POWERMARKETJAX: A JAX BENCHMARK SUITE FOR MULTI-AGENT REINFORCEMENT LEARNING IN POWER MARKETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Power markets are a natural testbed for multi-agent reinforcement learning (MARL), where multiple self-interested participants repeatedly submit bids. A market-clearing mechanism then determines dispatch and prices subject to power grid constraints and market settlement rules. However, existing MARL environments typically focus on a single market setting, implement simplified clearing mechanisms, or rely on CPU-based optimization solvers that slow large-scale training and limit the systematic study of bidding strategies and market behavior. We introduce PowerMarketJax, a benchmark suite for MARL across five power markets: day-ahead wholesale, real-time balancing, ancillary services, peer-to-peer double auctions, and local flexibility. Each environment implements its own clearing, pricing, and settlement rules while providing a common framework for learning and evaluation. We find that learned bidding behavior depends strongly on the market design: independent learners can miss better strategies when gains require many agents to change together, when more profitable strategies lie beyond a region of lower profit, or when profits disappear as more agents adopt the same strategy. PowerMarketJax implements both market simulation and policy training in JAX, allowing the entire pipeline to run on the GPU with $1\,024 \times 1\,200$ parallelisms across both environments and market participants, achieving up to $33\times$ speedup over CPU-based baselines. Our open-source benchmark is available at: `https://github.com/powermarketjax/PowerMarketJax`.

1 INTRODUCTION

Unlike most commodities, electricity cannot be stored efficiently at scale, so power markets are needed to continuously match supply and demand through mechanisms that determine which resources produce and at what price. Historically, these markets involved a relatively small set of large generators and mainly traded energy, making bidding strategies easier to characterize (David & Wen, 2000). This structure is now changing. Renewable generation, flexible demand, and energy storage have expanded both who participates and what is traded (Matamala & Strbac, 2025): alongside energy, markets now trade reserves, balancing, and ancillary services across multiple timescales and grid levels, each with its own clearing, pricing, and settlement rules (Zhu et al., 2025). This growing diversity of participants, products, and market mechanisms makes bidding behavior increasingly difficult to understand, creating an urgent need to study how participants bid and interact across different market settings. The repeated interactions among many market participants make power markets a natural setting for multi-agent reinforcement learning (MARL) (Liang et al., 2020).

In a MARL formulation, each market participant can be modeled as an agent that observes market information, submits bids, receives clearing outcomes, and updates its bidding strategy over time. Since a participant's profit depends on both its own bid actions and the actions of others through market clearing, bidding in power markets is inherently a multi-agent decision problem. MARL enables participants to learn such bidding strategies through repeated market interactions without requiring explicit knowledge of competitors' exact strategies (Du et al., 2021).

A number of environments have been developed for learning-based bidding in power markets, but important gaps remain for systematic MARL study. Most focus on a single market study, limit-

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.

ing their use as general benchmarks for MARL-based bidding (Yeh et al., 2023; Harder et al., 2025; El Helou et al., 2023; Salazar-Peña et al., 2026). A second gap is computational burden. An environment must solve a market-clearing optimization subject to grid physical constraints after every round of bids. Existing implementations typically rely on CPU solvers, making the millions of clearing problems required during MARL training a major computational bottleneck, especially with many agents and parallel environments (Wolgast & Nieße, 2024). To reduce this computational cost, some environments simplify pricing or settlement rules, but these simplifications can remove important market features that shape bidding incentives (Wolgast & Nieße, 2024). As a result, it remains difficult to systematically study how bidding strategies emerge and differ across power markets.

Here, we introduce PowerMarketJax, a power market MARL benchmark implemented entirely in JAX (Bradbury et al., 2018). It parallelizes both optimization-based market environments and multi-agent policy computation on the GPU. By compiling clearing and learning into a single computation graph, PowerMarketJax avoids repeated CPU solver calls and host-device synchronization at each market step, enabling large-scale exploration of bidding strategies across different markets. This design achieves up to 33$\times$ higher throughput than CPU-based implementations and runs 1 024 parallel 1 200-participant market environments at over 27 000 environment steps per second (Appendix D).

Our contributions are summarized as follows. First, we present a unified MARL benchmark covering five power markets: day-ahead wholesale, real-time balancing, ancillary services, peer-to-peer energy trading, and local flexibility. The benchmark spans different clearing mechanisms, timescales, and grid levels while preserving each market's clearing, pricing, and settlement rules under a common interface. Second, we implement both market-clearing optimization and MARL policy training entirely in JAX, enabling GPU-based parallel computation across environments and agents for large-scale training and evaluation. Third, using this benchmark, we empirically show that market design strongly shapes what independent learners can discover from their own profit signals: profitable outcomes may require coordinated changes by many agents, lie beyond locally unfavorable regions, or disappear as more agents adopt the same strategy. These findings reveal learning challenges that arise from the interaction between strategic incentives and physically constrained power markets.

2 Related Work

Power Market RL Benchmarks. A growing number of environments support RL for bidding in power markets. At the transmission level, SustainGym (Yeh et al., 2023) studies storage bidding in a network-constrained real-time market. ASSUME (Harder et al., 2025) supports RL-based bidding in European power markets. At the distribution level, OpenGridGym (El Helou et al., 2023) studies competing bidding agents under distribution network constraints. MARLEM (Salazar-Peña et al., 2026) considers MARL-based storage participation in local energy markets. Collectively, these works demonstrate the potential of (MA)RL for learning bidding strategies under different market mechanisms and physical constraints.

However, existing environments typically focus on a single market, limiting systematic study of bidding strategies across diverse power markets. PowerMarketJax instead covers five representative markets spanning transmission and distribution levels. It further provides a unified framework for agent–market interaction while preserving each market's clearing, pricing, settlement, and physical constraints. Finally, existing environments commonly separate CPU-based market simulation from GPU-based policy learning, creating repeated CPU–GPU data transfers and making market clearing a major bottleneck for MARL training. PowerMarketJax implements both market clearing and agent policy computation in JAX, enabling parallel execution on the GPU.

JAX-based RL Benchmarks. A growing number of RL benchmarks have been implemented in JAX, motivated by the need for fast and highly parallel simulation (Rutherford et al., 2024). Isaac Gym (Makoviychuk et al., 2021) showed that moving environment simulation onto the GPU can reduce CPU–GPU communication overhead. JAX (Bradbury et al., 2018) further supports accelerator-native RL through compilation and vectorization. These advances have enabled high-throughput JAX environments across video games (Koyamada et al., 2023; Radji et al., 2026), physics (Freeman et al., 2021), robot learning (Zakka et al., 2025), and autonomous driving (Gulino et al., 2023).

Recent work has also brought JAX-based acceleration to economic and trading environments. EconoJax (Ponse et al., 2025) simulates agent-based markets with order matching, JAX-LOB (Frey

et al., 2023) accelerates limit-order-book simulation. These environments primarily rely on predefined if–else rule-based matching logic rather than constrained optimization. In contrast, power markets provide a distinctive setting where strategic economic decisions are tightly coupled to a real-world physical grid, requiring market outcomes to satisfy both market rules and engineering constraints. PowerMarketJax targets this setting by implementing optimization-based market clearing and strategic bidding entirely in JAX across five representative power markets.

3 BACKGROUND

Power systems maintain supply-demand balance through a range of markets for energy, reserves, and flexibility operating at different timescales and grid levels. Following the market structures and flexibility mechanisms discussed in Kirschen & Strbac (2026), this benchmark considers five representative settings: day-ahead wholesale, real-time balancing, ancillary services, peer-to-peer (P2P) energy trading, and local flexibility, as illustrated in Figure 1. They cover dispatch scheduling against day-ahead forecasts, adjustment to real-time imbalances, procurement of system reserves, local energy exchanges among prosumers, and distribution network support from flexible resources.

These markets share a common decision structure but differ in their mechanisms. In each market, self-interested participants such as generators, storages, and consumers submit bids based on their asset states and public market signals, without knowing competitors' private information. Their bids jointly determine market clearing, prices, and profits. However, the five markets differ in their traded products, information structures, physical constraints, and clearing and settlement rules. These differences shape participants' incentives and bidding behavior, ultimately affecting market outcomes.

M1: Day-ahead wholesale (Day $d-1$, transmission level). Generation must be scheduled before delivery day d because thermal units require time to start up and are subject to time-coupling constraints such as minimum up/down times. The day-ahead market clears supply–demand bids using forecasts for day d, determining unit commitment, generation schedules, and locational marginal prices (LMPs) that reflect transmission constraints. Cleared schedules then establish the baseline for subsequent real-time balancing.

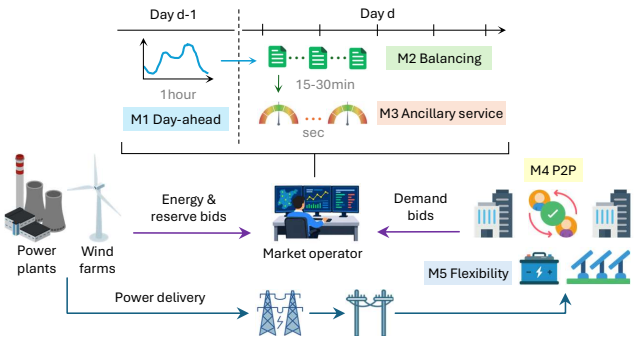


Figure 1: Five power markets in PowerMarketJax across different timescales and grid levels.

M2: Real-time balancing (Day d, transmission level). Actual demand and renewable generation may deviate from day-ahead forecasts because weather and consumption patterns cannot be predicted exactly one day in advance. The real-time market redispatches available resources over short intervals, typically every 15–30 minutes on day d, to maintain supply-demand balance under updated system conditions. Deviations from day-ahead schedules are settled at real-time prices, which can be more volatile and therefore expose participants to greater price risk.

M3: Ancillary services (Day d, transmission level). Scheduled energy alone cannot ensure reliable operation because system disturbances such as a sudden generator outage can unbalance the system within seconds, far faster than the 15–30-minute energy dispatch interval. The ancillary services market procures capabilities such as frequency regulation and operating reserves, with requirements on availability and response speed. In this market, service providers are paid for reserved capacity, so their revenue can depend on maintaining available capacity rather than solely on energy production.

M4: Peer-to-peer energy trading (Day d, distribution level). Prosumers with rooftop solar and storage can trade energy directly with nearby consumers instead of relying solely on upstream retailers. The P2P market matches buy bids and sell offers within a local energy community. The market-clearing price is determined by the intersection of the aggregated supply and demand curves, while matched quantities are determined from the submitted buy and sell quantities. Unmatched de-

mand is supplied by the upstream grid at the higher retail price, while unmatched supply is exported to the grid at the lower feed-in tariff, creating an economic incentive for local peer-to-peer trading.

M5: Local flexibility (Day d, distribution level) Even when sufficient electricity is available from the transmission system, distribution networks may become overloaded or local voltages may violate operating limits. To address these issues, the distribution system operator (DSO) procures flexibility from local resources, paying them to increase or decrease generation or consumption at specified locations, times, and durations. Under the pay-as-bid rule considered here, accepted flexibility providers are paid their submitted bid prices rather than a common clearing price.

4 POWERMARKETJAX: POWER MARKET ENVIRONMENTS IN JAX

PowerMarketJax is a suite of five power market environments implemented in JAX, with each market formulated as a MARL environment, as illustrated in Figure 2. The environments span day-ahead wholesale (M1), real-time balancing (M2), ancillary services (M3), peer-to-peer energy trading (M4), and local flexibility (M5). The markets preserve their distinct clearing, pricing, settlement, and physical constraints while following a common agent–market interaction interface.

Despite their different market rules, all five environments follow the same step sequence. Agents first observe the information available to them and submit their market actions. The market then clears according to its specific mechanism, the resulting dispatches and prices are settled, and the environment returns the next states and rewards. We define the reward as the participant profit under the corresponding settlement rule. Together, these operations define one transition of the corresponding POSG and are implemented through a common `step` interface. Further details of the five environments and their accelerator-based execution are provided below.

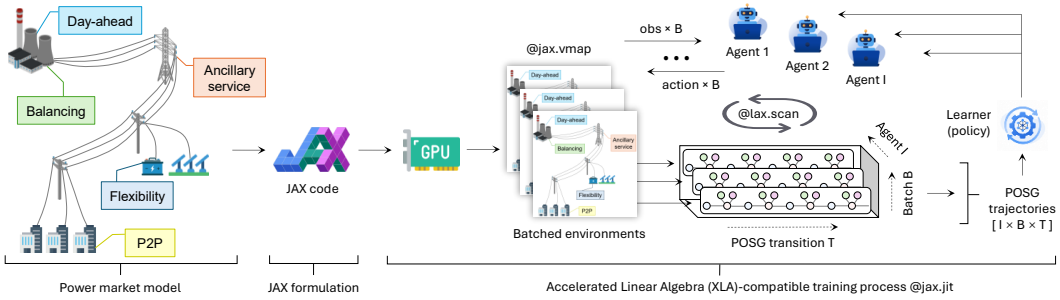


Figure 2: Execution paradigm of PowerMarketJax.

4.1 FIVE MARKET ENVIRONMENTS AS POSGS

We formulate each market environment as a general-sum partially observable stochastic game (POSG) (Hansen et al., 2004). For each environment, we describe the market step, agent observations and actions, the clearing mechanism, reward, and state transition. Detailed market models and complete POSG formulations are provided in Appendix C.1–C.5.

M1: Day-ahead wholesale. One step corresponds to one day-ahead market clearing, in which all 24 hours of the following day are cleared jointly. Each generator agent observes its initial commitment status, initial power output, and residual minimum up- and down-time requirements, together with the public day-ahead demand forecast. Its action consists of the prices submitted across all bid segments and delivery periods. Given the joint bids, the market-clearing mechanism determines unit commitment, dispatches, and LMPs, and each generator receives its daily profit as the reward.

M2: Real-time balancing. One step corresponds to one 30-minute real-time market clearing during the delivery day. Each generator agent observes its previous realized dispatch, the day-ahead commitment and schedule for the corresponding hour, the associated day-ahead LMP, and the public realized net-demand conditions. Its action consists of the prices submitted across all bid segments for the current real-time interval. Given the joint bids from all generators, the market-clearing mechanism redispatches the committed generators subject to network and ramping constraints and

determines the real-time LMPs. Each generator receives its realized interval profit under the two-settlement rule as the reward, where the day-ahead schedule remains settled at the day-ahead LMP and only deviations are settled at the real-time LMP. The realized dispatch of the current interval is carried forward as the previous dispatch for the next interval.

M3: Ancillary services. One step corresponds to one 30-minute joint energy–reserve market clearing during the delivery day. Each generator agent observes its previous realized dispatch, the day-ahead commitment and schedule for the corresponding hour, the associated day-ahead LMP, the public realized net-demand conditions, and the published reserve requirements. Its action consists of the energy bid prices across all bid segments together with one reserve bid price for each reserve product. Given the joint bids of both energy and reserve, the market-clearing mechanism co-optimizes energy dispatch and reserve procurement subject to network, ramping, response time, and reserve capacity constraints, and determines both real-time LMPs and reserve prices. Each generator then receives its realized interval profit as the reward, including day-ahead and real-time energy settlements and reserve capacity payments. The realized dispatch of the current interval is carried forward as the previous dispatch for the next interval.

M4: Peer-to-peer energy trading. One step corresponds to one 15-minute P2P trading interval. Each prosumer agent observes its own battery state of charge, net energy position, and the public retail and export prices, where a positive net position corresponds to a buy bid and a negative net position to a sell offer. It does not observe the private states or contemporaneous bids and asks of other prosumers. Its trading quantity is fixed before the auction based on its local generation, demand, and battery operation, so its action is the price submitted as an ask when selling or a bid when buying. Given the joint submissions, the double-auction mechanism sorts sellers and buyers, determines the matched quantities and uniform local clearing price, and settles any unmatched surplus or deficit with the upstream grid. Each prosumer then receives its realized interval profit as the reward, accounting for both P2P and residual grid transactions. The battery state of charge evolves to the next interval, while generation and demand update the prosumer's next net energy position.

M5: Local flexibility. One step corresponds to one 1-hour local flexibility procurement interval. Each aggregator agent observes its own battery state of charge, photovoltaic generation, local demand, and available flexibility, together with the public voltage and line-congestion requirements published by the DSO. It does not observe the private operating states, delivery costs, or contemporaneous bids of competing aggregators. Its action is a price–quantity bid specifying the requested payment and the maximum flexibility quantity it is willing to provide. Given the joint offers, the DSO clears the market by minimizing procurement cost subject to network voltage, line flow, and flexibility delivery constraints, and may partially accept a bid. Each aggregator then receives its realized interval profit as the reward under the pay-as-bid rule, with accepted flexibility paid at the submitted bid price. The accepted flexibility changes the battery state of charge and therefore determines the flexibility available in the next interval, while photovoltaic generation, local demand, and network conditions evolve to form the next market state.

4.2 Clearing and learning in a single computation graph

Despite their different clearing and settlement mechanisms, all five environments share the same accelerator-based execution framework, with market simulation and policy learning running within a single compiled computation graph.

Every operation within a training iteration, including market clearing, runs inside a single compiled computation graph on the device. `reset` and `step` are pure functions: each step sequentially applies action submission, market clearing, settlement, and observation construction, while the environment state is represented as a fixed-structure `pytree` that is passed as input and returned as output. Each step is compiled with `jit`, vectorized across parallel environments with `vmap`, and repeated over a fixed rollout horizon using `lax.scan`. The policy is evaluated within the same graph: during rollout, it maps observations to actions that are passed directly to the environment step. Policy updates are implemented with a second `lax.scan` over epochs and minibatches. A complete training iteration, consisting of rollout and update, is compiled as a single function. Further details of the compiled execution graph are provided in Appendix E.8.

Market clearing remains inside the compiled graph because each clearing mechanism is expressed with fixed shape operations and iteration counts. The optimization-based markets use a common

primal–dual interior-point solver for linear programs, with a fixed number of Newton iterations. The day-ahead market additionally includes binary commitment decisions. Rather than invoking a mixed-integer solver inside the compiled graph, we use a three-stage procedure. We first relax the commitment problem to a linear program, round the resulting commitment decisions, and then re-solve the continuous dispatch problem with the rounded commitment fixed (Appendix C.1.1). It is noted that the P2P market does not require an optimization solver. Its sorting, matching, and settlement operations are implemented directly using JAX Primitives (Bradbury et al., 2018).

We validate these market-clearing implementations in three ways (Appendix D.4). First, clearing and settlement results are compared against a NumPy reference implementation. Second, dual prices produced by the JAX linear-program solver are compared with those obtained from HiGHS (Huangfu & Hall, 2018), an independent high-performance optimization solver. Third, the three-stage procedure is compared with the exact mixed-integer solution from HiGHS, with the commitment decisions kept binary. The resulting optimality gap is reported in Appendix D.4.

We parallelize computation across both environments and agents. Each environment has its own market state and random key, and `vmap` batches multiple environments for parallel execution. Within each environment, agents remain coupled through the market-clearing mechanism, since their submitted actions jointly determine the market outcome. For policy evaluation, agents are also vectorized, so the actions of all agents across all parallel environments are computed simultaneously on the GPU in a single batched policy evaluation.

5 RESULTS

Systems and data. The three transmission-level markets (M1–M3) use the British `case29gb` system[1] with 29 buses and 66 generators as the main test case (Appendix B). They use one year of Great Britain demand and day-ahead forecast data, with 329 days for training and 36 held-out days for evaluation. M2 and M3 inherit the commitment and schedules obtained from the day-ahead market M1. Additional experiments on the 73-unit RTS-GMLC system[2] and the 151-unit Australian NEM system[3] are reported in Appendices G.1 to G.3. At the distribution level, M4 uses 1,200 Belgian households with PV from the Fluvius smart meter dataset[4]. M5 uses the 129-bus SwissDN feeder (Zapparoli et al., 2025) with battery fleets of 24 and 34 aggregators under its 2040 and 2050 scenarios. Detailed market formulations, system parameters, and data descriptions are provided in Appendices C and B.

MARL algorithms. We evaluate two independent MARL algorithms, IPPO (de Witt et al., 2020) and SAC (Haarnoja et al., 2018) (the independent version), each with parameter sharing (PS) and agent-specific parameters (NoPS), giving four learners: IPPO-PS, IPPO-NoPS, SAC-PS, and SAC-NoPS. Each agent observes only its own available information and maximizes its own market profit, without a centralized critic or inter-agent communication. We use common default hyperparameters across markets and parallel environment rollouts on the GPU. Full training configurations and market-specific hyperparameters are provided in Appendix F.

Baselines and evaluation. We compare the learned policies against truthful bidding and the corresponding untrained networks. Truthful bidding provides a competitive reference, while the untrained network is used to measure the effect of policy training. We further compare the learned policies with market-specific predefined non-learning strategies, including tests over different generator bidding levels in M1–M3, a fixed battery arbitrage schedule in M4, and unilateral flexibility-price deviations in M5. M1–M4 use 3 random seeds per experiment, while M5 uses 3–10 depending on the configuration. Evaluation uses held-out data and averages results across seeds. Full evaluation protocols and additional results are reported in Appendix G.

All experiments run on a workstation with RTX 4500 Ada GPUs, described in Appendix A.3. The market models and data, the evaluation protocol and further results are in Appendices C to G.

[1] `https://webhomes.maths.ed.ac.uk/OptEnergy/NetworkData/reducedGB/`
[2] `https://github.com/GridMod/RTS-GMLC`
[3] `https://github.com/akxen/egrimod-nem-dataset`
[4] `https://opendata.fluvius.be/`

5.1 DAY-AHEAD WHOLESALE MARKET

We start with the day-ahead market (M1) on `case29gb`. Each generator i bids at α_i times its marginal generation cost, where $\alpha_i \geq 1$. Here, $\alpha_i = 1$ corresponds to truthful bidding, while $\alpha_i > 1$ represents bidding above marginal cost. Figure 3(a) shows that all four learners end near a bidding strategy around $\alpha_i = 1.5$. Over the 36 evaluation days, IPPO-PS and IPPO-NoPS learn to raise market prices with $\alpha_i = 1.50$ and $\alpha_i = 1.56$, respectively, and thereby increasing the total generator profits by £643 and £917 million. The other two test systems show the same effect on a smaller scale, with α_i reaching 1.07–1.29 (Appendix G.1.3).

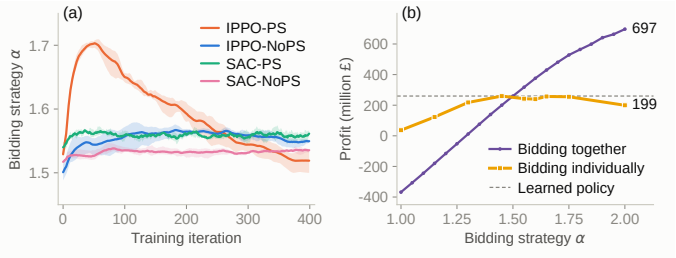


Figure 3: M1 day-ahead wholesale market (a) learning curves, (b) profit evaluation under joint and unilateral bidding strategies.

However, the learned policies still leave substantial joint profit unrealized. As shown in Figure 3(b), when all 66 generators use the same α, total profit reaches the learned policy level near $\alpha = 1.5$ and rises to £697 million at $\alpha = 2$. In contrast, when one generator's α is increased at a time while the others keep their learned IPPO-PS bids fixed, the resulting profit never exceeds the learned-policy level and is highest around $\alpha = 1.5$–1.7. The learners therefore settle near a level where raising the bid further does not benefit an individual generator, even though all generators could gain substantially by raising their bids together.

This gap arises for two reasons. First, the joint gain is not visible in each agent's individual learning signal. An independent learner observes only how its own bidding strategy affects its own profit. For example, cheap nuclear units almost never set the market price (Appendix G.1.4), so raising their bids alone has little effect on price. Second, independent exploration rarely reaches the high-profit joint bidding region. The sampled α values remain concentrated around 1.5, making it unlikely that all units simultaneously explore the higher-bid region where the joint gain occurs.

5.2 REAL-TIME BALANCING AND ANCILLARY SERVICES MARKETS

We next consider the real-time balancing (M2) and ancillary services (M3) markets. In M2, raising bid alone is rarely profitable because a generator can lose output to its competitors. Accordingly, α_i of the 30 day-ahead committed generators decreases during training under all four learners, reaching 1.04 under IPPO-PS and 1.24–1.40 under the other three methods (Figure 4(a,b)). This contrasts with the day-ahead market, where bidding above cost can increase an individual generator's profit.

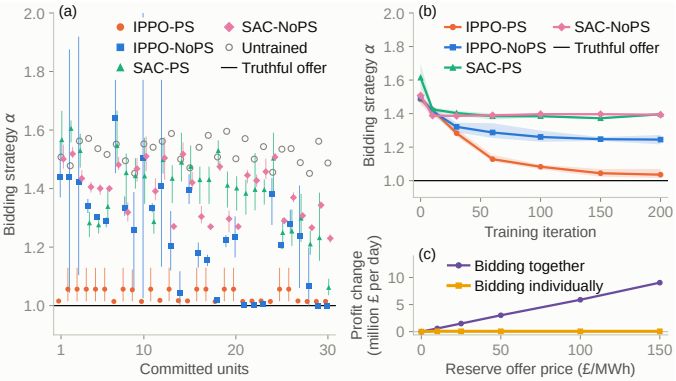


Figure 4: M2 real-time balancing market (a) bidding strategy, (b) learning curves, and M3 ancillary services market (c) profit under joint and unilateral bidding strategies.

The difference between individual and joint incentives is even clearer in M3 (Figure 4(c)). Raising the reserve bid of a single generator produces little gain: across the 29 committed generators, the largest profit increase is only £0.08 million per day. However, when all 66 generators raise their reserve bids together to

£150/MWh, total profit increases by £9.05 million per day. However, the learned policies remain far below this joint gain, earning only £0.56–£3.69 million per day more than truthful bidding, and none outperforms its corresponding non-learning strategies.

These results show how markets shape the learning signal available to each independent learner. In M2, raising a bid can reduce a generator's own profit, pushing the learned bid toward cost. In M3, the individual profit signal is nearly zero, making the much larger joint gain difficult to discover.

5.3 Peer-to-peer energy market

We next consider the P2P energy market (M4), where 1,200 households with PV and a battery trade in a double auction every 15 minutes. The learned policies improve household economics relative to truthful bidding. IPPO-NoPS increases household profit on all evaluation days, while IPPO-PS achieves the highest average profit, reducing the daily net cost per household from €1.20 under truthful bidding to €0.69–0.77 (Appendix G.4.3).

The learners, however, adopt different battery strategies (Figure 5(a)). IPPO-NoPS and SAC-NoPS charge during the midday PV surplus and discharge during the evening demand peak. IPPO-PS and SAC-PS keep their batteries close to the minimum state of charge after selling their initial energy. To understand these difference, we let an increasing number of households follow a fixed schedule that charges from 11:00–15:00 and discharges from 18:00–22:00. A single household gains about €1.13 per day, but this gain decreases as more households adopt the same strategy and becomes negative at around 210–240 households (black line in Figure 5(b)), detailed in Appendix G.4.4.

The decline is caused by the resulting change in market prices (Figure 5(c)). Simultaneous charging raises the midday clearing price, while simultaneous discharging lowers the evening price, reducing the price differentials on which the arbitrage relies. Once enough households participate, the remaining differential no longer covers efficiency losses and degradation costs. Thus, a strategy that is profitable for one household may become unprofitable when widely adopted. Figure 5(b) further suggests that PS can make market outcomes requiring heterogeneous behavior harder to learn: IPPO-NoPS leads only a subset of households to arbitrage, while IPPO-PS leads none to do so.

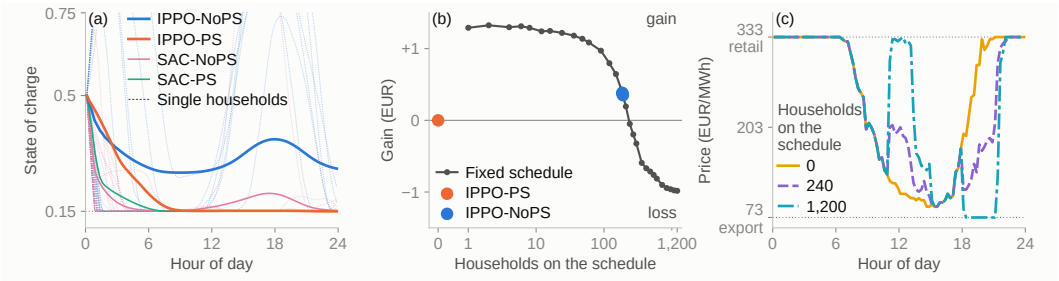


Figure 5: P2P energy market (a) battery state of charge, (b) household gain, (c) local clearing price.

5.4 Local flexibility market

Finally, we consider the local flexibility market (M5), where the DSO procures flexibility from battery aggregators to relieve network congestion. The market is pay-as-bid with a price floor. All four learners remain close to the floor. Yet a single aggregator can earn more by raising its offer sufficiently far above the floor; in the two 3.9 MW configurations, its return exceeds the floor-level return again at prices around 3.2 times the floor (Figure 6(a,b)).

The reason is the shape of the individual return. A small bidding price increase from an aggregator causes the operator to procure from other cheaper aggregators instead, sharply reducing the deviating aggregator's award and profit. Only at much higher prices does its return recover. Gradient-based learners therefore encounter a local barrier: small moves away from the floor reduce profit, while the more profitable region lies much farther away. The price parameterization further weakens exploration near the floor because the softplus mapping becomes nearly flat.

The learners instead increase profit through their planned charging decisions. Planned charging raises the baseline flow on the congested line and therefore increases the flexibility requirement that the operator subsequently procures. With SAC-PS learners, the share of periods with procurement rises from 5.0% to 18.7% with 24 aggregators and from 4.9% to 15.4% with 34 aggregators (Figure 6(c)). A baseline monitor that excludes this learner-induced charging largely removes the effect, causing both procurement and learner returns to fall sharply. Thus, M5 exposes two distinct challenges for MARL in markets: profitable deviations may lie beyond a local learning barrier, and agents can change the demand for the service they are paid to provide.

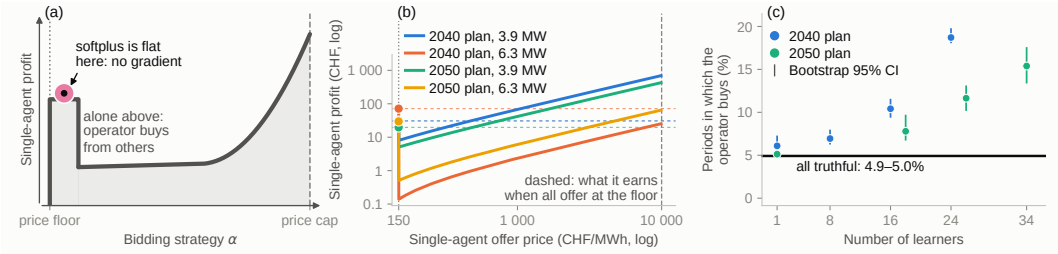


Figure 6: M5 local flexibility market (a) schematic return versus own offer price, (b) measured return in four system configurations, (c) procurement frequency versus the number of SAC-PS learners (seed mean with 95% bootstrap CI).

6 Conclusion

PowerMarketJax shows that MARL benchmarks for power markets can preserve market-design fidelity while remaining computationally scalable. By integrating market clearing, settlement, policy evaluation, and learning into a GPU-accelerated JAX pipeline, the benchmark supports fast multi-agent rollouts while retaining the mechanisms and physical constraints that shape strategic bidding. The five markets provide a common POSG framework for evaluating learned bidding behavior and market outcomes across diverse market designs.

Discussion. The experiments show that learned bidding behavior depends strongly on market design and the profit signal available to each agent. Across the five markets, learners may miss profitable joint outcomes, move toward truthful bidding when individual deviations are unprofitable, lose arbitrage gains when many agents adopt the same strategy, or fail to reach more profitable strategies beyond lower-profit regions. More broadly, these results show that market mechanisms shape the learning landscape itself: they determine which incentives are applied to individual agents and how the actions of many learners feed back into prices, dispatch, and procurement. This reflects challenges for MARL beyond maximizing return, including coordinated exploration, endogenous market responses, and heterogeneous agent behavior. Such learning-based simulations can also help stress-test market designs and reveal strategic incentives before new mechanisms are deployed.

