energies

MDPI

Article

Computing Day-Ahead Dispatch Plans for Active Distribution Grids Using a Reinforcement Learning Based Algorithm

Eleni Stai *, Josua Stoffel and Gabriela Hug

EEH—Power Systems Laboratory, ETH Zürich, Physikstrasse 3, 8092 Zürich, Switzerland
* Correspondence: elstai@ethz.ch

Abstract: The worldwide aspiration for a sustainable energy future has led to an increasing deployment of variable and intermittent renewable energy sources (RESs). As a result, predicting and planning the operation of power grids has become more complex. Batteries can play a critical role to this problem as they can absorb the uncertainties introduced by RESs. In this paper, we solve the problem of computing a dispatch plan for a distribution grid with RESs and batteries with a novel approach based on Reinforcement Learning (RL). Although RL is not inherently suited for planning problems that require open loop policies, we have developed an iterative algorithm that calls a trained RL agent at each iteration to compute the dispatch plan. Since the feedback given to the RL agent cannot be directly observed because the dispatch plan is computed ahead of operation, it is estimated. Compared to the conventional approach of scenario-based optimization, our RL-based approach can exploit significantly more prior information on the uncertainty and computes dispatch plans faster. Our evaluation and comparative results demonstrate the accuracy of the computed dispatch plans as well as the adaptability of our agent to input data that diverge from the training data.

Keywords: active distribution grids; battery control; reinforcement learning; dispatch plan

Citation: Stai, E.; Stoffel, J.; Hug, G. Computing Day-Ahead Dispatch Plans for Active Distribution Grids Using a Reinforcement Learning Based Algorithm. *Energies* **2022**, *15*, 9017. https://doi.org/10.3390/en15239017

Academic Editor: Alberto Geri

Received: 31 October 2022
Accepted: 23 November 2022
Published: 29 November 2022

1. Introduction and Background

The growing world aspiration of a sustainable energy future has led to an increasing deployment of renewable energy sources (RESs) [1], which however pose new challenges to grid operators due to their intermittency and uncertainty. Not only structural changes are required, caused by the shift from large, centralized towards smaller, decentralized power plants but also adaptions and progress in the grid operation are necessary. The generation of conventional power plants such as nuclear or coal-fired power plants is controllable and can directly follow demand. On the contrary, the generation of RESs depends on uncertain natural factors such as wind, solar irradiation or river flow and can only be adapted up to a certain degree [2]. Therefore, predicting and planning the operation of modern power grids with RESs has become much more complex than before. Batteries can serve as the means for safely integrating RESs into power grids in large scales, as they can absorb the uncertain fluctuations of RESs' generation by charging with the surplus energy when generation is higher than demand and discharging to cover the extra demand vice versa. Due to the capability of adapting the power output within short notice, batteries are suitable for providing such ancillary services [3]. Additionally, also batteries' construction cost is expected to keep decreasing [4]. However, in order to fully exploit the batteries' potentials towards enabling a more sustainable electrical grid, their operation needs to be optimally planned thereby exploiting any available prior knowledge on the RESs generation.

In this paper, we study the planning of the operation of active distribution grids with RESs and batteries with the aim of increasing the predictability of the joint power output. In particular, we design a mechanism to compute an ahead-of-operation (for instance day-ahead) dispatch plan [5] at a point of the power grid where several distributed resources (e.g., RESs and batteries) are aggregated together. A dispatch plan is defined as a time

sequence of power values at this aggregation point over a future time horizon. In addition, the aggregation point is usually the Point of Common Coupling (PCC) of the distribution grid with an upper level grid (e.g., the transmission system). During real-time operation, the goal of the power grid operator is to control batteries so as to follow the pre-computed dispatch plan at the PCC, while taking into account the battery limitations as well as the grid constraints. The conventional solution approach of the dispatch plan computation problem for a distribution grid with RESs and batteries under grid security and batteries' constraints involves handling a non-convex AC Optimal Power Flow (AC OPF). Considering the RESs uncertainties further complicates the problem as this is achieved by most often either incorporating scenarios (scenario-based optimization) [5–7] or chance constraints (chance-constrained optimization) [8,9].

There exist several approaches in the literature proposing solution methods of the non-convex AC OPF for power grids, and particularly distribution grids. The majority of them focuses on radial grids, e.g., [5,10–17]. These methods apply relaxation techniques and some provide guarantees on the exactness of the solution under specific conditions (e.g., [5,10–13]). Among them, in [5], we have developed CoDistFlow, an efficient heuristic iterative scheme for dispatching, which is proven to give a solution that satisfies the full nonlinear power flow equations and the full non-convex grid security and battery constraints for scenario-based optimization. This approach is further extended in [18] for intra-day dispatching.

There exist also a few approaches for meshed topologies. For instance, the approaches of [19,20] apply the Lassere moment relaxations in AC OPFs that have a polynomial structure with respect to the complex voltage phasors. In [21], the authors apply Taylor approximations to linearize the AC OPF. Another linearization approach is proposed in [22], using prior knowledge on the statistics of the demand. Finally, there exist heuristic approaches that apply to both meshed and radial topologies and provide locally optimal solutions such as genetic algorithm-based schemes [23] and Lagrangian-based nonlinear optimization methods [24].

The challenge with current model-based methods is that solving the OPF for dispatching and other applications for large systems is computationally intensive and therefore often cannot exploit the full prior knowledge on the uncertainty [5,19,22]. Especially when updates to the dispatch plan are to be carried out intra-day, the available computational time is limited [18]. For instance, although thousands of historical scenarios of RESs generation trajectories might be available, in scenario-based optimization, we can usually account for only tenths of those obtained via scenario-reduction [5,18,25]. Using chance-constrained optimization on the other hand leads to tractability issues, requiring approximations.

To solve computational issues and to account for all the available prior information on the uncertainty, in this paper, we propose a machine learning technique for computing an operation-ahead dispatch plan for a distribution grid with RESs and batteries. In particular, we train a Reinforcement Learning (RL) agent [26] that will be used to solve the dispatching problem. The RL agent learns by data collected from the system as well as by penalties and rewards; traditional optimization techniques and a model of the system are no longer required. Although training is time-consuming, it can be conducted offline in advance while considering much more information on the uncertain RESs and loads than, e.g., scenario-based optimization is capable of. In addition, we do not have to handle the tractability issues related to chance-constrained optimization. After training is performed, using a trained agent is very fast and practical.

However, there exist only a few works in the literature applying RL for the dispatching problem. The reason is that RL is not inherently suited for planning problems because the dispatch plan has to be decided for the whole next day a priori and thus it is not possible to receive feedback for a decided value of the plan before choosing the following ones. Moreover, the few existing related works do focus on different problems than the one considered in this paper, i.e., computing operation-ahead plans for EV charging [27], battery charging [28] and demand response [29,30]. In particular, Ref. [27] computes a day-ahead

plan for charging an EV fleet with fitted Q-Iteration. In [29], batch fitted Q-learning is used to construct an offline policy for day-ahead scheduling and an online policy for dynamic pricing in DR. In [30], the authors use RL for computing a day-ahead dispatch plan for a hybrid power system consisting of a renewable energy resource (wind or solar) along with a thermal power element. The objective is to minimize the system's operating cost, the active network losses and the renewable energy disposal. Finally, in [28], a battery energy system is dispatched in a microgrid so as to minimize the monetary costs and the deviation from a fixed power at the PCC.

In this paper, to the best of our knowledge for the first time, an algorithm for computing dispatch plans for active distribution grids with RESs and batteries is presented that is based on an RL agent and inherits all the previously mentioned advantageous properties of RL. Our RL agent is trained with the Deep Deterministic Policy Gradient (DDPG) [31] algorithm that, contrary to the Fitted Q-Iteration used in most of the works mentioned above, is for both continuous action sets and continuous state sets. The DDPG algorithm is an actor–critic RL scheme. The actor is a function that represents the policy that selects the actions, and the critic is another function that approximates the Q-value rather than storing it for each state action pair as in tabular Q-learning. Both the actor and the critic functions are expressed as neural network approximations.

The rest of the paper is organized as follows: Section 2 presents the system model. Section 3 gives the conventional AC OPF optimization problem for computing dispatch plans, which, in Section 4, is formulated as a Markov Decision Process so that it can be solved with RL. In Section 5, we provide the proposed solution using the DDPG algorithm and, in Section 6, we present the evaluations and comparative results. Finally, Section 7 concludes the paper.

2. System Model

In this section, we describe the considered grid and battery models. The models are explicitly used for the conventional AC OPF problem formulation given in Section 3. On the contrary, our proposed RL-based approach for dispatching is model-free and does not require the exact model of the system. However, we will still use the models below to simulate the grid behavior and compute the reward values for the RL algorithms.

We consider the more general case of a meshed distribution network that consists of one slack bus that is the PCC of the distribution grid with the upper level grid and a number of PQ buses. We index the slack bus by 0 and the PQ buses by positive integers. At each PQ bus, a number of non-controllable resources and a battery can be connected. Non-controllable resources include inflexible loads and non-curtailable RESs. Since the non-controllable loads are uncertain, we use scenarios to model their uncertainties. In particular, the non-controllable resources are modeled as given power values for each time and scenario. The scenarios are used both for the scenario-based AC optimal power flow problem formulation in Section 3 and for training our RL agent later in Section 5.

The distribution lines are modeled using the π model with shunt elements, which is illustrated in Figure 1. Particularly, for line l, the longitudinal impedance is denoted as $z_l = r_l + jx_l$ and each of the two shunt capacitances is given by $j\frac{b_l}{2}$. Let Y represent the admittance matrix of the distribution grid.

In order to model the lossy batteries, we employ the so-called resistance-line model proposed in [5]. In detail, a battery connected to PQ bus i is replaced by (i) a virtual PQ bus i' that connects to bus i, (ii) the line between buses i' and i that is considered as purely resistive, and (iii) a lossless battery that becomes attached to the virtual bus i'. Then, the battery's internal active power losses are computed as the power losses on the purely resistive line. This modeling method is illustrated in Figure 2. Of course, with this battery model, the number of buses in the grid model increases, i.e., for N_B batteries in the grid, the number of buses in the grid model becomes $N + N_B$, and the Y matrix is updated accordingly. Finally, after replacing all batteries with their models, only the virtual buses $\{N+1, ..., N+N_B\}$ have a non-zero battery capacity.

Finally, the set of scenarios is denoted as $\mathcal{D} = \{1, ..., D\}$ and the set of time indices as $\mathcal{K} = \{1, ..., K\}$. The duration of a time interval is Δt.

