BISKY: DYNAMIC BIPARTITE COORDINATION FOR OPEN MULTI-AGENT ENERGY SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Distributed energy resources make local decisions but are coupled through shared power grids and energy markets. Multi-agent reinforcement learning can coordinate these decisions. During deployment, the active agent set may change as resources join or leave. Agents that remain active may also change their operating patterns or face new grid and market conditions. Newly active agents have limited local experience, so both groups must adapt. We consider a decentralized setting in which raw observations, actions, and rewards remain local to each agent. The challenge is to share useful learned knowledge as agents join, leave, or change behavior without collecting joint trajectories for a centralized critic.

To address these limitations, we propose **BISKY**, a decentralized framework that shares knowledge through a dynamic bipartite graph. The graph connects active agents to channels representing recurring operating behaviors. Each agent forms a compact routing signature from its local characteristics and recent experience and connects to the channel matching its current behavior. Channels combine compact belief-adapter or policy updates from connected agents and return the resulting knowledge for local use. Each agent retains its own policy and value function, and channel connections can change as the active set or an agent's behavior changes. We evaluate BISKY in peer-to-peer energy trading and electric-vehicle charging with fixed and changing active agent sets. In the closed setting, BISKY reaches about 95% of the cost savings, 88% of the grid reduction, and 88% of the electric-vehicle charging reward achieved by MAPPO, whose centralized critic uses joint system information. Results show that behavior-based channels improve coordination without requiring global information or a fixed agent population.

1 INTRODUCTION

Modern energy systems increasingly integrate distributed energy resources, including rooftop solar (Feng et al., 2020) and other small generators (Wang et al., 2021), batteries, and electric vehicles (Zhang et al., 2021; 2022; Vinitsky et al., 2022; Vazquez-Canteli et al., 2020a;b), within shared power grids and peer-to-peer energy markets (Qiu et al., 2021a). Each resource is managed by a local decision maker that uses local observations rather than relying on a single central controller (Chen et al., 2020). Their decisions are made locally but jointly affect feeder loading, aggregate grid import, peak demand, and market prices. Coordination also depends on which resources are active. At any time t, the coordination algorithm operates over an active set $\mathcal{A}_t$, which identifies the agents participating at that time and may vary throughout deployment. Multi-agent systems with a changing active set are known as **open multi-agent systems** (Chandrasekaran et al., 2016; Eck et al., 2020). In our setting, an agent that remains in $\mathcal{A}_t$ may also change its operating behavior as prices, seasons, or energy needs shift. We study decentralized learning in this setting.

We formulate this coordination task as a multi-agent reinforcement learning (MARL) problem (Littman, 1994; Zhang et al., 2018), in which agent i selects actions according to a local policy π_i. We consider three requirements for MARL in this decentralized open setting. **Locality**: each agent has its own policy, value function, and rollout data; raw observations, actions, and rewards are not shared, and learning does not rely on a centralized critic. **Coordination**: agents should account for the combined effects of their decisions under shared grid and market constraints without communicating directly with every other agent. **Openness**: the coordination mechanism should continue to operate as the active set $\mathcal{A}_t$ and agents' operating behaviors change over time.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Existing MARL approaches use centralized critics (Foerster et al., 2018; Yu et al., 2022), independent learning (De Witt et al., 2020), or agent-interaction models (Nayak et al., 2023; Iqbal & Sha, 2019; Wang et al., 2020a). Centralized methods require information from multiple agents, and independent learning provides no mechanism for sharing coordination knowledge. Centralized methods with fixed-size joint inputs may also require changes to their training setup when the active population changes (Wang et al., 2020b). Agent-interaction models can represent relationships among agents, but often need information about other agents or updates to agent-level relations when participation changes. Open ad hoc methods such as GPL and POAM model other agents from observable behavior and keep some agents' policies fixed (Rahman et al., 2021; Wang et al., 2024a). In our setting, every active agent learns locally, the active set and individual behavior may change, and no agent observes another agent's raw trajectory. A newly active agent may need knowledge that other agents have learned under similar operating conditions. An agent that changes behavior may need knowledge learned under a different operating pattern. What is missing is a mechanism that allows independently learning agents to share and reuse coordination knowledge without requiring direct observations of other agents.

To address these limitations, we propose **BISKY**, a decentralized coordination framework built on a dynamic bipartite graph. The key design is to separate the agents that make decisions from the channels that combine coordination knowledge. The graph places active agents in one partition and behavior channels in the other. Each channel corresponds to a recurring operating pattern rather than a particular agent. Knowledge contributed to a channel can remain available when the active set changes. When $\mathcal{A}_t$ or an active agent's behavior changes, BISKY updates the affected connections and channels rather than rebuilding the entire coordination structure.

Each agent constructs a compact routing signature from its local characteristics and recent trajectory to select a channel matching its current operating behavior. Each agent has its own actor, critic, and rollout buffer, and keeps its raw observations, actions, and rewards local. The agent sends a compact update to the selected channel, which combines updates from agents with related operating patterns and returns the resulting knowledge for local use. This process requires neither global observations nor a centralized critic.

The form of the compact update depends on the channel-aggregation variant. In the belief-sharing variant, each agent trains a local belief model and sends a residual-adapter update. The channel combines these updates, and each agent evaluates the returned adapter update using held-out local data before using it. In the policy-sharing variant, each agent sends a local policy-gradient update, and the channel combines updates from agents associated with related operating patterns to support adaptation after $\mathcal{A}_t$ changes. The channels do not select actions, and action selection remains local.

Our main contributions are as follows:

- We introduce **BISKY**, a decentralized MARL framework that uses a dynamic bipartite graph to associate coordination knowledge with recurring operating behaviors rather than fixed agent identities.
- We develop one channel-aggregation mechanism that combines residual-adapter updates with local validation and another that combines policy-gradient updates.
- We evaluate BISKY in peer-to-peer energy trading and electric-vehicle charging under changes in the active agent set, showing improved open-system performance over decentralized and open ad hoc teamwork baselines.

2 Related Work

We first formulate the decentralized open MARL problem and then discuss existing methods for coordination under changes in the active agent set.

Problem Formulation: We formulate decentralized energy coordination as a multi-agent reinforcement learning problem with a time-varying active set $\mathcal{A} * t$ (Littman, 1994; Zhang et al., 2018; Oliehoek et al., 2016). At time t, $\mathcal{A} * t$ denotes the energy-resource agents participating in the system. Each agent $i \in \mathcal{A} * t$ receives a local observation $o * i, t$, selects an action $a * i, t$ according to its local policy π_i, and receives a reward $r * i, t$. Although each action is selected locally, the joint action $\mathbf{a} * t = (a * i, t)_{i \in \mathcal{A}_t}$ determines the system transition and the resulting grid and market

outcomes. During deployment, both $\mathcal{A}_t$ and the operating behavior of an active agent may change (Chandrasekaran et al., 2016; Eck et al., 2020). Each agent updates its policy and value function using its own local trajectory, and no centralized learner observes the global state or joint action.

Centralized and Independent MARL: Previous work has applied MARL to coordinate distributed energy resources under shared grid and market constraints (Wang et al., 2021; Zhang et al., 2021; 2022). These approaches commonly use centralized training with decentralized execution or independent learning. Centralized training with decentralized execution uses system-wide information during training and allows each agent to select actions using a local policy during deployment (Lowe et al., 2017; Foerster et al., 2018; Rashid et al., 2020; Yu et al., 2022; Sunehag et al., 2017). In Multi-Agent Proximal Policy Optimization (MAPPO), agent i selects actions using a local policy $\pi_i(a_{i,t} \mid o_{i,t})$, but a centralized value function estimates the system return from the global state or information collected from multiple agents (Yu et al., 2022). This estimate is used to compute the advantages that guide the local policy updates. In the standard fixed-agent formulation, the input to the centralized value function is defined for the agents included during training. When $\mathcal{A}_t$ changes, this input may no longer match the new set of active agents. MAPPO also depends on collecting information from multiple agents during training, which is unavailable when raw agent trajectories must remain local.

