Optimal Energy Management of Renewable Energy Communities: a Multi-Agent Reinforcement Learning Approach

Samuele Delpero, Davide Salaorni, and Marcello Restelli

Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano
samuele.delpero@mail.polimi.it,
{davide.salaorni, marcello.restelli}@polimi.it

Abstract. The transition toward decentralized and sustainable energy systems has emphasized the role of Renewable Energy Communities (RECs) as a promising organizational model that promotes energy self-sufficiency, reduces dependence on the main grid, and improves both economic and environmental outcomes for their participants. A central challenge in RECs is the design of an efficient and fair Energy Management System (EMS) capable of coordinating the actions of multiple heterogeneous entities while optimizing the overall community-wise performance. In this work, we present the *INcentive Allocation for Interest Alignment* (INAIA) algorithm, a novel Multi-Agent Reinforcement Learning (MARL) approach for addressing the energy management problem in RECs, considering a decentralized EMS architecture aligned with the current Italian regulatory framework. Our approach integrates mechanism design principles into the MARL algorithm to align individual participant interests with global community objectives. We conduct an extensive experimental campaign across tasks of increasing complexity based on real-world data. In a representative scenario, our method achieves a 17.3% increase in self-consumption compared to the best-performing baseline. We corroborate such results with ablation studies, confirming the effectiveness and robustness of the proposed solution.

Keywords: Multi-Agent Reinforcement Learning · Renewable Energy Communities · Decentralized Energy Management Systems.

1 Introduction

The increasing global demand for energy, coupled with the urgent need to mitigate environmental impact, is driving a profound transformation of the energy landscape. Renewable Energy Sources (RESs) are central to this transformation. RESs, including solar, wind, hydroelectric, biomass, and geothermal, offer a sustainable and low-emission alternative to fossil fuels with a minimal environmental footprint [4]. However, their inherent intermittent nature presents challenges in maintaining a stable, reliable, and high-quality power supply [2]. To address these issues, Distributed Energy Resources (DERs), including small-scale generation and storage systems, are playing an increasingly important role in decentralizing energy production and improving grid flexibility.

Within this decentralized landscape, Renewable Energy Communities (RECs) represent a key organizational model for the collective production, consumption, and management of energy at the local level. RECs are composed of *prosumers*, i.e., entities that both produce and consume energy, who collaborate to share energy, balance supply and demand, and increase local self-sufficiency. In this context, we will use the term broadly to include also pure consumers. Aggregating and coordinating RESs and DERs, RECs can reduce reliance on the main grid, reduce energy costs, and contribute to environmental sustainability.

A critical component of RECs is the Energy Management System (EMS), which acts as the community's control layer. The EMS regulates how the energy is produced, stored, and exchanged both within the community and with the external grid. While traditional EMS architectures are often centralized [1, 16], relying on a single decision-making unit, the growing pervasive presence of RESs and DERs increasingly calls for decentralized control strategies. In such a scenario, the EMS is distributed among autonomous control agents, each associated with an individual participant. These agents make local decisions while coordinating with one another to optimize the global performance of the community.

A promising framework that is increasingly being adopted in the context of sustainable energy and electrical grids is Reinforcement Learning (RL) [18]. RL is well-suited to energy management problems due to its capacity to learn optimal decision-making policies through interaction with complex environments, even in the absence of an explicit model of the system dynamics. This is particularly valuable in RECs, where energy consumption patterns, renewable energy production, and system interactions are inherently stochastic and difficult to model analytically. In this setting, RL naturally extends to the Multi-Agent Reinforcement Learning (MARL) framework, where multiple decentralized agents learn and adapt their strategies through experience to achieve common or conflicting objectives. This makes MARL an ideal fit for decentralized EMS architectures. When combined with mechanism design principles, this approach fosters the alignment of prosumer interests with overall REC goals, promoting cooperative behavior that enhances the overall efficiency, resilience, and stability of the REC.

In this work, we present a novel MARL approach to address the energy management problem within a REC system. Specifically, our contributions are as follows:

- We propose an alternative formulation of the energy management problem in RECs, grounded in a decentralized EMS architecture with a central coordinating entity and aligned with the current Italian regulatory framework.
- We design and implement *INcentive Allocation for Interest Alignment* INAIA, a MARL-based control strategy built on a state-of-the-art RL algorithm combined with mechanism design principles, training the model using historical real-world datasets and a high-fidelity digital twin simulator for Battery Energy Storage Systems (BESSs).[1]
- We conduct an extensive experimental evaluation across multiple scenarios with increasing complexity, assessing the benefits of our approach from both

[1] The codebase is available at `https://github.com/saamur/INAIA-REC`.

individual and community-wise perspectives. Additionally, we conduct ablation studies to evaluate the robustness and effectiveness of the proposed solution.

2 Related Works

In the literature, various approaches have been proposed to tackle the energy management problem in RECs, with increasing emphasis on autonomy, scalability, and coordination. In [17], the authors compare two rule-based EMSs: one focused on individual node self-sufficiency and another on collective REC-level self-sufficiency, showing that the latter achieves comparable local performance while reducing reliance on the national grid. Similarly, in [15], the authors analyze two rule-based and two optimization-based methods, each either household-oriented or community-oriented, and show the benefits of community organization over isolated control. While rule-based strategies can be effective in static environments, they rely on manually crafted heuristics that lack adaptability. Optimization-based approaches are more flexible, but require accurate forecasts, detailed models, and high computational effort, limiting real-time use.

To address these issues, RL solutions have been proposed. In [1], a centralized agent controls community assets to minimize electricity bills. Community members can exchange energy internally within the REC or externally with the grid, incurring different fees. The paper compares Model Predictive Control (MPC) with a centralized RL agent trained via Proximal Policy Optimization (PPO) [21], showing that MPC performs better but is computationally demanding, whereas RL is faster at runtime but costlier to train. In [7], later extended in [16], a centralized RL agent schedules BESS operations to maximize daily social welfare, defined as the sum of all node costs and revenues. Training is guided by a MILP formulation that also serves as a baseline. However, this approach overlooks household self-interest and assumes all BESS units are empty at both the start and end of each day.

