Microgrid Control Under Uncertainty: A Preprint

Avishai Halev[1], **Yongshuai Liu**[2], **Xin Liu**[2]
[1]Department of Mathematics
University of California Davis
Davis, CA, USA
[2]Department of Computer Science
University of California Davis
Davis, CA, USA
ahalev@ucdavis.edu, yshliu@ucdavis.edu, xinliu@ucdavis.edu

Abstract

Microgrids – decentralized electrical grids that can function both in conjunction with wide area macrogrids and without – are a powerful tool to address energy resiliency and climate change mitigation. Microgrid control, however, remains a challenge; their bespoke nature and the existence of multiple sources of uncertainty lead to a control problem that traditional grid modeling and control techniques are ill-suited to handle. In this work, we analyze three different approaches to the microgrid control problem: rule-based control, model predictive control, and reinforcement learning in the context of forecast uncertainty and model uncertainty. We utilize random network distillation and domain randomization to support reinforcement learning in the context of uncertainty and show that reinforcement learning algorithms are able to achieve performance comparable to model predictive control algorithms and superior to rule based control algorithms in scenarios with no uncertainty and outperform model predictive control algorithms under model uncertainty. We present an in-depth analysis of algorithm performance on one microgrid and a high-level overview of additional microgrids and build a simulator for future research.

1 Introduction

Driven by the urgency of climate change, the energy sector is faced with the necessity of rapid transformation in order to continue to meet energy needs while meeting ambitious sustainability targets. Microgrids – clusters of distributed energy resources, including local electricity generation, consumption, and storage capacity – have emerged as a powerful tool in this revolution due to their ability to increase energy system efficiency by allowing energy generation and consumption to be located in closer proximity. Microgrids also aid system resiliency and reliability by providing alternative sources of energy during macrogrid failures.

However, significant challenges persist before widespread adoption of microgrid technology is feasible. Foremost among them is the difficulty in controlling microgrids efficiently and effectively. Traditional microgrid control models such as model predictive control (MPC) are model-based, requiring explicit models of the various microgrid components as well as accurate load and production forecasting (Bordons, Garcia-Torres, and Ridao 2020). MPC approaches can also be computationally intensive, involving the frequent solution of large optimization problems (Lamburn, Gibbens, and Dumble 2014).

Due to their small scale, microgrids encounter significant volatility compared to wide-area macrogrids. This volatility manifests in multiple ways, including dynamic electrical market prices, demand and photovoltaic (PV) prediction uncertainty, and modeling uncertainty in microgrid components such as batteries (Nakabi and Toivanen 2021). Stochastic MPC approaches attempt to manage this problem by generating trajectories and utilizing their empirical mean for forecasting (Zhang et al. 2018).

Reinforcement learning approaches have recently emerged as a promising solution to the microgrid control problem under uncertainty. In (François-Lavet et al. 2016), a convolutional neural network architecture was used as a Q-learner in a discrete action space environment; both current and previous state information is passed to the agent in order to extract meaningful features from time-series. Zeng et al. treat the microgrid control problem as a finite-horizon Markov decision process (MDP) and use approximate dynamic programming in conjunction with a recurrent neural network to make decisions under uncertainty (Zeng et al. 2019). In the continuous control space, Guo et al. use proximal policy optimization algorithms (PPO) as a model-free method of microgrid control, while Liu et al. show that deep deterministic policy gradient (DDPG) algorithms are effective in a microgrid with stochasticity without direct modeling of the uncertainty (Guo et al. 2022; Liu et al. 2023). These works broach the problem of uncertainty handling in microgrid control by applying reinforcement learning algorithms but do not make specific modifications to directly enhance performance under uncertainty.

We expand on the above and explicitly lay out conditions for efficacy of reinforcement learning algorithms in electrical microgrids of increasing uncertainty, using techniques tailored for stochastic scenarios. We utilize forecasts of varying lengths and accuracies as well as battery models of varying accuracy. In addition to usage of PPO and DDPG, we incorporate domain randomization (DR) and random network distillation (RND) to encourage exploration and generalization.

Overall, we quantify different levels of microgrid uncertainty and forecast lengths and compare reinforcement learning, model predictive control and rule based control algo-

rithm across these levels. Our main contributions are as follows:

1. *Microgrid control overview*: we consider three methods of controlling microgrids – rule-based control (RBC), MPC, and RL – and show that reinforcement learning outperforms traditional algorithms in high-uncertainty scenarios.
2. *Performance under model uncertainty*: we analyze these methods of controlling microgrids under battery model uncertainty and show that RL models overtake MPC in performance as uncertainty increases.
3. *Performance under forecast inaccuracy*: we consider forecasts of varying accuracy and show that while forecast-dependent MPC grow inaccurate as forecast inaccuracy increases, this effect is minor due to replanning.
4. *Modular microgrid simulator*: we build an open-source microgrid simulator that is able to simulate custom microgrids at the tertiary level with arbitrary combinations of microgrid components. This simulator is able to handle constraints internally, allowing algorithms to focus on economic optimization. Using this simulator, we present benchmarks for future research.

The paper is organized as follows. In Section 2, we introduce the microgrid control problem followed by a description of the microgrid simulation environment and sources of uncertainty in Section 3. In Section 4, we lay out the algorithms under consideration, including rule-based control, model predictive control, reinforcement learning, domain randomization, and random network distillation. We present our results in Section 5 and conclude with a discussion in Section 6.

2 Background

Microgrids are small, self-contained electrical grids that can operate both standalone or incecction with the overall macrogrid. Specifically, they are a combination of loads (energy demands) and distributed energy resources (electric generation units located near the source of demand) with an explicit electrical boundary, acting as a single controllable entity (Olivares et al. 2014). Microgrids generally contain the following components:

1. *Local generation*: local sources of energy, such as on-site solar panels, wind turbines or gas generators (gensets).
2. *Consumption*: sources of energy demand.
3. *Energy storage*: batteries or accumulators to ensure power quality and provide backup power.
4. *Point of common coupling*: the connection to the main grid. While many microgrids contain a point of common coupling, some do not; these islanded microgrids are commonly found in remote sites.

