Optimized cost function for demand response coordination of multiple EV charging stations using reinforcement learning

Manu Lahariya
IDLab, Ghent University – imec
manu.lahariya@ugent.be

Nasrin Sadeghianpourhamami
IDLab, Ghent University – imec
nasrin.sadeghianpourhamami@ugent.be

Chris Develder
IDLab, Ghent University – imec
chris.develder@ugent.be

ABSTRACT

Electric vehicle (EV) charging stations represent a substantial load with significant flexibility. Exploitation of that flexibility in demand response (DR) algorithms becomes increasingly important to manage and balance demand and supply in power grids. Model-free DR based on reinforcement learning (RL) is an attractive approach to balance such EV charging load. We build on previous research on RL, based on a Markov decision process (MDP) to simultaneously coordinate multiple charging stations. However, we note that the computationally expensive cost function adopted in previous research leads to large training times, which limits the feasibility and practicality of the approach.

We therefore propose an improved cost function which essentially forces the learned control policy to always fulfill any charging demand that does not offer any flexibility. We rigorously compare the newly proposed batch RL fitted Q-iteration implementation with the original (costly) one, using real world data. Specifically, for the case of load flattening, we compare the two approaches in terms of (i) the processing time to learn the RL-based charging policy, as well as (ii) overall performance of the policy decisions in terms of meeting the target load for unseen test data. The performance is analyzed for different training periods and varying training sample sizes. In addition to both RL policies' performance results, we provide performance bounds in terms of both (i) an optimal all-knowing strategy, and (ii) a simple heuristic spreading individual EV charging uniformly over time.

CCS CONCEPTS

• **Computing methodologies** → *Intelligent agents.*

KEYWORDS

Smart grid, demand response, electric vehicle, smart charging, reinforcement learning, markov decision process

ACM Reference Format:
Manu Lahariya, Nasrin Sadeghianpourhamami, and Chris Develder. 2019. Optimized cost function for demand response coordination of multiple EV charging stations using reinforcement learning. In *BuildSys 2019: ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, November 13–14, 2019, New York, NY, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3360322.3360992

BuildSys 2019, November 13–14, 2019, New York, NY, USA

ACM ISBN 978-1-4503-7005-9/19/11…$15.00
https://doi.org/10.1145/3360322.3360992

1 INTRODUCTION

Demand response (DR) algorithms are pivotal to ensure demand-supply balance in smart grids with intermittent renewable energy resources and new loads (e.g., electric vehicles, EVs). In traditional approaches for coordinating EV charging [3], DR is cast as an optimization problem (e.g., model predictive control, MPC). However, this approach requires accurate models (e.g., of user behavior, energy demand, flexibility that is available to exploit) which have uncertainty associated with them. Furthermore, such approaches do not generalize from one scenario to the other.

Aforementioned challenges are tackled with recent data-driven DR algorithms, where the charging coordination problem is cast as a time-series decision making problem and is formulated using Markov decision process (MDP) with unknown system dynamics. Reinforcement learning (RL) is then used to estimate the optimum charging coordination policy (e.g., [2]). For a recent overview of RL in DR, we refer to [7]. In terms of objectives, different DR targets have been addressed, including (i) reducing electricity costs, (ii) maximizing profits for the provider, and (iii) load balancing in the grid. For example, Chis *et al.* [1] propose a reduction in long term cost for user for charging a single EV.

Previous research [4] formulated an MDP for a set of EV charging stations, aiming at a model-free DR approach for EV charging stations to exploit time flexibility provided by users. In [5], a refined MDP and experimental performance evaluation is provided, thus giving a proof-of-principle of adopting RL for jointly coordinating charging for a set of EVs. This approach can simultaneously control multiple EV charging at once, in contrast to [1], which optimizes charging for just a single EV. The MDP definition scales independently of the number of charging stations ($S_{\max}$) and number of maximum cars ($N_{\max}$), and thus can be easily deployed in multiple scenarios. While the learned RL policy's performance demonstrated effectiveness in terms of meeting the DR objective, it comes at the cost of requiring a large set of experiences (past data), long training periods and computational power.