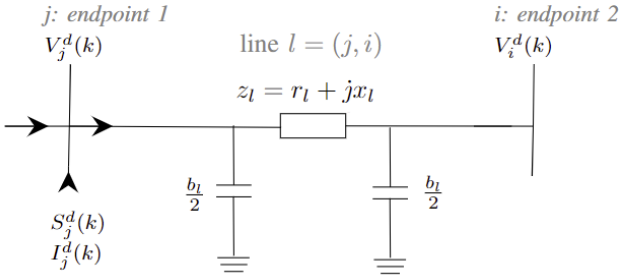

Figure 1. π model of the distribution line.

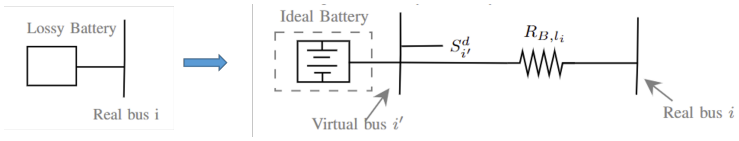

Figure 2. Resistance-line model of the battery.

3. Dispatching Active Distribution Grids: AC Optimal Power Flow Formulation

In this section, we formulate the conventional scenario-based AC OPF problem for computing an operation-ahead dispatch plan at the PCC. In particular, its solution is a dispatch plan denoted by S^{DP}, i.e., a sequence of complex power values (indicating either import from or export to the main grid) over a future time horizon with length K, at the PCC. Since the set $\mathcal{D}$ may be large for formulating a tractable scenario-based AC OPF problem, we define a subset of $\mathcal{D}$, denoted as $\mathcal{D}_{OPT}$ and derived by scenario reduction from $\mathcal{D}$ (performed similarly as in [5]), which is then used for the formulation of the scenario-based AC OPF. Note that $\mathcal{D}$ is usually in the order of thousands, whereas $\mathcal{D}_{OPT}$ in the order of tenths of scenarios.

3.1. Constraints

3.1.1. Power Flow Constraints, $\forall d \in \mathcal{D}_{OPT}$, $\forall k \in \mathcal{K}$

We apply the node injection form of the power flow equations in rectangular coordinates [32] that is exact for both radial and meshed distribution grid topologies. Let $V^d(k) \in \mathbb{C}^{N+N_B}$ be the complex voltage vector for scenario d and time k with length equal to the number of buses $N + N_B$. Then,

$$V^d(k) \;=\; V_{re}^d(k) + jV_{im}^d(k) \tag{1}$$

where $V_{re}^d(k)$, $V_{im}^d(k) \in \mathbb{R}^{N+N_B}$ are vectors consisting of the real and imaginary voltage parts, correspondingly. Let the vector $I^d(k) \in \mathbb{C}^{N+N_B}$ include the complex current injections to all buses at time k and scenario d (defined similarly to the voltage vector). In addition, let the vector $S^d(k) \in \mathbb{C}^{N+N_B}$ include the aggregated per bus controllable and non-controllable complex power injections for all buses at time k and scenario d. Then, the load flow constraint for scenario d and time k is given by:

$$\begin{aligned} S^d(k) \;&= diag(V^d(k))(I^d(k))^* \\ &= diag(V_{re}^d(k) + jV_{im}^d(k))\; Y^*\; (V_{re}^d(k) + jV_{im}^d(k))^*,\; \forall k \in \mathcal{K},\; \forall d \in \mathcal{D}_{OPT} \end{aligned} \tag{2}$$

Note that $S^d(k)$ is constant for the buses $i \in \{1, ..., N\}$ and controllable for the virtual buses $i \in \{N + 1, ..., N + N_B\}$ (for which it represents the battery power injections) and for the PCC, i.e., bus 0.

3.1.2. Voltage Constraints, $\forall d \in \mathcal{D}_{OPT}$, $\forall k \in \mathcal{K}$

The voltage constraints correspond to

$$V^d_{re,0}(k) = 1,\ V^d_{im,0}(k) = 0 \tag{3}$$

$$diag(\underline{V})\underline{V} \leq diag(V^d(k))V^{d*}(k) \leq diag(\overline{V})\overline{V},\ \forall k \in \mathcal{K},\ \forall d \in \mathcal{D}_{OPT} \tag{4}$$

where (3) sets the slack bus voltage to the nominal value (equal to 1), and (4) expresses lower and upper bounds on the nodal voltage magnitudes, where $\underline{V}, \overline{V} \in \mathbb{R}^{N+N_B}$ are vectors including, respectively, the lower and upper voltage magnitude bounds for all buses ($diag(x)$ stands for a diagonal matrix with the elements of vector x.) Note that, for $i \in \{N+1, ..., N+N_B\}$, $\underline{V}_i$ should be set very low and $\overline{V}_i$ should be set very high, since we do not impose voltage constraints on the virtual buses as they are not real buses but parts of the battery modeling.

3.1.3. Battery Constraints $\forall d \in \mathcal{D}_{OPT}, \forall k \in \mathcal{K}, \forall i \in \{N+1, ..., N+N_B\}$

Let us first introduce the following notation. $SoE^d_{B,i}(k)$ is the battery state-of-energy (SoE) at virtual bus i, time k and scenario d. Parameters $\underline{SoE}_{B,i}$, $\overline{SoE}_{B,i}$ are respectively the SoE lower and upper bounds expressing the capacity limits of the battery. Finally, $S^R_{B,i}$ is the rated power of the battery at virtual bus i.

Then, the battery constraints are given by

$$SoE^d_{B,i}(k+1) = SoE^d_{B,i}(k) + \Re\{S^d_i(k)\}\Delta t \tag{5}$$

$$\underline{\text{SoE}}_{B,i} \leq SoE^d_{B,i}(k) \leq \overline{SoE}_{B,i} \tag{6}$$

$$(\Re\{S^d_i(k)\} + (I^d_{l_i}(k))^2 \cdot R_{B,l_i})^2 + (\Im\{S^d_i(k)\})^2 \leq (S^R_{B,i})^2 \tag{7}$$

where l_i is the (unique) virtual line connected to virtual bus i, R_{B,l_i} is its resistance and $I^d_{l_i}(k)$ is the current flowing through it. Equation (5) gives the state-of-energy evolution equation for a battery at virtual bus i, for scenario d and time k. Note that $\text{SoE}^d_{B,i}(0)$ is given. Constraint (6) expresses lower and upper bounds on the battery state-of-energy and (7) bounds the battery apparent power. Finally, note that the current value is the same along the virtual line l_i since there are no shunt elements.

3.2. Objective Function

We define the following objective function:

$$\sum_d \lambda_d \left(\sum_k \left(w_1 |\Im\{S^d_0(k)\}| + w_2 |\Re\{S^d_0(k)\}| + w_3 \Re\{S^d_0(k)\} + w_4 ||S^d_0(k) - S^{DP}(k)||^2 \right) \right) \tag{8}$$

which is a weighted sum of four sub-objectives with w_i, $i = 1, \ldots, 4$ being the non-negative weight values. The weights $\{w_i\}_{i=1:4}$ determine the relevant importance of each sub-objective with respect to the others, i.e., if w_i is higher than w_j, then the sub-objective i is more important than the sub-objective j. The objective function is further expressed as an expected value over the scenarios with λ_d being the occurrence probability of scenario d and $\{\lambda_d\}_{d \in \mathcal{D}_{OPT}}$ a probability distribution over all scenarios, i.e., $\sum_d \lambda_d = 1$. The first sub-objective aims at minimizing the reactive power at the PCC, which is essentially equivalent to maximizing the power factor at the PCC. The second and the third sub-objectives in combination aim at minimizing the cost of the power exchanged with the upper level grid. In particular, for the power exported to the main grid, the price received is $w_3 - w_2$ and for the power imported from the upstream grid, the price paid is $w_3 + w_2$. The fourth sub-objective penalizes any deviation of the realized complex power at the PCC for each scenario from the dispatch plan value. In this work, the value of the weight w_4 is set higher than the other three weight values in order to achieve our main goal, which, as explained

earlier, is to compute a dispatch plan that, with high probability, will be followed in real-time operation with an as small as possible incurred regulation error.

3.3. Optimization Problem Formulation and Characteristics

The non-convex scenario-based AC OPF is formulated as follows:

$$\min \ \ (8), \ \ s.t. \ (2)-(7) \ \forall k, \ d, \ i \tag{9}$$

The problem is non-convex and the non-convexities are introduced by the nonlinear power flow equations (2), the left part of the voltage constraint (4) as well as the apparent battery power constraint (7).

In the following, we solve problem (9) using RL. Another possible approach would be to use supervised machine learning as, e.g., in [33] that would however involve solving the hard non-convex problem (9) several times during the learning phase to obtain the targets. On the contrary, with RL, we completely avoid solving (9).

4. Definition of a Markov Decision Process for the Dispatching Problem

In order to define a Markov decision process (MDP), we need to determine its state, action, transition function and reward. In particular, in an MDP framework the following interaction with a stochastic environment takes place: At each time k, the current state $\chi(k) \in \mathcal{S}$ is observed. Then, an action $a(k) \in \mathcal{A}$ is selected based on the policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ and applied. After the application of the action and the realization of the stochastic quantities included in $d(k)$, the state of the environment updates to $\chi(k+1)$ following the so-called transition function, $\chi(k+1) = \mathbf{p}(\chi(k), a(k), d(k))$ that models the environment dynamics. The environment also produces a reward signal $r(k)$. The objective of the MDP framework is to determine a policy that is optimal in the sense of maximizing the expected discounted long-term reward $\sum_{\ell=1}^{K} \gamma^{\ell} \mathbb{E}[r(\ell)]$, with $\gamma \in [0,1]$ being the discount factor.

In order to solve the optimization problem (9) using RL, we first write an MDP related to it, i.e., we determine the state and actions as well as the transition function and the rewards. One possible way to define such an MDP is to consider as action the full dispatch plan over a whole day, i.e., to set the action as K complex power values at the PCC, one for each time interval. However, this would require a large action space, which in turn would imply curse of dimensionality effects [34]. To avoid this, we formulate an MDP that considers a single time interval and outputs the dispatched complex power at the PCC only for this time interval. The challenge with this approach is that the state should include the system feedback from the action decided in the previous time interval based on the uncertainty realization. However, we do not have this feedback as the dispatch plan values are not applied in the system when computed but only the following day. In other words, the challenge is that we need to design an open loop policy to choose $a(k)$ for all $k \in \mathcal{K}$ when knowing only the initial state $\chi(0)$ and not knowing $\{\chi(1), \chi(2), \dots, \chi(K-1)\}$. To do so, we need to estimate the feedback for each agent's action based on forecasts of the uncertain quantities. There are a few preliminary approaches that tackle this issue, e.g., [27–29] that use distance metrics and Monte Carlo to estimate $\{\chi(1), \chi(2), \dots, \chi(K-1)\}$. We will detail more on this estimation when describing the transition function below. First, we will describe the states and action spaces. We use the same notation as in Section 3 but without the scenario index.