All of these components can be stochastic: batteries can charge or discharge at unpredictable rates, for example, and demand is variable. Stochasticity in the point of common coupling occurs when the overall macrogrid is unstable and prone to outages, in which case the macrogrid is referred to as a **weak** grid; macrogrids with no outages are considered **strong** grids. Microgrids generally serve a small communities or geographic areas and are typically powered by a combination of renewable energy sources and traditional fossil fuels.

In our work, we focus on the tertiary level of microgrid control at the hourly scale. At this level, the main concern is system level optimization and energy management with a secondary focus on economic efficiency and emissions. Controls manage the dispatch of energy generation components with the aim of optimizing economic cost while ensuring energy production and consumption remain balanced.

2.1 Rule-Based Control

Rule-based control (RBC), also referred to as expert or heuristic control, is a type of control strategy that uses a set of predefined rules or heuristics to determine the control action for a given process (Grosan and Abraham 2011). These rules can be seen as an expert-defined decision tree from which to make decisions. RBC is relatively simple to implement and has been used in a variety of applications, such as ship energy systems, geoscience, agricultural pest management, and sensor control (Leondes 2001).

RBC is a common choice for microgrid energy management (Fatin Ishraque et al. 2021; Shezan et al. 2021) due to its simplicity and interpretability, which is valuable in communicating information to stakeholders and regulators as well as detecting and addressing system issues. As rules can be modified to reflect changing conditions, RBC is suited for microgrid control, where the available energy sources and demand for power may vary over time. However, the number of rules scales with the number of potential input/-control combinations, and defining an appropriate rule set to handle changing conditions or unforeseen scenarios can become cumbersome or even impossible.

2.2 Model Predictive Control

Model Predictive Control (MPC) is a control strategy that uses a model of a process to predict its future behavior and optimally control its inputs in order to achieve a desired output (Schwenzer et al. 2021). This is done by optimizing the underlying model over a future window, applying one set of controls, iterating to the next step and repeating.

MPC is widely deployed in industrial control systems due to its ability to handle constraints and optimize performance over a future horizon (Qin and Badgwell 2003). In microgrid applications, studies have shown that MPC can save up to 30% of energy usage when compared to using traditional RBC methods (Dai, Liu, and Zhang 2020; Mirakhorli and Dong 2016). However, MPC depends heavily on a model and forecast for accurate results, and inaccurate models and forecasts can lead to underperformance (Lucia et al. 2014).

2.3 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning that focuses on training agents to make decisions based on maximizing a reward signal. In an RL problem, the agent interacts with an environment by taking actions and receiving feedback in the form of rewards or penalties depending

on the success or failure of those actions. Through this trial-and-error process, the agent learns to make better decisions over time (Sutton and Barto 2018).

RL's ability to learn in situations where analytic solutions are unavailable or environmental information can only be collected through interaction is a powerful paradigm, and RL has been successfully applied to a variety of domains (Naeem, Rizvi, and Coronato 2020). However, RL can be computationally intensive and sensitive to both hyperparameter and system stochasticity (Mahmood et al. 2018). In our work, we alleviate these issues in the context of microgrid control by utilizing DR and RND techniques, which encourage agent exploration and reduce sensitivity to underlying stochasticity.

Domain Randomization Generalization to unseen data – specifically, for real world transfer – is a task that is notoriously difficult in RL (Cobbe et al. 2019). In order to bridge this gap, we use domain randomization (DR) to allow the agent to experience a greater breadth of scenarios.

DR is based on a simple concept: an agent exposed to a wide variety of environments will be better prepared to adapt to an unseen scenario – such as one in the real world – than one trained repeatedly on a single simulated environment. This idea is accomplished by training on environments based on a ground-truth environment where certain situational parameters are randomized before each episode (Tobin et al. 2017). Agents are trained on trajectories that are collected from interaction with these randomized environments; once trained, they are able to generalize well to unseen tasks. More sophisticated domain randomization approaches involve sampling environmental parameters with external feedback (Chebotar et al. 2019).

Random Network Distillation We utilize Random Network Distillation (RND) in order to incentive curiosity-driven exploration in the training process (Burda et al. 2018). In particular, we aim to spur investigation of different strategies of battery utilization, as it is through these strategies that cost efficiencies can be uncovered.

RND augments the exploration process by encouraging the RL agent to seek out novel states and scenarios. It operates by maintaining two neural networks, a target and a predictor, and measuring the deviation between the fixed target and the variable predictor to determine the novelty of new states. This curiosity-driven exploration not only aids in learning a more robust and adaptive control policy but also significantly reduces the need for extensive and expensive real-world data collection.

3 Microgrids and Uncertainty

Our microgrid environments are built atop *pymgrid*, an open-source simulator for tertiary control (Henri et al. 2020). *pymgrid* ships with a built-in set of 25 microgrids scenarios for algorithmic testing and benchmarking. In our work, we present a deep-dive on one of these scenarios and benchmarks on ten additinal scenarios. In addition, we build upon the original simulator to build a fully modular and customizable simulator, able to handle a large variety of applications. To maximize accessibility for future users, full documentation of the project is available at pymgrid.readthedocs.io.

Scenarios	Grid	Genset	n_{act}
0, 4, 6	Strong	No	2
1, 8, 9	Weak	Yes	4
2, 3, 5, 7	No	Yes	3
10	Strong	Yes	4

Table 1: Architectures of the microgrid scenarios. The dimension of the action space is denoted by n_{act}.

