x}_s^{\text{total}}(d)$	Total number of EVs on the d^{th} upper diagonal of $\mathbf{x}_s$
C^{demand}	Cost of total power consumption
C^{penalty}	Penalty cost for unfinished charging
$C(s, \mathbf{u}_s, s')$	Instantaneous cost of state transition
$\mathcal{B}^{\text{test}}$	Test set
$\mathcal{B}^{\text{train}}$	Training set
e_i	set of tuples $(t^{\text{arrival}}, \Delta t^{\text{depart}}, \Delta t^{\text{charge}})$ in day i
Δt	Training data time span
C_π	Normalized cost of policy π
C_{BAU}	Normalized cost of business-as-usual policy
C_{RL}	Normalized cost of the learned policy
C_{opt}	Normalized cost of optimum solution

N. Sadeghianpourhamami, J. Deleu and C. Develder are with IDLab, Dept. of Information Technology, Ghent University – imec, Ghent, Belgium, e-mail: <firstname>.<lastname>@ugent.be

I. INTRODUCTION

DEMAND response (DR) algorithms aim to coordinate the energy consumption of customers in a smart grid to ensure demand-supply balance and reliable network performance. In initial DR studies, the demand response problem usually is cast as a model predictive control (MPC) approach (e.g., [1], [2]), typically formulated as an optimization problem to minimize the customer's electricity bill or maximize the energy provider's profit, subject to various operating constraints (e.g., physical characteristics of the devices, customer preferences, distributed energy resource constraints and energy market constraints). However, widespread deployment of such model-based DR algorithms is limited because (i) the heterogeneity of end user loads, the broad range of user behavior patterns and the uncertainty surrounding that behavior makes the modeling task very challenging [3], and (ii) model-based DR algorithms are difficult to transfer from one scenario to another, since the model is likely to require substantial customization/tweaking, e.g., for different user groups. In this paper, we therefore focus on developing an essentially model-free approach, that is purely data-driven.

Recently, reinforcement learning (RL) has emerged to facilitate such model-free control for coordinating user flexibility in DR and defines the DR problem as a Markov decision process (MDP). A coordinating agent interacts with the environment (i.e., DR participating customers, energy providers, energy market prices, etc.) and takes control actions while aiming to maximize the long term expected reward (or to minimize the long term expected cost). In other words, the agent learns by taking actions and observing the outcomes (i.e., states) and the rewards/costs in an iterative process. The DR objective (e.g., load flattening, load balancing) is achieved by appropriately designing the reward/cost signal. Hence, RL-based approaches do not need an explicit model of user flexibility behavior nor energy pricing: RL facilitates more practical and generally applicable DR schemes compared to model-based approaches. Yet, a main challenge of RL-based DR approaches is the curse of dimensionality due to the continuity and scale of the state

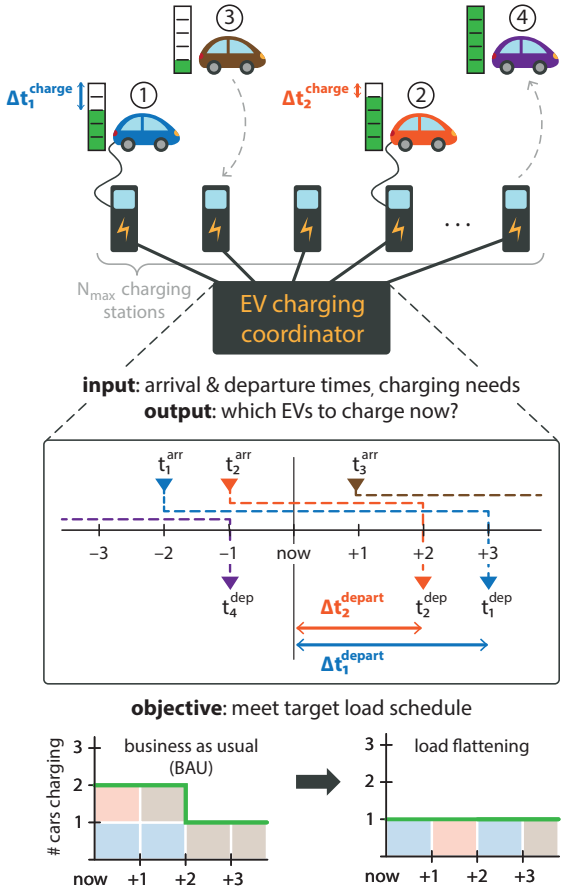

Fig. 1: An illustration of the EV charging coordination problem, with 2 cars currently connected (and remaining so for Δt^{depart}), with indicated arrival and departure times (t^{arr} and t^{dep}, measured in timeslots) as well as charging needs (noted as time left to charge, Δt^{charge}).

and action spaces, hindering applicability of RL-based DR to large-scale problems.

Here, we focus on formulating a scalable RL-based DR algorithm to jointly coordinate a group of electric vehicle (EV) charging stations, which generalizes to various group sizes and EV charging rates. The general problem is sketched in Fig. 1: we consider a system comprising N_{max} charging stations, where EVs dynamically come and go over time. We consider discrete timeslots, and need the coordination system to decide which cars to charge. To make that decision, we assume the state and future departure of the cars in the system (vehicles 1 and 2 in the example) are known, but future arrivals (such as vehicle 3) are not. The DR objective is to meet a target load schedule. In the current paper, we will focus on load flattening, i.e., minimize the load and thus spread out consumption equally over time — which amounts to a concept often referred to as peak shaving [4], [5] and/or valley filling [6], [7] — but the approach is generalizable to other objectives. Our work is aimed as an exploratory proof-of-concept study that demonstrates feasibility and explores the performance of a scalable RL approach.

Current literature indeed only offers a limited amount of model-free solutions for jointly coordinating multiple EV charging stations (see Section II). Such existing RL-based DR solutions are either developed for an individual EV or need a heuristic (which does not guarantee an optimum solution) to obtain the aggregate load of a set of EV charging stations during the learning process. Indeed, current literature does not provide a scalable Markov decision process (MDP) formulation that generalizes to a collection of EV charging stations with different characteristics (e.g., charging rates, size). We take the first step[1] to fill this gap by proposing an MDP and explore its performance in simulation experiments. Note that we frame this paper as a first proof-of-concept of our proposed MDP, focusing on demonstrating feasibility and exploring its main characteristics. More precisely, in this paper:

- We define a new MDP with compact state and action space representations, which do *not* linearly scale with the number of EV charging stations, can generalize to collections of various sizes and can be extended to cope with heterogeneous charging rates (Section III),
- We adopt batch reinforcement learning (fitted Q-iteration [9]) with function approximation to find an optimal EV charging policy (Section IV),
- We quantitatively explore the performance of the proposed RL approach with simulations using real-world data in experiments covering 10 and 50 charging stations (setup details in Section V), to answer research questions (Section VI):
  (Q1) What is the impact of varying parameter settings of the input training data?[2]
  (Q2) How does the RL policy perform compare to an optimal all-knowing oracle algorithm?
  (Q3) How does that performance vary over time (i.e., from one month to the next) using realistic data?
  (Q4) Does a learned approach generalize to different EV group sizes?

Section VII summarizes our conclusions and open issues.

II. Related Work

A substantial body of research has focused on proposing model-based DR algorithms (e.g., MPC approach) for EV charging coordination: for recent reviews, we refer to [10], [11]. However, as explained earlier, the widespread deployment of such algorithms in smart grid is often hindered by (i) the need for precise system models models, as well as (ii) limited generalizability of proposed solutions. With growing EV adoption, also the amount of available (and realistic) EV data increased. Hence, data-driven approaches to coordinate EV charging gained attention, with reinforcement learning (RL) as a notable example — which is also increasingly popular in demand response beyond the EV case (see [12] for a recent overview). For example, Shi *et al.* [13] adopt an RL-based approach and phrase an MDP to learn to control the charging and discharging of an **individual EV** under price uncertainty for providing vehicle-to-grid (V2G) services. Their

[1]The model is a significant refinement of our initial proposal [8], which just formulated an MDP and did not have any experimental results.

[2]The parameters of interest are (i) time span of the training data, and (ii) number of sampled trajectories from the decision trees. For details see Section IV-B and Section V-B.

