Managing Distributed Flexibility under Uncertainty by Combining Deep Learning with Duality

Georgios Tsaousoglou, Katerina Mitropoulou, Konstantinos Steriotis, Nikolaos G. Paterakis, *Member, IEEE*, Pierre Pinson, *Fellow, IEEE* , Emmanouel Varvarigos

***Abstract*—In modern power systems, small distributed energy resources (DERs) are considered a valuable source of flexibility towards accommodating high penetration of Renewable Energy Sources (RES). In this paper we consider an economic dispatch problem for a community of DERs, where energy management decisions are made online and under uncertainty. We model multiple sources of uncertainty such as RES, wholesale electricity prices as well as the arrival times and energy needs of a set of Electric Vehicles. The economic dispatch problem is formulated as a multi-agent Markov Decision Process. The difficulties lie in the curse of dimensionality and in guaranteeing the satisfaction of constraints under uncertainty. A novel method, that combines duality theory and deep learning, is proposed to tackle these challenges. In particular, a Neural Network (NN) is trained to return the optimal dual variables of the economic dispatch problem. By training the NN on the dual problem instead of the primal, the number of output neurons is dramatically reduced, which enhances the performance and reliability of the NN. Finally, by treating the resulting dual variables as prices, each distributed agent can self-schedule, which guarantees the satisfaction of its constraints. As a result, our simulations show that the proposed scheme performs reliably and efficiently.**

***Index Terms*—Distributed Energy Resources, Economic Dispatch, Energy Community, Electric Vehicles, Deep Learning**

I. Introduction

In modern power systems, there is an increasingly high penetration of small Distributed Energy Resources (DERs), such as rooftop solar panels, micro-generators and flexible controllable loads, predominantly Electric Vehicles (EVs). Moreover, many of these DERs exhibit high levels of uncertainty, in the sense that their constraints, costs and parameters are not deterministic. In such systems, the idea of local decision making (self-dispatch) is gaining increasing attention, where groups of DERs interact with the system as one aggregated entity. These entities are often called Energy Communities (ECs) [1], where the EC exchanges power with the system as a single entity, and an EC operator entity performs the energy management within the EC, i.e., coordinates the energy profiles of the community's DERs and decides on the power exchange with the main system.

Towards making dispatch decisions within an EC, local electricity markets have been the topic of several studies. Mechanism design approaches (e.g., [2], [3]) have been proposed, while many studies (e.g., [4], [5]) exploit duality to construct a price-based control scheme. However, in settings with high penetration of DERs, the local economic dispatch problem involves the energy management of numerous small entities, and also needs to be solved online and under uncertainty.

Regarding these new challenges, a certain part of the literature has focused on designing schemes for making efficient dispatch decisions under uncertainty. There is certain differentiation among studies, regarding the way they treat uncertainty. Some studies, e.g. [6], [7], [8], propose energy management schemes that solve a deterministic problem based on forecasts, and then reassess the scheduling using a rolling horizon technique. Some studies configure the energy management decisions with forecasting methods based on Machine Learning (ML). In [9], random forests are utilized in order to predict the uncertain parameters of flexible loads (EVs) before solving the economic dispatch problem, while [10] uses LSTM deep learning for a similar cause. However, in these approaches the uncertainty over forecasted parameters is not taken into account when making decisions.

Another group of studies uses stochastic programming to sample realization scenarios for the uncertain system parameters. In [11], the authors propose a stochastic-robust approach for the decisions of a system with EVs. In [12] a stochastic MILP formulation is proposed to assess the impact of RES and EV uncertainty on the energy management of a smart building, while in [13] and [14] the authors used stochastic dual dynamic programming to address RES uncertainty in dispatch problems. However, with multiple DERs, system parameters span over an exponentially large space, which means that the relatively very small number of scenarios sampled by a stochastic programming method can fail to generalize reliably.

A third family of studies leverages techniques from Artificial Intelligence to account for decision-making under uncertainty. The decision problem is modeled as a Markov Decision Process (MDP). In [15], a policy-rollout method is proposed to tackle the problem of a home energy management under uncertain electricity prices. In [16], a policy-improvement method is proposed, which is warm-started by assuming a "good" policy learned by experience. In [17], a battery is controlled using Approximate dynamic programming. Dynamic programming methods is a standard approach for tackling MDPs; however, the curse of dimensionality prevents this family of methods from generalizing to problems with multiple agents (e.g., EVs), unless simplifying assumptions are made.

Georgios Tsaousoglou received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No.754462. K. Steriotis and E. Varvarigos received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 863876 in the context of the FLEXGRID project.

G. Tsaousoglou, and N.G. Paterakis are with the Eindhoven University of Technology. K. Mitropoulou, K. Steriotis, and E. Varvarigos are with the National Technical University of Athens. P. Pinson is with the Technical University of Denmark.

Towards managing these issues, two ideas have been proposed in the literature: constructing customized techniques tailored to specific use cases, or leveraging the assistance of ML techniques for tackling multi-agent MDPs. Towards the first approach, the authors in [18] use dynamic programming to decide on the aggregated energy of an EV fleet, while an auction is used to allocate this energy among the EVs. A similar approach is taken in [19] for a system that also features RES generation. Towards the second approach, and in contrast to the studies that use ML only for forecasting system parameters, some studies opt for learning to optimize actions (dispatch decisions) directly. In [20] a deep reinforcement learning algorithm is proposed for making online dispatch decisions for EV charging stations. In [21], price-based control is realized via reinforcement learning, while a Neural Network (NN) is used as a function that maps prices to the agents' response. In [22], ML algorithms are trained (using the system's history) to make real time decisions on EV energy management.

An important drawback of these methods is that they cannot handle constraints explicitly, i.e., constraints are satisfied only in expectation [23]. In [24], a penalty term is designed to teach a NN to also respect the local constraints of the agents. As analyzed in [25], the design of such a penalty term comes with various trade-offs (e.g., efficiency is sacrificed in order to guarantee constraint satisfaction). The authors in [25], applied constrained policy optimization [26], which guarantees constraint satisfaction, in a setting with an EV that is charging under dynamic electricity prices. However, the method introduces high complexity and constraint satisfaction is relaxed in order to make the method faster. Moreover, the authors consider only one EV, i.e., the curse of dimensionality that occurs in multi-agent MDPs is not addressed.