4.1. Definition of the State

For every time interval, the current system state is observed and based on its value, a set of actions is decided. The system state at the beginning of each time interval k: $[k\Delta t, (k+1)\Delta t]$ is denoted as $\chi(k)$ and consists of (i) the states-of-energy of all batteries, $\{\text{SoE}_{B,i}(k)\}_{\forall i=N+1:N+N_B}$; (ii) the estimated values, $\{S_i(k)\}_{i=1:N}$, for each uncontrollable load as well as for the distributed RESs for the time interval k; (iii) the voltage values for all buses; (iv) the actual active and reactive power at the PCC; and (v) the time interval index. However, the bus voltages and the actual power at the PCC for the time interval k

are unknown at the beginning of the time interval k and are only revealed after the actions for k have been performed. Thus, in the state, we take into account their values from the previous time interval, i.e., those that occurred as a result of the (estimated) prosumption values $\{S_i(k-1)\}_{i=1:N}$ and the actions $a(k-1)$ of the previous decision step.

Therefore, for each time interval k, the state is defined as follows:

$$\chi(k) = \{\{\text{SoE}_{B,i}(k)\}_{\forall i=N+1:N_B}, V(k-1), \{S_i(k)\}_{i=1:N}, S_0(k-1), k\} \tag{10}$$

Note that including the time interval index in the state is important as the action (dispatch plan) refers to a particular time interval and the same state inputs but for different time intervals may lead to different actions.

4.2. Definition of the Action

At the beginning of the time interval k, after observing the state $\chi(k)$, a set of actions is taken that consists of (i) the active and reactive dispatch plan power values at the PCC and (ii) the charging or discharging power of the batteries. Therefore, at each time interval k, the following set of actions is decided:

$$a(k) = \{S^{DP}(k), \{S_i(k)\}_{i=N+1:N+N_B}\} \tag{11}$$

where $S^{DP}(k)$ includes the active and reactive dispatch at the PCC at time k.

4.3. Transition Function

The state transition function gives the new system state after the action $a(k)$ is applied and the uncertainty $d(k)$ is revealed. It includes the physical equations of the grid (i.e., the power flow Equation (2)) and the battery SoE evolution equation (i.e., (5)). However, for the day-ahead dispatching, as we have already mentioned, it is not possible to directly observe the real state of the distribution network after applying the action, i.e., the dispatched power at the PCC for a single time interval k. Since the dispatch plan is calculated day-ahead based only on the initial state of the system, the system states of all subsequent time intervals must be estimated. For this purpose, the transition function that leads from one state to the next is actually a process that is based on a load flow calculation using estimated prosumption values (e.g., forecasts). Figure 3 illustrates the process of estimating the state for the time interval $k+1$ based on the state and the actions for time k.

In particular, after the decision-making, random noise is added to the actions for exploration purposes and are forwarded to the simulated grid. Prior to this, random noise is also added to the renewable energy values in $\{S_i(k)\}_{i=1:N}$ to simulate uncertainty. Let us denote the obtained action and non-controllable estimated prosumption after the random noise is added as $\tilde{a}(k)$ and $\{\tilde{S}_i(k)\}_{i=1:N}$, correspondingly. Then, a simple load flow calculation is performed with the power injections $\{\tilde{S}_i(k)\}_{i=1:N+N_B}$ as inputs and the estimated bus voltages $V(k)$ as well as the estimated realized power at the PCC $S_0(k)$ as outputs. Those estimated values are then part of the new system state $\chi(k+1)$ and are also used in the reward function (12) (introduced in the next section). In addition, the SoE of the batteries must be updated based on the batteries active power values $\{\Re\{\tilde{S}_i(k)\}\}_{i=1:N+N_B}$ and the battery SoE evolution equation (i.e., (5)). The updated system state is then given by $\chi(k+1) = \{\{\text{SoE}_{B,i}(k+1)\}_{\forall i=N+1:N+N_B}, V(k), \{S_i(k+1)\}_{i=1:N}, S_0(k), k+1\}$, i.e., it consists of the estimated batteries SoE values, the estimated bus voltages, the estimated PCC power as well as the estimated values, (e.g., forecasts) of the non-controllable loads for the next time interval.

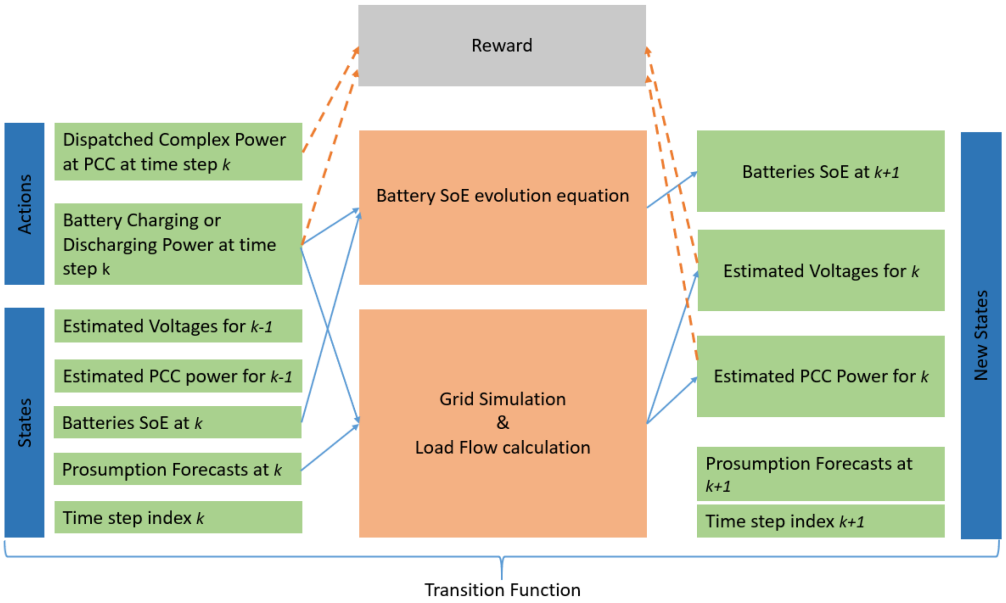

Figure 3. Transition function from one state to the next and inputs for computing the reward.

4.4. Reward Function

The reward function at time interval k is defined as:

$$\begin{aligned} r(\chi(k+1), a(k)) = -\Big[& w_1 |\Im\{S_0(k)\}| + w_2 \, |\Re\{S_0(k)\}| + w_3 \Re\{S_0(k)\} + w_4 ||S_0(k) - S^{DP}(k)||^2 \\ & + w_5 \cdot \sum_{i=1}^{N} \max\Big\{ \underline{V}_i^2 - V_i(k) V_i^*(k); \;\; 0; \;\; V_i(k) V_i^*(k) - \overline{V}_i^2 \Big\} \\ & + w_6 \cdot \sum_{i=N+1}^{N+N_B} \max\Big\{ (\Re\{S_i(k)\} + (I_{l_i,i}(k))^2 \cdot R_{B,l_i})^2 + (\Im\{S_i(k)\})^2 - (S_{B,k}^R)^2, 0 \Big\} \\ & + w_7 \cdot \sum_{i=N+1}^{N+N_B} |\Re\{S_i(k)\}| \Big] \end{aligned} \tag{12}$$

In particular, it is constructed by (i) the opposite of the time-step cost as defined in (8), (ii) a penalty term for the violation of the voltage constraints in (4), weighted by $w_5 \geq 0$, (iii) a penalty term for the violation of the battery apparent power constraint in (7), weighted by $w_6 \geq 0$ and (iv) a penalty term for the battery cycling operation, weighted by $w_7 \geq 0$. The inputs for computing the reward are illustrated in Figure 3. Note that an RL-based scheme cannot account for hard constraints, but it can be encouraged to satisfy the hard constraints by including them as penalties in the reward function. Thus, here we will encourage our RL-agent (presented in Section 5) to satisfy the hard voltage and battery apparent constraints by including penalties for each of them. However, learning provides no guarantees; therefore, constraints violations might still (rarely) occur. Due to this reason, we introduce a back up controller in Section 5.

5. Proposed Solution Using Reinforcement Learning

RL-algorithms are mainly based on computing approximations of the action-value function (Q-function) for finding the optimal policy. The Q-function is defined as the expected discounted long term reward when taking action a in state χ and following policy π afterwards, i.e.,

$$Q^{\pi}(\chi, a) = \mathop{\mathbb{E}}_{\pi} [G(k) \,|\, \chi(k) = \chi, a(k) = a] \tag{13}$$

where $G(k)$ is the expected discounted long term reward at time k (i.e., $G(k) = \sum_{\ell=k}^{K} \gamma^{\ell} \mathbb{E}[r(\chi(\ell+1), a(\ell))]$).

5.1. Solving the MDP with the DDPG Algorithm

Algorithm 1 describes the off-policy, model-free DDPG RL algorithm [31] that trains an RL agent so as to solve the MDP of Section 4. In Algorithm 1, $Q_\theta(\chi, a)$ (with parameter θ) stands for the Neural Network (NN) called the critic network that approximates the Q-function (i.e., (13)) and $\mu_\phi(\chi)$ (with parameter ϕ) stands for the NN called the policy network that approximates the policy. Algorithm 1 includes the two main enhancements that are typical in all DDPG-style algorithms, namely, (i) the experience replay used to alleviate any correlation effects that learning only via sequential data brings [35], and (ii) the target networks used to encourage learning stability. In detail, with experience replay, we store past transitions in a so-called replay buffer (here denoted as $\mathcal{B}$) and, in order to update the Q-network, we use a minibatch of transitions randomly sampled from $\mathcal{B}$ instead of the single most recent transition. For the target networks, we use two types: the target critic $Q_{\theta_{\text{targ}}}$, with parameter θ_{targ}, and the target actor $\mu_{\phi_{\text{targ}}}$, with parameter ϕ_{targ}. Their parameters are updated towards the parameters of the original networks, as follows:

$$\begin{aligned} \theta_{\text{targ}} &\leftarrow \tau\theta + (1-\tau)\theta_{\text{targ}} \\ \phi_{\text{targ}} &\leftarrow \tau\phi + (1-\tau)\phi_{\text{targ}} \end{aligned} \tag{14}$$

where $\tau \ll 1$ is called the soft update coefficient.