MDP has (i) a state space based on the hourly electricity price, state-of-charge and time left till departure), (ii) an action space to decide between charging (either to fulfill the demand or provide frequency regulation), delaying the charging and discharging for frequency regulation (iii) unknown state transition probabilities. The reward is defined as the energy payment of charging and discharging or the capacity payment (for the provided frequency regulation service). Chis *et al.* [14] use batch RL to learn the charging policy of again an **individual EV**, to reduce the long-term electricity costs for the EV owner. An MDP framework represents this problem, where (i) the state space consists of timing variables, minimum charging price for a current day and price fluctuation between the current and the next day, while (ii) the action is the amount of energy to consume in a day. Cost savings of 10%-50% are reported for simulations using real-world pricing data. Opposed to these cost-minimizing approaches assuming time-varying prices, as a first case study for our joint control of a group of EV charging stations, we will focus first on a load flattening scenario (i.e., electricity prices are assumed constant, but peaks need to be avoided).

In contrast to [13] and [14], which consider the charging of a single EV, Claessens *et al.* [4] use batch RL to learn a collective charging plan for a **group of EVs** in the optimization step of their previously proposed three step DR approach [5]. Their three-step DR approach comprises (i) an aggregation step, (ii) an optimization step, and (iii) a real-time control step. Step (i) aggregates individual EV constraints. In step (ii), the aggregated constraints are used by the batch RL agent to learn the collective charging policy for the EV fleet, which is translated to a sequence of actions (i.e., aggregated power consumption values for each decision slot) to minimize energy supply costs. Finally, in step (iii) a priority based heuristic algorithm dispatches the energy corresponding to the action determined in the optimization step from the individual EVs. Vandael *et al.* [15] also use batch RL to learn a cost-effective day-ahead consumption plan for a **group of EVs**. Their formulation also has two decision phases, (i) day-ahead and (ii) intra-day.

Note that our work is different from [4] and [15] in two aspects: (i) unlike [4], [15], our approach does not take control decisions in separate steps (i.e., first deciding on aggregate energy consumption and then coordinating individual EV charging to meet that aggregate target) and instead it takes decisions directly and jointly for all individual EVs using an efficient representation of the aggregate state of a group of EVs, hence (ii) our approach does not need a heuristic algorithm, but instead learns the aggregate load while finding an optimum policy to flatten the load curve. The next sections describe our MDP model and the batch reinforcement learning approach to train it.

III. Markov Decision Process (MDP)

A Markov decision process (MDP) as a mathematical framework is characterized by (i) a finite state space, (ii) a finite set of possible actions that impact transitions from one state to the next, (iii) transition probabilities dictating the likelihood of such stochastic state evolutions, and (iv) a cost function (or, alternatively, rewards) associated with those transitions. The *state* represents the problem/environment an agent is interacting with by taking an *action* and observing a *cost/reward* and the *next state*. In our specific setting, we aim to minimize the cost of charging a group of EVs for an aggregator, in a real-time slotted decision-making setup: we consider a sequential decision making problem and formulate it using an MDP with unknown transition probabilities.

Concretely, at each timeslot, we need to decide whether or not to charge each of the present EVs. The input for that decision, which represents the *state* of the EV coordinating problem, comprises the current timeslot and the joint charging demand of the connected EVs. Further, we represent the charging decisions for the present EVs at each state in form of an *action* that dictates which of the present EVs to charge in the current timeslot. The next state depends on the action taken and newly arriving EVs in the next timeslot. Since we do not assume prior knowledge of future arrivals, the *state transition probabilities* are unknown in our problem. The learning objective is to find a charging policy that determines what action to take in each state of the coordination problem such that the long term expected *cost* is minimized. In particular, we focus on a load flattening scenario (a concept also commonly referred to as peak shaving [4], [5] and/or valley filling [6], [7]), minimizing the peak-to-average ratio of the aggregate load of the group of EVs: a convex cost function sums the squared total power consumption over all timeslots in the decision time horizon. Further, as a user constraint, our cost function will include a penalty term to ensure that charging is completed before departure. More advanced DR objectives and constraints are left for future work.

Given that transition probabilities are a priori unknown, the (charging) policy is often learned from interactions with the environment via taking actions and observing the outcome in form of a cost and the next state. Note that in this process, the learning agent also implicitly learns the transition probabilities (or what to expect as future arrivals and their charging demands). Hence, our approach (or any other RL method) is based on the core assumption that EV arrivals and the associated charging needs exhibit some regularity/predictability. Previous studies of EV charging datasets (e.g., [16]) suggest that this assumption holds.

For a comprehensive overview of reinforcement learning, especially in demand response, we refer to [12]. Now, we will detail each of the MDP components of our approach in turn, while the adopted learning algorithm will be explained in Section IV.

A. State Space

An EV charging session is characterized by: (i) EV arrival time, (ii) time left till departure (Δt^{depart}), (iii) requested energy and (iv) EV charging rate. For (i), since we do not assume knowledge of future arrivals, we do not explicitly include arrival time in the current EVs' representation. For (iii)–(iv), we implicitly assume the same charging rate for all EVs in a group, and translate the requested energy to time

needed to complete the charging (Δt^{charge}). Thus, if we have N_s electric vehicles in the system, the (remaining times of) their sessions are represented as a set

$$\mathcal{V}_t = \{(\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}), \ldots, (\Delta t_{N_s}^{\text{depart}}, \Delta t_{N_s}^{\text{charge}})\}.$$

Each state s is represented using two variables: timeslot (i.e., $t \in \{1, \ldots, S_{\text{max}}\}$) and the aggregate demand (i.e., $\mathbf{x}_s$), hence $s = (t, \mathbf{x}_s)$. Inspired by [17], aggregate demand at each given timeslot is obtained via a binning algorithm: the demand is represented using a 2D grid, thus a matrix $\mathbf{x}_s$, with one axis representing Δt^{depart}, the other Δt^{charge}. That means that the element of $\mathbf{x}_s$ at position (i, j) counts the number EV charging sessions in the corresponding $(\Delta t^{\text{depart}}, \Delta t^{\text{charge}})$ bin, i.e., those for which $i = \left\lceil \frac{\Delta t^{\text{depart}}}{\Delta t^{\text{slot}}} \right\rceil$ and $j = \left\lceil \frac{\Delta t^{\text{charge}}}{\Delta t^{\text{slot}}} \right\rceil$. The resulting matrix $\mathbf{x}_s$ has dimension $S_{\text{max}} \times S_{\text{max}}$, which depends on the maximal connection time H_{max}, i.e., the longest duration of an EV being connected to a charging station: $S_{\text{max}} \triangleq H_{\text{max}}/\Delta t^{\text{slot}}$. This ensures scalability to various system sizes: the maximal number of cars N_{max} does not impact state size.

Further, we do not use absolute counts as entries in the matrix $\mathbf{x}_s$, but rather divide the number of sessions in a given (i, j) cell by the system capacity, i.e., the number of EV charging stations N_{max}. This makes the state representation scale-free, i.e., independent of N_{max}. Thus we aim the formulated MDP (and the learned control policy) to be generalizable to a different number of EV charging stations. Note that as time progresses, cars will move towards lower Δt^{depart} cells, and if charged) also lower Δt^{charge}.[3] Since time-of-day is likely to influence the expected evolution of the state $\mathbf{x}_s$ (and hence the required response action we should take), we also include the current timeslot t as explicit part of the state.

Figure 2 illustrates a simple scenario of $N_{\text{max}} = 2$ charging stations with a horizon of $S_{\text{max}} = 3$ slots. Let us assume that at time $t = 1$ we have $N_s = 2$ connecting cars: $\mathcal{V}_1 = \{(\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}) = (3, 2), (\Delta t_2^{\text{depart}}, \Delta t_2^{\text{charge}}) = (2, 1)\}$, with no other arrivals during the control horizon. Figure 2 depicts the resulting state space using the binning algorithm in the first timeslot. EVs are binned according to their Δt^{depart} and Δt^{charge} in a 2D grid of size 3×3. The resulting matrix is normalized by N_{max} ($= 2$ in this example). The shaded grid cells in Fig. 2 indicate bins with $\Delta t^{\text{charge}} \leq \Delta t^{\text{depart}}$: EVs in these bins have enough time to complete their charging.

Our state representation $\mathbf{x}_s$ not only has a scalability merit in that it summarizes the aggregated demand of connecting EVs (in terms of Δt^{depart} and Δt^{charge}) in a compact and comparative form, but also naturally facilitates making similar charging decisions for EVs with similar charging needs. In particular, the *flexibility* in terms of how long the charging can be delayed (denoted as $\Delta t^{\text{flex}} = \Delta t^{\text{depart}} - \Delta t^{\text{charge}}$) is easily inferred from the diagonals of $\mathbf{x}_s$:

$$\Delta t^{\text{flex}}(i, j) = j - i \qquad \forall i, j \in \{1, \ldots, S_{max}\}. \tag{1}$$

Equation (1) indicates that EVs binned into cells on the main diagonal of $\mathbf{x}_s$ (i.e., $i = j$) have zero flexibility, while the ones in cells on the upper diagonals of $\mathbf{x}_s$ are flexible. Negative

[3] An extension to consider the variable charging rate is possible by binning the EVs in a 3D grid with charging rate as the third dimension.