In this paper, we formulate an economic dispatch problem as a multi-agent MDP, which cannot be tackled by standard dynamic programming algorithms. We apply a machine learning algorithm in the dual problem space and take advantage of the fact that its dimension (number of variables) is drastically smaller than the one of the primal problem. Moreover, the proposed method is able to guarantee constraint satisfaction. Our contributions can be summarized as follows:

- We consider multiple agents and multiple sources of uncertainty, i.e., a number of EVs with deadlines, RES and inflexible demand, as well as uncertain real-time electricity prices for drawing energy from the main system. Conventional generation cost is also modeled and a power balance constraint couples the decisions of all agents.
- A NN is trained to perform energy management decisions in real-time, upon receiving the information about the current system state. Instead of the primal problem, the dual problem is used to train the NN. With this technique, we reduce the NN's mean absolute error and enhance the NN's reliability.
- We propose an algorithm for energy management, through which the satisfaction of all the constrains is guaranteed. In particular, the distributed flexible loads (EVs) are allowed to self-schedule based on the dual variables provided by the NN.
- We perform energy management in a rolling horizon fashion, so that the NN can adapt its decisions based on new information about the system's state. The algorithm's performance is evaluated using simulations.
- Our results indicate that the proposed method achieves a near-optimal performance and significantly outperforms the conservative, constraint-satisfying benchmark.

The rest of the paper is organized as follows. Section II presents a model for an EC. The economic dispatch problem is formulated in Section III and the proposed solution is presented in Section IV. The evaluation framework is described in Section V while the simulation results are presented in Section V-D. Finally, Section VI, concludes the paper.

II. System Model

We consider an economic dispatch problem in a setting with conventional generators, RES generation, inflexible demand and a set of EVs that ask for charging services upon arrival. In what follows we model the operational characteristics of an EC for a certain time horizon T, where continuous time is divided into discrete timeslots of equal duration.

The EC features a set G of power generators. The power output of a generator $j \in G$ in timeslot $t \in T$ is denoted by decision variable $g_{j,t}$. Each generator is characterized by its lower and upper operational limits

$$g_j^{\min} \leq g_{j,t} \leq g_j^{\max}, \quad \forall j \in G, t \in T \tag{1}$$

and also bears a cost function $C_j(g_{j,t})$ that maps its output to a certain monetary cost. $C_j(\cdot)$ is taken in this work to be a quadratic function

$$C_j(g_{j,t}) = \mathrm{c}_j(g_{j,t})^2, \quad \forall j \in G, t \in T \tag{2}$$

The EC draws energy from the main electricity system at a time-varying per-unit price l_t. The vector of prices for all timeslots is denoted as $\mathbf{l} = \{\mathrm{l}_1, \mathrm{l}_2, ..., \mathrm{l}_{|T|}\}$. Energy drawn at t is denoted as $g_{0,t}$, while the line capacity of the coupling point is denoted as K, i.e.,

$$g_{0,t} \leq \mathrm{K}, \quad \forall t \in T \tag{3}$$

The inflexible demand of the EC at t is denoted as $\mathrm{p_{infl,t}}$ and the EC's RES generation as $\mathrm{g_{RES,t}}$. The respective vectors containing the parameter values for all timeslots are denoted with a bold symbol, i.e., $\mathbf{p}_{\mathrm{infl}} = \{\mathrm{p_{infl,1}}, \mathrm{p_{infl,2}}, ..., \mathrm{p_{infl,|T|}}\}$ and $\mathbf{g}_{\mathrm{RES}} = \{\mathrm{g_{RES,1}}, \mathrm{g_{RES,2}}, ..., \mathrm{g_{RES,|T|}}\}$.

The EC also features a set of chargers for electric vehicles (EVs). The EC operator is responsible for satisfying a set of charging tasks A, where a charging task $i \in A$ refers to allocating a certain amount of energy to an EV. Throughout the paper, we refer to "EVs" and "tasks" interchangeably. Control variable $p_{i,t}$ denotes the amount of power allocated to task i in timeslot t. A power balance constraint, makes sure that the power generated equals the power consumed in every timeslot

$$g_{0,t} + \mathrm{g_{RES,t}} + \sum_{j \in G} g_{j,t} = \mathrm{p_{infl,t}} + \sum_{i \in A} p_{i,t}, \qquad \forall t \in T \tag{4}$$

A task $i \in A$ is characterized by a tuple $\Omega_i = \{a_i, b_i, d_i, p_i^{\min}, p_i^{\max}, E_i, \delta_i\}$, where a_i is the task's arrival time, b_i is the task's desired completion time, d_i is a completion deadline, $p_i^{\min}$ and $p_i^{\max}$ are the lower and upper bounds on the EV's power consumption, E_i is the total energy required for the task to be considered satisfied, and δ_i is a flexibility parameter that relates to the disutility that comes from delayed task satisfaction. Note that a task bears a departure time (deadline) d_i, upon which it must have received its required charging, but it is preferable to i to receive charging in earlier timeslots (preferably before b_i). Naturally, it is $b_i \leq d_i$. A bold symbol $\mathbf{\Omega}_A = \{\Omega_i\}_{i \in A}$ denotes an instance of all task tuples. No energy can be allocated to a task, before the task's arrival time or after its completion deadline (i.e. after the EV departs)

$$p_{i,t} = 0, \quad \forall t \notin [a_i, d_i], i \in A \tag{5}$$

while a task $i \in A$ can only be charged between the minimum and maximum charging rates of the respective EV

$$p_i^{\min} \leq p_{i,t} \leq p_i^{\max}, \quad \forall i \in A, t \in [a_i, d_i]. \tag{6}$$

The energy requirement of the task, has to be satisfied before the task's deadline,

$$\sum_{t \in [a_i, d_i]} p_{i,t} = E_i, \quad \forall i \in A \tag{7}$$

Finally, when the task charges at timeslots later than its desired completion time b_i, it suffers a cost (disutility)

$$U(p_{i,t}) = \frac{\delta_i^{t-b_i} p_{i,t}}{E_i} \tag{8}$$

where δ_i is a constant parameter. Intuitively, the term $\delta_i^{t-b_i}$ penalizes charging in timeslots later than the desired departure time b_i and favors earlier timeslots. Function $U(p_{i,t})$ can also be interpreted as the compensation that i requires from the EC operator, in order to shape its consumption profile.