At the end of time interval k, i.e., after the state has been observed, the agent's action applied and the next state and reward estimated, we update the replay buffer by storing in it the transition $(\chi(k), a(k), r(k), \chi(k+1))$.

In summary, the main steps of Algorithm 1 are as follows: First, the actor and critic networks as well as the target networks and the replay buffer are initialized (lines 1–3). Then, the algorithm iterates over several historical or forecast scenarios of the prosumption loads. In each iteration, the initial system state is observed first (line 6) and then the following process is performed for all time intervals of the day: The agent selects actions (planned dispatched complex power at PCC and battery active and reactive power) based on the current policy. As seen in line 8, noise for exploration is added after the actor has chosen the actions. In line 9, the battery charging or discharging power is corrected to ensure that batteries are not charged or discharged further if they are already full or empty, correspondingly. This is accomplished by simply taking the minimum of the selected action and the maximum possible charging or discharging power. This correction is formulated as follows and applied for $i = N+1 : N+N_B$:

$$\Re\{S_i(k)\} = \begin{cases} \min\left(\Re\{S_i(k)\}, \frac{\overline{\text{SoE}}_{B,i} - \text{SoE}_{B,i}(k)}{\Delta t}\right), & \Re\{S_i(k)\} \geq 0 \\ \max\left(\Re\{S_i(k)\}, \frac{\underline{\text{SoE}}_{B,i} - \text{SoE}_{B,i}(k)}{\Delta t}\right), & \Re\{S_i(k)\} < 0 \end{cases} \tag{15}$$

Before performing the load flow calculation to receive the new state and the reward, noise is added to the prosumption values to simulate uncertainties (line 10). The results of the load flow calculation describe the new system state and determine the reward (line 11). The derived four-tuple, namely state, action, reward and next state is stored in the replay buffer (line 12). Hereafter, the learning process begins, where first a minibatch of four-tuples is sampled from the replay buffer (line 13) and then the actor and critic parameters are updated (lines 14–16). Finally, the target networks are updated (line 17).

Algorithm 1: DDPG algorithm

Inputs: Initial values for the policy parameters ϕ, for the Q-function parameters θ, and for the soft update coefficient $\tau \ll 1$. Set of training scenarios $\mathcal{D}$. Initialize critic $Q_\theta(\chi, a)$ and actor $\mu_\phi(\chi)$ networks with weights θ and ϕ, respectively.
Initialize target networks $Q_{\theta_{targ}}$ and $\mu_{\phi_{targ}}$ with weights $\theta_{targ} \leftarrow \theta, \phi_{targ} \leftarrow \phi$.
Initialize replay buffer $\mathcal{B}$ as empty.
for $d = 1, ..., D$ **do**
    Initialize a random process $\mathcal{N}$ for adding noise.
    Observe initial state $\chi(1)$.
    for $k = 1, ..., K$ **do**
        Sample noise σ from $\mathcal{N}$. Select action $a(k) = \mu_\phi(\chi(k)) + \sigma$ according to the current policy.
        Correct battery actions $\{S_i^d(k)\}_{i=N+1:N+N_B}$ using (15) if necessary.
        Sample noise from $\mathcal{N}$ for each prosumption value in $\{S_i^d(k)\}_{i=1:N}$ and add it to the prosumption value.
        Simulate the grid and run a load flow with injections in $a(k)$ and the perturbed $\{S_i^d(k)\}_{i=1:N}$. Calculate the reward $r(k)$ and the new state $\chi(k+1)$.
        Store transition $(\chi(k), a(k), r(k), \chi(k+1))$ in $\mathcal{B}$.
        Sample a random minibatch of B transitions $(\chi_j, a_j, r_j, \chi'_j)$ from $\mathcal{B}$.
        Set $y_j = r_j + \gamma Q_{\theta_{targ}}(\chi'_j, \mu_{\phi_{targ}}(\chi'_j))$ for all transitions j in the minibatch.
        Update critic by minimizing the loss: $L = \frac{1}{B}\sum_j (y_j - Q_{\theta_{targ}}(\chi_j, a_j))^2$.
        Update actor using the sampled policy gradient:

$$\nabla_\phi J \approx \frac{1}{B}\sum_j \nabla_a Q_\theta(\chi, a)|_{\chi=\chi_j, a=\mu_{\phi_{targ}}(\chi_j)} \nabla_\phi \mu_\phi(\chi)|_{\chi_j}$$

        Update target networks:

$$\theta_{targ} \leftarrow \tau\theta + (1-\tau)\theta_{targ}$$

$$\phi_{targ} \leftarrow \tau\phi + (1-\tau)\phi_{targ}$$

    end
end

5.2. Generating the Day-Ahead Dispatch Plan

Algorithm 1 outputs a trained DDPG agent that gives the planned dispatched power at the PCC for a particular time interval k. However, we want to compute the dispatch plan over all K time intervals of the following day. To do so, we use the agent trained by Algorithm 1 as well as forecast scenarios for the prosumption at the buses following the process detailed in Algorithm 2. The set of prosumption forecast scenarios used in Algorithm 2 is denoted as $\mathcal{D}_{day}$ and is different and usually much smaller than $\mathcal{D}$ that is used for training. Note that Algorithm 2 does not train the RL agent, so $\mathcal{D}_{day}$ can be as small as desired. Usually, it contains the forecast scenarios that we have available for the day for which we compute the dispatch plan. In particular, for every time interval (in $\mathcal{K}$) and every scenario (in $\mathcal{D}_{day}$), Algorithm 2 calls the trained DDPG agent including in the inputs the forecasted prosumption for the particular time interval and scenario and obtains the planned dispatched power for this particular time and scenario (line 4). Afterwards, it estimates the state for the next time interval, using the transition function given in Section 4 but without adding noise. After repeating the process for all scenarios, the dispatch plan

value for each time interval is obtained as a weighted average over the planned dispatched power values for all scenarios (line 11), i.e.,

$$S^{DP}(k) = \frac{\sum_{d \in \mathcal{D}_{day}} \lambda^d_{day} S^{DP,d}(k)}{|\mathcal{D}_{day}|} \tag{16}$$

where λ^d_{day} is the probability of occurrence of the prosumption forecast scenario $d \in \mathcal{D}_{day}$ with $\sum_{d \in \mathcal{D}_{day}} \lambda^d_{day} = 1$. In our case, all scenarios were assumed to be equally likely, so $\lambda^d_{day} = \frac{1}{|\mathcal{D}_{day}|}$.

The process of Algorithm 2 is very fast, i.e., the calculation of the dispatch plan for a scenario can be achieved in less than a couple of seconds. Therefore, we can consider hundreds of prosumption scenarios for the calculation of the dispatch plan, which is not possible for traditional scenario-based optimization techniques such as CoDistFlow. Indeed, this is a major advantage of using RL, compared to previous approaches such as the CoDistFlow algorithm.

Algorithm 2: Generating the Dispatch Plan

1 **Inputs:** Dispatch plan horizon K, prosumption forecasts at all buses $\{S^d_i(k)\}_{i=1:N}$ for each scenario $d \in \mathcal{D}_{day}$ and time interval $k \in \mathcal{K}$, trained DDPG agent
2 **for** $d = 1, ..., |\mathcal{D}_{day}|$ **do**
3 Observe state $\chi(1)$
4 **for** k=1,...,K **do**
5 $a(k) \leftarrow \{S^{DP,d}(k), \{S^d_i(k)\}_{i=N+1:N+N_B}\}$ selected actions based on the current state, $\chi(k)$, using the trained DDPG agent
6 Correct battery actions $\{S^d_i(k)\}_{i=N+1:N+N_B}\}$ using (15) if necessary
7 Compute the new battery state $\mathrm{SoE}^d_{B,i}(k+1) = \mathrm{SoE}^d_{B,i}(k) + \Re\{S^d_i(k)\}\Delta t$
8 Run a load flow calculation with injections $\{S^d_i(k)\}_{i=1:N+N_B}$ to get $V^d(k)$ as well as $S^d_0(k)$
9 Compute the next state $\chi(k+1)$ using the transition process described in Section 4 but without adding noise
10 Store the actions $\Re\{S^{DP,d}(k)\}$ and $\Im\{S^{DP,d}(k)\}$
11 **end**
12 **end**
13 For each k, calculate the average of $\Re\{S^{DP,d}(k)\}$ and $\Im\{S^{DP,d}(k)\}$ over all d to obtain the final DP, S^{DP}, using (16).
14 **Output:** S^{DP}

In the numerical evaluations, we will study how Algorithm 2 performs when the prosumption forecast scenarios in $\mathcal{D}_{day}$ differ significantly from those used for training the DDPG agent in Algorithm 1 (i.e., $\mathcal{D}$).

6. Evaluation Results and Comparisons

In this section, we evaluate the performance of Algorithm 2 and compare it with CoDistFlow [5] that directly solves the scenario-based optimization problem (9).

6.1. Distribution Grid

For the evaluations, we use the 34-bus distribution grid (including the PCC) shown in Figure 4, which is the same as the one used for the experiments in [5,18]. The bus 0 is the PCC and connects the distribution grid to the upper level grid. At the remaining buses, loads, as well as one PV and one battery are connected. Specifically, the battery is connected at bus 1 and is replaced by the battery model explained in Section 2. Table 1 lists the grid parameter values for the 34-bus grid.

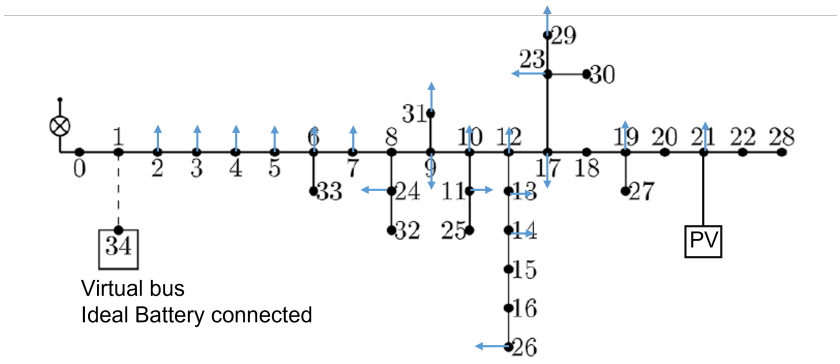

Figure 4. Diagram of the 34-bus grid. The blue arrows indicate connected loads.

