Multi-Agent Reinforcement Learning for Task Offloading in Wireless Edge Networks

Andrea Fox
LIA, Avignon University
Avignon, France
andrea.fox@univ-avignon.fr

Francesco De Pellegrini
LIA, Avignon University
Avignon, France
francesco.de-pellegrini@univ-avignon.fr

Eitan Altman
INRIA, Sophia Antipolis, France
eitan.altman@inria.fr

Abstract

In edge computing systems, autonomous agents must make fast local decisions while competing for shared resources [1,2]. Existing MARL methods often resume to centralized critics or frequent communication, which fail under limited observability and communication constraints. We propose a decentralized framework in which each agent solves a constrained Markov decision process (CMDP), coordinating implicitly through a shared constraint vector. For the specific case of offloading, e.g., constraints prevent overloading shared server resources. Coordination constraints are updated infrequently and act as a lightweight coordination mechanism. They enable agents to align with global resource usage objectives but require little direct communication. Using safe reinforcement learning, agents learn policies that meet both local and global goals. We establish theoretical guarantees under mild assumptions and validate our approach experimentally, showing improved performance over centralized and independent baselines, especially in large-scale settings.

1 Introduction

In decentralized systems, local decisions taken by agents collectively influence the outcome of other agents' actions or the availability of shared resources, with direct impact on individual performance. In mobile edge computing (MEC), for instance, this is the case of edge offloading techniques. They permit multiple devices to independently decide whether to offload computations to a shared edge server or to process them locally. But, while each agent operates based on local observations and limitations, such as battery level, workload, or latency requirements, the collective offloading decisions directly affect server responsiveness and network congestion. Actually, when too many agents offload simultaneously, the resulting overload can degrade performance across the system. In such settings, each agent's objective depends not only on its own decisions but also on the aggregated behavior of others, introducing a coordination challenge. This challenge is further amplified in environments with communication delays or asynchronous agent behavior, where real-time coordination is difficult or infeasible. Designing scalable learning methods that support implicit

[1] A full-length version of this work is available at https://arxiv.org/abs/2509.01257, and the code used in our experiments can be found at https://github.com/Andrea-Fox/multiAgentTaskOffloading.

[2] This work has been partially supported by the French National Research Agency (ANR) within the PARFAIT project (ANR-21-CE25-0013).

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI and ML for Next-Generation Wireless Communications and Networking (AI4NextG).

coordination under these system technical constraints is essential for efficient and robust distributed operation.

Many existing approaches to coordination in multi-agent reinforcement learning assume additive rewards or rely on shared incentives and communication protocols. However, in practical systems like edge computing and wireless access networks, agent interactions are tightly coupled through shared resources, and rewards are often non-additive due to congestion effects. To address these coordination challenges, we propose a decentralized learning framework based on independent constrained Markov decision processes (CMDPs). In this framework, coordination is achieved implicitly through shared constraints that are updated infrequently. These constraints regulate the maximum frequency at which each agent can offload tasks, effectively serving as a virtual coordination mechanism over shared resources. Each agent optimizes its own policy using techniques from constrained reinforcement learning, ensuring local autonomy while maintaining system-wide alignment through periodic, constraint-driven synchronization. This approach enables fast, scalable, and communication-efficient decision-making in distributed environments like edge computing, where centralized coordination is often impractical.

Our contributions are resumed hereafter:

- We introduce DCC, a general decentralized reinforcement learning framework for multi-agent coordination under shared resource constraints; we apply it to the problem of task offloading in wireless edge computing systems. Each device (agent) solves a local CMDP, and coordination emerges implicitly through the shared constraint vector that regulates offloading behavior across the network.
- We provide a tractable approximation of the global objective via decomposition, establish theoretical guarantees on its validity under mild assumptions and error bounds in the nonlinear case.
- We validate the DCC framework through preliminary numerical experiments in toy environments. While limited in scope, our results highlight the scalability of the approach and the net improvement over centralized and independent baselines. They lay the groundwork for more extensive evaluations in future research.

The paper is organized as follows: section 2 reviews related work, while section 3 introduces the Markov game and the system model under consideration. In section 4, we present the proposed algorithm for the agents' constrained policy optimization. Numerical results are provided in section 5, followed by concluding remarks in the final section.

2 Related works

Multi-agent reinforcement learning (MARL) has been extensively surveyed in [11, 33, 2]. Centralized Training Distributed Execution (CTDE) approaches are the most widely used in MARL. For example, MADDPG [18], MAPPO [32], and COMA [7] employ centralized critics to guide training, while policies are executed in a decentralized manner. Value-decomposition methods such as VDN [26] and QMIX [23] improve scalability further by factorizing value functions, though they operate under an individual-global-max assumption, which does not hold in settings where agents compete for scarce resources. Independent learners such as IQL [27] and IPPO [6] avoid centralization entirely, but struggle to coordinate in environments with interdependent rewards.

Constrained reinforcement learning (CRL) extends standard RL by requiring policies to satisfy long-term constraints in addition to maximizing rewards. A variety of approaches have been proposed to solve constrained Markov decision processes (CMDPs) [9, 17]. Primal-dual methods, such as RCPO [29], optimize reward and constraints concurrently, while value-based methods [5] adapt Q-learning to the constrained setting. CPO [1] formulates the problem as a constrained trust-region update, offering guarantees but incurring high computational cost. IPO [16] introduces a first-order method using barrier functions, and PCPO [31] projects policies onto the feasible set after unconstrained optimization.

While reinforcement learning has been widely applied to MEC offloading problems—typically optimizing task latency or energy consumption using single-agent algorithms such as DQN [15] and DDQN [28]—multi-agent settings remain comparatively underexplored. Some recent works integrate

MARL into this domain: for example, [8] combines QMIX with DQN to jointly minimize average delay and energy cost, while [12] employs decentralized actor–critic agents for user-level offloading decisions. However, these approaches either rely on centralized training, assume frequent inter-agent communication, or do not explicitly address coordination under shared resource constraints. In contrast, our method formulates offloading as a set of independent constrained MDPs, where coordination emerges implicitly via infrequent updates to shared constraints, enabling scalable, communication-efficient learning tailored to congestible wireless environments.

3 System model

In wireless edge computing, a collection of mobile devices must decide at each time step whether to execute computational tasks locally, offload them to a shared edge server, or defer processing altogether. Each device operates independently based on its local state, which may include factors such as task backlog, latency sensitivity, energy harvesting rate, local processing cost, the time elapsed since the last data processing, and battery level. The wireless channel and the edge server represent shared, capacity-limited resources. When too many devices offload simultaneously, contention and server overload degrade performance for all users. To mitigate this, we impose a constraint on the long-term frequency with which each agent may offload, serving as a virtual coordination mechanism. This constraint *does not* reflect a physical device limitation, but rather a system-level policy that governs fair and efficient resource usage.

Each device is modeled as an agent solving a local constrained MDP (CMDP) [3], optimizing its policy to balance local performance objectives with the global constraint on offloading frequency. Coordination emerges implicitly as agents adapt their policies based on local observations and periodic updates to the shared constraint, enabling decentralized yet system-aware behavior.

3.1 Markovian formulation

We model the system described in section 3 as a decentralized multi-agent problem in which N agents must coordinate their actions to minimize a global objective function. The joint state of the system is denoted by $s = (s_1, \ldots, s_N) \in \mathcal{S}$, where s_i is the local observation of agent i. Each agent selects an action a_i from its individual action space, and the joint action is $a = (a_1, \ldots, a_N) \in \mathcal{A}$.

We assume independent transition dynamics, i.e., the transition probability of the system factorizes across agents as $p(s' \mid s, a) = \prod_{i=1}^{N} p_i(s'_i \mid s_i, a_i)$, where $s' = (s'_1, \ldots, s'_N)$ is the next state, and p_i denotes the local transition kernel for agent i.

Each agent's reward consists of a local component and a global component that depends on the actions of the other agents. This formulation is sufficiently general to model a broad range of mobile edge computing applications.

$$r(s, a) = \sum_{i=1}^{N} r_i(s_i, a) = \sum_{i=1}^{N} u_i(s_i) + \mathbb{I}[a_i = a_{\text{crowd}}] \cdot d\left(N(a)\right) \tag{1}$$

where $u_i : \mathcal{S}_i \to \mathbb{R}$ describes the local utility function for agent i, $a_{\text{crowd}} \in \mathcal{A}_i$ is a designated *crowded action*, corresponding to offloading the task to the edge computer, $N(a) = \sum_{j=1}^{N} \mathbb{I}[a_j = a_{\text{crowd}}]$ is the number of agents choosing the crowded action and $d : [1, \infty) \to \mathbb{R}$ is a differentiable function with $d(1) = 0, d'(n) > 0$, and $d''(n)$ exists and does not change sign.
The function $d(n)$ models congestion at the edge: the more agents select the crowd action, the greater the penalty each receives for choosing it. This structure induces coordination challenges reminiscent of *crowding games* or *singleton congestion games*, where agents compete over a shared, congestible resource. This additive form preserves the separability of local rewards while explicitly modeling the coupling from congestion through $d(N(a))$, which captures the performance degradation when many agents offload simultaneously.

We assume the following misalignment between individual and collective incentives:

Assumption 1 (Individual Incentive for Crowd Action)**.** *Let $a \in \mathcal{A}$ be such that $N(a) = 1$ and $a_i = a_{crowd}$, i.e., only agent i selects the crowd action. Let $a' \in \mathcal{A}$ be identical to a except $a'_i \neq a_{crowd}$, with $a'_j = a_j$ for all $j \neq i$. Then, for all states s:*

$$r(s, a) < r(s, a'). \tag{2}$$

That is, the crowd action appears individually beneficial when selected in isolation. However, this creates a coordination dilemma: if every agent independently learns to select the crowd action, the shared penalty $d(N(a))$ increases, degrading the overall system performance. This structure highlights the need for coordination-aware learning methods beyond naive reward minimization.