Several works have adopted a multi-agent approach to better reflect the decentralized structure of RECs. In [25], intelligent and non-intelligent households trade energy within a private network. The energy price, determined by supply-demand ratio, is lower than the buying price from the external grid and higher than the tariff for selling surplus energy. Intelligent households, trained via Fuzzy Q-learning [6], optimize battery usage to reduce costs. Similarly, in [19], a group of buildings aims to achieve net-zero grid withdrawal by trading energy both internally and externally. Agents, trained with Deep Q-learning [14], share a global reward encouraging collaboration. In [24], prosumers join a peer-to-peer market brokered by a central Energy Trade Supporter (ETS), which also manages a Community Energy Storage (CES). The ETS, trained with Q-learning [23], matches bids and maximizes arbitrage profit. While this supports autonomy, reliance on Q-learning necessitates discrete state/action spaces, limiting expressiveness. Finally, leader-follower approaches have been explored. In [13], a central REC controller acts as leader while households are followers. A Stackelberg

game is formulated at every step, where the controller sets prices for T time slots based on prosumers' demand profiles. Households then update and forward new profiles, and the controller updates prices until a Stackelberg equilibrium is reached. While effective, it is computationally intensive, requires high communication, and Q-learning may yield suboptimal policies due to tabular limitations.

In conclusion, despite progress in REC energy management, existing approaches still face key limitations. Many rely on centralized control or simplified models of batteries and incentives, limiting applicability. Multi-agent learning has been explored, but often neglects the strategic behavior of prosumers and lacks mechanisms to align individual and community objectives, with adaptive incentives rarely embedded in the learning process.

3 Problem Formulation

In this section, we formalize the energy management problem within a REC. A REC is configured as a network of prosumers who cooperate by jointly managing energy production, consumption, and storage to enhance energy exchange efficiency, improve local grid stability, and reduce individual energy costs. Hereafter, we present the modeling of the system at both the node level and the community level, highlighting the associated objectives at each layer. Notably, we consider a REC operating under the Italian regulatory framework, as defined by national decrees [10, 12, 11], which govern, in particular, the distribution of financial incentives for shared energy usage.

Node Level. A node of the REC typically represents a residential micro-grid – which for simplicity we consider a household – with local energy loads, and potentially equipped with a photovoltaic (PV) system and/or a battery storage system. Nodes equipped with a BESS are considered controllable or *active*, as they are capable of participating in energy optimization decisions. Conversely, nodes without storage capabilities are treated as *passive*, contributing to the community primarily through consumption and local generation.

The behavior of each node involves energy exchanges with the external grid, either injecting or withdrawing power. Let $\mathcal{N} := \mathcal{N}_A \cup \mathcal{N}_P$ denote the set of nodes of the network, where $\mathcal{N}_A$ and $\mathcal{N}_P$ are the sets of active and passive nodes, respectively. At each time step t, the net power exchanged with the grid by node $i \in \mathcal{N}$ is given by

$$p_{i,t} = g_{i,t} - d_{i,t} - b_{i,t} \tag{1}$$

where $g_{i,t} \in \mathbb{R}$ and $d_{i,t} \in \mathbb{R}$ are the exogenous signals of the energy renewable generation and demand respectively and $b_{i,t} \in \mathbb{R}$ is the energy charged into or discharged from the battery (zero for *passive* nodes). A positive value of $p_{i,t}$ indicates that the node is injecting power into the grid, while a negative value implies that it is withdrawing energy.

A controller placed on an *active* node $i \in \mathcal{N}_A$, at each time t, must choose a control action which represents the current to apply to the battery. However, such

a decision must be compliant with its physical constraints, to avoid incurring in BESS overloading or overdraining phenomena (further details are provided in Appendix A).

Objective at Node Level. The primary objective of each household is to minimize its electricity-related expenses, including energy bills and the operational costs associated with PV systems and battery storage. Participation in a REC does not alter this fundamental goal. However, a household may willingly deviate from its typical cost-minimizing strategy to adopt a more cooperative behavior that benefits the community if the financial incentives provided by the REC are sufficiently attractive. Formally, a controller placed on an active node $i \in \mathcal{N}_A$ must optimize for the given period of time $T \in \mathbb{N}$ the BESS management to maximize the objective J_i defined as:

$$J_i = \sum_{t=1}^{T} \left[r_{i,t}^{\text{trad}} + r_{i,t}^{\text{deg}} + r_{i,t}^{\text{inc}} \right] \tag{2}$$

where $r_{i,t}^{\text{trad}} \in \mathbb{R}$ represents the traditional revenue or cost from energy exchange with the external grid, $r_{i,t}^{\text{deg}} \in \mathbb{R}$ is the cost due to the battery degradation and $r_{i,t}^{\text{cli}} \in \mathbb{R}$ captures the incentive received from contributing to the community's overall self-consumption. Based on this, we define the *social welfare* as $W = \sum_{i \in \mathcal{N}_A} J_i$, which serves as a metric to quantify the overall fulfillment of the individual objectives of the node agents.

It is important to recall that REC membership is defined by a legal agreement, not by physical separation from the main grid. All nodes interact with the external grid under standard conditions, adhering to the tariffs and contractual obligations of their energy providers. As a result, the energy purchase and sale prices may differ across households and are not influenced directly by REC operations.

Community Level. At the community level, the REC receives financial incentives from the government based on its collective self-consumption. The self-consumption is defined as the minimum, in each hourly interval, between the total injected energy and the energy withdrawn by all REC participants.

Formally, we define the energy injection from node $i \in \mathcal{N}$ as $p_{i,t}^{+} := \max(p_{i,t}, 0)$ and the energy withdrawn from the grid as $p_{i,t}^{-} := |\min(p_{i,t}, 0)|$. The self-consumed energy at the community level at time t is thus given by:

$$U_t := \min \left(\sum_i p_{i,t}^{+}, \sum_i p_{i,t}^{-} \right). \tag{3}$$

This value determines the financial incentives received by the REC for the given time interval and fosters both production and consumption coordination among participants. Following Italian regulatory decrees, we model the financial incentives V_t received by the REC at time t based on the self-consumption U_t as:

$$V_t := U_t \cdot (\alpha + \beta + \varphi_t) \tag{4}$$

where α is a cost refund term, addressing the fact that the electrical grid authority has spared management and repair costs since the self-consumed energy has not been transported over long distances by the national grid, β is a fixed coefficient of the actual incentives, and φ_t is a time-dependent component based on market energy prices.