Independent Proximal Policy Optimization (IPPO) trains each agent independently (De Witt et al., 2020). Agent i trains both its policy π_i and value function V_i using its own observations, actions, rewards, and rollout data. Its learning inputs do not depend on the identities or number of other active agents, so the same training structure can continue to operate as $\mathcal{A}_t$ changes. However, changes in the policies of other agents appear only through their effects on agent i's local observations and rewards. IPPO provides no mechanism for sharing coordination knowledge across agents. Newly active agents and agents with changing behavior can learn only from subsequent local interactions, which can require many samples before they coordinate effectively with the current active set.

Coordination through Agent Graphs: Graph-based MARL (Liu et al., 2020; Guestrin et al., 2001) represents active agents as nodes and their interactions as edges, allowing information to be aggregated across connected agents. InforMARL defines a graph from the agents and their locally observed neighborhoods and uses a graph neural network to aggregate neighborhood information for both the actor and critic (Nayak et al., 2023). Because the aggregation does not depend on a fixed ordering or number of neighbors, InforMARL can operate with different numbers of agents. However, its coordination context at each time step is computed from the agent nodes and connections currently observed. When an agent becomes inactive, its node and messages are no longer part of this context, although the learned graph-network parameters remain available. A newly active agent receives information from its current neighbors rather than knowledge accumulated by previous agents with similar operating behavior. Moreover, local neighborhood connections may not capture relationships between distant energy resources with similar demand, generation, or price-response patterns. Agent graphs therefore support changes in graph size but do not directly maintain coordination knowledge associated with recurring operating behaviors as the active agents change.

Ad Hoc and Open Ad Hoc Teamwork: Ad hoc teamwork (Stone et al., 2010; Wang et al., 2024a) trains one ad hoc agent to cooperate with other agents that were not jointly trained with it. Controlling Type Confounding in Ad Hoc Teamwork (CTCAT) (Xing et al., 2023) represents the other agents through behavior types and adjusts the training feedback so that the learned response does not depend too strongly on the particular agents observed during training (Xing et al., 2023). Graph-based Policy Learning (GPL) extends this setting by allowing the set of fixed-policy agents interacting with the learner to change over time (Rahman et al., 2021). GPL uses an agent model to predict their actions and a graph-based joint action-value function to estimate how those actions affect the learner's return. CIAO follows the same single-learner setting and uses cooperative game theory to estimate the contribution of each agent to the joint value (Wang et al., 2024b).

These methods handle unknown agents or a changing active set from the viewpoint of one ad hoc learner. In our setting, every active agent is a learner that updates its own local policy, and no agent observes another agent's raw trajectory. A method that depends on modeling the types or actions of specific other agents therefore cannot be directly applied. What is still needed is a coordination mechanism through which independently learning agents can share and reuse knowledge without directly observing one another.

In summary, existing methods address different parts of the decentralized open setting, but they do not directly allow every active agent to learn from its local trajectory and exchange reusable coordination knowledge as $\mathcal{A}_t$ and individual operating behaviors change. This gap motivates BISKY, which separates local policy and value learning from behavior-based knowledge aggregation.

3 METHODOLOGY

We design **BISKY** based on the observation that agent identities may change, but similar operating behaviors can recur across agents and over time. Each agent therefore trains its own policy and value function using local experience and summarizes its recent operating behavior with a routing signature. BISKY uses this signature to connect the agent to a coordination channel representing the same behavior pattern. Agents connected to a channel contribute compact model updates, and the channel combines these updates into knowledge that can be returned for local use. Each agent then evaluates or applies the returned knowledge through its own local learning process. When the active agent set or an agent's behavior changes, BISKY updates the corresponding channel connection without changing the local learning structure. Figure 1 shows this process.

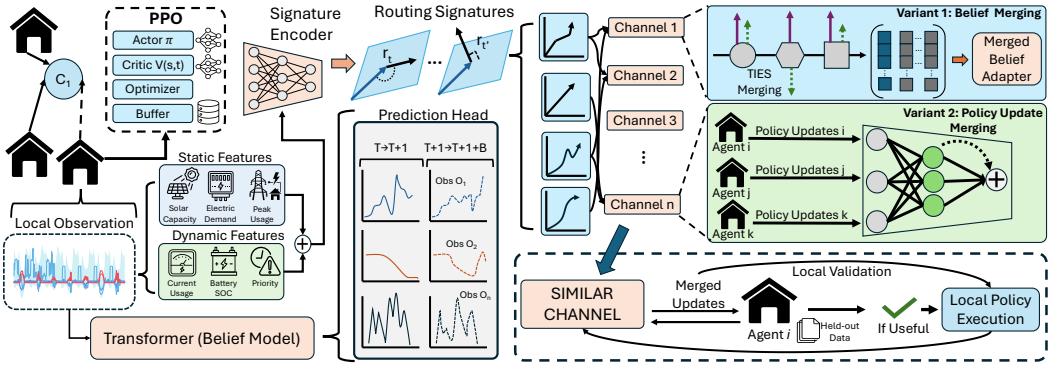


Figure 1: Overview of BISKY. Each agent trains a local policy, forms a routing signature from its local history, and connects to a coordination channel matching its behavior. Channels aggregate either belief adapter updates or policy updates.

Local Policy Learning: In BISKY, each agent trains an actor π_{θ_i} and value function V_{ψ_i} with PPO (Schulman et al., 2017), using its own observations, actions, rewards, optimizer, and rollout buffer. Training requires neither global observations nor a centralized critic. In the belief-sharing variant, the actor and value function receive

$$x_{i,t} = (o_{i,t}, b_{i,t}, c_{i,t}), \tag{1}$$

where $o_{i,t}$ is the local observation, $b_{i,t}$ is the local belief estimate, and $c_{i,t}$ is the context returned by the connected channel. In the policy update merging variant, we set $x_{i,t} = o_{i,t}$ and use the channel output only to update the local policy. In both variants, each agent selects actions using its policy.

Routing Signature: Local policy learning determines how each agent selects actions, but BISKY still needs to identify which agents may share useful knowledge. For this purpose, agent i summarizes its current operating behavior using a routing signature

$$s_{i,t} = [s^1_{i,t}, s^2_{i,t}, \ldots, s^{d_s}_{i,t}], \tag{2}$$

where d_s is the number of signature features. The signature combines long term characteristics, such as average energy use, generation capacity, and the balance between daytime and nighttime demand, with recent operating conditions, such as battery level, energy use, and pending energy requests. The selected features depend on the energy environment and are specified in the evaluation. Agent i computes all signature features from its own local information. BISKY uses $s_{i,t}$ only to determine the agent's channel connection. The signature does not enter the actor or value function.

Dynamic Bipartite Graph and Channel Assignment: BISKY uses the routing signatures to connect agents with related operating behaviors. These connections form a dynamic bipartite graph

$$G_t = (\mathcal{A}_t \cup \mathcal{C}_t, \mathcal{E} * t), \tag{3}$$

where the agent nodes $\mathcal{A} * t$ form one partition, the coordination channels $\mathcal{C} * t$ form the other, and $\mathcal{E} * t$ is the set of connections between them. The graph does not connect agents directly. Each channel c maintains a reference profile $p * c, t$ that summarizes the operating behavior represented by its connected agents. BISKY assigns agent i to the channel whose profile has the highest cosine similarity to its routing signature:

$$c * i, t = \arg\max * c \in \mathcal{C} * t \cos\left(s * i, t, p * c, t\right). \tag{4}$$

If the highest similarity is below a fixed threshold, BISKY creates a new channel from the agent's routing signature. To avoid frequent changes between channels with similar profiles, an agent changes its connection only when another channel exceeds its current match by a fixed margin. A channel with no connected agents is removed. The exact threshold and margin are reported in the implementation details. Once an agent is assigned, its channel determines which compact updates it contributes and which shared knowledge it receives. As the active set or agent behavior changes, BISKY updates the affected channel connections without changing the local actor or value function.