The objective function we want to minimize is

$$J(\pi, \beta) = \mathbb{E}_{a \sim \pi(s)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 \sim \beta \right] \tag{3}$$

While the dynamics are independent across agents, the reward function introduces a critical coupling through the crowd-dependent term $d(N(a))$, which depends on the number of agents selecting a specific action a_{crowd}.
This structure creates several challenges in analyzing or optimizing $J(\beta)$:

- **Non-decomposability:** The reward $r(s, a)$ is not additive across agents, as the congestion penalty depends on the global joint action a. As a result, standard decomposition techniques used in multi-agent systems with independent rewards (e.g., independent learners) are not applicable.
- **Coupled incentives:** Even if agents act independently, their optimal behavior is interdependent due to the shared penalty. The optimal joint policy may require implicit coordination to avoid overcrowding the sensitive action, a behavior that cannot be captured by decentralized greedy learners.
- **Noisy reward:** Since the congestion term depends on how many agents select a_{crowd}, individual agents receive noisy or misleading feedback about their own contribution to the reward. This complicates both policy gradient and value-based methods.
- **Non-stationarity in decentralized learning:** In decentralized training, each agent's policy evolves independently. Coupling in the reward makes the environment non-stationary from each agent's perspective, even though the transition dynamics are stationary.

These properties position the problem beyond standard cooperative multi-agent reinforcement learning (MARL) settings with additive rewards or shared goals. In the next section, we present a learning framework designed to address these challenges.

4 Using MARL for optimizing offloading

We now introduce the **DCC framework** (Decentralized Coordination via CMDPs), a general structure for multi-agent reinforcement learning in shared-resource environments. DCC enables scalable coordination among agents by combining three key ingredients:

1. **Lightweight Communication:** Agents operate independently, without exchanging real-time state or action information. Inter-agent communications do occur, but only infrequently and to share scalar constraint variables, which enables coordination.
2. **Constraint-Based Coupling:** Each agent solves a constrained MDP in which the constraint controls the crowded action. The update of the constraint variable serves as a coordination signal across agents.
3. **System-Level Alignment:** Optimizing the constraint vector steers individual agent behavior toward global objectives, without requiring centralized control or synchronous communication.

We remark that DCC is not a single algorithm but a general *framework* compatible with a broad class of reinforcement learning methods for local MDPs, including value-based and policy-gradient approaches. In this section we outline its formal structure and learning process across three timescales, with full details reported in Appendix A.

To approximate the global objective (3), we reformulate it as a sum of local value functions constrained by agent-specific action frequencies:

$$J(\pi^\star_{global}, \beta) \approx \inf_\theta \sum_{i=1}^{N} J_i(\pi_i^\star(\theta_i), \beta_i) \tag{4}$$

where $\pi^\star_{global}$ is the optimal (unknown) global polocy and $\pi_i^\star(\theta_i)$ is the optimal policy of agent i under the constraint value θ_i, obtained by solving its individual CMDP. This decomposition decouples learning while preserving system-level coordination through optimization of the shared constraint vector θ.

To facilitate the development of a three-timescale learning algorithm, we define an approximate reward function: it replaces the actual number of agents which choose the congested action with the expected value. It corresponds to number of agents who select the crowded action according to their local constraint. This approximation removes the direct dependency on joint actions: each agent optimizes its policy independently and yet full constraint vector θ accounts for the aggregate system behavior.

Our algorithm uses a three-timescale learning process to solve the approximation in eq. (4):

- **Fast and Intermediate Timescales:** For a fixed constraint vector θ, each agent independently optimizes its own policy by solving a local CMDP using a Lagrangian-based safe reinforcement learning approach. On the fastest timescale, the policy is updated using a shaped reward of the form $r_i^{\text{tot}}(s, a) = r_i(s_i, a) + \lambda_i c_i(s_i, a_i)$, where c_i denotes the cost associated with selecting the crowded action. Since both θ and λ_i are held fixed during this phase, *any standard reinforcement learning algorithm*, such as PPO, DQN, or even Q-learning if the system is small enough, can be used to optimize the local policy. On the intermediate timescale, the Lagrange multiplier λ_i is updated via a primal-dual method to enforce long-term satisfaction of the constraint.
- **Slow Timescale:** At the slowest time-scale, the constraint vector θ is optimized to improve global coordination. Instead of heuristically selecting θ, we treat it as a coordination variable and optimize it via stochastic approximation. Importantly, the structure of the objective allows each component of the gradient $\nabla_\theta J$ to be estimated using just three local policy evaluations per agent, regardless of the total number of agents. These evaluations can be performed locally by each device, ensuring that the algorithm remains scalable and communication-efficient even in large systems.

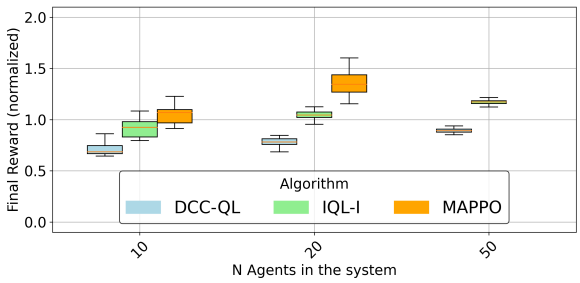
While constraint updates are performed synchronously across agents, they occur at a much slower timescale and can be implemented with minimal coordination overhead. In practice, such updates are orders of magnitude less frequent than policy learning steps, preserving the algorithm's scalability and decentralization. Notably, the computational complexity of DCC is similar to the one of the RL algorithm chosen, with the primary difference being the computation of the gradient—an operation that is computationally inexpensive.

Together, these components yield a principled and practical algorithm for decentralized coordination under shared resource constraints, with theoretical support for its approximation quality and optimization dynamics. The technical description of the algorithm is deferred to appendix A.

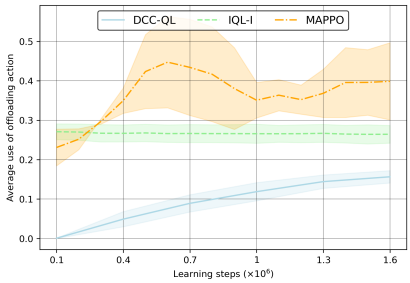
5 Numerical experiments

We consider the environment introduced in section 3, with the toy model detailed in appendix B. Each experiment is averaged over 15 runs on independently generated environments. We focus on small instances where the optimal policy can be determined via Q-learning, enabling clear validation of our approach.

We compare DCC-QL against two representative baselines. First, independent Q-learning (IQL), which represents the fully decentralized setting without communication. Second, MAPPO [32], a CTDE method that has shown strong performance in cooperative MARL benchmarks. We use IQL rather than the more common IPPO because the toy environments are very small, and IPPO's larger architectures tend to overfit, obscuring the relative benefits of our coordination mechanism.



(a) Normalized final average reward across methods and system sizes. Rewards are scaled so that 1 equals DCC-QL 's performance after the first 10^5 steps. Error bars indicate variability across seeds. MAPPO results for 50 devices (average ≈ 3.5) are omitted to avoid distorting the scale.



(b) Evolution of the offloading frequency in a 10-device system. Starting from $\theta = 0$, DCC-QL gradually converges to stable usage, while IQL quickly locks into a suboptimal policy with excessive offloading.

Finally, we stress that this is an early-stage exploration of the DCC framework. Our experiments on toy environments are intended as proof-of-concept to validate theoretical properties and highlight scalability trends, rather than as comprehensive benchmarks. Extending this evaluation to realistic wireless network simulators constitutes an important direction for future work.

Scalability properties The first experiment evaluates the scalability of DCC-QL by comparing its discounted reward after five iterations of constraint improvement—starting from $\theta = 0$, meaning agents are initially prohibited from using the common resource—with that of the baselines as the number of devices increases. All methods are given the same total number of policy-learning steps, and rewards are normalized so that 1 corresponds to the reward achieved by DCC-QL after the first 10^5 learning steps. As shown in Fig. a, DCC-QL consistently outperforms independent Q-learning across all system sizes, while MAPPO, though competitive in small systems, degrades rapidly as the number of devices grows, likely due to its fixed network architecture being unable to handle the enlarged state–action space. We also note that in larger systems, never using the offloading action—i.e., the normalization baseline obtained with $\theta = 0$ —yields higher rewards than IQL, underscoring how IQL's lack of coordination prevents it from capturing the system dynamics.

Offloading action frequency As a second observation, we analyze the frequency of the offloading action in the case with 10 devices (results for $N = 20$ and $N = 50$ are provided in the additional material). Figure b shows that DCC-QL gradually evolves from never using the offloading action (the initial evaluation and normalization baseline) to a moderate and stable usage level, consistent with the constraints derived in section 4. In contrast, IQL quickly converges to a suboptimal policy that overuses the offloading action, as expected given its attractiveness (Assumption 1) and the lack of any coordination mechanism. MAPPO, meanwhile, has not converged within the limited training budget considered here and continues to overuse the offloading action; however, when trained for 10 million steps (six times more than evaluated in this experiment), it eventually achieves performance comparable to DCC-QL , indicating that it can learn the dynamics of the small system but only at significantly higher sample complexity.

Additional experimental results are reported in the appendix. These include the effect of constraint initialization, comparisons with alternative baselines, the impact of non-linear penalties, and evaluations of some of the theoretical results using linear programming. Together, these supplementary experiments further support the conclusions drawn from the main results.