Table 1. Grid parameters for the 34-bus grid.

Description	Value	Unit
Base power (3-phase)	25	MW
Base voltage (3-phase)	21	kV
Battery capacity	6	MWh
Battery apparent power limit	6	MW
Initial battery SoE	33	%
max (magnitude) line impedance	$(0.34 + j0.24)$	Ω
min (magnitude) line impedance	$(0.025 + j0.01)$	Ω
max/min shunt capacitance	200.15/6.73	μ
resistance of the virtual battery line	0.017	Ω

The setting in Figure 4 is capable of enabling grid operation without voltage constraints violations, i.e., there exists decisions on the battery power and the dispatched power at the PCC that are feasible with respect to the voltage constraints (with such decisions, the fifth term of the reward function in (12) will be zero). The RL agent will learn to choose such decisions, as violations of the voltage constraints at any of the buses will lead to low reward values due to the penalty term weighted by w_5.

6.2. Training Data

We have trained the RL agent using thousands of scenarios. (We assume the training data as given and studying different ways of constructing them is out of the scope of this work.) Each scenario consists of time series of power injections at all buses, i.e., of the loads and the PV generation at all buses over K time intervals. The PV generation time-series are constructed with random variations around an exemplary PV generation time series of a summer day via Monte Carlo. The load time series are similarly constructed via Monte Carlo variations around given consumption time series. Examples of the PV and consumption time series that constitute the scenarios are shown in Figures 5 and 6, correspondingly.

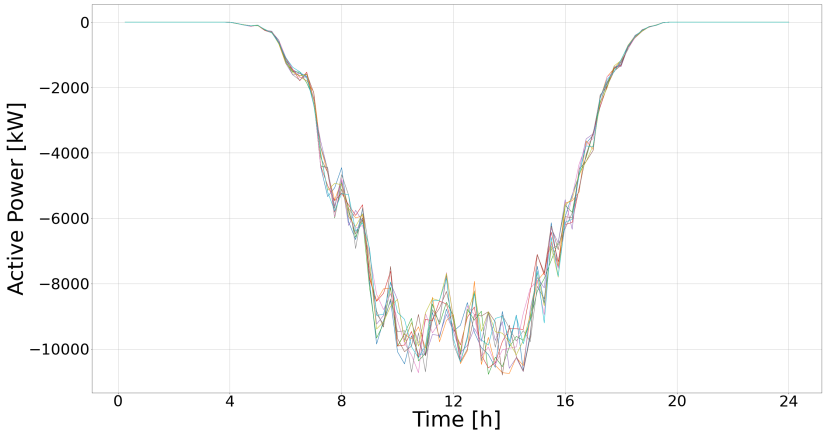

Figure 5. Examples of PV generation time series used for the construction of the training scenarios for the RL agent.

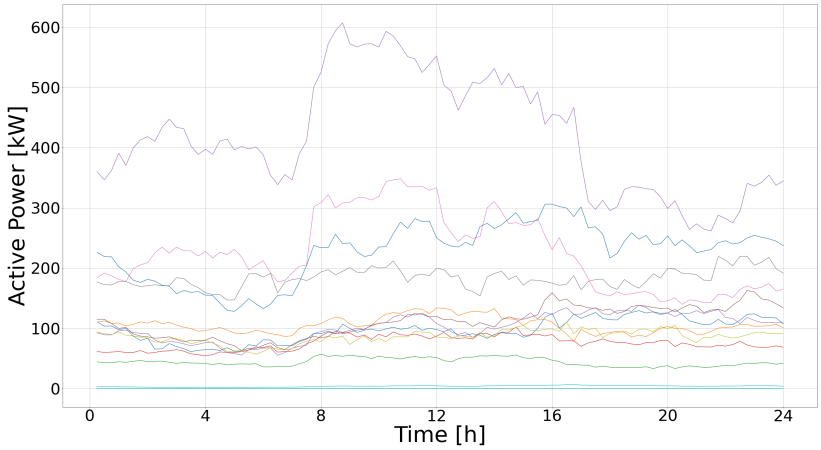

Figure 6. Examples of consumption time series used for the construction of the training scenarios for the RL agent.

6.3. Training of the RL Agent and Dispatch Plan Evaluation

We have trained the RL agent (Algorithm 1) with the settings listed in Table 2. The learning and neural network parameters are obtained via experimentation. The weights in the reward function are set as follows: $w_1 = 100$, $w_2 = 1$, $w_3 = 1$, $w_4 = 3000$, $w_5 = 1$, $w_6 = 1$, $w_7 = 100$.

Table 2. Training parameters for Algorithm 1.

Descripotion	Value
D	30,000 scenarios
Actor learning rate	0.00001
Critic learning rate	0.0001
Discount factor (γ)	0.99
Target soft-update parameter (τ)	0.001
Actor neural network number of layers	2
Critic neural network number of layers	3
Neural networks hidden layer size	128
Activation function	ReLu
Batch size for learning	2048 scenarios
K	96 time intervals
Δt	15 min

After training the agent, we have used Monte Carlo to construct 80 scenarios of similar shape (but not identical) to the ones used for training and we used them as input ($\mathcal{D}_{day}$) to Algorithm 2 for the computation of the dispatch plan. According to Algorithm 2, the final dispatch plan derives as a per time interval average of the 80 dispatch plans created based on the 80 scenarios and the trained RL agent. Figure 7 shows the obtained dispatch plan in comparison with the dispatch plan given by CoDistFlow. The two dispatch plans are very close to each other. However, Algorithm 2 is much faster than CoDistFlow. In particular, computing a dispatch plan with 80 scenarios and Algorithm 2 takes less than 2 min, whereas, with CoDistFlow, it takes around 5 times more (around 9 min) [18]. In addition, note that the RL agent is trained with thousands of historical scenarios on the uncertainty which cannot be considered with scenario-based optimization.

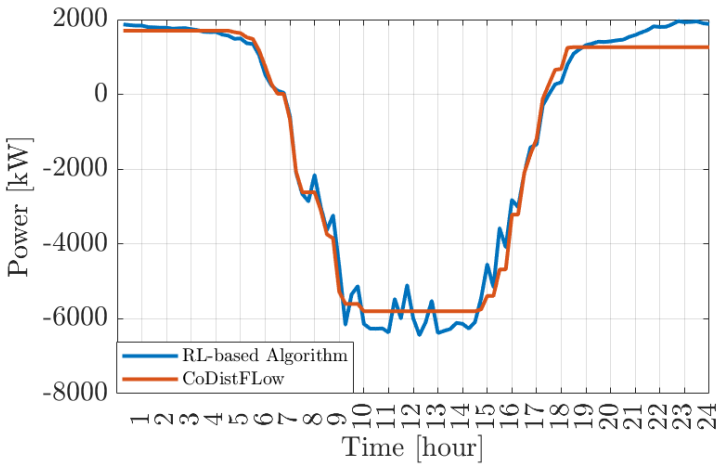

Figure 7. Dispatch plan obtained by Algorithm 2 with 80 scenarios as an input (i.e., in $\mathcal{D}_{day}$).

For better comparison purposes, we have applied a real-time controller with the goal to control the battery power so as to follow the dispatch plan at the PCC during network operation while the realization of the uncertain prosumption is revealing. To do so, it solves the following optimization problem at each time k:

$$\min_{S_0(k)} \left\| S_0(k) - S^{DP}(k) \right\|^2 \tag{17}$$

$$\text{s.t.} : (2) - (7) \ \forall i \tag{18}$$

The objective function expresses the error for not following the dispatch plan. This problem has the same constraint set as (9), but it considers a single time interval (k) and a single scenario that corresponds to the realized prosumption (consumption loads and PV generation). $S^{DP}(k)$ is given as an output of Algorithm 2 and thus it is not an optimization variable. Problem (19) is non-convex and we apply the CoDistFlow method of [5] to solve it. Note that we use this real-time controller just for evaluation purposes and we do not propose any new design for the real-time control. As in [5] and [18], we use the *daily dispatch plan tracking error* to assess the quality of the computed dispatch plan. It is defined as follows:

$$M_R = \Delta t \sum_{k=1}^{K} \left\| S_0(k) - S^{DP}(k) \right\| \tag{19}$$

i.e., as the daily sum of the absolute differences of the dispatched power and the realized power at the PCC multiplied by the time interval duration (Δt). Hence, lower values indicate better performance. We have tested the dispatch plan obtained by Algorithm 2 against different prosumption realization scenarios and computed the M_R for each one of them. Two different sets of prosumption realization scenarios are created via Monte Carlo

and tested; one derived by multiplying each value of the scenarios in $\mathcal{D}_{day}$ by a different random factor between 0.9 and 1.1, and the second derived similarly but with using random factors between 0.95 and 1.05. Figure 8 shows the histogram of the M_R values. We observe that the mismatch is very low for most of the scenarios, indicating that the dispatch plan could be generally followed in real time. Comparing the histogram with the results of CoDistFlow in Figure 9e of [5], it becomes clear that the RL algorithm achieves a similar quality. Figure 9 indicates the few time intervals during which the algorithm is unable to follow the dispatch plan by showing the M_R values for all time intervals averaged over all scenarios. In detail, the dispatch plan cannot be accurately followed at around 4:00 p.m., which is the time when the PV production is decreasing. Furthermore, Figure 9 shows that the different scaling factors do not have much influence on the overall quality of the dispatch plan.

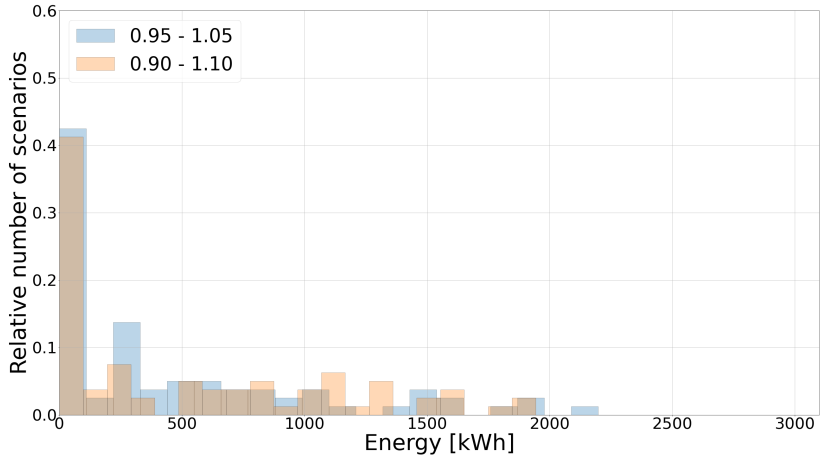

Figure 8. Histogram of M_R values for all prosumption realization scenarios.