This paper extends the previous work in [5] by improving the RL implementation: we reduce the computational complexity and dataset requirements in the MDP definition and RL training through (i) an updated cost function, and (ii) a reduced state-action space in MDP, resulting in a smaller exploration dataset. The rest of the paper presents the following contributions:

- We propose an updated MDP with a new cost function (§2.1);
- We train RL policies for both the original [5] and updated cost functions (using the algorithm summarized in §2.2);

- Simulation experiments (§3) to evaluate the policies and answer the following questions (§4):
  (Q1) What reduction of training time and computational complexity does the new cost function achieve?
  (Q2) How does varying the parameters of input training data impact the training time?
  (Q3) Does the updated cost function affect the resulting performance achieved by the RL policy?
  (Q4) How much of the offered flexibility does the RL policy use?

2 ALGORITHM

A Markov decision process (MDP) is defined by (i) a finite state space, (ii) an action space, (iii) a cost (or rewards) function for taking a particular action, given a state. The next subsections summarize (i) the MDP for jointly coordinate charging a set of EVs, and (ii) a batch reinforcement learning algorithm for training the policy.

2.1 Markov Decision Process (MDP)

State: To create a state for the set of connected EVs, at a given timeslot t, we assume to know for each EV (i) the time left until it departs (Δt^{depart}), and (ii) its charging requirement, which we quantify as the number of timeslots it needs to charge (Δt^{charge}), thus assuming the same constant charging rate for each EV. Hence, the state representation is of the form $s = (t, \mathbf{x}_s)$, where t is the timeslot (i.e., $t \in \{1, \ldots, S_{\max}\}$) and $\mathbf{x}_s$ is the aggregate demand of all EVs in the system. Formally, $\mathbf{x}_s$ is an $S_{\max} \times S_{\max}$ matrix, where $S_{\max}$ is the maximum timeslots in the horizon. The element of $\mathbf{x}_s$ at position (i, j) counts the fraction of EV charging stations that have a connected car in the corresponding $(\Delta t^{\text{depart}}, \Delta t^{\text{charge}})$ bin, i.e., for which $i = \Delta t^{\text{depart}}$ and $j = \Delta t^{\text{charge}}$.

Action: Per set of EVs that have a particular flexibility, the action dictates whether or not to charge them. We note that EVs with the same flexibility are positioned along the diagonals of the state matrix $\mathbf{x}_s$: the flexibility, defined as the amount of time shifting we can apply in the charging process, is indeed given by $\Delta t^{\text{flex}} = \Delta t^{\text{depart}} - \Delta t^{\text{charge}}$. We indicate the number of EVs on each diagonal of $\mathbf{x}_s$ as $\mathbf{x}_s^{\text{total}}(d)$ with $d = 0, \ldots, S_{\max} - 1$, where $\mathbf{x}_s^{\text{total}}(0)$ counts the EVs on the main diagonal, $\mathbf{x}_s^{\text{total}}(d)$ on the upper d^{th} diagonal, and $\mathbf{x}_s^{\text{total}}(-d)$ on the lower d^{th} diagonal of $\mathbf{x}_s$. The action $\mathbf{u}_s$ vector thus defines an action per diagonal, and for each of them states what fraction of the diagonal's EVs to charge.

Cost Function: We aim to flatten the aggregate EV charging load, while ensuring that every EV is fully charged before departing. The original proposition in [5] combined both aims as separate parts in the cost function for transitioning from s to s' by taking action $\mathbf{u}_s$:

$$C(s, \mathbf{u}_s, s')_{\text{old}} \triangleq C^{\text{demand}}(\mathbf{x}_s, \mathbf{u}_s) + C^{\text{penalty}}(\mathbf{x}_{s'}), \tag{1}$$

where $C^{\text{demand}}(\mathbf{x}_s, \mathbf{u}_s)$ is the (quadratic) power consumption from all connected EVs in the decision timeslot. $C^{\text{penalty}}(\mathbf{x}_{s'})$ is the penalty for unfinished charging, defined to be higher than simultaneously charging all EVs. Thus, C^{penalty} is activated when a car would move to below the main diagonal (where $\Delta t^{\text{charge}} > \Delta t^{\text{depart}}$), to ensure fully charging all EVs, including those without flexibility.