6 Conclusions

This paper introduced the DCC framework, a constraint-based approach to decentralized coordination in multi-agent reinforcement learning, and illustrated its potential in the context of task offloading at the wireless edge. Our focus here has been on laying the theoretical foundations and providing preliminary, proof-of-concept experiments to validate the core ideas. While the empirical evaluation

is intentionally limited to toy models, the results suggest that constraint-driven implicit coordination can scale better than centralized methods and consistently outperform independent learners.

Future work includes extending the framework to support asynchronous updates, exploring richer forms of shared constraints, and conducting broader experiments to further validate the effectiveness of our approach. In this early work we abstract away the system-level implementation of constraint updates, but envision that in practice they could be coordinated through lightweight periodic broadcasts from edge servers or distributed consensus protocols among devices. Exploring these mechanisms remains a valuable direction for future work.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proc. of ICML*, 2017.

[2] Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

[3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 1999.

[4] B. Barakat, H. Yassine, S. Keates, I. Wassell, and K. Arshad. How to measure the average and peak aoi in real networks? In *Proc. of EWC*, 2019.

[5] Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

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

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

[8] Zhaoyu Gan, Rongheng Lin, and Hua Zou. A multi-agent deep reinforcement learning approach for computation offloading in 5g mobile edge computing. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 645–648. IEEE, 2022.

[9] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 46(12), 2024.

[10] Zihao Guo, Richard Willis, Shuqing Shi, Tristan Tomilin, Joel Z Leibo, and Yali Du. Socialjax: An evaluation suite for multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:2503.14576*, 2025.

[11] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[12] Xiaoyan Huang, Supeng Leng, Sabita Maharjan, and Yan Zhang. Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks. *IEEE Transactions on Vehicular Technology*, 70(9):9282–9293, 2021.

[13] H.J. Kushner and D.S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Applied Mathematical Sciences. Springer, 1978.

[14] Pierre L'Ecuyer and Peter W Glynn. Stochastic optimization by simulation: Convergence proofs for the gi/g/1 queue in steady-state. *Management Science*, 40(11):1562–1578, 1994.

[15] Ji Li, Hui Gao, Tiejun Lv, and Yueming Lu. Deep reinforcement learning based computation offloading and resource allocation for mec. In *2018 IEEE wireless communications and networking conference (WCNC)*, pages 1–6. IEEE, 2018.

[16] Yongshuai Liu, Jiaxin Ding, and Xin Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4940–4947, 2020.

[17] Yongshuai Liu, Avishai Halev, and Xin Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In *Proc. of IJCAI*, 2021.

[18] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Proc. of NIPS*, 2017.

[19] James MacGlashan, Evan Archer, Alisa Devlic, Takuma Seno, Craig Sherstan, Peter Wurman, and Peter Stone. Value function decomposition for iterative design of reinforcement learning agents. *Advances in Neural Information Processing Systems*, 35:12001–12013, 2022.

[20] Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[22] 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.

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

[24] Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 656–663, 2003.

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

[26] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. In *Proc. of AAMAS*, 2017.

[27] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. of ICML*, 1993.

[28] Ming Tang and Vincent WS Wong. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, 21(6):1985–1997, 2020.

[29] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *Proc. of ICLR*, 2019.

[30] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[31] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. In *Proc. of ICLR*, 2020.

[32] 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. *Proc. of NeurIPS*, 2022.

[33] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

A DCC framework

This section provides a more precise and formal definition of the DCC framework, elaborating on the key components and methodologies involved in its design and implementation. It is organized into the following steps:

- Introduction of a decomposable approximation of the immediate reward (Appendix A.1): we propose a decomposable approximation of the immediate reward that approximates the long-term reward with provably bounded approximation error—and exact equivalence when the congestion function is linear.
- CMDP of the system (Appendix A.2): we introduce the single agent CMDP that will by used by the three timescale algorithm
- Computing the optimal constrained policy (Appendix A.3): We describe the process of computing the optimal constrained policy for a single agent, given a fixed value of the constraint.
- Differentiability of the objective (Appendix A.4): We establish that the long term reward of each agent is differentiable with respect to θ, which allows us to efficiently identify the minimum in (4)
- Efficient computation of the gradient (Appendix A.5): We show how the structure of the gradient can be leveraged to accelerate the computation of the infimum of the objective function.
- some technical proofs conclude the section (appendices A.6 and A.8).

A.1 Approximating the objective function via decomposition

In a general setting general, the value function decomposition is a viable alternative when

$$r(s,a) = \sum_i r_i(s_i, a_i)$$

meaning that each agent can act independently while still achieving the globally optimal outcome [19, 24]. However, in the setting studied in this work, this condition does not hold due to a coupling term that depends on how many agents select the *crowded* action. Specifically, the global reward takes the form

$$r(s, a = \pi_{\text{global}}(s)) = \sum_i f_i(s_i) + \mathbb{I}_{a=a_{\text{crowd}}}(a_i) \cdot d\left(N(a)\right) \tag{5}$$

where $N(a)$ denotes the number of agents that choose the crowded action under π_{global}. (as defined in section 3).

To address the challenge of decomposing this reward function, we propose a new (approximated) formulation of the reward :

$$\begin{aligned} \hat{r}(s,a;\theta) &= \sum_i u_i(s_i) + \mathbb{I}_{a=a_{\text{crowd}}}(a_i) \cdot d\left(1 + \sum_{j\neq i} \theta_j\right) \\ &= \sum_i \hat{r}_i(s_i, a_i; \theta_{-i}), \end{aligned} \tag{6}$$

where θ_j denotes the average constraint value for agent j, i.e. the expected frequency with which agent j selects the crowded action, and $\theta_{-i} = \sum_{j\neq i} \theta_j$.
Intuitively, this approximation replaces the randomness in the number of offloading agents ($d(N)$) with its expectation (θ), which reduces variance in the reward signal while still capturing the average congestion effect. This makes the problem more tractable for decentralized learning, while preserving the essential trade-offs.

This amounts to treating the policies of other agents (which are unknown to any given agent) as if they select the congested action as frequently as their individual constraints permit. Although this may not strictly hold in practice, it is justified by Assumption 1, which states that the congested action

yields the highest reward when chosen in isolation—implying that agents are likely to fully utilize their constraints.

This approximation brings two key benefits for decentralized learning. First, by replacing the actual number of agents choosing the crowded action with its expected value under the constraint vector θ, it removes the non-stationarity typically introduced by inter-agent coupling. As a result, each agent's reward depends only on its local state, action, and the fixed parameter θ, effectively decoupling the learning dynamics across agents during the fast timescale. Second, the use of an expected congestion term reduces the variance of the reward signal each agent observes, since it no longer depends on the stochastic actions of others. This lower-variance signal leads to more stable updates and can accelerate policy learning. Finally, in practical implementations such as edge computing, this formulation allows an agent to compute its reward without waiting for real-time feedback on how many others offloaded in the same timestep. While this does not reduce communication during deployment, it simplifies both training and simulation, and avoids introducing extra delays into the reward computation.

Error Bound for the Reward Approximation The following result gives an indication of the error that one could have when considering the approximated definition of the reward in (6). In particular, we define $J(\pi, \beta)$ the discounted reward associated to a policy π when considering the reward r and $\hat{J}(\pi, \beta)$ the one obtained when considering the approximated reward $\hat{r}$ defined in (6).

Lemma 1. *Given a global policy π and a vector $\theta \in \mathbb{R}^N$ such that*

$$\mathbb{E}_{a\sim\pi}\left[N_{i,t}(a)\right] = \theta_i,$$

where $N_i(a)$ denotes the random variable representing the frequency with which agent i selects the crowded action at time t, it is verified that for a non linear penalty function d

$$\mid J(\pi,\beta) - \hat{J}(\pi,\beta) \mid \leq \frac{1}{1-\gamma}\sum_i \theta_i \left(\frac{\theta_{-i}}{N_{agents}-1} d(N_{agents}) - d(1+\theta_{-i})\right)$$

Moreover, if d is linear, the two reward values coincide exactly, and the approximation becomes exact.

Proof. See appendix A.6. □

This leads to the following result, which shows that, under certain conditions on the immediate reward function, the approximation in (4) holds exactly. As a result, the decomposition is valid, and the algorithm proposed in this work can reliably recover the optimal policy. Importantly, this also establishes clear conditions under which the method is guaranteed to be optimal.

Proposition 1. *Let assume we know $\theta^* \in \mathbb{R}^N$ such that $\mathbb{E}_{a\sim\pi^*}\left[N_{i,t}(a_t)\right] = \theta_i^*\ \forall i$, for the optimal global policy π^* and assume that $d(\cdot)$ is linear. Then the value function does not depend on the type of reward that we consider and an optimal global policy can be obtained as the combination of the appropriate optimal local policies.*

Proof. From lemma 1, we know that if the function d is linear, then the discounted reward of a policy is invariant under the choice between the original reward r and the approximated reward $\hat{r}$. Assuming the optimal constraint values θ^* are known, we can compute the locally optimal policy for each agent using $\hat{r}$. A policy composed by combining these local policies, denoted as π^*_{composed}, is then a solution to the global minimization problem with respect to $\hat{r}$, and by the invariance, also with respect to r. This leads to the conclusion:

$$J(\pi_{\text{composed}}, \beta) = \hat{J}(\pi_{\text{composed}}, \beta) = \sum_i \hat{J}_i(\pi_i^\star(\theta^*), \beta)$$

□

A.2 Constrained MDP for Each Agent

As introduced in eq. (4), we approximate the global objective by decomposing it into a collection of independent Constrained Markov Decision Processes (CMDPs), one per agent. Constrained Markov Decision Process (CMDP) [3] extends the standard MDP by incorporating constraints on long-term cost signals. In this section, we describe the structure of the CMDP associated with each agent i.

