EPJ Web of Conferences **380**, 02005 (2026) https://doi.org/10.1051/epjconf/202638002005
*ICISCT'25*

# Multi-Agent Deep Reinforcement Learning for Sustainable Manufacturing with Integrated Microgrid: Benchmarking SAC and PPO

*Mohamed* Ballouch[1,*], *Omar* Souissi[1,**], and *Mohammed Raiss* El-Fenni[1,***]

[1]National Institute of Posts and Telecommunications (INPT), Rabat, Morocco

**Abstract.** Running a manufacturing line alongside an on-site renewable microgrid couples two decisions that used to be handled separately: when to run the machines, and how to manage local generation and storage. Both must be settled in real time, and each controller sees only a noisy local part of the plant. We cast the problem as a decentralized partially observable Markov decision process (Dec-POMDP) with five cooperating agents, namely two production machines, a wind turbine, a battery bank, and a backup generator. On this testbed we compare two actor-critic algorithms that treat exploration in opposite ways: Soft Actor-Critic (SAC), which is entropy-regularized, and Proximal Policy Optimization (PPO), which keeps each update inside a clipped trust region. Over 10,000 training episodes SAC earns a 15.6% higher mean episode reward (5990.65 against 5181.75), which we attribute to wider exploration of the joint state space, while PPO produces a 22% tighter reward spread (standard deviation 24.86 against 31.89). The learned storage policies differ as well: PPO cycles the battery across almost the full admissible 10%–90% state-of-charge (SOC) window, whereas SAC holds a narrow 25%–60% band. We discuss what these differences imply for battery wear and scheduling stability, check the results against published baselines and through cross-algorithm agreement, and turn the findings into practical algorithm-selection guidance for industrial operators.

## 1 Introduction

Cutting the carbon footprint of an industrial site increasingly means powering the production line from renewable generation on the same premises. An industrial microgrid gathers renewable sources, storage, and flexible loads behind one point of common coupling, and it can reduce both the carbon intensity and the energy cost of a plant. The difficulty is that a single controller must now solve together what used to be two separate problems: keeping the machines supplied and on schedule, and holding the local grid in balance as wind output, battery charge, and demand shift through the day [1]. Production targets, machine wear, variable generation, and battery aging are coupled, so an action taken for one of them reaches the others within minutes.

*e-mail: ballouch.mo@gmail.com
**e-mail: souissi@inpt.ac.ma
***e-mail: raiss@inpt.ac.ma

Deep reinforcement learning (DRL) suits control problems of this type, where the best action follows from a long and uncertain horizon rather than from a closed-form solution [2]. In multi-agent form, DRL has been applied to microgrid energy management [3, 4], to joint production-and-energy scheduling [5, 6], and to demand-side flexibility [7]. Much of this work shares one simplification: it assumes the full state is observable. Treating the plant as a standard Markov decision process (MDP) is convenient, but it sets aside the sensor noise, reporting delays, and missing telemetry that a real installation has to handle [8].

Our earlier conference paper [9] proposed a multi-agent POMDP model for coupled manufacturing-microgrid control and gave headline performance figures for SAC and PPO. The present study extends that work well past the aggregate numbers along three lines:

1. **Mechanistic algorithm analysis.** We explain *why* entropy regularization (SAC) and clipped policy-gradient updates (PPO) steer the learned policies in different directions here, instead of only reporting that they do (Section 4).

2. **Behavioral characterization of storage.** We study the storage policies each algorithm arrives at, including how deeply it cycles the battery and what that means for degradation, and we treat this emergent behavior as a result in its own right (Section 5).

3. **Broader contextualization and validation.** We set the comparison against a wider 2017–2025 body of work on multi-agent DRL for energy systems (Section 2) and add a validation step that combines convergence evidence, cross-algorithm agreement, and benchmarking against published results (Section 5.3).

The paper is organized as follows. Section 2 reviews related work and states the gap we address. Section 3 presents the system model and the POMDP formulation. Section 4 describes the two algorithms and contrasts their inductive biases. Section 5 reports the experiments, the validation, and the comparison. Section 6 ends with deployment guidance and open questions.

## 2 Literature Review

### 2.1 Foundations of Multi-Agent Deep Reinforcement Learning

Single-agent DRL matured around value-based methods such as deep Q-networks [10] and the policy-gradient and actor-critic families. Extending these methods to several interacting agents introduces a difficulty that single-agent theory does not face: from the viewpoint of any one agent the environment is non-stationary, because the other agents are learning at the same time. Lowe et al. [11] addressed this with MADDPG, pairing a centralized critic that sees the joint state with actors that act on local observations only, a design now known as centralized training with decentralized execution (CTDE). Survey treatments by Oroojlooy and Hajinezhad [8] and by Canese et al. [2] organize these methods and make clear that the choice between value factorization and policy-gradient learning is one of the field's central design axes, which is exactly the axis our SAC/PPO comparison probes.