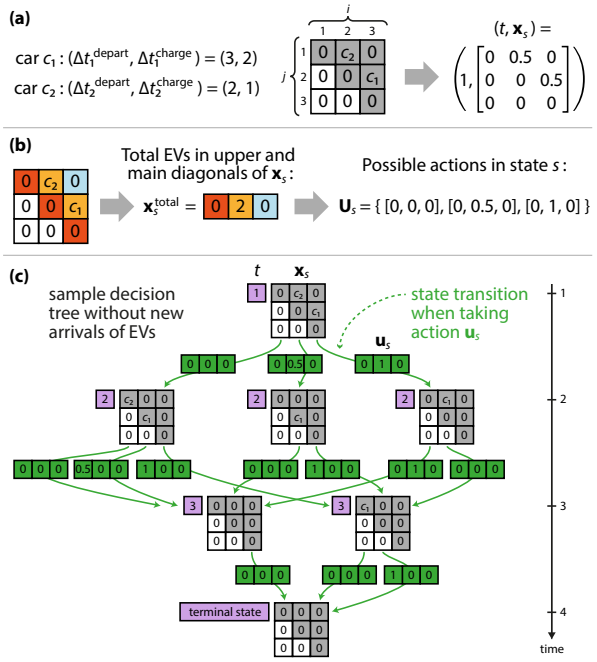

Fig. 2: A simple example for $N_{\text{max}} = 2$ charging stations: (a) state representation, (b) possible action states, (c) full decision tree over the horizon of $S_{\text{max}} = 3$ slots.

Δt^{flex}, corresponding to lower diagonals of $\mathbf{x}_s$ (i.e., white cells in the 2D grids of Fig. 2), implies EVs whose charging demand cannot be fulfilled. However, our control strategy will ensure that EVs charging demands are never violated, using a penalty term in our cost function (see Section III-C). Such straightforward identification of flexibility allows to make consistent charging decisions for EVs with similar flexibility (as explained in Section III-B).

Finally, while the dimension of $\mathbf{x}_s$ (thus the state space size) is independent of the maximal number of cars N_{max}, it is still dependent on S_{max} (thus H_{max} and Δt^{slot}). This independence from N_{max} ensures scalability of the state representation to various group sizes of EV charging stations (see our analysis for Q4). Yet, increasing S_{max} may still lead to a huge state space. Adapting the size of $\mathbf{x}_s$ (hence S_{max}) to limit the state space is left for future work: as previously indicated, the current paper focuses on providing a proof-of-concept for our proposed MDP formulation to jointly coordinate charging of multiple EVs.

B. Action Space

The action to take in state $s = (t, \mathbf{x}_s)$ is a decision whether or not to charge the currently connected EVs. We will make decisions based on Δt^{flex}. EVs with the same Δt^{flex} are binned into the cells on the same diagonal of $\mathbf{x}_s$, as explained in the previous section. We indicate each diagonal of $\mathbf{x}_s$ as $\mathbf{x}_s(d)$ with $d = 0, \ldots, S_{\text{max}} - 1$ where $\mathbf{x}_s(0)$ is the main diagonal, $\mathbf{x}_s(d)$ is the upper d^{th} diagonal, and $\mathbf{x}_s(-d)$ is the lower d^{th} diagonal of $\mathbf{x}_s$. We denote $\mathbf{x}_s^{\text{total}}(d)$ as the total number of EVs in the cells on the d^{th} diagonal. Since we assume no infeasible requests are issued (no arrivals of EVs in the lower

diagonal cells), and we will honor feasible requests (we do do not let cars move to those lower diagonals), the action taken in state s is defined as a vector $\mathbf{u}_s$ of length $S_{\max}$: we define the action vector for charging/delaying the cars on the main and upper diagonals of $\mathbf{x}_s$ only (colored cells in Fig. 2). This design choice keeps the action space relatively small and therefore easier to explore. Further, in each timeslot we charge any given car either at full power or not at all. Thus the d^{th} element of action vector $\mathbf{u}_s$ will be a number in $[0, 1]$: the fraction of EVs to charge from the corresponding d^{th} diagonal of $\mathbf{x}_s$. $\mathbf{U}_s$ denotes the set of possible actions from state s.

Figure 2(b) illustrates how $\mathbf{U}_s$ is constructed from state s, starting from matrix $\mathbf{x}_s$ with an intermediate vector $\mathbf{x}_s^{\text{total}}$. In the exemplary state, there are 2 EVs in the upper first diagonal only, hence the 1st and the 3rd element of $\mathbf{u}_s$ corresponding to the main and 2nd upper diagonal, are zero. There are 3 possible actions to take in this state, depending on the fraction of the EVs to charge on the 1st upper diagonal: charging none of the cars (i.e., $\mathbf{u}_s[2] = 0$), only one of them ($\mathbf{u}_s[2] = 0.5$), or both ($\mathbf{u}_s[2] = 1$) in the current slot.

C. Cost function

The objective we consider is to flatten the aggregate charging load of a group of EVs while ensuring each EV's charging is completed before departure.[4] Hence, the cost for a transition from state s to s' by taking action $\mathbf{u}_s$ has two parts:

$$C(s, \mathbf{u}_s, s') = C^{\text{demand}}(\mathbf{x}_s, \mathbf{u}_s) + C^{\text{penalty}}(\mathbf{x}_{s'}), \tag{2}$$

with

- $C^{\text{demand}}(\mathbf{x}_s, \mathbf{u}_s)$: the cost of the total power consumption from all the connected EVs for a decision slot, and
- $C^{\text{penalty}}(\mathbf{x}_{s'})$: the penalty for unfinished charging.

To achieve load flattening, we choose C^{demand} to be a quadratic function of the total power consumption for a decision slot. Since we assume the same charging rate for all EVs, the total power consumption in a decision timeslot is proportional to the number of EVs being charged. Hence, the first part of the cost for a transition $(s, \mathbf{u}_s, s')$ is:

$$C^{\text{demand}}(\mathbf{x}_s, \mathbf{u}_s) = \left(\sum_{d=0}^{S_{max}-1} \mathbf{x}_s^{\text{total}}(d)\, \mathbf{u}_s(d) \right)^2. \tag{3}$$

The second term of the cost function is a penalty proportional to the unfinished charging in the next state $s' = (t_{s'}, \mathbf{x}_{s'})$ due to taking action $\mathbf{u}_s$ in $s = (t, \mathbf{x}_s)$ and is defined as

$$C^{\text{penalty}}(\mathbf{x}_s, \mathbf{u}_s) = M \sum_{n \in \mathcal{V}_{t+1}} |\min(0, \Delta t_n^{\text{charge}} - \Delta t_n^{\text{depart}})| \tag{4}$$

The summation in (4) counts the amount of charging requests that are impossible to complete (for EVs with $\Delta t_n^{\text{depart}} < \Delta t_n^{charge}$) as a consequence of taking action $\mathbf{u}_s$ at state $s = (t, \mathbf{x}_s)$. M is a constant penalty factor, which we set to be greater than $2N_{\max}$ to ensure that any EV's charging is always completed before departing: having even just one incompletely charged EV will be costlier than simultaneously charging all EVs (see Appendix). This is the only user constraint we include in our formulations.

Note that in Eq. (2) the cost is independent of the timeslot variable t of the state and only depends on the aggregate demand variable $\mathbf{x}_s$: the quadratic function of total consumption (to achieve load flattening) indeed is time-independent. Still, we include time in the state definition $s = (t, \mathbf{x}_s)$ to enable easy extensions to other objectives (e.g., minimal cost under the time-varying pricing schemes). Also, we use the time component for the function approximator in Algorithm 1 (see Section V-B).

D. System Dynamics

In the MDP framework, system dynamics (via the environment) are defined using transition probabilities $P(s'|s, \mathbf{u}_s)$ for going from one state s to the next s'. These probabilities are unknown in our EV charging problem due to the stochasticity of EV arrivals and their charging demands. Perfect knowledge of EV arrivals (and their charging demands) within the control horizon would translate the problem into a decision tree as depicted in Fig. 2(c), where the cost of taking each action could be determined recursively using dynamic programming. In absence of such knowledge, the transition probabilities need to be estimated through interactions with the environment by taking actions and observing the instantaneous cost of the resulting state transitions, as explained next.