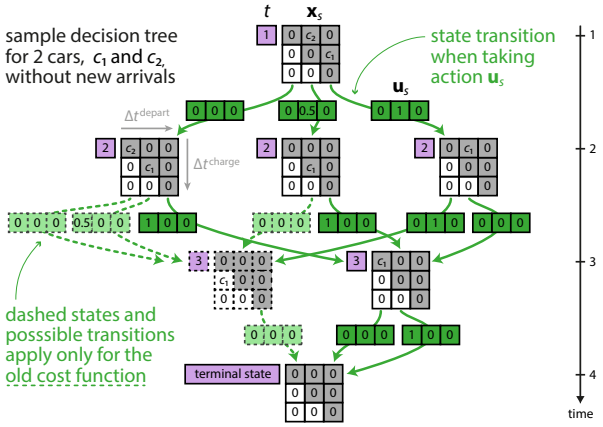

Figure 1: Example state-action decision tree for $N_{\max} = 2$, $S_{\max} = 3$.

Our newly defined cost for taking action $\mathbf{u}_s$ to get from cost state s to s' amounts to the charging power demand cost only:

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

In the updated cost definition, we no longer define the penalty term: we impose the policy to a priori charge all cars without any flexibility (i.e., those on the main diagonal, where indeed $\Delta t^{\text{charge}} = \Delta t^{\text{depart}}$, thus $\Delta t^{\text{flex}} = 0$), rather than having it learn to do that from experiencing a high penalty cost. This results in a reduced state-action space and significantly faster training for the model.

Size of State-Action Space: As opposed to the *action space* for a given state s defined in [5], our updated action space is smaller, because we improve the algorithm by not allowing to exploit flexibility where there is none. The total number of possible actions from a given state in the original algorithm (where, for each flexibility $\Delta t^{\text{flex}} = d$ we could charge any number of cars between $[0, \mathbf{x}_s^{\text{total}}(d)]$) is:

$$|\mathbf{U}_s|_{\text{old}} = \prod_{d=0}^{S_{max}-1} \left(\mathbf{x}_s^{\text{total}}(d) + 1\right). \tag{3}$$

This is updated to:

$$|\mathbf{U}_s|_{\text{updated}} = 1 + \prod_{d=1}^{S_{max}-1} \left(\mathbf{x}_s^{\text{total}}(d) + 1\right). \tag{4}$$

The first term in Eq. (4) reflects the single "choice" we have for the cars without flexibility (for $d = 0$, where $\Delta t^{\text{flex}} = 0$). This reduced action space size also shrinks the exploration space for the RL agent (see §2.2).

Figure 1 illustrates a scenario of $N_{\max} = 2$ EV charging stations with a horizon of $S_{\max} = 3$ slots. At time $t = 1$ we have $N_s = 2$ connected cars: C_1 with $(\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}) = (3, 2)$, and C_2 with $(\Delta t_2^{\text{depart}}, \Delta t_2^{\text{charge}}) = (2, 1)$ with no other arrivals during the control horizon. In timeslot 2, the leftmost state in Fig. 1 has both cars on the main diagonal, implying they have no flexibility ($\Delta t^{\text{charge}} = \Delta t^{\text{depart}}$, thus $\Delta t^{\text{flex}} = 0$) and a fortiori need to be charged. Hence, two feasible actions from previous MDP [5] will not be considered in our updated MDP implementation.

Value function: The learning objective is to minimize the expected T-step[1] return, which for a policy π at timestep t is defined as:

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

The policy then amounts to evaluate a state-action value function, and select the action that minimizes it. This value function, commonly named Q-function, is:

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

2.2 Batch Reinforcement Learning

We adopt the same batch RL algorithm as in [5, Algorithm 1], fitted Q-iteration (FQI), to approximate $\widehat{Q}^*(s, \mathbf{u})$ from past experiences generated using a non-optimum (e.g., random) policy. Each experience is defined in terms of (i) an initial state s, (ii) the action taken $\mathbf{u}_s$, (iii) the resulting state s' after taking the action, and (iv) the associated costs $C(s, \mathbf{u}_s, s')$. The experience set $\mathcal{F}$ is generated based on the cost function used:

- $\mathcal{F}_1$ (*old cost implementation*): Generate all possible actions for a given state, and use the cost function from Eq. (1).
- $\mathcal{F}_2$ (*updated cost implementation*): Only allow actions from the updated space-action tree, and use the cost function from Eq. (2).

