When Does a Learned Swarm Beat the Rule?

An Honest Multi-Domain Study of Parameter-Shared and Residual Multi-Agent Reinforcement Learning

Nihal

November 2026

Abstract

Multi-agent reinforcement learning (MARL) is usually compared with other learned methods, which leaves open whether it beats the hand-written rules that operators already use. We run one parameter-shared, value-decomposed swarm engine, without per-domain algorithm changes, on four problems: grid-battery peak shaving on real ERCOT load, drone wildfire suppression, hotel housekeeping and hotel peak-hour energy, plus the CityLearn and VMAS benchmarks. Every method is trained with multiple seeds and compared on held-out test episodes with the strongest simple rule we could write, MAPPO, and, where tractable, a perfect-foresight bound, using paired hierarchical-bootstrap tests. Learned swarms beat the rules only where the rules are structurally limited. On grid batteries the swarm reduces load spread by 10.8% [10.6, 11.0] against 9.1% for the best rule and 6.8% for a forecast-driven MPC, and on CityLearn it scores 0.96 against 1.03 for the reference rule controller. Where a simple rule is already near the bound, nothing learned beats it. A residual swarm, which starts as an exact copy of the rule and learns when to override it, ties the rule in housekeeping and wildfire (-0.3 points, $p = 0.81$) and is never significantly worse. A hand-written coordination rule outperforms every learned drone swarm, and learned drone swarms lose ground as the swarm grows beyond its training size, while the rule does not. We release the engine, configurations, per-seed episodes and the scripts that generate every table.

1 Introduction

Many operational problems are decentralised. Each battery, drone or staff member sees only its surroundings and must act without a central planner. MARL promises controllers that coordinate such agents, and parameter sharing lets one small network drive any number of them [Christianos et al., 2021]. In deployed settings, however, the alternative to MARL is rarely another MARL method. It is a simple rule that an operator wrote, such as "discharge in proportion to load" or "fly to the nearest fire". Benchmarks such as CityLearn [Vázquez-Canteli et al., 2020] and recent multi-KPI studies [Khouja et al., 2026] mostly compare learned methods with each other, which leaves the practical question open.

We ask *when a learned swarm beats the rule*. To answer it we fix the learner and vary the domain. One engine runs every domain through the same environment interface with the same shared-weight agent and value decomposition [Sunehag et al., 2018]. Each domain is evaluated against its best simple rule, a standard MARL baseline (MAPPO, Yu et al., 2022) and, where we can compute one, a perfect-foresight bound. Every result is a mean over independently trained seeds on held-out test episodes, with paired confidence intervals [Henderson et al., 2018, Agarwal et al., 2021]. We report results where learning loses as fully as those where it wins.

Contributions.

1. An evaluation across four application domains and two community benchmarks, with rule, MAPPO and optimum references, held-out test splits and seed-level statistics (Sections 3.4–5).
2. Evidence that a learned swarm wins when the rule's structure cannot express the good policy (keeping battery charge for the exact peak hour), and does not win when the rule is near the bound (Section 5).
3. A residual swarm that starts as an exact copy of the rule [Silver et al., 2018, Johannink et al., 2019]. It is never significantly worse than the rule, but in our domains it never beats it either (Section 5).
4. Robustness and scale studies: graceful degradation under agent failure, a sensitivity of value-based swarms to sensor noise that MAPPO does not share, and a failure of learned drone swarms to scale beyond their training size (Section 6).

2 Related work

Cooperative MARL. Value decomposition sums per-agent values into a team value (VDN, Sunehag et al., 2018) or mixes them monotonically (QMIX, Rashid et al., 2018). MAPPO shows that PPO with a centralised critic is a strong baseline [Yu et al., 2022]. Parameter sharing makes the network size independent of the number of agents [Christianos et al., 2021]. We use VDN with shared weights by default and report QMIX, independent learners and MAPPO as variants.