E. Learning Objective: State-Action Value Function

The learning objective is to find an optimum control policy $\pi^* : \mathbf{S} \rightarrow \mathbf{U}$ that minimizes the expected T-step return for any state in $\mathbf{S}$. The expected T-step return starting from s at a given t and following policy π (i.e., $\mathbf{u}_s = \pi(s)$) is:

$$J_T^{\pi}(s) = \mathbb{E}\left[\sum_{i=t}^{t+T} C(\underbrace{(t, \mathbf{x}_s)}_{s}, \mathbf{u}_s, \underbrace{(t+1, \mathbf{x}_{s'})}_{s'}) \right]. \tag{5}$$

The policy π is commonly characterized using a state-action value function, named Q-function:

$$Q^{\pi}(s, \mathbf{u}_s) = \mathbb{E}\left[C(s, \mathbf{u}_s, s') + J_T^{\pi}(s') \right], \tag{6}$$

where $Q^{\pi}(s, \mathbf{u}_s)$ is the cumulative return starting from state s, taking action $\mathbf{u}_s$, and following policy π afterwards. The optimal $Q^{\pi}(s, \mathbf{u}_s)$, denoted as $Q^*(s, \mathbf{u}_s)$, corresponds to $\min_{\pi} Q^{\pi}(s, \mathbf{u}_s)$ and satisfies the Bellman equation:

$$Q^*(s, \mathbf{u}_s) = \min_{\mathbf{u} \in \mathbf{U}} \mathbb{E}\left[C(s, \mathbf{u}_s, s') + Q^*(s', \mathbf{u}) \right]. \tag{7}$$

Solving (7) requires knowledge of the transition probabilities $P(s'|s, \mathbf{u}_s)$, which are unknown in our setting. Hence, a learning algorithm is used to obtain approximation $\widehat{Q}^*(s, \mathbf{u})$. Then taking control action $\mathbf{u}_s$ follows an optimal policy π^*:

$$\mathbf{u}_s = \pi^*(s) \in \underset{u \in \mathbf{U}_s}{\operatorname{argmin}}\, \widehat{Q}^*(s, \mathbf{u}). \tag{8}$$

[4] We assume only feasible requests are presented to the system, i.e., $\Delta t^{\text{charge}} \leq \Delta t^{\text{depart}}$ for each EV.

Algorithm 1: Fitted Q-iteration using function approximation for estimating the T-step return

Input : $\mathcal{F} = \{(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))|s = 1, \ldots, |\mathcal{F}|\}$;
1 Initialize $\widehat{Q}_0$ to be zero everywhere on $\mathbf{X} \times \mathbf{U}$;
2 **foreach** $n = 1, \ldots, T$ **do**
3 **foreach** $(s, \boldsymbol{u}_s, s', C(s, \boldsymbol{u}_s, s'))) \in \mathcal{F}$ **do**
4 $Q_n(s, \mathbf{u}_s) \leftarrow C(s, \mathbf{u}_s, s') + \min_{\mathbf{u}_{s'} \in \mathbf{U}_{s'}} \widehat{Q}_{n-1}(s', \mathbf{u}_{s'})$
5 Use function approximator to obtain $\widehat{Q}_n$ from $\mathcal{T}_{reg} = \left\{((s, \mathbf{u}_s), Q_n(s, \mathbf{u}_s))|s = 1, \ldots, |\mathcal{F}|\right\}$
6 **return** $\widehat{Q}_T$

IV. Batch Reinforcement Learning

We adopt batch mode RL algorithms to approximate $\widehat{Q}^*(s, \mathbf{u})$ from past experience instead of online interactions with the environment. The adopted approach is so-called off-policy value-iteration algorithm. In such algorithms, the optimum policy is learned from experiences gathered by a non-optimum (e.g., random) policy. Hence, we use historical EV data in terms of arrival/departures and energy demands to generate experiences by following a random policy (i.e., randomly taking an action from a set of possible actions in a particular state). Experiences are collected in the form of state transitions from s to s' when taking action $\mathbf{u}_s$, with associated costs $C(s, \mathbf{u}_s, s')$. We then use fitted Q-iteration to approximate $\widehat{Q}^*(s, \mathbf{u})$ from the samples, detailed next.[5]

A. Fitted Q-iteration

The fitted Q-iteration (FQI) algorithm, listed in Algorithm 1, takes as input a set of past experiences, $\mathcal{F}$, in the form of tuples $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))$ where $C(s, \mathbf{u}_s, s')$ is the immediate cost of a transition, calculated using Eq. (2). The tuples are used to iteratively estimate the optimum action value function. The state-action value function Q is initialized with zeros on the state-action space (Line 1) hence, $Q_1(s, \mathbf{u}_s) = C(s, \mathbf{u}_s, s')$ in the first iteration. In subsequent iterations, Q_n is calculated for each tuple in $\mathcal{F}$ by summing the immediate cost with the cost of taking the best action that can be taken from the next state (Line 4). The 'best' action is indeed the one that minimizes the Q-function from that next state s', which is estimated as $\widehat{Q}_{n-1}(s', \mathbf{u}'_s)$, i.e., the latest approximation of the action-value function obtained in the previous iteration. The set comprising all those $Q_n(s, \mathbf{u}_s)$ is then used to form a labeled dataset $\mathcal{T}_{reg}$. Based on this dataset, regression provides an updated estimate $\widehat{Q}_n$ for all state-action pairs (Line 5).

For the function approximation, we adopt a fully connected artificial neural network (ANN). Details on the ANN architecture used in our experiments follow in Section V-B2.

[5]Note that instead of randomly taking actions, it is possible to go back to the environment with the learned policy so far, gather more experience, and retrain the learning agent to improve performance. Investigation of such more efficient exploration strategies and thus (re)learning the policy is left for future work (see Section VII, future work item (1)).

B. The size of the state-action space

The input for FQI (i.e., set $\mathcal{F}$) is constructed from past interactions with the environment: randomly or deterministically taking actions from the action space of state $s = (t, \mathbf{x}_s)$ and recording the tuple $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s')))$. Note that the number of possible actions from a given state s is

$$|\mathbf{U}_s| = \prod_{d=1}^{S_{max}} \left(\mathbf{x}_s^{\text{total}}(d) + 1\right), \tag{9}$$

since for each flexibility $\Delta t^{\text{flex}} = d$ we can choose to charge between $[0, \mathbf{x}_s^{\text{total}}(d)]$ cars.

The goal of the RL algorithm (hence the goal of the FQI) is to estimate the T-step return for every possible action from every possible state in the environment. Estimating the T-step return starting from a state s leads to exploring a tree with an exponentially growing number of branches at the next steps: while the state and action representations are independent of the group size (N_{max}), the state-action space still grows exponentially with a growth rate given by Eq. (9).

For example, consider $N_{\text{max}} = 50$ charging stations and a control horizon of $S_{\text{max}} = 10$ timeslots. In a state where all EV charging stations are occupied ($N_s = N_{\text{max}} = 50$), there are at least 51 possible actions when all EVs have the same flexibility, and thus are on the same diagonal in the state matrix (i.e., $\mathbf{x}_s^{\text{total}} = [50, 0, 0, 0, 0, 0, 0, 0, 0, 0]$). On the other hand, if they are scattered across different Δt^{flex}, e.g., $\mathbf{x}_s^{\text{total}} = [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]$, there are $|\mathbf{U}_s| = 6^{10}$ possible actions.

This implies it is infeasible to include the entire state-action space in $\mathcal{F}$ as input for FQI. We thus only provide a subset of the state-action space, randomly sampling trajectories from the decision tree (that has branching factor $|\mathbf{U}_s|$). This leads to question **Q1** (answered in Section VI-A): How many sample trajectories from the state-action space suffice to learn an optimum policy for charging a real-world group of EVs?

The computational complexity of the FQI algorithm (and value iteration algorithms in general) depends on the calculation of the $\min$ operation (Line 4 of Algorithm 1). The $\min$ is taken over all possible actions from state s, i.e., the set $\mathbf{U}_s$, whose size depends on the arrangement of the EVs on the diagonals of matrix $\mathbf{x}_s$ as demonstrated above and quantified by Eq. (9). However, the overall computational complexity of the FQI algorithm is not explicitly quantifiable and is influenced by the states encountered (as part of the transitions in $\mathcal{F}$) during the training process.