Objective at Community Level. At the community level, the REC controller is the one responsible for allocating financial incentives to the households. At each time step t, the controller selects an action, representing the distribution of incentive payments across all nodes in the network. This action lies on a simplex, ensuring that the total available incentive is appropriately divided among the participants.

The objective of the community-level controller is to maximize the aggregate self-consumption U_t over time. To achieve this, the incentive allocation strategy must be designed to steer the behavior of individual nodes toward actions that align with the collective goal. This involves leveraging the fact that node-level controllers are inherently self-interested and aim to optimize their own utility functions. By carefully shaping incentives, the community controller can indirectly influence these decentralized agents to act in a way that also benefits the overall REC performance.

4 Proposed Method

In this section, we formalize the energy management problem as a two-phase stochastic game, a mathematical framework well-suited for modeling multi-agent systems. We provide detailed descriptions of the agents operating at both the node and community levels. Eventually, we elaborate on the algorithm design conceived to tackle the stochastic game.

4.1 Two-Phase Stochastic Game Formulation

We model the control problem within the REC as a two-phase stochastic game. In the first phase, at the beginning of the timestep, a set of distributed agents act simultaneously. In the second phase, at the end of the timestep, a centralized controller selects its action based on the global state and the joint actions of all distributed agents. We define the game as a tuple $G := \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{A}_c, P, \mathcal{R}, r_c, \mathcal{Z}, \mathcal{O}, O_c, \gamma \rangle$, where:

- $\mathcal{I}$ is the set of distributed agents;
- $\mathcal{S}$ is the state space of the environment;
- $\mathcal{A} = \bigtimes_{i \in \mathcal{I}} \mathcal{A}_i$ is the joint action space of the distributed agents, with $\mathcal{A}_i$ denoting the set of actions of agent i;
- $\mathcal{A}_c$ is the action space of the central agent;
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{A}_c \times \mathcal{S} \to [0, 1]$ is the state transition function, specifying the probability of transitioning to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ when the distributed agents take joint action $\boldsymbol{a} \in \mathcal{A}$ and the central agent takes action $a_c \in \mathcal{A}_c$;

- $\mathcal{R} = \{r_i | i \in \mathcal{I}\}$ is the set of the reward functions for the distributed agents, where each $r_i : \mathcal{S} \times \mathcal{A} \times \mathcal{A}_c \to \mathbb{R}$ is the reward function of agent i;
- $r_c : \mathcal{S} \times \mathcal{A} \times \mathcal{A}_c \to \mathbb{R}$ is the reward function of the central agent;
- $\mathcal{Z} = \{\mathcal{Z}_i | i \in \mathcal{I} \cup \{c\}\}$, where $\mathcal{Z}_i$ is the set of observations for agent i;
- $\mathcal{O} = \{O_i | i \in \mathcal{I}\}$ is the set of observation functions, where $O_i : \mathcal{S} \to \mathcal{Z}_i$ is the observation function for agent i;
- $O_c : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}_c$ is the observation function of the central agent;
- $\gamma \in [0, 1]$ is the discount factor, capturing the importance of future rewards.

In a stochastic game, each agent aims to maximize its own expected discounted cumulative reward over time. In our setting, the distributed agents are the agents at the REC node level, while the central one is the agent at the community level. Each round of the game is played in two phases: in the first phase, the node agents select and execute their actions simultaneously based on their local observations. In the second phase, the REC agent observes the state and the joint actions of the node agents and selects its action accordingly. After that, all agents receive their respective rewards based on the new state and the actions taken.

4.2 Node Agent

At the node level, each *active* node is equipped with an agent that controls the charging and discharging of its own BESS. Hereafter, we define observation and action spaces and reward function of the node agents. We will omit the subscript i to unload the notation.

Observation Space. The observation space of a node agent is composed of local variables, such as energy demand and generation and the internal battery state, global variables, such as the mean energy injected in the network, and time variables. Formally:

$$o_t := \Big(\sigma_t, K_t, d_{t-1}, g_{t-1}, c^{sell}_{t-1}, c^{buy}_{t-1}, p^{+}_{t-1}, p^{-}_{t-1}, \\ \sin\left(\varphi^d_t\right), \cos\left(\varphi^d_t\right), \sin\left(\varphi^y_t\right), \cos\left(\varphi^y_t\right)\Big)$$

where: $\sigma_t \in [0, 1]$ is the battery SoC at time t; $K_t \in \mathbb{R}$ is the battery temperature at time t; $d_{t-1} \in \mathbb{R}$ is the energy demand of the previous step; $g_{t-1} \in \mathbb{R}$ is the energy generation of the previous timestep; $c^{\text{sell}}_{t-1} \in \mathbb{R}$ and $c^{\text{buy}}_{t-1} \in \mathbb{R}$ are the energy market selling and buying prices, respectively; $p^{+}_{t-1} := \sum_{i \in \mathcal{N}} p^{+}_{i,t-1}$ and $p^{-}_{t-1} := \sum_{i \in \mathcal{N}} p^{-}_{i,t-1}$ are the two components of the REC self-consumption, namely the sum of the energy injections and the sum of the energy withdrawals, respectively; $\varphi^d_t \in [0, 2\pi]$ is the angular position of the clock in a day, given by $\varphi^d_t = \frac{2\pi\tau^d_t}{T^d}$, where $\tau^d_t \in [0, T^d]$ is the current time of the day in seconds and T^d is the total number of seconds in a day; $\varphi^y_t \in [0, 2\pi]$, similarly, is the angular position of the clock in a day, given by $\varphi^y_t = \frac{2\pi\tau^y_t}{T^y}$, where $\tau^y_t \in [0, T^y]$ is the current time of the day in seconds and T^y is the total number of seconds in a day.