Residual and rule-guided learning. Residual policy learning adds a learned correction to a fixed controller [Silver et al., 2018, Johannink et al., 2019]. Recent work applies residual learning to safety-critical multi-agent decisions [Wang et al., 2025]. Deep Q-learning from demonstrations pre-trains a Q-network on expert moves with a large-margin loss [Hester et al., 2018]. We use the domain's rule both as the residual base and as the demonstrator.

Energy. CityLearn standardises multi-agent demand response [Vázquez-Canteli et al., 2020]. Khouja et al. [2026] benchmark six MARL variants on it with seeds, bootstrap intervals and robustness to agent removal, but without rule or optimum baselines. MPC-based RL has been applied to multi-agent battery storage [Kordabad et al., 2021]. We add rule, MPC and perfect-foresight references, and we report CityLearn against its own rule-based controllers.

Wildfire. HIVEX includes multi-agent wildfire resource-management and aerial-suppression tasks [Siedler, 2025], and cellular-automaton models are used for autonomous wildfire response [Ramadan, 2024]. Our forest is a stochastic cellular automaton with wind, burn-out and refill pads. We test it from 6 to 1,000 drones.

Hotel operations. We found no MARL work on hotel housekeeping, peak-hour energy or shift exchange. The nearest work is general multi-agent scheduling with RL [Zhou et al., 2022]. We provide mixed-integer optima or lower bounds as references.

3 Method

3.1 Swarm environment interface

Each domain implements one interface: `reset`, `step`, `action_mask` and `episode_metrics`. At each step agent i receives a local observation o_i and a mask of legal actions. The environment returns a team reward r, optionally with small per-agent shaping terms.

3.2 Shared-weight agent with value decomposition

All agents use one network $Q_\theta(o_i, a)$, optionally with a one-hot agent id. The actions of all agents in all parallel environment copies are chosen in one batched forward pass. The team value is $Q_{\text{tot}} = \sum_i Q_\theta(o_i, a_i)$ (VDN). It is trained with n-step double-Q targets [van Hasselt et al., 2016], a soft-updated target network and action masks. Independent Q-learning and a QMIX mixer are configuration switches. MAPPO uses a shared actor and a pooled centralised critic.

3.3 Learning from the rule and the residual swarm

Let π_{rule} be the domain's best rule. *Imitation, then RL* records transitions from π_{rule}, pre-trains with a large-margin loss [Hester et al., 2018], and then fine-tunes with RL while the demonstration loss fades. In the *residual swarm*, each agent observes o_i together with the rule's suggested action $\pi_{\text{rule}}(o_i)$. It chooses either one of its normal actions (an override) or an extra action, *follow the rule*. Pre-training on "follow the rule" makes the untrained swarm behave exactly like π_{rule}, so RL only has to learn when deviating pays. Every deviation is an explicit, logged decision, which makes the policy easy to audit.

3.4 Evaluation protocol

Each domain has disjoint train, validation and test splits. For energy these are January to August, September, and October to December 2020. For generated domains, episode seeds start at 0, 5000 and 10000 respectively. Checkpoints and settings are chosen on validation, and the test split is used only for reported numbers. Each method is trained with independent seeds: 10 for energy, 5 for the main methods elsewhere and 3 for variants. All seeds and all rules play the same test episodes. We report means with 95% hierarchical bootstrap intervals (5,000 resamples), which resample training seeds and, jointly, the shared test episodes [Agarwal et al., 2021]. Comparisons with a rule are paired differences on the same episodes, with a two-sided bootstrap p-value from the same resamples, so $p < 0.05$ exactly when the interval excludes zero.

4 Domains

Grid batteries. Ten batteries (100 MW / 400 MWh each, 90% round-trip efficiency) shave ERCOT COAST native load for 2020, joined with NREL NSRDB weather for Houston. Each agent chooses to charge, hold or discharge every hour. The rules are a threshold rule and a load-proportional rule. The references are an MPC controller on a 24-hour seasonal-naive forecast and perfect-foresight programs that minimise the peak (LP) or the squared deviation from mean load (QP).