3.1 Microgrid Simulation

The state transition discussed below is utilized for simulation of tertiary control. RBC and MPC algorithms interact with the microgrid directly via this state transition, while RL agents interact with an abstracted microgrid environment.

Microgrid State Transition The microgrid simulator is a composition of load, PV, macrogrid, genset, and battery components. Load and PV components serve as an interface for external load and PV information, with load components demanding energy and PV components supplying energy from the microgrid each hour based on their underlying timeseries. Macrogrid, genset, and battery components require user/agent input to determine the quantity of energy to consume or produce within respective limits on production or consumption. At each step, the agent provides an action defining the amount of energy for each macrogrid, genset, and battery component to consume or produce, with gensets also accepting a boolean value denoting whether the genset should power on or off:

$$A_t = \left(\Delta B^{(t)}_{\text{charge}}, D^{(t)}, G^{(t)}, \mathbb{I}^{(t)}_G\right) \tag{1}$$

where $\Delta B^{(t)}_{\text{charge}}$ denotes the amount the battery is charged or discharged, $D^{(t)}$ denotes the amount to import or export from the external macrogrid, $G^{(t)}$ denotes genset production, and $\mathbb{I}^{(t)}_G$ denotes the genset status modification. Note that some microgrids do not contain grids or gensets and thus omit the respective elements in the action space.

Given this control information, the microgrid undergoes its state transition process. Energy demand from the load module is determined, followed by the application of agent-provided controls to macrogrid, genset, and battery components. Finally, PV energy is collected to the extent necessary to balance energy production and consumption. If PV production exceeds the amount required to match production and consumption, it can be curtailed as necessary with any remaining excess production garbage-collected as energy overgeneration. If PV production is insufficient to allow energy equilibrium, the excess becomes loss load. Batteries transition via the linear transition function

$$B^{(t+1)} = B^{(t)} + \eta B_{\text{charge}} - \frac{1}{\eta} B_{\text{discharge}} \tag{2}$$

where $B^{(t)}$ is the current battery charge, $\eta \in (0, 1]$ is the battery efficiency, and B_{charge} and $B_{\text{discharge}}$ are charge/discharge amounts respectively. For our purposes, charging and discharging are mutually exclusive and one of B_{charge} and $B_{\text{discharge}}$ is zero at every transition.

At the conclusion of this process, successor state information consisting of details on the status of microgrid components is collected and returned to the agent. In practice, we include the net load (load less PV production), battery state of charge, and forecasted grid prices in the state space:

$$S_t = \left(NL^{(t)}, B^{(t)}_{\text{SOC}}, C^{(t)}_{D_0}, C^{(t)}_{D_{t+1}}, ..., C^{(t)}_{D_{t+H-1}}\right) \tag{3}$$

where $NL^{(t)}$ is the net load, $B^{(t)}_{\text{SOC}} = \frac{B^{(t)}}{B_{\text{capacity}}}$ is the battery state of charge, $C^{(t)}_{D_0}$ is the current grid import price, $C^{(t)}_{D_{t+i}}$ is the grid import price forecast i hours ahead at step t, and the forecast horizon H defines the number of steps to look ahead. We vary the number of grid import price elements in the forecast from zero ($H = 0$), which includes neither the current grid import price nor any forecasts, to 24 ($H = 24$), which includes the current import price and 23 steps of forecasting.

The reward R_t is computed and returned as the negative of the cost C_t of deploying all microgrid modules added to any penalties induced by overgeneration or loss load:

$$\begin{aligned} C_t = C_B \left|\Delta B^{(t)}_{\text{charge}}\right| + C^{(t)}_{D_0} \cdot DI^{(t)} + C_G \cdot G^{(t)} \\ + C_{LL} \cdot LL^{(t)} + C_O \cdot O^{(t)} \end{aligned} \tag{4}$$

$$R_t = -C_t \tag{5}$$

where $DI^{(t)} = D^{(t)} \cdot \mathbb{1}_{D^{(t)}>0}$ is the grid import amount, $LL^{(t)}$ is the loss load, $O^{(t)}$ is the overgeneration, and C_B, $C^{(t)}_{D_0}$, C_G, C_{LL} and C_O are battery usage, grid import price at time t, genset production, loss load and overgeneration costs, respectively. Note that grid exporting is ignored as all microgrids under consideration have macrogrid export prices of zero. See Section 1 of the Supplementary Materials for additional details on the microgrid state transition.

Naive Environment The most straightforward way to define the action space for an RL agent is to utilize the action space $\mathcal{A}$ consisting of actions as defined in (1). While intuitive, this formulation does not allow for RL algorithms to discover cost-competitive polices as detailed in Section 5.

Net Load Environment While the overall objective of microgrid control is to meet load demand while minimizing costs, this objective can be reduced, without loss of generality, to producing just enough to meet the step-wise net load $NL^{(t)}$. This is due to the fact that PV production has no marginal cost, which implies that optimal policies should request maximal PV consumption at every step.

We thus reformulate our action space to define actions relative to the net load. Specifically, the action space $\mathcal{A} = [-1, 1]^{n_{\text{act}}}$ consists of vectors containing elements a_i such that the respective energy request is $a_i NL^{(t)}$, where $NL^{(t)}$ is the net load at that step. Any genset status value updates remain unchanged. Specifically, our action space consists of actions $A_t = (a_0, ..., a_{n_{\text{act}}})$ such that the microgrid receives the vector

$$A_t = NL^{(t)} \left(a_B, a_D, a_G, {NL^{(t)}}^{-1} a_{\mathbb{I}_G}\right)^\top \tag{6}$$

as controls corresponding to the same values in (1). As in (1), macrogrid or genset components may be omitted depending on the microgrid architecture.