Action Space. The action of a node agent is a scalar value $a_t \in \mathbb{R}$, and it represents the current to apply to the battery. If $a_t > 0$ the battery is charged; otherwise, with $a_t < 0$, it is discharged. Physical constraints on the feasible actions are described in Appendix A.

Reward Function. The reward function of a node agent is defined as:

$$r_t := \left[r_t^{\text{trad}} + r_t^{\text{deg}} + r_t^{\text{clip}} \right] + r_t^{\text{inc}} \tag{5}$$

where the first three terms collectively define the local, self-interested reward component r_t^{loc}, while the last term captures the community-level incentive. Specifically, r_t^{trad} accounts for the monetary costs or revenues derived from energy transactions (*purchasing* and *selling*) with the main grid; r_t^{deg} represents the cost associated with BESS degradation; r_t^{clip} is a penalty term applied when an agent attempts physically infeasible actions (more details in Appendix A); r_t^{inc} denotes the agent's share of incentives distributed by the REC, a component that will be further elaborated in subsequent sections.

4.3 REC Agent

At the community level, the centralized REC agent acts to distribute the incentives among the nodes of the network. In particular, the portion of incentives allocated to passive nodes is $V_t^{\mathcal{N}_P} := \frac{|\mathcal{N}_P|}{|\mathcal{N}|} \cdot V_t$, which is divided uniformly among passive households. The remaining amount of incentives, i.e. $V_t^{\mathcal{N}_A} := \frac{|\mathcal{N}_A|}{|\mathcal{N}|} \cdot V_t$, is distributed among the active nodes according to the actions of the REC agent.

Observation Space. The observation space of the REC agent is composed of local data of every active node, some global statistics of the REC and time variables. Formally:

$$o_t^{\text{REC}} := \Big(\{d_{i,t}\}_{i \in \mathcal{N}_A}, \{g_{i,t}\}_{i \in \mathcal{N}_A}, \{b_{i,t}\}_{i \in \mathcal{N}_A}, p_t^+, p_t^-, \bar{d}_t, \bar{g}_t, \bar{b}_t, \\ \sin\left(\varphi_t^d\right), \cos\left(\varphi_t^d\right), \sin\left(\varphi_t^y\right), \cos\left(\varphi_t^y\right) \Big)$$

where: $d_{i,t} \in \mathbb{R}$ is the current energy demand of active node i; $g_{i,t} \in \mathbb{R}$ is the current energy generation of active node i; $b_{i,t} \in \mathbb{R}$ is the current energy used to charge/discharge the battery of active node i; $p_t^+ := \sum_{i \in \mathcal{N}} p_{i,t}^+$ and $p_t^- := \sum_{i \in \mathcal{N}} p_{i,t}^-$ are the two components of the REC self-consumption, namely the sum of the energy injections and the sum of the energy withdrawals, respectively; $\bar{d}_t \in \mathbb{R}$ and $\bar{g}_t \in \mathbb{R}$ are the mean demand and generation, respectively, among all nodes; $\bar{b}_t \in \mathbb{R}$ is the mean energy for charging/discharging the batteries of active nodes; $\varphi_t^d \in [0, 2\pi]$ is the angular position of the clock in a day, given by $\varphi_t^d = \frac{2\pi\tau_t^d}{T^d}$, where $\tau_t^d \in [0, T^d]$ is the current time of the day in seconds and T^d is the total number of seconds in a day; $\varphi_t^y \in [0, 2\pi]$, similarly, is the angular position of the clock in a day, given by $\varphi_t^y = \frac{2\pi\tau_t^y}{T^y}$, where $\tau_t^y \in [0, T^y]$ is the current time of the day in seconds and T^y is the total number of seconds in a day.

Action Space. The action space of the REC agent is the $(|\mathcal{N}_A| - 1)$-dimensional simplex, that is, the set of all possible vectors of proportions specifying the share of incentives each node agent receives. Formally:

$$\boldsymbol{a}_t^{\text{REC}} \in \Delta^{|\mathcal{N}_A|-1} \tag{6}$$

The vector of the incentive rewards of node agents will therefore be:

$$\boldsymbol{r}_t^{\text{inc}} := V_t^{\mathcal{N}_A} \cdot \boldsymbol{a}_t^{\text{REC}} \tag{7}$$

Reward Function. The reward function for the REC agent is simply the self-consumption of the REC, i.e. $r_t^{\text{REC}} := U_t$.

4.4 Algorithm

From the formulation of the stochastic game, we derive that every round of the game can be seen as a general-sum game, where agents can both cooperate and compete. Although node agents have a reward function that depends largely on local self-interested dynamics, by collaborating effectively they can increase the REC's self-consumption and the overall incentives as a consequence, resulting in higher social welfare. General-sum games are notoriously difficult to solve, as the presence of multiple learning agents makes the environment non-stationary from the perspective of each agent. Moreover, if agents do not explicitly account for the behavior of others, the learning process often converges to sub-optimal solutions.

To foster collaboration among node agents, we designed INAIA (*INcentive Allocation for Interest Alignment*), a meta-learning algorithm inspired by *Learning with Opponent-Learning Awareness* (LOLA) [5]. While in standard MARL settings each agent optimizes its own expected discounted return independently, without explicitly considering the learning dynamics of other agents, LOLA enables each agent to anticipate and incorporate the expected policy updates of others into its own optimization process, effectively accounting for their learning behavior. With INAIA we apply this idea to the REC agent by updating its parameters while accounting for the effect of its actions on the future behavior of the node agents. However, our setting presents unique characteristics: the REC agent differs from standard RL agents in that its immediate reward depends solely on the actions of the node agents, not on its own current actions. Moreover, the REC agent does not directly influence the environment's state dynamics; instead, its role is limited to shaping the reward signals received by the node agents. Consequently, the only means by which the REC agent can improve its objective is by influencing the learning trajectories of the node agents over time. This perspective naturally frames our problem through the lens of mechanism design.