III. Problem Formulation

If all the information of the system (i.e., RES generation $\mathbf{g}_{\text{RES}}$, inflexible demand $\mathbf{p}_{\text{infl}}$, electricity prices $\mathbf{l}$ and tuples $\mathbf{\Omega}_A$ for the charging tasks of set A) was known before hand, the cost minimization problem of the EC operator would be a deterministic, convex optimization problem, which reads as

$$\min_{g_{0,t}, g_{j,t}, p_{i,t}} \left\{ \sum_{t \in T} \left(g_{0,t} \mathrm{l}_t + \sum_{j \in G} C_j(g_{j,t}) \right) + \sum_{i \in A} \sum_{t \in T} U(p_{i,t}) \right\} \quad \text{s.t. (1)–(8)} \tag{9}$$

However, the system parameters $\mathbf{g}_{\text{RES}}$, $\mathbf{p}_{\text{infl}}$, $\mathbf{l}$ and tuples Ω_i are not known beforehand (e.g. the EVs with their charging tasks arrive stochastically within the horizon without prior notice). Therefore, the optimal EC operation becomes a problem of decision making under uncertainty. We assume that the EC operator has access to statistical information about the uncertain parameters, in the form of a joint probability distribution $\mathcal{P}$ and that $\mathcal{P}$ is Markovian. Then, the problem structure is modeled as a Markov Decision Process (MDP) $\mathcal{M}$, defined as follows.

Formulation of MDP $\mathcal{M}$

- *State*: $\left\{ \tau, \mathbf{\Omega}_A(\tau), \left\{ \sum_{t \in [a_i, \tau-1]} p_{i,t} \right\}_{i \in A}, \mathrm{g}_{\text{RES},\tau}, \mathrm{p}_{\text{infl},\tau}, \mathrm{l}_\tau \right\}$
  The setting's *State* consists of all relevant information available at a given system instance. *State* variable τ denotes the current operation timeslot. Variable $\mathbf{\Omega}_A(\tau) = \{\Omega_i\}_{i \in A / \{\iota\}_{a_\iota > \tau}}$, denotes the tuples of tasks i that have arrived up to τ (i.e. $a_i \leq \tau$). Tasks that have not arrived yet are not included in the *State* information since their tuples are unknown at τ. Variable $\sum_{t \in [a_i, \tau-1]} p_{i,t}$ is the power that has been allocated to task i up until timeslot $\tau - 1$. Parameters $\mathrm{g}_{\text{RES},\tau}, \mathrm{p}_{\text{infl},\tau}, \mathrm{l}_\tau$ refer to the current operational timeslot τ.
- *Actions*: $\left\{ g_{0,\tau}, \{g_{j,\tau}\}_{j \in G}, \{p_{i,\tau}\}_{i \in A} \right\}$
  The action space is spanned over the possible values of decision variables $g_{0,\tau}$, $g_{j,\tau}$ and $p_{i,\tau}$.
- *Transition Functions*:
  - $\tau \rightarrow \tau + 1$
    After a decision on the power allocation for a certain timeslot τ is made, the system transitions to the next timeslot $\tau + 1$.
  - $\mathbf{\Omega}_A(\tau) \xrightarrow{\mathcal{P}} \mathbf{\Omega}_A(\tau + 1)$
    At the next timeslot $\tau + 1$, new tasks arrive stochastically, where their tuples Ω_i are derrived based on joint probability distribution $\mathcal{P}$.
  - $\left\{ \sum_{t \in [a_i, \tau-1]} p_{i,t} \right\}_{i \in A} \rightarrow \left\{ \sum_{t \in [a_i, \tau]} p_{i,t} \right\}_{i \in A}$
    At the next timeslot $\tau + 1$, the new allocated energy for each task is calculated by adding the allocation decision $p_{i,\tau}$ of the last timeslot, to the the previously allocated energy $\sum_{t \in [a_i, \tau-1]} p_{i,t}$.
  - For parameters $\mathrm{g}_{\text{RES},\tau}, \mathrm{p}_{\text{infl},\tau}, \mathrm{l}_\tau$, we assume stationary Markovian transition functions.
- *Cost* : $\mathrm{l}_\tau g_{0,\tau} + \sum_{j \in G} C_j(g_{j,\tau}) + \sum_{i \in A} U(p_{i,\tau})$
  The cost of a certain *Action* at a certain *State* is defined as the sum of the energy procurement cost and the aggregated disutility. ■

The goal is to find a policy π, i.e., a mapping from each *State* to an *Action*, which minimizes the expected cost $J(\pi)$

$$J(\pi) = \mathbb{E}_{\psi \sim \pi} \left[\sum_{t \in T} \left(\mathrm{l}_t g_{0,t} + \sum_{j \in G} C_j(g_{j,t}) + \sum_{i \in A} U(p_{i,t}) \right) \right] \tag{10}$$

of the EC in the horizon T, where $\psi \sim \pi$ is the set of *State-Action* trajectories conditioned over policy π. Moreover, the policy must belong to the set F_π, which contains the policies that respect the constraints (1)–(8)

$$\pi^* = \text{argmin}_{\pi \in F_\pi} J(\pi) \tag{11}$$

The number of possible *States* and *Actions* in MDP $\mathcal{M}$ grows exponentially in the number of charging tasks, gen-

erators, and horizon timeslots while $\mathcal{M}$ also features continuous *State* and *Action* variables. Thus, problem (11) cannot be tackled by traditional dynamic programming algorithms. Moreover, constraints (7) depend on the whole *State-Action* trajectory and not only on the current *Action*. This makes it very difficult to guarantee their satisfaction.