Wildfire. Six drones with a payload of 6 and a view radius of 3 cover a 16×16 forest. Two random ignitions spread with probability 0.12 (0.30 downwind) and burn out after 3 to 6 ticks. Drones move three cells per fire tick, drop retardant or hold, and refill on a launch pad. The rule flies to the nearest burning cell and drops. A second action space lets each drone choose *which* fire to fight. In it, a hand-written *spread rule* takes the nearest fire that no other drone is closer to.

Hotel housekeeping. Eight attendants clean the departures of a 300-room, 10-floor hotel from 08:00 to 17:00 in 10-minute steps. Each chooses among six candidate rooms. The objective is guest waiting time, with paid early check-ins weighted three times. The rule cleans the room needed soonest. The reference is a perfect-foresight MILP lower bound that ignores elevator time.

Hotel peak energy, shifts and relocation. Sixteen equipment agents shift load away from the grid's five annual peak hours, where an Ontario-style global-adjustment charge dominates the bill. Open shifts across five properties are filled by a contract-net market under Ontario Employment Standards Act limits. Guests on oversold nights are relocated by an auction. The last two are deterministic market mechanisms, not learned, and are compared with simple rules and an exact optimum.

Benchmarks. To check the engine we run it unchanged on CityLearn Challenge 2022 (17 homes) and on VMAS navigation [Bettini et al., 2022] with 4 robots.

Forests and hotels are synthetic, and their generators are released.

5 Results

Table 1: Energy: 10 batteries (1 GW / 4 GWh fleet) on ERCOT COAST load, test months Oct–Dec 2020. Mean [95% CI] over training seeds (10 for VDN, 5 otherwise); rules and bounds are deterministic. Last column: paired difference in load-spread reduction against the proportional rule.

Policy	Spread cut (%)	Peak cut (MW)	vs proportional (pts)
Swarm, VDN (ours)	10.8 [10.6, 11.0]	704.3 [546.8, 857.3]	1.73 [1.52, 1.97] $p < 0.001$
Swarm, QMIX	10.6 [10.4, 10.7]	829.7 [710.6, 948.7]	1.53 [1.37, 1.67] $p < 0.001$
Swarm, independent learners	10.6 [10.1, 11.0]	248.5 [126.2, 309.7]	1.52 [1.06, 1.92] $p < 0.001$
Swarm, 3-step returns	10.8 [10.6, 11.0]	313.6 [61.9, 631.1]	1.74 [1.55, 1.94] $p < 0.001$
Swarm, no validation selection	11.0 [10.7, 11.4]	482.9 [318.7, 729.5]	1.96 [1.59, 2.32] $p < 0.001$
MAPPO	8.9 [8.4, 9.5]	351.6 [65.0, 674.5]	-0.19 [-0.68, 0.42] $p = 0.478$
Proportional rule	9.1	0.0	
MPC, 24 h forecast	6.8	0.0	
Threshold rule	5.0	0.0	
Bound: flattest load (perfect foresight)	14.7	948.7	
Bound: minimum peak (perfect foresight)	-0.1	948.7	

Grid batteries: the swarm beats every rule. The VDN swarm reduces load spread by 10.8% [10.6, 11.0], against 9.1% for the proportional rule, 6.8% for MPC and 5.0% for the threshold rule ($p < 0.001$ each; Table 1). The rules empty the batteries in the hours before the evening peak. The

Table 2: Wildfire: 6 drones, 16×16 forest, 50 unseen fires. Seeds: 5 for imitation then RL, 3 otherwise. Last column: paired difference in forest saved from the nearest-fire rule in the same run. The target-choice rows use a different action space (drones pick which fire to fight), whose nearest-fire rule saves 88.9%.