Mechanism design is the subfield of game theory studying techniques for constructing rules (called *mechanisms*) that guide self-interested agents toward outcomes that optimize a predefined social welfare function. When applied to

Algorithm 1 INAIA

Require: Iterations $I \in \mathbb{N}$, REC episodes $E_{\text{REC}} \in \mathbb{N}$, node episodes $E_{\text{node}} \in \mathbb{N}$, simulated steps $K \in \mathbb{N}$, learning rates $\alpha_{\text{REC}}, \alpha_{\text{node}}, \alpha_{\text{node}}^{\text{sim}} \in \mathbb{R}^+$

1: Initialize node parameters $\theta \in \Theta$, REC parameters $\phi \in \Phi$
2: **for** $i = 1, \ldots, I$ **do**
3: Save $\theta_{\text{orig}} \leftarrow \theta$
4: **for** $e = 1, \ldots, E_{\text{REC}}$ **do**
5: **for** $k = 1, \ldots, K$ **do**
6: Collect trajectory τ_{sim} using π_θ^B and π_ϕ^{REC}
7: Simulate update on θ: $\theta \leftarrow \theta - \alpha_{\text{node}}^{\text{sim}} \nabla_\theta \mathcal{L}_{\text{node}}(\tau_{\text{sim}})$
8: Collect new trajectory τ'_{sim} with updated θ
9: Compute REC loss: $\mathcal{L}_{\theta,\phi}(\tau'_{\text{sim}}) = -\frac{1}{|\tau'_{\text{sim}}|} \sum_{\tau'_{\text{sim}}} R_{\theta,\phi}^{REC}$
10: Update ϕ: $\phi \leftarrow \phi - \alpha_{\text{REC}} \nabla_\phi \mathcal{L}_{\theta,\phi}(\tau'_{\text{sim}})$
11: **end for**
12: Reset $\theta \leftarrow \theta_{\text{orig}}$
13: **end for**
14: **for** $e = 1, \ldots, E_{\text{node}}$ **do**
15: Collect trajectory τ using π_θ^B and π_ϕ^{REC}
16: Update θ: $\theta \leftarrow \theta - \alpha_{\text{node}} \nabla_\theta \mathcal{L}_{\text{node}}(\tau)$
17: **end for**
18: **end for**

MARL, this translates into shaping the agents' reward functions to bias their learning process towards regions of the state space that are globally more desirable. In our setting, we can consider the REC agent as the mechanism designer that shapes the incentive rewards of node agents to promote behaviors that enhance overall self-consumption, i.e., our social welfare measure.

We now provide a detailed explanation of Algorithm 1, which implements INAIA, outlining the rationale behind each of its components. Firstly, we need to initialize the number of iterations I of the algorithm, each composed of E_{REC} episodes, in which the REC's policy is updated, and E_{node} episodes, where node agents' policies are updated. These two parameters allow to vary the learning speed of the two types of agents. Each episode in E_{REC} is further divided into K simulated steps. In addition, we need to provide the learning rates: α_{REC} for the REC agent, α_{node} and $\alpha_{\text{node}}^{\text{sim}}$ for the node agents, where the former is used for the updates of their policies and the latter for the simulated updates. After initializing the node agents' parameters θ and the REC's parameters ϕ, the algorithm can start. In each iteration, after saving the node agent's parameters, we start the REC's episodes and their simulated steps. Each step of E_{REC} is structured as follows: after collecting a trajectory (line 6) we simulate an update for the node agents (line 7). Then, we collect a new trajectory with the updated parameters (line 8) and we compute the loss of the REC agent, that is the negative mean of the self-consumption (line 9). Hence, we update the parameters using the gradient of the loss, effectively maximizing the self-consumption (line 10). At the end of the episode, we restore the parameters of the node agents, since the previous updates were used only for updating the REC's parameters

(line 12). In the second part of the iteration, namely the node agents' episodes, we perform classic updates, i.e., we collect a trajectory (line 15) and we perform the actual update of the node agents' parameters (line 16).

Notice that the self-consumption used in the REC loss (Line 9) depends on the updated node agents' parameters θ, which in turn depend on the REC agent's parameters ϕ. Therefore, the gradient $\nabla_\phi \mathcal{L}_{\theta,\phi}$ is calculated also through the parameters of the node agents. This is how the REC agent takes the node agents' learning dynamics into account when it distributes the incentives, shaping their reward functions. In addition, another crucial aspect of INAIA is that, when collecting trajectories to update the parameters of the REC agent (Line 8), the actions of the node agents are deterministic, and not stochastic as usual. This helps in decreasing the variance of the policy updates and results in a better learning process for the REC agent.

5 Experiments

In this section, we present the experiments conducted to validate the proposed approach. Specifically, we compare INAIA against reference baselines for the REC agent, while keeping the node agents always trained using PPO [21] with fixed hyperparameters across all experiments. This ensures that performance differences can be attributed solely to the behavior of the REC agent.

We evaluate our method in two complementary scenarios. The first investigates the scalability of the system as the number of active nodes increases. The second focuses on a representative heterogeneous REC configuration, allowing for a deeper analysis of both training dynamics and final performance. In this latter scenario, we also perform an ablation study.

The experimental campaign is conducted, for both training and testing, within a realistic simulation environment. In particular, we leverage a simulator based on a BESS digital twin [20], which accurately replicates battery dynamics. Additionally, we use real-world datasets collected in Italy, covering energy consumption, photovoltaic (PV) generation, market price evolution and ambient temperature. Further details on datasets are provided in Appendix B.2.

The baselines against which we compare our approach are the following:

- **PPO REC agent**: The REC node is trained using PPO as for the node agents.
- **Rule-based REC agent**: Since the self-consumption is the minimum between the total injected and withdrawn energy, a reasonable rule-based policy for the REC agent is to reward the node agents based on their contributions to the lower of the two components. Thus, the component of the REC's action for agent i is:

$$a_{c,t}^{(i)} = \begin{cases} \frac{1}{|\mathcal{N}|} & \text{if } p_t^+ = 0 \vee p_t^- = 0 \\ \frac{p_{i,t}^+}{p_t^+} & \text{if } p_t^+ \leq p_t^- \\ \frac{p_{i,t}^-}{p_t^-} & \text{if } p_t^+ > p_t^- \end{cases} \tag{8}$$

Notice that the first branch is needed only to avoid divisions by zero; the REC agent's reward will be null in any case, since the self-consumption is also null.
- **LOLA**: The REC agent is trained using LOLA.