Local Belief Model: Channel assignment identifies agents with related operating behaviors, but the agents still need a compact form of knowledge that can be combined without sharing their trajectories. In the belief sharing variant, BISKY uses a local self supervised predictor for this purpose. Agent i forms a local observation and action vector

$$u_{i,t} = (o_{i,t}, a_{i,t}) \tag{5}$$

and stores a history of length L:

$$H_{i,t} = (u_{i,t-L+1}, \ldots, u_{i,t}). \tag{6}$$

The local belief model f_i predicts selected components of the agent's future trajectory:

$$P_{i,t} = f_i\left(H_{i,t}\right). \tag{7}$$

Depending on the environment, $P_{i,t}$ may include future residual demand, battery state of charge, or energy requests. We implement f_i using a Transformer encoder (Vaswani et al., 2017) with a small residual adapter (Houlsby et al., 2019). The adapter learns a correction to the local forecast from recent prediction errors. Only the adapter update is sent to the connected channel. The Transformer, observations, actions, and local history remain with the agent.

Each agent evaluates its belief model using complete episodes stored in a separate validation buffer. Using complete episodes prevents overlapping windows from appearing in both the training and validation data. Agent i measures its validation error as

$$\mathcal{L} * i = \frac{1}{N_i^{\mathrm{val}}} \sum_t \left| f_i\left(H * i, t\right) - y_{i,t} \right| * 2^2, \tag{8}$$

where $y * i, t$ is the realized future trajectory and N_i^{val} is the number of validation windows. BISKY later uses $\mathcal{L}_i$ to weight the agent's contribution and evaluate the update returned by the channel.

Merging Belief Adapters: Agents connected to the same channel have related operating behaviors, so an adapter learned by one agent may improve the forecasts of other agents in that channel. At each channel update interval, agent i sends its adapter update $\Delta\theta_i$ to the connected channel. Its Transformer, policy, value function, observations, actions, and local history are not shared. All adapters begin from the same fixed initialization, which makes their parameter updates comparable across agents (Ilharco et al., 2022).

Directly averaging the adapter updates can remove useful information when different agents modify the same parameter in opposite directions (Wortsman et al., 2022; Matena & Raffel, 2022). BISKY therefore applies TIES Merging (Yadav et al., 2023). TIES first retains the largest k% of entries in each update. For each parameter, it identifies the dominant update direction and combines the updates that agree with that direction. This process reduces conflicts between local adapter updates and produces a merged update $\Delta\theta_{\mathrm{merge}}$.

BISKY also weights each contribution according to the amount of local experience and the accuracy of the local belief model:

$$w_i = \frac{n_i}{1 + \mathcal{L}_i}, \tag{9}$$

where n_i is the number of local training samples and $\mathcal{L}_i$ is the validation error of agent i (McMahan et al., 2017). The channel scales each adapter update by w_i before applying TIES Merging. Agents with more local samples and lower validation error contribute more to the merged update.

A merged update may not improve every agent, even when the connected agents have similar routing signatures. Each agent evaluates $\Delta\theta_{\text{merge}}$ using its own held out validation episodes. The agent accepts the update only when it does not increase its forecast error. If accepted, agent i updates its local adapter as

$$\theta_i^{\text{new}} = \theta_i^{\text{local}} + \alpha_i \Delta\theta_{\text{merge}}, \tag{10}$$

where $\alpha_i \in [0, 1]$ is the acceptance weight. BISKY sets $\alpha_i = 0$ when the merged update increases the agent's validation error, leaving the local adapter unchanged (Li et al., 2021). This local check allows each agent to use shared belief knowledge only when it improves its own prediction.

Policy Gradient Merging: The belief sharing variant exchanges forecasting knowledge. The policy sharing variant instead combines updates learned by local policies to support faster adaptation when the active set or agent behavior changes. Each agent computes its PPO gradient using its own observations, actions, rewards, and rollout data. Only the resulting policy update is sent to the connected channel.

Each channel c maintains a channel policy with parameters θ_c. If n_c agents are connected to channel c, the channel combines their local PPO gradients as

$$\Delta\theta_c = \text{Merge}\left(g_1, g_2, \ldots, g_{n_c}\right), \tag{11}$$

where g_i is the local policy gradient from agent i. The channel then updates its policy parameters:

$$\theta_c^{\text{new}} = \theta_c^{\text{old}} + \eta \Delta\theta_c, \tag{12}$$

where η is the policy learning rate. Raw observations, actions, rewards, and rollout data remain with the agents.

The channel policy represents policy knowledge for one operating behavior rather than a shared actor for the full population. A newly active agent can initialize its local policy from the channel matching its routing signature and continue learning from its own experience. If an active agent changes its operating behavior, it can connect to a different channel and use the policy knowledge associated with that behavior. The channel policy does not select actions. Each agent continues to act using its own local policy.

Channel Knowledge Integration: In the belief sharing variant, the channel returns the merged adapter update and a compact channel summary to each connected agent. The agent first evaluates the merged adapter using its local validation data, as described above. If the update is accepted, the agent applies it to its local belief model and produces a channel informed forecast $b_{i,t}^{\text{ch}}$ from its own history. The agent also keeps its original local forecast $b_{i,t}^{\text{loc}}$.

A reliability weight $\rho_i \in [0, 1]$ determines how much the channel forecast contributes to the final belief $b_{i,t}$. When previous channel updates have not improved local prediction, the agent relies mainly on $b_{i,t}^{\text{loc}}$. When the channel has been useful, $b_{i,t}^{\text{ch}}$ has greater influence. The final belief $b_{i,t}$ and the compact channel context $c_{i,t}$ are then included in the local policy input $x_{i,t}$. The channel forecast does not replace the agent's local model or select its actions.

All channel information is delayed by one time step. At time t, an agent uses only channel information produced before time t. This prevents an agent from accessing the current observations or actions of other agents. In the policy sharing variant, channel knowledge is used only to initialize or update the local policy and does not enter the policy input.

Online Training Procedure: BISKY trains all active agents online. At each time step, agent i receives its local observation and forms the policy input $x_{i,t}$. In the belief sharing variant, this input includes the local observation, belief estimate, and channel context from the previous time step. In the policy sharing variant, the policy input includes only the local observation. The agent selects an action using its local policy and stores the resulting transition in its own rollout buffer. The same local information is added to its belief and routing histories.

At fixed routing intervals, each agent updates its routing signature and BISKY reevaluates its channel connection. In the belief sharing variant, each agent also updates its local belief model and sends its

adapter update to the connected channel. The channel merges the received updates, and each agent evaluates the returned adapter using its local validation data before incorporating it. In the policy sharing variant, each agent computes a PPO gradient from its local rollout data. The connected channel merges these gradients and updates the corresponding channel policy.

When $\mathcal{A}_t$ changes, BISKY updates the graph without changing the learning structure of the remaining agents. A newly active agent receives its own actor, value function, optimizer, and rollout buffer and connects to a channel using its routing signature. When an agent becomes inactive, BISKY removes its graph connections and unfinished rollout data. The remaining agents continue training with their existing local models. No centralized critic, joint value function, or global observation is introduced at any stage.

4 Experimental Evaluation

Evaluation Questions. BISKY keeps raw trajectories and policy learning local by design. Our experiments therefore focus on coordination and openness. We consider a closed setting with a fixed active agent set and an open setting in which $\mathcal{A}_t$ changes at predefined training intervals. We address three questions. (1) Does BISKY maintain coordination as the active agent set changes? (2) How does BISKY compare with centralized, independent, graph based, and open ad hoc methods? (3) How much do dynamic routing, channel knowledge, local validation, and the two merging variants contribute to performance?