Slack Environment We further reformulate the action space by observing that effective policies must balance the energy production and consumption of macrogrid, genset and battery components with the net load, as failing to do so will lead to loss load or overgeneration penalties. These penalties can overwhelm second-order signals such as balancing the various controllable components for efficiency. We can alleviate the agent of the task of meeting the energy balance by treating one controllable component (the genset or grid) as a slack component. We remove it from the action space and have it satisfy any remaining energy excess or shortfall implicitly after processing the remaining controllable components, whose action are defined as above in (6). The energy request for the slack component is defined automatically as

$$A^{\text{slack}}_t = NL^{(t)} \left(1 - \sum_{a_i \in \mathcal{A}} a_i^{\text{energy}}\right) \tag{7}$$

where a_i^{energy} denotes net-load relative energy requests for the remaining components as in (6).

Table 1 overviews the microgrid scenarios we consider in this work. We present an in-depth analysis of scenario zero as well as benchmarks of our algorithm's performance on ten additional scenarios.

3.2 Reward scaling

Minimization of the cost (4) over the course of a particular time period is our overall objective, and it is intuitive to define a reward signal as the negative of the cost at each step as in (5). This definition, however, elides the fact that much of the magnitude of this reward signal is gratuitous in the slack environment formuation. In this environment, the loss load and overgeneration costs are uniformly zero and the cost of the action $A_t = \overrightarrow{0}$ (requesting nothing from controllable components) can be treated as a baseline as in this situation energy balance is met by the slack component only. As a result, we shape the reward by adding this baseline (positive) cost to the original (negative) reward:

$$R'_t = R_t + C^{(t)}_{\text{slack}} \max(NL^{(t)}, 0) \tag{8}$$

where $C^{(t)}_{\text{slack}}$ is the grid import price or genset production cost if the slack module is a grid or genset, respectively. This shaped reward is positive if the cost of running the microgrid is less than utilizing only the slack component and negative otherwise.

3.3 Sources of uncertainty

As detailed above, each microgrid contains a load, a source of PV, and a battery. Each of these are potential sources

of uncertainty in the underlying models for MPC: the first two through forecast inaccuracy, and the third through divergence in battery model and true battery behavior. Macrogrids are a further source of uncertainty in microgrids that contain them through stochasticity in forecasts of their import prices.

In this work, we consider the effects of three types of battery models, all based on the transition model (2): (1) an **ideal** model, whose transitions model the behavior in (2) exactly, (2) a **biased** model, whose transitions are based on a battery efficiency η^* that diverges from the true battery efficiency, (3) a **decay** model, whose transitions are based on a battery whose efficiency is constant in time, but whose true efficiency decays gradually: $\eta_{t+1} = (1-\gamma)\eta_t$ for some decay rate $\gamma \ll 1$, and (4) a **cycle decay** model, whose transitions are based on a battery whose efficiency is constant in time, but whose true efficiency decays gradually as a function of the number of cycles: $\eta_t = \eta_0 \cdot (1-\gamma)^{n_{\text{cycles}}}$ for some decay rate $\gamma \ll 1$, where n_{cycles} is the number of battery cycles the battery has undergone.

We simulate timeseries forecast uncertainty by adding noise to true future values, as follows. Suppose the microgrid is currently at time T, and the forecast horizon is H, such that we consider the MPC problem or RL state over the timesteps $\{T, T+1, ...T+H\}$. Let the true future values over this horizon be $v_{T_H}^{\text{true}} \in \mathbb{R}^H$. Then the simulated forecasted values are $v_{T_H}^{\text{sim}} = v_{T_H}^{\text{true}} + \varepsilon_{T_H}$, where $\varepsilon_{T_{H_i}} \sim \mathcal{N}(0, |\mu| \log(1+i))$, i.e. the noise i steps in the future is Gaussian with mean zero and standard deviation $|\mu| \log(1+i)$, with μ as the mean of the underlying timeseries. This simulated noise is added to load, PV, and macrogrid timeseries, as applicable in a given microgrid. An example of this noisy forecasting for the load is plotted in Supplementary Fig. S1.

4 Methods

4.1 RBC Formulation

We define an RBC control algorithm for tertiary control as the greedy deployment of microgrid components as follows. We meet the net load by utilizing the microgrid's components consecutively from lowest to highest marginal cost until energy production and consumption is balanced, resulting in energy production in the components if $NL^{(t)}$ is positive and consumption if negative. This RBC method is greedy as no future planning is done and conservation of energy for future timesteps is not considered in deployment, and the resultant algorithm is a decision tree dependent on the net load and the available components. An example of this tree for Scenario zero is available in Supplementary Fig. S2.

4.2 MPC Formulation

Our formulation of MPC is based on the receding horizon control implementation in (Borrelli, Bemporad, and Morari 2017). Define $\mathbf{u}_t$ and $\mathbf{x}_t$ as the controls for controllable components and the state at time t, respectively, analogous to actions and states in the RL formulation. We can write the step-wise cost (4) as a linear combination of $\mathbf{u}_t$ and $\mathbf{x}_t$:

$$C_t(\mathbf{x}, \mathbf{u}) = \mathbf{c}_{\mathbf{x},t}^\top \mathbf{x} + \mathbf{c}_{\mathbf{u},t}^\top \mathbf{u} \tag{9}$$

where $\mathbf{c}_{\mathbf{x},t}$ and $\mathbf{c}_{\mathbf{u},t}$ are marginal cost vectors for the state and control, respectively. Controls at time T are selected as the minimizer of the cost over the forecast horizon H:

$$\boldsymbol{u}_T^* = \underset{\boldsymbol{u}_T \dots \boldsymbol{u}_{T+H}}{\arg\min} \sum_{t=T}^{T+H} C(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{10}$$

subject to

$$\begin{aligned} A\boldsymbol{x}_t + B\boldsymbol{u}_t &\leq \boldsymbol{\alpha}, \\ M\boldsymbol{x}_t + N\boldsymbol{u}_t &= \boldsymbol{\beta} \quad \forall t \in \{T, ..., T+H\}, \end{aligned} \tag{11}$$

a set of constraints on the various controls and state components. The optimal controls at step T are then applied to the system, the state transition occurs, and the optimization problem is reset and solved over $[T+1, T+H+1]$. For full details on the cost function and constraints, see the supplementary materials and the accompanying code base.