V. Experiment setup

A. Data Preparation

Our analysis uses real-world EV charging session transactions collected by ElaadNL since 2011 from 2500+ public charging stations deployed across Netherlands, as described and analyzed in [16]. For each of the over 2M charging sessions, a transaction records the charging station ID, arrival time, departure time, requested energy and charging rate during the session. The EVs are privately owned and thus comprise a mixture of different and a priori unknown car types.

To represent the EV transactions in ElaadNL as state transitions $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))$, we first need to choose a

reasonable size for the state and the action representations. We set the maximum connection duration to $H_{\max} = 24\,\text{h}$, since more than 98% of the EV transactions in the ElaadNL dataset cover sessions of less than 24 hours [16]. We further set the decision timeslot, i.e., the time granularity of control actions, to $\Delta t^{\text{slot}} = 2\,\text{h}$, resulting in $S_{\max} = H_{\max}/\Delta t^{\text{slot}} = 12$. Hence, a state s is represented by a scalar variable t and a matrix $\mathbf{x}_s$ of size $S_{\max} \times S_{\max} = 12 \times 12$. The corresponding action $\mathbf{u}_s$ taken from state s is a vector of length 12 (with a decision for each of the upper diagonals, one per flexibility window Δt^{flex}). The motivation of choosing $\Delta t^{\text{slot}} = 2\,\text{h}$ is to limit the branching factor $|\mathbf{U}_s|$ (which depends on $S_{\max}$ in Eq. (9)) from each state. This yields a manageable state-action space size and allows model training (specifically, the $\min$ operation in Line 4 of Algorithm 1) in a reasonable amount of time.[6]

Furthermore, we make the ElaadNL dataset *episodic* by assuming that all the EVs leave the charging stations before the end of a day, thus yielding an empty car park in between two consecutive days.[7] We define such an episodic 'day' to start at 7 am and end 24 h later (the day after at 7 am) and refer to it simply as a 'day' in the rest of the paper. The empty system state in between two days is always reached after $S_{\max} + 1$ timeslots and represented by an aggregate demand matrix $\mathbf{x}_s$ of all zeros. This ensures that, while each day can have a different starting state (depending on the first arrivals and their energy demand), traversing the decision tree always leads to a unique terminal state (see Fig. 2(c) for an exemplary decision tree). This is motivated by Riedmiller [18], showing that, when learning with FQI and adopting a neural network as function approximator, having a terminal goal state stabilizes the learning process. It ensures that all trajectories end up in a state where no further action/transition is possible and hence is characterized by an action-value of zero.

To create a group of $N_{\max}$ EV charging stations, we select the busiest $N_{\max}$ ones based on the number of recorded transactions per station. We use two subsets, one with the top-10, the other with the top-50 most busy stations. We use the actual session data of these stations in terms of car arrivals, departures and charging needs:[8] this determines the possible actions that can be taken at any time, which will be used to construct the set of experiences $\mathcal{F}$, as described in the RL algorithm settings described next.

B. Algorithm Settings

Since we consider day episodes of 24 h, and use timeslots of 2 h, we have $S_{max} = 12$. Hence, fitted Q-iteration (FQI) needs to estimate the 12-step return and we have 12 iterations in Algorithm 1.

1) Creating set $\mathcal{F}$: To collect the set $\mathcal{F}$, we begin from the starting state of a day characterized as $(t_1, \mathbf{x}_1)$. We then randomly choose an action from all possible ones in that state, observe the next state, note the associated state transition cost, and repeat until we reach the terminal state.[9] The state transitions in each trajectory are thus recorded in $\mathcal{F}$ as tuples $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))$. We randomly sample multiple possible trajectories from each day and analyze the effect of the number of sampled trajectories on the performance of the resulting policy. The notion of a *sample* in the following thus refers to a full trajectory from initial to terminal state of a day.

2) Neural network architecture: We use an artificial neural network (ANN) comprising an input layer, 2 hidden layers (of 128 and 64 neurons respectively) with ReLU activation function, and an output layer. Since the ANN is used for regression, the output layer has a single neuron and a linear activation function. Each state-action pair is fed to the input layer in form of a vector of length $S_{\max}^2 + S_{\max} + 1$, by reshaping the state s and concatenating it with the action vector $\mathbf{u}_s$ (of size $S_{\max} = 12$). The state s comprises a scalar time variable t and an aggregate demand matrix $\mathbf{x}_s$ of size $S_{\max} \times S_{\max}$, which we reshape to a vector of size $S_{\max}^2 + 1$ ($= 145$ in our case). Inspired by Mnih *et al.* [19], we also found that using Huber loss [20] instead of mean-squared-error stabilizes learning.

C. Performance Evaluation

We use the ElaadNL transactions of 2015 and select the last 3 months as the test set for evaluation, i.e., $\mathcal{B}^{\text{test}} = \{e_i | i = 274, \ldots, 365\}$ containing $|\mathcal{B}^{\text{test}}| = 92$ days.[10] Each day e_i in the test set contains transactions in form of tuples $(t^{\text{arrival}}, \Delta t^{\text{depart}}, \Delta t^{\text{charge}})$ where t^{arrival} is the arrival time of the connecting EVs. We consider training sets of varying lengths (to determine the impact of training set size, see research question Q1), with training time spans of $\Delta t \in \{1, 3, 5, 7, 9\}$ months. For a given Δt, we randomly pick 5 contiguous periods within the range of Jan. 1, 2015 until Sep. 30, 2015 (except for $\Delta t = 9$ months, since that covers all training data). We define the training set for time span Δt and run j $(= 1 \ldots 5)$ as $\mathcal{B}^{\text{train}}_{\Delta t, j} = \{e_i | i = e^{\text{start}}_{\Delta t, j}, \ldots, e^{\text{start}}_{\Delta t, j} + \Delta t - 1\}$, where $e^{\text{start}}_{\Delta t, j}$ is the randomly selected starting date of the training set.[11]

To evaluate the performance of the learned policy, we define the metric of *normalized cost* relative to the cost C_{opt} of the optimum policy from an all-knowing oracle. This oracle policy is found by following an MPC approach minimizing a quadratic objective function, namely the sum of the squared loads over a horizon of $H_{\max}$ timeslots. For each Δt and j we thus define the *normalized cost* as

$$C_{\pi(\Delta t, j)} = \frac{1}{|\mathcal{B}^{\text{test}}|} \sum_{e \in \mathcal{B}^{\text{test}}} \frac{C^e_{\pi(\Delta t, j)}}{C^e_{\text{opt}}}, \tag{10}$$

where $\pi(\Delta t, j)$ is a policy learned from run j for training data time span Δt. Further, $C^e_{\pi(\Delta t, j)}$ is the cost for day e under

[6] We use an Intel Xeon E5645 processor, 2.4 GHz, 290 GB RAM.

[7] The charging demands of EVs are adjusted to ensure the requested charging can be fulfilled within 24 hours.

[8] Note that in the real-life ElaadNL sessions, no smart charging was applied. But from the charging needs we can infer flexibility in terms of potential delayed charging, and thus all *possible* actions.

[9] Recall that we consider an episodic setting, i.e., case where the system empties (definitely after $S_{\max}$ timeslots).

[10] Recall from Section V-A that each day in the set starts at 7 am and ends 24 h later, with an empty system.

[11] Note that the training set $\mathcal{B}^{\text{train}}$ is different from set $\mathcal{F}$. Similar to the test set, each day in the training set also contains tuples $(t^{\text{arrival}}, \Delta t^{\text{depart}}, \Delta t^{\text{charge}})$. The control problem for each day of the training set translates into a control tree (see Section III for MDP construction from tuples in training set) and is randomly sampled using a methodology explained in Section V-B1 to obtain and include the tuples $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))$ in set $\mathcal{F}$.

policy $\pi(\Delta t, j)$. C^e_{opt} is the optimum cost for day e, found by solving the load flattening problem as a quadratic optimization problem. The cost for day e under policy π is calculated by summing the instantaneous cost (defined by Eq. (2)) of state transitions encountered when taking actions according to the policy under evaluation (using Eq. (8)). Clearly, if a learned policy achieves the optimum policy, then $C_{\pi(\Delta t,j)} = 1$.

We compare the performance of the learned policy not only with the optimum policy but also with the business-as-usual (BAU) policy where the charging of an EV starts immediately upon arrival. In the next section, we present our analysis using the normalized cost of BAU, optimum and learned policies denoted as C_{BAU}, C_{opt} and C_{RL} respectively.

VI. Experimental results

A. Learning the Charging Coordination (Q1–Q2)

To answer **Q1** we study the learned policy's performance in function of (i) the time span covered by the training data (i.e., Δt), and (ii) the number of sample trajectories per day of training data. Figure 3 compares the normalized cost of a learned policy with that of a BAU and optimum policy for varying Δt and number of samples per training day, for the case of $N_{\text{max}} = 10$ and 50 charging stations respectively.