The method is implemented in Python, leveraging the JAX [3] and Flax [8] libraries, the former for efficient JIT-compiled computation, and the latter for neural network modeling built on top of it. A detailed description of network architectures and hyperparameters used in our solution are provided in Appendix B.3.

5.1 Scaling Number of Node Agents

In the first experiment, we explore how INAIA and the presented baselines behave in increasingly complex network configurations. We consider a REC with a variable number of active nodes, namely 3, 5, and 8, and no passive nodes. The power generation will be the same for all nodes: solar panels oriented South with a nominal peak power of 3 kW.

We report the results for the three configurations in Table 1. The values represent the cumulative social welfare (left) and total self-consumption (right) over five years of testing. For each experiment, we applied early stopping based on the social welfare metric to select the best-performing model. All agents were tested using the same set of energy demand profiles to ensure comparability. As can be seen, our approach consistently outperforms all baselines across every setting, maximizing both social welfare and self-consumption. Notably, the performance gap between INAIA and the other methods widens as the number of active nodes increases, demonstrating the scalability and robustness of our method in scenarios that more closely resemble real-world REC networks involving dozens of participants. Among the baselines, the best strategy is the *rule-based* one, suggesting that a straightforward application of MARL, without careful problem formulation and exploitation of domain-specific structure, may lead to suboptimal results. Finally, our approach exhibits superior space efficiency compared to the original LOLA implementation, which proved computationally infeasible to run on our servers (specs detailed in Appendix C).

5.2 Reference Scenario

In this experiment, we consider a REC composed of three active nodes and one passive node. Each active household is equipped with a 3 kW PV system, but the panels are oriented differently: one 60° East-facing (favoring morning production), one South-facing (producing most around noon), and one 60° West-facing (favoring late afternoon production). The passive node has no energy generation systems. The difference in panel orientation leads to naturally shifted production peaks across the day, encouraging a form of temporal specialization in energy storage behavior.

Table 1: Comparison of algorithms in different scenarios of increasing difficulties due to the growing numbers of agents (best results in bold).

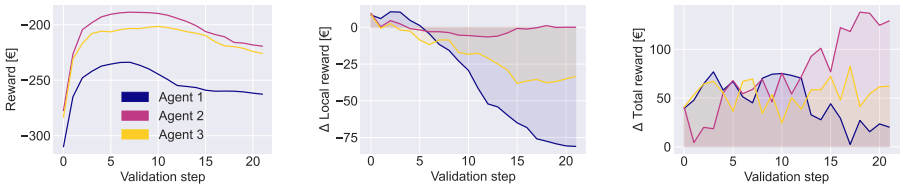
Algorithm	Social Welfare [€]			Self-Consumption [MWh]		
	3	5	8	3	5	8
Rule-based	-2651	-4016	-4908	3.67	8.81	11.66
PPO	-2673	-4104	-5125	3.46	7.82	10.69
LOLA	-2875	-4637	—*	3.28	5.90	—*
INAIA	**-2579**	**-3780**	**-4264**	**4.68**	**16.65**	**85.74**

* Results are missing due to unfeasible computation for too demanding hardware requirements.

To evaluate the benefit of household in participating to a REC, we compare the agents' performance with and without participation in the community. In Figure 1a, we plot the mean yearly validation rewards during training, obtained by the agents when the REC is not in place – that is, when they receive no incentives to deviate their local self-interested strategy. Figure 1b shows the change in local rewards due to the REC, computed as the difference between the local reward with the REC and the local reward without it (from Figure 1a). The curves are mostly negative, indicating that agents sacrifice part of their local reward when participating in the REC. This is expected in our incentive-driven mechanism design where agents are encouraged to act cooperatively even if it occasionally conflicts with their narrow self-interest. Finally, Figure 1c displays the change in total rewards under the REC, again relative to the rewards in the non-REC setting. Here, all curves are positive, demonstrating that the designed incentive scheme effectively aligns individual and collective interests, making REC membership beneficial for every agent. The confidence intervals have been omitted from the plots, since they would be very wide due to the high heterogeneity of the demand profiles. However, every agent has been tested on the same profiles throughout the whole training in both REC and non-REC cases, so the curves are nonetheless comparable. From a self-consumption point of view, the setting with the REC in place trained with INAIA obtained a value 35.1% higher with respect to the case without REC and 17.3% higher with respect to the case with the REC trained with the rule-based baseline.

5.3 Ablation Study

In the reference scenario, we conduct an ablation study to further validate the effectiveness of our approach. Specifically, in our proposed method, node agents act deterministically during the REC training phase to reduce the variance in the gradient estimates for the REC agent (lines 5–11). When this mechanism is ablated – sampling node agents' actions from distributions – the increased variability in the REC agent's gradient estimation negatively affects its learn-

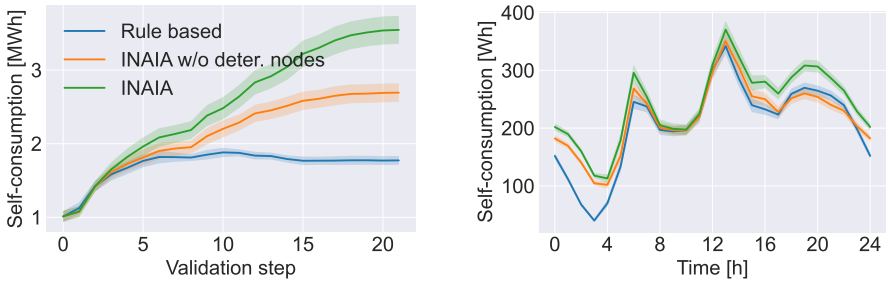


(a) Reward w/o REC (b) Variation of $r_{i,t}^{\text{loc}}$ (c) Variation of $r_{i,t}$

Fig. 1: Comparison of the yearly reward of setting without and with REC agent. Figures 1b and 1c show the impact of the REC agent in local and total rewards w.r.t. the case without it (Figure 1a).

ing process. As shown in Figure 2a, the self-consumption performance during training for the ablated variant is consistently outperformed by our approach, with the rule-based baseline performing worst overall. The benefits of the proposed mechanism are further corroborated by Figure 2b, which illustrates the average daily self-consumption profile aggregated over five years of test data. In this plot, our solution achieves higher hourly self-consumption than the other methods, demonstrating effective coordination capabilities among node agents.