### 2.2 DRL for Manufacturing Systems

Within manufacturing, reinforcement learning has been applied to scheduling, predictive maintenance, and resource allocation. Lu et al. [6] applied MADDPG to demand response in discrete manufacturing and cut energy cost by 9.8% without sacrificing throughput. Yang et al. [5] coupled temporal-difference learning with deterministic policy gradients for joint

production-energy control, but their MDP framing left observation noise out of the picture. Closer to the present setting, Waseem et al. [1] tackled integrated microgrid-manufacturing control with explicit battery-degradation awareness, using a Dec-POMDP solved with MADDPG and validated on hardware; their results underline how strongly storage health couples to long-horizon scheduling decisions.

### 2.3 DRL for Microgrid Energy Management

On the energy side, Zhu et al. [3] added attention mechanisms to a multi-agent controller for a multi-energy industrial park and showed that learned inter-agent communication improves coordination. Guo and Gong [4] embedded prioritized experience replay in MADDPG for networked multi-microgrid systems and reported faster convergence under partial observability. Upadhyay et al. [7] combined PPO with load forecasting for industrial microgrid scheduling and achieved roughly 20% cost savings. These energy-management studies improve coordination and cost, but they adopt an MDP framing that sets observation noise aside, which is the gap the present partially observable formulation targets.

### 2.4 SAC versus PPO: A Comparative Gap

SAC [12] and PPO [13] are both widely deployed, yet head-to-head comparisons in industrial control remain uncommon, and most are reported only as aggregate scores. SAC's entropy bonus keeps the policy stochastic and sustains exploration, which tends to lift policy quality in complex environments at the price of higher variance. PPO's clipped objective holds each update close to the previous policy, favoring stability but risking premature, suboptimal convergence. Our earlier paper [9] reported first evidence of this trade-off; here we move from *that* it happens to *why*, and we connect the difference to the storage behavior each algorithm produces.

### 2.5 Gap and Contribution

Table 1 positions this study against the closest prior work. To the best of our knowledge, no earlier study compares an entropy-regularized algorithm against a clipped policy-gradient algorithm inside a POMDP-based manufacturing-microgrid testbed while also analyzing the emergent storage policies and their consequences for battery life.

**Table 1.** Positioning relative to existing approaches

| Reference | Multi-Agent | POMDP | Algorithm Comparison | ESS Behavior |
|---|---|---|---|---|
| Yang et al. [5] | No | No | No | No |
| Lu et al. [6] | Yes | No | No | No |
| Upadhyay et al. [7] | Partial | No | No | No |
| Zhu et al. [3] | Yes | No | No | No |
| Guo & Gong [4] | Yes | Partial | No | No |
| Waseem et al. [1] | Yes | Yes | No | Yes |
| Ballouch et al. [9] | Yes | Yes | Partial | Partial |
| **This work** | **Yes** | **Yes** | **Yes** | **Yes** |

# 3 System Model and POMDP Formulation

## 3.1 System Architecture

The testbed (Fig. 1) represents a manufacturing facility supplied by a hybrid microgrid. Five agents interact: two production machines ($m_1$, $m_2$), a wind turbine ($w$), a lithium-ion battery bank ($b$), and a diesel backup generator ($g$). Each machine runs in one of three modes (ON, IDLE, OFF), and the microgrid meets the resulting demand from wind generation, battery discharge, or the backup generator, in that order of preference.

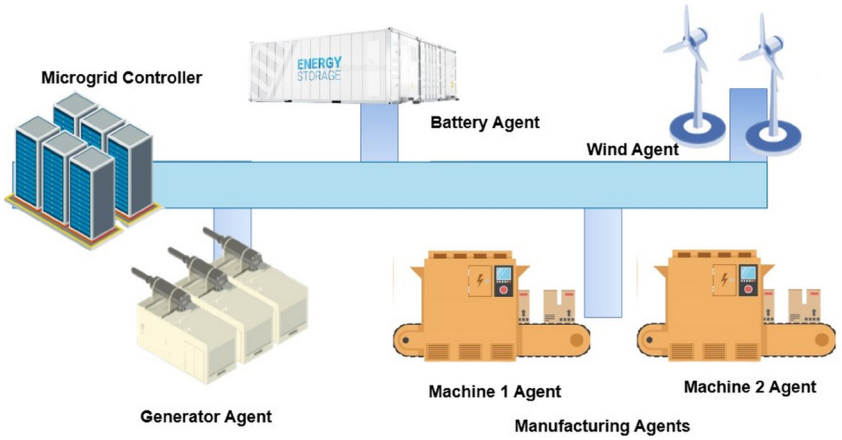


**Figure 1.** Multi-agent manufacturing-microgrid system architecture. Each component is an autonomous agent acting on its own local observation.