In particular, in standard MDP-solving frameworks, these kinds of constraints are incorporated in the *Cost* function of the MDP, multiplied by a penalty term ν, i.e. the MDP's *Cost* is extended with a term $\nu \sum_{i \in A} \left| \sum_{t \in [a_i, d_i]} p_{i,t} - E_i \right|$. The issue is that if the value of ν is not large enough, constraint satisfaction is not guaranteed. On the other hand, choosing a very large value for ν, forces the algorithm to pursue an overly conservative policy in order to make sure that constraints will be satisfied. Intuitively, in our context, an overly conservative policy would be to charge all EVs as fast as possible. This could lead to important efficiency loss as will be also shown in the simulations of Section V-D. In what follows we present a method that handles the intractability of the MDP $\mathcal{M}$, while constraint satisfaction is explicitly guaranteed.

IV. Solution Approach

In this section we present a novel method to tackle the MDP $\mathcal{M}$. We first present the high-level description of the algorithm. A very useful observation for MDP $\mathcal{M}$, is that its state variables can be conveniently separated in two special categories: the first category includes exogenous state variables $\mathbf{\Omega}_A(\tau), \mathrm{g_{RES,\tau}}, \mathrm{p_{infl,\tau}}, \mathrm{l_\tau}$, which have probabilistic transition functions but do not depend on the *Action* taken. The second category, includes τ and endogenous variables $\left\{ \sum_{t \in [a_i, \tau - 1]} p_{i,t} \right\}_{i \in A}$ (that depend on the *Action*), that both have deterministic transition functions. Thus, the *State* variables that have probabilistic transition functions are exogenous. By exploiting this property, the EC operator can run experiments for the setting, by sampling instances of $\mathbf{g}_{\mathrm{RES}}$, $\mathbf{p}_{\mathrm{infl}}$, $\mathbf{l}$, $\mathbf{\Omega}_A$ from $\mathcal{P}$ and, for each experiment, solve problem (9) as a deterministic problem to obtain an optimal solution.

In a given experiment k, the system transitions through a certain trajectory ψ_k, of $|T|$ *States* and respective optimal *Actions*. Therefore, after simulating an experiment k, we can derive a set of $|T|$ mappings, where a mapping $\mathcal{D}_S^{(k)}$ expresses an association between *State* $S_t^{(k)}$ (in which the system found itself) and the optimal *Action* (that the deterministic optimization method took in that *State*).

By running multiple offline experiments, the EC operator can generate multiple instances of $\mathcal{D}_S^{(k)}$ (i.e. for a number of different *States*). These instances can be conceived as Data, which are then used to perform supervised learning on a Neural Network (NN). Thus, given an amount of Data $\boldsymbol{\mathcal{D}}_S = \{\mathcal{D}_S^{(k)}\}_{\forall k}$, the NN can be trained so as to provide a function that maps a *State* to an optimal *Action*. Therefore, in real-time operation, the EC operator can observe the system's current *State* and feed it into the NN to obtain the NN's prediction of what would be the optimal *Action* in that *State*. The high-level procedure is illustrated in Fig. 1.

There are two challenges with the proposed approach. First, there are $|T|(1 + |G| + |A|)$ *Action* variables, which requires an equal number of output neurons. As a result, for only moderately large numbers of EVs in set A, efficiently training the NN becomes quite challenging. Second, the NN does not guarantee constraint satisfaction, which means that if an EV's power allocation is determined by the NN output, it is not guaranteed that the EV will fulfill its charging requirements. In order to tackle these challenges, it is useful to consider the Lagrangian relaxation of problem (9). By relaxing constraint (4), the Lagrangian is written as

$$\begin{aligned}\mathcal{L}(g_{0,t}, g_{j,t}, p_{i,t}, \lambda_t) = \\ \sum_{t \in T} \left(g_{0,t} \mathrm{l}_t + \sum_{j \in G} C_j(g_{j,t}) \right) + \sum_{i \in A} \sum_{t \in T} U(p_{i,t}) \\ - \sum_{t \in T} \left(g_{0,t} + \mathrm{g_{RES,t}} + \sum_{j \in G} g_{j,t} - \mathrm{p_{infl,t}} - \sum_{i \in A} p_{i,t} \right) \lambda_t\end{aligned}$$

where λ_t is the dual variable corresponding to constraint (4), for timeslot t. An EV's optimal response to a set of mutipliers $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_{|T|}\}$ is defined as

$$\begin{aligned} Q_i = \min_{p_{i,t}} \left\{ \sum_{t \in T} \left(U(p_{i,t}) + \lambda_t p_{i,t} \right) \right\} \\ \text{s.t. (5)–(8)} \end{aligned} \tag{12}$$

while the response of the generation side is defined as

$$\begin{aligned} R = \min_{g_{j,t}, g_{0,t}} \left\{ \sum_{t \in T} \left(g_{0,t} \mathrm{l}_t + \sum_{j \in G} C_j(g_{j,t}) \right) \right. \\ \left. - \sum_{t \in T} \lambda_t (g_{0,t} + \sum_{j \in G} g_{j,t}) \right\} \\ \text{s.t. (1)–(3)} \end{aligned} \tag{13}$$

The dual problem is defined as

$$\begin{aligned} \max_{\lambda_t} \left\{ R + \sum_{i \in A} Q_i \right\} \\ \text{s.t. } \lambda_\mathrm{t} \geq 0, \forall \mathrm{t} \in \mathrm{T} \end{aligned} \tag{14}$$

and since the primal problem (9) is a convex optimization problem, strong duality holds and the solution $\boldsymbol{\lambda}^*$ to problem (14) achieves optimality for problem (9).

Based on these observations, we can define the EC Operator's *Action* in terms of dual variables λ_t instead of primal variables $p_{i,t}, g_{j,t}, g_{0,t}$, which dramatically reduces the *Action* variables from $|T|(1 + |G| + |A|)$ to only $|T|$. In turn, this greatly facilitates the NN training. According to this formulation, a piece of data $\mathcal{D}_S^{(k)}$ derived from experiment k, is defined as a mapping from a system *State* $S_t^{(k)}$ to a set of dual variables $\boldsymbol{\lambda}^{(k)}$

$$\mathcal{D}_S^{(k)} \equiv S_t^{(k)} \rightarrow \boldsymbol{\lambda}^{(k)} \tag{15}$$

The exact procedure of the data generation step is described in Algorithm 1.