State space The state space $\mathcal{S}_i$ for agent i consists of its local observation s_i, such that the full joint state is $(s_1, \ldots, s_N) \in \mathcal{S}$.

In the task offloading scenario considered, each agent's state is given by $s_i = (x_i, e_i)$, where x_i denotes the *Age of Information*, i.e., the number of timesteps since the last data processing was completed, and e_i represents the current energy level of the agent's battery.

Action space The action space $\mathcal{A}_i(s_i)$ includes a designated *crowded action*, a_{crowd}, which represents use of a shared resource. All other actions are independent of shared usage.

For example, in the task offloading scenario described in section 3, we have $\mathcal{A}_i(s_i) = \{$"wait", "local processing", "offload"$\}$, where "offload" corresponds to a_{crowd}.

Reward function The agent receives a reward according to the approximated formulation described in appendix A.1, where the influence of other agents is captured through the fixed (in the slow and intermediate timescale) parameter θ_{-i}:

$$\hat{r}_i(s, a; \theta_{-i}) = u_i(s_i) + \mathbb{I}_{a_i = a_{\text{crowd}}} \cdot d\,(1 + \theta_{-i})\,.$$

Cost function The cost signal $c_i(s_i, a_i)$ is a naive function active only when the shared resource is used:

$$c_i(s_i, a_i) = \begin{cases} 1, & \text{if } a_i = a_{\text{crowd}} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Constraint The constraint is defined by the parameter θ_i, which limits the long-term frequency with which agent i can select the crowded action.

While our formulation is based on the discounted reward criterion, the DCC-RL framework is not inherently restricted to this setting. In principle, the same decomposition and coordination structure can be applied in the average reward case, with appropriate adjustments to the underlying reinforcement learning algorithm. The choice between average and discounted formulations depends primarily on the application context and the algorithm used to solve each agent's constrained MDP. In the discounted case, we define the discounted reward and discounted cost under policy π_i and initial state distribution β_i as follows:

$$J_i(\pi_i, \beta_i; \theta_{-i}) = \mathbb{E}_{\pi,\, s_0 \sim \beta}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{t=0}^{T} \hat{r}_i(S_t, A_t; \theta_{-i}) \,\middle|\, S_0 = s_0\right] \tag{8}$$

$$K_i(\pi_i, \beta_i) = \mathbb{E}_{\pi,\, s_0 \sim \beta}\left[\sum_{t=0}^{\infty} \gamma^t c_i(S_t, A_t) \,\middle|\, S_0 = s_0\right] \tag{9}$$

The optimization problem faced by each agent, assuming fixed constraint parameters θ, is then:

$$\begin{aligned} \text{minimize:} \quad & J_i(\pi_i, \beta_i; \theta_{-i}) \\ \text{subject to:} \quad & K_i(\pi_i, \beta_i) \leq \theta_i \end{aligned} \tag{$CMDP_i$}$$

A.3 Learning the Optimal Local Constrained Policy

A central advantage of our framework is its flexibility: the local constrained policy optimization can be performed using any safe reinforcement learning (SRL) method suited to the environment. In particular, by adopting a Lagrangian-based approach, we decouple constraint enforcement from reward maximization. This separation enables the use of any standard reinforcement learning algorithm on the fastest timescale, with constraint satisfaction handled independently on a slower timescale. This modularity is crucial in decentralized settings with heterogeneous agents or dynamics, allowing DCC to adapt to a wide range of learning scenarios and observation structures.

Lagrangian methods are a natural fit for our setting, as they allow constraint satisfaction to be enforced without the need for projection or repair, and are compatible with standard RL algorithms. While such methods can exhibit instability or slow convergence—particularly in settings where constraint violations incur severe penalties or pose safety risks—these issues are mitigated in our

case. Here, constraints reflect system-level efficiency or coordination objectives rather than hard safety requirements, so temporary violations during training do not lead to catastrophic outcomes. As a result, we can safely use Lagrangian updates without sacrificing performance or stability, making them a principled and practical choice for decentralized learning under shared constraints.

In particular, we follow the structure of RCPO [29] for constrained optimization, employing either PPO [25], DQN [21] or Q-learning [30], depending on the scale of the system, as the algorithm to handle the fastest timescale

In this framework, in the fastest timescale each RL agent optimizes a shaped reward of the form:

$$\hat{r}_i^{\text{total}}(s_i, a_i; \theta_{-i}) = \hat{r}_i(s_i, a_i; \theta_{-i}) + \lambda_i c_i(s_i, a_i), \tag{10}$$

where $\hat{r_i}$ and c_i have been defined in appendix A.2.

The scalar coefficient λ_i serves as a Lagrange multiplier, adaptively tuned during training to ensure constraint satisfaction. To formalize this, we define the agent-specific Lagrangian function:

$$\mathcal{L}_i(\pi_i, \lambda_i; \theta) = \mathbb{E}_{\pi_i}\left[\sum_{t=0}^{\infty} \gamma^t \hat{r}_i(s_t, a_t; \theta_{-i}) + \lambda_i \left(c_i(s_t, a_t) - \theta_i\right)\right] \tag{11}$$

where θ_i is the fixed constraint threshold for agent i, and θ_{-i} denotes the sum of the vector of constraint parameters associated with all other agents. In this formulation, θ_{-i} is treated as fixed during the agent's local optimization, capturing the expected influence of other agents on the congestion dynamics. The k-th update of the Lagrange multiplier is then computed as follows:

$$\begin{aligned}
\lambda_i^{k+1} &= \Gamma_{\lambda_i}\left[\lambda_i^k + \eta_i(k)\nabla_\lambda \mathcal{L}_i(\pi_i, \lambda_i^k; \theta)\right] \\
&= \Gamma_{\lambda_i}\left[\lambda_i^k + \eta_i(k)\left(\mathbb{E}_{\pi_i}\left[\sum_{t=0}^{\infty}\gamma^t \sum_{t=0}^{T} c_i\right] - \theta_i\right)\right]
\end{aligned}$$

and Γ_{λ_i} is a projection on the space of possible Lagrange multipliers, i.e. the set of positive real numbers and $\eta_i(t)$ is a descending sequence converging to 0.

Crucially the value of λ_i is optimized for each agent independently, allowing the coordination signal to emerge in a fully decentralized manner. Our method enables agents to learn how the cost function should influence their behavior individually, in order to satisfy long-term constraints while maintaining local performance.

A.4 Coordination via optimization of the constraints

Having described the two faster timescales, we now focus on the slowest one, where the constraint vector θ is optimized to improve global coordination.

In particular, we want to show that we can converge to the local optimal value of the constraint vector $\theta \in \mathbb{R}^{\mathbb{N}}$. The safe RL algorithms described in appendix A.3 will then be able to find the optimal policy for that value.

We consider the long term Lagrangian reward with regard to θ, assuming that we can solve the corresponding constrained problem and find the optimal values of π and λ. This corresponds to the quantity we want to minimize for each agent. With abuse of notation, we define the quantity

$$\begin{aligned}
\hat{J}_i(\theta) &= \hat{J}(\pi_i^\star(\theta)) \\
&= \mathcal{L}_i(\pi_i^\star(\theta), \lambda_i^\star(\theta); \theta) \\
&= \mathbb{E}_{a\sim\pi_i^\star(\theta)}\left[\sum_{t=0}^{\infty}\gamma^t \sum_{t=0}^{T} \hat{r}_i(s_t, a_t; \theta_{-i}) + \lambda_i^\star \cdot \left(c_i(s_t, a_t) - \theta_i\right)\right]
\end{aligned} \tag{12}$$

and the function we want to minimize is

$$\hat{J}(\theta) = \sum_i \hat{J}_i(\theta) \tag{13}$$

First we show the differentiability of this function with regard to the constraint θ.

Lemma 2. *$\hat{J}_i(\theta)$ is differentiable in θ almost everywhere.*

Proof. The proof of this result is provided in appendix A.8, where we show that a variation of the Envelope Theorem [20] can be applied within the context of our setting. □

The following result formalizes the intuition regarding the sign of the derivatives of the objective function. In particular, it shows that relaxing the constraint for one device decreases its own reward (i.e., benefits it from a selfish perspective) while penalizing the others. This behavior arises naturally from the problem's structure: if one device is allowed to offload more frequently, it experiences a lower individual reward, whereas the remaining devices must offload less often due to the increased congestion penalty. We also numerically verified these results in appendix C.7, using linear programming to compute the optimal policy in small systems.

Proposition 2. *The following is verified:*

1. $\frac{\partial \hat{J}_i(\theta)}{\partial \theta_i} \leq 0$ *a.e.,*
2. $\frac{\partial \hat{J}_i(\theta)}{\partial \theta_j} \geq 0,\ \forall j \neq i$ *a.e.*

Proof. It is once again a consequence of theorem 2; in particular of the definition of the directional derivatives. The full proof is provided in appendix A.9. □

In order to discuss the optimization of the virtual constraint we want to use the stochastic gradient ascent methods of the Kiefer-Wolfowitz family [13]. The iteration scheme is

$$\theta^{n+1} = \Pi_\Theta \left(\theta^n - \alpha_n \hat{g}_n\right) \tag{14}$$

where θ^n is the nth iterate of the parameter, $\hat{g}_n$ represents an estimate of the gradient of the objective function, $\{\alpha_n\}_n$ is a sequence converging to 0 and Π_Θ is a projection on the space of possible virtual constraint vectors Θ.

If we drop λ and π for the sake of clarity, the i-th component of the gradient estimate writes

$$(\hat{g}_n)_i = \frac{\hat{J}\left(\theta + c_n \left(\Delta_n\right)_i\right) - \hat{J}\left(\theta\right)}{c_n (\Delta_n)_i} \tag{15}$$

where $\{c_n\}$ is a sequence converging to 0 and $\{\Delta_n\}$ is an i.i.d. vector sequence of perturbations of i.i.d. components $\{(\Delta_n)_i, i = 1, \ldots, N\}$ with zero mean and where $\mathbb{E}\left[|(\Delta_n)_i^{-2}|\right]$ is uniformly bounded.