## 3.2 Decentralized POMDP Formulation

We model the plant as a decentralized POMDP:

$$\mathcal{M} = \langle \mathcal{I}, \{S_i\}, \{A_i\}, \mathcal{T}, \{R_i\}, \{\Omega_i\}, \{O_i\}, \gamma \rangle \tag{1}$$

where $\mathcal{I} = \{m_1, m_2, w, b, g\}$ is the set of agents, $S_i$ and $A_i$ are the local state and action spaces, $\mathcal{T}$ is the joint transition function, $R_i$ are the individual reward functions, $\Omega_i$ are the observation spaces, $O_i$ are the observation functions, and $\gamma \in (0, 1)$ is the discount factor. The Dec-POMDP framing is what separates this model from the MDP-based formulations that dominate the manufacturing-energy literature: it states explicitly that no agent has access to the global state at decision time.

### *3.2.1 Observation Model*

Each agent $i$ receives a noisy view of the global state:

$$o_{i,t} = [s_t^m,\ s_t^o,\ b_t,\ e_t] + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma^2 I) \tag{2}$$

where $s_t^m$ encodes the agent's own machine state, $s_t^o$ carries shared information about the other agents, $b_t$ are buffer levels, $e_t$ are energy variables, and the additive Gaussian term with $\sigma = 0.1$ stands in for sensor noise.

#### 3.2.2 Manufacturing Agent Dynamics

Each machine $m_i$ has discrete states {ON, IDLE, OFF} with stochastic transitions governed by a failure probability $p_f$ derived from the mean time between failures (MTBF):

$$P(s_{t+1}|s_t, a_t) = \begin{cases} 1 - p_f & \text{if } s_{t+1} = s_t,\ s_t = \text{ON} \\ p_f & \text{if } s_{t+1} = \text{OFF},\ s_t = \text{ON} \\ 1 & \text{if } s_{t+1} = a_t,\ s_t \neq \text{ON} \end{cases} \tag{3}$$

#### 3.2.3 Wind Turbine Model

Wind power generation follows the standard aerodynamic relation:

$$P_{\text{gen}} = \tfrac{1}{2}\rho A v^3 C_p \eta_{\text{gear}} \eta_{\text{elec}} \tag{4}$$

where $\rho$ is air density, $A$ the swept area, $v$ the wind speed, $C_p$ the power coefficient, and $\eta_{\text{gear}}$, $\eta_{\text{elec}}$ the mechanical and electrical efficiencies. The wind agent observes the current generation $P_{\text{gen}}$, a 24-hour forecast $P_{\text{fore}}$, and the battery SOC.

#### 3.2.4 Battery Storage Dynamics

The battery SOC evolves according to:

$$\text{SOC}_{t+1} = \text{SOC}_t + \frac{\eta_c P_c - P_d/\eta_d}{E_{\text{cap}}} \Delta t \tag{5}$$

subject to $\text{SOC}_{\min} \leq \text{SOC}_t \leq \text{SOC}_{\max}$ and the power bounds $0 \leq P_c \leq P_{c,\max}$, $0 \leq P_d \leq P_{d,\max}$, where $\eta_c$, $\eta_d$ are the charge and discharge efficiencies and $E_{\text{cap}}$ is the nominal capacity.

#### 3.2.5 Generator Model

The backup generator incurs the operating cost:

$$C_g = c_{\text{fuel}} P_g + c_{\text{start}} n_{\text{start}} + c_{\text{maint}} \tag{6}$$

subject to the capacity constraint $P_{\min} \leq P_g \leq P_{\max}$ and the ramp-rate limit $|P_g(t) - P_g(t-1)| \leq R_{\max} \Delta t$.