Limitations. PowerMarketJax is a research benchmark rather than a deployment-ready market simulator or bidding system, and the five environments simplify aspects of real market operation, network modeling, participant constraints, and market rules. Our experiments cover only the considered systems and independent PPO and SAC variants, so the learned policies should not be interpreted as predictions of real participant behavior or as equilibrium strategies. Future work could extend the benchmark to richer market designs, participant models, uncertainty, and broader classes of MARL algorithms.

AI USE STATEMENT

In this work, we used generative AI tools in three ways. To aid and polish writing: the text was drafted by the authors, and AI tools edited it for readability and checked spelling and grammar. For retrieval and discovery: AI tools identified and summarised related work, and every cited work was checked against its source. For research ideation and execution: with the algorithms, the theory and the scenario choices provided by the authors, AI tools run as a coding agent were used for implementing methods, designing experiments, cleaning and reformatting datasets, suggesting experimental parameters, creating figures. We have not used generative AI tools for generating synthetic datasets; formulating or proving mathematical claims is not applicable to this work. We have reviewed all AI-assisted work. AI-written code was verified by tests against independent reference implementations, and the design decisions, the claims and the final text were decided by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects and no new data collection. The only household-level data used in the reported results is the Fluvius smart-meter panel, published by its provider as an anonymised sample with no name, address or location; the release keeps it in the published form and adds no attribute that could identify a dwelling (Appendix B.2). Every dataset is redistributed under its publisher's licence, with the required attribution (Appendix A.2). The benchmark studies how learning agents bid, including strategies that raise prices above cost, and such insight could in principle inform the exercise of market power. We consider the benefit to market design and monitoring to outweigh this risk: the markets are research models rather than replicas of a real market, and results on them describe how learners behave under a given mechanism, not how to profit in a particular market (Appendix B.5). The authors declare no competing interests.

REPRODUCIBILITY STATEMENT

The source code of PowerMarketJax, together with the experiment scripts (drivers) that produced every reported result, is released at the anonymous repository given in Appendix A.1; Appendix A also lists what the release contains, the licences of the code and of each dataset, and the software and hardware the experiments ran on. The five market models' clearing problems, pricing and settlement rules, observations, actions and rewards are stated formally in Appendix C. The datasets, how they were obtained and how they were processed are described in the datasheet of Appendix B, and each shipped series carries the script that rebuilds it from its public source. Appendix E shows how to install the package, construct each market, launch training and evaluation from the command line, and find where the learner and scenario configuration of a run is recorded. The four learners, their hyperparameters, the training scale and the evaluation protocol are given in Appendix F, and the setting and training runs behind the results of each market open the corresponding subsection of Appendix G. The released test suite checks the clearing operators and the settlement against independent reference implementations (Table 1).

REFERENCES

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Thomas Brown, Jonas Hörsch, and David Schlachtberger. PyPSA: Python for power system analysis. *Journal of Open Research Software*, 6(1):4–4, 2018.

A.K. David and Fushuan Wen. Strategic bidding in competitive electricity markets: a literature survey. In *2000 Power Engineering Society Summer Meeting (Cat. No.00CH37134)*, volume 4, pp. 2168–2173 vol. 4, 2000.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the StarCraft Multi-Agent Challenge? arXiv:2011.09533, 2020.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Yan Du, Fangxing Li, Helia Zandi, and Yaosuo Xue. Approximating Nash equilibrium in day-ahead electricity market bidding with multi-agent deep reinforcement learning. *Journal of Modern Power Systems and Clean Energy*, 9(3):534–544, 2021.

Rayan El Helou, Kiyeob Lee, Dongqi Wu, Le Xie, Srinivas Shakkottai, and Vijay Subramanian. OpenGridGym: An open-source AI-friendly toolkit for distribution market simulation. *IEEE Transactions on Smart Grid*, 14(2):1555–1565, 2023.

Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Sascha Yves Frey, Kang Li, Peer Nagy, Silvia Sapora, Christopher Lu, Stefan Zohren, Jakob Foerster, and Anisoara Calinescu. JAX-LOB: A gpu-accelerated limit order book simulator to unlock large scale reinforcement learning for trading. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pp. 583–591, 2023.

Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36: 7730–7742, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pp. 709–715. AAAI Press / The MIT Press, 2004.

Nick Harder, Kim K. Miskiw, Manish Khanra, Florian Maurer, Parag Patil, Ramiz Qussous, Christof Weinhardt, Marian Klobasa, Mario Ragwitz, and Anke Weidlich. ASSUME: An agent-based simulation framework for exploring electricity market dynamics with reinforcement learning. *SoftwareX*, 30:102176, 2025.

Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

Daniel S. Kirschen and Goran Strbac. *Fundamentals of Power System Economics*. Wiley, 3rd edition, 2026.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:45716–45743, 2023.

Yanchang Liang, Chunlin Guo, Zhaohao Ding, and Huichun Hua. Agent-Based modeling in electricity market using deep deterministic policy gradient algorithm. *IEEE Transactions on Power Systems*, 35(6):4180–4192, 2020.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-Based physics simulation for robot learning. arXiv:2108.10470, 2021.

Carlos Matamala and Goran Strbac. Strategic bidding in the frequency-containment ancillary services market. *Applied Energy*, 401:126811, 2025.

Koen Ponse, Aske Plaat, Niki van Stein, and Thomas M. Moerland. Econojax: A fast & scalable economic simulation in jax. In *AAMAS '25: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1679–1687, Richland, SC, 2025.

Waris Radji, Thomas Michel, and Hector Piteau. Octax: Accelerated chip-8 arcade environments for reinforcement learning in jax. In *International Conference on Learning Representations*, volume 2026, pp. 72220–72256, 2026.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Foerster. JaxMARL: Multi-Agent RL environments and algorithms in JAX. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, pp. 50925–50951, 2024.

Nelson Salazar-Peña, Alejandra Tabares, and Andrés González-Mancera. MARLEM: A multi-agent reinforcement learning simulation framework for implicit cooperation in decentralized local energy markets. *Applied Energy*, 412:127546, 2026.

Thomas Wolgast and Astrid Nieße. Approximating energy market clearing and bidding with Model-Based reinforcement learning. *IEEE Access*, 12:145106–145117, 2024.

Christopher Yeh, Victor Li, Rajeev Datta, Julio Arroyo, Nicolas Christianson, Chi Zhang, Yize Chen, Mohammad Mehdi Hosseini, Azarang Golmohammadi, Yuanyuan Shi, Yisong Yue, and Adam Wierman. SustainGym: Reinforcement learning environments for sustainable energy systems. In *Advances in Neural Information Processing Systems 36*, pp. 59464–59476, 2023.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Demonstrating MuJoCo Playground. In *Robotics: Science and Systems XXI*, 2025.

Lorenzo Zapparoli, Alfredo Oneto, María Parajeles Herrera, Blazhe Gjorgiev, Gabriela Hug, and Giovanni Sansavini. Future deployment and flexibility of distributed energy resources in the distribution grids of Switzerland. *Scientific Data*, 12:1491, 2025.

Ziqing Zhu, Siqi Bu, Ka Wing Chan, Fangxing Li, Yujian Ye, Chi Yung Chung, and Goran Strbac. Designing the future electricity spot market with high renewables via reliable simulations. *Nature Reviews Electrical Engineering*, 2(5):320–337, 2025.

APPENDIX CONTENTS

A Reproducibility and Release Metadata **15**
A.1 Hosting and Maintenance . . . 15
A.2 Licences and Data Package . . . 15
A.3 Compute Disclosure . . . 15
A.4 Release Contents . . . 15

B Datasheet for the Benchmark **17**
B.1 Motivation . . . 17
B.2 Composition . . . 17
B.3 Collection Process . . . 18
B.4 Preprocessing . . . 18
B.5 Use . . . 18
B.6 Distribution . . . 18
B.7 Maintenance . . . 18

C Market Models and Partially Observable Stochastic Game **19**
C.1 M1: Day-ahead Wholesale Market . . . 19
C.2 M2: Real-Time Balancing Market . . . 23
C.3 M3: Ancillary Services Market . . . 26
C.4 M4: Peer-to-Peer Local Energy Market . . . 30
C.5 M5: Local Flexibility Market . . . 34

D Computation Performance **39**
D.1 Three layouts on the peer-to-peer market . . . 39
D.2 Across the five markets . . . 39
D.3 Against open-source tools . . . 39
D.4 Implementation validation . . . 40

E Quickstart and API **43**
E.1 Installation . . . 43
E.2 Minimal rollout . . . 43
E.3 The environment interface . . . 45
E.4 Constructing the five markets . . . 45
E.5 Scenario and learner configuration . . . 46
E.6 Command-line usage . . . 46
E.7 Cases and data . . . 47
E.8 Clearing and learning in a single computation graph . . . 47

F Training Setup and Hyperparameters **50**

F.1 Four learners 50
F.2 Hyperparameters 50
F.3 Training scale 51
F.4 Evaluation protocol 51

G Additional Market Results **52**
G.1 M1: Day-ahead Wholesale Market 52
G.2 M2: Real-Time Balancing Market 60
G.3 M3: Ancillary Services Market 64
G.4 M4: Peer-to-Peer Local Energy Market 67
G.5 M5: Local Flexibility Market 71

A Reproducibility and Release Metadata

This section gives the release information a reader needs before reproducing the results: what is released, and under what licences the code and the data are distributed.

A.1 Hosting and Maintenance

PowerMarketJax is released as a single implementation at `https://github.com/powermarketjax/PowerMarketJax`. The package ships the five market environments with their clearing and settlement layers, the interior-point solver they share, the 18 network cases with the power-flow solvers and device models they are built on, the data loader with the datasets it resolves, and the IPPO and SAC learners, together with the experiment scripts that produced the reported results and the data acquisition scripts.

The repository is hosted under an account created for anonymous review. Commit history uses placeholder author metadata and neither the repository nor the account links to author-identifying content.

A.2 Licences and Data Package

Code licence. PowerMarketJax is released under the MIT Licence, with the licence file at the release root. Parts of the package are vendored from an MIT-licenced library, with its notice retained in each copied file.

Data package. Each of the 15 registered series has a JSON manifest giving its source organisation and URL, resolution, date range, time zone and column mapping into the loader's schema; a market loads a manifest, not a path. Six further Parquet files are read directly: four hold the Swiss feeder's topology, storage siting and representative photovoltaic days, and two the annual electricity tariffs published by the Swiss regulator ElCom, by consumer category and by municipality. Each shipped series has a script that rebuilds it from the public source.

Data licences and redistribution. Each series keeps its publisher's licence; the code licence covers none of them. Two obligations travel with every copy: the Flemish distribution operator requires its name together with the dataset's last update date, which is carried in the dataset metadata, and the RTS-GMLC disclaimer must appear in all copies, so the file stating it ships beside its case. Every shipped dataset carries its attribution.

Author responsibility. The authors take responsibility for the released materials. One series is household-level smart-meter measurements, published by their providers as anonymised samples with no location or identity attached; how they were sampled is described in Appendix B.2. To the authors' knowledge the release contains no personally identifiable information. If we are notified of a rights violation the maintainers will address it; contact details are omitted during anonymous review.

A.3 Compute Disclosure

All experiments run under Python 3.11.15 with JAX and jaxlib 0.10.2, CUDA support added as the same-version plugin rather than by changing either, and double precision and highest matmul precision enabled, on a single shared workstation with three NVIDIA RTX 4500 Ada (24 GB) GPUs and an AMD Ryzen Threadripper PRO 7985WX host (64 cores, 128 threads). Throughput measurements are in Appendix D.

A.4 Release Contents

Table 1 lists what the release contains and where each item lives.

Table 1: Release contents

Component	Location	Purpose
Environments, solver, cases, learners	`powermarketjax/`	Inspect the five markets, the shared interior-point solver, and the reward and constraint-cost interface
Datasets and their registry	`powermarketjax/data/`	15 manifests, and the 20 Parquet files read through them or directly by the code
Test suite	`tests/`	169 test files over four layers
Experiment scripts of the reported results	`tools/benchmark/` and the per-market directories	Train, evaluate, and run the non-learning baselines; the release includes only the scripts behind reported results
Learner and scenario configuration	`tools/benchmark/hyperparams.py` (learners), constants at the top of each experiment script, each also a command-line flag (scenario)	Reconstruct the hyperparameters and the experimental configuration
Data acquisition scripts	`tools/data_prep/`	Rebuild any dataset from its public source and see what processing was applied

B DATASHEET FOR THE BENCHMARK

This datasheet covers the released artifact of Appendix A: five market environments, the 18 network cases they are built on, and the 15 registered measurement series they read. The formal market models are in Appendix C.

B.1 MOTIVATION

Existing MARL environments for power markets cover one market at a time, simplify the clearing rule, or call a CPU solver once per step. This benchmark was built to lift those three limits together: five markets with clearing rules that differ in kind, each cleared by its own mechanism rather than by a pricing heuristic, and the whole loop compiled to run on the accelerator. It was assembled by the authors for the study reported here.

B.2 COMPOSITION

What does the benchmark include? An instance is one episode of one market: a network case, a set of scenario parameters, a calendar day from a real measurement series, and a seed. Fifteen series are registered: demand, generation by type, day-ahead and dispatch prices, rooftop photovoltaic output, and household and feeder metering from Great Britain, Australia, Belgium and Switzerland. Six further files are read directly rather than through a manifest: four hold the Swiss feeder's topology, storage siting and representative photovoltaic days, and two the annual electricity tariffs published by the Swiss regulator ElCom, by consumer category and by municipality. Eighteen cases ship with them, ten transmission and eight distribution, from 5 to 2 383 buses.

How many instances? There is no fixed instance count; Table 2 gives the configurations this paper reports, and their scenario constants are in Appendix C.

Table 2: The five markets at the experimental configurations reported here. One step of M1 is a market day, because the day-ahead market clears all periods of the next day at once; one step of the others is one trading period.

Market	Case	Agents	Period × periods	Steps	Evaluation set
M1 Day-ahead	`case29gb`, `case73rts`, `case813nem`	66 73 151	1 h × 24	1	36 days (one year)
M2 Real-time	`case29gb`, `case73rts`	66 73	0.5 h × 48	48	the same days as M1
M3 Ancillary	`case29gb`, `case73rts`, `case813nem`	66 73 151	0.5 h × 48	48	the same days as M1
M4 Peer-to-peer	1 200 households	1 200	0.25 h × 96	96	1 024 held-out episodes (see Appendix G.4)
M5 Local flex.	SwissDN `459_0`, 129 buses	24 or 34	1 h × 24	24	36 test episodes per configuration

Are there missing values? No: nothing is imputed in the series this paper uses, and the Flemish panel is complete over its 1 200 households used here by 35 136 quarter hours with no missing value.

Does it relate to people? Yes. One series is household-level smart-meter measurements, published by their providers as anonymised samples with no name, address or location attached; the authors had no contact with the households, and consent for collection and release rests with the providers. The Flemish panel is a stratified sample of 2 400 households in eight groups of 300, defined by three household attributes: a photovoltaic array, a heat pump as main heating, and a home-charging electric vehicle. The four groups with an array, 1 200 households, are used here. Quarter-hourly consumption can reveal when a household is at home, so the release keeps the series in the providers' published form and adds no attribute that could link a trace to a dwelling.

B.3 Collection Process

Each time series is a processed derivative of a public release, obtained by a script that ships with the benchmark. Cases are compiled from published network and unit data, and the two compiled from third-party releases carry that release's licence file beside them. The reported results use the following periods: Great Britain demand from 2023-07-10 to 2024-07-08 for the day-ahead, real-time and ancillary markets; Fluvius household metering from 2024 for the peer-to-peer market; and, for the local flexibility market, the one-year load profile published with the SwissDN feeder, with its battery fleet taken from the dataset's 2040 and 2050 projections. The two further transmission cases of Appendix G.1 use RTS-GMLC data for 2020 and Australian market data from 2025-02 to 2026-01.

B.4 Preprocessing

Each series is passed through column harmonisation into one schema, unit conversion, and construction of a uniform time base with an explicit time zone; no value is imputed in any series used for the reported results. The release does not ship a separate raw copy of each series alongside the processed one. The acquisition script of Appendix B.3 reconstructs the unprocessed form from the public source rather than the release retaining it.

Four reported configurations depart from the measured data through a declared scenario parameter rather than an edit to a series: the line ratings of `case29gb` are scaled by 0.60, because at their published values no line congests; the ancillary market computes its reserve requirement from the realised demand times 1.02 in place of the published forecast, identically for the learning runs and the baselines; the local flexibility market multiplies the feeder load by 1.50 (Appendix G.5); and `case73rts` nets hydro, wind and solar output off demand (Appendix G.1). The shipped series themselves are not modified.

B.5 Use

What can the benchmark be used for? So far it has been used only for the study in this paper (Section 5), but it is designed for wider use. This benchmark can be used to test multi-agent learning methods on strategic bidding problems, to compare market mechanisms under the same learning algorithms, and to run scenario studies that change the network, the market rules or the population of participants without changing the learning code. It also supports large-population and mixed-population studies, in which part of the agents follow fixed non-learning strategies, and constrained reinforcement learning, since constraint violations are returned separately from the reward.

B.6 Distribution

Licence terms and redistribution status are stated in Appendix A.2.

B.7 Maintenance

The maintainers commit to at least two years of issue tracking, semantic versioning of breaking changes, and dependency-bump support for current JAX/CUDA/Python combinations. Contact details are omitted during anonymous review and added in the camera-ready version.

C MARKET MODELS AND PARTIALLY OBSERVABLE STOCHASTIC GAME

C.1 M1: DAY-AHEAD WHOLESALE MARKET

The day-ahead market is a centralized auction held on day $d-1$ that clears all T hourly delivery periods of day d simultaneously, with $\Delta^{\mathrm{da}} = 1$ hour. Market participants submit their strategic bids, which the market operator clears through a security-constrained unit commitment (SCUC) problem. The resulting schedules are priced and settled using locational marginal prices (LMPs). Section C.1.1 presents the market clearing model, section C.1.2 introduces the pricing and settlement rules, and section C.1.3 formulates the partially observable stochastic game (POSG) for strategic bidding in day-ahead wholesale market.

C.1.1 MARKET MODEL

Each generator agent i submits, for each delivery period t, a stepwise supply curve consisting of K segments. Each segment k has quantity

$$G_{i,k} = \frac{\bar{p}_i - \underline{p}_i}{K} \tag{1}$$

and bid price $\pi_{i,k,t}$, where $\underline{p}_i$ and $\bar{p}_i$ are the minimum and maximum outputs of the unit. Bid prices are required to be non-decreasing across segments, i.e., $\pi_{i,k,t} \le \pi_{i,k+1,t}$. Technical parameters, including ramp rates, minimum up- and down-times, start-up costs, and no-load costs, are registered with the market operator and are not part of the per-period bid. Demand is inelastic and treated as exogenous.

The day-ahead market jointly determines unit commitment and dispatch over all T hourly delivery periods by solving

$$\min_{\{g,\,u,\,v,\,w,\,s\}} \sum_{t=1}^{T} \left[\sum_{i,k} \pi_{i,k,t}\, g_{i,k,t} + \sum_{i} \Delta^{\mathrm{da}}\, \mathrm{NL}_i\, u_{i,t} + \mathrm{VOLL} \sum_{n} s_{n,t} \right] + \sum_{i,t} S_i\, v_{i,t}, \tag{2a}$$

subject to

$$p_{i,t} = \underline{p}_i\, u_{i,t} + \sum_{k} g_{i,k,t}, \tag{2b}$$

$$0 \le g_{i,k,t} \le G_{i,k}\, u_{i,t}, \tag{2c}$$

$$P_{n,t} = \sum_{i\,:\,\mathrm{bus}(i)=n} p_{i,t} + s_{n,t} - d_{n,t}, \tag{2d}$$

$$\sum_{n} P_{n,t} = 0 \quad (:\lambda_t), \tag{2e}$$

$$-F \le \mathrm{PTDF} \cdot P_t \le F \quad (:\mu_t^{+},\, \mu_t^{-}), \tag{2f}$$

$$0 \le s_{n,t} \le d_{n,t} \qquad (:\rho_{n,t}), \tag{2g}$$

$$-R_i^{\mathrm{dn}} \Delta^{\mathrm{da}} - \bar{p}_i\, w_{i,t} \;\le\; p_{i,t} - p_{i,t-1} \;\le\; R_i^{\mathrm{up}} \Delta^{\mathrm{da}} + \underline{p}_i\, v_{i,t}, \tag{2h}$$

$$u_{i,t} - u_{i,t-1} = v_{i,t} - w_{i,t}, \tag{2i}$$

$$v_{i,t} + w_{i,t} \le 1, \tag{2j}$$

$$\sum_{\tau=t-\mathrm{UT}_i+1}^{t} v_{i,\tau} \le u_{i,t}, \tag{2k}$$

$$\sum_{\tau=t-\mathrm{DT}_i+1}^{t} w_{i,\tau} \le 1 - u_{i,t}, \tag{2l}$$

$$u_{i,t},\, v_{i,t},\, w_{i,t} \in \{0,1\}. \tag{2m}$$

The objective (2a) minimizes the total offered operating cost over the delivery day. The first term represents the cost of accepted energy offers, the second accounts for the no-load cost of committed generators, and the third penalizes involuntary load shedding. The start-up cost S_i is incurred

whenever generator i transitions from off to on. Here, $\Delta^{\mathrm{da}} = 1\ \mathrm{h}$ is the duration of each day-ahead delivery period, NL_i is the no-load cost, and VOLL is the value of lost load, which is set sufficiently high so that load shedding is used only as a last resort.

Constraints (2b) and (2c) define the output of each generator. The binary variable $u_{i,t}$ indicates whether generator i is committed. When $u_{i,t} = 1$, its output consists of the minimum generation level $\underline{p}_i$ plus the accepted quantities $g_{i,k,t}$ from its offer segments. Each accepted segment is bounded by $G_{i,k}$, so the total output remains within $[\underline{p}_i, \bar{p}_i]$. When $u_{i,t} = 0$, all segment quantities and the generator output are zero.

Constraints (2d)–(2f) impose the network constraints. Equation (2d) defines the net injection $P_{n,t}$ at bus n as local generation plus load shedding minus exogenous demand $d_{n,t}$. Constraint (2e) enforces system-wide supply–demand balance. Under the DC power-flow approximation, constraint (2f) maps the nodal injections to transmission-line flows through the power transfer distribution factor matrix $\mathrm{PTDF} \in \mathbb{R}^{N^{\mathrm{line}} \times N^{\mathrm{bus}}}$ and limits each line flow by its rating F. Network losses are not modeled.

Constraint (2g) bounds load shedding at each bus by the local demand. The variable $s_{n,t}$ provides emergency feasibility when available generation and transmission capacity are insufficient to serve all demand. Because load shedding carries the large penalty VOLL in the objective, it is used only when lower-cost feasible alternatives are unavailable.

Constraint (2h) limits the change in generator output between consecutive delivery periods according to the upward and downward ramp rates R_i^{up} and R_i^{dn}. The start-up term $\underline{p}_i v_{i,t}$ allows a unit that starts in period t to move from zero output to at least its minimum generation level, while the shut-down term $\bar{p}_i w_{i,t}$ allows a unit that shuts down to reduce its output to zero.

Constraints (2i) and (2j) describe commitment transitions. The binary variables $v_{i,t}$ and $w_{i,t}$ denote start-up and shut-down events, respectively. Equation (2i) links these events to changes in commitment status, while equation (2j) prevents a generator from starting up and shutting down in the same period.

Constraints (2k) and (2l) enforce the minimum up- and down-times UT_i and DT_i. After a start-up, the unit must remain committed for at least UT_i periods; after a shut-down, it must remain offline for at least DT_i periods. The initial commitment status $u_{i,0}$, initial output $p_{i,0}$, and any residual minimum up- or down-time requirements from the previous day are treated as exogenous initial conditions.

Solution method. Optimization (2) is a security-constrained unit commitment (SCUC) problem formulated as a mixed-integer linear program (MILP). Exact MILP solvers rely on dynamic branch-and-bound procedures that are difficult to batch and compile efficiently on accelerators. We therefore use a relax-round-resolve procedure. First, the binary commitment variables are relaxed to $u_{i,t} \in [0, 1]$, and the resulting linear program is solved. Second, the relaxed commitment is rounded by setting $u_{i,t}^{\mathrm{int}} = 1$ if $u_{i,t} > \varepsilon$ and $u_{i,t}^{\mathrm{int}} = 0$ otherwise, from which the start-up and shut-down indicators $v_{i,t}^{\mathrm{int}}$ and $w_{i,t}^{\mathrm{int}}$ are determined. Third, these commitment decisions are fixed and the continuous dispatch problem is re-solved as a security-constrained economic dispatch (SCED). Dispatch schedules are obtained from this final solve, while LMPs are computed from the dual variables of its power balance and transmission constraint. This separation follows the common practice of determining commitment first and obtaining dispatch and prices with commitment fixed. The optimality gap relative to the exact solution is evaluated offline and reported in Appendix D.4.

C.1.2 Pricing and settlement

Given the fixed commitment obtained from the solution procedure above, LMPs are computed from the dual variables of the final SCED problem. The LMP at bus n at period t is the marginal cost of serving one additional MWh at that bus and is given by

$$\mathrm{LMP}_{n,t} = \lambda_t - \sum_l \left(\mu_{l,t}^{+} - \mu_{l,t}^{-}\right) \mathrm{PTDF}_{l,n} - \rho_{n,t}, \tag{3a}$$

where the first term is the system marginal energy price, which is common across the transmission network; the second term is the congestion component, which is zero when no transmission

constraint is binding and causes nodal prices to differ under congestion; and the third term can be non-zero only when the upper bound on load curtailment is binding. There is no loss component because transmission losses are not modeled.

Cleared quantities $p_{i,t}$ are settled at the LMP of the bus where unit i is located. The revenue, cost, and profit of unit i over the market day are

$$\text{revenue}_i = \sum_t \Delta^{\text{da}} \cdot \text{LMP}_{\text{bus}(i),t}\, p_{i,t}, \tag{3b}$$

$$\text{cost}_i = \sum_t \left[\Delta^{\text{da}} \cdot \text{TC}_i(p_{i,t}) + \Delta^{\text{da}} \cdot \text{NL}_i\, u_{i,t}\right] + \sum_t S_i\, v_{i,t}, \tag{3c}$$

$$\text{profit}_i = \text{revenue}_i - \text{cost}_i, \tag{3d}$$

where $\text{TC}_i(p) = \int_0^p \text{MC}_i(x)\,dx$ is the true variable generation cost rate of unit i, and $\text{MC}_i(p) = a_i p^2 + b_i p + c_i$ is its true marginal cost. Setting $a_i = 0$ recovers a linear marginal cost curve. The true variable generation cost $\text{TC}_i(\cdot)$ is used only to evaluate participant profit and does not enter market clearing; the clearing problem instead uses submitted energy offer prices together with the registered no-load and start-up costs. Settlement does not include uplift payment, so a committed unit may earn negative profit when its energy market revenue does not recover its start-up and no-load costs.

C.1.3 Partially Observable Stochastic Game

We formulate repeated participation in the day-ahead wholesale market as a general-sum POSG

$$\mathcal{G}^{\text{DA}} = \langle\, \mathcal{I},\, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}},\, P,\, \{r_i\}_{i\in\mathcal{I}},\, \gamma\,\rangle\,, \tag{4}$$

where $\mathcal{I}$ is the set of strategic generator agents, $\mathcal{S}$ is the market state space, $\mathcal{O}_i$ and $\mathcal{A}_i$ are the observation and action spaces of agent i, P is the state transition kernel, r_i is the individual profit-based reward, and $\gamma \in [0,1]$ is the discount factor. One episodic market step corresponds to one day-ahead market clearing, in which all T delivery periods of the following day are cleared jointly.

State. At market day d, the underlying state contains the system information required to clear the following delivery day,

$$s_d = (\, x_d^{\text{sys}}, \{x_{i,d}\}_{i\in\mathcal{I}}\,) \in \mathcal{S}, \tag{5}$$

where x_d^{sys} contains the day-ahead forecast of system demand $d_{n,t,d}$ and other exogenous system conditions such as renewable generation. The generator state is

$$x_{i,d} = \left(u_{i,d}^0, p_{i,d}^0, \tau_{i,d}^{\text{up}}, \tau_{i,d}^{\text{dn}}\right), \tag{6}$$

where $u_{i,d}^0$ and $p_{i,d}^0$ are the commitment status and power output of generator i entering the delivery day, and $\tau_{i,d}^{\text{up}}$ and $\tau_{i,d}^{\text{dn}}$ denote any residual minimum up- and downtime requirements carried over from the previous day. Other system and generator parameters, such as the PTDF matrix PTDF, transmission limits F, generation capacity limits $(\underline{p}_i, G_{i,k})$, and ramp rates $(R_i^{\text{up}}, R_i^{\text{dn}})$, are fixed environment parameters known to the market operator.

Observation. Each generator i observes its own private information and the public information available before bidding,

$$o_{i,d} \in \mathcal{O}_i(s_d) = \left(x_{i,d}, x_d^{\text{pub}}\right), \tag{7}$$

where x_d^{pub} includes the public day-ahead system information of demand forecast $d_{n,t,d}$. An agent does not observe the private generation costs or contemporaneous offers of competing generators before market clearing. Hence, $o_{i,d}$ does not fully reveal s_d, making the bidding problem partially observable.

Action. The action of generator i is a single markup $\alpha_{i,d}$ on its truthful offer $\pi_{i,1}^0 \le \cdots \le \pi_{i,K}^0$, in which each segment is priced at the unit's true marginal cost MC_i, giving its complete day-ahead price bid over all segments and delivery periods,

$$a_{i,d} = \alpha_{i,d} \in \mathcal{A}_i = [1, \bar{\alpha}], \qquad \pi_{i,k,t,d} = \alpha_{i,d}\, \pi_{i,k}^0, \quad k = 1,\ldots,K;\ t = 1,\ldots,T, \tag{8}$$

where $\bar{\alpha}$ is the largest admissible markup. The segment quantities $G_{i,k}$ are fixed as defined in Section C.1.1, while the agent chooses the markup. Admissible bids satisfy the market price bounds and the monotonicity rule

$$\underline{\pi} \le \pi_{i,1,t,d} \le \pi_{i,2,t,d} \le \cdots \le \pi_{i,K,t,d} \le \overline{\pi}, \qquad \forall t. \tag{9}$$

The joint action $a_d = (a_{1,d}, \ldots, a_{|\mathcal{I}|,d})$ therefore represents the set of supply bids submitted by all strategic generators for the following delivery day.

Market clearing and outcome. Given state s_d and joint action a_d, the market operator applies the day-ahead clearing model and pricing rules described in Sections C.1.1–C.1.2. We denote this mapping by

$$z_d = \mathcal{C}^{\mathrm{DA}}(s_d, a_d), \tag{10}$$

where the market outcome z_d contains the commitment decisions $u_{i,1\ldots T,d}$, accepted generation quantities $g_{i,k,1\ldots T,d}$, dispatch schedules $p_{i,1\ldots T,d}$, load curtailment $s_{n,1\ldots T,d}$, and nodal LMPs $\mathrm{LMP}_{n,1\ldots T,d}$ for all T delivery periods. Through this clearing mechanism, the payoff of one agent depends not only on its own offer but also on the bids submitted by the other agents.

Reward. Each agent i receives its realized market profit as reward. Using the settlement defined in Section C.1.2,

$$\begin{aligned} r_{i,d} &= \mathrm{profit}_{i,d} = \mathrm{revenue}_i - \mathrm{cost}_i \\ &= \sum_{t=1}^{T} \Delta^{\mathrm{da}}\, \mathrm{LMP}_{\mathrm{bus}(i),t,d}\, p_{i,t,d} \\ &\quad - \sum_{t=1}^{T} \left[\, \Delta^{\mathrm{da}}\, \mathrm{TC}_i(p_{i,t,d}) + \Delta^{\mathrm{da}}\, \mathrm{NL}_i u_{i,t,d} \,\right] - \sum_{t=1}^{T} S_i\, v_{i,t,d}. \end{aligned} \tag{11}$$

Thus, submitted bidding prices affect the reward only through the resulting market-clearing outcomes; the true variable generation cost $\mathrm{TC}_i(\cdot)$ is used only to evaluate the realized cost and then profit. Since each generator maximizes its own profit rather than a common system-wide reward, the resulting POSG is general-sum.