Policy	Forest saved (%)	Peak burning cells	vs nearest-fire (pts)
Residual swarm (ours)	89.8 [86.6, 92.0]	9.9 [8.6, 11.3]	-0.3 [-3.3, 2.5] $p = 0.808$
Imitation, then RL	87.2 [82.3, 90.9]	11.5 [9.4, 14.0]	-2.9 [-7.1, 0.8] $p = 0.132$
Imitation, then RL, QMIX	86.8 [82.8, 90.1]	11.5 [9.6, 13.5]	-3.3 [-7.0, -0.1] $p = 0.038$
Imitation, then RL, multi-pad	87.2 [81.4, 91.3]	11.6 [9.3, 14.5]	-2.9 [-8.0, 1.6] $p = 0.190$
Imitation, then RL, neighbour messages	82.1 [77.6, 86.0]	13.3 [11.1, 15.6]	-8.0 [-12.3, -4.2] $p < 0.001$
RL from scratch	79.2 [73.2, 84.8]	14.4 [11.8, 17.4]	-10.9 [-16.6, -5.7] $p < 0.001$
MAPPO	72.1 [62.2, 81.0]	16.9 [13.3, 20.4]	-18.0 [-27.6, -9.2] $p < 0.001$
Target choice: swarm	88.0 [83.6, 91.3]	11.2 [9.2, 13.8]	-0.9 [-4.2, 2.0] $p = 0.582$
Target choice: spread rule	91.2 [88.9, 93.2]	9.5 [8.2, 10.9]	
Nearest-fire rule	90.1 [87.4, 92.5]	10.4 [8.8, 12.0]	
Random	36.8 [30.8, 42.8]	34.7 [31.4, 38.0]	
No drones	33.2 [27.5, 39.4]	37.9 [34.5, 41.4]	

Table 3: Hotel housekeeping: 8 attendants, 300 rooms, 50 generated test days, 5 seeds per method. Guest waiting in hours per day (lower is better); last column is the paired difference from the soonest-needed rule.

Policy	Guest wait (h/day)	vs soonest-needed
Residual swarm (ours)	21.5 [18.7, 24.4]	0.00 $p = 1.000$
MAPPO	21.6 [18.8, 24.5]	0.07 [-0.10, 0.28] $p = 0.434$
Swarm, independent learners	21.9 [18.9, 24.8]	0.31 [0.07, 0.64] $p = 0.006$
Swarm, QMIX	22.3 [19.4, 25.3]	0.78 [0.38, 1.29] $p < 0.001$
Swarm, VDN	22.4 [19.4, 25.4]	0.84 [0.39, 1.40] $p < 0.001$
Soonest-needed rule	21.5 [18.7, 24.3]	
Random room	29.8 [26.3, 33.1]	
Floor order (today's practice)	111.3 [98.6, 123.6]	
Lower bound (perfect foresight)	18.1 [15.6, 20.6]	

swarm learns to keep charge for the peak hour itself. It closes 74% of the gap between doing nothing and the perfect-foresight bound on its own objective (14.7%), and 31% of the gap between the best rule and that bound. Its peak cut varies by seed, 704 MW [547, 857] against a bound of 949 MW set by the fleet's power after losses. QMIX gives a similar result. Independent learners, 3-step returns and MAPPO keep less of the peak. MAPPO is not significantly better than the proportional rule on spread. Choosing checkpoints on validation does not change spread but helps keep the peak cut (704 vs 483 MW, $p = 0.11$).

Wildfire: nothing learned beats the rule. The nearest-fire rule saves 90.1% of the forest (Table 2). The residual swarm ties it (-0.3 points $[-3.3, 2.5]$, $p = 0.81$) with fewer burning cells at the peak. Learning from the rule first is the best non-residual recipe at 87.2%. Learning from scratch reaches 79.2%, MAPPO 72.1%, and neighbour messages lower the imitation swarm to 82.1%. The idea that helps is hand-written. When drones choose which fire to fight, the spread rule saves 91.2%. A swarm trained in that action space reaches 88.0%, level with that space's nearest-fire rule (88.9%, $p = 0.58$) and below the spread rule ($p = 0.004$).