### 3.3 Reward Structure

Each agent maximizes a weighted reward that blends production, energy, and coordination terms:

$$R_i = w_p R_{i,\text{prod}} + w_e R_{i,\text{energy}} + w_c R_{\text{coord}} \tag{7}$$

with $w_p = 0.5$, $w_e = 0.3$, $w_c = 0.2$. The system-level objective is the discounted sum of individual rewards:

$$\max_{\{\pi_i\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{i \in \mathcal{I}} R_i(s_t, a_t)\right] \tag{8}$$

subject to $s_{t+1} \sim \mathcal{T}(s_t, a_t)$, $a_{i,t} \sim \pi_i(o_{i,t})$, and $o_{i,t} \sim O_i(s_t)$. We note that battery degradation does not appear in Eq. (7); this omission becomes important when we interpret the storage policies in Section 5.5.

## 4 Algorithmic Paradigms

We compare two actor-critic algorithms that resolve the exploration-exploitation dilemma in opposite ways. Both run inside a CTDE framework [11] implemented with RLlib [14], and both are tuned under the same budget and procedure. Keeping the network architecture, the training budget, and the environment fixed across the two algorithms is deliberate: it lets us attribute any behavioral difference to the learning objective rather than to implementation details.

### 4.1 Soft Actor-Critic (Entropy-Regularized)

SAC [12] adds an entropy term to the usual return, rewarding the policy for staying stochastic:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\big[r(s_t,a_t) + \alpha\,\mathcal{H}(\pi(\cdot|s_t))\big] \tag{9}$$

where $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}[\log\pi(a|s_t)]$ and $\alpha > 0$ is the temperature. The temperature is tuned automatically to track a target entropy $\bar{\mathcal{H}}$:

$$\alpha_{t+1} = \alpha_t - \eta_\alpha\,\nabla_\alpha\,J(\alpha) \tag{10}$$

and twin critics guard against value overestimation:

$$Q_{\text{target}} = r + \gamma \min_{j=1,2} Q_{\phi'_j}(s',a') - \alpha\log\pi(a'|s') \tag{11}$$

**Why this matters here.** The entropy term stops the policy from settling on a single deterministic rule too early. That helps in this plant, where the best action depends on the operating regime: a windy hour needs a different storage and scheduling response than a calm one. The price is that the agent keeps trying off-peak actions even after it has effectively learned the task, which later shows up as wider reward variance.

### 4.2 Proximal Policy Optimization (Clipped Surrogate)

PPO [13] optimizes a clipped surrogate that keeps each new policy close to the old one:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)\,\hat{A}_t\Big)\Big] \tag{12}$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the importance ratio and $\epsilon = 0.2$ is the clip range. Advantages come from generalized advantage estimation (GAE):

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}, \qquad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{13}$$

with $\lambda = 0.95$ setting the bias-variance balance.

**Why this matters here.** Clipping prevents the large, abrupt policy jumps that, in a manufacturing setting, could turn into buffer overflows or sudden energy shortfalls. The trade-off is a slower climb and a tendency to lock onto the first good-enough strategy it finds, which is just what we see in the storage policy.

### 4.3 Implementation Details

The two algorithms share an identical network: two hidden layers of 256 units with ReLU activations. The common hyperparameters are learning rates $\alpha_\pi = \alpha_Q = 3 \times 10^{-4}$, batch size 256, replay buffer size $10^6$ (SAC only), discount $\gamma = 0.99$, and an initial SAC temperature $\alpha_0 = 0.2$ (auto-tuned thereafter). Under CTDE, the critic uses global state information during training, while at execution time each agent acts on its local observation alone, so the trained policies remain deployable on plants with limited inter-agent bandwidth.

## 5 Experimental Results and Discussion

### 5.1 Experimental Configuration

The industrial parameters are taken from Hu et al. [15] and listed in Tables 2 and 3. Each experiment runs 10,000 episodes of 24-hour operation, sampled at 1,440 one-minute steps per episode, on an Intel i7-10750H with an NVIDIA GTX 1650.

**Table 2.** Manufacturing system configuration

| Parameter | Machine 1 | Machine 2 | Units |
|---|---|---|---|
| MTBF | 95 | 105 | minutes |
| MTTR | 10 | 15 | minutes |
| Operational Power | 1,000 | 1,000 | kW |
| Idle Power | 600 | 600 | kW |
| Production Rate | 1 | 1 | units/cycle |

**Table 3.** Microgrid energy system specifications

| Component | Parameter | Value |
|---|---|---|
| Wind Turbine | Rated Capacity | 400 kW |
| | Combined Efficiency | 0.48 |
| | O&M Cost | \$0.03/kWh |
| Battery Storage | Capacity | 600 kWh |
| | Round-trip Efficiency | 0.81 |
| | O&M Cost | \$0.1/kWh |
| Backup Generator | Rated Capacity | 400 kW |
| | Fuel Efficiency | 0.35 |
| | O&M Cost | \$0.2/kWh |