State transition. After the delivery periods cleared at market step d, the terminal operating condition of each generator is carried into the next market step. Specifically, the initial commitment status and power output for the next day are

$$u^0_{i,d+1} = u_{i,T,d}, \qquad p^0_{i,d+1} = p_{i,T,d}, \tag{12}$$

while the residual minimum up- or downtime requirements $\tau^{\mathrm{up}}_{i,d+1}$ and $\tau^{\mathrm{dn}}_{i,d+1}$ are updated according to the terminal commitment trajectory. Hence, the next generator state

$$x_{i,d+1} = \left(u^0_{i,d+1}, p^0_{i,d+1}, \tau^{\mathrm{up}}_{i,d+1}, \tau^{\mathrm{dn}}_{i,d+1}\right) \tag{13}$$

depends on the commitment and dispatch outcomes produced by the joint bids at market step d.

The exogenous system state x^{sys}_{d+1}, including the next day-ahead forecasts of demand, evolves according to the underlying data process. The overall state transition is therefore

$$s_{d+1} \sim P(\cdot \mid s_d, a_d), \tag{14}$$

where the dependence on a_d arises through the cleared commitment and dispatch decisions carried into the next market step.

Agent objective. Each generator uses a policy $\sigma_i(a_{i,d} \mid o_{i,d})$ to choose its day-ahead bid and seeks to maximize its expected discounted profit over an episode of H market days,

$$J_i(\sigma_i, \sigma_{-i}) = \mathbb{E}\left[\sum_{d=1}^{H} \gamma^{d-1} r_{i,d}\right]. \tag{15}$$

where σ_{-i} denotes the joint policies of all other generators. Because market clearing depends on the joint bids of all generators, the expected return of agent i depends on both its own policy σ_i and the policies of the other agents σ_{-i}. Each agent therefore learns its bidding policy through repeated market interactions without observing competitors' private information or policies.

C.2 M2: Real-Time Balancing Market

The real-time balancing market consists of a sequence of auctions during delivery day D, with one auction for each real-time interval $\Delta^{\mathrm{rt}} = 30$ mins. It takes the commitment and schedules established by the day-ahead wholesale market as fixed and redispatches available resources against realized system demand. Only deviations from the day-ahead schedules are settled at real-time LMPs, while the day-ahead schedules remain settled at day-ahead LMPs. This section presents the real-time balancing market clearing model in subsection C.2.1, the two-settlement rule in subsection C.2.2, and the corresponding POSG in subsection C.2.3.

C.2.1 Market model

Let τ index the 30-minute real-time intervals and let $h(\tau)$ denote the corresponding hourly period in the day-ahead market. The day-ahead outcome enters the real-time market as fixed data through the commitment $u^{\mathrm{da}}_{i,h(\tau)}$, scheduled generation $q^{\mathrm{da}}_{i,h(\tau)}$, and day-ahead price $\mathrm{LMP}^{\mathrm{da}}_{n,h(\tau)}$. The realized net demand at bus n is $\tilde{d}_{n,\tau}$. Generators submit price–quantity bids with the same structure as in the day-ahead market, and one market clearing problem is solved for each real-time interval:

$$\min_{\{g,\, s\}} \sum_{i,k} \pi_{i,k,\tau}\, g_{i,k,\tau} + \mathrm{VOLL} \sum_{n} s_{n,\tau}, \tag{16a}$$

subject to

$$p_{i,\tau} = \underline{p}_i\, u^{\mathrm{da}}_{i,h(\tau)} + \sum_{k} g_{i,k,\tau}, \tag{16b}$$

$$0 \le g_{i,k,\tau} \le G_{i,k}\, u^{\mathrm{da}}_{i,h(\tau)}, \tag{16c}$$

$$P_{n,\tau} = \sum_{i:\mathrm{bus}(i)=n} p_{i,\tau} + s_{n,\tau} - \tilde{d}_{n,\tau}, \tag{16d}$$

$$\sum_{n} P_{n,\tau} = 0 \qquad (:\lambda_\tau), \tag{16e}$$

$$-F \le \mathrm{PTDF} \cdot P_\tau \le F \qquad (:\mu^+_\tau, \mu^-_\tau), \tag{16f}$$

$$0 \le s_{n,\tau} \le \max(\tilde{d}_{n,\tau}, 0) \quad (:\rho_{n,\tau}), \tag{16g}$$

$$-R^{\mathrm{dn}}_i \Delta^{\mathrm{rt}} - \bar{p}_i w^{\mathrm{da}}_{i,\tau} \le p_{i,\tau} - p_{i,\tau-1} \le R^{\mathrm{up}}_i \Delta^{\mathrm{rt}} + \underline{p}_i v^{\mathrm{da}}_{i,\tau}. \tag{16h}$$

The objective minimizes the as-bid redispatch cost and load-shedding penalty for each real-time interval Δ^{rt}. Unlike the day-ahead market, unit commitment is not re-optimized: $u^{\mathrm{da}}_{i,h(\tau)}$ is fixed by the day-ahead outcome, and the real-time market only adjusts the dispatch of committed units. The nodal balance and transmission constraints ensure that the resulting dispatch remains physically feasible under realized net demand. Consecutive real-time intervals are coupled through the ramping constraint, where $p_{i,\tau-1}$ is the realized dispatch from the previous interval. The start-up and shut-down indicators $v^{\mathrm{da}}_{i,\tau}$ and $w^{\mathrm{da}}_{i,\tau}$ are determined by changes in the mapped day-ahead commitment and are non-zero only when the commitment status changes between consecutive real-time intervals.

Constraint (16b) defines the real-time output of generator i. Because the commitment $u^{\mathrm{da}}_{i,h(\tau)}$ is fixed by the day-ahead market, only generators committed in the corresponding day-ahead period can produce in real time. Their output equals the minimum generation level plus the accepted quantities from the submitted bid segments. Constraint (16c) limits each accepted segment $g_{i,k,\tau}$ to its available quantity $G_{i,k}$. Constraint (16d) defines the net injection $P_{n,\tau}$ at bus n as local generation plus load shedding minus realized net demand. Constraint (16e) enforces system-wide supply–demand balance, while constraint (16f) keeps transmission flows within the line ratings. The associated dual variables λ_τ, μ^+_τ, and μ^-_τ are used to determine the real-time LMPs.

Constraint (16g) allows emergency load curtailment only up to the positive net demand at each bus. Hence, $s_{n,\tau} = 0$ when renewable generation exceeds local demand. The dual variable $\rho_{n,\tau}$ is associated with the upper bound on load shedding. Constraint (16h) limits changes in generator output between consecutive real-time intervals according to the upward and downward ramp rates.

Here, $p_{i,\tau-1}$ is the realized dispatch from the previous real-time interval. The fixed indicators $v^{\mathrm{da}}_{i,\tau}$ and $w^{\mathrm{da}}_{i,\tau}$ account for start-up and shut-down transitions implied by the day-ahead commitment.

Unlike the day-ahead market, the real-time market does not re-optimize unit commitment. The day-ahead commitment is fixed, and the real-time clearing adjusts only the dispatch quantities in response to realized system conditions. Consecutive real-time intervals are coupled through generator ramping.

C.2.2 Pricing and settlement

Real-time LMPs are computed from the dual variables of the real-time clearing problem using the same energy–congestion decomposition as in equation (3a):

$$\mathrm{LMP}^{\mathrm{rt}}_{n,\tau} = \lambda_\tau - \sum_l \left(\mu^+_{l,\tau} - \mu^-_{l,\tau}\right) \mathrm{PTDF}_{l,n} - \rho_{n,\tau}. \tag{17}$$

Unlike day-ahead LMPs, the real-time LMPs reflect realized demand and short-term operating constraints. They can therefore be more volatile and may become negative under local oversupply or binding downward-flexibility constraints.

The market follows a two-settlement rule. The day-ahead schedule remains settled at the day-ahead LMP, while only the deviation from that schedule is settled at the real-time LMP. For generator i in real-time interval τ,

$$\mathrm{revenue}^{\mathrm{da}}_{i,\tau} = \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)}\, q^{\mathrm{da}}_{i,h(\tau)}, \tag{18a}$$

$$\mathrm{revenue}^{\mathrm{rt}}_{i,\tau} = \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{rt}}_{\mathrm{bus}(i),\tau} \left(p_{i,\tau} - q^{\mathrm{da}}_{i,h(\tau)}\right), \tag{18b}$$

$$\mathrm{cost}_{i,\tau} = \Delta^{\mathrm{rt}}\, \mathrm{TC}_i(p_{i,\tau}) + \Delta^{\mathrm{rt}}\, \mathrm{NL}_i u^{\mathrm{da}}_{i,h(\tau)} + S_i v^{\mathrm{da}}_{i,\tau}, \tag{18c}$$

$$\mathrm{profit}_{i,\tau} = \mathrm{revenue}^{\mathrm{da}}_{i,\tau} + \mathrm{revenue}^{\mathrm{rt}}_{i,\tau} - \mathrm{cost}_{i,\tau}. \tag{18d}$$

In this context, a generator producing above its day-ahead schedule sells the positive deviation at the real-time LMP, while a generator producing below its schedule effectively buys back the shortfall at the real-time LMP. The day-ahead revenue, commitment status, and associated no-load and start-up costs are fixed by the day-ahead outcome and do not depend on the real-time bid. They are retained in the reward so that the reported profit represents the full economic outcome of delivery.

Remark on the day-ahead and real-time coupling. In practice, the two markets operate at different temporal scales: the day-ahead market is cleared on day $D-1$, while the real-time market is cleared at each interval of delivery day D. The commitment $u^{\mathrm{da}}_{i,h(\tau)}$, binding schedule $q^{\mathrm{da}}_{i,h(\tau)}$, and day-ahead price $\mathrm{LMP}^{\mathrm{da}}_{n,h(\tau)}$ are fixed parameters of the real-time clearing, while the real-time market prices deviations arising from uncertainty not captured by the day-ahead forecast.

C.2.3 Partially Observable Stochastic Game

The real-time balancing market can be also formulated as a general-sum partially observable stochastic game (POSG),

$$\mathcal{G}^{\mathrm{rt}} = \langle\, \mathcal{I},\, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}},\, P,\, \{r_i\}_{i\in\mathcal{I}},\, \gamma \,\rangle\,, \tag{19}$$

where $\mathcal{I}$ is the set of strategic generators. One episodic market step τ corresponds to one 30-minute real-time interval Δ^{rt}. For a 24-hour delivery day, the real-time market therefore contains $T^{\mathrm{rt}} = 48$ sequential clearing steps.

State. The state at interval τ is

$$s_\tau = \left(\, x^{\mathrm{sys}}_\tau,\, x^{\mathrm{da}}_\tau,\, \{x_{i,\tau}\}_{i\in\mathcal{I}} \,\right) \in \mathcal{S}, \tag{20}$$

where x^{sys}_τ contains the realized system conditions used for real-time clearing, i.e., the net bus demand $\tilde{d}_{n,\tau}$. The day-ahead component is

$$x^{\mathrm{da}}_\tau = \left\{\, u^{\mathrm{da}}_{i,h(\tau)},\, q^{\mathrm{da}}_{i,h(\tau)},\, \mathrm{LMP}^{\mathrm{da}}_{n,h(\tau)} \,\right\}_{i,n}, \tag{21}$$

which contains the fixed commitment $u^{\mathrm{da}}_{i,h(\tau)}$, binding generation schedules $q^{\mathrm{da}}_{i,h(\tau)}$, and nodal prices $\mathrm{LMP}^{\mathrm{da}}_{n,h(\tau)}$ from the corresponding day-ahead period. The operating state of generator i is

$$x_{i,\tau} = \left(p_{i,\tau-1},\, v^{\mathrm{da}}_{i,\tau},\, w^{\mathrm{da}}_{i,\tau} \right), \tag{22}$$

where $p_{i,\tau-1}$ is the realized dispatch from the previous real-time interval $\tau - 1$, and $v^{\mathrm{da}}_{i,\tau}$ and $w^{\mathrm{da}}_{i,\tau}$ are the start-up and shut-down indicators implied by the mapped day-ahead commitment, which are used in the ramping constraint (16h) to account for output changes associated with start-up and shut-down transitions. Network parameters, generator operating limits, and cost parameters are fixed environment parameters and are omitted from the dynamic state.

Observation. Each generator i observes its own operating information together with public market information. Its observation at real-time interval τ is

$$o_{i,\tau} = \left(p_{i,\tau-1},\, u^{\mathrm{da}}_{i,h(\tau)},\, q^{\mathrm{da}}_{i,h(\tau)},\, \mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)},\, x^{\mathrm{pub}}_{\tau} \right) = \Omega_i(s_\tau) \in \mathcal{O}_i, \tag{23}$$

where $p_{i,\tau-1}$ is the generator's realized dispatch in the previous real-time interval $\tau - 1$, $u^{\mathrm{da}}_{i,h(\tau)}$ and $q^{\mathrm{da}}_{i,h(\tau)}$ are its day-ahead commitment and schedule, and $\mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)}$ is the corresponding day-ahead LMP. The public real-time market information is denoted by x^{pub}_{τ}, which is the realized system conditions of net bus demand $x^{\mathrm{sys}}_{\tau} = \tilde{d}_{n,\tau}$. A generator does not observe the private parameters or contemporaneous bids of other generators. The market is therefore partially observable from the perspective of each participant.

Action. At each real-time interval, generator i submits a stepwise supply bid $\boldsymbol{\pi}_{i,\tau}$, obtained from a single markup $\alpha_{i,\tau}$ on its truthful offer $\pi^0_{i,1} \le \cdots \le \pi^0_{i,K}$, in which each segment is priced at the unit's true marginal cost MC_i,

$$a_{i,\tau} = \alpha_{i,\tau} \in \mathcal{A}_i = [1, \bar{\alpha}], \qquad \pi_{i,k,\tau} = \alpha_{i,\tau}\, \pi^0_{i,k}, \quad k = 1, \ldots, K, \tag{24}$$

where $\bar{\alpha}$ is the largest admissible markup, subject to the admissible bid-price bounds and the monotonicity condition

$$\pi_{i,1,\tau} \le \pi_{i,2,\tau} \le \cdots \le \pi_{i,K,\tau}. \tag{25}$$

The segment quantities $G_{i,k}$ are fixed, so the strategic action determines the bid prices associated with these quantities. The joint action is $a_\tau = (a_{i,\tau})_{i \in \mathcal{I}}$.

Market clearing and reward. Given the current state and joint bids, the market-clearing mechanism

$$z_\tau = \mathcal{C}^{\mathrm{rt}}(s_\tau, a_\tau) \tag{26}$$

solves optimization (16) and returns the realized dispatch $p_{i,\tau}$, accepted bid quantities $g_{i,k,\tau}$, and real-time LMPs $\mathrm{LMP}^{\mathrm{rt}}_{n,\tau}$. Each generator receives its realized profit as the stage reward,

$$\begin{aligned} r_{i,\tau} &= \mathrm{profit}_{i,\tau} = \mathrm{revenue}^{\mathrm{da}}_{i,\tau} + \mathrm{revenue}^{\mathrm{rt}}_{i,\tau} - \mathrm{cost}_{i,\tau} \\ &= \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)}\, q^{\mathrm{da}}_{i,h(\tau)} + \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{rt}}_{\mathrm{bus}(i),\tau} \left(p_{i,\tau} - q^{\mathrm{da}}_{i,h(\tau)} \right) \\ &\quad - \left(\Delta^{\mathrm{rt}}\, \mathrm{TC}_i(p_{i,\tau}) + \Delta^{\mathrm{rt}}\, \mathrm{NL}_i u^{\mathrm{da}}_{i,h(\tau)} + S_i v^{\mathrm{da}}_{i,\tau} \right). \end{aligned} \tag{27}$$

where $\mathrm{profit}_{i,\tau}$ is defined by the two-settlement rule in equation (18). The reward therefore accounts for the fixed day-ahead position, the real-time settlement of deviations, and the realized generation cost.

State transition. The clearing result determines the operating state entering the next real-time interval. In particular, the realized dispatch $p_{i,\tau}$ becomes the previous-period dispatch in interval $\tau + 1$, so that

$$x_{i,\tau+1} = \left(p_{i,\tau}, v^{\mathrm{da}}_{i,\tau+1}, w^{\mathrm{da}}_{i,\tau+1} \right). \tag{28}$$

Thus, the dispatch determined at interval τ enters the ramping constraint of interval $\tau + 1$. The corresponding day-ahead commitment, schedule, and price are updated deterministically according to $h(\tau+1)$, while realized net demand evolves according to the exogenous data process. The overall state transition is therefore

$$s_{\tau+1} \sim P\left(\cdot \mid s_\tau, a_\tau \right). \tag{29}$$

Agent objective. Each generator uses a policy $\sigma_i(a_{i,\tau} \mid o_{i,\tau})$ to select its real-time bid. Over a horizon of H real-time intervals, generator i seeks to maximize its expected discounted profit,

$$J_i(\sigma_i, \sigma_{-i}) = \mathbb{E}\left[\sum_{\tau=1}^{H} \gamma^{\tau-1} r_{i,\tau}\right], \tag{30}$$

where σ_{-i} denotes the joint policies of the other generators. Because both dispatch and real-time prices are determined jointly by all submitted bids, the return of each generator depends on both its own bidding policy and those of the other participants. Agents therefore learn their real-time bidding policies through repeated market interactions without observing competitors' private information or policies.

C.3 M3: Ancillary Services Market

The ancillary services market procures operating reserve, where generation capacity is held available to respond to system contingencies. Unlike the energy market, it primarily trades availability: a provider is paid for maintaining reserve capacity that can be activated when needed but may never be called. In some market designs, operating reserve is co-optimized with real-time energy, so energy dispatch and reserve procurement are determined jointly for the same resources subject to shared operating constraints. This section presents the ancillary services market clearing model in subsection C.3.1, the pricing and settlement rules in subsection C.3.2, and the corresponding POSG in subsection C.3.3.

C.3.1 Market model

The market contains $N^{\text{prod}} = 2$ reserve products distinguished by their response times: a fast product with $\theta_1 = 10$ mins and a slow product with $\theta_2 = 30$ mins. Before bids are submitted, the system operator publishes the reserve requirement for each product, which can be modeled as a fraction of forecast system demand,

$$d^{\text{res}}_{j,\tau} = \beta_j \sum_n d^{\text{forecast}}_{n,\tau}, \tag{31}$$

where $d^{\text{forecast}}_{n,\tau}$ is the published demand forecast at bus n and β_j is the requirement fraction for reserve product j.

For each real-time interval, generator i submits the energy offer described in section C.2.1, together with a reserve offer price $\pi^{\text{res}}_{i,j,\tau} \geq 0$ for each product j, expressed in \$/MW-h. The energy and reserve markets are cleared jointly as

$$\min_{\{g,\, r,\, s,\, s^{\text{res}}\}} \sum_{i,k} \pi_{i,k,\tau} g_{i,k,\tau} + \sum_{i,j} \pi^{\text{res}}_{i,j,\tau} r_{i,j,\tau} + \text{VOLL} \sum_n s_{n,\tau} + \text{VOLR} \sum_j s^{\text{res}}_{j,\tau}, \tag{32a}$$

subject to

$$p_{i,\tau} = \underline{p}_i \, u^{\text{da}}_{i,h(\tau)} + \sum_k g_{i,k,\tau}, \tag{32b}$$

$$0 \leq g_{i,k,\tau} \leq G_{i,k} \, u^{\text{da}}_{i,h(\tau)}, \tag{32c}$$

$$P_{n,\tau} = \sum_{i:\text{bus}(i)=n} p_{i,\tau} + s_{n,\tau} - \tilde{d}_{n,\tau}, \tag{32d}$$

$$\sum_n P_{n,\tau} = 0 \qquad (:\lambda_\tau), \tag{32e}$$

$$-F \leq \text{PTDF} \cdot P_\tau \leq F \qquad (:\mu^+_\tau,\, \mu^-_\tau), \tag{32f}$$

$$0 \leq s_{n,\tau} \leq \max(\tilde{d}_{n,\tau}, 0) \qquad (:\rho_{n,\tau}), \tag{32g}$$

$$-R^{\text{dn}}_i \Delta^{\text{rt}} - \bar{p}_i w^{\text{da}}_{i,\tau} \leq p_{i,\tau} - p_{i,\tau-1} \leq R^{\text{up}}_i \Delta^{\text{rt}} + \underline{p}_i v^{\text{da}}_{i,\tau}, \tag{32h}$$

$$0 \leq r_{i,j,\tau} \leq R^{\text{up}}_i \theta_j, \tag{32i}$$

$$p_{i,\tau} + \sum_j r_{i,j,\tau} \leq \bar{p}_i \, u^{\text{da}}_{i,h(\tau)} \qquad (:\nu^{\text{cap}}_{i,\tau}), \tag{32j}$$

$$d^{\mathrm{res}}_{j,\tau} - \sum_i r_{i,j,\tau} - s^{\mathrm{res}}_{j,\tau} \le 0 \qquad (:\lambda^{\mathrm{res}}_{j,\tau}), \tag{32k}$$

$$0 \le s^{\mathrm{res}}_{j,\tau} \le d^{\mathrm{res}}_{j,\tau}. \tag{32l}$$

The objective in equation (32a) minimizes the joint as-bid cost of energy dispatch and reserve procurement for each real-time interval Δ^{rt}, together with penalties for unserved energy and unmet reserve requirements. The first term represents the accepted energy offers, and the second term represents payments associated with reserve capacity held available for contingency response. The penalties VOLL and VOLR are assigned to load shedding and reserve shortage, respectively. We set VOLR $<$ VOLL so that, when available capacity is insufficient to satisfy both requirements, the model permits a reserve shortage before shedding firm load.

Constraints (32b)–(32h) describe the energy dispatch and follow the real-time balancing model in section C.2.1. Constraint (32b) defines the output of generator i as its committed minimum output plus the accepted quantities from its energy-offer segments. Constraint (32c) bounds each accepted segment by its available quantity. Constraints (32d) and (32e) define nodal injections and enforce system-wide supply–demand balance, respectively. Constraint (32f) keeps transmission flows within the line ratings, and (32g) allows emergency load shedding when available generation is insufficient. Constraint (32h) limits the change in generator output between consecutive real-time intervals.

Constraints (32i)–(32l) introduce the reserve products. Constraint (32i) limits the reserve award $r_{i,j,\tau}$ to the amount of additional output that generator i can deliver within the required response time θ_j. Therefore, a product with a longer response time allows a generator to provide more reserve capacity for a given ramp rate. Constraint (32j) couples energy and reserve procurement. The scheduled energy output and the total reserve capacity awarded to a generator must jointly remain below its committed capacity. Consequently, capacity used to produce energy cannot simultaneously be sold as reserve, creating an opportunity-cost trade-off between the two products. The dual variable $\nu^{\mathrm{cap}}_{i,\tau}$ represents the marginal value of additional available capacity of generator i. Finally, constraint (32k) requires the total procured capacity for reserve product j to satisfy the published requirement $d^{\mathrm{res}}_{j,\tau}$. The slack variable $s^{\mathrm{res}}_{j,\tau}$ represents any unmet reserve requirement and is bounded by constraint (32l). The corresponding dual variable $\lambda^{\mathrm{res}}_{j,\tau}$ gives the marginal clearing price of reserve product j.

C.3.2 Pricing and settlement

Energy prices follow the same LMP formulation as in equation (17), evaluated using the dual variables of the joint energy–reserve clearing problem (32). The reserve price of product j is determined by the dual variable $\lambda^{\mathrm{res}}_{j,\tau}$ associated with the reserve requirement constraint (32k). It represents the marginal cost of increasing the system-wide requirement for reserve product j by one additional unit of capacity.

A key feature of co-optimization is that the opportunity cost of reserving generation capacity is determined within the clearing problem. When the response-time constraint is non-binding, the marginal cost of providing reserve consists of two components: the submitted reserve offer and the opportunity cost of using generation capacity for reserve rather than energy. The latter is represented by the shadow price $\nu^{\mathrm{cap}}_{i,\tau}$ of the shared capacity constraint. Hence, for a marginal reserve provider,

$$\lambda^{\mathrm{res}}_{j,\tau} = \pi^{\mathrm{res}}_{i,j,\tau} + \nu^{\mathrm{cap}}_{i,\tau}. \tag{33}$$

If the same generator is also marginal in the energy market and no additional operating constraint is binding, the capacity shadow value corresponds to the energy margin,

$$\nu^{\mathrm{cap}}_{i,\tau} = \mathrm{LMP}^{\mathrm{rt}}_{\mathrm{bus}(i),\tau} - \pi_{i,k,\tau}, \tag{34}$$

and therefore

$$\lambda^{\mathrm{res}}_{j,\tau} = \pi^{\mathrm{res}}_{i,j,\tau} + \left(\mathrm{LMP}^{\mathrm{rt}}_{\mathrm{bus}(i),\tau} - \pi_{i,k,\tau}\right). \tag{35}$$

Thus, the reserve price accounts not only for the submitted reserve offer but also for the value of energy production that the marginal provider gives up when capacity is allocated to reserve.

Energy is settled according to the two-settlement rule in equation (18). Reserve capacity is settled separately at the clearing price of each product. For generator i,

$$\text{revenue}^{\text{res}}_{i,\tau} = \Delta^{\text{rt}} \sum_{j} \lambda^{\text{res}}_{j,\tau}\, r_{i,j,\tau}, \tag{36}$$

and its total profit for the interval is

$$\text{profit}_{i,\tau} = \text{revenue}^{\text{da}}_{i,\tau} + \text{revenue}^{\text{rt}}_{i,\tau} + \text{revenue}^{\text{res}}_{i,\tau} - \text{cost}_{i,\tau}. \tag{37}$$

where

$$\text{revenue}^{\text{da}}_{i,\tau} = \Delta^{\text{rt}}\, \text{LMP}^{\text{da}}_{\text{bus}(i),h(\tau)}\, q^{\text{da}}_{i,h(\tau)}, \tag{38}$$

$$\text{revenue}^{\text{rt}}_{i,\tau} = \Delta^{\text{rt}}\, \text{LMP}^{\text{rt}}_{\text{bus}(i),\tau} \left(p_{i,\tau} - q^{\text{da}}_{i,h(\tau)} \right), \tag{39}$$

$$\text{cost}_{i,\tau} = \Delta^{\text{rt}}\, \text{TC}_i(p_{i,\tau}) + \Delta^{\text{rt}}\, \text{NL}_i u^{\text{da}}_{i,h(\tau)} + S_i v^{\text{da}}_{i,\tau}. \tag{40}$$

Reserve availability itself does not consume fuel, so no additional generation cost is assigned to a reserve award unless the reserve is activated. Every cleared MW of the same reserve product is paid the same product price, so a provider whose reserve offer is below the clearing price earns an inframarginal margin.

Remark on energy–reserve co-optimization. Constraint (32j) couples energy dispatch and reserve procurement because both compete for the same generation capacity. Increasing the reserve award can reduce the capacity available for energy production, while increasing energy output reduces the headroom available for reserve. The joint clearing therefore allows an agent's energy and reserve offers to interact through this shared capacity constraint.

The ancillary services market is implemented as a separate benchmark task that extends the real-time balancing market with reserve products. The day-ahead commitment, schedule, and prices enter as fixed inputs, while real-time energy and reserve are cleared jointly. Setting $d^{\text{res}}_{j,\tau} = 0$ for all reserve products removes the reserve layer and recovers the real-time balancing model of section C.2.

C.3.3 Partially Observable Stochastic Game

The ancillary services market is formulated as a general-sum POSG,

$$\mathcal{G}^{\text{as}} = \langle\, \mathcal{I},\, \mathcal{S},\, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}},\, P,\, \{r_i\}_{i\in\mathcal{I}},\, \gamma \,\rangle, \tag{41}$$

where $\mathcal{I}$ denotes the set of strategic generators. One episodic market step corresponds to one 30-minute real-time interval τ, during which energy and reserve are cleared jointly.

State. The state at interval τ is

$$s_\tau = \left(x^{\text{sys}}_\tau,\, x^{\text{da}}_\tau,\, x^{\text{res}}_\tau,\, \{x_{i,\tau}\}_{i\in\mathcal{I}} \right) \in \mathcal{S}, \tag{42}$$

where x^{sys}_τ contains the realized system conditions of demand $\tilde{d}_{n,\tau}$ used in the joint energy–reserve clearing. The day-ahead component is

$$x^{\text{da}}_\tau = \left\{ u^{\text{da}}_{i,h(\tau)},\, q^{\text{da}}_{i,h(\tau)},\, \text{LMP}^{\text{da}}_{n,h(\tau)} \right\}_{i,n}, \tag{43}$$

which contains the fixed commitment, binding generation schedules, and nodal prices from the corresponding day-ahead period. The reserve component is

$$x^{\text{res}}_\tau = \left(d^{\text{res}}_{1,\tau}, \ldots, d^{\text{res}}_{N^{\text{prod}},\tau} \right), \tag{44}$$

where $d^{\text{res}}_{j,\tau}$ is the system-wide requirement for reserve product j published before bidding. The operating state of generator i is

$$x_{i,\tau} = \left(p_{i,\tau-1},\, v^{\text{da}}_{i,\tau},\, w^{\text{da}}_{i,\tau} \right), \tag{45}$$

where $p_{i,\tau-1}$ is the realized dispatch from the previous real-time interval, and $v^{\text{da}}_{i,\tau}$ and $w^{\text{da}}_{i,\tau}$ are the start-up and shut-down indicators implied by the mapped day-ahead commitment. Network parameters, reserve-product response times, generator operating limits, and cost parameters are fixed environment parameters and are omitted from the dynamic state.

Observation. Each generator observes its own operating information together with the public energy and reserve market information available before bidding. Its observation at interval τ is

$$o_{i,\tau} = \Big(p_{i,\tau-1},\, u^{\mathrm{da}}_{i,h(\tau)},\, q^{\mathrm{da}}_{i,h(\tau)},\, \mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)},\, x^{\mathrm{sys}}_{\tau},\, x^{\mathrm{res}}_{\tau} \Big) = \Omega_i(s_\tau) \in \mathcal{O}_i. \tag{46}$$

Thus, the generator observes the realized net-demand conditions and the published reserve requirements in addition to its own day-ahead position and previous dispatch. It does not observe competitors' private cost information or their contemporaneous energy and reserve offers. The market is therefore partially observable from the perspective of each participant.

Action. At each interval, generator i jointly submits an energy offer and one reserve offer price for each reserve product. Its action is

$$a_{i,\tau} = \big(\boldsymbol{\pi}^{\mathrm{en}}_{i,\tau},\, \boldsymbol{\pi}^{\mathrm{res}}_{i,\tau} \big) \in \mathcal{A}_i, \tag{47}$$

where

$$\boldsymbol{\pi}^{\mathrm{en}}_{i,\tau} = (\pi_{i,1,\tau}, \ldots, \pi_{i,K,\tau}), \qquad \pi_{i,k,\tau} = \alpha_{i,\tau}\, \pi^0_{i,k}, \tag{48}$$

is the stepwise energy-offer price vector, obtained from a single markup $\alpha_{i,\tau} \in [1, \bar{\alpha}]$ on the truthful offer $\pi^0_{i,1} \le \cdots \le \pi^0_{i,K}$ as in the real-time balancing market, and

$$\boldsymbol{\pi}^{\mathrm{res}}_{i,\tau} = \Big(\pi^{\mathrm{res}}_{i,1,\tau}, \ldots, \pi^{\mathrm{res}}_{i,N^{\mathrm{prod}},\tau} \Big) \tag{49}$$

contains the reserve offer prices. The energy offer satisfies

$$\pi_{i,1,\tau} \le \pi_{i,2,\tau} \le \cdots \le \pi_{i,K,\tau}, \tag{50}$$

while $0 \le \pi^{\mathrm{res}}_{i,j,\tau} \le \mathrm{VOLR}$ for each reserve product j. The energy segment quantities $G_{i,k}$ are fixed, and no reserve quantity is submitted by the agent; the reserve awards are determined by the co-optimization. The joint action is $a_\tau = (a_{i,\tau})_{i \in \mathcal{I}}$.

Market clearing and reward. Given the current state and joint energy–reserve offers, the market-clearing mechanism

$$z_\tau = \mathcal{C}^{\mathrm{as}}(s_\tau, a_\tau) \tag{51}$$

solves the co-optimization problem (32). The clearing outcome includes the energy dispatch $p_{i,\tau}$, reserve awards $r_{i,j,\tau}$, real-time LMPs $\mathrm{LMP}^{\mathrm{rt}}_{n,\tau}$, and reserve prices $\lambda^{\mathrm{res}}_{j,\tau}$.