Each experience set of $(s, \mathbf{u}_s, s', C(s, \mathbf{u}_s, s'))$ tuples is trained separately to deliver an optimum policy $\widehat{Q}^*(s, \mathbf{u})$, which is then used evaluated for the testing period. A fully connected Artificial Neural network (ANN) is used as function approximation for $\widehat{Q}^*(s, \mathbf{u})$.

3 EXPERIMENT SETUP

Our simulations will compare the original (see [5]) cost function and associated state-action space, with the updated ones, in terms of the resulting policies, trained with the respective updated and old cost datasets ($\mathcal{F}_1$ and $\mathcal{F}_2$). Experiments are run on a system with an Intel Xeon E5645 3.1 GHz processor and 16 GB RAM.

3.1 Parameters and Settings

Data preparation: Our dataset is derived from real world data collected by ElaadNL since 2011, from 2500+ public charging stations [6]. The maximum connection duration is set to $H_{\max} = 24$ h with time granularity $\Delta t^{\text{slot}} = 2$ h, which means $S_{\max} = 12$ timeslots. We jointly coordinate $N_{\max} = 10$ charging stations.

As summarized in §2.2, the input to the policy learning is a set of experiences, which depends on the cost function used: $\mathcal{F}_1$ (*old cost implementation*) and $\mathcal{F}_2$ (*updated cost implementation*). These experiences are collected as so-called trajectories by beginning at the starting of the day (t_1,x_1), taking actions randomly until a terminal state is reached (for details, see [5]). Each such trajectory comprises 12 tuples $(s_i, \mathbf{u}_{s,i}, s_{i+1}, C(s_i, \mathbf{u}_{s,i}, s_{i+1}))$. We vary the number of unique trajectories per day, $N_{\text{traj}} \in \{5K, 10K, 15K, 20K\}$.

Function Approximator: We use the same artificial neural network (ANN) architecture as in [5], comprising an input layer and 2 hidden layers with ReLU activation functions.

[1] As previously stated, we note the specific control time horizon as $T = S_{\max}$.

3.2 Evaluation

Training time: Defined as the time it takes for RL agent to be trained. To generate training data, we take contiguous period for a given duration of $\Delta t \in \{1, 3, 5, 7, 9$ months$\}$. For each Δt, we randomly selected 5 training data periods, each within the range between Jan. 1, 2015 and Sep. 30, 2015. We run 12 iterations to train the ANN for each selected period for a given Δt. We record the training time for each of these training datasets, for both agents (one trained on $\mathcal{F}_1$ and the other on $\mathcal{F}_2$).

Cost comparison: The last 3 months of 2015 are used as the test set for evaluation, i.e., $\mathcal{B}^{\text{test}} = \{e_i | i = 274, \ldots, 365\}$ containing 92 days. We report the same normalized cost as in [5], given by:

$$C_{\pi(\Delta t, j)} = \frac{1}{|\mathcal{B}^{\text{test}}|} \sum_{e \in \mathcal{B}^{\text{test}}} \frac{C^e_{\pi(\Delta t, j)}}{C^e_{\text{opt}}}. \quad (7)$$

Following costs are calculated and compared to analyze the various policies:

- $C_{\text{RL,updated}}$: cost of the policy trained with the updated cost.
- $C_{\text{RL,old}}$: cost of the policy obtained by [5].
- C_{BAU}: cost of the business-as-usual (BAU) policy[2].
- C_{opt}: for an optimum policy, derived from optimization with perfect knowledge of future EV connections.
- C_{heur}: for a discrete-action heuristic.

The latter heuristic policy assumes that individual EVs are charged uniformally over their entire connection time.[3]

4 RESULTS

4.1 Training Time

Figure 2 shows the training time for each of the old and new policies, i.e., using the respective cost functions Eq. (1) and Eq. (2), to answer research questions **Q1** (i.e., what training time reduce does the update cost function achieve?) and **Q2** (i.e., how do training set parameters affect training time?). Figure 2(a) compares the training time for increasing training dataset size in terms of number of sampled trajectories, for a training period of $\Delta t = 5$ months.[4] We note that our updated cost function and resulting policy achieves a reduction of training time compared to the old ones (from [5]) in the range of 42%–54% (for 5k–20k sampled trajectories per training day; averages over 5 runs).

Figure 2(b) reports similar relative differences in training time between old and updated policies when varying training period time spans, i.e., $\Delta t \in \{1, 3, 5, 7, 9$ months$\}$. Training time savings now range from 37%–53% when varying Δt from 1 to 7 months.