As reported in these plots, INAIA improves the training stability and allows for faster gains in self-consumption than its ablated version. While both variants ultimately converge to similar long-term social welfare outcomes – yielding -2261€ for our method and -2274€ for the ablated version over five years – the variance reduction introduced by our approach significantly improves REC self-consumption and overall learning efficiency. Overall, these results highlight the strengths of INAIA: it not only improves individual agent returns through coordinated behavior but also enhances global efficiency by increasing self-consumption.



(a) Validation self-consumption (b) Daily self-consumption

Fig. 2: Self-consumption with 95% confidence intervals

6 Conclusions

In this work, we presented INAIA, a novel MARL framework for the optimal energy management of RECs grounded in a decentralized architecture. Our method incorporates mechanism design principles to align the self-interested objectives of individual households with the global community goal of maximizing energy self-consumption. Through a comprehensive experimental evaluation based on real-world data, we demonstrated that our approach substantially outperforms several competitive baselines across a range of settings. In particular, in a representative heterogeneous scenario, our method increases self-consumption by 17.3% compared to the best-performing baseline. Ablation studies further confirm the robustness of the algorithm and the impact of specific design choices.

While our approach is inherently designed to encourage REC participants to collaborate and specialize in complementary roles, it naturally results in varying incentive distributions across nodes. This behavior arises from the absence of explicit constraints on allocation policies, allowing the algorithm to prioritize agents that demonstrate higher responsiveness to incentives. Although such flexibility enhances the system's overall efficiency, it may raise concerns regarding perceived fairness among community members in real-world deployments. To address this, in future works, we will investigate the integration of *fairness* mechanisms aimed at promoting a more balanced and equitable distribution of incentives, aligning technical performance with social acceptability.

Acknowledgments. This work has been funded by the Research Fund for the Italian Electrical System under the Three-Year Research Plan 2022-2024 (DM MITE n. 337, 15.09.2022), in compliance with the Decree of April 16th, 2018.

References

1. Aittahar, S., Bolland, A., Derval, G., Ernst, D.: Optimal Control of Renewable Energy Communities subject to Network Peak Fees with Model Predictive Control and Reinforcement Learning Algorithms (Feb 2024)
2. Basit, M.A., Dilshad, S., Badar, R., Sami ur Rehman, S.M.: Limitations, challenges, and solution approaches in grid-connected renewable energy systems. International Journal of Energy Research **44**(6), 4132–4162 (2020)
3. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018), `http://github.com/jax-ml/jax`
4. Ellabban, O., Abu-Rub, H., Blaabjerg, F.: Renewable energy resources: Current status, future prospects and their enabling technology. Renewable and sustainable energy reviews **39**, 748–764 (2014)
5. Foerster, J.N., Chen, R.Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., Mordatch, I.: Learning with Opponent-Learning Awareness (Sep 2018)
6. Glorennec, P.: Fuzzy Q-learning and dynamical fuzzy Q-learning. In: Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference (Jun 1994)

7. Guiducci, L., Palma, G., Stentati, M., Rizzo, A., Paoletti, S.: A Reinforcement Learning approach to the management of Renewable Energy Communities. In: 2023 12th Mediterranean Conference on Embedded Computing (MECO). pp. 1–8 (Jun 2023), iSSN: 2637-9511
8. Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., van Zee, M.: Flax: A neural network library and ecosystem for JAX (2024), `http://github.com/google/flax`
9. Huld, T., Müller, R., Gambardella, A.: A new solar radiation database for estimating PV performance in Europe and Africa. Solar Energy **86**(6) (Jun 2012)
10. Italian Government: Testo coordinato del decreto-legge 30 dicembre 2019, n. 162 (2019), `www.gazzettaufficiale.it/eli/id/2020/02/29/20A01353/sg`
11. Italian Government: Decreto 16 settembre 2020 (2020), `https://www.gazzettaufficiale.it/eli/id/2020/11/16/20A06224/sg`
12. Italian Parliament: Legge 28 febbraio 2020, n. 8 (2020), `www.gazzettaufficiale.it/eli/id/2020/02/29/20G00021/sg`
13. Lai, B.C., Chiu, W.Y., Tsai, Y.P.: Multiagent Reinforcement Learning for Community Energy Management to Mitigate Peak Rebounds Under Renewable Energy Uncertainty. IEEE Transactions on Emerging Topics in Computational Intelligence **6**(3), 568–579 (Jun 2022)
14. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (Feb 2015), publisher: Nature Publishing Group
15. Mustika, A.D., Rigo-Mariani, R., Debusschere, V., Pachurka, A.: A two-stage management strategy for the optimal operation and billing in an energy community with collective self-consumption. Applied Energy **310**, 118484 (Mar 2022)
16. Palma, G., Guiducci, L., Stentati, M., Rizzo, A., Paoletti, S.: Reinforcement Learning for Energy Community Management: A European-Scale Study. Energies **17**(5), 1249 (Jan 2024)
17. Pasqui, M., Felice, A., Messagie, M., Coosemans, T., Bastianello, T.T., Baldi, D., Lubello, P., Carcasci, C.: A new smart batteries management for Renewable Energy Communities. Sustainable Energy, Grids and Networks **34**, 101043 (Jun 2023)
18. Ponse, K., Kleuker, F., Fejér, M., Álvaro Serra-Gómez, Plaat, A., Moerland, T.: Reinforcement learning for sustainable energy: A survey (2024)
19. Prasad, A., Dusparic, I.: Multi-agent Deep Reinforcement Learning for Zero Energy Communities. In: 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe). pp. 1–5 (Sep 2019)
20. Salaorni, D., Bianchi, F., Colnago, S., Barisione, A., Trovò, F., Restelli, M.: A novel digital twin for battery energy storage systems in micro-grids (Jan 2025)
21. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017)
22. Staffell, I., Pfenninger, S., Johnson, N.: A global model of hourly space heating and cooling demand at multiple spatial scales. Nature Energy **8** (09 2023)
23. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3) (May 1992)
24. Zang, H., Kim, J.: Reinforcement Learning Based Peer-to-Peer Energy Trade Management Using Community Energy Storage in Local Energy Market. Energies **14**(14), 4131 (Jan 2021)
25. Zhou, S., Hu, Z., Gu, W., Jiang, M., Zhang, X.P.: Artificial intelligence based smart energy community management: A reinforcement learning approach. CSEE Journal of Power and Energy Systems **5**(1), 1–10 (Mar 2019)