Each generator receives its total realized profit as the stage reward,

$$\begin{aligned} r_{i,\tau} &= \mathrm{profit}_{i,\tau} = \mathrm{revenue}^{\mathrm{da}}_{i,\tau} + \mathrm{revenue}^{\mathrm{rt}}_{i,\tau} + \mathrm{revenue}^{\mathrm{res}}_{i,\tau} - \mathrm{cost}_{i,\tau} \\ &= \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{da}}_{\mathrm{bus}(i),h(\tau)}\, q^{\mathrm{da}}_{i,h(\tau)} + \Delta^{\mathrm{rt}}\, \mathrm{LMP}^{\mathrm{rt}}_{\mathrm{bus}(i),\tau} \Big(p_{i,\tau} - q^{\mathrm{da}}_{i,h(\tau)} \Big) \\ &\quad + \Delta^{\mathrm{rt}} \sum_j \lambda^{\mathrm{res}}_{j,\tau}\, r_{i,j,\tau} - \big(\Delta^{\mathrm{rt}}\, \mathrm{TC}_i(p_{i,\tau}) + \Delta^{\mathrm{rt}}\, \mathrm{NL}_i u^{\mathrm{da}}_{i,h(\tau)} + S_i v^{\mathrm{da}}_{i,\tau} \big). \end{aligned} \tag{52}$$

as defined in section C.3.2. The reward therefore captures both energy-market profit and reserve-capacity revenue.

State transition. The clearing outcome determines the operating state entering the next real-time interval. In particular,

$$x_{i,\tau+1} = \big(p_{i,\tau}, v^{\mathrm{da}}_{i,\tau+1}, w^{\mathrm{da}}_{i,\tau+1} \big), \tag{53}$$

so the realized dispatch $p_{i,\tau}$ enters the ramping constraint of the next interval. The corresponding day-ahead commitment, schedule, and price are updated deterministically according to $h(\tau+1)$, while the realized net demand and reserve requirements evolve according to their exogenous data processes. The overall transition is therefore

$$s_{\tau+1} \sim P(\cdot \mid s_\tau, a_\tau). \tag{54}$$

Agent objective. Each generator uses a policy $\sigma_i(a_{i,\tau} \mid o_{i,\tau})$ to jointly determine its energy and reserve offers. Over a horizon of H real-time intervals, generator i maximizes its expected discounted profit,

$$J_i(\sigma_i, \sigma_{-i}) = \mathbb{E}\left[\sum_{\tau=1}^{H} \gamma^{\tau-1} r_{i,\tau}\right], \tag{55}$$

where σ_{-i} denotes the joint policies of the other generators. Because energy dispatch, reserve awards, and both energy and reserve prices are jointly determined by all submitted offers, the return of each generator depends on both its own bidding policy and those of other participants. Agents therefore learn how to allocate their available capacity between energy and reserve through repeated market interactions without observing competitors' private information or policies.

C.4 M4: Peer-to-Peer Local Energy Market

Rooftop photovoltaic generation and behind-the-meter storage have enabled many consumers to become prosumers that can both consume and supply electricity. When surplus energy is exported directly to the grid, it is typically settled at an export price such as feed-in-tariff, which is below the retail tariff paid for grid consumption. A peer-to-peer (P2P) local energy market allows neighboring participants to trade their surplus and deficit locally at prices between these two outside options, creating the possibility of mutual benefits for buyers and sellers. Unlike the day-ahead and real-time markets in sections C.1 and C.2, which operate at the transmission level, the P2P market considered here operates at the end-use level and facilitates local energy exchange among prosumers.

The P2P market uses 15-minute trading intervals. Before each auction, the net energy position of every participant is determined from its local generation, consumption, and storage operation. Hence, a participant knows the quantity it has available to buy or sell before submitting its bid, and its strategic decisions are its battery power and the submitted price. A local market operator collects the bids and offers, clears the auction, and settles the resulting transactions. The market does not issue physical dispatch instructions: the participants' net quantities are fixed before clearing, so the auction determines which quantities are matched and at what price rather than how the underlying devices are operated.

In our model, the P2P local energy market is implemented as a periodic double auction among a fixed population of N^{agent} prosumers, each equipped with rooftop photovoltaic generation, an inelastic local load, and a battery. In each trading interval, a participant appears on one side of the market according to its net energy position: a surplus is offered for sale, whereas a deficit is submitted as demand. The local market operator collects one price–quantity pair from each participant and constructs aggregate supply and demand curves. The market clears at their intersection, with all matched buyers and sellers settling at a common market clearing price. Any unmatched deficit is supplied by the upstream grid at the retail tariff, while any unmatched surplus is exported to the grid at the export price. This section presents the market clearing model in subsection C.4.1, the pricing and settlement rules in subsection C.4.2, and the corresponding POSG in subsection C.4.3.

C.4.1 Market model

Let τ index the 15-minute P2P trading intervals, with $\Delta^{\text{p2p}} = 0.25\ \text{h}$. Before each auction, the local generation, consumption, and battery operation of participant i determine its net power position,

$$p^{\text{net}}_{i,\tau} = -p^{\text{pv}}_{i,\tau} + p^{\text{ch}}_{i,\tau} + d_{i,\tau} - p^{\text{dis}}_{i,\tau}, \tag{56a}$$

and the corresponding quantities available for P2P trading are

$$q^{\text{sell}}_{i,\tau} = \Delta^{\text{p2p}} \max\!\left(-p^{\text{net}}_{i,\tau}, 0\right), \qquad q^{\text{buy}}_{i,\tau} = \Delta^{\text{p2p}} \max\!\left(p^{\text{net}}_{i,\tau}, 0\right). \tag{56b}$$

Here, $p^{\text{pv}}_{i,\tau}$ and $d_{i,\tau}$ denote photovoltaic generation and local demand, respectively. $p^{\text{dis}}_{i,\tau}$ and $p^{\text{ch}}_{i,\tau}$ denote battery discharging and charging power, respectively. Only one of $q^{\text{sell}}_{i,\tau}$ and $q^{\text{buy}}_{i,\tau}$ can be positive. A participant with surplus energy is therefore a seller, while one with an energy deficit is a buyer.

The battery is the only local device that carries a physical state between trading intervals. Its state of charge evolves according to

$$\mathrm{soc}_{i,\tau+1} = \mathrm{soc}_{i,\tau} + \frac{\Delta^{\mathrm{p2p}}}{E_i}\left(\eta_i^{\mathrm{ch}} p_{i,\tau}^{\mathrm{ch}} - \frac{p_{i,\tau}^{\mathrm{dis}}}{\eta_i^{\mathrm{dis}}}\right), \tag{57}$$

where E_i is the battery energy capacity, and η_i^{ch} and η_i^{dis} are the charging and discharging loss efficiencies. The feasible charging and discharging powers are bounded by

$$0 \le p_{i,\tau}^{\mathrm{dis}} \le \min\left(\bar{p}_i^{\mathrm{bat}}, \frac{(\mathrm{soc}_{i,\tau} - \underline{\mathrm{soc}}_i)\, E_i \eta_i^{\mathrm{dis}}}{\Delta^{\mathrm{p2p}}}\right), \tag{58}$$

and

$$0 \le p_{i,\tau}^{\mathrm{ch}} \le \min\left(\bar{p}_i^{\mathrm{bat}}, \frac{(\overline{\mathrm{soc}}_i - \mathrm{soc}_{i,\tau})\, E_i}{\eta_i^{\mathrm{ch}} \Delta^{\mathrm{p2p}}}\right), \tag{59}$$

where $\bar{p}_i^{\mathrm{bat}}$ is the battery power rating and $[\underline{\mathrm{soc}}_i, \overline{\mathrm{soc}}_i]$ is the admissible state-of-charge range. Charging and discharging are mutually exclusive within each trading interval.

Battery operation is chosen by the participant before the P2P auction. Hence, $p_{i,\tau}^{\mathrm{net}}$ and the corresponding trading quantity are fixed when the participant submits its bid, and within the auction only the submitted price is a strategic bidding decision.

The seller and buyer sets in interval τ are

$$\mathcal{S}_\tau = \left\{i : q_{i,\tau}^{\mathrm{sell}} > 0\right\}, \qquad \mathcal{B}_\tau = \left\{i : q_{i,\tau}^{\mathrm{buy}} > 0\right\}. \tag{60}$$

Each participant submits one price $\pi_{i,\tau}$: an ask if $i \in \mathcal{S}_\tau$ and a bid if $i \in \mathcal{B}_\tau$. Submitted prices are restricted by the outside options provided by the upstream grid,

$$\pi^{\mathrm{exp}} \le \pi_{i,\tau} \le \pi^{\mathrm{ret}}, \tag{61}$$

where π^{ret} is the retail tariff for grid imports and π^{exp} is the export price for grid injections, with $\pi^{\mathrm{exp}} < \pi^{\mathrm{ret}}$. The submitted quantity is fixed by the participant's net position, so $\pi_{i,\tau}$ is the only strategic variable in the auction.

The local market operator clears the market through a periodic double auction. Seller asks are sorted in ascending order of price, while buyer bids are sorted in descending order of price, with ties broken by participant index. Let $\sigma_\tau^{\mathrm{s}}(j)$ denote the participant occupying position j in the sorted seller order and $\sigma_\tau^{\mathrm{b}}(j)$ the participant occupying position j in the sorted buyer order. The associated cumulative quantities are

$$Q_{j,\tau}^{\mathrm{sell}} = \sum_{\ell=1}^{j} q_{\sigma_\tau^{\mathrm{s}}(\ell),\tau}^{\mathrm{sell}}, \qquad Q_{j,\tau}^{\mathrm{buy}} = \sum_{\ell=1}^{j} q_{\sigma_\tau^{\mathrm{b}}(\ell),\tau}^{\mathrm{buy}}. \tag{62}$$

The total quantities available on the two sides are

$$Q_\tau^{\mathrm{sell}} = \sum_{i\in\mathcal{S}_\tau} q_{i,\tau}^{\mathrm{sell}}, \qquad Q_\tau^{\mathrm{buy}} = \sum_{i\in\mathcal{B}_\tau} q_{i,\tau}^{\mathrm{buy}}. \tag{63}$$

The sorted submissions define the aggregate supply curve $S_\tau(x)$ and aggregate demand curve $D_\tau(x)$. The supply curve returns the ask price associated with the x-th unit of cumulative supply, while the demand curve returns the bid price associated with the x-th unit of cumulative demand. Both are step functions whose breakpoints are given by the cumulative quantities in equation (62).

Let $\mathcal{X}_\tau$ denote the union of the supply and demand breakpoints, together with zero. The cleared P2P trading volume is

$$Q_\tau^\star = \max\left\{x \in \mathcal{X}_\tau : x \le \min\left(Q_\tau^{\mathrm{sell}}, Q_\tau^{\mathrm{buy}}\right),\ D_\tau(x) \ge S_\tau(x)\right\}. \tag{64}$$

Thus, trading continues while the marginal buyer's bid is no lower than the marginal seller's ask. If no bid meets an ask, $Q_\tau^\star = 0$.

Individual awards follow the sorted orders. For the seller occupying position j,

$$q_{\sigma_\tau^{\mathrm{s}}(j),\tau}^{\mathrm{sell}\star} = \min\left[\max\left(Q_\tau^\star - Q_{j-1,\tau}^{\mathrm{sell}}, 0\right), q_{\sigma_\tau^{\mathrm{s}}(j),\tau}^{\mathrm{sell}}\right], \tag{65}$$

with $Q^{\text{sell}}_{0,\tau} = 0$. Similarly, the award to the buyer occupying position j is

$$q^{\text{buy}\star}_{\sigma^{\text{b}}_\tau(j),\tau} = \min\left[\max\left(Q^\star_\tau - Q^{\text{buy}}_{j-1,\tau}, 0\right), q^{\text{buy}}_{\sigma^{\text{b}}_\tau(j),\tau}\right], \tag{66}$$

with $Q^{\text{buy}}_{0,\tau} = 0$.

Equations (65) and (66) fill the most competitive submissions first: lower-priced sellers and higher-priced buyers receive priority. Since $Q^\star_\tau$ coincides with a cumulative-quantity breakpoint on at least one side of the market, all accepted participants on that side are either fully cleared or rejected. On the opposite side, only the marginal participant may be partially cleared.

C.4.2 Pricing and settlement

Let $S^\star_\tau$ and $D^\star_\tau$ denote the marginal accepted ask and bid, respectively, and let S^+_τ and D^+_τ denote the first rejected ask and bid beyond the cleared volume $Q^\star_\tau$. A uniform clearing price must be high enough for all accepted sellers and low enough for all accepted buyers, while remaining unattractive to rejected submissions. This defines the feasible price interval

$$\underline{\lambda}_\tau = \max\left(S^\star_\tau, D^+_\tau\right), \qquad \bar{\lambda}_\tau = \min\left(D^\star_\tau, S^+_\tau\right), \qquad \lambda^{\text{loc}}_\tau = \frac{1}{2}\left(\underline{\lambda}_\tau + \bar{\lambda}_\tau\right). \tag{67}$$

The local market therefore publishes the midpoint of the admissible price interval. When an adjacent accepted or rejected submission does not exist because one side of the market is exhausted, the export price π^{exp} and retail tariff π^{ret} provide the corresponding lower and upper price bounds. Consequently,

$$\pi^{\text{exp}} \le \lambda^{\text{loc}}_\tau \le \pi^{\text{ret}}. \tag{68}$$

Thus, a matched seller receives no less than the grid export price and a matched buyer pays no more than the retail tariff. If a marginal participant is only partially cleared, the admissible interval collapses to that participant's submitted price.

P2P awards are financially binding, while any unmatched quantity is settled with the upstream grid. Define the residual export and import quantities as

$$q^{\text{ex}}_{i,\tau} = q^{\text{sell}}_{i,\tau} - q^{\text{sell}\star}_{i,\tau}, \qquad q^{\text{im}}_{i,\tau} = q^{\text{buy}}_{i,\tau} - q^{\text{buy}\star}_{i,\tau}. \tag{69}$$

The settlement of participant i is then

$$\text{revenue}_{i,\tau} = \lambda^{\text{loc}}_\tau q^{\text{sell}\star}_{i,\tau} + \pi^{\text{exp}} q^{\text{ex}}_{i,\tau}, \tag{70a}$$

$$\text{cost}_{i,\tau} = \lambda^{\text{loc}}_\tau q^{\text{buy}\star}_{i,\tau} + \pi^{\text{ret}} q^{\text{im}}_{i,\tau} + c^{\text{deg}}_i \Delta^{\text{p2p}}\left(p^{\text{ch}}_{i,\tau} + p^{\text{dis}}_{i,\tau}\right), \tag{70b}$$

$$\text{profit}_{i,\tau} = \text{revenue}_{i,\tau} - \text{cost}_{i,\tau}. \tag{70c}$$

Here, c^{deg}_i is the battery degradation cost in EUR/MWh of throughput. Photovoltaic generation is assumed to have zero marginal cost, while local demand is inelastic. A participant with a net energy deficit may therefore have negative profit, corresponding to a net electricity bill; maximizing profit in this case is equivalent to minimizing that bill. Because all matched buyers and sellers settle at the same local price, P2P payments are budget balanced within the local market, excluding transactions with the upstream grid and any network charges not modeled here.

Remark on the P2P local market. The P2P market provides an intermediate trading opportunity between the two grid settlement prices. Sellers can receive more than the export price, while buyers can pay less than the retail tariff when mutually acceptable bids are matched. The market can also increase the share of locally produced renewable energy consumed within the local network. It therefore complements, rather than replaces, upstream electricity markets and grid transactions.

C.4.3 Partially Observable Stochastic Game

The P2P local energy market is formulated as a general-sum POSG,

$$\mathcal{G}^{\text{p2p}} = \langle\, \mathcal{I},\, \mathcal{S},\, \{\mathcal{O}_i\}_{i\in\mathcal{I}},\, \{\mathcal{A}_i\}_{i\in\mathcal{I}},\, P,\, \{r_i\}_{i\in\mathcal{I}},\, \gamma \,\rangle, \tag{71}$$

where $\mathcal{I}$ is the fixed set of N^{agent} prosumers. One game step corresponds to one 15-minute P2P trading interval τ.

State. The state at interval τ is

$$s_\tau = \left(\{x_{i,\tau}\}_{i\in\mathcal{I}}, \pi^{\mathrm{exp}}, \pi^{\mathrm{ret}} \right) \in \mathcal{S}, \tag{72}$$

where the local physical state of prosumer i is

$$x_{i,\tau} = \left(\mathrm{soc}_{i,\tau}, p^{\mathrm{pv}}_{i,\tau}, d_{i,\tau}, p^{\mathrm{ch}}_{i,\tau}, p^{\mathrm{dis}}_{i,\tau} \right). \tag{73}$$

These variables determine the participant's net power position $p^{\mathrm{net}}_{i,\tau}$ and therefore its available selling or buying quan tity through equation 56b. The export price π^{exp} and retail tariff π^{ret} define the outside settlement options with the upstream grid.

Observation. Each participant observes its own physical state and net trading position, together with the public grid settlement prices. Its observation is

$$o_{i,\tau} = \left(\mathrm{soc}_{i,\tau}, p^{\mathrm{net}}_{i,\tau}, q^{\mathrm{sell}}_{i,\tau}, q^{\mathrm{buy}}_{i,\tau}, \pi^{\mathrm{exp}}, \pi^{\mathrm{ret}} \right) = \Omega_i(s_\tau) \in \mathcal{O}_i. \tag{74}$$

Thus, each participant knows whether it is a seller or buyer and the quantity it has available to trade before submitting its price. It does not observe the contemporaneous bids, asks, or private local states of other participants. The market is therefore partially observable from the perspective of each individual prosumer.

Action. The trading quantity is fixed by the battery power before market clearing, so the strategic action of participant i is its battery power together with its submitted price,

$$a_{i,\tau} = \left(p^{\mathrm{bat}}_{i,\tau}, \pi_{i,\tau} \right), \qquad \pi_{i,\tau} \in \left[\pi^{\mathrm{exp}}, \pi^{\mathrm{ret}} \right], \tag{75}$$

where $p^{\mathrm{bat}}_{i,\tau} = p^{\mathrm{dis}}_{i,\tau} - p^{\mathrm{ch}}_{i,\tau}$ is the signed battery power within the feasible range (58)–(59). If $q^{\mathrm{sell}}_{i,\tau} > 0$, the submitted price is an ask; if $q^{\mathrm{buy}}_{i,\tau} > 0$, it is a bid. A participant with zero net position does not actively participate in the auction. The joint action is $a_\tau = (a_{i,\tau})_{i\in\mathcal{I}}$.

Market clearing and reward. Given the participants' net positions and joint submitted prices, the local market-clearing mechanism

$$z_\tau = \mathcal{C}^{\mathrm{p2p}}(s_\tau, a_\tau) \tag{76}$$

sorts the asks and bids and applies the double-auction clearing rules in Section C.4.1. The clearing outcome contains the total traded volume $Q^\star_\tau$, the uniform local clearing price $\lambda^{\mathrm{loc}}_\tau$, and the individual awards $q^{\mathrm{sell}\star}_{i,\tau}$ and $q^{\mathrm{buy}\star}_{i,\tau}$. Any unmatched quantity is settled with the upstream grid.

The stage reward of participant i is its realized profit,

$$\begin{aligned} r_{i,\tau} &= \mathrm{profit}_{i,\tau} = \mathrm{revenue}_{i,\tau} - \mathrm{cost}_{i,\tau} \\ &= \left(\lambda^{\mathrm{loc}}_\tau q^{\mathrm{sell}\star}_{i,\tau} + \pi^{\mathrm{exp}} q^{\mathrm{ex}}_{i,\tau} \right) - \left(\lambda^{\mathrm{loc}}_\tau q^{\mathrm{buy}\star}_{i,\tau} + \pi^{\mathrm{ret}} q^{\mathrm{im}}_{i,\tau} + c^{\mathrm{deg}}_i \Delta^{\mathrm{p2p}} (p^{\mathrm{ch}}_{i,\tau} + p^{\mathrm{dis}}_{i,\tau}) \right) \end{aligned} \tag{77}$$

where $\mathrm{profit}_{i,\tau}$ is defined by equation 70. For a seller, the reward reflects revenue from P2P sales and any residual export to the grid. For a buyer, the reward reflects the cost of P2P purchases and any residual import from the grid, together with battery degradation cost. A buyer may therefore receive a negative reward corresponding to its electricity bill, so maximizing reward is equivalent to minimizing that bill.

State transition. The battery state of charge provides the physical coupling between consecutive trading intervals. Following equation 57,

$$\mathrm{soc}_{i,\tau+1} = \mathrm{soc}_{i,\tau} + \frac{\Delta^{\mathrm{p2p}}}{E_i} \left(\eta^{\mathrm{ch}}_i p^{\mathrm{ch}}_{i,\tau} - \frac{p^{\mathrm{dis}}_{i,\tau}}{\eta^{\mathrm{dis}}_i} \right). \tag{78}$$

Photovoltaic generation and local demand evolve according to their exogenous data processes, while battery operation for the next period is determined before the next P2P auction. The next net position and trading quantity are then obtained from equation 56b. The overall state transition is

$$s_{\tau+1} \sim P(\cdot \mid s_\tau, a_\tau). \tag{79}$$

Under the market design considered here, the submitted P2P price affects market matching and settlement but does not directly alter the participant's physical net position or battery dynamics.

Agent objective. Each participant uses a policy $\sigma_i(a_{i,\tau} \mid o_{i,\tau})$ to determine its submitted price. Over a horizon of H trading intervals, participant i seeks to maximize its expected discounted profit,

$$J_i(\sigma_i, \sigma_{-i}) = \mathbb{E}\left[\sum_{\tau=1}^{H} \gamma^{\tau-1} r_{i,\tau}\right], \tag{80}$$

where σ_{-i} denotes the joint policies of the other participants. Because the cleared volume, matching outcome, and local clearing price depend on all submitted bids and asks, the return of each participant depends on both its own pricing policy and those of the other prosumers. Participants therefore learn bidding strategies through repeated local-market interactions without observing competitors' contemporaneous submissions or private states.

C.5 M5: Local Flexibility Market

Growing electrification and distributed generation can place distribution networks under increasing operational stress, leading to feeder congestion and voltage violations. Reinforcing distribution infrastructure can be costly and slow, particularly when the underlying constraint occurs only during a limited number of periods. A local flexibility market provides an alternative means of managing these constraints by allowing the distribution system operator (DSO) to procure temporary changes in power injection or consumption from resources already connected to the affected network.

The market operates at the distribution level and clears once per delivery period. It addresses local network constraints that are not represented in the transmission-level wholesale markets, while the upstream energy price faced by each aggregator is treated as exogenous. The DSO is the sole buyer, making the market a one-sided procurement auction. It specifies the locations and periods where flexibility is required, and accepts offers that restore a feasible distribution-network operating point at minimum procurement cost. The sellers are aggregators, with one aggregator at each bus, each representing a portfolio of distributed energy resources.

Two features distinguish this market from the preceding settings. First, the traded product is a deviation from a predefined baseline rather than energy itself. Second, settlement follows a pay-as-bid rule: each accepted aggregator is paid its submitted offer price for the flexibility it provides, rather than a common market-clearing price. The settlement therefore does not rely on dual-based marginal prices. This section presents the market-clearing model in subsection C.5.1, the pricing and settlement rule in subsection C.5.2, and the corresponding POSG in subsection C.5.3.

C.5.1 Market model

We consider a radial distribution network with N^{bus} buses supplied by a single substation, indexed by 0, whose voltage is fixed at $v_0 = 1$ p.u. Let τ index the local flexibility trading intervals, with duration $\Delta^{\mathrm{flex}} = 1$ hour. Before the flexibility market clears, each aggregator has a baseline operating position determined by local photovoltaic generation, demand, and planned battery charging. With no flexibility activated, the baseline active-power injection at bus n is

$$P_{n,\tau}^{\mathrm{base,inj}} = \sum_{i:\mathrm{bus}(i)=n} \left(p_{i,\tau}^{\mathrm{pv}} - d_{i,\tau} - p_{i,\tau}^{\mathrm{plan}}\right) - d_{n,\tau}^{\mathrm{bg}}, \tag{81}$$

where $p_{i,\tau}^{\mathrm{pv}}$ is the photovoltaic output, $d_{i,\tau}$ is the local demand of aggregator i, $p_{i,\tau}^{\mathrm{plan}}$ is its planned battery charging power, and $d_{n,\tau}^{\mathrm{bg}}$ denotes background demand not controlled by any aggregator.

Under the radial-network approximation, the corresponding baseline line flows and squared voltages are obtained from

$$P_{l,\tau}^{\mathrm{base}} = -\sum_{m \in \mathcal{D}(l)} P_{m,\tau}^{\mathrm{base,inj}}, \tag{82}$$

and

$$\left(v_{n,\tau}^{\mathrm{base}}\right)^2 = v_0^2 + 2\sum_{m} \left(R_{n,m} P_{m,\tau}^{\mathrm{base,inj}} + X_{n,m} Q_{m,\tau}^{\mathrm{base,inj}}\right), \tag{83}$$

where $\mathcal{D}(l)$ denotes the set of buses downstream of line l. The quantities $R_{n,m}$ and $X_{n,m}$ are the sums of resistance and reactance over the network path shared by buses n and m, respectively. Reactive injections are determined from the active loads using a fixed power-factor assumption.

Before offers are submitted, the DSO publishes indicators of the local flexibility requirement. For an undervoltage at bus n, the indicator is

$$\mathrm{req}^{\mathrm{v}}_{n,\tau} = \frac{\max\left[\left(\underline{v}_n + \gamma^{\mathrm{v}}\right)^2 - \left(v^{\mathrm{base}}_{n,\tau}\right)^2, 0\right]}{2R_{n,n}}, \tag{84}$$

while the requirement associated with forward-flow congestion on line l is

$$\mathrm{req}^{\mathrm{th}}_{l,\tau} = \max\left[P^{\mathrm{base}}_{l,\tau} - \left(1-\gamma^{\mathrm{th}}\right)\bar{P}_l, 0\right]. \tag{85}$$

These quantities are published as market information for the aggregators; the market clearing itself enforces the full network constraints below.

The traded product is upward flexibility, defined as an increase in net active power injection relative to the baseline. Aggregator i submits one price–quantity offer $(\pi_{i,\tau}, \bar{q}_{i,\tau})$, where $\pi_{i,\tau}$ is its offer price and $\bar{q}_{i,\tau}$ is the maximum flexibility quantity offered. Any quantity $q_{i,\tau} \in [0, \bar{q}_{i,\tau}]$ may be accepted by the DSO. Unlike the stepwise offers used in the wholesale markets, each aggregator submits a single price–quantity pair in each flexibility trading interval.

Upward flexibility can be delivered by reducing planned battery charging or, once planned charging has been fully removed, by discharging the battery. The offered quantity must therefore satisfy

$$0 \le \bar{q}_{i,\tau} \le q^{\mathrm{phys}}_{i,\tau}, \tag{86}$$

where

$$q^{\mathrm{phys}}_{i,\tau} = p^{\mathrm{plan}}_{i,\tau} + \min\left(\bar{p}^{\mathrm{dis}}_i, \frac{\eta^{\mathrm{dis}}_i E_i \left(\mathrm{soc}_{i,\tau} - \underline{\mathrm{soc}}_i\right)}{\Delta^{\mathrm{flex}}}\right). \tag{87}$$

Here, $\bar{p}^{\mathrm{dis}}_i$ is the battery discharge-power limit, E_i is its energy capacity, η^{dis}_i is its discharge efficiency, and $\underline{\mathrm{soc}}_i$ is the minimum admissible state of charge. Because $q^{\mathrm{phys}}_{i,\tau}$ depends on the current state of charge, the amount of flexibility available from an aggregator varies across trading intervals.

Given the submitted offers, the DSO selects flexibility awards by solving

$$\min_{\{q,s,P^{\mathrm{inj}},P,v\}} \ \Delta^{\mathrm{flex}}\left[\sum_i \pi_{i,\tau} q_{i,\tau} + \mathrm{VOLL}\sum_n s_{n,\tau}\right], \tag{88a}$$

subject to

$$P^{\mathrm{inj}}_{n,\tau} = \sum_{i:\mathrm{bus}(i)=n}\left(p^{\mathrm{pv}}_{i,\tau} - d_{i,\tau} + q_{i,\tau} - p^{\mathrm{plan}}_{i,\tau}\right) - d^{\mathrm{bg}}_{n,\tau} + s_{n,\tau}, \tag{88b}$$

$$P_{l,\tau} = -\sum_{m\in\mathcal{D}(l)} P^{\mathrm{inj}}_{m,\tau}, \tag{88c}$$

$$v^2_{n,\tau} = v^2_0 + 2\sum_m \left(R_{n,m}P^{\mathrm{inj}}_{m,\tau} + X_{n,m}Q^{\mathrm{inj}}_{m,\tau}\right), \tag{88d}$$

$$\left(\underline{v}_n + \gamma^{\mathrm{v}}\right)^2 \le v^2_{n,\tau} \le \left(\bar{v}_n - \gamma^{\mathrm{v}}\right)^2, \qquad n \ne 0, \tag{88e}$$

$$-\left(1-\gamma^{\mathrm{th}}\right)\bar{P}_l \le P_{l,\tau} \le \left(1-\gamma^{\mathrm{th}}\right)\bar{P}_l, \tag{88f}$$

$$0 \le q_{i,\tau} \le \bar{q}_{i,\tau}, \tag{88g}$$

$$0 \le s_{n,\tau} \le d^{\mathrm{bg}}_{n,\tau} + \sum_{i:\mathrm{bus}(i)=n} d_{i,\tau}. \tag{88h}$$

The objective (88a) minimizes the total procurement cost of local flexibility in each trading interval. The first term represents the pay-as-bid cost of accepted flexibility offers, while the second penalizes emergency load curtailment. The value of lost load VOLL is set above the admissible flexibility offer prices so that curtailment is used only when the available flexibility cannot restore a feasible network operating point.

Constraint (88b) defines the active-power injection at each distribution bus. Without flexibility, the baseline injection consists of photovoltaic generation minus local demand and planned battery

charging. An accepted flexibility quantity $q_{i,\tau}$ increases the bus injection relative to this baseline, either by reducing charging or by discharging the battery. Background demand $d^{\mathrm{bg}}_{n,\tau}$ represents demand not controlled by the aggregators, while $s_{n,\tau}$ denotes emergency load curtailment.

Constraint (88c) computes the active-power flow on each line as the total downstream withdrawal under the radial-network structure. Here, $\mathcal{D}(l)$ denotes the set of buses downstream of line l. Constraint (88d) uses a linearized radial distribution power-flow model. The quantities $R_{n,m}$ and $X_{n,m}$ represent the resistance and reactance shared by the paths from the substation to buses n and m, respectively. The reactive injections $Q^{\mathrm{inj}}_{m,\tau}$ are determined from the fixed power-factor assumptions of the underlying loads and are not strategic decision variables.

Constraints (88e) and (88f) enforce the distribution-network operating limits. The parameters γ^{v} and γ^{th} introduce safety margins by tightening the admissible voltage range and thermal line ratings used during clearing. Constraint (88g) allows the DSO to accept any quantity between zero and the quantity offered by aggregator i. The submitted quantity $\bar{q}_{i,\tau}$ must itself satisfy the physical deliverability condition (87). Consequently, the clearing may partially accept an offer when the full quantity is not required to restore network feasibility. Finally, constraint (88h) bounds emergency curtailment by the total active demand at each bus. Curtailment acts as a feasibility backstop when the submitted flexibility offers are insufficient to satisfy the tightened network constraints.

Since the network equations (88b)–(88d) are affine in the clearing variables under the linearized radial power-flow model, the resulting flexibility-market clearing problem is a linear program.

C.5.2 Payment and settlement

Settlement follows a pay-as-bid rule. Each accepted aggregator is paid its submitted offer price for the quantity cleared by the DSO,

$$\mathrm{revenue}_{i,\tau} = \Delta^{\mathrm{flex}} \pi_{i,\tau} q_{i,\tau}, \qquad \mathrm{profit}_{i,\tau} = \mathrm{revenue}_{i,\tau} - \mathrm{cost}_{i,\tau}, \tag{89}$$

where $\mathrm{cost}_{i,\tau}$ is the true delivery cost defined in Section C.5.3. Because settlement is pay-as-bid, two aggregators providing flexibility at the same bus and in the same interval may receive different payments if they submit different offer prices.