### 5.2 Convergence and Reward Analysis

Table 4 and Figs. 2–3 report the training outcomes. SAC settles at a mean episode reward of 5990.65 after roughly 4,200 episodes (about 13 h of wall-clock time), and PPO reaches 5181.75 after about 3,500 episodes (about 12 h). The 15.6% gap in favor of SAC reflects the entropy bonus steering the policy into high-reward regions that PPO's cautious updates never reach. PPO, in turn, posts a 22% smaller standard deviation (24.86 against 31.89), which is the stabilizing effect of the clipped objective made visible.

The extra training cost of SAC, about 8% in wall-clock time, comes from the twin Q-network updates and the temperature tuning. It buys a clearly better final policy, which suggests that the entropy bonus helps the five agents find coordination patterns the more conservative algorithm leaves untouched.

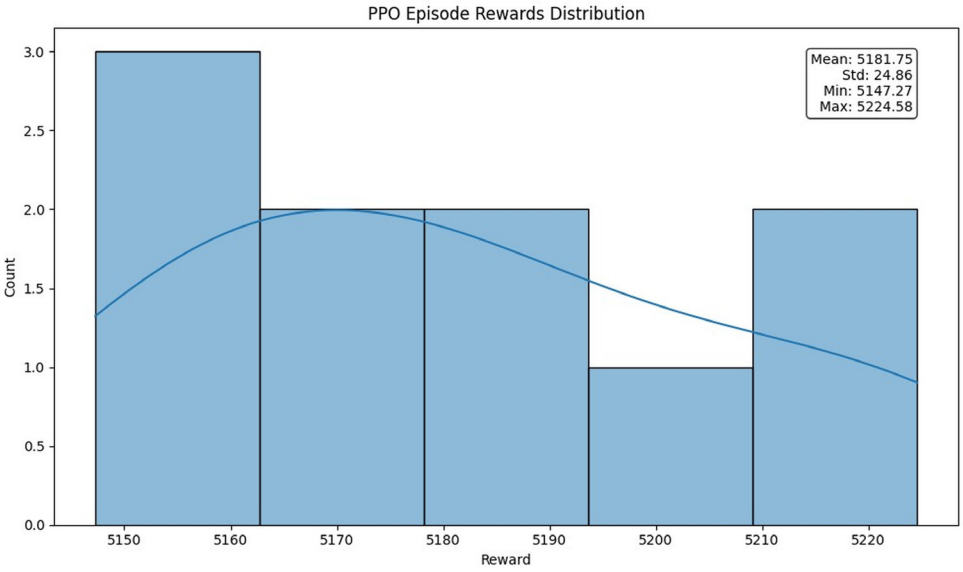


**Figure 2.** PPO episode-reward distribution over 20 evaluation rollouts. The tight cluster around the mean (5181.75) is the signature of conservative policy updates.

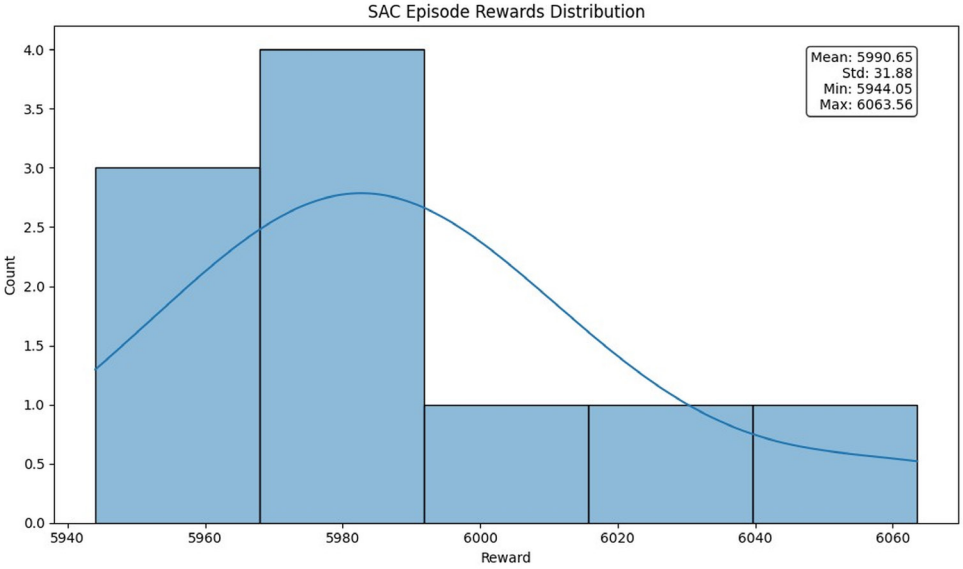


**Figure 3.** SAC episode-reward distribution over 20 evaluation rollouts. The wider spread reflects entropy-driven stochasticity, which trades some stability for exploration.