Housekeeping: the rule is near the bound. Every learned swarm cuts guest waiting about five-fold against floor order (Table 3). None beats the soonest-needed rule at 21.5 h/day, and the MILP lower bound leaves at most 3.5 h/day to be won. The residual swarm learns that overriding the rule does not pay and reproduces it exactly. MAPPO ties the rule (+0.07 h, $p = 0.43$). VDN, QMIX and independent learners are significantly worse by 0.3 to 0.8 h.

Table 4: Hotel peak-hour energy (HARLTON-01): 16 equipment agents, summer test weeks, 5 seeds. Net saving per year under the global-adjustment tariff assumptions in the config.

Policy	Net saving ($/yr)	Peak cut (kW)
Swarm (VDN)	211,249 [165,657, 250,491]	486.3 [382.3, 577.7]
No action	0	0.0
Every weekday	250,807	561.3
Peak-alert rule	251,685	561.3

Table 5: Hotel shift exchange across five properties under Ontario ESA limits (contract-net market, deterministic). Cost per week of open shifts.

Policy	Cost ($/wk)	Filled (%)	Agency (h)
Agency for every shift	8,000	0.0	200.0
First to accept	5,249	97.6	4.8
Contract-net market	4,816	97.0	6.1
Optimum (MILP)	4,685	98.9	2.2

Hotel peak energy and markets. With 5 seeds, the learned peak-energy swarm saves $211k per year [166k, 250k]. That is $40k less than the peak-alert rule ($p = 0.03$; Table 4), because the rule already sheds load at all five peaks. An earlier single-seed run had suggested the opposite, which shows why multiple seeds matter. The deterministic market swarms come within 3% of the optimum on shift cost and 5% on relocation value, ahead of the simple rules (Tables 5 and 6).

Table 6: Hotel guest relocation on oversold nights (auction, deterministic). Net value per night.

Policy	Net ($/night)	Kept in group (%)
Walk to competitor	-10,158	0.0
Nearest sister hotel	3,570	85.6
Relocation auction	4,619	86.4
Optimum (MILP)	4,851	87.4

Table 7: CityLearn Challenge 2022 (17 homes, Oct–Dec test), 3 seeds. Scores are normalised so that no battery control is 1.0; lower is better.

Policy	Score	Cost	Daily peak
Swarm (VDN)	0.96 [0.95, 0.97]	0.92 [0.91, 0.93]	0.85 [0.83, 0.86]
No battery	1.00	1.00	1.00
CityLearn BasicRBC	1.13	1.11	1.12
CityLearn OptimizedRBC	1.03	1.03	0.95

Table 8: VMAS navigation, 4 robots, 50 test episodes, 3 seeds.

Policy	Goals reached	Collisions
Swarm (VDN DQN)	0.91 [0.83, 0.98]	0.1 [0.0, 0.2]
MAPPO	0.93 [0.83, 1.00]	2.7 [1.6, 4.0]
Go-to-goal rule	0.98 [0.94, 1.00]	16.0 [8.0, 25.6]
Random	0.00	3.5 [1.9, 5.4]

External benchmarks. On CityLearn Challenge 2022 the swarm's district score is 0.96 [0.95, 0.97], against 1.03 for CityLearn's OptimizedRBC and 1.13 for its BasicRBC (Table 7). It cuts cost by 8% and the average daily peak by 15% relative to no battery control. On VMAS navigation the swarm reaches 91% of goals with 0.07 collisions per episode, while the go-to-goal rule reaches 98% with 16 collisions (Table 8). Both results suggest the engine learns sensible policies on tasks it was not designed for.

6 Robustness and scale

Table 9: Energy robustness: load-spread reduction (%) when batteries fail mid-period or observations are noisy. No policy is retrained.

Policy	Normal	10% fail	30% fail	50% fail	Noise 0.3
Swarm (VDN)	10.8	10.6	10.0	9.0	10.4
Proportional rule	9.1	8.6	7.6	6.5	9.1
MPC, 24 h forecast	6.8	6.5	6.0	5.2	6.8
Threshold rule	5.0	5.2	5.2	4.8	5.0

No policy is retrained for these tests. A failed agent takes the domain's null action from the failure time on, and the other agents are not told.