4.3 RL: Policy gradient methods

Policy Gradient methods find solutions to continuous-action MDPs by modeling and optimizing the policy directly (Sutton and Barto 2018). The objective is to maximize performance of a policy $\pi(a|s, \theta)$ with respect to a performance measure $J(\theta)$ by taking gradient steps with respect to the policy parameter θ.

PPO Proximal Policy Optimization (PPO) (Schulman et al. 2017) is an on-policy algorithm that utilizes an objective intended to ensure that the policy changes from gradient steps are relatively small. The objective $J(\theta)$

$$\mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{12}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\,\text{old}}(a_t|s_t)}$ and A_t is the advantage function. PPO removes much of the variance in traditional policy gradient algorithms.

DDPG Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al. 2019) is an off-policy policy gradient method with an objective function based on Q-learning:

$$\mathbb{E}_t\Big[\big(Q(s_t, a_t; \theta_Q) - (r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}; \theta_\mu); \theta_Q))\big)^2\Big]$$

where $Q(\cdot, \cdot\, \theta_Q)$ is the Q-function and $\mu(\cdot, \theta_\mu)$ is the policy parameterized by θ_Q and θ_μ, respectively, and s_t, a_t, r_t, and γ denote states, actions, rewards, and the discount factor, respectively.

DR and RND We implement DR by adding Gaussian noise to the underlying timeseries of a given microgrid at the beginning of each episode. Upon commencement of each episode, new timeseries are generated as applicable for load, PV, and macrogrids by adding noise to the ground truth timeseries; the training process then proceeds in a standard fashion.

We implement RND by augmenting rewards collected at the end of each episode with an intrinsic reward. RND consists of two randomly initialized fully-connected neural nets,

	Baseline			Forecast Unc. (10%)		Biased Batt. (Over)		Biased Batt. (Under)		Decay Batt.	
H	RBC	MPC	RL	MPC	RL	MPC	RL	MPC	RL	MPC	RL
7	384	361	355	359	360	373	359	361	358	408	375
24	384	352	356	352	365	364	358	352	358	401	372

Table 2: Performance of algorithms under uncertainty with forecast horizons of seven and 24 as measured by cost on the evaluation set. Baseline refers to perfect modeling and forecasting, forecast uncertainty refers to inaccurate forecasting (10% error), biased battery refers to the situation when the recorded battery efficiency over or underestimates the true efficiency, and decay battery refers to when the battery capacity decays over time. Values are in thousands of dollars.

a target and a predictor, that take in environment observations as input and return an 128-dimensional encoding. The target net is fixed while the predictor net is trained to predict the output of the target; in this way, observations that result in large differences in target and predictor encodings are novel while small differences suggest these observations have been encountered previously. The mean squared difference in encodings is used as an intrinsic reward and the weighted sum of intrinsic and standardized extrinsic rewards are used as the reward signal to the algorithm.

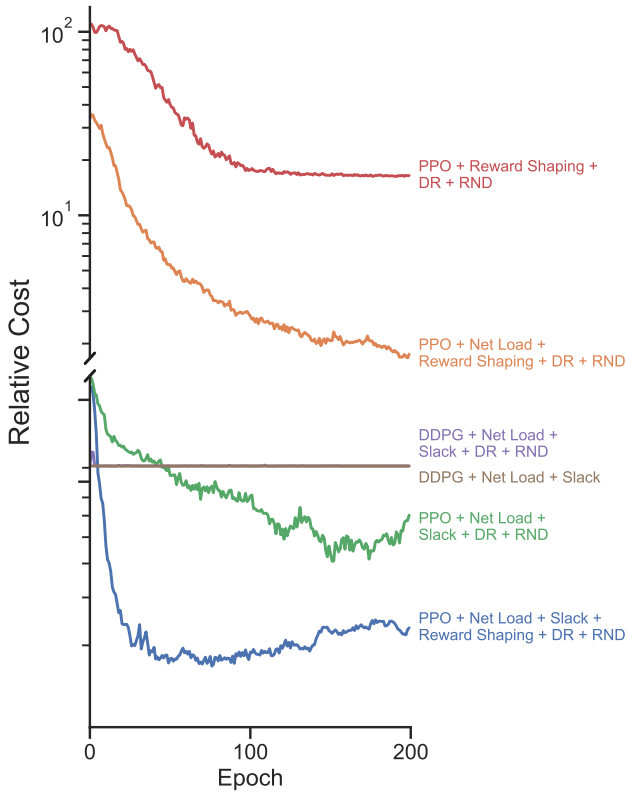

Figure 1: Comparison of environment formulations and RL algorithms. Environments are based on a 24-step forecast horizon. The relative cost is one less than the cost over the test set relative to using perfect MPC.

5 Results

In this section we present a relative comparison of algorithms. All evaluation is done on the final four months of data, consisting of 2920 hourly steps. RL algorithm training is done on 30-day episodes by randomly selecting time intervals of 720 steps from the first eight months of data.

5.1 Scenario zero

Scenario zero is a microgrid with a strong grid and no genset. With a perfect 23 step forecast and perfect battery model, RBC and MPC obtain test set costs of \$383,667 and \$351,555, respectively, making RBC performance 9.1% worse than MPC performance. We refer to this MPC algorithm, with a perfect 23 step forecast and a perfect battery model, as the **perfect MPC**, an algorithm that is effectively optimal given its environmental knowledge.