Environments. We study two energy settings with shared physical or market constraints. The first is a residential peer-to-peer energy trading market that combines the household control problem from SolarSys (Gupta et al., 2026), a double-sided auction market (Qiu et al., 2021a), and the power-flow model from PowerGridWorld (Biagioni et al., 2022). The second is an electric-vehicle charging network built on SustainGym (Yeh et al., 2023), where each charging station is controlled by a separate agent. In the closed setting, the active agent set remains fixed throughout training. In the open setting, agents become active according to a predefined schedule. Environment definitions, datasets, observation and action spaces, reward functions, population schedules, and training settings are provided in the supplementary material.

Baselines. We compare BISKY with methods representing different coordination structures. IPPO (De Witt et al., 2020) trains a separate actor and value function for each agent. MAPPO (Yu et al., 2022) uses local actors and a centralized value function and serves as the benchmark in the closed setting. InforMARL (Nayak et al., 2023) exchanges information through a graph defined by the agents and their local neighborhoods. POAM (Wang et al., 2024a) and GPL (Rahman et al., 2021) represent open ad hoc teamwork methods that model other agents from observable information. All applicable methods use the same observations, reward functions, and active agent schedules.

Metrics. For peer-to-peer energy trading, we report cumulative reward, household cost savings, and reduction in grid energy use. Cost savings measure the economic benefit to individual households. Grid reduction measures system-level coordination under shared supply and demand. For electric-vehicle charging, we report cumulative reward. Calculation details are provided in the supplementary material.

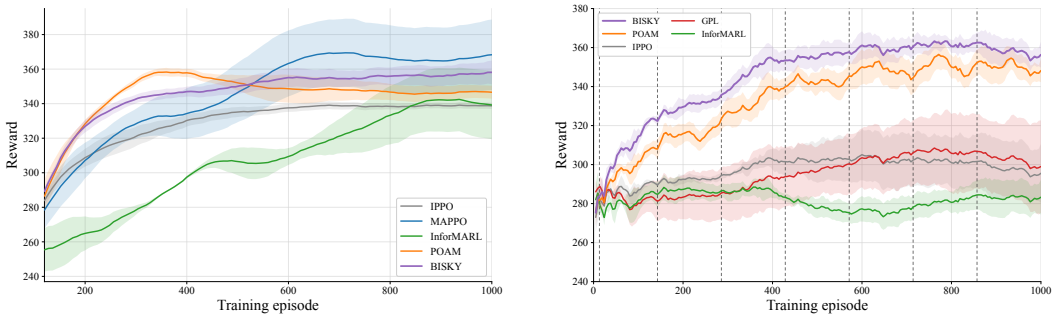
4.1 Overall Performance Comparison

Table 1 reports performance in the closed and open settings. MAPPO is evaluated only in the closed setting because the centralized value function in the evaluated implementation uses a joint input defined for a fixed agent population.

Why Reward Alone Is Insufficient. The peer-to-peer trading environment uses a composite reward (Gupta et al., 2026). One component penalizes each household's energy cost. The remaining components encourage grid reduction, successful peer-to-peer exchange, battery use, and fair trading. An agent can reduce part of its own cost through local battery scheduling and price aware grid use. In contrast, grid reduction and peer-to-peer exchange require agents to coordinate their actions under shared supply and demand. A method can therefore obtain a relatively high reward by improving individual costs without achieving the same improvement in system-level coordination. We report cost savings and grid reduction together with reward to separate these effects.

Table 1: Performance comparison in peer-to-peer energy trading and electric-vehicle charging. Values report the mean and standard deviation. Best results in each setting are shown in bold.

Method	Peer-to-Peer Energy Trading						EV Charging	
	Cost Savings (%)		Grid Reduction (%)		Rewards		Rewards	
	Closed	Open	Closed	Open	Closed	Open	Closed	Open
MAPPO	$\mathbf{35.0 \pm 5.8}$	NA	$\mathbf{26.3 \pm 4.6}$	NA	$\mathbf{368.3 \pm 45.2}$	NA	$\mathbf{3.3 \pm 0.8}$	NA
IPPO	18.4 ± 1.8	15.6 ± 5.4	14.1 ± 0.4	11.9 ± 3.9	338.7 ± 14.2	295.4 ± 24.1	1.8 ± 0.4	1.6 ± 0.7
InforMARL	22.1 ± 4.2	14.2 ± 7.6	16.6 ± 3.9	10.5 ± 6.9	339.2 ± 41.8	283.2 ± 37.5	2.1 ± 0.7	2.3 ± 0.9
POAM	30.7 ± 1.6	28.4 ± 3.1	22.4 ± 0.9	21.5 ± 2.7	346.6 ± 9.4	348.1 ± 31.3	2.8 ± 0.2	2.7 ± 0.6
GPL	13.2 ± 0.5	17.4 ± 5.9	9.5 ± 0.3	13.2 ± 4.6	100.8 ± 25.6	299.0 ± 20.7	2.3 ± 0.5	1.9 ± 0.8
BISKY	33.2 ± 2.6	$\mathbf{29.6 \pm 4.2}$	23.2 ± 0.5	$\mathbf{23.1 \pm 3.8}$	358.1 ± 11.9	$\mathbf{356.2 \pm 12.4}$	2.9 ± 0.3	$\mathbf{3.1 \pm 0.4}$



(a) Closed setting.

(b) Open setting.

Figure 2: Training reward across episodes in the closed and open settings. Lines show the mean across runs, and shaded regions show one standard deviation. Vertical dashed lines in (b) mark changes in the active agent set.

Closed Setting. As shown in Table 1, MAPPO obtains the highest closed setting performance, reaching $35.0 \pm 5.8\%$ cost savings, $26.3 \pm 4.6\%$ grid reduction, and a reward of 368.3 ± 45.2 in peer-to-peer trading. BISKY reaches $33.2 \pm 2.6\%$ cost savings, $23.2 \pm 0.5\%$ grid reduction, and a reward of 358.1 ± 11.9. It also obtains a reward of 2.9 ± 0.3 in electric-vehicle charging, compared with 3.3 ± 0.8 for MAPPO. BISKY approaches the centralized benchmark without global observations or a centralized value function.

The results also show why reward alone does not fully describe coordination. IPPO and InforMARL obtain peer-to-peer trading rewards of 338.7 ± 14.2 and 339.2 ± 41.8, respectively, but their grid reductions are only $14.1 \pm 0.4\%$ and $16.6 \pm 3.9\%$. BISKY reaches a similar reward range but produces substantially greater grid reduction. This difference shows that BISKY improves system-level coordination in addition to the locally optimizable cost component.

Open Setting. As shown in Table 1, BISKY obtains the best result in every open setting column. In peer-to-peer trading, it reaches $29.6 \pm 4.2\%$ cost savings, $23.1 \pm 3.8\%$ grid reduction, and a reward of 356.2 ± 12.4. It also reaches a reward of 3.1 ± 0.4 in electric-vehicle charging. Compared with POAM, the strongest open decentralized baseline in peer-to-peer trading, BISKY improves cost savings by 1.2 percentage points and grid reduction by 1.6 percentage points.

BISKY's grid reduction changes only from 23.2% in the closed setting to 23.1% in the open setting, and its reward changes from 358.1 to 356.2. In comparison, IPPO and InforMARL show larger reductions in both metrics. These results show that BISKY continues coordinating the active agents as $\mathcal{A}_t$ changes.

4.2 Learning Dynamics

Figure 2a shows the moving average training reward in the closed setting. BISKY improves quickly and reaches a stable reward with less variation than MAPPO and InforMARL. MAPPO reaches the highest reward but requires more training to stabilize and shows greater variation across runs. InforMARL improves more slowly and continues to show substantial variation near the end of training.

Table 2: Ablation results for the belief sharing and policy sharing variants in the closed and open settings. Values are normalized returns reported as the mean and standard deviation over five seeds.