**Table 4.** Summary of training performance metrics

| Metric | PPO | SAC | Relative Difference |
|---|---|---|---|
| Mean Reward | 5181.75 | 5990.65 | +15.6% (SAC) |
| Std. Deviation | 24.86 | 31.89 | −22% (PPO) |
| Training Time | 12 h | 13 h | +8.3% (SAC) |
| Convergence Episode | ∼3,500 | ∼4,200 | +20% (SAC) |

### 5.3 Validation

Because the study rests on simulation rather than on a physical plant, we validate the results in three complementary ways instead of relying on a single metric.

**Convergence.** Both policies reach a stable plateau well before the episode budget is exhausted (around episode 3,500 for PPO and 4,200 for SAC, against a 10,000-episode horizon), and the evaluation rollouts in Figs. 2–3 stay concentrated around their respective means. The remaining episodes therefore act as a stability check rather than as continued learning.

**Cross-algorithm consistency.** SAC and PPO come from different algorithm families, off-policy entropy-regularized learning and on-policy clipped policy gradients, yet they settle on nearly the same core manufacturing behavior, with Machine 1 production rates of 0.864 and 0.859 (Table 5). When two independent learners agree on the production optimum, that points to a property of the environment rather than an artifact of one algorithm.

**Agreement with the literature.** The reward and cost improvements we observe sit inside the range reported for comparable multi-agent DRL energy controllers: Lu et al. [6] report 9.8% energy-cost reduction with MADDPG, and Upadhyay et al. [7] report roughly 20% with a PPO-based scheduler. Our gains are of the same order, which supports the plausibility of the simulated dynamics. A full parametric sweep of the observation-noise level $\sigma$ and the reward weights, together with hardware-in-the-loop testing, is left to future work (Section 6).

### 5.4 Manufacturing Performance Comparison

Table 5 breaks down production per machine. Both algorithms keep Machine 1 highly utilized (0.864 for PPO, 0.859 for SAC), which confirms that moving to a partially observable formulation does not compromise the core production target. The informative difference is on Machine 2: PPO reaches 0.772 against SAC's 0.742, a 4% edge that we attribute to PPO's lower-entropy, more deterministic coordination between the two machines.

**Table 5.** Machine production rate statistics

| Algorithm | Machine | Mean | Std. Dev. | Range |
|---|---|---|---|---|
| PPO | M1 | 0.864 | 0.462 | [0.0, 1.32] |
| | M2 | 0.772 | 0.530 | [0.0, 1.32] |
| SAC | M1 | 0.859 | 0.475 | [0.0, 1.32] |
| | M2 | 0.742 | 0.535 | [0.0, 1.32] |

### 5.5 Emergent Energy Storage Strategies

The clearest behavioral split appears in how the two policies use the battery. Figs. 4 and 5 show representative SOC trajectories.

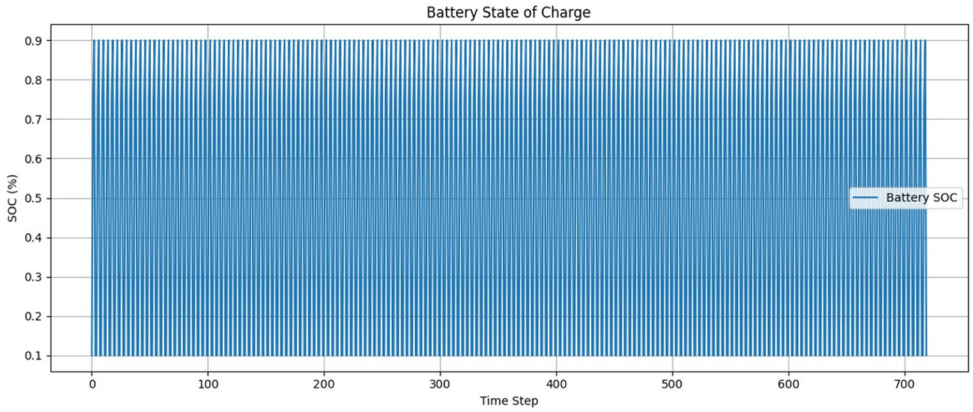


**Figure 4.** Battery SOC trajectory under the PPO policy. The agent uses the full admissible 10%–90% range with rapid charge-discharge cycles.