As mentioned in Section 3.1, we include varying amounts of grid import price data. It is through this information that both MPC and RL plan and from which cost efficiencies over RBC are achieved. Specifically, cost savings are realized when battery state of charge is conserved during times when import prices are low to then later discharge when import prices are high.

5.2 On-policy and off-policy learning

We find that off-policy learning via DDPG struggles to learn more than a simple policy, meeting but not exceeding RBC's performance with every combination of hyperparameters (Fig. 1). DDPG algorithms are sample efficient and quickly converge to the RBC cost but fail to surpass it. On-policy learning with PPO, on the other hand, is able to surpass RBC costs given the correct environment context. Specifically, PPO in slack environments is able to outperform RBC with baseline reward shaping increasing the rate of convergence and making the ultimate policy more effective.

The effectiveness of PPO in the slack environment holds over a wide array of hyperparameters and forecast horizons, with many converging to roughly the same cost of $\$358,996 \pm \$900$ (Supplementary Fig. S5). There are, however, some small outliers on the positive end: with forecast horizons of seven and 24, we are able to achieve costs of $\$355,344$ and $\$355,803$, with RND and domain randomization. These values are 1.1% and 1.2% worse than the perfect MPC model, respectively.

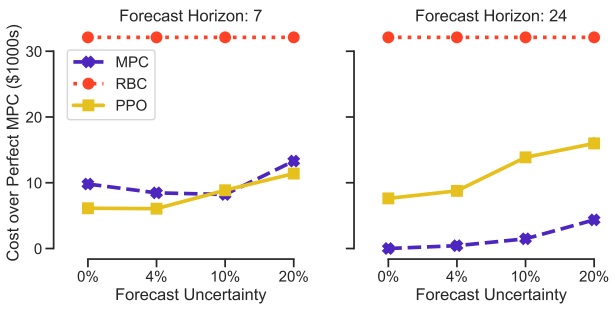

Figure 2: Algorithm performance under increasing forecast uncertainty. y-axis denotes the cost on the evaluation set with the given algorithm less the cost of perfect MPC.

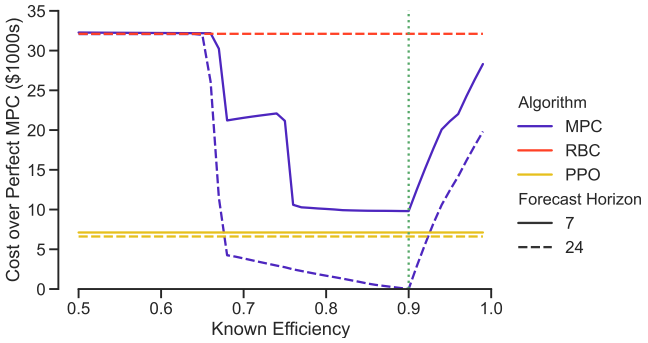

Figure 3: Performance with biased battery model. x-axis denotes efficiency known to MPC; true efficiency is 0.9 in all cases and is denoted by the dotted green line.

5.3 Planning

With a forecast horizon of seven, RL outperforms MPC in over half of all hyperparameter combinations. With this forecast horizon, MPC obtains a cost of $\$361,362$ on the test set while the average PPO policy achieves a cost of $\$360,082 \pm 2947$. DR and RND aid RL's performance: PPO models utilizing intrinsic reward weights of 0.01 or 0.1 and DR noise of 0.01 or 0.1 achieve costs of $\$358,514 \pm 368$.

It is clear that RL is able to leverage future forecast information to make efficient decisions in the present. The convergence, and to a certain extent performance, of RL increases steadily with increasing forecast horizon (Supplementary Fig. S5). The presence of RND mitages poor performance with short forecast horizons: with $H = 1$, higher intrinsic reward weights lead to sufficient exploration for the agent to exceed RBC performance by forcing the agent to explore different battery states and thus discover the value of conserving battery in certain scenarios. On the other hand, with a forecast horizon of 24, the highest intrinsic reward weights lead the agent to over-explore and performance suffers as a result.

Furthermore, environments with a forecast horizon of zero (no grid pricing information), converge, as DDPG algorithms do, to the RBC cost, with test set costs averaging $\$384,304 \pm 157$. This cost is strikingly close to the RBC cost: the difference is less than 0.17%. In fact, the RL policy converges almost identically to the RBC policy in this case (Supplementary Fig. S3). Both RBC and RL with a zero-length forecast horizon can be seen to charge the battery very rarely, with the battery SOC rarely exceeding four tenths of the capacity and spending most timesteps at the minimum. MPC and RL with longer forecast horizons, on the other hand, are more aggressive in their battery usage and the opposite pattern occurs: significant amounts of time are spent with the battery mostly charge, interspersed with periods of aggressive charging and discharging when the net load is high (Supplementary Fig. S3).

5.4 State Space Importance

We perform a permutation importance on the observation space to evaluate the extent of the RL agents' planning. Specifically, we evaluate the policy on the test set with one element of each observation replaced with a randomly selected number throughout the episode. By averaging over multiple episodes, we are able to discern decreases in performance due to this shuffling. Greater decreases in performance correspond to the RL agent inferring more from that component of the observation space.

The net load and current grid price are the most valuable features, corresponding to importances of over $\$12,000$ with forecast horizons of both seven and 24 (Supplementary Fig. S4). The battery state of charge is important to the $H = 24$ agent but not the $H = 7$ agent, suggesting that this information is only taken into account on longer time-horizons when planning for a farther future is feasible. Agents in both $H = 7$ and $H = 24$ environments give significant weight to a grid import price forecast in the far future: six and 18 steps ahead, respectively. This suggests that the models ingest future price information and utilize it to make battery deployment decisions.