A Problem Constraints

As described in the main paper, the actions that the node agents can apply on the BESS they control must satisfy some physical constraints. Considering a single node and omitting the subscript i for clarity, at each time step t, the action that represents the current to apply to the BESS must be bounded as follows:

$$a_{\mathbf{min},t} \leq a_t \leq a_{\mathbf{max},t} \tag{9}$$

where:

$$a_{\mathbf{min},t} := \frac{\sigma_{\mathbf{min}} - \sigma_t}{\Delta_t} \cdot C_t \tag{10}$$

and

$$a_{\mathbf{max},t} := \frac{\sigma_{\mathbf{max}} - \sigma_t}{\Delta_t} \cdot C_t \tag{11}$$

σ_t is the current State of Charge (SoC), $\sigma_{\mathbf{min}}$ and $\sigma_{\mathbf{max}}$ are the minimum and maximum SoC of the battery respectively, Δ_t is the length of each time step and C_t is the capacity of the battery.

Also the clipping component of the reward for the node agents, r_t^{clip} depends on these bounds. In particular:

$$r_t^{\text{clip}} = \begin{cases} 0 & a_{\mathbf{min},t} \leq a_t \leq a_{\mathbf{max},t} \\ -\left(a_{\mathbf{min},t} - a_t\right)^2 & \text{if } a_t < a_{\mathbf{min},t} \\ -\left(a_t - a_{\mathbf{max},t}\right)^2 & \text{if } a_t > a_{\mathbf{max},t} \end{cases} \tag{12}$$

B Experiment Details

B.1 Incentives parameters

In the main paper we showed that the financial incentives given to the whole REC are:

$$V_t := U_t \cdot (\alpha + \beta + \varphi_t)$$

- α is a cost refund term that can slightly change from year to year, and has a value of about 8–10 €/MWh. In our experiments we used 8 €/MWh;
- β is the constant term of the incentives. Its base value is 80 €/MWh, but depending on what region the REC is located in, 4 €/MWh (in center regions) or 10 €/MWh (in northern regions) are added to it. We used its base value;
- φ_t is the variable part of the incentives, that changes depending on the market energy price. It has value between 0 and 40 €/MWh.

B.2 Data

To train the agents we used data related to four domains: demand, generation, energy market and temperature. For the energy demand, we used a dataset of 397 year-long profiles with hourly data on the energy consumption of Italian households. For the energy generation, we used synthetic data, generated with the PVGIS tool [9], developed by the European Commission's Joint Research Centre. This tool takes into account the location and orientation of the solar panels, the losses of the conversion to electricity, as well as the weather, other atmospheric data and many other variables. This results in high-quality data that is very representative of real solar panels' energy output. We generated data from 2015 to 2018 for training and from 2019 for testing. For market prices, we used the time series of the Italian energy market as the basis for the buying price. To simulate the common difference between purchase and sale prices, where households typically pay more for energy than they earn from selling it, we modeled the selling price using the same time series, shifted downward by 87 €/MW. For the external temperature, we used a dataset sourced from [22] and obtained from `https://www.renewables.ninja`. It contains daily measurements of the same periods as for the generation.

B.3 Network architectures and hyperparameters

In both the experiments with INAIA and with the baselines, the node agents are trained with PPO, so each of them has an actor network and a critic network. Both of them have two hidden layers with 64 and 32 neurons respectively. In Table 2 it is possible to see the main hyperparameters for the training of the node agents.

Table 2: Node agents training hyperparameters.

Hyperparameter	**Value**	**Value for simulated update**
Initial learning rate	$5 \cdot 10^{-5}$	$1 \cdot 10^{-2}$
Final learning rate	$1 \cdot 10^{-7}$	-
Learning rate schedule	cosine	constant
Number of steps	8192	256
Number of minibatches	32	2
Number of epochs	10	3
Discount factor	0.99	0.99
GAE lambda	0.98	0.98

The REC agent, in the case of IPPO and LOLA, includes an actor and a critic network. These networks have a multi-branch architecture: each branch

consists of a network with two hidden layers of 64 and 32 neurons. Each branch takes in input the data of a single node agent i, namely $d_{i,t}$, $g_{i,t}$ and $b_{i,t}$, along with the aggregated REC-level data (p_t^+, p_t^-, $\bar{d}_t$, $\bar{g}_t$), and the time variables. The scalar outputs of all the branches are then concatenated. In the actor network, this concatenated vector serves as the parameter of the Dirichlet distribution used to sample actions in PPO. In the critic network, the branch outputs are instead combined via a weighted sum using a parameter vector. For INAIA no critic network is required, and the policy is deterministic: in this case, the concatenated branch outputs pass through a softmax layer to produce the action. The main hyperparameters for the training of the REC agent are available in Table 3.

Table 3: REC agent training hyperparameters.

Hyperparameter	**INAIA**	**PPO**	**LOLA**
Initial learning rate	$8 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	0.1
Final learning rate	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$
Learning rate schedule	cosine	cosine	cosine
Number of steps	256	8192	8192
Number of minibatches	-	64	-
Number of epochs	-	10	-
Discount factor	-	0.99	0.99
Number of simulated steps	32	-	-
E_{REC} (see Algorithm 1)	3	1	-
E_{node} (see Algorithm 1)	1	1	-

C Hardware specifications

The server used to run the experiments is equipped with an AMD EPYC 7453 28-Core Processor, 24 GB of RAM and a Nvidia L4 with 24 GB of vRAM. The bottleneck for the missing experiments has been the amount of vRAM available.