Although the market imposes no uniform clearing price, the curtailment backstop creates an implicit upper value for flexibility. Emergency curtailment can relieve the same network constraints at a marginal cost of VOLL, so a flexibility offer is selected only when its network benefit justifies its submitted cost relative to this alternative. This value is inherently locational because the effect of an injection depends on where it enters the network. For a binding forward-flow constraint, an additional MW of injection downstream of the constrained line reduces the line flow by one MW under equation 88c, whereas an injection outside that downstream region does not relieve that constraint. For a binding voltage constraint, the value of flexibility depends on the voltage sensitivity of the participant's bus, so injections at electrically more effective locations can support higher accepted offer prices than injections with weaker voltage impact.

The flexibility product is defined as a deviation from the pre-market baseline. For aggregator i, this baseline is determined from its photovoltaic generation, local demand, and planned battery charging,

$$p^{\mathrm{base}}_{i,\tau} = p^{\mathrm{pv}}_{i,\tau} - d_{i,\tau} - p^{\mathrm{plan}}_{i,\tau}. \tag{90}$$

An accepted flexibility award $q_{i,\tau}$ therefore requires the aggregator to increase its net injection by exactly that amount relative to the baseline. In the benchmark, delivery is assumed to equal the cleared award, so no payment is made for flexibility that is not delivered.

C.5.3 Partially Observable Stochastic Game

The local flexibility market is formulated as a general-sum partially observable stochastic game (POSG),

$$\mathcal{G}^{\mathrm{flex}} = \langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, P, \{r_i\}_{i\in\mathcal{I}}, \gamma \rangle, \tag{91}$$

where $\mathcal{I}$ denotes the set of strategic aggregators. One game step corresponds to one local flexibility trading interval τ.

State. The state at interval τ is

$$s_\tau = (x_\tau^{\text{sys}}, \{x_{i,\tau}\}_{i\in\mathcal{I}}) \in \mathcal{S}, \tag{92}$$

where x_τ^{sys} describes the distribution-network operating condition before flexibility procurement. It contains the baseline bus injections, line flows, voltages, and the flexibility-need indicators published by the DSO,

$$x_\tau^{\text{sys}} = \left(\mathbf{P}_\tau^{\text{base,inj}}, \mathbf{P}_\tau^{\text{base}}, \mathbf{v}_\tau^{\text{base}}, \mathbf{req}_\tau^{\text{v}}, \mathbf{req}_\tau^{\text{th}}\right). \tag{93}$$

The local physical state of aggregator i is

$$x_{i,\tau} = \left(\text{soc}_{i,\tau}, p_{i,\tau}^{\text{pv}}, d_{i,\tau}, p_{i,\tau}^{\text{plan}}\right), \tag{94}$$

which, together with the planned charging in its action, determines its baseline injection and the maximum physically deliverable flexibility $q_{i,\tau}^{\text{phys}}$ through equation 87. Network parameters, battery ratings, and other fixed technical parameters are omitted from the dynamic state.

Observation. Each aggregator observes its own operating state together with the public network information released by the DSO. Its observation is

$$o_{i,\tau} = \left(\text{soc}_{i,\tau}, p_{i,\tau}^{\text{pv}}, d_{i,\tau}, \mathbf{req}_\tau^{\text{v}}, \mathbf{req}_\tau^{\text{th}}\right) = \Omega_i(s_\tau) \in \mathcal{O}_i. \tag{95}$$

Thus, each aggregator knows its own available flexibility and the locations and magnitudes of the network needs published before bidding. It does not observe the private operating states, delivery costs, or contemporaneous offers of other aggregators. The market is therefore partially observable from the perspective of each participant.

Action. At each trading interval, aggregator i submits one price–quantity offer together with its planned battery charging,

$$a_{i,\tau} = \left(\pi_{i,\tau}, \bar{q}_{i,\tau}, p_{i,\tau}^{\text{plan}}\right) \in \mathcal{A}_i, \tag{96}$$

subject to

$$c^{\text{rep}} \le \pi_{i,\tau} \le \overline{\pi}, \qquad 0 \le \bar{q}_{i,\tau} \le q_{i,\tau}^{\text{phys}}, \qquad 0 \le p_{i,\tau}^{\text{plan}} \le \bar{p}_{i,\tau}^{\text{ch}}. \tag{97}$$

The offer price $\pi_{i,\tau}$ determines the payment requested per unit of flexibility, $\bar{q}_{i,\tau}$ specifies the maximum quantity the aggregator is willing to provide, and $p_{i,\tau}^{\text{plan}}$ enters the baseline injection (81). Here, c^{rep} is the replacement cost of one delivered MWh, i.e., the energy price and the degradation cost corrected for the round-trip efficiency, $\overline{\pi} < \text{VOLL}$ is the market price cap, and $\bar{p}_{i,\tau}^{\text{ch}}$ is the charging power the battery can absorb in the interval given its state of charge. The DSO may accept any quantity $q_{i,\tau} \in [0, \bar{q}_{i,\tau}]$. The joint action is $a_\tau = (a_{i,\tau})_{i\in\mathcal{I}}$.

Market clearing and reward. Given the current state and joint offers, the DSO applies the network-constrained clearing mechanism

$$z_\tau = \mathcal{C}^{\text{flex}}(s_\tau, a_\tau), \tag{98}$$

which solves equation 88. The clearing outcome contains the accepted flexibility quantities $q_{i,\tau}$, resulting bus injections, line flows and voltages, together with any emergency load curtailment.

Under the pay-as-bid settlement rule, the revenue of aggregator i is

$$\text{revenue}_{i,\tau} = \Delta^{\text{flex}} \pi_{i,\tau} q_{i,\tau}. \tag{99}$$

Its stage reward is the realized profit,

$$r_{i,\tau} = \text{profit}_{i,\tau} = \text{revenue}_{i,\tau} - C_i^{\text{flex}}\left(q_{i,\tau}; x_{i,\tau}\right), \tag{100}$$

where $C_i^{\text{flex}} = \Delta^{\text{flex}}\left[\lambda_\tau\, p_{i,\tau}^{\text{ch}} + c_i^{\text{cyc}}\left(p_{i,\tau}^{\text{ch}} + p_{i,\tau}^{\text{dis}}\right)\right]$ is the true cost of the interval, the energy bought for charging at the exogenous price λ_τ plus the degradation cost c_i^{cyc} per MWh of battery throughput, with p^{ch} and p^{dis} given by the state transition below. This cost depends on the aggregator's physical operating state and the quantity actually delivered rather than on its submitted offer price.

State transition. An accepted flexibility award changes the battery operating trajectory and therefore affects the flexibility available in subsequent intervals. Let

$$q_{i,\tau}^{\mathrm{red}} = \min\Big(q_{i,\tau}, p_{i,\tau}^{\mathrm{plan}}\Big) \tag{101}$$

denote the part of the flexibility award delivered by reducing planned charging, and let

$$q_{i,\tau}^{\mathrm{dis}} = \max\Big(q_{i,\tau} - p_{i,\tau}^{\mathrm{plan}}, 0\Big) \tag{102}$$

denote the remaining part delivered through battery discharge. The realized battery charging and discharging powers are therefore

$$p_{i,\tau}^{\mathrm{ch}} = p_{i,\tau}^{\mathrm{plan}} - q_{i,\tau}^{\mathrm{red}}, \qquad p_{i,\tau}^{\mathrm{dis}} = q_{i,\tau}^{\mathrm{dis}}. \tag{103}$$

The state of charge evolves according to

$$\mathrm{soc}_{i,\tau+1} = \mathrm{soc}_{i,\tau} + \frac{\Delta^{\mathrm{flex}}}{E_i}\left(\eta_i^{\mathrm{ch}} p_{i,\tau}^{\mathrm{ch}} - \frac{p_{i,\tau}^{\mathrm{dis}}}{\eta_i^{\mathrm{dis}}}\right). \tag{104}$$

Hence, providing flexibility in the current interval can reduce the battery energy available for future flexibility provision. Photovoltaic generation, local demand, planned battery operation, and the resulting network operating conditions evolve according to their exogenous data processes. The overall state transition is

$$s_{\tau+1} \sim P(\cdot \mid s_\tau, a_\tau)\,. \tag{105}$$

Agent objective. Each aggregator uses a policy $\sigma_i(a_{i,\tau} \mid o_{i,\tau})$ to determine its flexibility offer. Over a horizon of H trading intervals, aggregator i seeks to maximize its expected discounted profit,

$$J_i(\sigma_i, \sigma_{-i}) = \mathbb{E}\left[\sum_{\tau=1}^{H} \gamma^{\tau-1} r_{i,\tau}\right], \tag{106}$$

where σ_{-i} denotes the joint policies of the other aggregators. Because the DSO jointly selects offers according to their submitted prices, available quantities, locations, and effects on network constraints, the return of each aggregator depends on both its own bidding policy and those of the other participants. Aggregators therefore learn price–quantity offering strategies through repeated market interactions without observing competitors' private states, costs, or contemporaneous offers.

D COMPUTATION PERFORMANCE

This section compares the throughput of the same environment code under three layouts: all on the GPU; environment on the host CPU, policy on the GPU; all on the host CPU. With 64 parallel environments, the throughput of all on the GPU is 2.3 to 22 times that of all on the host CPU and 2.3 to 33 times that of environment on the host CPU, policy on the GPU, in every one of the five markets, and it does not change with the number of host threads allotted. Figure 7 compares the three layouts in the market with the most participants, the M4 peer-to-peer local energy market with 1 200 households; Figure 8 covers all five markets, and Figure 9 open-source tools. Hardware and software are those of Appendix A.3; each measurement point is run three times and compilation time is excluded. Throughput is the number of environment steps advanced per second, summed over the parallel environments, where one step covers the market clearing and one forward pass of the policy, without gradient updates.

D.1 THREE LAYOUTS ON THE PEER-TO-PEER MARKET

In the peer-to-peer local energy market with 64 parallel environments, all on the GPU advances 27 929 environment steps per second, 22 times the throughput of all on the host CPU and 33 times that of environment on the host CPU, policy on the GPU (Figure 7a). Raising the number of parallel environments from 64 to 1 024 leaves the throughput unchanged, and so does the number of host threads allotted (Figure 7b).

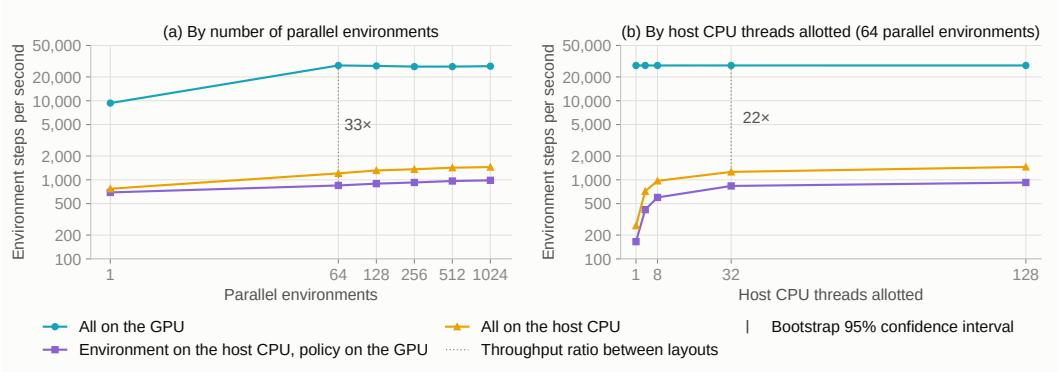


Figure 7: Throughput of the peer-to-peer local energy market environment (1 200 households, 96 periods per episode) under three layouts: (a) against the number of parallel environments; (b) against the number of host CPU threads allotted, with 64 parallel environments. Each point is the mean of three runs with a bootstrap 95% confidence interval; each dotted line joins two points and the number beside it is their throughput ratio.

D.2 ACROSS THE FIVE MARKETS

All five markets show the speed and throughput advantage of PowerMarketJax: with 64 parallel environments, all on the GPU gives 2.3 to 22 times the throughput of all on the host CPU and 2.3 to 33 times that of environment on the host CPU, policy on the GPU (Figure 8a); all on the GPU, raising the number of parallel environments from 1 to 128 multiplies the throughput by 2.9 to 12 (Figure 8b).

D.3 AGAINST OPEN-SOURCE TOOLS

The clearing problem of the M2 real-time balancing market was also implemented in PyPSA (Brown et al., 2018) and in cvxpy (Diamond & Boyd, 2016; Agrawal et al., 2018), each in its default usage (default formulation and solver), stepping one environment at a time. All on the GPU with 64 parallel environments gives 71 times the throughput of PyPSA and 8.8 times that of cvxpy (Figure 9). Training PPO on the real-time balancing market for the same number of environment steps, Pow-

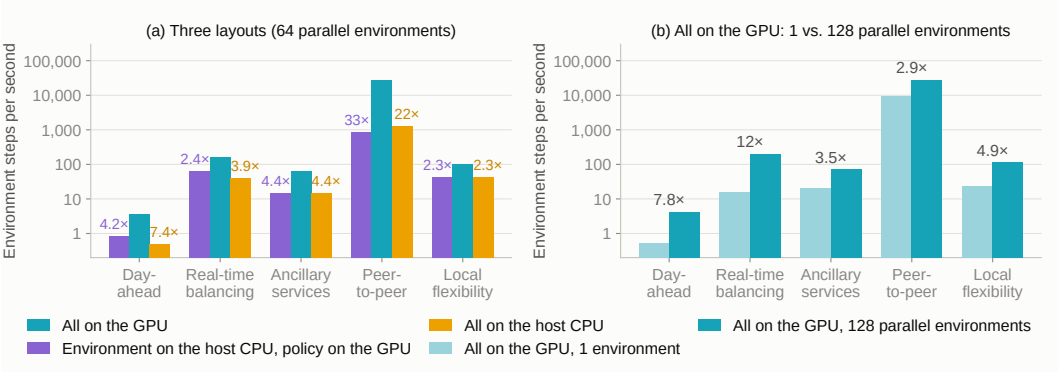


Figure 8: Throughput of the five market environments: (a) the three layouts with 64 parallel environments, the number above each host bar being the ratio of all on the GPU to that layout; (b) all on the GPU with 1 and with 128 parallel environments, the number above each pair being the ratio of the two bars. Each bar is the mean of three runs.

erMarketJax all on the GPU is 3.8 times faster than Stable-Baselines3 (Raffin et al., 2021) with its default settings (wall-clock time, including gradient updates).

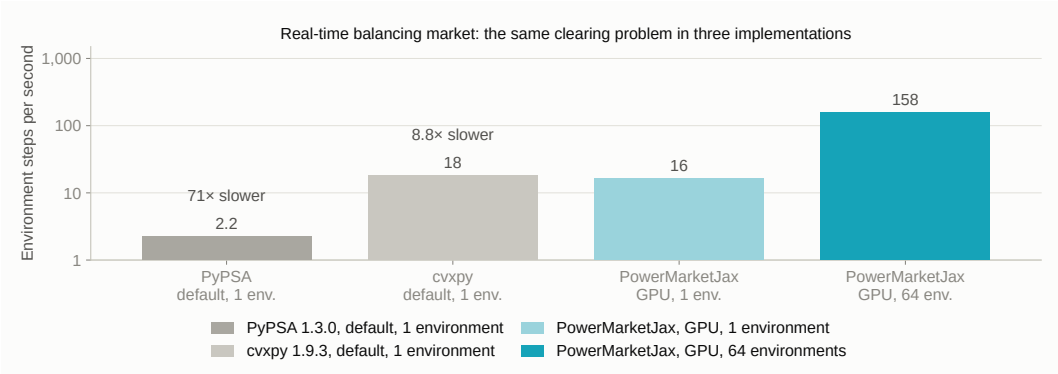


Figure 9: Throughput of the real-time balancing market environment in three implementations: PyPSA 1.3.0 and cvxpy 1.9.3 in their default usage, one environment at a time, and PowerMarketJax all on the GPU with 1 and with 64 parallel environments. Each bar is the mean of three runs; the label above each grey bar is how many times slower it is than PowerMarketJax with 64 parallel environments.

D.4 Implementation validation

We validate the implementation in three ways: market clearing and settlement against a NumPy reference implementation; the dual prices of the JAX linear-program solver against HiGHS (Huangfu & Hall, 2018), an independent solver; and the three-stage clearing of the M1 day-ahead market against the exact solution of the same commitment problem, computed by HiGHS with the commitment decisions kept binary.

Against a NumPy reference implementation. Each market has a separate NumPy reference implementation that rewrites the same clearing and settlement rules with explicit matrices and loops. We run both on every period of the evaluation days of each market under truthful offers and compare each quantity (Table 3). In M3, load shedding and reserve shortfall are compared as totals per period, because the optimum does not fix how they are split across buses and reserve products.

Table 3: Largest difference between the JAX implementation and the NumPy reference. The relative difference divides the absolute difference by the largest absolute reference value.

Market	Quantity	Data	Absolute difference	Relative difference
M1	LMP	36 days × 24 h, case29gb	2.2×10^{-8} \$/MWh	1.4×10^{-10}
	Award		1.3×10^{-11} MW	1.7×10^{-15}
	Objective value		9.3×10^{-10} \$	4.9×10^{-16}
	Settlement (profit)		2.8×10^{-9} \$	6.3×10^{-16}
M2	LMP	36 days × 48 half-hours, case29gb	9.1×10^{-8} \$/MWh	9.1×10^{-12}
	Award		1.6×10^{-11} MW	2.1×10^{-15}
	Settlement (profit)		1.5×10^{-10} \$	1.7×10^{-17}
M3	LMP	36 days × 48 half-hours, case29gb	7.3×10^{-6} \$/MWh	7.3×10^{-10}
	Reserve price		1.2×10^{-4} \$/MWh	4.7×10^{-7}
	Award		5.1×10^{-9} MW	6.5×10^{-13}
	Load shed, total		1.4×10^{-11} MW	7.7×10^{-15}
	Reserve shortfall, total		9.3×10^{-12} MW	4.0×10^{-15}
	Objective value		3.0×10^{-4} \$	3.0×10^{-11}
	Settlement (profit)		4.7×10^{-2} \$	5.5×10^{-9}
M4	Clearing price	64 episodes × 96 quarter-hours, 1 200 households	6.1×10^{-6} EUR/MWh	1.8×10^{-8}
	Traded volume		1.4×10^{-8} MWh	1.0×10^{-7}
	Award		1.9×10^{-8} MWh	3.9×10^{-6}
	Settlement (profit)		1.8×10^{-6} EUR	1.0×10^{-6}
M5	Award	36 days × 24 h, feeder 459_0	3.4×10^{-9} MW	1.3×10^{-8}
	Load shed		1.5×10^{-9} MW	5.0×10^{-7}
	Objective value		5.2×10^{-7} CHF	1.1×10^{-9}
	Settlement (profit)		5.2×10^{-7} CHF	1.5×10^{-8}

Dual prices against HiGHS. The prices of M1, M2 and M3 are duals of a linear program. On every period of the same evaluation days, we solve the same linear program with HiGHS and compare its prices with those of the JAX solver (Table 4). M4 and M5 set prices without a linear program.

Table 4: Largest difference between the JAX solver and HiGHS.

Market	Test system	Price	Periods	Price difference (\$/MWh)	Relative objective difference
M1	case29gb	LMP	864	2.6×10^{-8}	3.1×10^{-16}
M1	case73rts	LMP	864	4.1×10^{-8}	6.2×10^{-8}
M1	case813nem	LMP	864	3.5×10^{-7}	1.2×10^{-7}
M2	case29gb	LMP	1 728	1.1×10^{-7}	1.8×10^{-12}
M3	case29gb	LMP	1 728	7.3×10^{-6}	3.1×10^{-11}
M3	case29gb	Reserve price	1 728	2.1×10^{-7}	3.1×10^{-11}

Gap to the exact optimum. The M1 day-ahead clearing has binary commitment variables, and no mixed-integer solver runs inside the compiled graph, so the market is cleared by a three-stage procedure: a linear program with the on/off decisions relaxed to values between 0 and 1, rounding of the relaxed commitment, and a re-solve of the economic dispatch with the commitment fixed. On the 36 evaluation days of each of the three M1 test systems, under truthful offers, we solve the same unit commitment problem exactly and offline with HiGHS (commitment variables 0 or 1) and compare its production cost with that of the three-stage procedure.

Over the 36 evaluation days, the median excess production cost of the three-stage procedure over the exact optimum is 2.04% on case29gb, 0 on case73rts and 1.10% on case813nem, and the maximum is 3.78%, 1.79% and 1.95% (Figure 10). On 23 of the 36 evaluation days of case73rts, the three-stage commitment is the same as the exact optimum and the gap is 0.

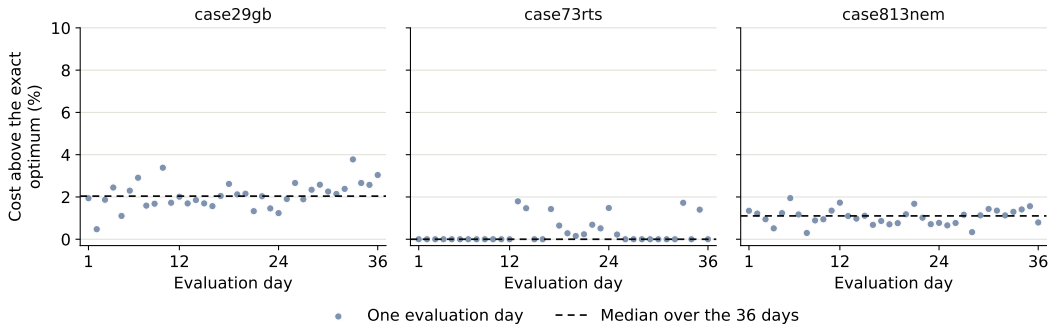


Figure 10: Production cost of the three-stage clearing above the exact optimum on the evaluation days of the three M1 test systems, under truthful offers. Each point is one evaluation day, in date order; the dashed line is the median.

E QUICKSTART AND API

This section is a walk-through of how to install PowerMarketJax, run a minimal rollout, read the interface contract that the five markets share, construct each of them, find where an experimental configuration is written down, and reproduce a result from the command line. Listing E.2 is a complete runnable program, Listing E.2 a rollout function that takes an environment built by any of the five constructors, Listing E.5 is abridged from a released source file, and the bash blocks in Sections E.1 and E.6 cover installation and experiment script usage. The market models themselves, including the observation, action, clearing and settlement of each market, are in Appendix C; this section covers only the software interface to them.

E.1 INSTALLATION

PowerMarketJax requires Python 3.10–3.12 and has no build step. The base install ships a CPU-only JAX; CUDA 12 support is opt-in via the `cuda` extra, which adds the CUDA plugin alongside the already resolved `jaxlib` at the same version and therefore does not move the JAX version the suite is verified on. GPU support follows JAX's own platform support: the CUDA build is available on Linux only, so on Windows the package runs on the CPU natively and on the GPU through WSL2. All reported results and test runs were obtained on Linux. The learning stack of `rlax` and `distrax` used by the P2P trading experiment script lives behind `rl`, and the figure scripts behind `figs`. Nothing under `powermarketjax/` imports `matplotlib`, `rlax` or `distrax`, so the five markets and both learners are available on a `dev` install alone.

Listing E.1: PowerMarketJax Installation commands

```
git clone https://github.com/powermarketjax/PowerMarketJax
conda create -y -n powermarketjax python=3.11 && conda activate
    powermarketjax

pip install -e ".[dev]"                    # CPU JAX, the five markets,
     both learners
pip install -e ".[dev,cuda,rl,figs]"       # Linux + CUDA 12, plus
    scripts and figures

python -c "import powermarketjax, jax; print(powermarketjax.
    __version__, jax.__version__, jax.devices())"
```

E.2 MINIMAL ROLLOUT

Listing E.2 opens the day-ahead wholesale market on the 29-bus GB case, resets it, and runs a three-step rollout under `jax.lax.scan` with every agent bidding at twice its cost. One step is one market day, so the rollout clears three consecutive days of a multi-period security-constrained unit commitment; every transition, the clearing included, stays inside `jit`.

Listing E.2: Minimal rollout on `case29gb` (day-ahead wholesale)

```
import jax, jax.numpy as jnp
jax.config.update("jax_enable_x64", True) # M1/M2/M3/M5 refuse to
    build without it

from powermarketjax.case import load_case
from powermarketjax.envs.day_ahead import make_env, load_commitment,
    load_gb_demand
from powermarketjax.utils.jax_utils import scan_rollout

env, spec = make_env(load_case("29gb"),
                     load_commitment(n_periods=24), load_gb_demand(),
                     kind="markup", markup_max=2.0,
                     cap_scale=0.60, ramp_scale=1.00)
params = env.make_params(episode_len=3)
```

```
obs, state = env.reset(jax.random.PRNGKey(0), params)
actions = jnp.full((3, spec["n_agents"]), 2.0)          # (n_steps,
    n_agents)
final_state, obs_traj, reward_traj, cost_traj, done_traj, info_traj =
    jax.jit(lambda k, s: scan_rollout(env, k, s, params, actions)
)(jax.random.PRNGKey(1), state)
```

At this configuration the market has 66 agents, one per thermal unit: `obs_traj` is `(3, 66, 112)`, `reward_traj` is `(3, 66)` and `cost_traj` is `(3, 66, 2)`, and `info_traj["converged"]` reports per step whether the clearing met its tolerance. `scan_rollout` drives `step_auto_reset`, so episodes restart at their boundary without a Python conditional and the trajectory has a fixed length.

The scenario parameters have no defaults. `cap_scale` and `ramp_scale` in Listing E.2 are required: omitting either raises `ValueError`. The two scale the line ratings and the ramp limits the case ships with, so `cap_scale` 1.00 means every line keeps the capacity written in the case file and in that network no line ever reaches its limit. With no line at its limit there is no congestion, and with no congestion every bus clears at the same price.

Listing E.2 is the same contract on any of the five markets, vectorised over a batch of environments. `unpack_env` accepts whichever shape a market's constructor returns (Section E.4) and yields the same four objects, so the rollout below is market-agnostic.

Listing E.3: One rollout function for five markets: `vmap` over environments inside `scan` over time

```
import math, jax, jax.numpy as jnp, numpy as np
from functools import partial
from powermarketjax.learning import unpack_env
from powermarketjax.resources.battery import make_battery_bundle
from powermarketjax.envs.p2p import (make_p2p_env, make_p2p_params,
    load_fluvius_households)
jax.config.update("jax_enable_x64", True) # M1/M2/M3/M5 refuse to
    build without it
# 'built' is whatever a market's constructor returned: the (env, spec)
     pair  for M1/M2, or the (reset, step, step_auto_reset, spec)
    tuple for M3-M5.  Here it is M4, the P2P trading market, on 12
    metered houses.
N, DT, ETA = 12, 0.25, math.sqrt(0.85)
series = load_fluvius_households(n_households=N)
battery = make_battery_bundle(n_devices=N, dt_hours=DT,
                              capacity_mwh=[0.01] * N,
                              power_mw=[0.005] * N,
                              eta_charge=ETA, eta_discharge=ETA,
                              soc_min=0.15, soc_max=1.0,
                              initial_soc=0.5,
                              cycle_cost_per_mwh=0.0)
params = make_p2p_params(p_pv=series.injection, load=series.offtake,
                         battery=battery,
                         kappa=np.full(N, 13.88, np.float32),
                         learner_mask=np.ones(N, bool),
                         episode_len=96)
built = make_p2p_env(N, 73.0, 333.4, DT)    # n_agents, export/retail,
    dt
reset, step, step_auto_reset, spec = unpack_env(built)
# unpack_env takes either shape and gives the four names below.

def rollout(key, params, policy, n_envs, n_steps):
    k_reset, k_roll = jax.random.split(key)
    obs, state = jax.vmap(reset, in_axes=(0, None))(
        jax.random.split(k_reset, n_envs), params)

    def one_step(carry, k):
```

```
        obs, state = carry
        action = policy(obs)                          # (n_envs, *
    action_shape)
        obs, state, reward, costs, done, info = jax.vmap(
            step_auto_reset, in_axes=(0, 0, 0, None))(
                jax.random.split(k, n_envs), state, action, params)
        return (obs, state), (reward, costs, done)

    return jax.lax.scan(one_step, (obs, state),
                        jax.random.split(k_roll, n_steps))

# The action shape comes from the market, not from the learner.
zero = lambda obs: jnp.zeros((obs.shape[0], *spec["action_shape"]),
                             jnp.float32)
(obs, state), (reward, costs, done) = jax.jit(
    partial(rollout, policy=zero, n_envs=8, n_steps=6)
)(jax.random.PRNGKey(0), params)
```

The returned `reward` has shape `(n_steps, n_envs, N)` and `costs` has shape `(n_steps, n_envs, N, C)`: time, parallel environments and market participants are three separate axes, and the second and third are the two the accelerator exploits. `params` is closed over as a broadcast argument, so a sweep over anything carried there is a loop over calls and not a recompilation.

E.3 The environment interface

Signatures. Every market exposes the same three pure functions:

Listing E.4: Market environment signatures

```
obs, state = reset(key, params)
obs, state, reward, costs, done, info = step(key, state, action,
    params)
obs, state, reward, costs, done, info = step_auto_reset(key, state,
    action, params)
```

With N agents and C constraint channels, `obs` is (N, d), `reward` is $(N,)$, `costs` is (N, C) and `done` is a scalar boolean. The agent index is one array axis: observations, actions, rewards and costs are aligned on it, and N is fixed at compile time.

Reward and constraint costs. The reward of an agent is the profit it earns in that step, settled on the quantities and prices produced by the market's own clearing (Appendix C). Violations of physical or market constraints, such as shed load or an unmet reserve requirement, are returned separately in `costs`, one column per name in `spec["cost_names"]`, and are never subtracted from the reward. A user decides how to handle them, for example by adding a penalty to the reward or by using a constrained learning method.

Episode boundary. `spec["termination"]` tells a learner what the end of an episode means. In four markets it is `"truncation"`: the episode is cut off in time, the market would continue, and the value of the last state should be bootstrapped. In the P2P market it is `"terminal"`: the last reward already pays each household for the energy left in its battery, so bootstrapping would count that energy twice.

E.4 Constructing the five markets

Each market has its own constructor, since the scenarios differ: the wholesale markets take a network case, a precomputed day-ahead commitment and schedule, and a demand series; the ancillary market adds the reserve products; the P2P trading market takes a population size and the export and retail tariffs; the local flexibility market takes a feeder and the placement of the aggregators. The ancillary, P2P and local flexibility constructors return `(reset, step, step_auto_reset, spec)`, and the day-ahead and real-time constructors return `(env, spec)`; `powermarketjax.learning.unpack_env` converts either to the first form, as in

Listing E.2. Anything that changes the size of the clearing problem, such as the network or the placement of agents, is fixed when the environment is built, and changing it requires recompiling; anything that varies within a market, such as the demand series or the episode length, is passed in `params` on every call.

E.5 Scenario and learner configuration

There is no configuration file format: an experimental configuration is Python. The learner configuration is `tools/benchmark/hyperparams.py`, abridged in Listing E.5. The scenario parameters are module-level constants of each market's experiment script, each also exposed as a command-line flag.

Listing E.5: The shared learner configuration

```
N_ENVS, MINIBATCHES = 64, 32
HORIZON = {"01": 4, "02": 48, "03": 48}    # steps per update, per
    market
for _m, _h in HORIZON.items():             # the batch must divide into
     minibatches
    assert (N_ENVS * _h) % MINIBATCHES == 0

SHARED = IPPOConfig(
    n_envs=N_ENVS,
    horizon=HORIZON["02"],   # ours, per market via shared_for()
    epochs=10,               # update_epochs
    minibatches=MINIBATCHES,# num_minibatches
    lr=3e-4,                 # learning_rate (no annealing here)
    clip_eps=0.2,            # clip_coef
    gamma=0.99,              # gamma
    gae_lambda=0.95,         # gae_lambda
    vf_coef=0.5,             # vf_coef
    ent_coef=0.0,            # ent_coef
    max_grad_norm=0.5,       # max_grad_norm
    hidden=(64, 64),         # Agent: two tanh layers of 64
    weight_decay=0.0,        # zero reproduces optax.adam exactly
    init_scale=1.0,
)

def shared_for(market):
    return dataclasses.replace(SHARED, horizon=HORIZON[market])
```

E.6 Command-line usage

Each market has one experiment script for its learning runs. The non-learning baselines are run by a separate script for the day-ahead, real-time and ancillary markets (`tools/benchmark/run_eval_0*.py`), by the learning script itself for the P2P trading market (`--arm truthful`) and for the local flexibility market, where they run alongside the learners. The learning script takes the learner (`--algo ippo|sac`), parameter sharing or not (`--per-agent-params` for one network per agent), the seed, the iteration count, the scenario parameters and an output directory. It trains on the training days only and then evaluates the mean action of the policy on the held-out days, so the reported number excludes exploration noise.