5.5 Performance Under Uncertainty

As forecasting and modeling of real-world microgrids is imperfect, it is crucial to evaluate algorithm performance in the context of inexact forecasts and models that better mimic real-world scenarios. We showed that RL can outperform MPC with a forecast horizon of seven steps even if both are equipped with a perfect forecast in Section 5.3; in this section, we should that RL outperforms MPC in the context of inaccurate battery models but not imperfect forecasts.

MPC's continued superiority over RL in the latter setting is due to the fact that MPC does not suffer significantly from imperfect forecasting in our simulated conditions (Fig. 2). Despite this, RL remains able to outperform RBC consistently with imperfect forecasts. Unlike our previous experiments, we observe that overtraining becomes a factor with larger forecast uncertainties in RL after roughly 60 epochs, suggesting that RL begins to infer from forecast noise.

MPC's performance does not extend to inaccurate battery models, and in this context RL is the best-performing algorithm with both biased and decay battery models. In the case of biased models, RL outperforms MPC when the known efficiency is three or more percentage points higher (0.93 vs 0.9) or 23 percentage points lower than the true effi-

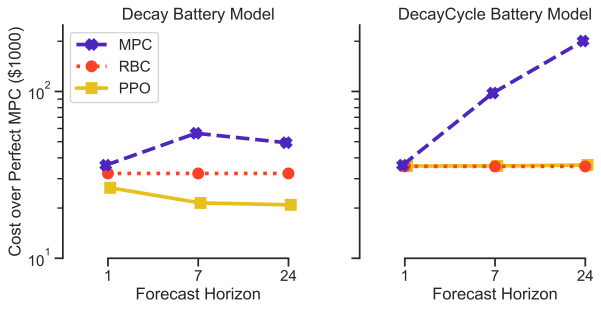

Figure 4: Performance with batteries whose efficiency decays over time. *Decay Battery* and *DecayCycle Battery* refer to the battery models as described in Section 3.3 and use decay rates of $4.2 \cdot 10^{-5}$ and $4.8 \cdot 10^{-4}$, respectively.

ciency of 0.9 (Fig. 3). While MPC performance suffers if the known efficiency deviates in both directions from the true efficiency, this effect is not symmetric and known efficiencies lower than the true efficiency do not lessen performance to the same extent: overestimating efficiency is much more harmful than underestimating it in MPC.

RL continues to outperform MPC with batteries that decay temporally (Fig. 4). MPC struggles in this case, with longer forecast horizons leading to worse MPC performance with the cycle decay battery; this is consistent across decay rates. RL outperforms MPC in both battery decay scenarios and outperforms RBC in the decay battery scenario. With the decay cycle battery, RL is able to replicate RBC performance but does not exceed it.

5.6 Generalization

Our approach generalizes to many other microgrid scenarios, as can be seen in Table 3, and significantly outperforms RBC on microgrids two, five, and seven. On microgrids four and six, RL and RBC performance is effectively equivalent, while on the remaining micorgrids RL struggles more. Additional specialization on these microgrids could improve performance, with different hyperparameter combinations potentially allowing for improved performance.

Cross-referencing Table 1, it is apparent that the scenarios where RL performs the most poorly are those with both macrogrids and gensets. With that in mind, we further investigate the efficacy of our algorithms in the presence of both macrogrid and genset in scenario ten. Specifically, we consider the case where we allow the algorithm to control both the output and the status (on or off) of the genset along with the case where we allow the genset's output to be controlled while its status is always on as well as the case where the genset is off. We find that forcing the genset on is ineffective and it is clear than optimal policies here do not use the genset perpetually, if at all (Supplementary Fig. S6). In addition, random network distillation is ineffective in this scenario, with PPO in the presence of RND underperforming PPO without RND.

Scenario	1	2	3	4	5
RL	1.99	**1.05**	**1.03**	1.02	**1.03**
RBC	**1.00**	16.60	16.65	**1.01**	16.65
Scenario	6	7	8	9	10
RL	1.01	**1.07**	3.38	1.36	3.86
RBC	**1.00**	16.60	**1.12**	**1.02**	**1.15**

Table 3: Performance on the evaluation set on nine additional microgrid scenarios as measured by cost relative to MPC with perfect 24-step forecast. MPC costs with perfect forecast in each scenario are scaled to 1.00.

6 Discussion

We present an evaluation of rule-based control (RBC), model predictive control (MPC), and reinforcement learning (RL) on a tertiary control microgrid simulator. We devise a microgrid environment that abstracts away high-level load balancing requirements and allows algorithms to optimize for efficiency and show that RL agents are effective and able to outperform RBC in a variety of scenarios. We utilize random network distillation and domain randomization to promote exploration and generalization and find that both aid performance. We train RL agents that achieve costs of $\$355,344$ and $\$355,803$ with forecast horizons of seven and 24, respectively, 1.1% and 1.2% worse than the perfect MPC model performance of $\$351,555$ with a forecast horizon of 24, and 1.7% and 1.5% better than MPC with a forecast horizon of seven, which obtains a cost of $\$361,362$.

We show that MPC persists as the most powerful algorithm when models are accurate and forecast uncertainty is low and that MPC is relatively robust to forecast uncertainty. As model uncertainty increases, we show that RL prevails as the most effective algorithm. This is true with both biased battery models that have an inaccurate but constant efficiency, and battery models that have a true underlying model with temporally decaying efficiences. In scenarios with no forecasting information, we show that RL converges to the RBC policy.

This work serves as a building block for building effective control algorithms for electrical microgrids in the context of uncertainty. These approaches are crucial in addressing real-world energy management challenges, particularly in developing methods to improve climate resilience.