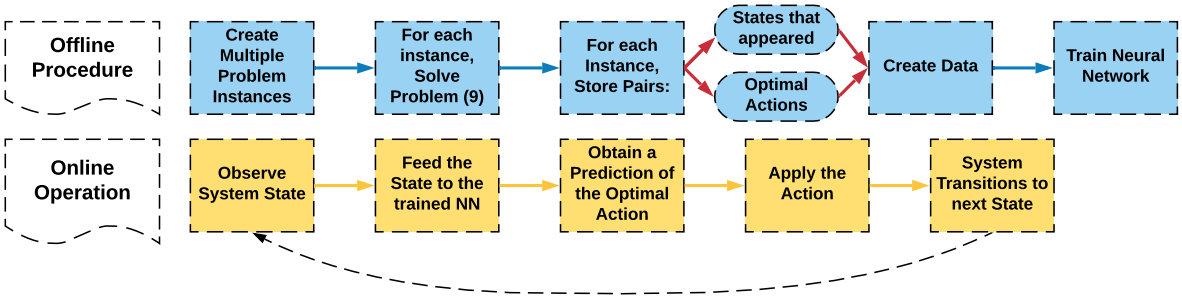

Fig. 1. High-level procedures of data creation and online *Action* selection

Algorithm 1 Data Generation for training the Neural Network

Initialize $k = 1$
While $k <$ number of experiments
  Sample $\mathbf{g}_{\text{RES}}^{(k)}$, $\mathbf{p}_{\text{infl}}^{(k)}$, $\mathbf{l}^{(k)}$, $\mathbf{\Omega}_A^{(k)}$ from $\mathcal{P}$
  Solve problem (9)
  Store optimal primal variables $\{g_{0,t}^{(k)}, g_{j,t}^{(k)}, p_{i,t}^{(k)}, \ \forall t \in T, i \in A\}$
  Store optimal dual variables $\boldsymbol{\lambda}^{(k)}$
  for each $\tau \in T$
  Create *State*
  $S_\tau^{(k)} = \left\{\tau, \mathbf{\Omega}_A^{(k)}(\tau), \left\{\sum_{t\in[a_i,\tau]} p_{i,t}^{(k)}\right\}_{i\in A}, \mathrm{g}_{\text{RES},\tau}^{(k)}, \mathrm{p}_{\text{infl},\tau}^{(k)}, \mathrm{l}_\tau^{(k)}\right\}$
  Create one piece of Data $D_S^{(k)} \equiv S_\tau^{(k)} \rightarrow \boldsymbol{\lambda}^{(k)}$
  end for
  $k = k + 1$

In real-time operation, the EC Operator only decides the dual variables λ_t (by observing the NN output) and communicates them to the EV-task agents. Thus, instead of a fixed power allocation, each agent is given a set of multipliers (prices) which it can use to solve its local problem and make sure that its local constraints are satisfied. At timeslot τ of online operation, agent i has already received power $\tilde{p}_{i,t}$ for timeslots $t < \tau$. Thus, i's local problem at τ, is defined as

$$\begin{aligned} &\min_{p_{i,t}} \left\{ \sum_{t\in T} \left(U(p_{i,t}) + \lambda_t p_{i,t}\right) \right\} \\ &\text{s.t. (5)–(8)} \\ &p_{i,t} = \tilde{p}_{i,t}, \ \ \forall t < \tau \end{aligned} \tag{16}$$

After the agents decide their charging power, they communicate it to the EC operator, and the latter can make sure that the power balance constraint is satisfied in the most cost-efficient way, by solving

$$\begin{aligned} &\min_{g_{j,t}, g_{0,t}} \left\{ \sum_{t\in T} \left(g_{0,t}\mathrm{l}_t + \sum_{j\in G} C_j(g_{j,t}) \right) \right\} \\ &\text{s.t. (1)–(3), (4)} \\ &g_{j,t} = \tilde{g}_{j,t}, \ \ \forall t < \tau \\ &g_{0,t} = \tilde{g}_{0,t}, \ \ \forall t < \tau \end{aligned} \tag{17}$$

where, again, $\tilde{g}_{j,t}$ and $\tilde{g}_{0,t}$ denote the decisions made in previous timeslots. The decisions for the current timeslot are implemented, and the procedure repeats for the next timeslot in a rolling horizon fashion. The exact procedure is described in Algorithm 2.

Although the space of possible states is exponentially large and the NN's training dataset cannot possibly explore it, we expect that the NN can learn the underlying structure of the optimal set of multipliers and provide a near-optimal output at any *State*. Moreover, by requesting that the NN provides the dual variables as output (instead of the primal variables $g_{0,t}$, $g_{j,t}$ and $p_{i,t}$) we reduce the number of output neurons from $|T|(1+|G|+|A|)$ to only $|T|$, which enhances the NN's performance. At the same time, the agents' decisions are made locally and, thus, constraint satisfaction is enforced.

Algorithm 2 Rolling Horizon Energy Management Algorithm

Initialize $\tau = 1$
While $\tau \in T$
  Observe system *State* $S_\tau^{(k)}$
  Feed *State* $S_\tau^{(k)}$ to the NN
  Obtain the NN output $\boldsymbol{\lambda}$
  Solve problem (16) using $\boldsymbol{\lambda}$
  Solve problem (17), where for constraints (4), $p_{i,t}$ is given by the solution of problem (16)
  Implement solutions $p_{i,\tau}, g_{j,\tau}, g_{0,\tau}$ for the current timeslot
  $\tau = \tau + 1$

A. Neural Network

The NN of Algorithm 2 is implemented as a Multi-Layer Perceptron (MLP), a class of feed-forward artificial neural networks. MLP is essentially a supervised learning algorithm that learns a non-linear function approximator (most common is stochastic gradient descent) for either classification or regression. An MLP trains on a dataset and learns a function $f(\cdot) : R^m \rightarrow R^n$, where m represents the number of dimensions for input, which is equal to the number of *State* variables in our case, and n represents the number of dimensions for output, which in our case is equal to $|T|$, since we have one dual variable for each timeslot.

The model of each neuron in each layer of the network includes a set of weighted inputs that are summed and passed through a nonlinear differentiable activation function. The purpose of our network is to receive as input all the (known) *State* variables at a given operational timeslot τ and return the output prices $\boldsymbol{\lambda}$, one for each timeslot of the horizon T.