System Setting	Independent PPO	IPPO with Local Belief	Random Routing	Static Routing	No Retain Knowledge	No Validation	Full BISKY
Belief, Closed	60.5 ± 3.2	71.4 ± 3.8	88.1 ± 6.7	89.2 ± 2.4	89.7 ± 2.1	85.2 ± 3.6	91.2 ± 3.8
Belief, Open	52.9 ± 8.7	68.3 ± 7.4	70.4 ± 6.1	73.1 ± 5.8	68.5 ± 6.9	60.1 ± 8.2	89.5 ± 5.6
Policy, Closed	60.5 ± 3.2	NA	90.4 ± 6.1	91.5 ± 4.3	91.8 ± 3.6	NA	93.1 ± 2.9
Policy, Open	52.9 ± 8.7	NA	75.5 ± 6.3	78.3 ± 7.1	72.4 ± 8.4	NA	92.3 ± 6.7

POAM learns quickly during the early episodes but settles below BISKY. These results show that BISKY can learn coordinated policies consistently without relying on a centralized value function.

Figure 2b shows the moving average reward in the open setting. The vertical dashed lines mark changes in the active agent set. Each change modifies the coordination problem and introduces agents whose policies may be at different stages of learning. BISKY recovers after these changes, continues improving, and maintains the highest reward. POAM remains the closest baseline, and IPPO, GPL, and InforMARL reach lower final rewards. The narrower variation around BISKY shows more consistent learning across runs as the active population changes.

4.3 Ablation Study

Table 2 presents ablation results for both the belief sharing and policy sharing variants. We report normalized returns from 0 to 100, where higher values represent better performance. Entries give the mean and standard deviation over $N = 5$ seeds.

Closed Setting. As shown in Table 2, removing one component causes a relatively small reduction when the active agent set remains fixed. For the belief sharing variant, full BISKY reaches 91.2 ± 3.8, compared with 88.1 ± 6.7 under random routing, 89.2 ± 2.4 under static routing, and 85.2 ± 3.6 without local validation. The policy sharing variant shows a similar pattern, with full BISKY reaching 93.1 ± 2.9. These results suggest that a fixed population reduces the effect of individual components because the coordination structure does not need to respond to changes in $\mathcal{A}_t$.

Open Setting. As shown in Table 2, the differences become larger when the active agent set changes. In the belief sharing variant, full BISKY reaches 89.5 ± 5.6, compared with 70.4 ± 6.1 under random routing and 73.1 ± 5.8 under static routing. This result shows that channel connections based on current operating behavior become more important when the population changes. Removing local validation reduces performance to 60.1 ± 8.2, below the 68.3 ± 7.4 obtained with only the local belief model. Local validation therefore prevents unsuitable merged updates from reducing prediction and control performance.

The policy sharing variant reaches 92.3 ± 6.7 in the open setting. Random and static routing reduce this value to 75.5 ± 6.3 and 78.3 ± 7.1, respectively. Full BISKY also remains close to its closed setting performance in both variants. The belief sharing variant changes from 91.2 to 89.5, and the policy sharing variant changes from 93.1 to 92.3. Together, these results show that behavior based routing and channel knowledge are most useful when $\mathcal{A}_t$ changes.

5 Conclusion

We introduced **BISKY**, a decentralized coordination framework for open multi-agent energy systems. Its dynamic bipartite graph connects active agents to channels representing recurring operating behaviors. Each agent trains its own actor and value function, and the channels combine compact belief-adapter or policy updates without accessing raw agent trajectories. A newly active agent can connect to a channel containing knowledge from agents with similar operating patterns. If an active agent's behavior changes, it can change its channel connection without resetting its local learner. Action selection remains local, without global observations or a centralized critic. Experiments in peer-to-peer energy trading and electric-vehicle charging show that BISKY remains effective as the active agent set changes. The ablation results indicate that dynamic routing and retained channel knowledge contribute to its performance under changing participation.

AI USE STATEMENT

In this work, we used generative AI tools primarily to suggest revisions to the wording of the manuscript and improve its clarity and readability. We reviewed and edited all AI-assisted suggestions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have released the source code and data used in this study at `https://huggingface.co/Bisky2026/BISKY`. In this paper, Section 3 describes the BISKY methodology, and Section 4 presents the experimental settings, baselines, metrics, and results.

REFERENCES

David Biagioni, Xiangyu Zhang, Dylan Wald, Deepthi Vaidhynathan, Rohit Chintala, Jennifer King, and Ahmed S Zamzam. Powergridworld: A framework for multi-agent reinforcement learning in power systems. In *Proceedings of the thirteenth ACM international conference on future energy systems*, pp. 565–570, 2022.

Muthukumaran Chandrasekaran, Adam Eck, Prashant Doshi, and Leenkiat Soh. Individual planning in open and typed agent systems. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 82–91, 2016.

Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3414–3421, 2020.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Adam Eck, Maulik Shah, Prashant Doshi, and Leen-Kiat Soh. Scalable decision-theoretic planning in open and typed multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7127–7134, 2020.

Yuzhou Feng, Qi Li, Dong Chen, and Raju Rangaswami. Solartrader: Enabling distributed solar energy trading in residential virtual power plants. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp. 60–69, 2020.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. *Advances in neural information processing systems*, 14, 2001.

Anany Pratap Gupta, Xiangyu Li, Keyang Yu, and Qi Li. Solarsys: Scalable peer-to-peer energy trading for residential virtual power plants. In *Proceedings of the 13th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp. 429–439, 2026.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2022.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pp. 2961–2970. PMLR, 2019.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International conference on machine learning*, pp. 6357–6368. PMLR, 2021.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7211–7218, 2020.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Michael S Matena and Colin Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International conference on machine learning*, pp. 25817–25833. PMLR, 2023.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Dawei Qiu, Jianhong Wang, Junkai Wang, and Goran Strbac. Multi-agent reinforcement learning for automated peer-to-peer energy trading in double-side auction market. In *IJCAI*, pp. 2913–2920, 2021a.

Jian Qiu, Yujie Tang, and Jianhui Wang. A scalable multi-agent deep reinforcement learning for peer-to-peer energy trading. *IEEE Transactions on Smart Grid*, 12(4):3079–3090, 2021b. doi: 10.1109/TSG.2020.3047122.

Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *International conference on machine learning*, pp. 8776–8786. PMLR, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Elizabeth L Ratnam, Steven R Weller, Christopher M Kellett, and Alan T Murray. Residential load and rooftop pv generation: an australian distribution network dataset. *International Journal of Sustainable Energy*, 36(8):787–806, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Christopher Alan Smith. The pecan street project: developing the electric utility system of the future. 2009.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pp. 1504–1509, 2010.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jose R Vazquez-Canteli, Sourav Dey, Gregor Henze, and Zoltan Nagy. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020a.

Jose R Vazquez-Canteli, Gregor Henze, and Zoltan Nagy. Marlisa: Multi-agent reinforcement learning with iterative sequential action selection for load shaping of grid-interactive connected buildings. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pp. 170–179, 2020b.

Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in neural information processing systems*, 35:3962–3974, 2022.

Caroline Wang, Muhammad Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. N-agent ad hoc teamwork. *Advances in Neural Information Processing Systems*, 37:111832–111862, 2024a.

Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. *Advances in neural information processing systems*, 34:3271–3284, 2021.

Jianhong Wang, Yang Li, Yuan Zhang, Wei Pan, and Samuel Kaski. Open ad hoc teamwork with cooperative game theory. *arXiv preprint arXiv:2402.15259*, 2024b.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020a.

Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7293–7300, 2020b.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022.

Dong Xing, Pengjie Gu, Qian Zheng, Xinrun Wang, Shanqi Liu, Longtao Zheng, Bo An, and Gang Pan. Controlling type confounding in ad hoc teamwork with instance-wise teammate feedback rectification. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38272–38285. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/xing23a.html`.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Christopher Yeh, Victor Li, Rajeev Datta, Julio Arroyo, Nicolas Christianson, Chi Zhang, Yize Chen, Mohammad Mehdi Hosseini, Azarang Golmohammadi, Yuanyuan Shi, et al. Sustaingym: Reinforcement learning environments for sustainable energy systems. *Advances in Neural Information Processing Systems*, 36:59464–59476, 2023.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *International conference on machine learning*, pp. 5872–5881. PMLR, 2018.