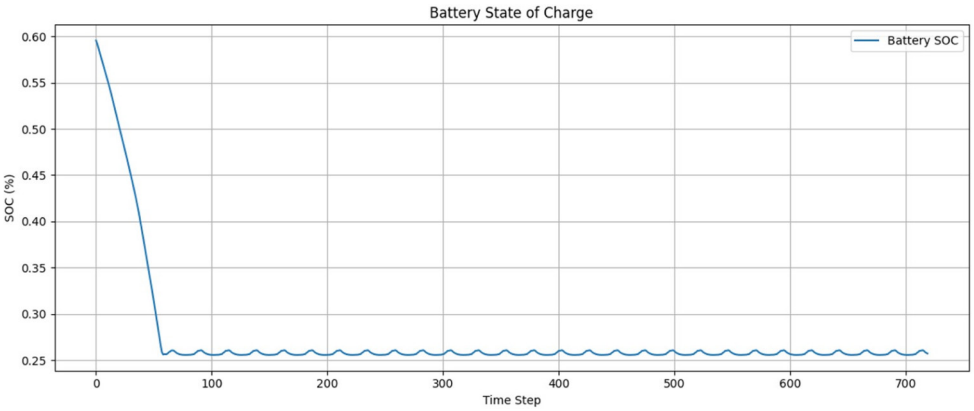


**Figure 5.** Battery SOC trajectory under the SAC policy. The agent settles into a conservative 25%–60% band, reducing cycling depth.

**PPO policy.** The PPO battery agent moves across almost the entire 10%–90% window (Fig. 4). Cycling this deeply keeps the largest possible energy buffer available, so the plant can absorb large wind ramps and ride out calm periods without cutting production. The cost is wear: deep and frequent cycling is what ages lithium-ion cells fastest, and on a 600 kWh system it can shorten service life by 30–40% compared with shallow cycling [1].

**SAC policy.** The SAC agent keeps the SOC within 25%–60% (Fig. 5), after a short initial discharge that then settles into a steady oscillation. Staying in a narrow band holds mechanical and thermal stress down and tends to extend battery life. We read the entropy bonus as the reason the policy avoids an extreme cycling rule: it pushes SAC toward a middle ground that keeps energy available while sparing the cells.

**Interpretation.** The two policies follow different optimization instincts. PPO favors immediate utility, that is, maximum energy throughput, whereas SAC, through wider ex-

ploration, settles on a strategy that in effect accounts for the long term, even though battery degradation never appears in the reward of Eq. (7). A degradation-aware behavior that emerges without any degradation term is the clearest result of the study, and it suggests that entropy regularization can act as an implicit push toward more sustainable use of the asset.

### 5.6 Practical Deployment Considerations

Three engineering factors mattered during the experiments:

1. **Replay-buffer sizing.** SAC's $10^6$-transition buffer has to be provisioned carefully. Diverse experience stabilizes off-policy learning, but a buffer that is too large dilutes recent transitions and slows adaptation.

2. **Distributed training.** RLlib's parallel rollout workers [14] cut wall-clock training time by roughly 3× compared with single-worker runs.

3. **CTDE at deployment.** Centralized training with decentralized execution gave us effective coordination during learning while keeping the deployed agents independent of high-bandwidth inter-agent links.

### 5.7 Algorithm Selection Guidelines

The findings translate into a short decision rule for practitioners:

- **Just-in-time manufacturing** with tight delivery windows favors PPO, whose deterministic policy yields predictable throughput at low variance.
- **Batch or flexible manufacturing** where energy cost dominates and some production variability is acceptable benefits from SAC's higher reward and gentler battery usage.
- **Hybrid schemes** could run PPO during nominal operation and lean on SAC's exploration during commissioning or after a regime change. Combining the two in a principled way is an open research question.

### 5.8 Limitations

Several simplifications temper any direct transfer to a real installation. The failure model uses a constant MTBF rather than condition-dependent degradation. Wind profiles are synthetic, not site-measured. Inter-agent communication is treated as instantaneous and lossless. Finally, battery degradation is absent from the reward, yet SAC still finds a degradation-aware policy, which suggests that adding an explicit degradation term could push results further. Closing these gaps through hardware-in-the-loop validation and pilot deployment is our main next step.