4.2 Cost Reduction Comparison

Figure 3 compares the normalized cost achieved by our improved formulation with that of previous work [5] as well as the baselines

[2] Continuously charge each EV upon arrival.

[3] Specifically, the heuristic spreads the c slots that the EV needs to charge over the total available number of slots d. This amounts to distributing $d - c$ no-charge slots evenly over the total number of d slots, thus splitting them into $d - c + 1$ parts. Assuming for simplicity that $c \geq d/2$, this means we insert a no-charge slot every $\lfloor d/(d - c + 1) \rfloor$ other slots. (For $c < d/2$, similarly distribute 'charge' slots evenly over the majority of 'no-charge' slots.)

[4] We noted a similar trend for all time spans, i.e., for all $\Delta t \in \{1, 3, 5, 7, 9$ months$\}$.

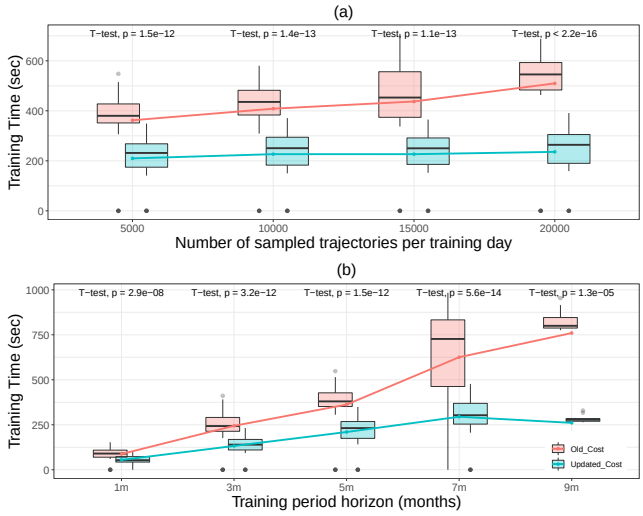

Figure 2: Policy training time for varying (a) number of samples per day N_{traj} (for Δt = 5 months), and (b) training period horizon Δt (for N_{traj} = 5000) .

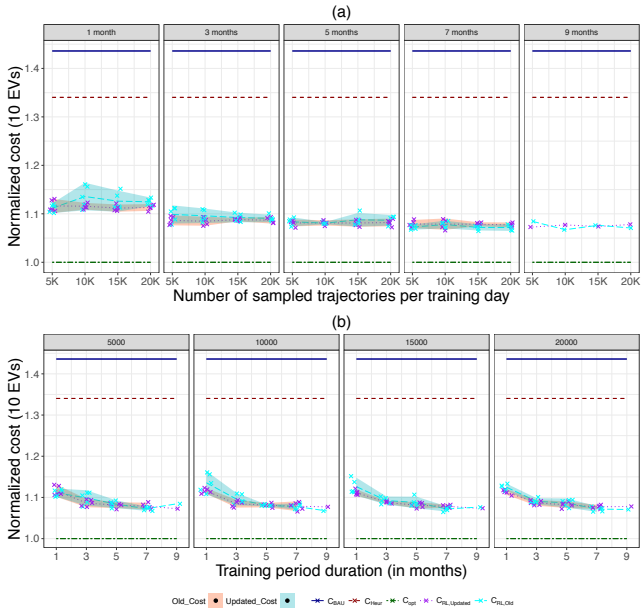

Figure 3: Normalized costs for increasing (a) N_{traj} and (b) Δt.

from §3.2. We note how normalized costs for the various RL policies vary with increasing training sets, in terms of (a) number of sampled trajectories per training day, and (b) training period time spans. Comparing old and new cost policies, we note no significant difference in the normalized cost (i.e., $C_{\text{RL,updated}} \approx C_{\text{RL,old}}$). There are slight variations in the distributions, which are statistically insignificant, and may be caused by the randomization of the samples. It is also worth noting that the RL algorithm performs better than both business-as-usual (BAU) and the newly defined heuristic policy. Unexpectedly, it performs worse the the all-knowing optimum