Weijia Zhang, Hao Liu, Fan Wang, Tong Xu, Haoran Xin, Dejing Dou, and Hui Xiong. Intelligent electric vehicle charging recommendation based on multi-agent reinforcement learning. In *Proceedings of the Web Conference 2021*, pp. 1856–1867, 2021.

Weijia Zhang, Hao Liu, Jindong Han, Yong Ge, and Hui Xiong. Multi-agent graph convolutional reinforcement learning for dynamic electric vehicle charging pricing. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2471–2481, 2022.

APPENDIX

A ENVIRONMENT DETAILS

We evaluate BISKY in two energy environments: residential peer-to-peer (P2P) energy trading and electric-vehicle (EV) charging. We use both environments because they expose different forms of multi-agent dependence. In P2P trading, agents interact through both an economic market and a shared electrical network. In EV charging, agents do not trade with one another, but their charging decisions compete for the same network capacity. Together, these environments test whether BISKY can coordinate changing populations when coupling arises through prices, physical constraints, or both.

A.1 PEER-TO-PEER ENERGY TRADING

The first environment represents a residential community containing consumers and prosumers with rooftop solar and battery storage. A household with surplus solar can sell energy locally instead of exporting it entirely to the utility, while a household with an energy deficit can purchase energy from its peers instead of relying only on the grid. P2P trading can therefore improve the local use of renewable generation and reduce household costs. However, these benefits do not arise from independent household optimization alone: a trade requires a compatible buyer and seller, and the resulting energy schedule must remain feasible on the shared distribution network.

To preserve these dependencies, our environment combines three components: the residential control problem from SolarSys Gupta et al. (2026), the double-sided auction mechanism for the market layer Qiu et al. (2021b), and the physical power-system simulation provided by PowerGridworld Biagioni et al. (2022). Each component addresses a different part of the coordination problem.

P2P energy-management mechanism: SolarSys provides the household-level energy-management problem, including electricity demand, rooftop-solar generation, battery storage, utility-grid interaction, and local energy exchange. Each household is controlled by an independent agent that determines how to allocate its available energy among local consumption, battery storage, P2P trading, and grid exchange. The environment uses the SolarSys household objective, which accounts for energy cost, grid dependence, successful P2P exchange, battery utilization, and trading outcomes. This layer therefore defines what each household wants to achieve.

Each household receives a per-step reward that combines its realized energy cost with incentives for reducing grid dependence, completing P2P trades, using its battery, and distributing trading opportunities fairly:

$$\begin{aligned} r_{i,t} = & -w_7 C_{i,t} - w_1 G_{i,t} \bar{p}_t^{\,g} + w_2 S_{i,t}^{\mathrm{p2p}} \bar{p}_t^{\,p} \\ & + w_3 \left(B_{i,t}^{\mathrm{ch}} + B_{i,t}^{\mathrm{dis}} \right) - w_4 E_{i,t} - w_5 D_{i,t} + w_6 J_t. \end{aligned} \tag{13}$$

Here, $C_{i,t}$ is the household's net cost after grid and P2P transactions, $G_{i,t}$ is grid import, $S_{i,t}^{\mathrm{p2p}}$ is the executed P2P sale, $B_{i,t}^{\mathrm{ch}}$ and $B_{i,t}^{\mathrm{dis}}$ are the battery charge and discharge quantities, $E_{i,t}$ is grid export, $D_{i,t}$ is the residual deficit that must be supplied through forced grid import, and J_t is the fairness term over trading opportunities. The normalized grid and peer prices are $\bar{p}_t^{\,g} = p_t^g / p_{\max}^g$ and $\bar{p}_t^{\,p} = p_t^p / p_{\max}^g$, respectively.

Demand and solar-generation data are obtained from real smart-meter data. The primary dataset contains half-hourly demand and rooftop-solar measurements from 300 Ausgrid households Ratnam et al. (2017) in Sydney, Australia, covering July 2010 through June 2013. We also use Pecan Street data Smith (2009), which includes households with rooftop solar as well as households without it. Each episode represents one day and contains $T = 48$ half-hour control intervals. Sampling different days exposes the agents to natural variations in demand, weather, and solar generation.

Market layer: The market layer follows the double-sided auction formulation in Qiu et al. (2021b). Each order specifies an energy quantity and a price. Buy orders are ranked by decreasing bid price and sell orders by increasing ask price, and compatible orders are matched while the buyer's bid is no lower than the seller's ask. The clearing outcome determines the executed P2P quantities and settlement prices; unmatched demand or surplus is subsequently handled through grid exchange, storage, or curtailment.

This layer is essential because an agent's proposed trade is not automatically executed. Its outcome depends on the contemporaneous quantities and prices selected by the other active agents. Consequently, changes in the active population also change the available supply, demand, competition, and market-clearing outcome, even when an incumbent household's own demand and generation remain unchanged.

Physical layer: The cleared market schedule is passed to PowerGridworld, which evaluates the corresponding injections and withdrawals on the distribution network Biagioni et al. (2022). PowerGridworld computes power-flow quantities such as feeder import, bus voltages, line loading, transformer loading, and network losses, and identifies violations of the operating limits. Importantly, a financially balanced P2P transaction does not remove the associated physical power flow; energy purchased by one household must still travel through the shared electrical network.

The physical layer therefore prevents the market from treating every economically valid schedule as physically feasible. Two sets of bids may produce similar trading costs while imposing very different loading on the feeder. Coordination must consequently account for both market compatibility and the physical consequences of the cleared transactions. This three-part construction allows us to evaluate the complete path from a household's local decision, to market execution, to its realized effect on the power network.

Observation: Each household receives an observation containing its demand, solar generation, battery state of charge, grid and peer prices, time of day, device availability, and the aggregate supply and demand. The observation does not contain the other agents' policies, private learning states, or future actions.

Action: Each household selects a six-dimensional continuous action specifying energy bought from and sold to the utility grid, energy bought from and sold to peers, and battery charging and discharging. The market and physical layers resolve these requests with available energy, auction outcomes, device limits, and network constraints.

A.2 SustainGym EV Charging

Our second environment adapts SustainGym's `EVChargingEnv` into an open, per-station multi-agent control problem Yeh et al. (2023). SustainGym models real EV charging networks, including charging session demand, charging device restrictions, nonlinear battery behavior, unbalanced three-phase power flow, and shared electrical constraints. We assign one learning agent to each selected charging station while retaining the underlying SustainGym network dynamics and objective.

This environment complements P2P trading because the agents interact through a different coordination mechanism. The stations do not trade energy with one another, but their charging currents pass through common transformers and electrical lines. Therefore, the action selected by one station changes how much capacity remains available to the other stations.

Charging sessions: The main configuration uses the Caltech charging network and the Summer 2019 session model. A sampled session specifies its arrival time, estimated and actual departure times, requested energy, and charging-station identity. Time-varying marginal operating emissions rate (MOER) signals represent the carbon intensity of the electricity supply.

We use a fixed pool of 24 charging stations, of which between 10 and 16 are active during an open-system rollout. Each episode represents one day and contains $T = 288$ five-minute intervals. The physical charging network remains fixed when the active population changes. Only the stations controlled by active agents are allowed to submit nonzero charging requests.

Let $\mathcal{A}_t$ denote the active agent set. The action submitted to the underlying charging network is masked according to

$$\widetilde{a}_{i,t} = \begin{cases} a_{i,t}, & i \in \mathcal{A}_t, \\ 0, & i \notin \mathcal{A}_t. \end{cases} \tag{14}$$

Thus, a station leaving the open system immediately stops requesting current, but its physical location and network connections are not removed.