– *Influence of the time span covered by the training set:* Fig. 3(b) shows that increasing Δt from 1 to 3 months and beyond reduces the normalized cost of the learned policy for both 10 and 50 charging stations. Additionally, the performance gain when increasing Δt from 1 to 3 months is bigger than for raising Δt beyond 3 months. This suggests that the RL approach needs at least 3 months of training data to reach maximal performance (in case of ElaadNL).

– *Influence of the number of sample trajectories:* Fig. 3(a) shows that when $\Delta t \geq 3$ months, increasing the number of samples does not result in a significant cost reduction for the learned policy (i.e., C_{RL}) for either 10 or 50 charging stations. Our analysis suggests that we need a training data time span of at least 3 months to have a comparable performance over various number of samples per day and that when training data time span is at least 3 months, fewer samples (~5K trajectories) can still achieve a comparable performance (with respect to training with larger samples per day). This answers **Q1**.

Next, we answer **Q2** (i.e., how does the RL policy perform compare to an optimal all-knowing oracle algorithm?) by referring to the best performance measures in Fig. 3. We observe that the best performance is achieved when $\Delta t = 9$ months for both $N_{\text{max}} = 10$ and 50. The relative improvement in terms of normalized cost reduction, compared to a business-as-usual uncontrolled charging scenario, C_{BAU}, amounts to 37% and 29.4% for 10 and 50 charging stations respectively. Note that C_{RL} is still 7% and 8.5% more expensive than the optimal policy cost C_{opt} for 10 and 50 charging stations respectively. Still, it is important to realize that for the optimal policy we assume perfect knowledge of future EV charging sessions, including arrival and departure times and energy requirements. Clearly, having that future knowledge is infeasible in reality: the proposed RL approach, which does not require such knowledge, thus is a more practical solution.

Finally, we note that there is an increase in the variance between simulation runs for larger N_{max} (see shaded regions in Fig. 3). Analysis revealed that there are no notable differences in the distributions of arrival and departure times, nor energy demands, among the $N_{\text{max}} = 10$ vs. 50 cases. We rather hypothesize that the increased variance in performance is because of the considerably larger state-action space for bigger N_{max}, given in Eq. (9). The performance of FQI is indeed greatly influenced by the given training set $\mathcal{F}$. Since we use random sampling, essential parts of the state-action space (e.g., best and worst trajectories) will not necessarily be included in $\mathcal{F}$. With larger trees, that possibility is even more limited. We leave techniques to achieve more efficient exploration of this large state-action space as future work, and for now refer to [21] for an overview of exploration algorithms.[12]

B. Variance of performance over time (Q3)

In Fig. 3, the test set was fixed: the performance was evaluated using the last 3 months of 2015 from the ElaadNL data as test set (i.e., $\mathcal{B}^{\text{test}} = \{e_i | e_i = 274, \ldots, 365\}$). Now, to analyze how performance of our RL approach would vary throughout the year, we consider each month of 2015 as a separate test set (e.g., $\mathcal{B}^{\text{test}}_{\text{Jan}} = \{e_i | i = 1, \ldots, 31\}$). We therefore no longer fix the training set to a single given time span, but for each test month use the immediately preceding Δt months as training data (with $\Delta t = 1$ or 5 months). This analysis investigates whether learning the EV charging characteristics is more challenging for different times of the year, hence answering Q3.

Figure 4 shows normalized costs for coordinating $N_{\text{max}} = 10$ charging stations (top) and cost improvement compared to the business-as-usual scenario, C_{BAU} (bottom).

Figure 4 shows that C_{BAU} varies across test months: for some (e.g., May and Aug), the difference $C_{\text{BAU}} - C_{\text{opt}}$ is larger than for others. This indicates that the charging sessions in these months have higher flexibility, which is exploited by the optimum solution. For such months with higher $C_{\text{BAU}} - C_{\text{opt}}$, our proposed RL approach also achieves a higher cost reduction compared to C_{BAU} (Fig. 4, bottom). Still, the achieved C_{RL} is more expensive than C_{opt} compared to months that offer less flexibility. We found that days on which the optimal charging pattern requires the exploitation of larger charging delays are more challenging to learn by RL approach, in the sense that RL has greater difficulty in approaching the optimum (i.e., obtaining higher C_{RL}). One reason is scarcity of such days in the training set, resulting in imbalanced training data. Another reason is that random sampling does not guarantee inclusion of the scarce (but crucial) parts of the large state-action space in the training set as fed to FQI. We further investigate the effect of increasing the training data Δt from 1 to 5 preceding months for each test set. For most months, this results in improvement with respect to C_{BAU} (Fig. 4, bottom).

The analysis in this section thus answers **Q3** (i.e., How does the performance vary over time using realistic data?): the

[12] As indicated previously, we limit this paper's focus to proposing the (scalable/generalizable) MDP formulation and experimentally exploring the resulting RL-based charging performance using real-world EV data.

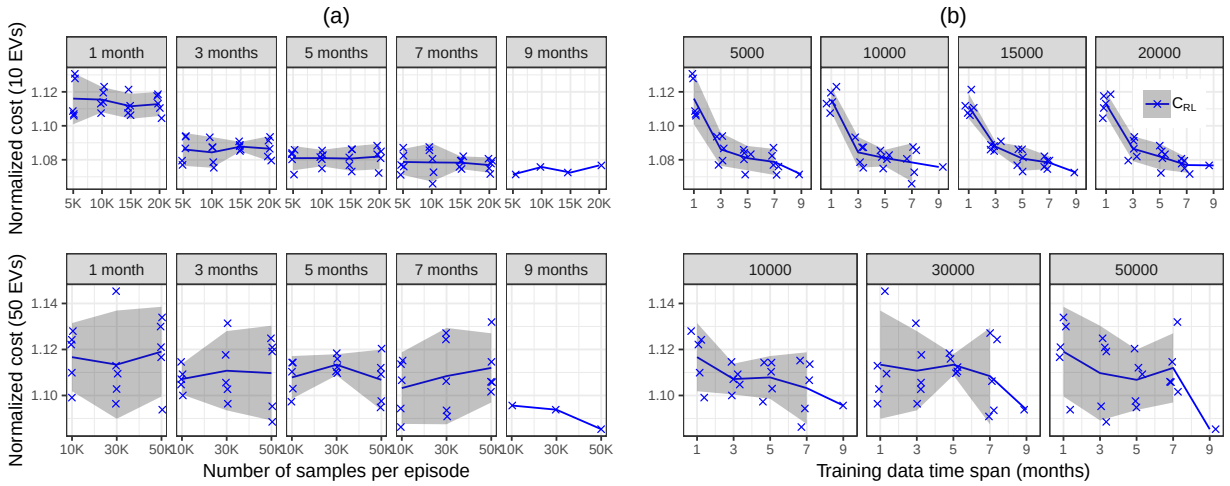

Fig. 3: Normalized costs for the learned policy (C_{RL}) for 10 (top) and 50 (bottom) charging stations, as a function of (a) the number of sample trajectories per training day for various Δts, and (b) Δt for various numbers of samples. BAU policy cost is $C_{\text{BAU}} = 1.43$ and 1.38 respectively for 10 and 50 stations. Obviously, $C_{\text{opt}} = 1$. Markers are individual runs, lines are run averages and ribbons indicate 95% confidence intervals.

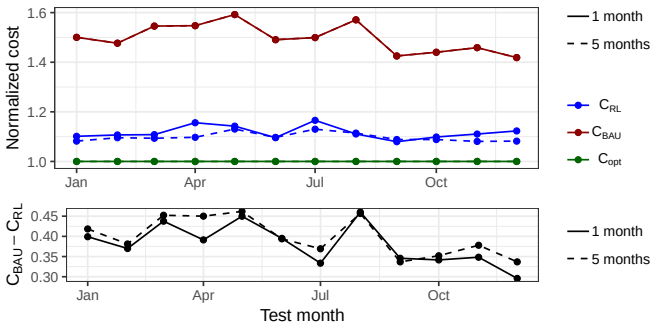

Fig. 4: Performance using different months as test set and different time spans of the training set (1–5 months).