Acknowledgements

The authors would like to acknowledge Mauricio Araya and Denis Akhiyarov (TotalEnergies EP R&T US) for their assistance in data considerations and model development. The authors thank TotalEnergies for partially support this effort.

References

Bordons, C.; Garcia-Torres, F.; and Ridao, M. A. 2020. Model Predictive Control Fundamentals. In Bordons, C.; Garcia-Torres, F.; and Ridao, M. A., eds., *Model Predictive Control of Microgrids*, Advances in Industrial Control, 25–44.

Cham: Springer International Publishing. ISBN 978-3-030-24570-2.

Borrelli, F.; Bemporad, A.; and Morari, M. 2017. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press.

Burda, Y.; Edwards, H.; Storkey, A.; and Klimov, O. 2018. Exploration by Random Network Distillation. ArXiv:1810.12894 [cs, stat].

Chebotar, Y.; Handa, A.; Makoviychuk, V.; Macklin, M.; Issac, J.; Ratliff, N.; and Fox, D. 2019. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. *arXiv:1810.05687 [cs]*. ArXiv: 1810.05687.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying Generalization in Reinforcement Learning. In *International Conference on Machine Learning*, 1282–1289. PMLR. ISSN: 2640-3498.

Dai, X.; Liu, J.; and Zhang, X. 2020. A review of studies applying machine learning models to predict occupancy and window-opening behaviours in smart buildings. *Energy and Buildings*, 223: 110159.

Fatin Ishraque, M.; Shezan, S. A.; Ali, M. M.; and Rashid, M. M. 2021. Optimization of load dispatch strategies for an islanded microgrid connected with renewable energy sources. *Applied Energy*, 292: 116879.

François-Lavet, V.; Taralla, D.; Ernst, D.; and Fonteneau, R. 2016. Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*.

Grosan, C.; and Abraham, A. 2011. *Intelligent Systems: A Modern Approach*. Springer Science & Business Media. ISBN 978-3-642-21004-4. Google-Books-ID: c1fzgQj5lhkC.

Guo, C.; Wang, X.; Zheng, Y.; and Zhang, F. 2022. Real-time optimal energy management of microgrid with uncertainties based on deep reinforcement learning. *Energy*, 238: 121873.

Henri, G.; Levent, T.; Halev, A.; ALAMI, R.; and Cordier, P. 2020. pymgrid: An Open-Source Python Microgrid Simulator for Applied Artificial Intelligence Research. In *NeurIPS 2020 Workshop on Tackling Climate Change with Machine Learning*.

Lamburn, D. J.; Gibbens, P. W.; and Dumble, S. J. 2014. Efficient constrained model predictive control. *European Journal of Control*, 20(6): 301–311.

Leondes, C. T. 2001. *Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century*. Elsevier. ISBN 978-0-08-053145-8. Google-Books-ID: 5kSamKhS560C.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2019. Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*. ArXiv: 1509.02971.

Liu, D.; Zang, C.; Zeng, P.; Li, W.; Wang, X.; Liu, Y.; and Xu, S. 2023. Deep reinforcement learning for real-time economic energy management of microgrid system considering uncertainties. *Frontiers in Energy Research*, 11.

Lucia, S.; Andersson, J. A. E.; Brandt, H.; Diehl, M.; and Engell, S. 2014. Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, 24(8): 1247–1259.

Mahmood, A. R.; Korenkevych, D.; Vasan, G.; Ma, W.; and Bergstra, J. 2018. Benchmarking Reinforcement Learning Algorithms on Real-World Robots. ArXiv:1809.07731 [cs, stat].

Mirakhorli, A.; and Dong, B. 2016. Occupancy behavior based model predictive control for building indoor climate—A critical review. *Energy and Buildings*, 129: 499–513.

Naeem, M.; Rizvi, S. T. H.; and Coronato, A. 2020. A Gentle Introduction to Reinforcement Learning and its Application in Different Fields. *IEEE Access*, 8: 209320–209344. Conference Name: IEEE Access.

Nakabi, T. A.; and Toivanen, P. 2021. Deep reinforcement learning for energy management in a microgrid with flexible demand. *Sustainable Energy, Grids and Networks*, 25: 100413.

Olivares, D. E.; Mehrizi-Sani, A.; Etemadi, A. H.; Cañizares, C. A.; Iravani, R.; Kazerani, M.; Hajimiragha, A. H.; Gomis-Bellmunt, O.; Saeedifard, M.; Palma-Behnke, R.; Jiménez-Estévez, G. A.; and Hatziargyriou, N. D. 2014. Trends in Microgrid Control. *IEEE Transactions on Smart Grid*, 5(4): 1905–1919. Conference Name: IEEE Transactions on Smart Grid.

Qin, S. J.; and Badgwell, T. A. 2003. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7): 733–764.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*. ArXiv: 1707.06347.

Schwenzer, M.; Ay, M.; Bergs, T.; and Abel, D. 2021. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5): 1327–1349.

Shezan, S. A.; Hasan, K. N.; Rahman, A.; Datta, M.; and Datta, U. 2021. Selection of Appropriate Dispatch Strategies for Effective Planning and Operation of a Microgrid. *Energies*, 14(21): 7217. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. Cambridge, Mass: MIT Press. ISBN 978-0-262-19398-6.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *arXiv:1703.06907 [cs]*. ArXiv: 1703.06907.

Zeng, P.; Li, H.; He, H.; and Li, S. 2019. Dynamic Energy Management of a Microgrid Using Approximate Dynamic Programming and Deep Recurrent Neural Network Learning. *IEEE Transactions on Smart Grid*, 10(4): 4435–4445.

Zhang, Y.; Meng, F.; Wang, R.; Zhu, W.; and Zeng, X.-J. 2018. A stochastic MPC based approach to integrated energy management in microgrids. *Sustainable Cities and Society*, 41: 349–362.