Listing E.6: Command line for RL training

```
# One-year inputs for M1-M3 (2023-07-10 to 2024-07-08), built once.
python tools/commitment/precommit.py --mode relax --start-day 5 --days
     365 --cap-scale 0.6 --ramp-scale 1.0 --out commit_y1.npz
python tools/commitment/da_position.py --case 29gb --fixture commit_y1
    .npz --out position_y1.npz

# M1 day-ahead: parameter sharing, then one network per agent.
```

```
python tools/benchmark/run_rl_01.py --fixture commit_y1.npz --algo
    ippo --seed 0 --iterations 400 --out-dir runs/m1-ippo-ps
python tools/benchmark/run_rl_01.py --fixture commit_y1.npz --algo
    ippo --seed 0 --iterations 400 --per-agent-params --out-dir runs/
    m1-ippo-nops

# M2 real-time and M3 ancillary settle against the day-ahead position.
python tools/benchmark/run_rl_02.py --position position_y1.npz --seed
    0 --out-dir runs/m2-ippo-ps
python tools/benchmark/run_rl_03.py --position position_y1.npz --seed
    0 --out-dir runs/m3-ippo-ps

# M4 P2P, 1200 households.
python tools/benchmark/run_rl_04.py --agents 1200 --seed 0 --out-dir
    runs/m4-ippo-ps

# M5 local flexibility.
PYTHONPATH=tools/flex_experiment python -m concentration_baseline --
    algo ippo --seeds 3 --out runs/m5-ippo-ps

# Non-learning baselines for M1 on the held-out days.
python tools/benchmark/run_eval_01.py --fixture commit_y1.npz --days
    eval --out-dir results/m1-baselines
```

E.7 Cases and data

`powermarketjax.case.list_cases()` returns the 18 built-in network cases, 10 transmission and 8 distribution, sorted by bus count and with aliases removed; `load_case` accepts a canonical identifier or an alias. One case requires an explicit voltage-band declaration and its zero-argument factory raises until it is given, which is recorded on the case metadata so that code enumerating cases filters on the property rather than on a name. The demand, generation, price and smart-meter series are resolved through the 15 manifests under `powermarketjax/data/manifests/`, one per dataset, recording the source and provenance of each series; the parquet files are carried in the repository, so every listing above runs on a fresh checkout with no separate download step.

E.8 Clearing and learning in a single computation graph

A training iteration has two parts, a rollout and an update. In the rollout, N environments advance together for T steps. In each step the participants submit offers, the market clears, the awards are settled, and the next observation is constructed; the policy maps the observation to the action at every step, and the observation, action, reward and constraint cost of one step form one experience. The update takes gradient steps on the $N \times T$ experiences of the rollout, split by epoch and minibatch. The whole iteration, including the clearing problem of every step, is compiled as one function on the GPU: the host makes one call and receives the training metrics, and every clearing, every policy forward pass and every gradient step in between runs on the GPU in sequence. Figure 11 shows the data flow of one iteration, and Listing E.7 writes the same iteration as loops, with the JAX primitive that implements each loop in the comment above it. The values of N and T for each market are in Table 6, the epochs and minibatches in Table 5, and n, the number of agents, in Appendix C.

The loop over the N environments is `vmap(step)`. The function `step` advances one environment by one step and is written for one environment only; `vmap` returns the function that advances N environments at once, whose input and output gain one axis of length N. The clearing, settlement and observation of the N environments are one batched computation over N sets of data, not N calls one after another. The forward pass of the policy goes through `vmap` in the same way, so the actions are produced inside the same function and passed directly to `step`, and the observations never reach the host.

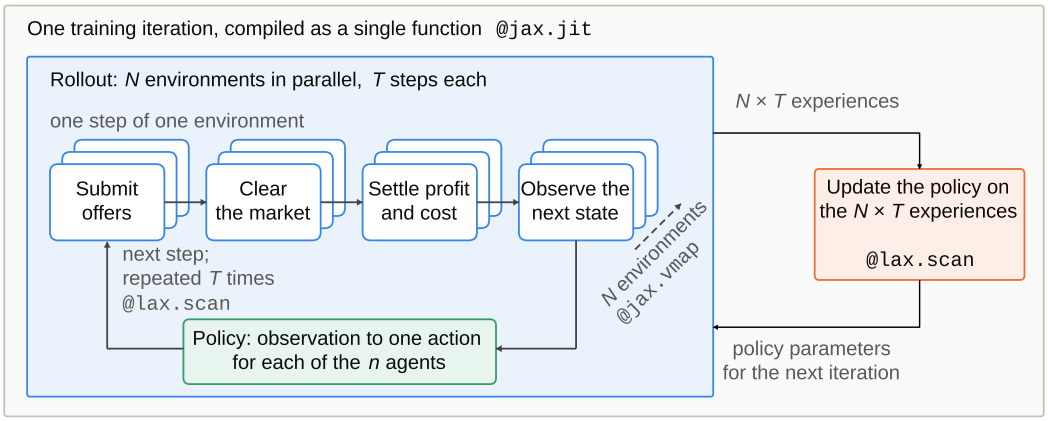


Figure 11: One training iteration of IPPO, compiled as a single function. Rollout: N environments advance in parallel for T steps; in each step the n agents submit offers, the market clears, the awards are settled and the next observation is constructed, and the policy maps the observation to the next action inside the same function. Update: the policy is updated on the $N \times T$ experiences, with a second scan over epochs and minibatches (Table 5). The values of N and T for each market are in Table 6.

Listing E.7: One training iteration as loops, with the primitive that implements each loop

```
# iteration = jit(iteration): the whole body below is one compiled
#   function on the GPU, called once per training iteration
iteration(params, states, observations):
    # rollout
    # scan(one_step, carry, length=T): carry = (states, observations);
    #   returns the final carry and the T records stacked
    for t = 1 .. T:
        # policy_N = vmap(policy), step_N = vmap(step):
        #   each takes N inputs and returns N outputs at once
        for each of the N environments:
            for each of the n agents: action <- policy(observation)
            # step(state, actions), one environment
            offers          <- submit(actions)
            awards, prices  <- clear(offers)
            reward, cost    <- settle(awards, prices)
            observation     <- observe(the state after clearing)
            if the episode ended: reset this environment in place
        # one record per step, holding the N environments
        record (observations, actions, rewards, costs)
    # N x T experiences, each with n agents
    experiences <- the T records
    # update
    # scan(gradient_step, ...): the passes over the experiences,
    #   by epoch and minibatch
    for each pass over the experiences:
        params <- optimiser step on the gradient of the loss
    return params, states, observations, training metrics
```

The loop over the T steps is `lax.scan(one_step, carry, length=T)`. It takes the loop body `one_step`, the initial carry, which is the N states and the N observations, and the number of steps T; runs the body T times, feeding the carry returned by one run into the next; stacks the record of each run into an array of length T; and returns the final carry together with this array. The horizon T is fixed, and an environment whose episode ends before step T is reset in place inside the body, so the input and output of the body have the same shape at every step, which is the condition under which `lax.scan` applies. The gradient steps of the update are a second `lax.scan`. None

of these loops runs on the host, and the host does not see the boundary between two consecutive steps.

The outer function is `jit(iteration)`: the rollout followed by the update, compiled for the GPU at the first call and called once per iteration. The host passes in the parameters, the environment states and the observations, and receives the updated parameters, states, observations and training metrics; no intermediate result returns to the host, and the only loop left on the host is the loop over iterations. The iteration of SAC has the same form: the rollout is the same, and the update first places the experiences of the rollout into a replay buffer of fixed size, which also lives inside this function and is carried from one iteration to the next, then samples from it for the gradient steps, which are again a `lax.scan`.

F Training Setup and Hyperparameters

The results of Section 5 and Appendix G share four learners, one set of hyperparameters and one evaluation protocol; where a market departs from them, its subsection of Appendix G gives the difference. Appendix E.5 shows where these values are set in the code.

F.1 Four learners

Each market is run with four learners: IPPO-PS, IPPO-NoPS, SAC-PS and SAC-NoPS. All four are independent learners: every agent sees only its own observation and optimises only its own profit, with no centralised value network. PS and NoPS stand for parameter sharing and no parameter sharing. Under PS all agents share one set of networks and pool their samples for training; under NoPS each agent has its own set and trains only on its own samples. For IPPO the set is one actor and one value network that share two $\tanh$ layers, and the log standard deviation of the actor does not depend on the observation; for SAC it is one stochastic actor and a pair of Q-networks with their target networks, with an automatically tuned temperature.

F.2 Hyperparameters

The hyperparameters build on the defaults of the continuous-action PPO and SAC implementations in CleanRL; the five markets use one set, listed in Table 5, without tuning for each market. The profit of one step in M1 is of the order of 10^5 to 10^6 dollars and the Q regression of SAC is sensitive to this magnitude, so SAC divides the reward by a constant before the Q regression: the standard deviation of the rewards under truthful offers, computed once before training. SAC runs the gradient steps of an iteration as one block after the rollout. In all four learners each component of the observation is normalised with constants fixed before training.

Table 5: Hyperparameters of the two IPPO and the two SAC learners, shared by the five markets.

Hyperparameter	IPPO	SAC
Optimiser	Adam	Adam
Learning rate	3×10^{-4}, constant	actor 3×10^{-4}, Q-networks and temperature 10^{-3}
Discount factor γ	0.99	0.99
Hidden layers	two layers of 64 units, $\tanh$	two layers of 256 units, ReLU
GAE parameter λ	0.95	—
Clipping range	0.2	—
Update epochs per iteration	10	—
Minibatches per epoch	32	—
Value loss coefficient	0.5	—
Entropy coefficient	0	—
Maximum gradient norm	0.5	—
Initialisation scale of the actor output layer	1.0	—
Replay buffer	—	32 768 environment steps
Batch size	—	256 environment steps, each carrying the transitions of all agents
Gradient steps per environment step	—	1
Random-action steps before training	—	0
Actor update interval	—	every gradient step
Target smoothing coefficient τ	—	0.005
Temperature	—	initial value 0.2, tuned automatically
Target entropy	—	minus the action dimension
Range of the log standard deviation	—	$[-5, 2]$

The two SAC learners of M4 and the two IPPO learners of M5 differ from Table 5 in the settings given in Sections G.4.2 and G.5.2.

F.3 TRAINING SCALE

Table 6 gives the training scale of each market.

Table 6: Training scale.

Market	Seeds per learner	Iterations	Parallel environments and steps per iteration	Environment steps per iteration
M1 day-ahead wholesale	3	400	64 environments, 4 steps each	256
M2 real-time balancing	3	200	64 environments, 48 steps each	3 072
M3 ancillary services	3	200	64 environments, 48 steps each	3 072
M4 peer-to-peer local energy	3	400	64 environments, 96 steps each	6 144
M5 local flexibility	3 to 10	early stopping[a], at most 300	64 environments, 24 steps each	1 536

[a] M5 evaluates a checkpoint on the validation days every 10 iterations and stops early once the mean validation return of the last 5 checkpoints exceeds that of the 5 checkpoints before them by 0 to 5%.

F.4 EVALUATION PROTOCOL

M1 to M4 evaluate the policy at the last iteration. In M1 to M3, evaluation takes the mean action of the policy and runs day by day on the 36 evaluation days of the one-year window of case29gb, 2023-07-10 to 2024-07-08 (the dates of the other two test systems are in Table 7). In M4, evaluation also takes the mean action and runs on 1 024 held-out episodes (Table 9). In M5, the checkpoint with the highest validation return is evaluated on the 36 test days with sampled actions (Section G.5.2). The training return is the return under sampled actions and includes exploration noise.

The untrained network is the network of the same parameter layout (PS or NoPS) at iteration 0, evaluated in the same way.

Truthful offers are the reference policy in which every agent offers at its true cost: in M1 to M3 every unit takes the markup $\alpha_i = 1$, with a reserve offer of 0 in M3; in M4 the batteries are kept still, sellers ask the export price and buyers bid the retail tariff; and in M5 every aggregator offers its full deliverable quantity at the replacement cost and plans no charging.

G Additional Market Results

G.1 M1: Day-ahead Wholesale Market

This section first present detailed data, training curves and evaluation results of the three test systems, and then elaborate the finding in Section 5.1 with the supply curves, the offer distribution and the joint sampling probability on the British test system.

G.1.1 Market setting

Each unit i offers a price–quantity curve with a single segment ($K = 1$ in Section C.1.1). Its offer price is $\alpha_i m_i$, where m_i is its segment cost (the mean of its true marginal cost over its output range) and $\alpha_i \in [1, \bar{\alpha}]$ is its markup. At $\alpha_i = 1$ the offer is truthful, and $\bar{\alpha} > 1$ is the markup cap. A markup above one represents economic withholding: the unit offers its capacity at a price above its cost.

We consider three test systems. As shown in Table 7, each test system uses one year of data, of which 36 days are used for evaluation and the rest for training. On each test system, three selected days are taken from the 36 evaluation days: the days at the 10th, 50th and 90th percentiles of total daily demand, referred to below as the low-load day, the mid-load day and the high-load day. Detailed collection, preprocessing and licences of the data are described in Appendix B.

Table 7: Data of the three test systems.

Test system	Grid	Demand data	Time span	Training / evaluation days	Markup cap $\bar{\alpha}$
case29gb	A reduced 29-bus model of the GB transmission network[a] (network diagram[b]), 66 units	GB transmission system demand (NESO)[c], day-ahead forecast from Elexon[d]	2023-07-10 to 2024-07-08	329 / 36	2
case73rts	The RTS-GMLC test system[e], 73 units	Load, wind, solar and hydro of RTS-GMLC[f]	2020-01-01 to 2020-12-31	330 / 36	1.4
case813nem	An open grid model of the Australian National Electricity Market[g], 151 units	Operational demand and day-ahead forecast of the four mainland regions from the Australian Energy Market Operator (AEMO)[h]	2025-02-01 to 2026-01-31	329 / 36	2

[a] https://webhomes.maths.ed.ac.uk/OptEnergy/NetworkData/reducedGB/;
[b] https://webhomes.maths.ed.ac.uk/OptEnergy/NetworkData/reducedGB/GBreducednetwork.pdf;
[c] https://www.neso.energy/data-portal/historic-demand-data; [d] https://bmrs.elexon.co.uk/;
[e] https://github.com/GridMod/RTS-GMLC;
[f] https://github.com/GridMod/RTS-GMLC/tree/master/RTS_Data/timeseries_data_files;
[g] https://github.com/akxen/egrimod-nem-dataset;
[h] https://visualisations.aemo.com.au/aemo/nemweb/index.html.

In the British test system case29gb, generation lies mostly in the north and load mostly in the south, and power flows from north to south: the north has 63% of the total capacity but only 43% of the load. The three example days are 2024-06-23, 2024-03-16 and 2024-01-23. Under truthful offers, 2, 3 and 5 lines are congested on these days, all on central and southern corridors (Figure 12). Of the 66 units, 14 are nuclear, 23 coal and 29 gas. In order of segment cost, all 14 nuclear units are among the 16 cheapest units, and their total capacity is only 4.359 GW. These 16 units have a total capacity of 19.2 GW, while hourly system demand over one year ranges from 17.2 to 47.4 GW and is above 19.2 GW in almost every hour. The nuclear units therefore sit near the left of the supply curve with demand almost always to their right, which means a unilateral markup by one nuclear unit does not move the price, and only a joint markup raises it.

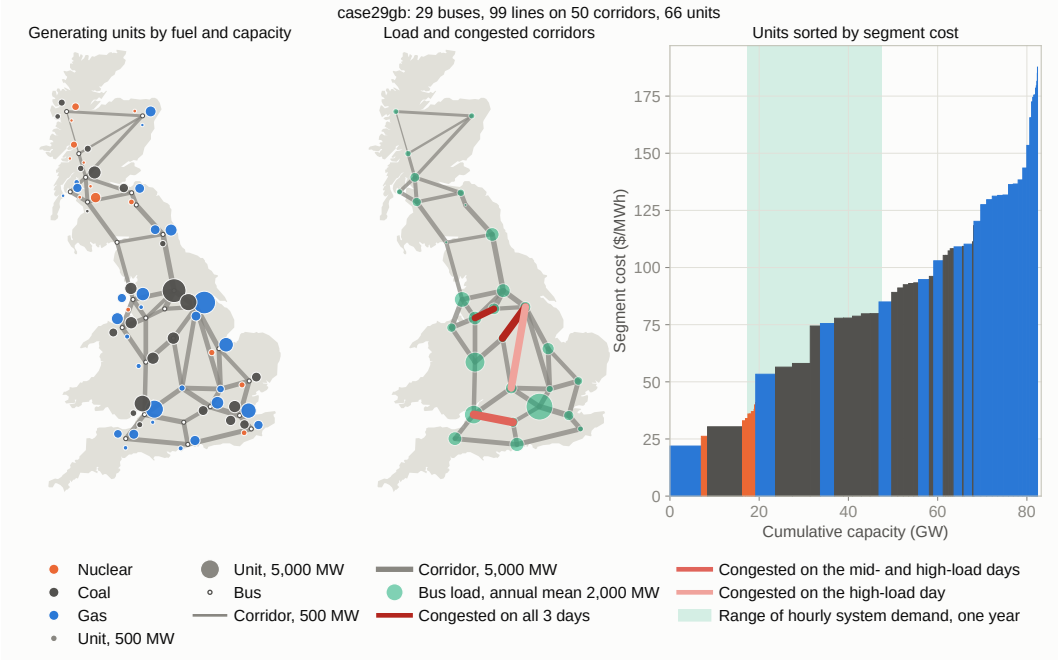


Figure 12: The test system case29gb. Left: the 66 units at their buses, coloured by fuel and sized by maximum output; corridor width scales with capacity. Middle: annual mean bus load and the corridors congested under truthful offers on the three example days. Right: the supply curve ordered by segment cost, against the range of hourly system demand over one year. Generation sits mostly in the north and load mostly in the south, so the congested corridors are central and southern. All 14 nuclear units, 4 359 MW in total, are among the 16 cheapest units, which sum to 19.2 GW; hourly demand ranges from 17.2 to 47.4 GW and exceeds 19.2 GW in almost every hour. The nuclear units therefore sit near the left of the supply curve with demand almost always to their right: a unilateral markup does not move the price, while a joint markup lifts the whole curve and the price with it.

In the RTS-GMLC test system, case73rts, wind, solar and hydro (referred to below as renewables) do not submit offers, and the demand cleared by the market is the net demand (total load minus renewable output). The three example days are 2020-02-24, 2020-03-24 and 2020-07-16. Under truthful offers, 0, 2 and 7 lines are congested on these days (Figure 13).

Regarding the Australian test system, case813nem, the three example days are 2025-09-20, 2025-12-10 and 2025-06-24. Under truthful offers, 3, 4 and 4 lines are congested on these days (Figure 14).

On all three test systems the market clears the demand that actually occurs on the day, while the units observe the day-ahead forecast before they submit offers (Figure 15). On case73rts, net demand has a floor of 2 500 MW, and it sits at that floor in more than half of the hours of the year. On case813nem, demand has a floor of 11 500 MW, reached in only a few hours.

G.1.2 Agent training

The training setup of the four learners is that of Appendix F. On the British test system, the training return of IPPO-NoPS, SAC-PS and SAC-NoPS ends close to its highest level (top row of Figure 16). IPPO-PS stays below the other three and is the only one whose return rises and then falls; the markup it samples also rises to about 1.7 and then returns to about 1.5 (Figure 3c).

On the other two test systems only IPPO-PS and IPPO-NoPS are trained (bottom row of Figure 16). The training return of both learners falls from its starting level. The fall comes from competition among the units: at the start of training every unit offers near the middle of its markup range, which amounts to a joint markup; during training the units that set the price learn to offer closer to cost, because a unit that stays high alone loses its output to its rivals, so the price falls while production cost does not, and profit falls with it.

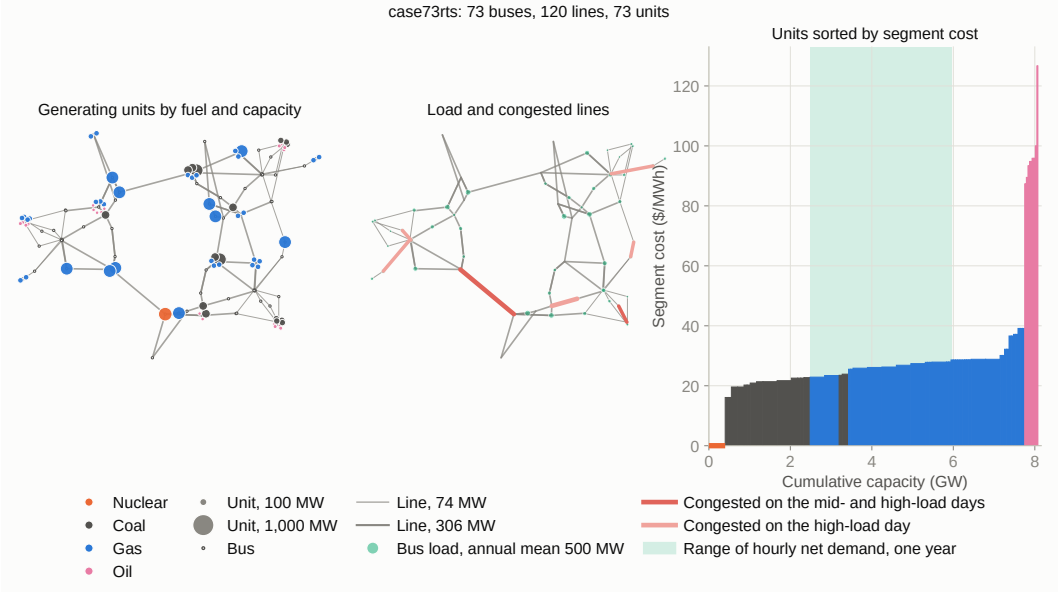


Figure 13: The test system case73rts, drawn as Figure 12: the three-area RTS-GMLC system with 73 units. Left: units by fuel and maximum output. Middle: annual mean bus load and the lines congested under truthful offers on the three example days. Right: the supply curve ordered by segment cost against the range of hourly net demand over one year. Gas units make up most of the supply curve; no line is congested on the low-load day and seven are on the high-load day.

G.1.3 Discussion of results

On each test system, the policies learned by IPPO-PS and IPPO-NoPS are compared with truthful offers on the 36 evaluation days, each learner taken as the mean of three seeds.

As shown in Figure 17, across all three test systems, the mean LMP under the learned policies (load-weighted over all periods and buses of the day) is higher than under truthful offers on every one of the 36 evaluation days (production cost barely changes) and the profit of all units rises accordingly. Averaged over the 36 days, the mean LMP under the policies learned by IPPO-PS and IPPO-NoPS is 1.50 and 1.56 times, respectively, its value under truthful offers on case29gb, 1.10 and 1.07 times on case73rts, and 1.23 and 1.29 times on case813nem. For both learners, the total production cost over the 36 days differs from that under truthful offers by less than 1%. The increase in mean LMP is smallest on case73rts, where the mean markups of IPPO-PS and IPPO-NoPS are only 1.19 and 1.21, respectively.

G.1.4 Details of supply curves

At the peak demand periods of the British test system, the committed units are sorted by offer into a stepped supply curve as shown in Figure 18. Under both truthful and learned offers, the steps of the 14 nuclear units lie far to the left of the demand line. The policy learned by IPPO-PS raises the whole supply curve while the demand line stays in place, and the price rises with it.

G.1.5 Markups of different learners

For each of the four learners the mean markup over the 66 units is near 1.5, but the spread across units differs. IPPO-PS gives almost the same markup to every unit; IPPO-NoPS has the widest spread, with some units near truthful offers and some near the cap; the two SAC learners fall between the two, as shown in Figure 19. For each algorithm, NoPS has a wider spread than PS. All four learners give the 14 nuclear units a mean markup below their own mean over all 66 units, and IPPO-NoPS gives these units markups close to truthful offers.

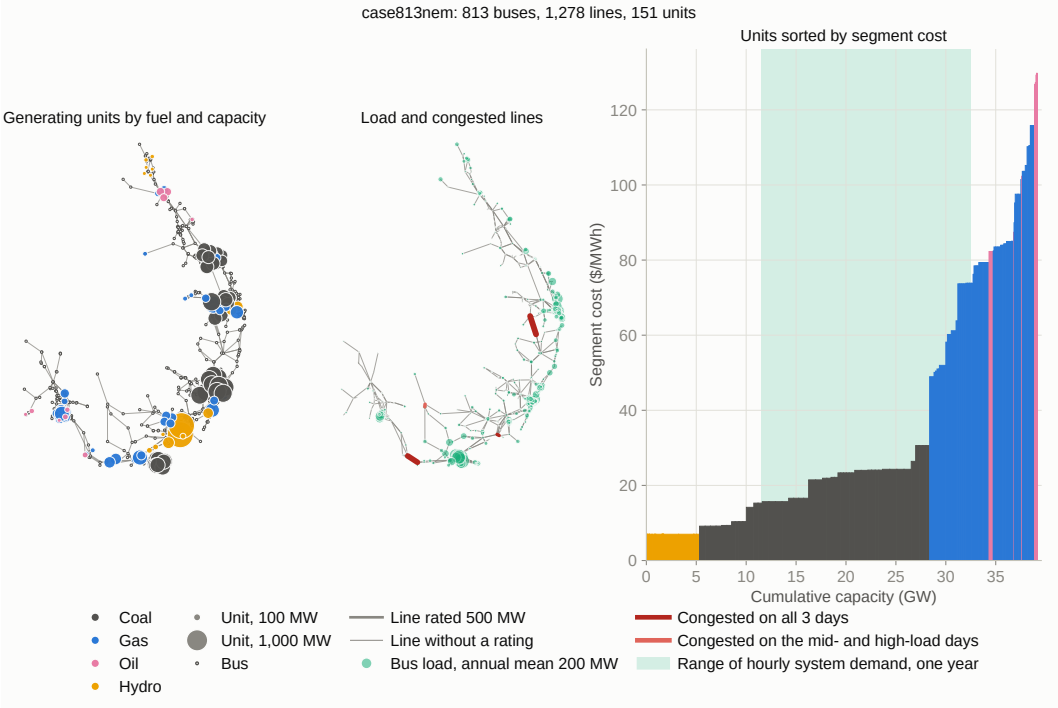


Figure 14: The test system case813nem, drawn as Figure 12: an open grid model of the mainland Australian National Electricity Market with 151 units. Only 7 of its 1 278 lines carry a published rating. Left: units by fuel and maximum output. Middle: annual mean bus load and the lines congested under truthful offers on the three example days. Right: the supply curve ordered by segment cost against the range of hourly demand over one year. Hydro and then coal fill the left of the supply curve; three lines are congested on all three days and a fourth on the mid- and high-load days.

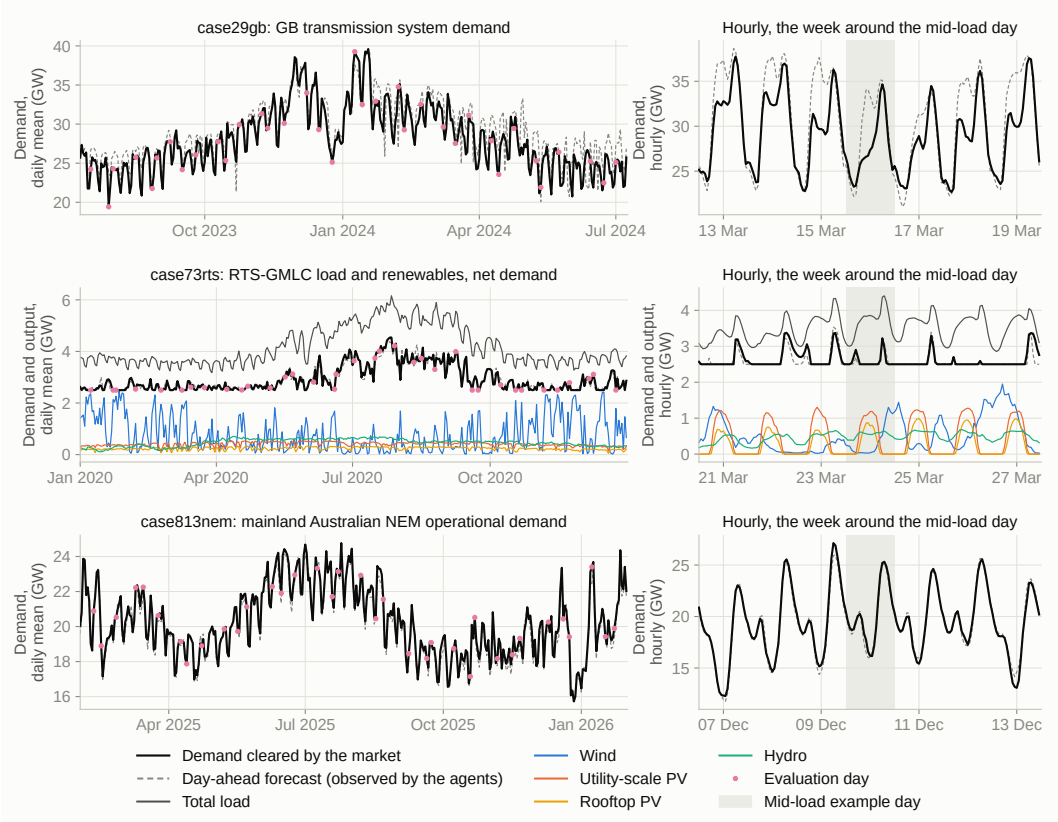


Figure 15: Demand on the three test systems, and the renewable output netted out on case73rts. Left: daily means over the one-year window, with the 36 evaluation days marked. Right: hourly values in the week around the mid-load example day. The thick black line is the demand the market clears; the dashed line is the day-ahead forecast the agents observe. On case73rts the market clears net demand, total load minus the renewable output shown, raised to a floor of 2 500 MW; it sits at that floor in more than half of the hours of the year. Demand on case813nem has a floor of 11 500 MW, reached in a few hours.

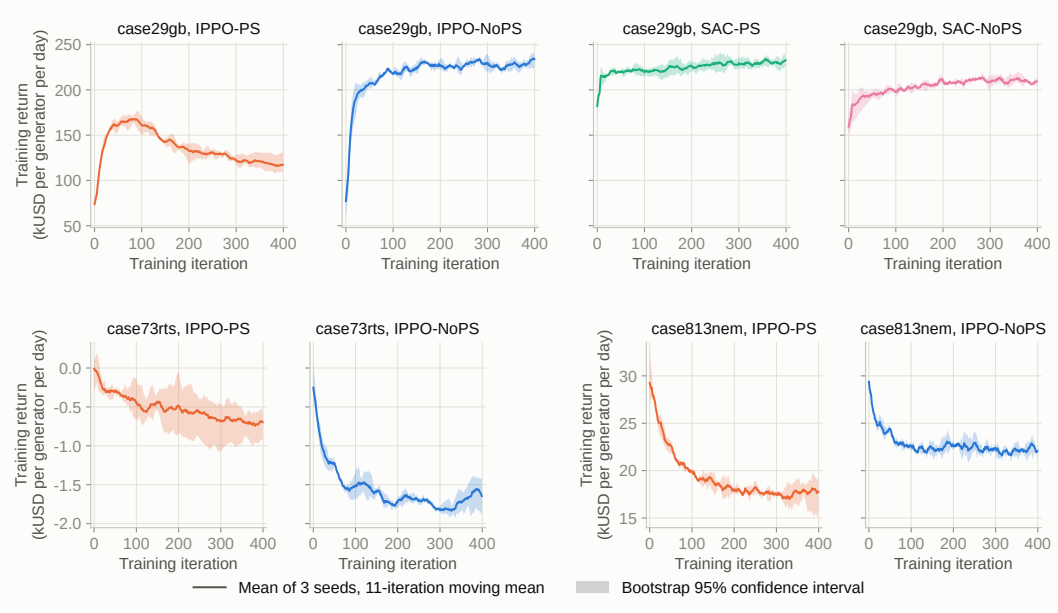


Figure 16: Training return on the three test systems. Top: the four learners on case29gb. Bottom: IPPO-PS and IPPO-NoPS on case73rts and case813nem. Each line is the mean of three seeds, each seed first smoothed by an 11-iteration centred moving mean, and the shaded band is the bootstrap 95% confidence interval. Return is the mean daily profit per generator over all sampled environment steps of an iteration. On case29gb IPPO-PS is the only learner that rises and then falls; on the other two systems the return of both learners falls from its starting level.