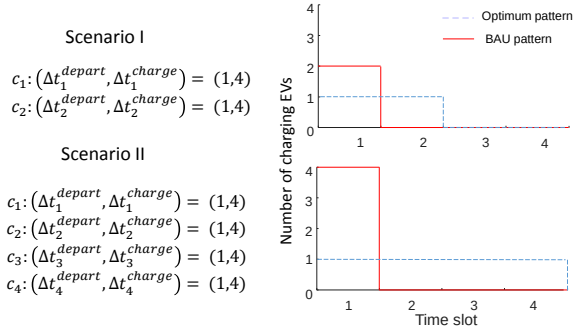

Fig. 5: Effect of scaling up the group size on the normalized cost for a policy learned from 10 charging stations.

RL algorithm performance depends on the available flexibility, with greater flexibility (expectedly) leading to larger cost reductions compared to the BAU uncontrolled charging, but greater difficulty in approaching the optimum performance.

C. Generalization to Larger Scales (Q4)

While model-free approaches based on RL eliminate the need for accurate knowledge of the future EV session characteristics (as opposed to optimization approaches), they still require a reasonably long training time to be able to efficiently coordinate the EV charging sessions. The runtime for the largest training set size (covering 9 months, with 5K sample trajectories per day) is approximately 3 h for $N_{\max} = 10$ charging stations, while for $N_{\max} = 50$ it is about 48 h.[13]

Since our proposed formulations are independent of the number of charging stations ($N_{\max}$), we can investigate how a policy learned from training with a small number of charging stations performs when applied to coordinating a larger group. We use the policy learned from data of $N_{\max} = 10$ charging stations with $\Delta t = 9$ months, with the EV sessions of the last quarter of 2015 as test set. To investigate the effect of scaling up the number of charging stations without changing other system characteristics, we replicate EV charging sessions with a factor *scale* to create a test set of larger $N_{\max}$. Such scaling may change the optimum solution, as illustrated with a simple example in Fig. 5 where the length of the control horizon is $S_{\max} = 4$ slots. In Scenario I of Fig. 5, at time $t = 1$ we have 2 connecting cars: $V = \{(\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}) = (1,4), (\Delta t_2^{\text{depart}}, \Delta t_2^{\text{charge}}) = (1,4)\}$ and no other arrivals during the control horizon. The best action is to charge 50% of the cars at $t = 1$ and 2 to flatten the load curve. In Scenario II of Fig. 5, set V is duplicated once and the best action now is to charge 25% of the cars in each of the control timeslots.

The normalized costs (relative to C_{opt}) of the learned policy for scaled-up group sizes are shown in Fig. 6 for various scales and number of samples per day in the training set. A scale of 1 corresponds to the original test set (no replication). The largest jumps in C_{RL} are observed for doubling the group size (scale factor $2\times$). Further increases of $N_{\max}$ beyond $2\times$ only

[13] Running on an Intel Xeon E5645 processor, 2.4 GHz, 290 GB RAM.

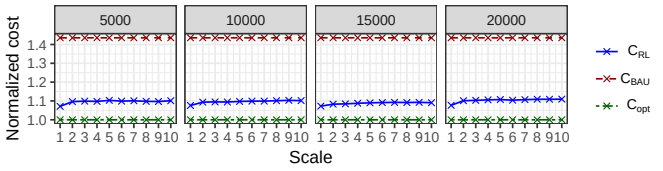

Fig. 6: The effect of scaling up $N_{\max}$ on a normalized cost of a policy learned from $N_{\max} = 10$ charging stations for different numbers of sampled trajectories (ranging from 5K to 20K).

lead to marginal increases in normalized cost for any number of sample trajectories per day. These analyses confirm that our proposed MDP formulations are generalizable to varying group sizes and that a policy learned from a smaller group of charging stations can be used to coordinate the charging of a larger group, at least provided that the distribution of EV arrivals, departures and energy demands are similar.

VII. Conclusion

In this paper, we proposed a reinforcement learning approach for jointly controlling a group of EV charging stations. We formulated an MDP with scalable representation of an aggregated state that effectively takes into account individual EV charging characteristics (i.e., arrival time, charging and connection duration). Our formulations are independent of the number of charging stations and charging rates and hence generalize to a varying number of charging stations. We used a 1-year long real-world EV charging dataset [16] to experimentally evaluate the performance of the proposed approach compared to an uncontrolled business-as-usual (BAU) policy, as well as an optimum solution that has a perfect knowledge of the EV charging session characteristics (in terms of arrival and departure times). In our case study, the analyses **concluded** the following:[14]

(1) While the representation of the state and action are independent of the number of charging stations, because of its sheer size it is still infeasible to feed the entire state-action space to the FQI learning algorithm. This raised question **Q1**: What are appropriate training data time spans and number of sampled trajectories from the decision trees? We found that when training data spans more than 3 months, the performance of a policy does not improve when trained with more than a relatively small number ($\sim$5K) of sampled trajectories per training day.

(2) We investigated how the RL policy performs compared to an optimal all-knowing oracle algorithm (**Q2**). We show that our approach learns a policy that can reduce the cost of coordinating charging across 10 and 50 charging stations by 37% and 29.4% respectively compared to an uncontrolled BAU charging policy. The achieved cost reduction in our approach does not require future knowledge on EV charging sessions and is only 7% (for $N_{\max}$ =10 charging stations) and 8.5% (for $N_{\max}$ =50) more expensive than an optimum solution that assumes perfect knowledge of future EV charging demand.

(3) We then analyzed how the performance of our RL approach varies over time using realistic data (i.e., **Q3**) by checking whether the learned policy performs similarly for various months of the year (when training on the preceding months). The results indicate that flexibility, and hence cost reduction, varies across various months: months with larger flexibility exhibit larger cost reduction achieved by the learned policy, compared to the cost of the BAU policy. Still, the cost gap between the learned policy and the optimal one is larger for those higher flexibility months. This is due to (a) scarcity of days with larger flexibility in the training set, and (b) random sampling of the state-action space, which does not guarantee inclusion of the rare but crucial parts of the state-action space in the training set as fed to the FQI algorithm.

(4) Finally, we trained an agent using an experience from 10 charging stations and applied the learned policy to control a higher number of stations (up to a factor of $10\times$ more arrivals) to check generalization of the learned policy (question **Q4**). These analyses confirmed that our proposed MDP formulations are generalizable to groups of varying sizes and that a policy learned from a small number of charging stations may be used to coordinate charging in a larger group, at least for similar distributions of EV arrivals, departures and energy demands.

Our **future research** will look into possible improvements:

(1) We used random exploration of state-action space to collect the experience as an input to our learning algorithm. We will investigate whether incorporating an efficient exploration strategies to perform a more informed sampling of the state-action space improves the performance.

(2) We used a fully connected neural network for function approximation in the FQI algorithm. Since we represent our aggregate state of demand using a matrix, it is relevant to investigate whether using convolutional neural networks (similar to the function approximation adopted in [17]) will further improve performance.

(3) We represented the aggregate demand in each state as a matrix, i.e., using a 2D grid with one axis being Δt^{depart}, the other Δt^{charge}. This approach efficiently represents the aggregate demand while retaining individual EV charging characteristics. However, the discretization during the binning process introduces an approximation error for the cost function. The error can be minimized by refining the time granularity (i.e., using more precise Δt^{depart} and Δt^{charge} quantization) in the aggregate demand representation of a state. Still, refining the time granularity results in a larger state space, affecting the scalability and the learning speed of the proposed approach. Hence, as a next step, we will analyze how this approximation error may influence the optimization result at the aggregated EV level. We will also investigate the possibility of identifying an optimum time granularity that results in an acceptable approximation error at the aggregated level without jeopardizing scalability and learning speed of the proposed approach.

[14]Note that the quantitative statements made are clearly only valid under the current experiment settings.

(4) Note that the adopted RL approach essentially learns from experience, implying that it gathers some knowledge on, e.g., typical arrival and departure patterns of EVs, but does not explicitly models those. One could also explore explicit forecasting of future EV sessions, and use the timings of forecasted EV charging sessions as additional inputs to the decision system. We leave systems exploiting such EV forecasting as future work.

(5) The learning algorithm in our RL approach is based on value iteration, where the state-action value is estimated for various state-action pairs and an optimum policy is deduced form these estimations. We will investigate whether using policy iteration methods improves performance. The use of policy iteration will also allow taking continuous actions, by avoiding a minimization problem as in eq. (8), instead of the discrete ones.

(6) A real-world deployment is the ultimate goal of any DR algorithm. Our proposed RL based approach already circumvents various unrealistic assumptions by eliminating the need for models and learning only from data. Yet, follow-up work is required to further validate our solution's applicability: (a) validating our proof-of-concept with simulations covering broader ranges of parameter settings (e.g., use of finer decision timeslots than the currently used coarse 2 h long timeslots), (b) taking into account a more comprehensive set of user constraints in proposed MDP and the use of heuristics to maximize user acceptance, e.g., based on the previous analysis of the user data [16], (c) improving safety and interoperability of the proposed RL algorithm [22], and finally (d) conducting real-word operational tests.