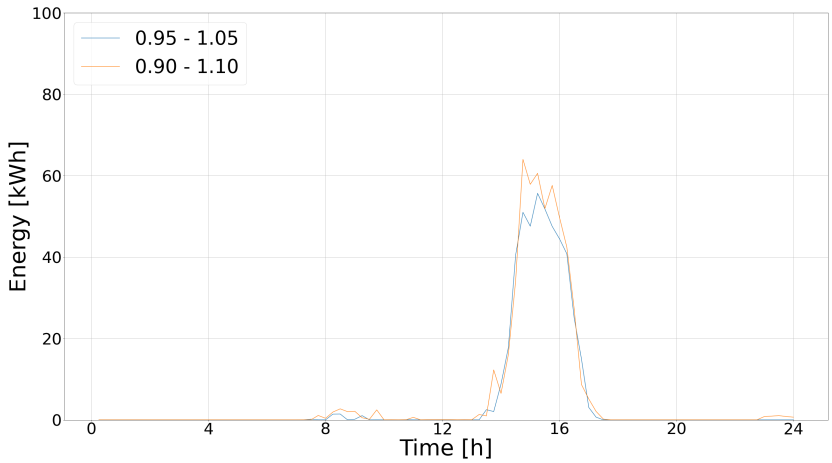

Figure 9. Average M_R value over all scenarios for all time intervals.

6.4. Testing the Trained Agent with Input Data That Diverge from Those Used for Training

In this section, we study the performance of the trained RL agent of Section 6.3 (as produced by Algorithm 1) in computing dispatch plans for scenario sets $\mathcal{D}_{day}$ that diverge from those used for training in $\mathcal{D}$. In particular, we examined two cases, namely, (i) shifting the PV generation time series backwards and forwards in time by one hour, and (ii) up and down-scaling the PV generation time series by 20%.

Note that, when having an RL agent trained with day-ahead PV forecasts, errors in the realized PV generation compared to the forecasts are expected and a deviation order up to 20% can be considered realistic (see Figure 10 of [36]). Shifting the PV generation

time series by one hour can introduce errors of more than 20% for several time intervals, e.g., in the early morning and late afternoon hours (see Figure 5), so it accounts for enough deviation from the training data and therefore we did not study shifting for more hours.

6.4.1. Time Shift of the PV Generation Time Series

First, the PV generation time series that were part of the 80 scenarios used in Section 6.3 to compute the dispatch plan are shifted backwards by one hour (Here, with backwards, we mean e.g., from 12:00 to 11:00 and forwards vice versa.) The new scenarios are given again as input to Algorithm 2, and a new dispatch plan is produced. The same procedure is repeated when shifting the PV generation time series by one hour forwards. The new dispatch plans are depicted in Figure 10 from which it can be observed that the agent is able to track the time shift in the prosumption scenarios and adapt accordingly the produced dispatch plan. We also computed the M_R values for each new dispatch plan, the histograms of which are shown in Figures 11 and 12. The daily dispatch plan tracking errors are in general very low, and the performance of the agent is of similar quality as in the original case in Section 6.3. Thus, we can conclude that the RL agent is able to handle input data that are time shifted compared to the training data.

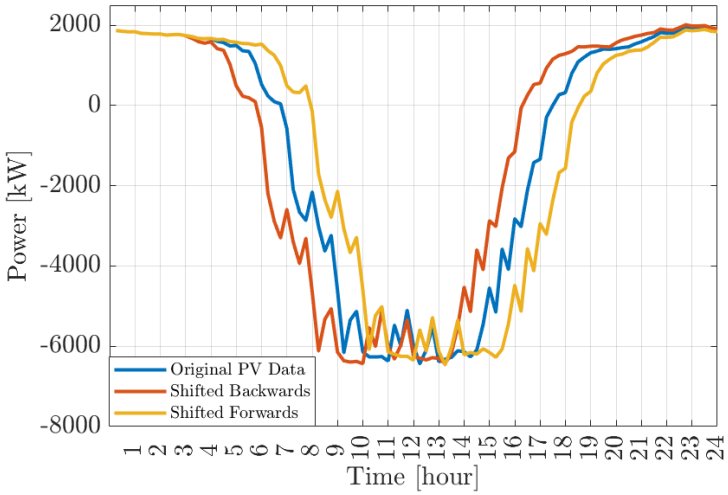

Figure 10. Dispatch plan with shifted PV generation time series forwards and backwards.

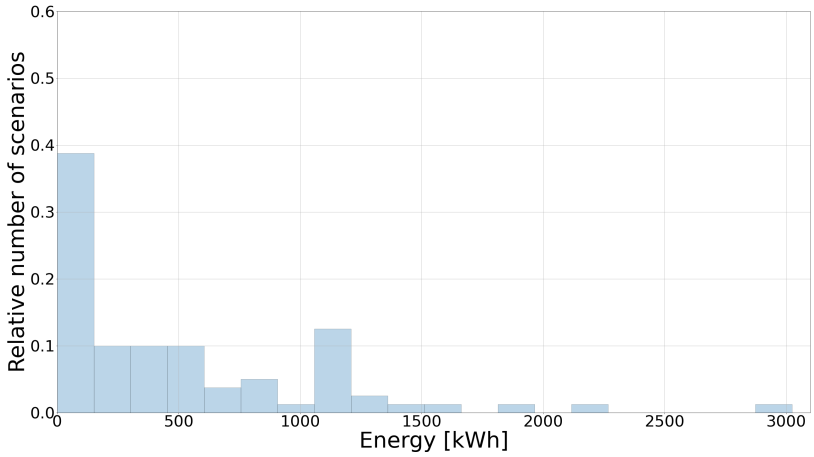

Figure 11. Histogram with PV generation time series shifted backwards by one hour.

6.4.2. Scaling of the PV Generation Time Series

First, the PV generation time series included in the 80 scenarios used in Section 6.3 to compute the dispatch plan are scaled by a factor of 0.8, i.e., the magnitude of the PV

generation time series has been reduced by 20% (scaled down). The new scenarios are given as input to Algorithm 2, and a new dispatch plan is produced. The same procedure is repeated when scaling the PV generation time series by a factor of 1.2, i.e., increasing the PV generation magnitude by 20% (scaled up).

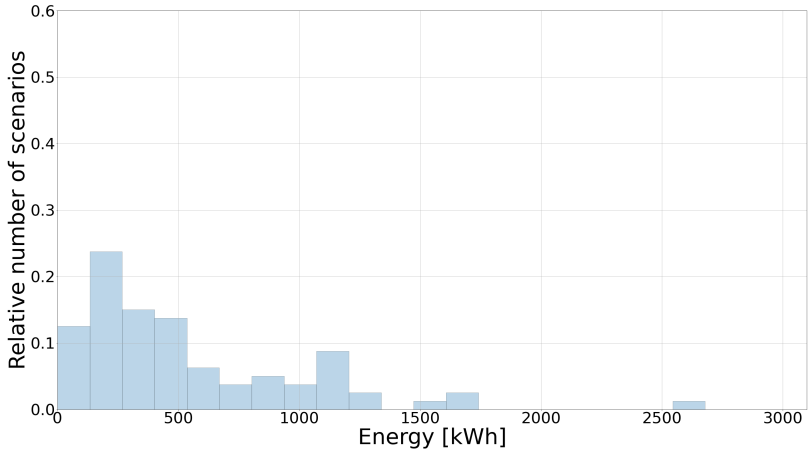

Figure 12. Histogram with PV generation time series shifted forwards by one hour.

The new dispatch plans are depicted in Figure 13. We also computed the M_R values for each new dispatch plan, the histograms of which are shown in Figures 14 and 15. While the performance of the RL agent is satisfactory when the PV generation time series are scaled down (i.e., their magnitude is decreased by 20%), as observed in Figure 14, the trained agent is no longer able to create a correct dispatch plan when the PV scenarios are scaled by a factor of 1.2 as concluded from the increased M_R values in Figure 15. To analyze this result, we show the average M_R values over time in Figure 16 for the increased PV generation. We can observe that the M_R values are significantly higher than those in Figure 9 starting from 12:00 p.m. up to around 5:00 p.m., which coincides with half of the period of higher than expected PV generation. This is caused by the fact that, after noon, the battery is already fully charged and cannot absorb the non-expected surplus energy from the PV. This already occurs in the case shown in Figure 9 but in much smaller scale than in Figure 16.

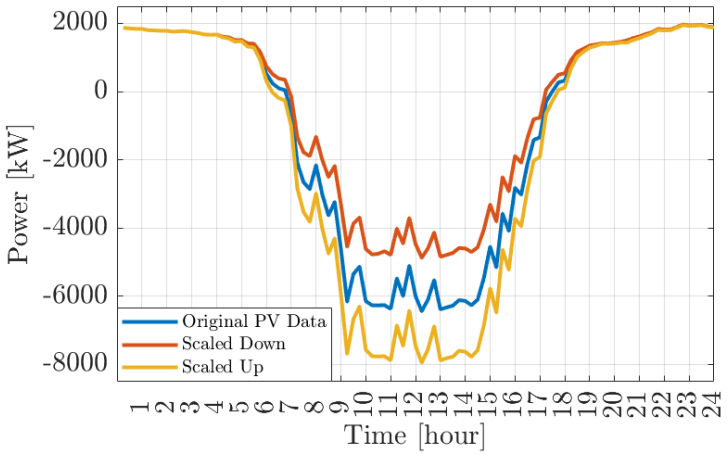

Figure 13. Dispatch plan with scaled PV generation time series.

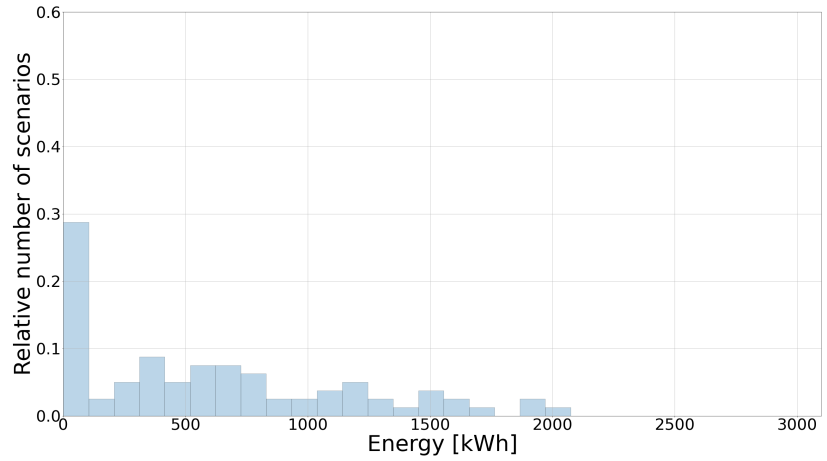

Figure 14. Histogram with PV time series magnitude decreased by 20%.