Observation: Each active station receives an 11-dimensional observation,

$$\mathbf{o}_{i,t} = \begin{bmatrix} \tau_t,\ \widehat{d}_{i,t},\ e_{i,t},\ z_{i,t}, \\ m_t,\ \overline{m}_t^{6},\ \overline{m}_t^{36},\ h_i, \\ \rho_{-i,t},\ \overline{e}_{-i,t},\ \ell_{i,t-1} \end{bmatrix}^{\top}. \tag{15}$$

Here, τ_t is the normalized time of day, $\widehat{d}_{i,t}$ is the estimated time remaining until departure, $e_{i,t}$ is the vehicle's remaining energy demand, and $z_{i,t}$ indicates whether a vehicle is currently connected. The terms m_t, $\overline{m}_t^{6}$, and $\overline{m}_t^{36}$ are the current MOER and its average forecast over the next 30 minutes and three hours. The fixed indicator h_i identifies the charging-device type. Finally, $\rho_{-i,t}$ and $\overline{e}_{-i,t}$ provide coarse summaries of the occupancy and remaining demand of the other active stations, while $\ell_{i,t-1}$ reports the previous period's congestion affecting station i.

The observation is intentionally compact. A station does not receive the actions, complete session information, or future charging decisions of every other station. It must therefore infer whether shared capacity is likely to be available from local information and limited network feedback.

Action and charging-device restrictions. Station i selects one continuous action,

$$a_{i,t} \in [0, 1], \qquad p_{i,t}^{\mathrm{raw}} = 32\,\widetilde{a}_{i,t}\ \mathrm{A}, \tag{16}$$

representing a requested pilot signal between zero and 32 amperes. The request is then mapped to a charging current supported by the corresponding EVSE:

$$p_{i,t} = Q_i\big(p_{i,t}^{\mathrm{raw}}\big)\,, \tag{17}$$

where $Q_i(\cdot)$ rounds the request to the nearest feasible pilot. The AeroVironment stations support

$$\mathcal{P}^{\mathrm{AV}} = \{0\} \cup \{6, 7, \ldots, 32\}\ \mathrm{A}, \tag{18}$$

whereas the ClipperCreek stations support

$$\mathcal{P}^{\mathrm{CC}} = \{0, 8, 16, 24, 32\}\ \mathrm{A}. \tag{19}$$

A session terminates when its demand is satisfied or when its actual departure time is reached. Consequently, charging too slowly may leave part of the vehicle's requested energy undelivered.

The SustainGym reward contains three components: charging profit, carbon cost, and network-constraint violations.

$$R_{i,t} = P_{i,t} - C_{i,t}^{\mathrm{carbon}} - C_{i,t}^{\mathrm{viol}}. \tag{20}$$

The charging profit and carbon cost are

$$P_{i,t} = \phi_p c_{i,t}, \tag{21}$$

$$C_{i,t}^{\mathrm{carbon}} = \phi_c c_{i,t} m_t, \tag{22}$$

where $c_{i,t}$ is the realized charging current and m_t is the current MOER. The first term rewards delivering energy, while the second encourages charging during periods with cleaner electricity.

The violation cost $C_{i,t}^{\mathrm{viol}}$ in Eq. equation 20 distributes the network-constraint penalty across the stations that caused it. A station receives a larger penalty when it contributes more current to an

overloaded constraint. Since the contribution shares sum to one for each active constraint, the station rewards exactly recover the original SustainGym objective:

$$\sum_i R_{i,t} = R_t^{\mathrm{SustainGym}}. \tag{23}$$

Why coordination is necessary. Every station would prefer to charge its vehicle quickly, especially when departure is near. However, the transformer limit depends on the *sum* of the stations' actions, not on any one action by itself. A request of 32 amperes may be safe when the other stations are idle but unsafe when several neighboring stations also request 32 amperes. Therefore, the best action for one agent depends on what the other agents do.

If every station always requests its maximum current, the vehicles charge quickly but the network can become overloaded. If every station is too conservative, the network remains safe but vehicles may depart without receiving their requested energy. Good coordination lies between these two extremes: agents should use the available capacity aggressively when it is free and yield some capacity when other vehicles have more urgent charging requirements.

The main experiments disable SustainGym's optional centralized action projection. Consequently, an infeasible joint action is not silently corrected before execution; instead, its constraint excess appears in the violation term $C_{i,t}^{\mathrm{viol}}$ of Eq. equation 20. This makes network feasibility a behavior that the agents must learn together rather than a decision made by an external centralized optimizer.

A.3 Open-System Evaluation

A fixed-population evaluation would test coordination only among agents that remain present throughout training. It would not test BISKY's central claim: that behavior-associated knowledge remains useful when the identities and composition of the active population change. We therefore convert both environments into open agent systems in which households or charging stations may join, leave, and later return while learning is in progress.

For every random seed, the complete sequence of entry and exit events is generated before training and replayed identically for every evaluated method. Thus, all methods encounter the same agents and the same active population at each training checkpoint. Inactive P2P households contribute neither demand nor solar generation to the current episode, while inactive EV stations submit zero pilot current. The underlying household pool and physical charging network remain fixed, so an event changes participation without silently changing the environment itself.

When an agent leaves, its identifier-specific learning state, including its actor, critic, optimizer, and local rollout buffer, is removed. If the same identifier later returns, it is treated as a new participant and receives newly initialized private state. In contrast, BISKY's behavior channels persist independently of individual identities. A newcomer can therefore access knowledge associated with a relevant operating behavior without recovering the departed agent's private model.

This protocol separates behavior-level knowledge reuse from simple memorization of participant identities. It also creates two simultaneous adaptation requirements: newcomers must become useful with limited local experience, while incumbent agents must remain effective as market conditions and shared physical loads change around them. In the P2P experiment reported in the main text, three households join and two leave during training; the same paired-event principle is used for the EV-charging experiments.

B Simulation demo

Each panel is a bipartite graph between agents (circles) and the behaviour channels that BISKY maintains at that point in training (squares). As described in the methodology, each agent summarizes its recent behaviour into a signature and compares that signature, by cosine similarity, against every live channel prototype. We draw an edge from an agent to every channel for which this similarity is positive, and edge thickness reflects the magnitude of the similarity. A channel square grows with the number of agents it holds, and a dashed border marks a channel created within the last 200 episodes, so newly formed channels can be tracked across the four panels.

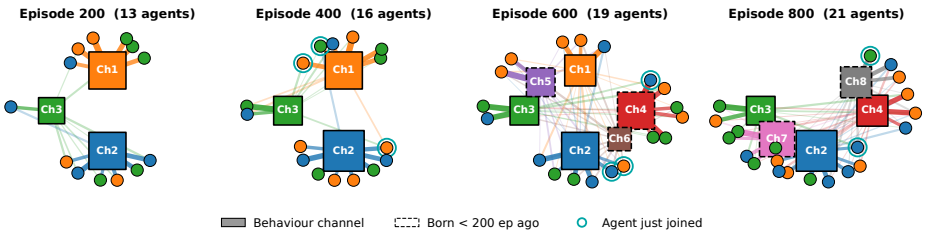


Figure 3: Visualization of BISKY's online behavioral routing on an open agent system

Circle colour denotes an agent's ground-truth demand type, while routing is driven entirely by the learned behavioural embedding. Square color is only a label and carries no meaning. Because routing ignores type, a channel often gathers agents of more than one type: two households of different types can produce very similar net-load and price-response behaviour and are therefore pulled into the same channel. This is why differently colored circles attach to a single square.

The four panels show BISKY under an open population. Training starts with 13 agents, and newcomers arrive in small groups, three before episode 400, three before episode 600, and two before episode 800, giving 21 agents at the end. Agents that joined most recently are marked with a ring. Early in training (episode 200) the population organizes into three stable channels. As new agents arrive and the range of behaviours widens, BISKY creates additional channels on demand, and six are live by episode 600 (Ch4, Ch5, and Ch6 are newly created). BISKY also closes channels that no longer attract members. By episode 800, Ch1, Ch5, and Ch6 have disappeared, while Ch7 and Ch8 have formed, leaving five live channels.. The channel set therefore tracks the current behaviour of the population rather than its membership history.