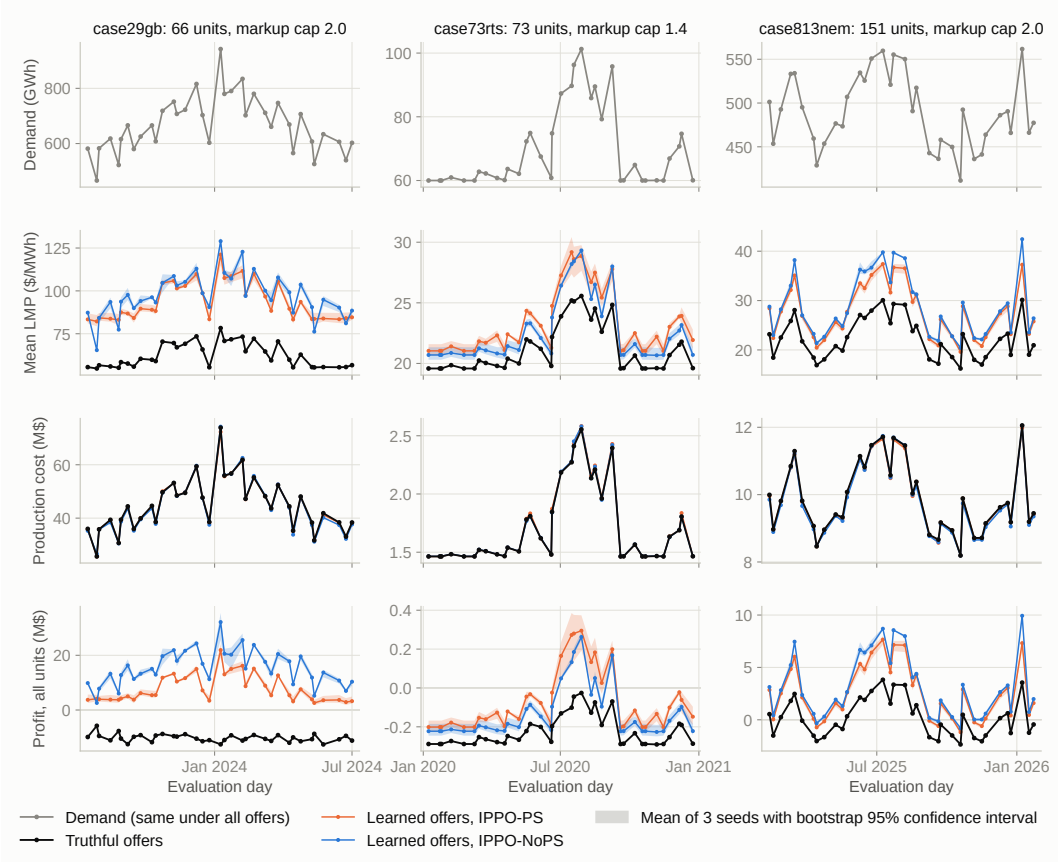


Figure 17: Daily system quantities on the 36 evaluation days of each test system, truthful offers (black) against the learned offers of IPPO-PS (orange) and IPPO-NoPS (blue); each learned line is the mean of three seeds, with a bootstrap 95% confidence interval. Rows: daily demand, the load-weighted mean LMP, production cost, and the total profit of all units.

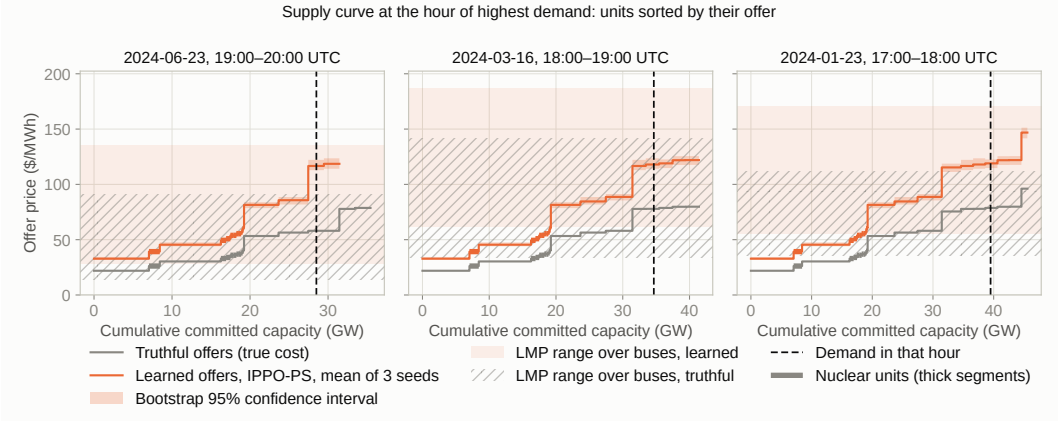


Figure 18: Supply curves at the hour of highest demand on each example day of case29gb, the evaluation days at the 10th, 50th and 90th percentiles of daily demand (2024-06-23, 2024-03-16 and 2024-01-23): committed units sorted by offer under truthful (grey) and learned (orange) offers, with the steps of the 14 nuclear units drawn thicker and the demand in that hour as a dashed line; the shaded bands are the range of LMP over the 29 buses under each set of offers. The learned offers are the mean over three IPPO-PS seeds at the end of training, with a bootstrap 95% confidence interval for each step; the learned LMP band is the range over buses of the seed-mean LMP. The nuclear steps lie far to the left of demand under both sets of offers, and the learned offers raise the whole curve.

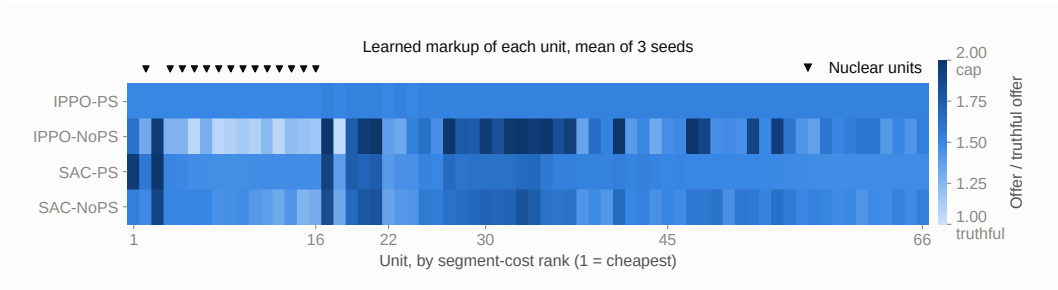


Figure 19: Markup of each unit under the four learners on case29gb; rows are learners, each the mean over three seeds, columns the 66 units ordered by segment cost. Colour is the markup of that unit, averaged over the 36 evaluation days and the three seeds, on the full action range from 1.00 (truthful) to 2.00 (cap). Black triangles above the top row mark the 14 nuclear units.

G.2 M2: Real-Time Balancing Market

In this section, we discuss the result of the four learners in the RTM balancing market.

G.2.1 Market setting

The action of unit i is a markup $\alpha_i \in [1, \bar{\alpha}]$ on its generation cost (i.e., true marginal cost over its output range); $\alpha_i = 1$ is a truthful offer, and an offer is submitted once per half-hour period. The test systems of this market are case29gb and case73rts (detailed data in Table 7), with the markup cap $\bar{\alpha}$ set to 2. The grids and units are shown in Figures 12 and 13.

Most output is scheduled day ahead, so real-time offers price only small deviations: on the 36 evaluation days, the median absolute half-hourly demand deviation is 214 MW, less than 1% of mean demand (Figure 20). This limits the gain from raising an offer alone (Section G.2.4).

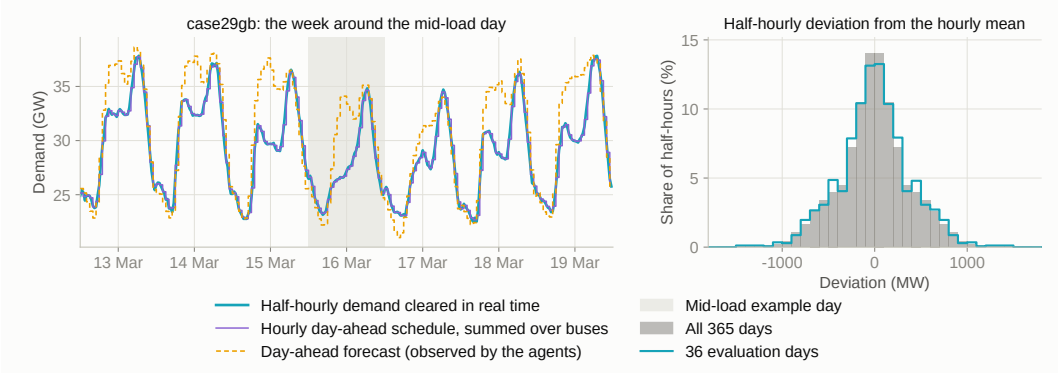


Figure 20: Demand on case29gb. Left: the week around the mid-load day (2024-03-16): the half-hourly demand the real-time market clears (teal), the hourly day-ahead schedule summed over buses, which equals the hourly mean of that demand (purple steps), and the day-ahead forecast the units observe (dashed yellow). Right: the deviation of the demand in each half-hour from the mean of its hour, over all 365 days (grey) and the 36 evaluation days (teal). On the evaluation days the absolute deviation has a median of 214 MW, a 90th percentile of 638 MW and a maximum of 1 478 MW, against a mean demand of 27 730 MW. The day-ahead schedule follows the half-hourly demand to within a few hundred MW.

G.2.2 Agent training

The training return is the mean daily profit per unit under sampled actions, which is the objective of the learner itself. Under all four learners the mean markup of the 30 committed units starts between 1.48 and 1.61; at the end of training it is 1.04 under IPPO-PS, and 1.24, 1.40 and 1.39 under IPPO-NoPS, SAC-PS and SAC-NoPS; the markup along training is shown in Figure 4b. On case29gb the training return of IPPO-PS falls throughout and ends lowest of the four learners; IPPO-NoPS and SAC-PS also fall, by less; SAC-NoPS rises first and then stays level (top row of Figure 21). On case73rts IPPO-PS and IPPO-NoPS are trained, and the training return of both rises slowly (bottom row of Figure 21).

G.2.3 Discussion of results

On case29gb, IPPO-PS returns to truthful offers, while the other three learners end with higher total profit and lower system cost, as shown in Figure 22. This position is not learned. The untrained networks already give the units different markups, and training only moves these markups towards truthful offers. The reason is that an independent learner sees only how its own return responds to its own action. A unit that raises its offer alone only loses output, as shown in section G.2.4. If its output depends on its offer, its gradient points towards truthful offers; if not, its gradient is zero. Without parameter sharing, each unit follows its own gradient: the first kind returns to truthful offers and the second stays where it started. With sharing, the updates from the first kind move all units, so every markup falls together.

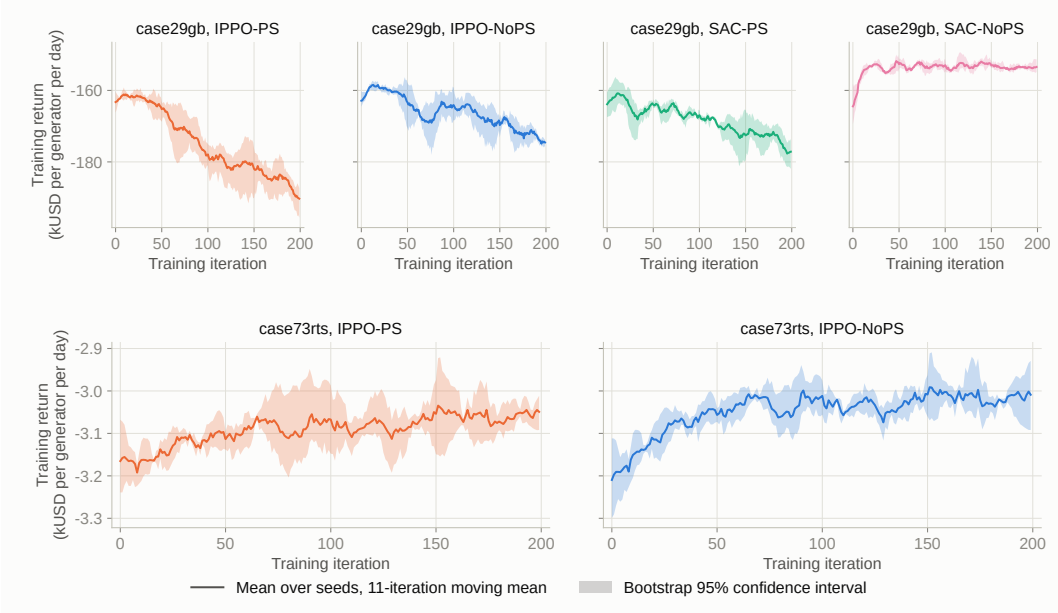


Figure 21: Training return on two test systems in the real-time market. Top: the four learners on case29gb. Bottom: IPPO-PS and IPPO-NoPS on case73rts. Each line is the mean over seeds, smoothed by an 11-iteration moving mean, and the shaded band is the bootstrap 95% confidence interval of the mean. Return is the profit of a unit per day, averaged over units and over the environment steps of an iteration. On case29gb the return of IPPO-PS falls the most, and SAC-NoPS is the only learner whose return rises. On case73rts the return of both learners rises slowly.

There are 36 units that are never committed, which give a clean test. Their offers never enter the clearing, so their own return carries no signal. Without information sharing, training leaves their markups unchanged (IPPO-NoPS from 1.48 to 1.49, SAC-NoPS from 1.50 to 1.51). With sharing, they fall with the committed units (IPPO-PS from 1.52 to 1.03, SAC-PS from 1.57 to 1.47). SAC-PS falls much less, because under SAC the committed units themselves stay far from truthful offers (Section 5.2). These results show an information limit in electricity markets: a unit learns a bidding strategy only when its offer changes its own outcome. Otherwise, its learned offer comes from the initialization (without parameter sharing) or from the other units (with it), and should not be read as a strategy.

G.2.4 Unilateral and joint markups

We extend the discussion in Appendix G.2.3 in this section.

A unit gains little by raising its offer alone. We let each of the 30 committed units raise its offer alone, at 21 markup levels from 1.00 to 2.00, while the other 65 units offer truthfully as shown in the left subfigure of Figure 23. For 19 units, the most profitable level is the truthful offer, and no unit gains more than \$0.23 million per day. At a markup of 2.00, 27 of the 30 units lose profit, by up to \$8.0 million per day. The unit that raises its offer produces less, and the other units take over both its output and its lost profit (Figure 23, right, slope -0.97). A unilateral markup therefore moves profit between units but leaves the total almost unchanged.

Raising all offers together helps only if the markups differ across units. With the same markup for all 66 units, the order of the offers and the dispatch schedules do not change as shown in Figure 24. The reason is the two-settlement rule equation 18. Since the day-ahead market clears the actual hourly demand, output below the schedule in one half-hour is roughly offset by output above it in the other. A higher real-time price therefore moves money between units without raising the total. Different markups, in contrast, change the dispatch, raise total profit and lower system cost, as under IPPO-NoPS, SAC-PS and SAC-NoPS (as shown in Figure 22). This joint gain appears only when units offer differently, and no single unit sees it in its own profit.

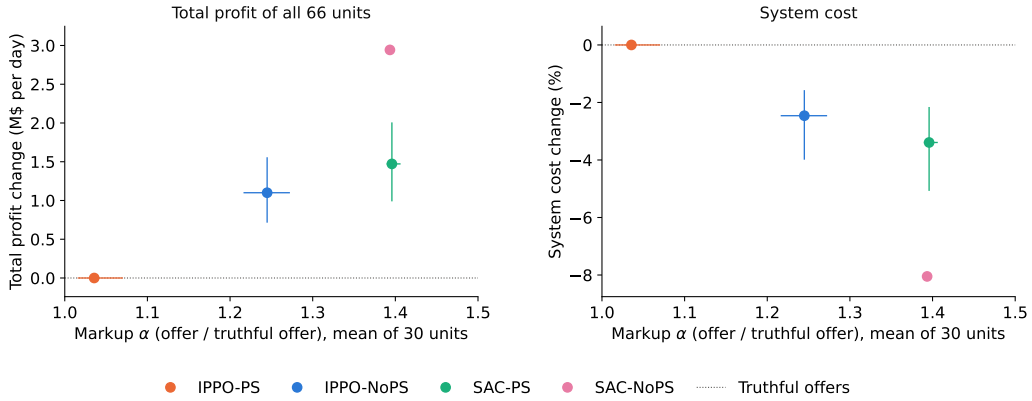


Figure 22: Learned offers against truthful offers on case29gb over the 36 evaluation days. Left: change in total profit of all 66 units compared to truthful offers (in million dollars per day); right: change in system cost compared to truthful offers (in percent). Each circle indicates one learner after training, placed at the mean markup of the 30 committed units. The bars are the 95% confidence interval of the mean over seeds; the dotted line is truthful offers. IPPO-PS returns to truthful offers; the other three learners end with higher total profit and lower system cost.

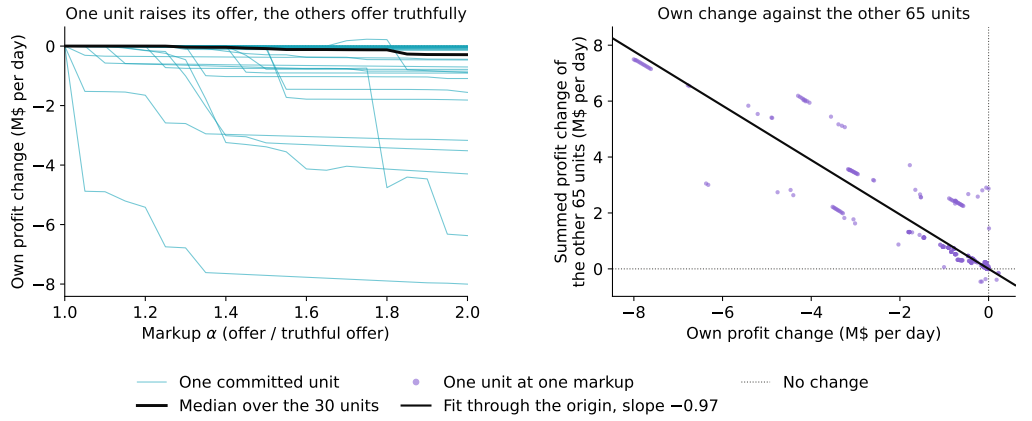


Figure 23: Each of the 30 committed units raises its offer alone while the others offer truthfully. Left: own profit change against markup (one line per unit, median in black). Right: own profit change against the profit change of the other units (with a least-squares fit, near-zero points are omitted). Means over the 36 evaluation days, in million dollars per day.

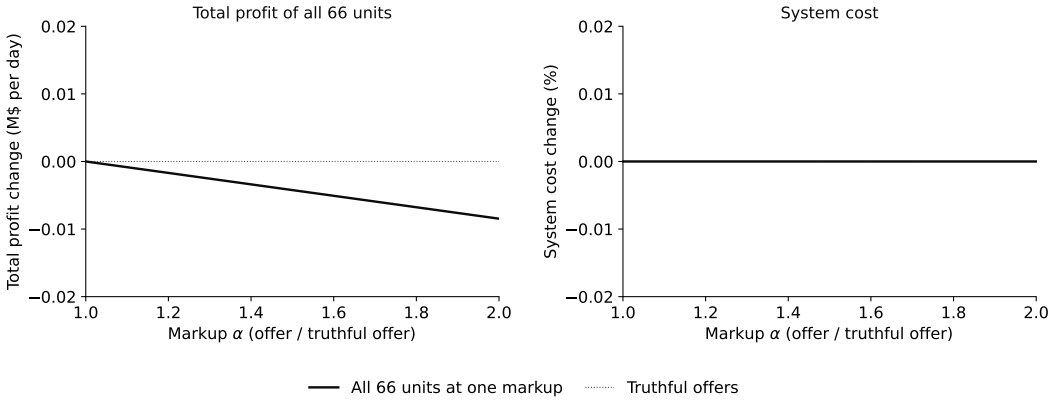


Figure 24: All 66 units at the same markup, from 1.00 to 2.00. Left: change in total profit (million dollars per day). Right: change in system cost (%). Both are relative to truthful offers (dotted line), averaged over the 36 evaluation days. A common markup leaves both nearly unchanged.

G.3 M3: Ancillary Services Market

This section first gives the data, training curves and the discussion about results of the ancillary services market.

G.3.1 Market setting

We consider the setting in which M2 and M3 are jointly cleared. Therefore, the action of each unit i has two parts: an energy offer with markup $\alpha_i \in [1, \bar{\alpha}]$, and one reserve offer $\pi^{\text{res}}_{i,j,t}$ per period t on each of the two reserve products $j = 1, 2$, in £/MWh. The two products require a response within 10 and 30 minutes respectively. The reserve a unit is awarded on a product, called its reserve held, is at most the output it can ramp up within that time. The reserve requirement of each product is 5% of the demand forecast of the period. Unmet reserve requirement is priced at VOLR (as shown in Table 8), and the reserve price $\lambda^{\text{res}}_{j,t}$ does not exceed it.

This market uses the same three test systems as the day-ahead market: case29gb, case73rts and case813nem. Table 7 describes their data, and Table 8 gives the markup cap and the value of lost reserve for each system. Figures 12 to 14 show the grids and units, and Figure 15 shows the demand.

Table 8: Markup cap and value of lost reserve of the ancillary services market on the three test systems.

Test system	Markup cap $\bar{\alpha}$	Value of lost reserve VOLR (£/MWh)
case29gb	2	250
case73rts	2	136
case813nem	2	147

G.3.2 Agent training

The four learners are trained as described in Appendix F, and Figure 25 shows their training return. On case29gb test system, IPPO-PS is the only learner whose return falls, from -106 k £per generator per day over the first 5 iterations to -126 over the last 20. The other three rise slightly and stay close to where they started. In test systems case73rts and case813nem, the return of both IPPO learners rises.

G.3.3 Discussion of results

Training does not make the learners more profitable in this market. The untrained networks, with random reserve offers, already earn £2 million to £4.5 million per day more than truthful offers as shown in Figure 26. IPPO-PS gives almost all of this back: after training, it earns only £0.56 million per day more than truthful offers, £3.95 million less than its untrained network. The other three learners stay near their starting point, at £2.71 million to £3.69 million per day above truthful offers. No learner receives more gains compared to its own untrained network.

A much larger gain is available, but no learner finds it. If all units raise their reserve offers together to £150/MWh, total profit rises by £9.05 million per day as shown in the subfigure c of Figure 4. Only £0.51 million to £1.63 million per day of the learners' profit comes from reserve payments, and the rest comes from energy. Production cost stays within 0.6% of its truthful level as shown in the subfigure b of Figure 26.

The other two test systems show the same results, but the differences are small: all changes in profit are within a few tens of thousands of dollars per day as shown in subfigures c and d of Figure 26. After training, IPPO-PS again earns the same as or less than before training, while IPPO-NoPS earns slightly more.

G.3.4 Reserve offer analysis

In ancillary service markets, a unit gains almost nothing by raising its reserve offer alone, but all units gain a lot by raising it together as shown in Figure 27. When one of the 29 committed units raises its reserve offer from 0 to £150/MWh while the others offer truthfully, its own profit usually

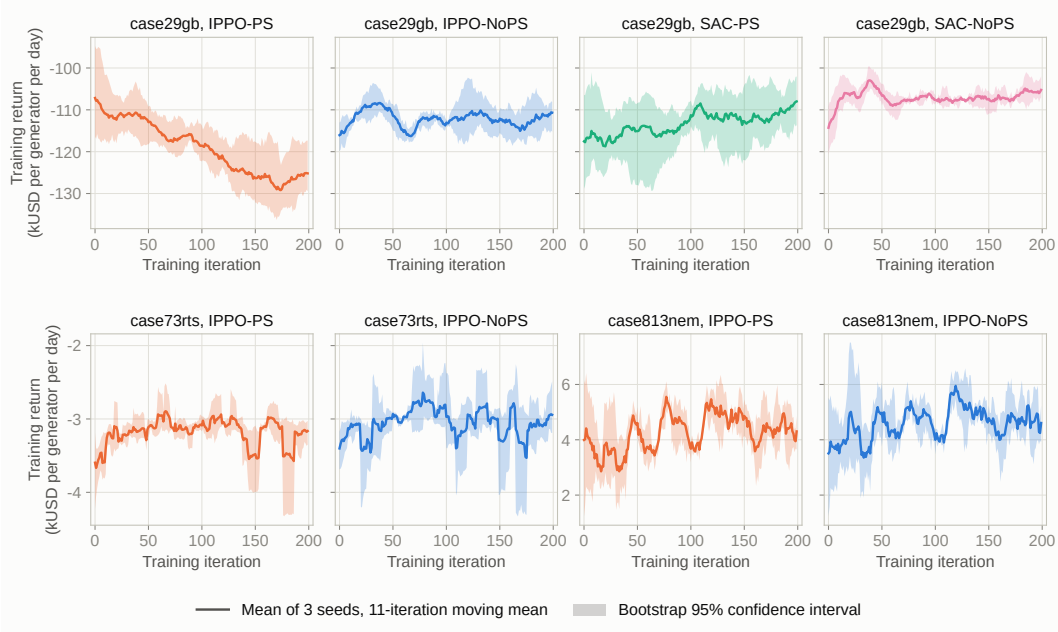


Figure 25: Training return on the three test systems. Top: the four learners on case29gb. Bottom: IPPO-PS and IPPO-NoPS on case73rts and case813nem. Return is the mean daily profit per generator in each iteration. Lines are the mean of three seeds, each smoothed over 11 iterations, and bands are bootstrap 95% confidence intervals. On case29gb, only the return of IPPO-PS falls, and on the other two systems, both learners rise.

does not change. Across the 29 units, the median change is zero, the largest gain is £0.08 million per day, and the largest loss is £0.32 million per day. When all 66 units raise their reserve offers together, total profit rises linearly with the offer, by £9.05 million per day at £150/MWh, while production cost changes by less than 0.001%. evaluation days).

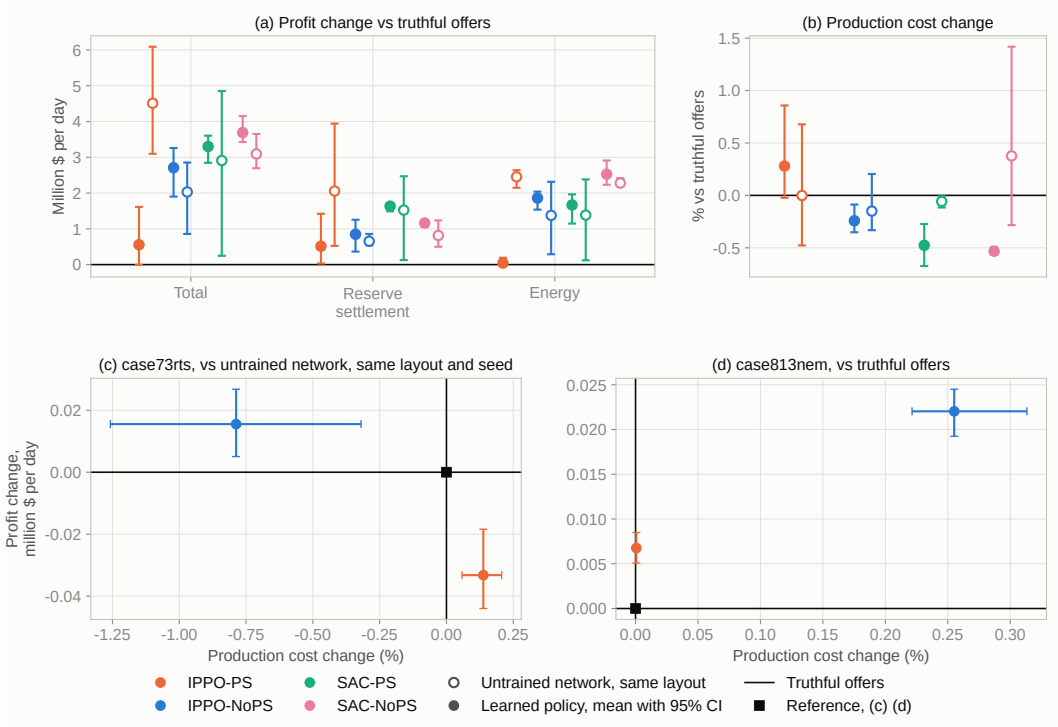


Figure 26: Evaluation on the 36 evaluation days. (a) case29gb: change in total profit of all 66 units relative to truthful offers (million dollars per day), split into reserve payments and energy. (b) case29gb: change in production cost. Filled markers are trained policies; hollow markers are untrained networks. (c) case73rts, relative to the untrained network (truthful offers not evaluated), and (d) case813nem, relative to truthful offers: change in total profit against change in production cost for IPPO-PS (orange) and IPPO-NoPS (blue). The black square marks the reference. Markers are seed means with bootstrap 95% confidence intervals. On case29gb, all learners earn more than truthful offers at nearly unchanged cost, but only IPPO-PS earns less than its untrained network; on the other two systems, the differences are small.



Figure 27: Raising the reserve offer on case29gb from 0 to £150/MWh on both energy and reserve products. Teal: own profit change of each of the 29 committed units when it raises its offer alone, with the others truthful. Black: the median. Orange: total profit change of all 66 units when all raise their offers together. Values in million dollars per day. Raising alone gains almost nothing, while raising together gains £9.05 million per day at £150/MWh.

G.4 M4: Peer-to-Peer Local Energy Market

Section 5.3 finds that an arbitrage that pays for a single household (charging at noon and discharging in the evening) cancels itself when many households adopt it, and that whether a learner finds it depends on the parameter layout. This section first describes the community and its metering data (Section G.4.1), then gives the training curves (Section G.4.2) and the evaluation results on held-out episodes (Section G.4.3); the last section covers how the gain from arbitrage changes with the number of households arbitraging (Section G.4.4).

G.4.1 Market setting

The test system at community level has the 1 200 households with PV in the Fluvius 2024 metering data, as shown in Figure 28. Each household is considered as one agent. Each period is a quarter-hour, and an episode is the 96 periods of one day. The smart meters record only the injection and the offtake of each household, and the market takes their difference as the net position of each household. The export price is $\pi^{\mathrm{exp}} = 73.0$ EUR/MWh and the retail tariff is $\pi^{\mathrm{ret}} = 333.4$ EUR/MWh. Each household has one battery with a capacity of 0.011 MWh, a round-trip efficiency of 0.85 and a battery degradation cost of 13.88 EUR per MWh of throughput; its state of charge (as a fraction of capacity) stays between 0.15 and 1.0 and starts at 0.5. Appendix B describes how the data were collected and preprocessed.

In every period, each household chooses a battery power (charge or discharge) and a price between the two grid prices. A household with a positive net position sells, and the price is its ask; one with a negative net position buys, and the price is its bid. As shown in Figure 29, at noon most households inject more than they take off and are sellers: around 15:00 about 80% households are sellers, and at night injection is close to zero.