Table 10: Housekeeping robustness: guest waiting (h/day) when attendants leave at 11:00 and on unseen busier days.

Policy	Normal	2 leave	4 leave	Busier day	More early
Swarm (VDN)	22.4	28.1	48.1	38.9	45.5
Soonest-needed rule	21.5	25.6	43.1	36.0	43.0
Floor order	111.3	157.6	205.6	184.4	202.4

Table 11: Wildfire robustness: forest saved (%) when drones fail, under Gaussian observation noise, and on unseen conditions. Noise is added to the agents' observations; the nearest-fire rule reads the true state.

Policy	Normal	50% fail	Noise 0.1	Noise 0.3	Strong wind	4 fires
Residual swarm	89.8	88.0	78.8	63.8	79.6	70.4
Imitation, then RL	87.2	84.5	70.0	44.2	75.4	63.6
RL from scratch	79.2	74.0	66.5	51.5	66.0	48.7
MAPPO	72.1	68.6	72.4	72.3	59.5	48.1
Nearest-fire rule	90.1	87.0	90.1	90.1	77.8	70.5
Target choice: spread rule	91.2	90.1	88.8	88.8	80.8	76.7

Agent failure is handled gracefully. With 30% of batteries failing mid-period, the energy swarm still reduces spread by 10.0%, more than any rule with all of its batteries working (Table 9). In wildfire, every policy keeps at least 93% of its result with half the drones failed (Table 11). In housekeeping, the learned swarm and the rules degrade in proportion as attendants leave or the day gets busier (Table 10).

Value-based swarms are sensitive to sensor noise. With Gaussian observation noise of $\sigma = 0.3$, the imitation swarm falls from 87% to 44% forest saved and the residual swarm from 90% to 64%. MAPPO is unaffected (72% in every noise setting). The rules read the true state in our simulator, so this comparison favours them. A deployed rule would face the same sensors.

Table 12: Compute per swarm size on one CPU core (wildfire). The network has 109,574 parameters at every size.

Drones	Act (ms)	Env step (ms)	Update (ms)
6	0.20	1.3	4.1
24	0.22	1.7	13.5
100	0.42	3.7	60.0
1,000	2.74	63.4	594.8

Learned drone swarms do not scale. One network runs unchanged at any swarm size with the same parameters, and choosing actions for 1,000 drones takes 2.7 ms on one CPU core (Table 12). With a single corner launch pad, refill trips dominated at large sizes, so we tile the forest with pads. The nearest-fire rule then saves 86 to 92% of the forest at every size (Figure 1). The swarm trained with 6 drones matches the rule at its training size (90.5%) but falls to 82.6%, 65.9% and 47.9% at 24, 100 and 1,000 drones. A curriculum from 6 to 24 drones was worse at every size. The battery swarm keeps its peak cut at 10, 100 and 1,000 agents, but the fleet's total power stays 1 GW. This

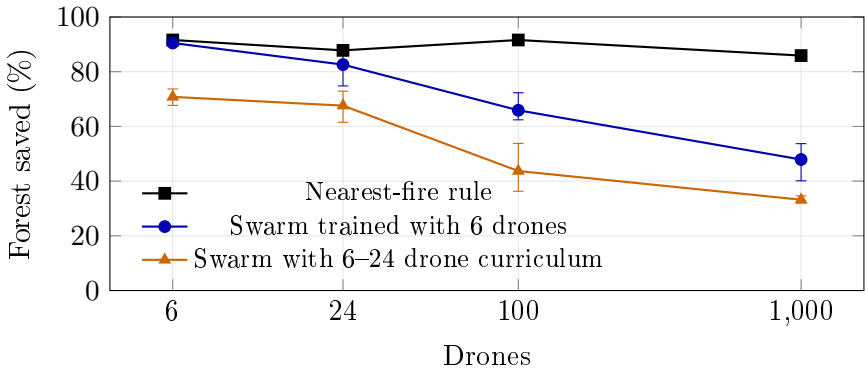


Figure 1: Wildfire scale with launch pads tiling the forest. Each model runs unchanged at every size; error bars are 95% intervals over 3 training seeds.

shows stable coordination, not added benefit.