The following result defines the conditions on the objective function, step-size sequence α_n, and gradient estimates $\hat{g}_n$ that give the convergence to a local minima.

Theorem 1. *Assume that*

- *$\{\alpha_n\}$ and $\{c_n\}$ be such that $\sum_n \alpha_n = \infty$, $\sum_n \left(\frac{\alpha_n}{c_n}\right)^2 < \infty$*
- *$\mathbb{E}\left[|(\Delta_n)_i|^{-2}\right]$ is uniformly bounded on Θ*

Then the algorithm converges to a local optimal value of $\min_\theta \sum_i \hat{J}_i(\theta)$.

Proof. Lemma 2 guarantees the differentiability of the objective function. The assumptions on the sequences α_n and c_n allow us to use Proposition 1 in [14] for a subset of Θ which contains the initial vector θ_0. This proves the convergence of the algorithm to a locally optimal solution. □

See Proposition 1 in [14] for the statement and the necessary conditions for the convergence to the global optimal point.

A.5 Numerical optimization of the constraints via evaluations of the local policies

We conclude this section by showing how the structure of the gradient can be further simplified, leading to a more efficient computation.

The following result, which is an immediate consequence of the chain rule, reveals how the computation of the gradient of the objective function can be simplified, thereby improving the overall efficiency of the algorithm:

Proposition 3.

$$\frac{\partial}{\partial \theta_i} \hat{J}(\theta_i, \theta_{-i}) = \frac{\partial \hat{J}_i(\theta_i, \theta_{-i})}{\partial \theta_i} + \sum_{j \neq i} \frac{\partial \, \hat{J}_j(\theta_i, \theta_{-i})}{\partial \theta_{-j}} \tag{16}$$

Proof. After applying the chain rule, it is sufficient to observe that $\frac{\partial \theta_{-j}}{\partial \theta_i} = 1$ when $j \neq i$. □

Proposition 3 illustrates how the problem structure, together with the form of the constrained value function, can be leveraged to compute the gradient of the approximate objective function efficiently. Specifically, this result allows each component of the gradient at a given point θ to be estimated using only three stochastic evaluations per agent, rather than $N + 1$.

- $\hat{J}_i(\theta_i, \theta_{-i})$
- $\hat{J}_i(\theta_i + \epsilon, \theta_{-i})$
- $\hat{J}_i(\theta_i, \theta_{-i} + \epsilon)$

The framework described in this section provides a principled and scalable approach to decentralized coordination in multi-agent systems with limited communication. By decomposing the global objective through constrained MDP formulations and optimizing across three timescales, agents can independently learn behaviors that align with system-level goals. The theoretical guarantees and structural properties of the reward function justify this decomposition and enable efficient learning despite the inherent coupling in agent behavior. These results lay the foundation for the empirical evaluation in the next section, where we assess the algorithm's performance under realistic edge computing scenarios.

A.6 Proof of Lemma 1

Proof.

$$\begin{aligned} J(\pi, \beta) - \hat{J}(\pi, \beta) &= \mathbb{E}_{a_t \sim \pi} \left[\sum_t \gamma^t \sum_i r_i(s_t, a_t) \mid s_0 \sim \beta \right] - \mathbb{E}_{a_t \sim \pi} \left[\sum_t \gamma^t \sum_i \hat{r}_i(s_t, a_t) \mid s_0 \sim \beta \right] \\ &= \mathbb{E}_{a_t \sim \pi} \left[\sum_t \gamma^t \sum_i \mathbb{I}_{a_{t,i} = a_{crowd}} \left(d(N(a_t) - d(1 + \theta_{-i}) \right) \right] \\ &\leq \frac{1}{1 - \gamma} \left(\sum_i \theta_i \mathbb{E}_{a_t \sim \pi} \left[d(N(a_t) - d(\mathbb{E}\left[N(t)\right]) \mid a_i = a_{crowd} \right] \right) \end{aligned}$$

In the case with linear penalty function d, we can easily conclude that $R_\pi(\beta) = \hat{R}_\pi(\beta)$. For the nonlinear case, first we show that

$$\mathbb{E}_{a_t \sim \pi} \left[d(1 + \theta_{-i}) \mid a_i = a_{crowd} \right] = d(1 + \theta_{-i})$$

When d is convex, we can notice that

$$\begin{aligned}\mathbb{E}_{a_t\sim\pi}\left[d(N(a_t) \mid a_i = a_{crowd}\right] \leq & \left(1 - \frac{\mathbb{E}\left[N_t \mid a_i = a_{crowd}\right] - 1}{N_{agents} - 1}\right) d(1) + \\ & + \frac{\mathbb{E}\left[N_t \mid a_i = a_{crowd}\right] - 1}{N_{agents} - 1} d(N_{agents}) \\ = & \frac{\mathbb{E}\left[N_t \mid a_i = a_{crowd}\right] - 1}{N_{agents} - 1} d(N_{agents}) \\ = & \frac{1 + \theta_{-i} - 1}{N_{agents} - 1} d(N_{agents}) \\ = & \frac{\theta_{-i}}{N_{agents} - 1} d(N_{agents}) \end{aligned} \tag{17}$$

Note that $d(1) = 0$ due to its definition. This implies that, for convex penalty function d it is verified that

$$J(\pi, \beta) - \hat{J}(\pi, \beta) \leq \frac{1}{1-\gamma} \sum_i \theta_i \left(\frac{\theta_{-i}}{N_{agents} - 1} d(N_{agents}) - d(1 + \theta_{-i}) \right)$$

Finally, for concave penalty function d, we can easily prove that

$$\mathbb{E}_{a_t\sim\pi}\left[d(N(a_t) \mid a_i\right] \geq \frac{\theta_{-i}}{N_{agents} - 1} d(N_{agents})$$

and therefore conclude that, for every nonlinear penalty function d it it verified that

$$\left| J(\pi, \beta) - \hat{J}(\pi, \beta) \right| \leq \frac{1}{1-\gamma} \sum_i \theta_i \left(\frac{\theta_{-i}}{N_{agents} - 1} d(N_{agents}) - d(1 + \theta_{-i}) \right) \tag{18}$$

□

A.7 Theoretical background for the proof of lemma 2

Before proving the main result, we first recall the relevant theoretical background, including the statement of Theorem 2. For clarity, we briefly review the envelope theorem, which applies broadly to optimization problems with parameterized constraints and objective functions defined over choice sets with arbitrary topology. A full treatment can be found in Section 3.5 of [20].

Consider the following maximization program with k parameterized inequality constraints:

$$\begin{aligned} V(t) &= \sum_{x \in X : g(x,t) \geq 0} f(x,t), \quad \text{where } g : X \times [0,1] \to \mathbb{R}^k \\ X^\star(t) &= \{x \in X : g(x,t) \geq 0, f(x,t) = V(t)\}. \end{aligned} \tag{19}$$

In the reference framework used by [20], X is a convex set, f and g are such that zero duality gap holds, and the Slater constraint qualification is verified at some $\hat{x} \in X$, i.e., $g_h(\hat{x}, t) > 0$ for all $h = 1, \ldots, k$.

The set of saddle points of the Lagrangian over $(x, y) \in X \times \mathbb{R}^k_+$ at parameter value t takes the form $X^\star(t) \times Y^*(t)$, where $X^*(t)$ is the set of solution to (19) and $Y^*(t)$ is the set of the solutions of the dual program

$$Y^*(t) = \arg\min_{y \in \mathbb{R}^\star_k} \left(\sup_{x \in X} L(x, y, t) \right)$$

When the zero duality gap condition holds, the value $V(t)$ of the constrained maximization problem equals the saddle of the Lagrangian with parameter t, i.e., $V(t) = \min_{y \in \mathbb{R}^k_+} \max_{x \in X} L(x, y, t) = \max_{x \in X} \min_{y \in \mathbb{R}^k_+} L(x, y, t)$

The following result, which is denoted as Corollary 5 in [20], is crucial to prove the differentiability of (13).

Theorem 2. *Suppose that X is a convex compact set in a normed linear space, f and g are continuous in x, for the Lagrangian zero duality condition holds, $f_t(x,t)$ and $g_t(x,t)$ are continuous in (x,t), and there exists $\hat{x} \in X$ such that $g(\hat{x},t) \gg 0$ for all $t \in [0,1]$. Then:*

1. *V is absolutely continuous and for any selection $(x^*(t), y^*(t)) \in X^*(t) \times Y^*(t)$,*

$$V(t) = V(0) + \int_0^t L_t(x^*(s), y^*(s), s)ds$$

2. *V is directionally differentiable, and its directional derivatives equal:*

$$V'(t+) = \max_{x \in X^*(t)} \min_{y \in Y^*(t)} L_t(x,y,t) = \min_{y \in Y^*(t)} \max_{x \in X^*(t)} L_t(x,y,t) \quad \textit{for } t < 1$$
$$V'(t-) = \min_{x \in X^*(t)} \max_{y \in Y^*(t)} L_t(x,y,t) = \max_{y \in Y^*(t)} \min_{x \in X^*(t)} L_t(x,y,t) \quad \textit{for } t > 0$$

The result holds since optimal dual variables are proved bounded, and hence from Theorems 4 and 5 in [20] can be applied.