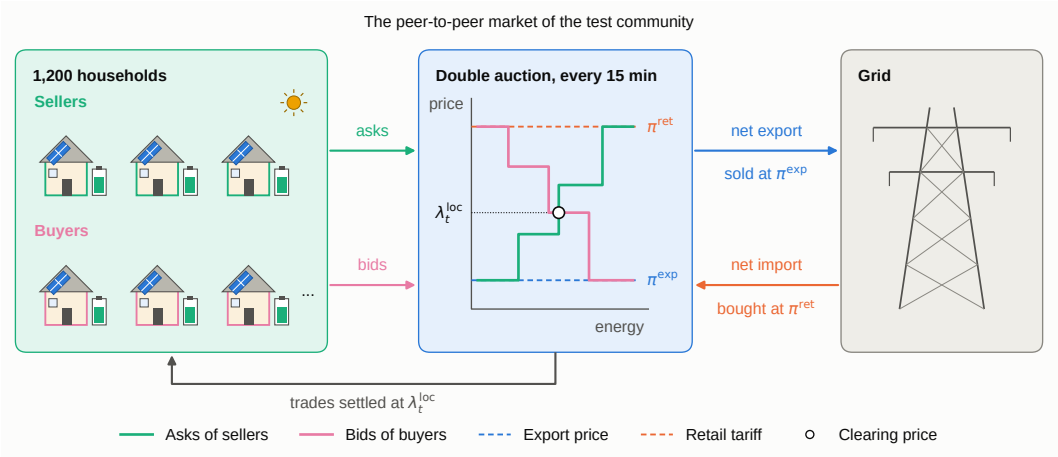


Figure 28: The P2P market of the test community. Each of the 1 200 households has PV and a battery. In every quarter-hour, sellers ask and buyers bid in a community double auction, which sets the clearing price λ_t^{loc} between the export price π^{exp} and the retail tariff π^{ret}; trades in the community are settled at λ_t^{loc}. Energy the auction leaves unmatched is settled with the grid.

G.4.2 Agent training

The training setup of the four learners is that of Appendix F, and the two SAC learners differ from Table 5 only in using two hidden layers of 64 units, a replay buffer of 8 192 environment steps, a batch size of 16 environment steps and one gradient step every 4 environment steps. The training return of both IPPO learners rises fastest in the first 50 iterations, and both SAC learners level off within the first few dozen iterations. After that the return changes little, as shown in Figure 30: from iterations 100–149 to iterations 351–400, the mean return of each IPPO learner rises by only 0.0009 to 0.0027 EUR per household per quarter-hour.

Table 9: Data of the test community.

Data source	Households	Period covered	Training and held-out	Licence
Quarter-hour injection and offtake from Fluvius smart meters[a]	1 200, all with PV	2024-04-01 to 2024-10-26, 209 days	3 consecutive days held out of every 15; 14 702 training and 2 702 held-out episode starts; evaluation on 1 024 held-out episodes, of which the figures below use 64	Fluvius open data licence

[a] `opendata.fluvius.be/explore/dataset/1_50-verbruiksprofielen-dm-elek-kwartierwaarden-voor-een-volledig-jaar/`.

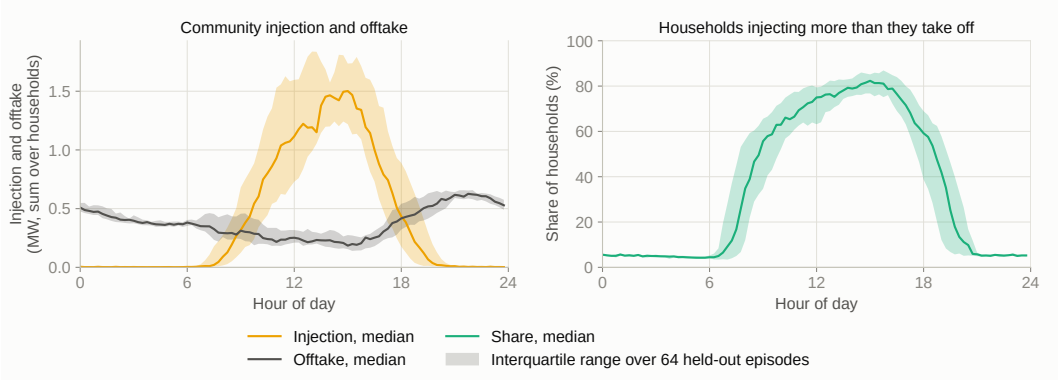


Figure 29: The test community of 1 200 households: community injection and offtake (left) and the share of households injecting more than they take off (right), by hour of day over the 64 held-out episodes.

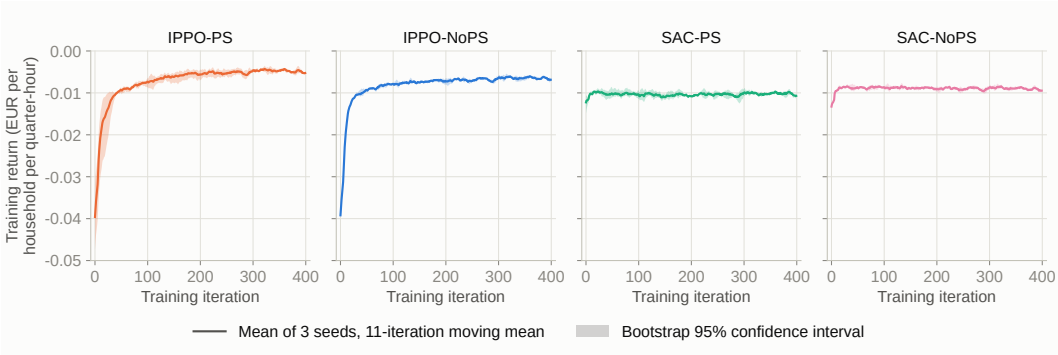


Figure 30: Training return of the four learners up to iteration 400; the policy at iteration 400 is the one evaluated throughout this section. Shown is the mean over three seeds of the 11-iteration moving mean of the return on 64 sampled training episodes, with the shading the bootstrap 95% confidence interval. Both IPPO learners rise fastest in the first 50 iterations and both SAC learners reach their level within a few dozen; after iteration 50 all four change little.

G.4.3 Discussion of results

We evaluate the results under truthful offers and under the considered learners. Truthful offers mean that households with batteries stay idle, and sellers ask the export price and buyers bid the retail tariff. As shown in Figure 31, IPPO-NoPS is the most consistent learner: all three of its seeds lower the clearing price and raise the profit per household on every one of the 64 episodes. IPPO-PS and SAC-NoPS are less consistent, with all three seeds earning more than under truthful offers on 54 and 57 episodes, respectively. Averaged over the 64 episodes, the profit per household is -1.201 EUR per episode under truthful offers and -0.769 to -0.685 EUR over the three seeds of IPPO-PS, the

highest of the four learners. The three seeds of SAC-PS disagree on the clearing price: the mean clearing price of two of them is 270.3 and 272.9, against 238.1 EUR/MWh under truthful offers.

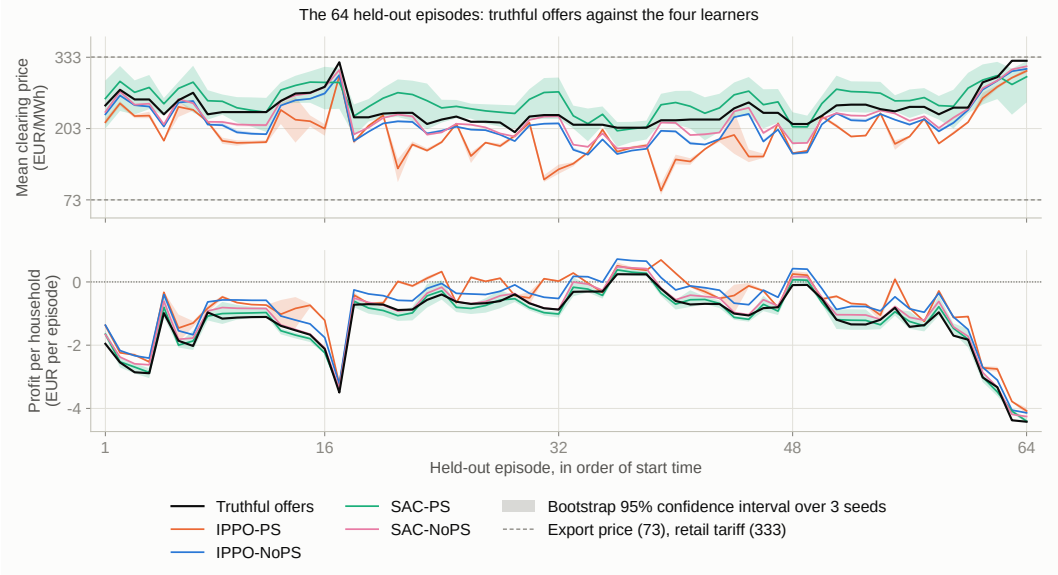


Figure 31: Results on the 64 held-out episodes ordered by start time. Top: mean clearing price per episode. Bottom: profit per household per episode. Black is truthful offers; each learner is the mean of three seeds with a bootstrap 95% confidence interval. IPPO-NoPS lowers the price and raises the profit on every episode, while IPPO-PS and SAC-NoPS gain on 54 and 57 episodes. The seeds of SAC-PS disagree, and two of them raise the price.

Figure 32 shows the actions each learner learns. The state of charge in each period of the day (the last 12 hours of each episode only: in every episode the learned policies first sell the energy stored at the initial state of charge of 0.5) splits the different learners into two behaviours according to whether they share parameters: IPPO-NoPS and SAC-NoPS charge in the hours of PV surplus and discharge in the evening, IPPO-NoPS by a large amount and SAC-NoPS by a small one; IPPO-PS and SAC-PS stay near the lowest state of charge. The learners also differ in their bidding prices. Under both IPPO learners, most sellers ask a price close to the export price. Buyers lean towards the retail tariff under IPPO-PS, but bid at all levels under IPPO-NoPS. Under SAC-NoPS, most asks and bids are close to one of the two grid prices, and few lie in between. The three seeds of SAC-PS learn very different prices, so SAC-PS has the widest confidence intervals in the figure.

G.4.4 Arbitrage gain and participation

This section asks how the gain from arbitrage changes with the number of households arbitraging at the same time. The fixed schedule of Section 5.3 charges at 2.62 kW from 11:00 to 15:00 and discharges at 2.62 kW from 18:00 to 22:00, with all households offering truthfully; it is replayed on the 64 held-out episodes with k households on the schedule and the rest idle (Figure 33). The gain is the change in profit per household per episode (EUR) relative to all batteries idle.

Arbitrage pays only when few households take part, as shown in the left subfigure in Figure 33. A single arbitraging household gains 1.129 EUR per episode, but the gain falls as more households join, turns negative at 210 to 240 households, and reaches -1.248 EUR when all 1 200 take part. The households that keep their batteries idle benefit instead: with 600 households arbitraging, each idle household gains 0.573 EUR.

More specifically, arbitrage closes the price spread it relies on as shown in the right subfigure in Figure 33: charging at noon raises the noon price, and discharging in the evening lowers the evening price. Once about 240 households take part, the remaining spread no longer covers the round-trip losses and the degradation cost. This saturation comes from the market, not from the learner. For a single agent (household) the price is exogenous, but as more agents follow the same strategy, their

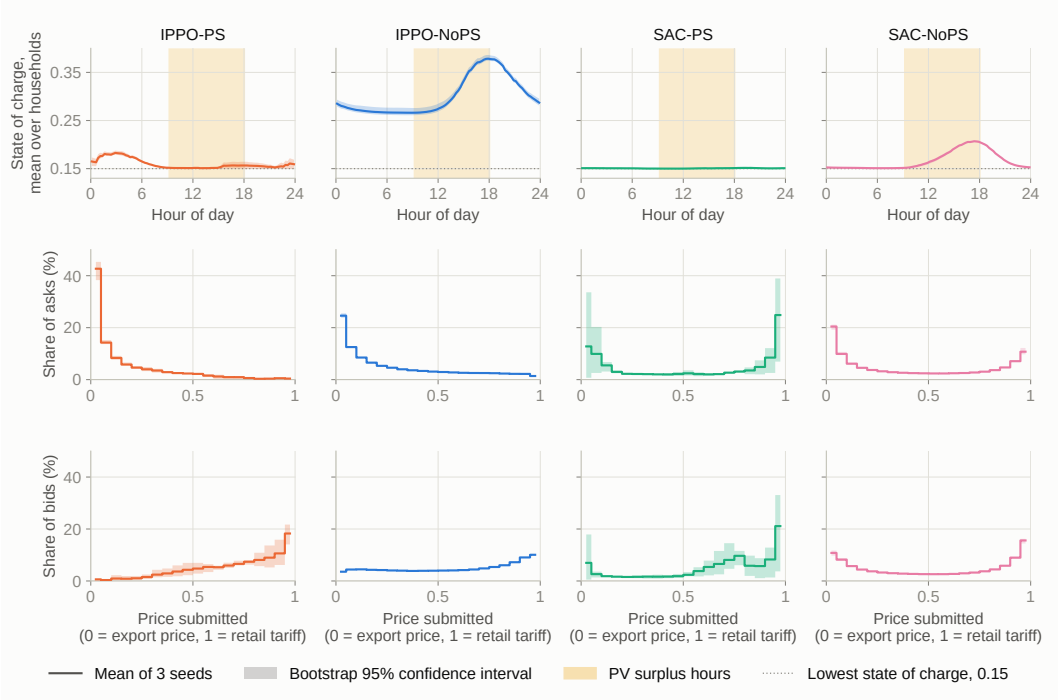


Figure 32: Learned actions of the four learners on the 64 held-out episodes (mean of three seeds, bootstrap 95% confidence interval). Top: state of charge averaged over households by hour of day (last 12 hours of each episode only). The band marks the PV surplus hours, those in which the community injects more than it draws on most days. Middle and bottom: asks and bids, scaled from the export price (0) to the retail tariff (1).

actions change the price, which becomes endogenous. A profit measured for one agent, or against historical prices, therefore overstates what the strategy earns when many adopt it.

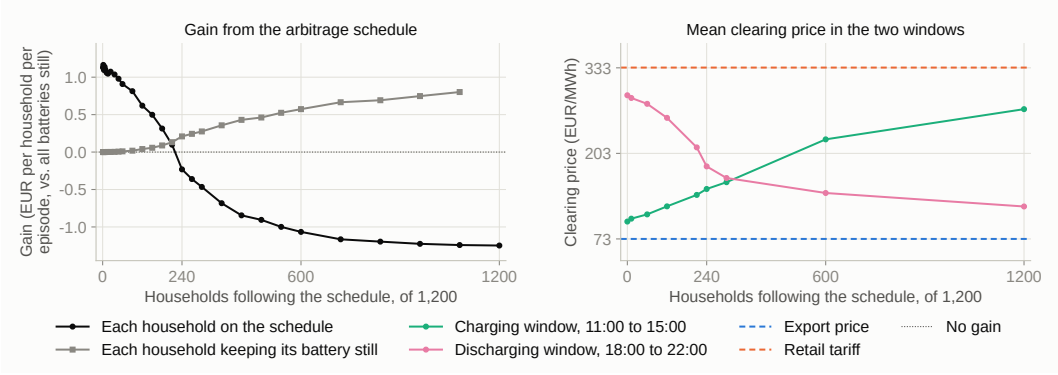


Figure 33: On the 64 held-out episodes, k households follow a fixed arbitrage schedule (charge 11:00–15:00, discharge 18:00–22:00) and the rest keep their batteries idle. Left: gain per household, for those arbitraging and those idle, relative to all batteries idle. Right: mean clearing price in the charging and discharging windows. The horizontal axis of both panels is k. Arbitrage pays while few households take part and turns into a loss at 210 to 240 households, before the two window prices cross.

G.5 M5: Local Flexibility Market

In this appendix section, we provide detailed market simulation settings and the discussion of the performance of the four learners.

G.5.1 Market setting

The test system is the Swiss distribution feeder 459_0, as shown in Figure 34). The system has 129 buses and 128 lines, and we scale its load by 1.50. We consider an episode as one day, and the market is cleared hour by hour. Each aggregator operates one battery, and aggregator i submits three quantity bids at period t, including a price $\pi_{i,t}$, an offered quantity, and planned charging. The distribution system operator (the operator) clears the market by minimising the cost of satisfying demand, as detailed in Appendix C.5. each winning aggregator is paid its own offer. The price floor is set to the replacement cost $c^{\text{rep}} = 149.88$ CHF/MWh.

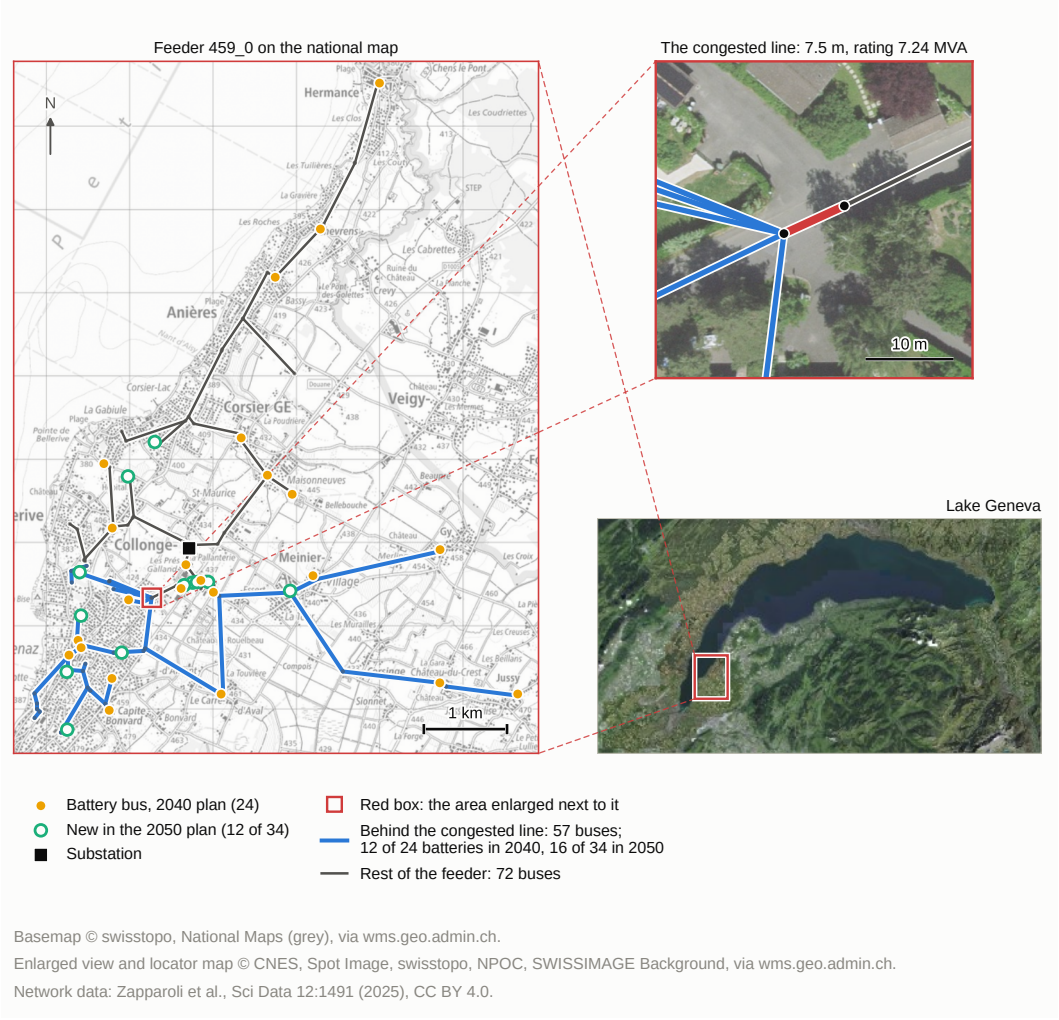


Figure 34: Feeder 459_0 in Switzerland. Bottom right: its location in the Lake Geneva region. Left: the feeder, with a small red box marking the congested line. Top right: an aerial image of the congested line. Lines behind the congested line are blue, and only batteries on these lines can relieve it.

The system requires a the bus voltage stays above 0.9 per unit, and on the flow of every line keeps within maximum value. The operator buys flexibility to keep voltage within its range and the congested line within its rating. The requirement is how much the flow on the line would exceed 98%

of its rating if the operator bought nothing. This flow includes the load and the charging the aggregators plan, so the aggregators can raise the requirement themselves as detailed in Section G.5.3. In the Swiss test system, voltage stays within its limits all year, so the line is the only reason to buy.

The flexibility market trades much less than the P2P energy market, because it buys only what is needed to keep the network within its limits. With all batteries idle, it buys on only 9 of the 36 test days, and never more than 0.43 MW, as shown in the left subfigure of Figure 35. Only batteries downstream of the line can relieve it: 12 of 24 in the 2040 placement and 16 of 34 in the 2050 placement. The 12 batteries alone can supply 2.51 MW, almost six times over the largest requirement as shown in the right subfigure of Figure 35. Supply therefore far exceeds demand.

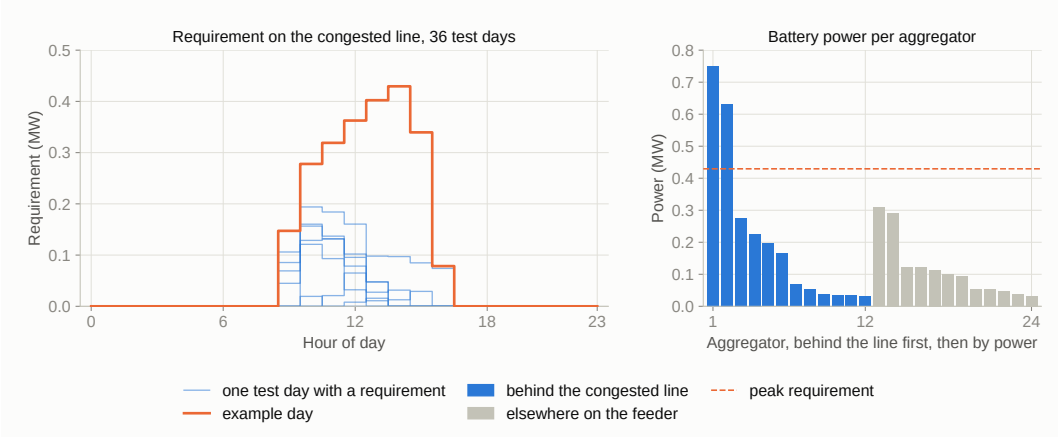


Figure 35: Demand and supply of flexibility. Left: the requirement in each hour of the 36 test days, with all batteries idle. Right: the power of each of the 24 batteries in the 2040 placement, with the 12 downstream of the congested line first.

Table 10: Overview of Swiss test system data.

Data	Source	Licence	Training days	Validation days	Test days
Swiss distribution feeder 459_0	SwissDN[a], Zapparoli et al. (2025)[b]	CC BY 4.0	293 days	36 days (the 5th, 15th and 25th of each month)	36 days (the 1st, 11th and 21st of each month)
Energy price (category C2, 2026)	ElCom[c]	opendata.swiss Open use	—	—	—

[a] https://doi.org/10.5281/zenodo.15056134; [b] https://doi.org/10.1038/s41597-025-05830-y; [c] https://energy.ld.admin.ch/elcom/electricityprice.

G.5.2 Agent training

The training setup is that of Appendix F, and IPPO differs from Table 5 only in the following: each iteration takes 4 full-batch gradient steps, the learning rate of 3×10^{-4} decays linearly to zero over 300 iterations, the log standard deviation of exploration is annealed linearly from -0.5 to -3.0 over the same span, and the initialisation scale of the actor output layer is 0.01 rather than 1.0. Training uses the training days of Table 10; the stopping rule and the checkpoint evaluated on the test days are given in Table 6. On the test days the selected checkpoint acts with sampled actions, IPPO at a log standard deviation of -3.0, the end point of its annealing, and SAC at the log standard deviation of its own actor.

G.5.3 Analysis of the procurement requirement

Under truthful offers, the operator buys flexibility in about 5% of periods: 43 of 864 hours across the 36 test days in the 2040 placement, as shown in Figure 35. All four learners increase this share.

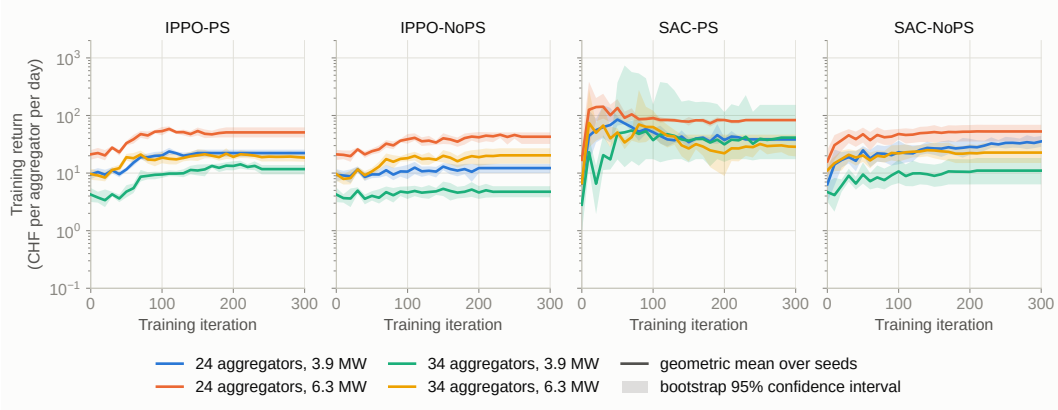


Figure 36: Training return of the four learners against training iteration; lines are the geometric mean over the seeds of each configuration and shaded bands the bootstrap 95% confidence interval, each seed held at its last value after it stops. All four learners reach a plateau within about 100 iterations.

More than 80% of daily procurement lies above what truthful offers buy, and aggregator returns track procurement as shown in Figure 37. The excess comes from planned charging, which raises baseline flow on the congested line above 98% of its rating and leads the operator to buy flexibility from batteries behind it. Cleared prices remain near the floor: on the test days, all IPPO seeds and 75% of SAC seeds clear within 1% of it.

In the two 3.9 MW configurations, replacing truthful-offer aggregators with SAC-PS learners raises the share of periods with flexibility procurement from 5.0% to 18.7% with 24 aggregators, and from 4.9% to 15.4% with 34 aggregators (Figure 6(d)). With the monitor on, all six paired runs in the 34-aggregator configuration show daily procurement falling from about 3.2 to 0.15 MWh and daily return per learner falling from about 118 to 1 CHF.

G.5.4 Analysis of a unilateral markup

In the 24-aggregator, 3.9 MW configuration, aggregator 7's return from a unilateral offer rises with the offer up to the highest offer swept, 9 900 CHF/MWh, as shown in Figure 6(c). With all others at the floor, raising its offer to 150.25 CHF/MWh (0.25% above the floor) lets the cheaper aggregators behind the congested line clear first. On the example day, aggregator 7 is awarded only from 13:00 to 18:00, when their offers cannot meet the requirement as shown in the right subfigure of Figure 38. Its award over the 36 test days falls from 8.86 to 2.53 MWh and remains at 2.53 MWh even at 9 900 CHF/MWh. Its return therefore drops sharply (just above the floor), then rises almost linearly with its offer. For the curves in Figure 6(c), return exceeds its floor value again at about 481 and 489 CHF/MWh in the two 3.9 MW configurations, but only at about 4 590 CHF/MWh in the 34-aggregator, 6.3 MW configuration.

Near the floor, the bidding offer becomes nearly flat: $\pi = c^{\mathrm{rep}}[1 + \mathrm{softplus}(\alpha^{\pi})]$ gives $d\pi/d\alpha^{\pi} = c^{\mathrm{rep}}\mathrm{sigmoid}(\alpha^{\pi})$, which approaches zero as α^{π} decreases as shown in the left subfigure of Figure 38. This weakens the return gradient reaching the price action and helps explain why learners stay near the floor.

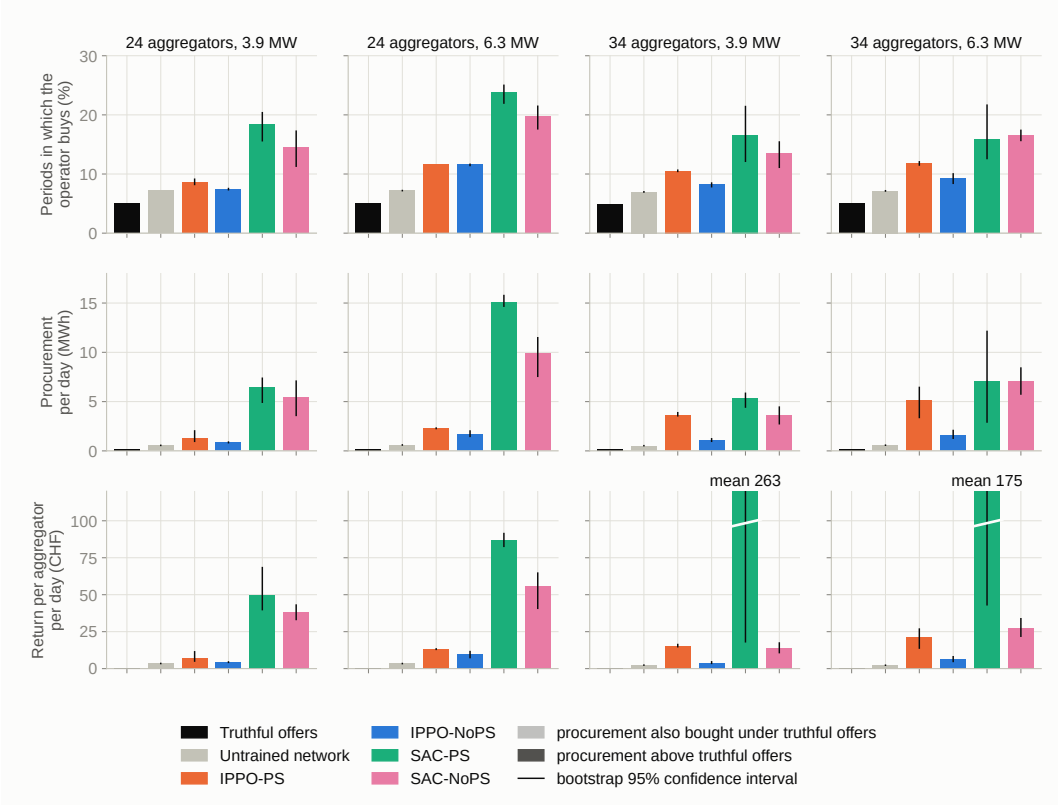


Figure 37: Source of the flexibility requirement over the 36 test days of the four configurations. Bars show truthful offers, the untrained shared and per-agent networks pooled in one bar, and the four learners at validation-selected checkpoints (seed mean and bootstrap 95% confidence interval). Two SAC-PS return bars exceed the axis and are cut at its top, with their means shown above. Rows show the share of periods with procurement; daily procurement split into what truthful offers also buy and the excess; and daily return per aggregator. Learners raise procurement above the truthful-offer level, and their returns rise with it.

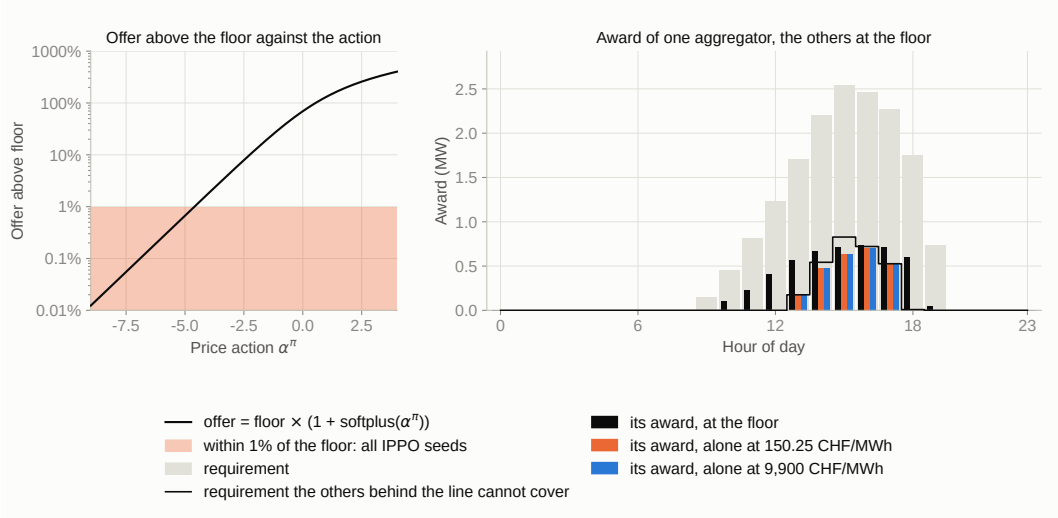


Figure 38: Unilateral offer increase. Left: offer above the floor against price action (log scale). near the floor, the gap shrinks by about a factor of e for each unit decrease in the action. Cleared prices for all IPPO seeds are within 1% of the floor. Right: aggregator 7's hourly award on the example day at offers of the floor, 150.25 and 9 900 CHF/MWh, with all other aggregators at the floor.