For our purposes, the standard MLP needs to be remodeled so as to allow dynamic input size. Given an operational timeslot τ of Algorithm 2, the tuples Ω_i of EVs that have not arrived yet (i.e., the state variables Ω_i for the EVs with $a_i > \tau$) are not observable. Therefore the input size is dependent on τ. However, neural networks are, by design, restricted to accept fixed length input. In order to tackle this issue, a masking layer was added before the input layer. The alternative would be to train $|T|$ different networks, one for every timeslot of the horizon. In the final NN, all *State* variables were included in the input layer and when at a given τ a number of *State* variables is unobservable, their input is fixed to a specific value. The above-mentioned masking layer is essentially an additional array that records whether a value is actually present for a given input, or whether it is missing and thus should be skipped during the processing of the data. This technique is called data masking.

V. Performance Evaluation

A. Testbed description

The data were generated from 1000 offline simulations. For each simulation we considered an horizon T of 24 timeslots, where the horizon represents a two-hour interval, divided into 24 timeslots of 5-minute duration each. A total of 50 EVs (charging tasks) were considered $A = \{1, 2, ..., 50\}$ within this two-hour interval. For each task $i \in A$,

- The lower bound $p_i^{\min}$ on the EV's charging rate was set to zero.
- The upper bound $p_i^{\max}$ was picked randomly from the set $\{2, 3, 4, ..., 12\}$(kW).
- The arrival time a_i was picked randomly from set $\{3, 4, ..., 9\}$ for the first 17 tasks, and from set $\{14, 15, ..., 20\}$ for the rest of the tasks.
- The desired departure time b_i was set to $a_i + 3$.
- The required energy E_i was set as $E_i = b_i p_i^{\max}(\text{kWh}\frac{5}{60})$.
- The deadline d_i was set to $d_i = b_i + \xi$, where ξ was picked randomly from set $\{1, 2, 3, 4\}$.
- Parameter δ_i was picked randomly from interval $[1, 1.25]$.

Two local generators were modeled, $G = \{1, 2\}$, with technical minimum points $g_1^{\min} = g_2^{\min} = 0$(kW) and capacity limits $g_1^{\max} = 0.5|A|\max_{i\in A}\{p_i^{\max}\}$(kW) and $g_2^{\max} = 1000$(kW). The respective cost parameters c_j were set as $\mathrm{c}_1 = 0.003$ and $\mathrm{c}_2 = 0.01$, so as to simulate a base generator and a more expensive generator for demand peaks.

The system's electricity price l_t at timeslot t was generated by a random normal distribution with average value $\mu(\mathrm{l}_t)$ and standard deviation $\sigma(\mathrm{l}_t) = 0.03$. The value of $\mu(l_1)$ (for timeslot 1) was set as $\mu(l_1) = 0.4\frac{\$}{\text{kW}\frac{5}{60}}$. For later timeslots, the average value of $\mu(\mathrm{l}_t)$ was assumed to follow a Markov chain, where $\mu(\mathrm{l}_t) = \mathrm{l}_{t-1} + 0.02$.

Similarly, for the inflexible demand $\mathrm{p}_{\mathrm{infl,t}}$, the standard deviation was set to $\sigma(\mathrm{p}_{\mathrm{infl,t}}) = 4$ and the average value was modeled as a Markov chain, where $\mu(\mathrm{p}_{\mathrm{infl,t}}) = \mathrm{p}_{\mathrm{infl,t-1}} + 4$(kW). For the first timeslot, it was set $\mu(\mathrm{p}_{\mathrm{infl},1}) = 100$. Finally, RES generation $\mathrm{g}_{\mathrm{RES,t}}$ was set as $\mu(\mathrm{g}_{\mathrm{RES,t}}) = \mathrm{g}_{\mathrm{RES,t-1}} + 2.5$, $\sigma(\mathrm{g}_{\mathrm{RES,t}}) = 2.5$, and $\mu(\mathrm{g}_{\mathrm{RES},1}) = 30$(kW).

B. Neural Network Design

The implemented architecture consists of a 5-layer structure in which the input layer and the output layer are interconnected with two intermediate non-linear hidden layers, while the masking layer precedes the input layer. Each layer is dense and fully connected; the first hidden layer contains 150 hidden units, and the second contains 100 hidden units. The activation function chosen is ReLu, for computational simplicity.

The Adam algorithm was used as optimizer; a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Mean squared error was used as loss function, and mean absolute error as our evaluation metric. The network was trained for 100 epochs. A dataset of 24000 samples was generated, since, based on Algorithm 1, a number of $|T|$ pieces of data are created at each one of the 1000 experiments. The 92% of the data were used for training, while the rest 8% was reserved for testing. Finally, it should also be noted that the data were scaled before being processed.

C. Benchmarks

We considered two benchmarks with which we compared the proposed method. The first is the optimal-in-hindsight solution, where we assume that the EC operator has perfect knowledge over all uncertain parameters for the horizon T and solves problem (9) deterministically in order to acquire the optimal solution. Naturally, this benchmark serves as a theoretical upper bound on the performance of the proposed method and its solution cannot be obtained in practice.

The second benchmark is the conservative solution, where each arriving EV is charged at its maximum possible rate $p_i^{\max}$ until its demand is fulfilled. This would be the solution provided by a traditional MDP-solving algorithm, when accommodated with a large penalty term $\nu|\sum_{t\in[a_i,d_i]} p_{i,t} - E_i|$ in order to guarantee that constraints (7) will be respected. For this case study, this solution can also be obtained by setting

$$p_{i,t}^{(\text{conservative})} = \begin{cases} p_i^{\max}, & \text{for } t \in [a_i, b_i] \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

and solving problem (17), where in constraint (4), it is set $p_{i,t} = p_{i,t}^{(\text{conservative})}$.