## 6 Conclusion

We benchmarked an entropy-regularized algorithm (SAC) against a clipped policy-gradient algorithm (PPO) on a partially observable manufacturing-microgrid control problem, modeled as a five-agent decentralized POMDP with observation noise. Over 10,000 training episodes the comparison yielded three results:

- SAC earns 15.6% higher mean reward through entropy-driven exploration, at the cost of 22% higher reward variance and about 8% more training time.

- PPO delivers steadier, more predictable throughput, including a 4% higher Machine 2 utilization that follows from its more deterministic coordination.
- The two algorithms learn qualitatively different storage policies, deep full-range cycling for PPO (10%–90% SOC) against shallow cycling for SAC (25%–60% SOC), with direct consequences for battery longevity.

Taken together, these results give operators a concrete basis for picking an algorithm by what they value most: energy efficiency, production stability, or asset life. The validation step, namely convergence, cross-algorithm agreement, and consistency with published baselines, supports the findings within the limits of a simulation study. Future work will pursue hybrid policies, an explicit battery-degradation term in the reward, additional renewable sources, a parametric study of noise and reward weights, and real-world pilot validation.

## References

[1] M. Waseem, M.S. Maithripala, Q. Chang, Z. Lin, Integrated Energy Optimization in Manufacturing Through Multiagent Deep Reinforcement Learning: Holistic Control of Manufacturing, Microgrid Systems, and Battery Storage. J. Manuf. Sci. Eng. **147**(6), 061003 (2025). https://doi.org/10.1115/1.4067614

[2] L. Canese, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, S. Spanò, Multi-Agent Reinforcement Learning: A Review of Challenges and Applications. Appl. Sci. **11**(11), 4948 (2021). https://doi.org/10.3390/app11114948

[3] D. Zhu, B. Yang, Y. Liu, Z. Wang, K. Ma, X. Guan, Energy Management Based on Multi-Agent Deep Reinforcement Learning for a Multi-Energy Industrial Park. arXiv:2202.03771 (2022). https://doi.org/10.48550/arXiv.2202.03771

[4] G. Guo, Y. Gong, Multi-Microgrid Energy Management Strategy Based on Multi-Agent Deep Reinforcement Learning with Prioritized Experience Replay. Appl. Sci. **13**(5), 2865 (2023). https://doi.org/10.3390/app13052865

[5] J. Yang, Z. Sun, W. Hu, L. Steinmeister, Joint Control of Manufacturing and Onsite Microgrid System via Novel Neural-Network Integrated Reinforcement Learning Algorithms. Appl. Energy **315**, 118982 (2022). https://doi.org/10.1016/j.apenergy.2022.118982

[6] R. Lu, Y.-C. Li, Y. Li, J. Jiang, Y. Ding, Multi-agent deep reinforcement learning based demand response for discrete manufacturing systems energy management. Appl. Energy **276**, 115473 (2020). https://doi.org/10.1016/j.apenergy.2020.115473

[7] S. Upadhyay, I. Ahmed, L. Mihet-Popa, Energy Management System for an Industrial Microgrid Using Optimization Algorithms-Based Reinforcement Learning Technique. Energies **17**, 3898 (2024). https://doi.org/10.3390/en17163898

[8] A. Oroojlooy, D. Hajinezhad, A review of cooperative multi-agent deep reinforcement learning. Appl. Intell. **53**(11), 13677–13722 (2023). https://doi.org/10.1007/s10489-022-04105-y

[9] M. Ballouch, O. Souissi, M.R. El-Fenni, Integrated Manufacturing-Microgrid Control Using Multi-Agent Deep Reinforcement Learning. IFAC-PapersOnLine **59**(10), 1372–1377 (2025). https://doi.org/10.1016/j.ifacol.2025.09.231

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, et al., Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Advances in Neural Information Processing Systems (NeurIPS) **30**, pp. 6379–6390 (2017). https://doi.org/10.48550/arXiv.1706.02275

[12] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, S. Levine, Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905 (2018). https://doi.org/10.48550/arXiv.1812.05905

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms. arXiv:1707.06347 (2017). https://doi.org/10.48550/arXiv.1707.06347

[14] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, I. Stoica, RLlib: Abstractions for Distributed Reinforcement Learning. arXiv:1712.09381 (2018). https://doi.org/10.48550/arXiv.1712.09381

[15] W. Hu, Z. Sun, Y. Zhang, Y. Li, Joint Manufacturing and Onsite Microgrid System Control Using Markov Decision Process and Neural Network Integrated Reinforcement Learning. Procedia Manuf. **39**, 1242–1249 (2019). https://doi.org/10.1016/j.promfg.2020.01.347