A.8 Proof of lemma 2

Proof. In order to prove the desired result for the function $\hat{J}_i(\theta)$ we need to use theorem 2 and make a distinction between two separate cases, $t = \theta_i$ and $t = \theta_j$.
We consider

$$V(t) = \min_{y \in \mathbb{R}^k_+} \max_{x \in X} L(x,y,t) = \max_{x \in X} \min_{y \in \mathbb{R}^k_+} L(x,y,t)$$

Next we define how all the quantities involved in the two different cases.

A.8.1 $t = \theta_i$

In this case, all the values of θ_j for $j \neq i$ are fixed and simply define the function f.
Consider the following:

- $X = L^\gamma(\beta)$: this is the set of stationary distributions given an initial distribuiton β and the discount factor γ
- it is safe to assume the existence of a vector $\theta_{max} \in \mathbb{R}$ such that, $\theta_i < \theta_{max}$. This allows us to normalize the values of the parameter θ. A possible value of θ_{max} could be the cost obtained when always choosing the crowded action. In general, as long as the cost is a bounded function, this property is easily verified
- $f(x,t) = f_{\theta_{-i}}(\rho) = \sum_{s,a} \rho_i(s,a)\hat{r}_i(s,a;\theta_{-i})$, with $\hat{r}_i : \mathcal{S} \times \mathcal{A} \times [0,1]^{N-1} \to \mathbb{R}$ defined by the fixed parameters θ_{-i}
- $g(x,t) = g(\rho, \theta_i) = \sum_{s,a} \rho(s,a)c(s,a) - \frac{\theta_i}{\theta_{max}}$, with with $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$
- the Lagrangian is explicitly written as

$$L_i(\rho, \lambda, \theta_i) = f_{\theta_{-i}}(\rho) + \lambda_i g(\rho, \theta_i)$$

  assuming we are studying the differentiability wrt θ_i

Now we show that the hypotheses of Theorem 2 are verified.

X is a convex compact set in a normed linear space This choice of the space of the probability distributions, allows us to prove that X is a convex compact set in a normed linear space, as proved in Corollary 10.1 in [3].

f and g are continuous and concave in x The continuity and concavity in x of the functions f and g is an immediate consequence of the definition given above.

$f_t(x,t)$ and $g_t(x,t)$ are continuous in (x,t) The immediate reward function does not depend on θ_i, therefore $f_t(\cdot) = 0$. On the other hand, $g_t(\cdot) = \frac{1}{\theta_{max}}$

Existence of $\hat{x} \in X$ such that $g(\hat{x},t) \gg 0$ for all $t \in [0,1]$ The existence of a point $\hat{x} \in X$ such that $g(\hat{x},t) \gg 0$ for all $t \in [0,1]$ corresponds to the existence of a stationary distribution that strictly satisfies all constraints. This condition is met, for example, by a policy that never selects the crowded action—ensuring all constraints are strictly satisfied—provided that $\theta_i > 0$.

A.8.2 $t = \theta_j$

In this case, all the values of θ_k for $k \neq j$ are fixed; they will be considered fixed parameters in the definition of f and g. Consider the following:

- $X = L^\gamma(\beta)$ this is the set of stationary distributions
- it is safe to assume the existence of a vector $\theta_{max} \in \mathbb{R}$ such that, $\theta_i < \theta_{max}$. This allows us to normalize the values of the parameter θ. A possible value of θ_{max} could be the cost obtained when always choosing the crowded action. In general, as long as the cost is a bounded function, this property is easily verified
- $f(x,t) = f_{\theta_{-i,j}}(\rho, \theta_j) = \sum_{s,a} \rho_i(s,a)\hat{r}_i(s,a;\theta_{-i,j},\theta_j)$, with $\theta_{-i,j} = \sum_{k \neq i,j} \theta_k$
- $g(x,t) = g_{\theta_i}(\rho) = \sum_{s,a} \rho(s,a)c(s,a) - \frac{\theta_i}{\theta_{max}}$, with with $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$
- the Lagrangian is explicitly written as

  $$L_i(\rho, \lambda, \theta_j) = f_{\theta_{-i,j}}(\rho, \theta_j) + \lambda_i g_{\theta_i}(\rho)$$

  assuming we are studying the differentiability wrt θ_i

To conclude our proof, it suffices to show that the hypotheses of Theorem 2 are verified for both choices of the parameter.

X is a convex compact set in a normed linear space This choice of the space of the probability distributions, allows us to prove that X is a convex compact set in a normed linear space, as proved in Corollary 10.1 in [3].

f and g are continuous and concave in x The continuity and concavity in x of the functions f and g is an immediate consequence of the definition given above.

$f_t(x,t)$ and $g_t(x,t)$ are continuous in (x,t) The continuity of $f_t(x,t)$ in (x,t) is an immediate consequence of the definition of the reward function r_i and the assumption that it is continuous with regard to θ_{-i} and therefore it is differential wrt θ_j, $\forall j \neq i$. On the other hand, $g_t(\cdot) = 0$

Existence of $\hat{x} \in X$ such that $g(\hat{x},t) \gg 0$ for all $t \in [0,1]$ The existence of a point $\hat{x} \in X$ such that $g(\hat{x},t) \gg 0$ for all $t \in [0,1]$ corresponds to the existence of a stationary distribution that strictly satisfies all constraints. This condition is met, for example, by a policy that never selects the crowded action—ensuring all constraints are strictly satisfied—provided that $\theta_j > 0$.

A.8.3 Conclusion of the proof

Thanks to what is showed in appendices A.8.1 and A.8.2 we know that the objective function $V(t)$ is absolutely continuous with regard to t, for all choices of t.
Consider now the function

$$\hat{J}_i^\rho(\theta) = \mathbb{E}_{s,a \sim \rho_i^\star}\left[\hat{r}_i(s,a;\theta_{-i)}\right]$$

where $\rho_i^\star$ depends on θ.

When we fix all coordinates θ_k for $k \neq j$, the mapping

$$\theta_j \mapsto \hat{J}_i^\rho(\theta)$$

coincides with $V(\theta_j)$, which is known to be absolutely continuous.
It then follows that $\hat{J}_i^{\rho}(\theta)$ is differentiable a.e.
Finally, note how we can use Theorem 3.3 in [3] to show how an optimal solution $\rho^\star$ for LP is such that the stationary policy $\pi(\rho^\star)$ is optimal for the original constrained problem. This implies that, from the optimal stationary distribution we can also retrieve the optimal policy.

This allows us to conclude that the function $\hat{J}_i(\theta)$ is differentiable wrt every component of θ and therefore that theorem 1 is verified for the problem studied.

□

A.9 Proof of proposition 2

Proof. Theorem 2 states that V is directionally differentiable, and its directional derivatives equal:

$$V'(t+) = \max_{x\in X^*(t)} \min_{y\in Y^*(t)} L_t(x,y,t) = \min_{y\in Y^*(t)} \max_{x\in X^*(t)} L_t(x,y,t) \quad \text{for } t<1$$
$$V'(t-) = \min_{x\in X^*(t)} \max_{y\in Y^*(t)} L_t(x,y,t) = \max_{y\in Y^*(t)} \min_{x\in X^*(t)} L_t(x,y,t) \quad \text{for } t>0$$

When considering the parameter $t=\theta_i$ and θ_j fixed for $j\neq i$, we know that

$$\begin{aligned} L(\rho,\lambda,\theta_i) &= f_{\theta_{-i}}(\rho) + \lambda_i g(\rho,\theta_i) \\ &= \sum_{s,a} \rho_i(s,a)\hat{r}_i(s,a;\theta_{-i,j},\theta_j) + \lambda_i \left(\sum_{s,a} \rho(s,a)c(s,a) - \frac{\theta_i}{\theta_{max}} \right) \end{aligned}$$

It is easily verified that $\frac{\partial L(\theta_i)}{\partial \theta_i} = -\frac{\lambda}{\theta_{max}}$ and therefore when $\theta_i \in (0,\theta_{max})$ it is verified

$$V'(\theta_i+) = -\max_{\rho\in L^\gamma(\beta)} \min_{\lambda\in\mathbb{R}} \frac{\lambda}{\theta_{max}}$$
$$V'(\theta_i+) = -\min_{\rho\in L^\gamma(\beta)} \max_{\lambda\in\mathbb{R}} \frac{\lambda}{\theta_{max}}$$

As noted in [20], this is a special case for which it is verified that

$$V'(t) = V'(\theta_i+) = V'(\theta_i-) = -\frac{\lambda^\star}{\theta_{max}} \leq 0 \text{ a.e.}$$

This yields the corresponding result, with $M=\theta_{max}$.