D. Results

In this subsection we present the simulation results. For a particular instance of the setting, Fig. 2 depicts the aggregated consumption of EVs $\sum_{i\in A} p_{i,t}$ as resulted by Algorithm 2 and the conservative benchmark, on top of the net demand which is the inflexible demand $\mathrm{p}_{\mathrm{infl,t}}$ minus the RES generation $\mathrm{g}_{\mathrm{RES,t}}$ for each timeslot. From Fig. 2, we can observe that Algorithm

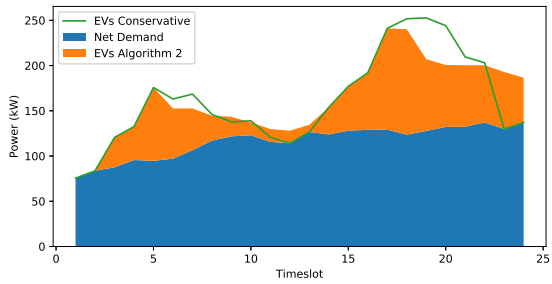

Fig. 2. Electric Vehicles load on top of Energy Community's net demand for the proposed and the conservative case

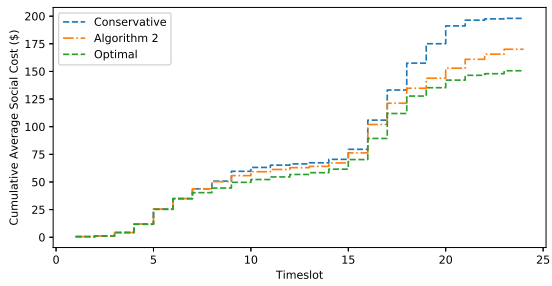

Fig. 3. Average Social Cost achieved by the proposed algorithm compared to the two benchmarks

2, in contrast to the conservative solution, opts for a peak shaving in timeslots 6-7, 18-21 and a valley-filling in the respective consequent timeslots.

Algorithm 2 and the two benchmarks were tested in a number M of different instances, where in each instance the testbed's parameters were sampled from the probability distributions described in Section V-A. The Social Cost $SC_{m,t}$ for an instance m and timeslot t was calculated as

$$SC_{m,t} = g_{0,t}\mathrm{l}_t + \sum_{j\in G} C_j(g_{j,t}) + \sum_{i\in A} U(p_{i,t})$$

The Social Cost for each timeslot was averaged out over all instances and the Cumulative Average Social Cost $\overline{SC}_t^{CuAv}$ is defined as

$$\overline{SC}_t^{CuAv} = \frac{\sum_{\tau\in[0,t]} \sum g_{0,\tau}\mathrm{l}_\tau + \sum_{j\in G} C_j(g_{j,\tau}) + \sum_{i\in A} U(p_{i,\tau})}{M}$$

The Cumulative Average Social Costs achieved by the proposed algorithm and the two benchmarks are presented in Fig. 3. The proposed approach on average achieves a 14.1% decrease in the EC cost compared to the conservative benchmark, while it suffers a 11.4% higher cost than the optimal-in-hindsight solution.

The proposed approach is able to achieve this performance, mainly for two reasons. First, thanks to the small number of output neurons, the Neural Network achieves a very good performance towards tracking the optimal dual variables of problem (9). This is shown in Fig. 4, where the Probability Density Function (PDF) of the Mean Absolute Error (MAE)

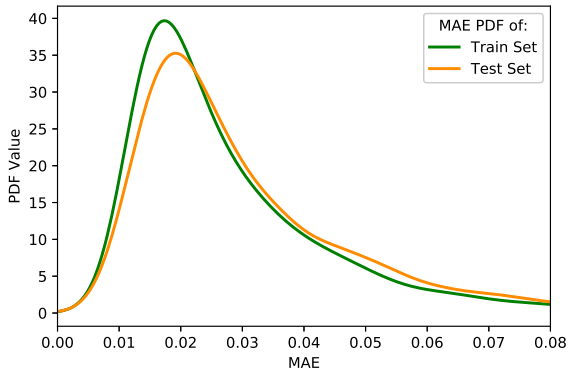

Fig. 4. Probability Density Function of the Mean Absolute Error for the train and test dataset

is depicted for both the train and the test dataset. Both PDFs follow a positively skewed normal distribution, the mode of which is around 0.02.

Second, the rolling horizon approach of Algorithm 2, allows the NN to update the dual variables in an online fashion upon receiving new information on the system's *State*. Indeed, for later timeslots τ (i.e., as time passes in the actual system operation), the NN's performance is improved since it has accumulated more information about the problem's instance. This is quantified and presented in Fig. 5, where the MAE is depicted separately for each timeslot, i.e., the samples are grouped by the last known timeslot for which system parameters are observable.

Moreover, as stated in line 8 of Algorithm 2, only the decisions for the current timeslot τ are implemented in the system, and when the system transitions to the next timeslot, the decisions are updated. In Fig. 6 the MAE for both timeslot τ and the whole horizon T is depicted as a function of the training epochs. As can be observed, the MAE for the current timeslot τ is even smaller compared to the MAE of the whole horizon. In practice, since the procedure is repeated in a rolling horizon fashion and only the decision for τ is implemented, it is more important that the NN achieves a good prediction of the optimal dual variable λ_τ for the current timeslot τ rather than for the whole horizon.

VI. Conclusions and Future Work

We considered an economic dispatch problem in a setting with multiple sources of uncertainty. The problem of minimizing the expected social cost was formulated as a Markov Decision Process (MDP). The MDP is intractable due to the curse of dimensionality, introduced by the presence of multiple agents (EVs). We trained a Neural Network (NN) on data that relates a system's state to the optimal set of dual variables. By having the NN learn the dual variables instead of the primal, we drastically reduce the number of neurons in the NN's output layer. Moreover, by treating the NN's output duals as prices, we allow each EV agent to self-schedule, which guarantees the satisfaction of local constraints. This allows the algorithm to escape from the standard approach, which either