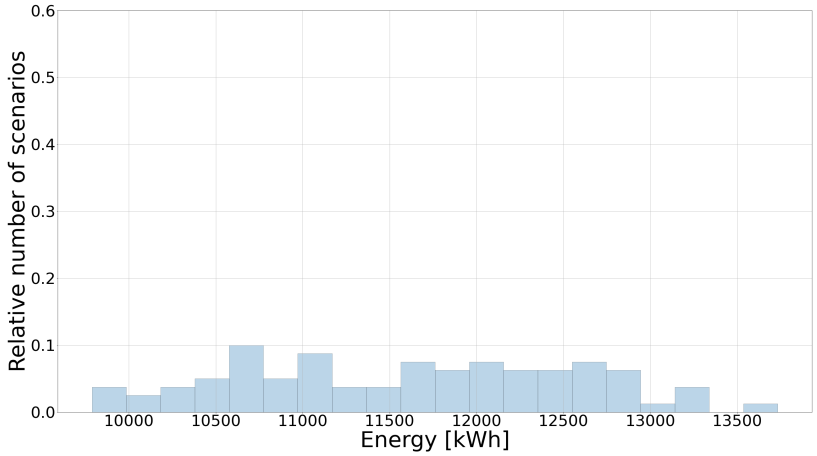

Figure 15. Histogram with PV time series magnitude increased by 20%.

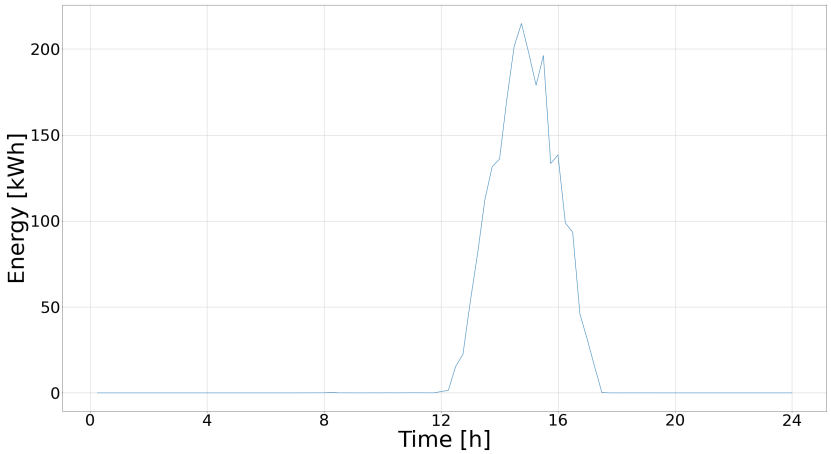

Figure 16. Average M_R over time with PV scenarios scaled up by a factor of 1.2.

6.4.3. Training a New RL Agent with Scaled PV Generation Time Series

To solve the issue raised in Section 6.4.2, we train a new agent using both the original scenarios and the scaled ones. The training parameters (Table 2) and the weight values in the reward function are kept the same as in the previous simulations. In particular, the PV generation time series used for training are shown in Figure 17 and are derived by scaling the PV generation time series of Figure 5 with a random factor between 0.5 and 1.5.

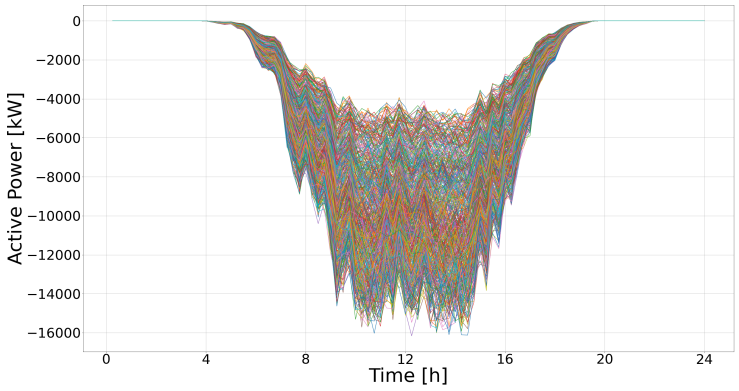

Figure 17. PV generation time series scaled with a random factor between 0.5 and 1.5.

With the newly trained agent, we again computed the dispatch plans when increasing/decreasing the magnitude of the PV generation time series by 20% as in Section 6.4.2, which are depicted in Figure 18. Then, the dispatch plans were evaluated with the real-time controller to obtain the histograms of the daily dispatch plan tracking errors M_R. Figures 19 and 20 show the histograms of the the M_R values for PV generation scaled by a factor of 0.8 and a factor of 1.2, respectively. As it can be seen, the newly trained agent is now able to create accurate dispatch plans (the M_R values are very small) also for the case when the PV generation is increased by a factor of 1.2. If comparing the dispatch plans for the increased PV generation in Figures 13 and 18, we can observe that the one using the newly trained agent, i.e., shown in Figure 18, exports more power to the main grid than the previously trained agent (i.e., it reaches values much lower than -8 MWh, which is the lowest power exported by the dispatch plan in Figure 13). Thus, it accounts for the extra PV generation and plans to export it to the main grid since the battery cannot handle extra charging. On the contrary the dispatch plans for the scaled down PV generation are very similar in both cases. Thus, when training with diverse PV trajectory shapes, the RL agent adapts satisfactorily in a larger range of PV input data compared to training with a single PV trajectory shape.

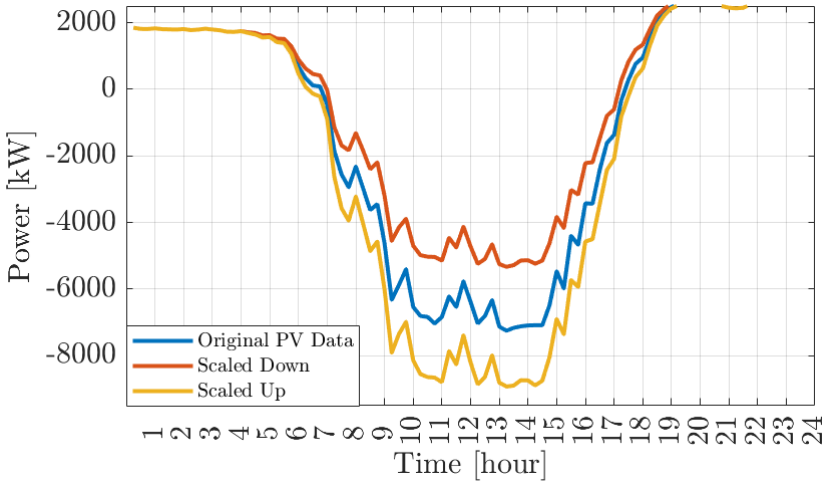

Figure 18. Dispatch plan with scaled PV generation time series and re-trained RL agent.

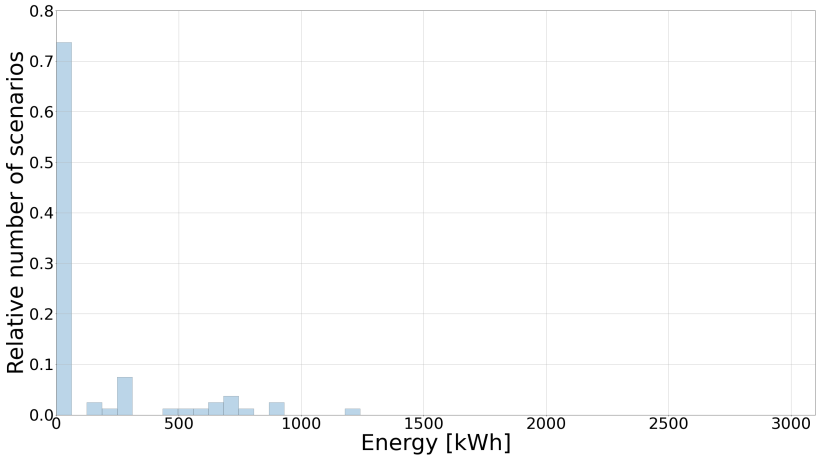

Figure 19. Histogram of M_R values for PV generation decreased by a factor of 0.8 and re-trained RL agent.

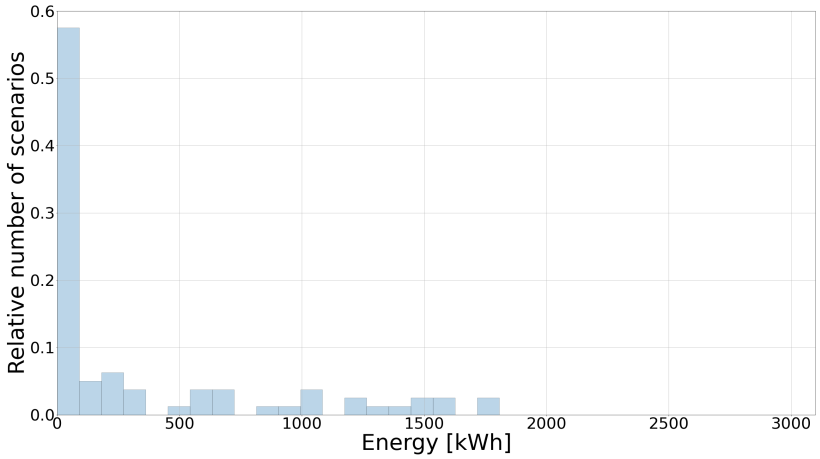

Figure 20. Histogram of M_R values for PV generation increased by a factor of 1.2 and re-trained RL agent.

7. Discussion

In this paper, we presented a novel RL-based algorithm for computing dispatch plans in a distribution grid with RESs and batteries. The proposed algorithm works iteratively and calls a trained RL agent at every iteration. The feedback provided to the agent is appropriately estimated to allow for an open loop application. The numerical evaluations and comparisons with the conventional scenario-based optimization approach show the effectiveness of our algorithm in solving the dispatching problem. In particular, the computed dispatch plan is similar to the dispatch plan obtained by scenario-based optimization, but our algorithm is computationally significantly more efficient.