APPENDIX

This appendix elaborates on setting the value of the penalty factor M in Eq. (4). The objective of defining that penalty cost is to ensure that even one incomplete unit of charging is costlier that simultaneously charging all EVs. Let us assume an instant where joint charging of the present EVs is at its maximum capacity: the cost of such simultaneous charging (without any delay) is N_{max}^2. Now, if only one unit of charging is delayed beyond the remaining sojourn of the EV involved, the total cost amounts to $(N_{\text{max}} - 1)^2 + M$. To achieve the aforementioned goal we need $N_{\text{max}}^2 < (N_{\text{max}} - 1)^2 + M$, hence $N_{\text{max}}^2 < N_{\text{max}}^2 - 2N_{\text{max}} + 1 + M$, thus $M > 2N_{\text{max}} - 1$.

ACKNOWLEDGMENTS

The work presented in this paper was supported in part by the Flemish Government, through the VLAIO SBO project MultI-agent LEarnIng for neTworks (SMILE-IT), grant number 140047. We thank prof. Pascal Poupart for advice on the reinforcement learning algorithms, and dr. Bert Claessens for constructive feedback. The car icon in Fig. 1 was designed by Freepik (http://www.freepik.com).

REFERENCES

[1] A. Afram and F. Janabi-Sharifi, "Theory and applications of HVAC control systems – A review of model predictive control (MPC)," *Build. Environ.*, vol. 72, pp. 343–355, 2014.

[2] J. Ma, J. Qin, T. Salsbury, and P. Xu, "Demand reduction in building energy systems based on economic model predictive control," *Chem. Eng. Sci.*, vol. 67, no. 1, pp. 92–100, 2012.

[3] N. Sadeghianpourhamami, T. Demeester, D. Benoit, M. Strobbe, and C. Develder, "Modeling and analysis of residential flexibility: Timing of white good usage," *Appl. Energ.*, vol. 179, pp. 790–805, 2016.

[4] B. J. Claessens, S. Vandael, F. Ruelens, K. D. Craemer, and B. Beusen, "Peak shaving of a heterogeneous cluster of residential flexibility carriers using reinforcement learning," in *Proc. IEEE PES Innovative Smart Grid Technol. Eur. (ISGT Europe 2013)*, Copenhagen, Denmark, 6–9 Oct. 2013, pp. 1–5.

[5] S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck, "A scalable three-step approach for demand side management of plug-in hybrid vehicles," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 720–728, 2013.

[6] N. Chen, T. Q. S. Quek, and C. W. Tan, "Optimal charging of electric vehicles in smart grid: Characterization and valley-filling algorithms," in *Proc. 3rd IEEE Int. Conf. Smart Grid Commun. (SmartGridComm 2012)*, Tainan City, Taiwan, 5–8 Nov. 2012, pp. 13–18.

[7] L. Gan, U. Topcu, and S. H. Low, "Optimal decentralized protocol for electric vehicle charging," *IEEE Trans. Power Sys.*, vol. 28, no. 2, pp. 940–951, May 2013.

[8] N. Sadeghianpourhamami, J. Deleu, and C. Develder, "Achieving scalable model-free demand response in charging an electric vehicle fleet with reinforcement learning," in *Proc. 9th ACM Int. Conf. Future Energy Systems (e-Energy 2018)*, 12–15 Jun. 2018.

[9] M. Riedmiller, "Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method," in *Proc. 16th Eur. Conf. Machine Learning (ECML 2005)*, vol. 3720, Porto, Portugal, 3–7 Oct. 2005, pp. 317–328.

[10] J. Hu, H. Morais, T. Sousa, and M. Lind, "Electric vehicle fleet management in smart grids: A review of services, optimization and control aspects," *Renew. Sust. Energ. Rev.*, vol. 56, pp. 1207–1226, Apr. 2016.

[11] Z. Yang, K. Li, and A. Foley, "Computational scheduling methods for integrating plug-in electric vehicles with power systems: A review," *Renew. Sust. Energ. Rev.*, vol. 51, pp. 396–416, Nov. 2015.

[12] J. R. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Appl. Energ.*, vol. 235, pp. 1072–1089, Feb. 2019.

[13] W. Shi and V. W. S. Wong, "Real-time vehicle-to-grid control algorithm under price uncertainty," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm 2011)*, Brussels, Belgium, 17–20 Oct. 2011, pp. 261–266.

[14] A. Chis, J. Lundén, and V. Koivunen, "Reinforcement learning-based plug-in electric vehicle charging with forecasted price," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 3674–3684, May 2017.

[15] S. Vandael, B. Claessens, D. Ernst, T. Holvoet, and G. Deconinck, "Reinforcement learning of heuristic EV fleet charging in a day-ahead electricity market," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1795–1805, July 2015.

[16] N. Sadeghianpourhamami, N. Refa, M. Strobbe, and C. Develder, "Quantitive analysis of electric vehicle flexibility: A data-driven approach," *Int. J. Electr. Power Energy Syst.*, vol. 95, pp. 451–462, 2018.

[17] B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3259–3269, 2018.

[18] M. Riedmiller, "10 steps and some tricks to set up neural reinforcement controllers," in *Neural networks: Tricks of the trade, 2nd Ed.*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 2012, ch. 30, pp. 735–757.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[20] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, Mar. 1964.

[21] R. McFarlane, "A survey of exploration strategies in reinforcement learning," *McGill University*, 2018. [Online]. Available: http://www.cs.mcgill.ca/~cs526/roger.pdf

[22] M. Jin and J. Lavaei, "Stability-certified reinforcement learning: A control-theoretic perspective," *arXiv e-prints*, Oct. 2018, arXiv:1810.11505.

Nasrin Sadeghianpourhamami obtained the B. Eng. degree (First Class Hons.) in electronics, majoring in telecommunications, and the M.Sc. degree in engineering science from the Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia, in 2007 and 2011 respectively, and the Ph.D. degree in computer science from the Faculty of Engineering, Ghent University, Ghent, Belgium, in 2018. From 2011 to 2014, she was a Lecturer with the Faculty of Engineering, Multimedia University. In Jan. 2015, she joined the research group IDLab in the Department of Information Technology (INTEC) at Ghent University - imec, Ghent, Belgium, as a PhD student. As a part of the smart grid team in IDLab, she developed solutions using statistical modeling, machine learning, and deep reinforcement learning, to facilitate practical demand response algorithms for residential customers and electric vehicles in the smart power grid. Her Ph.D. research led to seven international publications.

Johannes Deleu received the degree of Master of Science in computer science engineering from Ghent University, Belgium, in 2005. He then joined the research group IDLab in the Deptartment of Information Technology (INTEC), Ghent University - imec. He is a senior research engineer working on topics of information retrieval and extraction, machine learning, and in particular deep learning applied to natural language processing (NLP). He has participated in multiple research projects. This has resulted in several real-world applications for automatic content enrichment, nowadays in commercial use by several players in the Flemish media sector.

Chris Develder (SM'99) is associate professor with the research group IDLab in the Dept. of Information Technology (INTEC) at Ghent University - imec, Ghent, Belgium. He received the M.Sc. degree in computer science engineering and a Ph.D. in electrical engineering from Ghent University, Ghent, Belgium, in 1999 and 2003 respectively, as a fellow of the Research Foundation (FWO). From January 2004 to August 2005, he worked for OPNET Technologies, on (optical) network design and planning. In September 2005, he re-joined INTEC as a postdoctoral researcher, with a fellowship of the FWO in 2006–2012. In October 2007 he obtained a part-time, and since February 2010 a fulltime professorship at Ghent University. He has stayed as a research visitor at UC Davis, CA, USA (Jul.-Oct. 2007) and at Columbia University, NY, USA (Jan. 2013 – Jun. 2015). He was and is involved in various national and European research projects (e.g., FP7 Increase, FP7 C-DAX, H2020 CPN).

Chris currently leads two research teams within IDLab, one on converting text to knowledge (i.e., NLP, mostly information extraction using machine learning), the other on data analytics and machine learning for smart grids. He also has a substantial track record in optical networking (dimensioning, modeling, optimization, especially for grid/cloud computing). He has co-authored over 200 refereed publications in international conferences and journals. He is *Senior Member* of IEEE, *Member* of ACM, and *Member* of ACL.