On the other hand, when $t=\theta_j$, with $j\neq i$, we have

$$\begin{aligned} L(\rho,\lambda,\theta_j) &= f_{\theta_{-i,j}}(\rho,\theta_j) + \lambda_i g_{\theta_i}(\rho) \\ &= \sum_{s,a} \rho_i(s,a)\hat{r}_i(s,a;\theta_{-i,j},\theta_j) + \lambda_i \left(\sum_{s,a} \rho(s,a)c(s,a) - \frac{\theta_i}{\theta_{max}} \right) \end{aligned}$$

It then follows that

$$\begin{aligned} \frac{\partial}{\partial\theta_j} L(\rho,\lambda,\theta_j) &= \frac{\partial}{\partial\theta_j} \sum_{s_i,a_i} \rho_i(s_i,a_i)\hat{r}_i(s_i,a_i) \\ &= \frac{\partial}{\partial\theta_j} \sum_{s_i,a_i} \rho(s_i,a_i)\left(u_i(s_i,a_i) + \mathbb{I}_{a=a_{\text{crowd}}}(a_i)d(1+\theta_{-i})\right) \\ &= \frac{\partial}{\partial\theta_j} \sum_{s_i} \rho(s_i,a_{crowd})d(1+\theta_{-i}) \\ &= \sum_{s_i} \rho(s_i,a_{crowd}) \frac{\partial}{\partial\theta_j} d(1+\theta_{-i}) \geq 0 \quad \forall \rho,\lambda \end{aligned}$$

where the final inequality follows from the assumption that the function d is strictly increasing, i.e. $d' > 0$. Moreover, not how the derivative is equal to 0 if and only if the action a_{crowd} is never chosen. Therefore, we can further conclude that

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\hat{V}(\theta_j) &= \max_{\rho}\min_{\lambda}\sum_{s_i}\rho(s_i, a_{crowd})\frac{\partial}{\partial \theta_j}d(1+\theta_{-i}) \\ &= \sum_{s_i}\rho^{\star}(s_i, a_{crowd})\frac{\partial}{\partial \theta_j}d(1+\theta_{-i}) \geq 0\end{aligned}$$

where the optimal stationary distribution $\rho^{\star}$ is a function of the constraint vector θ. This quantity is larger than 0 due to the assumptions on the penalty function d. □

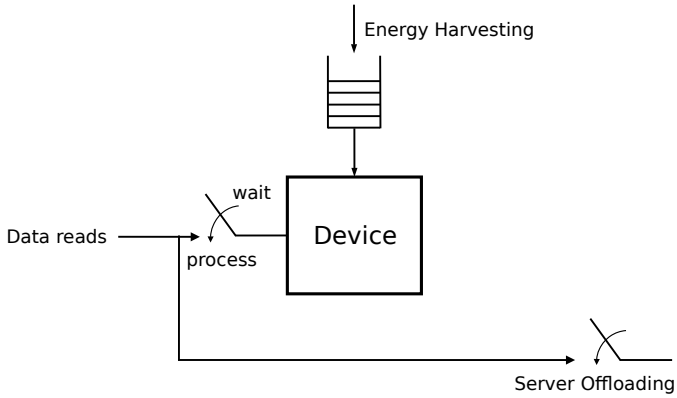


Figure 2: Device performing local processing of data batches with energy harvesting and offloading.

B Toy model considered

Description of the toy model The device-server scheme is represented in Fig. 2 for a single device: the device can read data batches, process them locally or offload them to the edge server. Thus, data batches are either read and processed locally on the device or read and offloaded. Time steps are discrete with index $t = 1, 2, \ldots$. To process data on the device at time t, a certain amount of energy units, denoted as C_t, is required. C_t is modeled as a Markov chain. When the device offloads to the edge server, the data processing task is offloaded and it has zero energy cost for the device. However, sending and retrieving processed data from the server may take some time, which depends on how many devices have offloaded the data. This will result in a reward that accounts for the delay in receiving the processed data. The energy to transmit data to the server is assumed negligible compared to that spent for local data processing. The device is equipped with a battery of capacity $B > 0$ energy units. Additionally it can harvest a certain number of energy units H_t per timestep t. As for the processing cost, the energy fetched per timestep, namely the *harvesting rate*, is described by a Markov chain.

Description of the CMDP For each device we can introduce a Constrained Markov Decision Process (MDP) to model the system. The state of the device i at time t is denoted as $s_{i,t} = (x_{i,t}, , e_{i,t})$, where $x_{i,t}$ represents the age of information (AoI), $e_{i,t}$ is the device battery level.

Note that $x_{i,t}$ is the AoI of the last data batch processed by a tagged device and the AoI is measured at the end of each time slot. It holds $x_{i,t} = 1$ when the device has just processed fresh data. Afterwards, the AoI increases of one timestep at every time slot t until either the device fetches a new data batch or the offloading occurs. In both cases, at the end of the data processing the AoI is reset to 1.

The freshness function, denoted as $u_i(x)$, represents the utility of processing a data batch x time units after the processing of the last batch. The function $u_i(\cdot)$ is bounded and non-increasing which means that beyond a certain value of AoI, any further processing delay would have minimal impact on the utility derived from the data. Consequently, it is assumed that there exists a positive constant M such that $u_i(M + k) = u_i(M)$ for any positive value of k, as it allows us to focus on the relevant time window $\{1, \ldots, M\}$ where the utility function exhibits meaningful changes in value in relation to the freshness of the data.

Finally, it is possible that the battery available at timeslot t is not sufficient to terminate the computation immediately, namely, $e_t + H_t - C_t < 0$. In this case, a delay is incurred in order to harvest a sufficient amount of energy and complete the processing of the current batch. The corresponding timeslot has a random duration, due to the stochastic nature of energy harvesting.

The state space of agent i is denoted $\mathcal{S}_i = \{1, \ldots, M\} \times \{0, \ldots, B\}$.

The action set is $\mathcal{A}_i = \{$"read", "local processing", "offload" $\}$. The action taken by agent i at time t is denoted $a_{i,t}$. If $a_{i,t} = 1$, device i fetches a new data batch, which is processed at energy cost

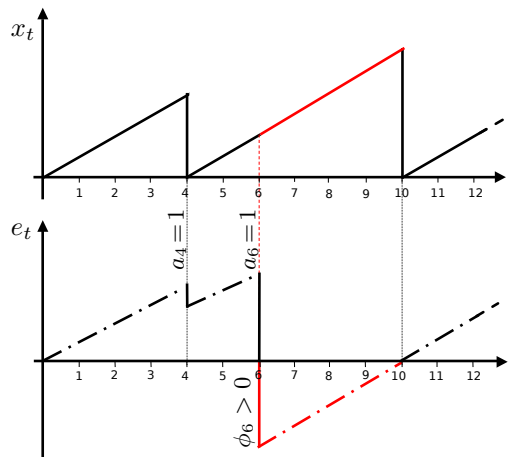


Figure 3: A sample path of the process $s_t = (x_t, e_t)$ for $H \equiv 1$ for a single agent: in red the vacation periods where the battery is empty. After the battery level gets negative at time $t = 6$ a total of 4 timesteps of recharging are needed to complete the execution of the task.

$C_t > 0$. Finally, the dynamics of the AoI for data batches is

$$x_{t+1} = \begin{cases} 1 & \text{if } a_t \in \{\text{"local processing" and } e_{i,t} + H_{i,t} - C_{i,t} \geq 0, \text{"offloading"}\} \\ \min\{x_t + 1, M\} & \text{if } a_t \in \{\text{"read", "local processing" and } e_{i,t} + H_{i,t} - C_{i,t} < 0\} \end{cases}$$

The AoI at the renewal instants is also called *peak AoI* ([4]) in the literature.

As described in section 3 the reward function is composed of two components: a local utility function and a component which represents the congestion. For the local components, we consider the reward function which penalizes having negative battery level:

$$u_i(s_i, a_i) = \begin{cases} x_i - e_i & e_i < 0 \\ x_i & e_i \geq 0 \end{cases} \tag{20}$$

For the sake of simplicity, for this work we consider a simple function also for the penalty component which depends on the congestion. In particular,

$$d(n) = n^{\alpha} \tag{21}$$

with $\alpha \in \mathbb{R}$. Clearly, for $\alpha = 1$ we are considering the case with linear penalty.

Finally, the cost function is defined as

$$c_i(s_i, a_i) = \begin{cases} 1 & a_i = \text{"offloading"} \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

C Additional numerical experiments

We introduce some additional experiments. Whenever reported, the reward is always normalized so that 1 corresponds to the discounted reward obtained by DCC-QL after 10^5 learning steps.

C.1 Evolution scalability comparison: additional data

We plot the reward evolution corresponding to the boxplot in section 5. This highlights why MAPPO results were excluded from the main figure to avoid distortion. The plot also shows that IQL converges quickly due to the small system size, while DCC-QL continues to improve over time as the constraint approaches its optimal value.

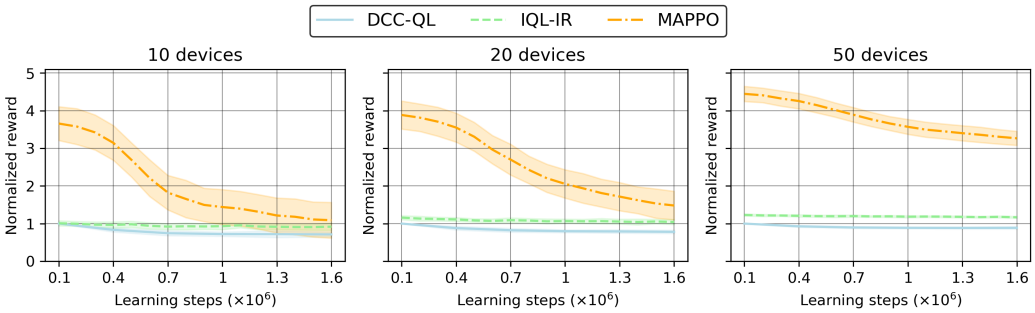


Figure 4: Evolution of the normalized reward over training episodes for different methods (corresponding to the boxplot in section 5).

C.2 Evolution offloading action: additional data

For completeness, we also plot the corresponding results in fig. 1b for $N = 20$ and $N = 50$.