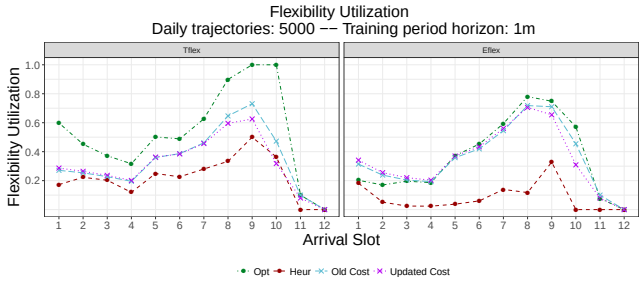

Figure 4: Flexibility Utilization for cars arrived in each slot.

policy baseline, since the latter has exact visibility of future EV arrivals. Still, the RL approach deviates less than 10%.

We conclude for **Q3** (effect of updating the RL cost function on the RL policy?) that performance of both cost functions is statistically similar (over multiple training sets).

4.3 Flexibility Utilization

To quantify the overall utilization of flexibility offered by users, we use previously introduced energy (E_{flex}) and time flexibility (T_{flex}) measures in [6]. These measures are important for energy providers and users alike (e.g., for providing incentives, cost minimization, time management). In short, E_{flex} reports the fraction of total charging load that is shiftable outside the BAU charging interval that is effectively deferred, and T_{flex} the fraction of the flex time window (Δt^{flex}) that is actually exploited.

To answer **Q4**, Fig. 4 plots T_{flex} and E_{flex} for EVs for each timeslot they arrive in, thus quantifying the utilized flexibility for the exemplary[5] training data of N_{traj} = 5K sample trajectories per day, and period Δt = 1 month. We note that the RL learning policy utilizes much more flexibility than the heuristic policy. On average, 40% of provided energy flexibility is utilized with RL. It is also important to note that RL policies energy flexibility utilization is close to that exploited in the all-knowing optimum policy. This indicates the trained RL agent balances loads similarly as the optimum policy, despite having no a priori exact knowledge of future EV arrivals.

5 CONCLUSION

We significantly improve the previously proposed reinforcement learning (RL) strategy in [5], to learn a policy coordinating the charging of multiple EVs. Our updated MDP definition, with a new cost function, strongly shrinks the the state-action space and thus significantly reduces training time (with > 50%), while retaining the beneficial performance of the RL trained policy. This training time reduction increases with larger training sets, thus suggesting the updated cost approach to be more practical. Future work includes evaluating an ϵ-greedy RL approach for the new MDP definition.

ACKNOWLEDGMENTS

The authors thank Joachim van der Herten for his feedback on [5] that inspired the current work.

[5] We notice similar trends in all sample sizes and time spans.

REFERENCES

[1] Adriana Chiș, Jarmo Lundén, and Visa Koivunen. 2016. Reinforcement learning-based plug-in electric vehicle charging with forecasted price. *IEEE Trans. Vehic. Technol.* 66, 5 (2016), 3674–3684.

[2] Bert J. Claessens, Stijn Vandael, Frederik Ruelens, Klaas De Craemer, and Bart Beusen. 2013. Peak shaving of a heterogeneous cluster of residential flexibility carriers using reinforcement learning. In *IEEE PES ISGT Europe 2013*. IEEE, 1–5.

[3] Junjie Hu, Hugo Morais, Tiago Sousa, and Morten Lind. 2016. Electric vehicle fleet management in smart grids: A review of services, optimization and control aspects. *Renew. Sust. Energ. Rev.* 56 (2016), 1207–1226.

[4] Nasrin Sadeghianpourhamami, Johannes Deleu, and Chris Develder. 2018. Achieving Scalable Model-Free Demand Response in Charging an Electric Vehicle Fleet with Reinforcement Learning. In *e-Energy2018, the 9th ACM International Conference on Future Energy Systems*. ACM Press, 1–3.

[5] Nasrin Sadeghianpourhamami, Johannes Deleu, and Chris Develder. 2019. Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning. *IEEE Transactions on Smart Grid* (2019).

[6] Nasrin Sadeghianpourhamami, Nazir Refa, Matthias Strobbe, and Chris Develder. 2018. Quantitive analysis of electric vehicle flexibility: A data-driven approach. *International Journal of Electrical Power & Energy Systems* 95 (2018), 451–462.

[7] José R. Vázquez-Canteli and Zoltán Nagy. 2019. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Appl. Energ.* 235 (2019), 1072–1089.