Regarding deviations of the realized PV generation from the training data for certain points in time, we can make the following two conclusions based on our simulation results. First, if the deviation is due to shifting, i.e., the deviation is compensated in the other direction at another point in time, then the trained RL agent will be able to handle it. If it is due to up-scaling so that the PV generation is higher than expected at certain points in time but not less than expected at every other point, then re-training the RL agent will be required.

Due to its computation speed and possibility to adapt to unseen input data in some cases, we believe that our proposed scheme will be very useful also for faster scales of computing dispatch plans with updated forecasts, e.g., intra-day over minutes. In such time

scales, it is not always possible to apply the conventional optimization-based approaches that impose computation time limitations [18].

One issue of the applied DDPG algorithm relates to the stability of the obtained trained RL agent in terms of quality. More specifically, using the DDPG algorithm to train several RL agents with the same parameters and training data might lead to trained RL agents that produce dispatch plans of different quality. In this work, to tackle this issue, we have trained several RL agents with the same parameters and training data in parallel and chosen the one giving the best dispatch plans among them. In our future work, we will investigate the use of other RL algorithms such as the soft actor critic (SAC) [37] that are developed with the aim to solve the stability issues of the DDPG algorithm.

Author Contributions: Conceptualization, E.S.; methodology, E.S. and J.S.; software, J.S.; validation, E.S. and J.S.; formal analysis, E.S. and J.S.; writing—original draft preparation, E.S.; writing—review and editing, E.S. and G.H.; visualization, E.S. and J.S.; supervision, E.S. and G.H.; project administration, E.S. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: The work is part of the Ortsnetz project which is supported by the pilot and demonstration program of the Swiss Federal Office of Energy SFOE.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. IEA. Renewables. 2021. Available online: https://www.iea.org/reports/renewables-2021 (accessed on 19 October 2021).
2. Joskow, P. Comparing the Costs of Intermittent and Dispatchable Electricity Generating Technologies. *Am. Econ. Rev.* **2011**, *101*, 3. [CrossRef]
3. Bozorg, M.; Sossan, F.; Boudec, J.-Y.L.; Paolone, M. Influencing the Bulk Power System Reserve by Dispatching Power Distribution Networks Using Local Energy Storage. *Electr. Power Syst. Res.* **2018**, *163*, 270–279. [CrossRef]
4. Electricity Advisory Committee. *Securing the 21st-Century Grid: The Potential Role of Storage in Providing Resilience, Reliability, and Security Services, Recommendations for the U.S. Department of Energy*; Technical Report; U.S. Department of Energy: Washington, DC, USA, 2018.
5. Stai, E.; Reyes-Chamorro, L.; Sossan, F.; Boudec, J.-Y.L.; Paolone, M. Dispatching Stochastic Heterogeneous Resources Accounting for Grid and Battery Losses. *IEEE Trans. Smart Grid* **2018**, *9*, 6522–6539. [CrossRef]
6. Campi, M.C.; Calafiore, G. Decision Making in an Uncertain Environment: The Scenario-based Optimization Approach. In *Working Paper*; Universita di Brescia: Brescia, Italy, 2004.
7. Pinson, P.; Madsen, H.; Nielsen, H.A.; Papaefthymiou, G.; Klöckl, B. From Probabilistic Forecasts to Statistical Scenarios of Short-term Wind Power Production. *Wind Energy* **2009**, *12*, 51–62. [CrossRef]
8. Schmidli, J.; Roald, L.; Chatzivasileiadis, S.; Andersson, G. Stochastic AC Optimal Power Flow with Approximate Chance-Constraints. In Proceedings of the 2016 IEEE Power and Energy Society General Meeting (PESGM), Boston, MA, USA, 17–21 July 2016; pp. 1–5.
9. Dall'Anese, E.; Baker, K.; Summers, T. Chance-Constrained AC Optimal Power Flow for Distribution Systems With Renewables. *IEEE Trans. Power Syst.* **2017**, *32*, 3427–3438. [CrossRef]
10. Li, N.; Chen, L.; Low, S.H. Exact Convex Relaxation of OPF for Radial Networks Using Branch Flow Model. In Proceedings of the 2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm), Tainan, Taiwan, 5–8 November 2012; pp. 7–12.
11. Farivar, M.; Clarke, C.R.; Low, S.H.; Chandy, K.M. Inverter VAR Control for Distribution Systems with Renewables. In Proceedings of the 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), Brussels, Belgium, 17–20 October 2011; pp. 457–462.
12. Farivar, M.; Low, S.H. Branch Flow Model: Relaxations and Convexification—Part I. *IEEE Trans. Power Syst.* **2013**, *28*, 2554–2564. [CrossRef]
13. Nick, M.; Cherkaoui, R.; Boudec, J.-Y.L.; Paolone, M. An Exact Convex Formulation of the Optimal Power Flow in Radial Distribution Networks Including Transverse Components. *IEEE Trans. Autom. Control* **2018**, *63*, 682–697. [CrossRef]
14. Yuan, Z.; Hesamzadeh, M.R.; Biggar, D.R. Distribution Locational Marginal Pricing by Convexified ACOPF and Hierarchical Dispatch. *IEEE Trans. Smart Grid* **2018**, *9*, 3133–3142. [CrossRef]
15. Wei, W.; Wang, J.; Li, N.; Mei, S. Optimal Power Flow of Radial Networks and Its Variations: A Sequential Convex Optimization Approach. *IEEE Trans. Smart Grid* **2017**, *8*, 2974–2987. [CrossRef]
16. Baran, M.; Wu, F.F. Optimal Sizing of Capacitors Placed on a Radial Distribution System. *IEEE Trans. Power Deliv.* **1989**, *4*, 735–743. [CrossRef]
17. Huang, S.; Filonenko, K.; Veje, C.T. A Review of The Convexification Methods for AC Optimal Power Flow. In Proceedings of the 2019 IEEE Electrical Power and Energy Conference (EPEC), Montreal, QC, Canada, 16–18 October 2019; pp. 1–6.

18. Stai, E.; Sossan, F.; Namor, E.; Boudec, J.-Y.L.; Paolone, M. A Receding Horizon Control Approach for Re-dispatching Stochastic Heterogeneous Resources Accounting for Grid and Battery Losses. *Electr. Power Syst. Res.* **2020**, *185*, 106340. [CrossRef]
19. Molzahn, D.K.; Hiskens, I.A. Convex Relaxations of Optimal Power Flow Problems: An Illustrative Example. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *63*, 650–660. [CrossRef]
20. Ghaddar, B.; Marecek, J.; Mevissen, M. Optimal Power Flow as a Polynomial Optimization Problem. *IEEE Trans. Power Syst.* **2016**, *31*, 539–546. [CrossRef]
21. Giraldo, J.S.; Vergara, P.P.; López, J.C.; Nguyen, P.H.; Paterakis, N.G. A Linear AC-OPF Formulation for Unbalanced Distribution Networks. *IEEE Trans. Ind. Appl.* **2021**, *57*, 4462–4472. [CrossRef]
22. Hohmann, M.; Warrington, J.; Lygeros, J. Optimal Linearizations of Power Systems With Uncertain Supply and Demand. *IEEE Trans. Power Syst.* **2019**, *34*, 1504–1512. [CrossRef]
23. Bakirtzis, A.G.; Biskas, P.N.; Zoumas, C.E.; Petridis, V. Optimal Power Flow by Enhanced Genetic Algorithm. *IEEE Trans. Power Syst.* **2002**, *17*, 229–236. [CrossRef]
24. Baptista, E.C.; Belati, E.A.; da Costa, G.R.M. Logarithmic Barrier-augmented Lagrangian Function to the Optimal Power Flow Problem. *Int. J. Electr. Power Energy Syst.* **2005**, *27*, 528–532. [CrossRef]
25. Römisch, W. Scenario Reduction Techniques in Stochastic Programming. In *Stochastic Algorithms: Foundations and Applications. SAGA. Lecture Notes in Computer Science*; Watanabe, O., Zeugmann, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5792. [CrossRef]
26. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2015.
27. Vandael, S.; Claessens, B.; Ernst, D.; Holvoet, T.; Deconinck, G. Reinforcement Learning of Heuristic EV fleet Charging in a Day-ahead Electricity Market. *IEEE Trans. Smart Grid* **2015**, *6*, 1795–1805. [CrossRef]
28. Shang, Y.; Wu, W.; Guo, J.; Ma, Z.; Sheng, W.; Lv, Z.; Fu, C. Stochastic Dispatch of Energy Storage in Microgrids: An Augmented Reinforcement Learning Approach. *Appl. Energy* **2020**, *261*, 114423. [CrossRef]
29. Ruelens, F.; Claessens, B.J.; Vandael, S.; Schutter, B.D.; Babuška, R.; Belmans, R. Residential Demand Response of Thermostatically Controlled Loads Using Batch Reinforcement Learning. *IEEE Trans. Smart Grid* **2017**, *8*, 2149–2159. [CrossRef]
30. Shi, Y.; Mu, C.; Hao, Y.; Ma, S.; Xu, N.; Chong, Z. Day-ahead Optimal Dispatching of Hybrid Power System Based on Deep Reinforcement Learning. In *Cognitive Computation and Systems*; Wiley: Hoboken, NJ, USA, 2022; pp. 1–11. [CrossRef]
31. Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971.
32. Dhople, S.V.; Guggilam, S.S.; Chen, Y.C. Linear Approximations to AC Power Flow in Rectangular Coordinates. In Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 29 September–2 October 2015; pp. 211–217. [CrossRef]
33. Karagiannopoulos, S.; Dobbe, R.; Aristidou, P.; Callaway, D.; Hug, G. Data-driven Control Design Schemes in Active Distribution Grids: Capabilities and Challenges. In Proceedings of the 2019 IEEE Milan PowerTech, Milano, Italy, 23–27 June 2019; pp. 1–6.
34. Powell, W. Approximate Dynamic Programming: Solving the Curses of Dimensionality. In *Wiley Series of Probability and Statistics*, 2nd ed.; Wiley & Sons: Hoboken, NJ, USA, 2011.
35. Mnih, V. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
36. El-Baz, W.; Tzscheutschler, P.; Wagner, U. Day-ahead Probabilistic PV Generation Forecast for Buildings Energy Management Systems. *Sol. Energy* **2018**, *171*, 478–490. [CrossRef]
37. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.