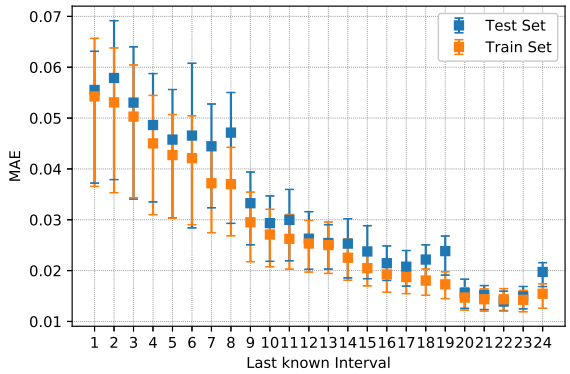

Fig. 5. MAE box-plots (Q1, Q3) for the train and test dataset

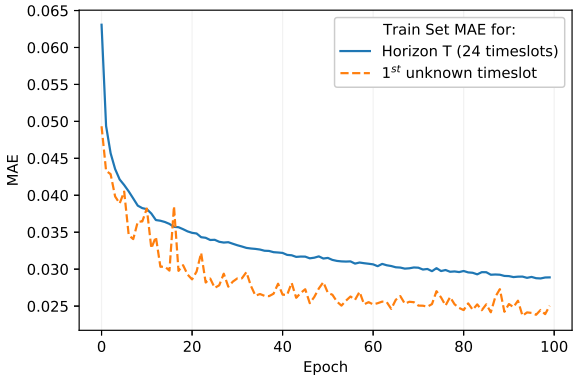

Fig. 6. MAE obtained during 100-epoch period training, taking into account the predictions of the whole horizon T (blue), versus considering only the predictions of the first timeslot of each simulation (orange)

satisfies constraints only in expectation, or opts for an overly conservative policy that results in loss of efficiency. A rolling horizon algorithm, containing the NN, was constructed and applied to a case study. Simulation results demonstrate the superiority of the proposed algorithm in comparison to the conservative solution. Future work can assess the method's suitability in different settings and use cases as well as in general economic dispatch problems.

References

[1] I. Mamounakis, N. Efthymiopoulos, P. Makris, D. J. Vergados, G. Tsaousoglou, and E. M. Varvarigos, "A novel pricing scheme for managing virtual energy communities and promoting behavioral change towards energy efficiency," *Electric Power Systems Research*, vol. 167, pp. 130 – 137, 2019.

[2] C. P. Mediwaththe, M. Shaw, S. Halgamuge, D. B. Smith, and P. Scott, "An incentive-compatible energy trading framework for neighborhood area networks with shared energy storage," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 1, pp. 467–476, 2019.

[3] G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, P. Makris, and E. Varvarigos, "Truthful, Practical and Privacy-aware Demand Response in the Smart Grid via a Distributed and Optimal Mechanism," *IEEE Transactions on Smart Grid*, pp. 1–1, 2020.

[4] F. Moret and P. Pinson, "Energy collectives: A community and fairness based approach to future electricity markets," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 3994–4004, 2019.

[5] Y. Zhang, N. Rahbari-Asr, J. Duan, and M.-Y. Chow, "Day-ahead smart grid cooperative distributed energy scheduling with renewable and storage integration," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 4, pp. 1739–1748, 2016.

[6] R. Deng, Z. Yang, J. Chen, N. R. Asr, and M. Chow, "Residential energy consumption scheduling: A coupled-constraint game approach," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1340–1350, 2014.

[7] S. Chakraborty, M. Cvetkovic, K. Baker, R. Verzijlbergh, and Z. Lukszo, "Consumer hedging against price volatility under uncertainty," in *2019 IEEE Milan PowerTech*, 2019, pp. 1–6.

[8] J. Sachs and O. Sawodny, "A two-stage model predictive control strategy for economic diesel-pv-battery island microgrid operation in rural areas," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 3, pp. 903–913, 2016.

[9] V. Lakshminarayanan, V. G. S. Chemudupati, S. K. Pramanick, and K. Rajashekara, "Real-time optimal energy management controller for electric vehicle integration in workplace microgrid," *IEEE Transactions on Transportation Electrification*, vol. 5, no. 1, pp. 174–185, 2019.

[10] J. Zhu, Z. Yang, Y. Chang, Y. Guo, K. Zhu, and J. Zhang, "A novel lstm based deep learning approach for multi-time scale electric vehicles charging load prediction," in *2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, 2019, pp. 3531–3536.

[11] S. Minniti, A. N. M. M. Haque, N. G. Paterakis, and P. H. Nguyen, "A hybrid robust-stochastic approach for the day-ahead scheduling of an ev aggregator," in *2019 IEEE Milan PowerTech*, 2019, pp. 1–6.

[12] D. Thomas, O. Deblecker, and C. S. Ioakimidis, "Optimal operation of an energy management system for a grid-connected smart building considering photovoltaics' uncertainty and stochastic electric vehicles' driving schedule," *Applied Energy*, vol. 210, pp. 1188 – 1206, 2018.

[13] A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur, "Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 2, pp. 547–558, 2017.

[14] R. Lu, T. Ding, B. Qin, J. Ma, X. Fang, and Z. Y. Dong, "Multi-stage stochastic programming to joint economic dispatch for energy and reserve with uncertain renewable energy," *IEEE Transactions on Sustainable Energy*, 2019.

[15] T. M. Hansen, E. K. P. Chong, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "A partially observable markov decision process approach to residential home energy management," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1271–1281, 2018.

[16] Y. Yang, Q. Jia, G. Deconinck, X. Guan, Z. Qiu, and Z. Hu, "Distributed coordination of ev charging with renewable energy in a microgrid of buildings," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6253–6264, 2018.

[17] C. Keerthisinghe, G. Verbič, and A. C. Chapman, "A fast technique for smart home management: Adp with temporal difference learning," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3291–3303, 2018.

[18] 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 Transactions on Smart Grid*, vol. 4, no. 2, pp. 720–728, 2013.

[19] D. Li and S. K. Jayaweera, "Distributed smart-home decision-making in a hierarchical interactive smart grid architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 75–84, 2015.

[20] M. Shin, D. Choi, and J. Kim, "Cooperative management for pv/ess-enabled electric vehicle charging stations: A multiagent deep reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3493–3503, 2020.

[21] Y. Du and F. Li, "Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1066–1076, 2020.

[22] K. L. López, C. Gagné, and M. Gardner, "Demand-side management using deep learning for smart charging of electric vehicles," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2683–2691, 2019.

[23] F. de Nijs, "Resource-constrained multi-agent markov decision processes," PhD Dissertation, Delft University of Technology, 2019.

[24] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-free real-time ev charging scheduling based on deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, 2019.

[25] H. Li, Z. Wan, and H. He, "Constrained ev charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2020.

[26] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, no. 10, 2017, p. 22–31.