7 Discussion

Across domains, one pattern explains our results. A learned swarm helps when the best simple rule cannot express the good policy. In energy, rules that react to the current load cannot hold charge for a peak that arrives hours later, but a learned value function can. When the rule's structure already matches the problem, as in nearest-fire suppression or soonest-needed cleaning, learning only adds variance. The residual swarm makes this trade-off explicit. It costs nothing where the rule is good, because it can always follow the rule. It has not yet found improvements where we measured none to find: in housekeeping, the bound leaves at most 3.5 h/day.

For practitioners, this suggests a simple order of work. First write the best rule and compute a bound. Then use learning where the gap between them is large and the rule's structure is the bottleneck. Our wildfire results also show that thinking about coordination by hand (the spread rule) can beat learning it.

8 Limitations

Energy results use one year of one ERCOT zone. Multi-year data and a learned load forecast are left for future work. Forests and hotels are synthetic, so absolute numbers (dollars, hours) depend on the generators' assumptions. The perfect-foresight references know the future and act continuously, so they are bounds, not deployable controllers. Several variants use 3 seeds, which leaves wide intervals. In our simulator the rules read the true state, which flatters them under sensor noise. No policy has been tested on hardware or with people.

9 Conclusion

A learned swarm beat hand-written rules where the rules' structure was the bottleneck: grid batteries and CityLearn. It did not beat them where a simple rule was near the bound: wildfire and housekeeping. Starting from the rule and learning only when to override it removes the downside, but it did not find an upside in those domains. Making learned swarms scale beyond their training size, and robust to noisy sensors, are the open problems this study points to.

Code and data availability

Code, configurations, per-seed results and the scripts that generate every table and figure in this paper are available at `https://github.com/nihalunfc/swarm-intelligence` under the MIT license. [TODO: the repository must be public, or a public copy made, before submission.]

Acknowledgements

The author used AI tools (Claude, Claude Code, Antigravity and Gemini) for software development and drafting. The author designed the study, checked all results and takes full responsibility for the content.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2108.13264.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A vectorized multi-agent simulator for collective robot learning. *arXiv preprint arXiv:2207.03530*, 2022.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. arXiv:2102.07475.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. arXiv:1709.06560.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. arXiv:1704.03732.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. arXiv:1812.03201.

Aymen Khouja, Imen Jendoubi, Oumayma Mahjoub, Oussama Mahfoudhi, Claude Formanek, Siddarth Singh, and Ruan de Kock. Characterizing MARL for energy control: A multi-KPI benchmark on the CityLearn environment. *arXiv preprint arXiv:2602.19223*, 2026.

Arash Bahari Kordabad, Wenqi Cai, and Sebastien Gros. Multi-agent battery storage management using MPC-based reinforcement learning. *arXiv preprint arXiv:2106.03541*, 2021.

Abdelrahman Ramadan. Wildfire autonomous response and prediction using cellular automata (WARP-CA). *arXiv preprint arXiv:2407.02613*, 2024.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1803.11485.

Philipp D. Siedler. HIVEX: A high-impact environment suite for multi-agent research. *arXiv preprint arXiv:2501.04180*, 2025.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018. arXiv:1706.05296.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

José R. Vázquez-Canteli, Sourav Dey, Gregor Henze, and Zoltán Nagy. CityLearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020.

Kaifeng Wang, Yinsong Chen, Qi Liu, Xueyuan Li, and Xin Gao. Dynamic residual safe reinforcement learning for multi-agent safety-critical scenarios decision-making. *arXiv preprint arXiv:2504.06670*, 2025.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022. arXiv:2103.01955.

Hongjian Zhou, Boyang Gu, and Chenghao Jin. Reinforcement learning approach for multi-agent flexible scheduling problems. *arXiv preprint arXiv:2210.03674*, 2022.