C Implementation Details

We run BISKY on a workstation with Ubuntu 22.04.5 LTS, an Intel Core Ultra 9 285K CPU, 64 GiB of DDR5 memory, and an NVIDIA GeForce RTX 5090 GPU with 32 GiB of VRAM using CUDA 12.9. Hyperparameters are selected through preliminary tuning runs based on training stability and validation performance, and are then fixed for all reported experiments. We conduct 10 independent runs using distinct random seeds and use the same seed set for every compared method. Each seed initializes the model parameters, environment sampling, and open-population event schedule. Results are reported as the mean and standard deviation across these runs. Table 3 summarizes the PPO, routing, and Transformer belief-module hyperparameters.

C.1 Baseline Implementation Details

This appendix describes how each baseline is implemented and adapted to the open-population setting. The purpose of these adaptations is to allow the baselines to operate when the number and identities of active agents change, without otherwise altering their central learning mechanisms.

C.2 Evaluation Protocol

Let $\mathcal{A}_u$ denote the set of active agents during training update u. The environment provides observations, actions, and rewards only for agents in $\mathcal{A}_u$. Joining and leaving events are applied between rollout updates, so the population remains fixed while a rollout is being collected.

All methods use the same environment observations, action ranges, reward functions, episode lengths, and population-event schedules. For a given random seed, the joining times, departure times, behavior changes, and environment days are generated once and reused across methods. Thus, differences between methods cannot be attributed to easier population schedules or different environment realizations.

The PPO-based methods use the same rollout length, discount factor, clipping parameter, optimization epochs, and actor architecture unless a baseline requires an additional encoder or centralized

Parameter	Value	Parameter	Value
PPO and Routing			
Actor hidden layers	$(64, 64)$	Critic hidden layers	$(64, 64)$
Initial actor learning rate	3×10^{-4}	Final actor learning rate	1×10^{-5}
Optimizer	Adam	Discount factor γ	0.99
PPO clipping parameter ϵ	0.20	Value-loss coefficient	0.50
Initial entropy coefficient	0.005	PPO epochs per update	4
Episodes per rollout	8	Episode length	48 steps
Routing interval	1 update	Signature history length	192 steps
Transformer Belief Module			
Encoder type	Transformer	Encoder layers	2
Input history window	16 steps	Input dimension	18 (12 obs. + 6 actions)
Model dimension d_{model}	64	Attention heads	4
Dimension per head	16	Feed-forward dimension	128
Activation	GELU	Prediction horizon	8 steps
Belief latent dimension	16	Belief learning rate	3×10^{-4}
Belief optimizer	AdamW	Training loss	Mean squared error
Shared parameters	Adapter only	Adapter merge method	TIES
TIES retained fraction	0.20	Initial acceptance weight α_i	0.50

Table 3: Hyperparameters used by BISKY in the open-population

critic. During evaluation, we use the mean action and disable exploration noise. Training and evaluation use different environment seeds.

A joining agent is assigned the learning state required by the corresponding method. A departing agent is removed from the active-agent data structures and does not contribute to updates. When an existing agent changes its behavioral profile without leaving, its identity and learned state are preserved. This tests whether the method can adapt its existing knowledge rather than restart learning.

C.3 Independent PPO

Independent PPO (IPPO; De Witt et al. 2020) treats each agent as a separate single-agent reinforcement-learning problem. Its action depends only on its local observation:

$$a_{i,t} \sim \pi_{\theta_i}(\cdot \mid o_{i,t}) \,. \tag{24}$$

We adapt IPPO to an open population using a set of learners indexed by agent identifier. When an agent joins, a new actor and critic are initialized from the same initialization distribution used at the beginning of training. When it leaves, its networks, optimizer state, and unfinished rollout are removed. If it later reappears as a new participant, it begins with a fresh learner. This makes IPPO naturally compatible with population changes.

IPPO is included as the decentralized reference. It satisfies locality and openness, but any coordination must emerge indirectly through the environment because agents do not exchange information.

C.4 Multi-Agent PPO

MAPPO Yu et al. (2022) follows centralized training with decentralized execution. All agents use a shared actor, while the critic receives information from the active population. A conventional MAPPO critic concatenates every agent's observation, which produces an input dimension that depends on the number of agents. Such a critic cannot directly process a changing population.

C.5 InforMARL

InforMARL Nayak et al. (2023) represents interactions among agents using a graph neural network. At time t, we construct an interaction graph

$$\mathcal{G}_t^{\text{info}} = \left(\mathcal{A}_t, \mathcal{E}_t^{\text{info}}\right) \tag{25}$$

over the currently active population. Each graph node contains an agent observation, and message passing produces an interaction representation $h_{i,t}$ for each agent. The policy is conditioned on the local observation and the graph representation:

$$a_{i,t} \sim \pi_\theta \left(\cdot \mid o_{i,t}, h_{i,t} \right). \tag{26}$$

For P2P energy trading, we construct a k-nearest-neighbor graph using similarity between the active agents' observation features. This allows agents with related current energy conditions to exchange information. For EV charging, edges connect stations that are coupled through a shared electrical constraint. The graph is reconstructed from the current active set, so joining and leaving agents can be incorporated without changing the GNN parameters.

InforMARL uses a shared actor and a centralized graph-pooled critic. A joining agent is inserted as a new graph node, while a departing agent and its incident edges are removed. Because graph aggregation is independent of the number of nodes, the network architecture remains fixed. Unlike BISKY, however, InforMARL requires access to graph-neighborhood information when constructing the policy representation. It is therefore used as a graph-coordination baseline rather than as a strictly local-execution method.

C.6 NAHT/POAM

We evaluate the POAM algorithm introduced in the NAHT setting Wang et al. (2024a); we refer to this baseline as **NAHT/POAM**. The method uses a shared policy together with a recurrent representation of teammate behavior. For each controlled agent, the recurrent state is updated from its observation and previous action:

$$z_{i,t} = \mathrm{GRU} \left(z_{i,t-1}, [o_{i,t}, a_{i,t-1}] \right). \tag{27}$$

The embedding $z_{i,t}$ is supplied to the actor and is also trained through auxiliary prediction objectives. These objectives encourage the embedding to contain information useful for predicting the observations or actions of currently present teammates.

The P2P implementation allocates prediction outputs for the maximum supported team size. Outputs corresponding to absent agents are masked and do not contribute to the auxiliary loss. Thus, the network shape remains fixed even when the population changes. In the EV implementation, where stations have a common action structure, the auxiliary decoder predicts a summary of the currently active teammates.

A joining agent begins with an empty recurrent state, and departed agents are excluded from the prediction mask. Recurrent states are also reset at episode boundaries. Teammate actions are used as training targets for the auxiliary model, but future actions are not available to the acting policy. This adaptation evaluates whether recurrent teammate modelling can handle changing membership when the maximum population size is known in advance.

C.7 Graph-Based Policy Learning

GPL Rahman et al. (2021) combines behavioral-type inference with a graph-based joint-value model. Its recurrent type model infers a representation for each active agent from its observation and previous action:

$$q_{i,t} = f_{\mathrm{type}} \left(o_{i,t}, a_{i,t-1}, q_{i,t-1} \right). \tag{28}$$

The inferred representations are processed by a graph model that estimates how the behavior of one agent affects the value of the others.

The original GPL formulation is designed primarily for discrete actions and uncertain teammate presence. Our environments instead have continuous actions and explicitly expose the current active-agent identifiers. We therefore make two necessary adaptations. First, the discrete action model is replaced by a Gaussian continuous-action model. Second, we use on-policy PPO updates rather than GPL's original discrete off-policy action learner. The recurrent behavioral-type model and graph joint-value structure same as GPL's original implementation.

A joining agent is added as a graph node with a zero-initialized recurrent type state. A departing agent is removed from the graph and no longer contributes to the joint-value estimate. This version

tests whether explicit teammate modelling and graph value factorization can coordinate a changing population when membership itself is observable.