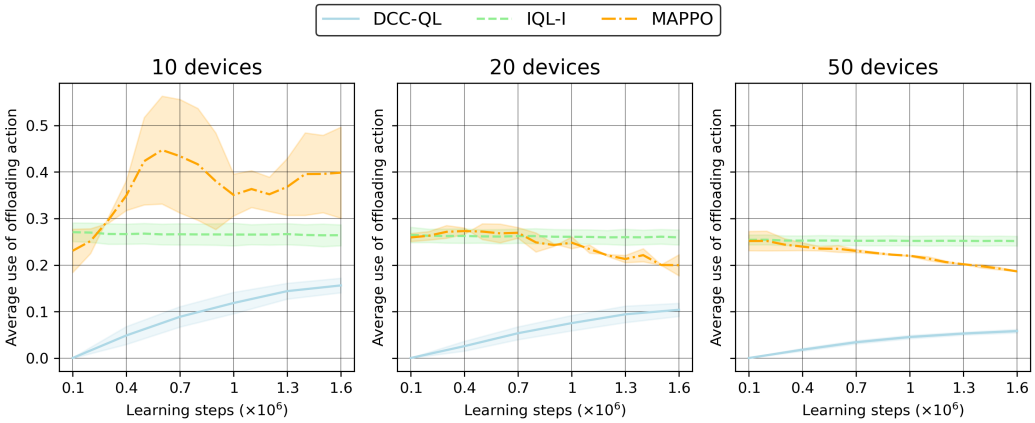


Figure 5: Evolution of the frequency of the offloading action in systems with $10, 20$ and 50 devices.

C.3 Comparison with MAPPO over longer timeframes

We include this comparison in Figure 6 to illustrate how MAPPO improves with extended training. While MAPPO achieves lower rewards after 10^7 learning steps, this comes at the cost of substantially more samples. In contrast, DCC-QL makes better use of available data thanks to its Q-learning foundation, achieving strong performance with far fewer interactions. Even with extended training, DCC-QL remains consistently better across system sizes, highlighting its sample efficiency and robustness.

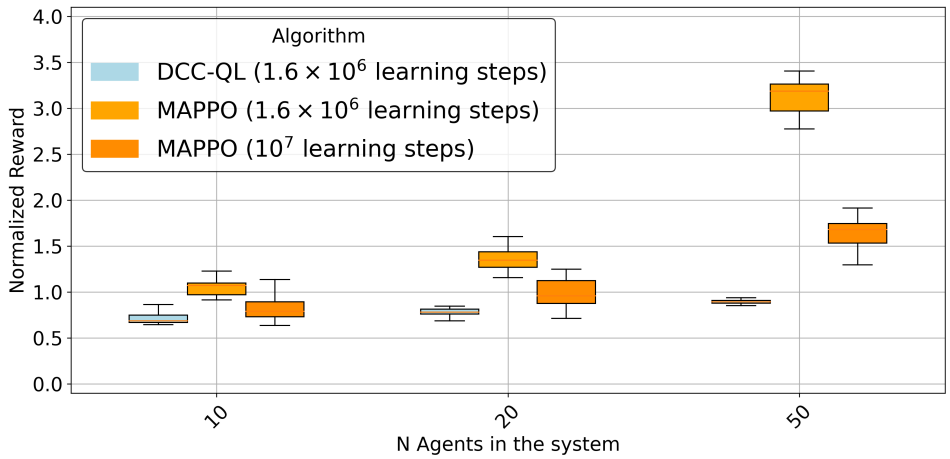


Figure 6: Comparison of DCC-QL with MAPPO after 1.6×10^6 learning steps (as in section 5) and after 10^7 learning steps.

C.4 Effect of Initial Constraint

In the main experiments, the environments considered were relatively homogeneous. In such cases, starting from uniform constraints is a natural choice. We therefore repeated the first main experiment from section 5, initializing DCC-QL with larger values of the constraints instead of $\theta = 0$. In particular, we chose as initial constraints values similar to the final values obtained by DCC-QL from naive initial constraints.

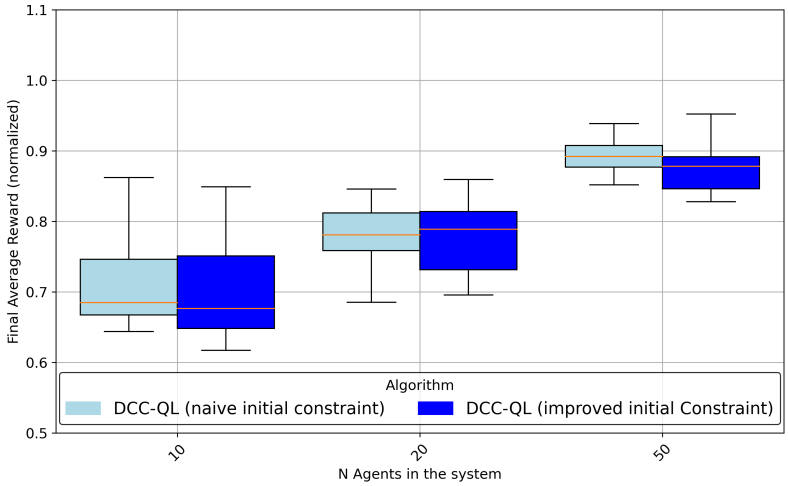


Figure 7: We compare the final normalized reward after 5 iteration of DCC-QL when starting from a naive constraint ($\theta = 0$) and when starting from an optimized value, where the optimized one has been chosen by looking at the final values obtained by DCC-QL when starting from naive initial constraints.

As shown in fig. 7, after five iterations of constraint improvement the final discounted reward is essentially the same for both initializations, with uniform initialization yielding slightly higher values. The difference is more pronounced in early iterations: when starting from uniform constraints, the algorithm achieves higher rewards sooner, particularly in systems with fewer devices. This confirms that a better prior on the constraints can accelerate learning, as observed in fig. 8.

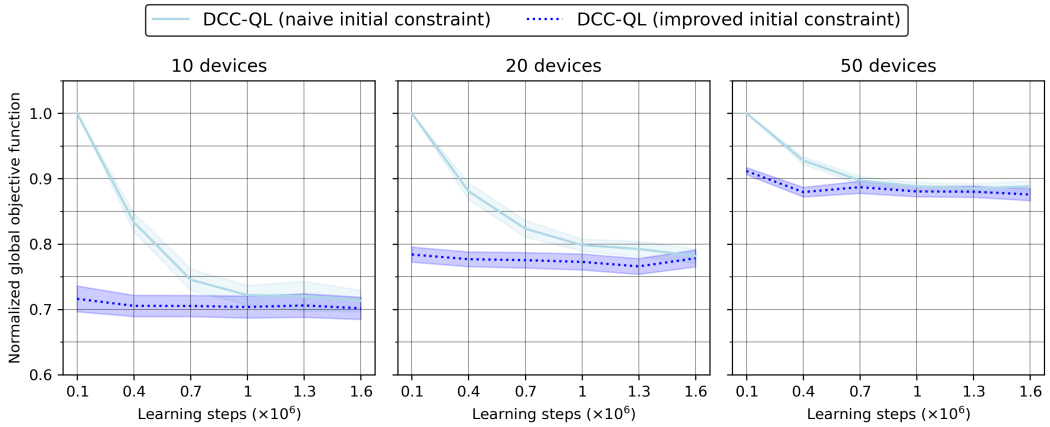


Figure 8: Comparison of the evolution of the reward as we start from optimized initial constraints in settings with a different amount of devices.

C.5 Additional algorithms for section 5

For completeness, fig. 9 also reports results for IQL with a common reward and for a centralized baseline using A2C. Sharing a global reward signal can, in principle, improve coordination among agents by aligning their incentives and reducing the risk of selfish behavior. However, this comes at the cost of increased variance in the learning signal, and in practice the benefits are limited in our setting. The centralized A2C results are obtained for small system sizes using the Stable Baselines implementation [22] with standard hyperparameters. This corresponds to a single agent observing the entire system and making joint decisions for all devices. To simplify the setup, we did not explicitly restrict illegal actions (e.g., processing with negative energy levels), but instead imposed large penalties, which is generally a suboptimal approach. While this method performs reasonably well with 10 devices, it fails to scale: for 20 devices, the average rewards were already about 15 times higher than the normalization factor, making meaningful comparison impossible. Thus, centralized methods serve here primarily as conceptual baselines rather than practical solutions.

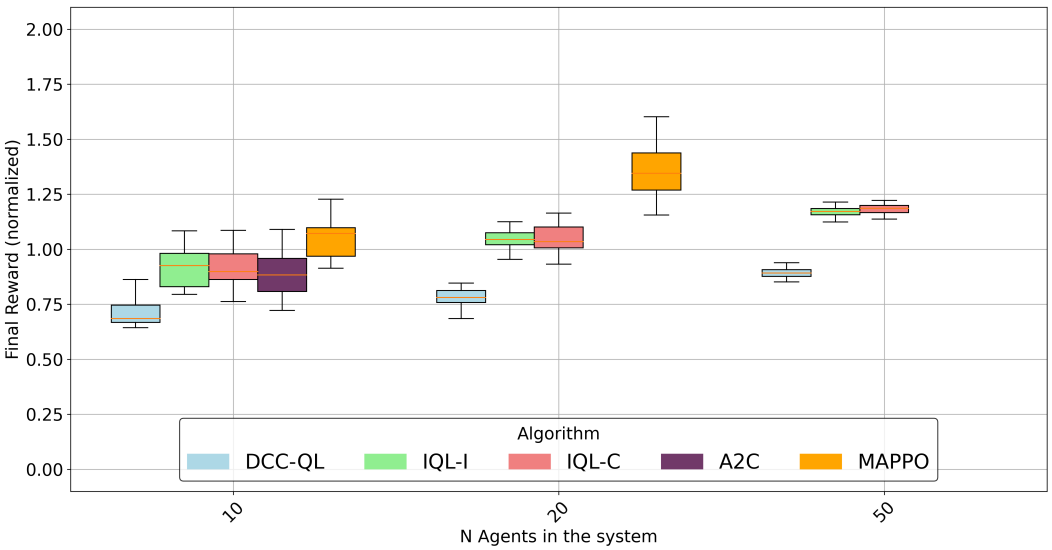


Figure 9: Performance comparison as in section 5, including IQL with common reward and centralized A2C (only for small system sizes).

C.6 Non linear penalty

We investigate how the frequency of the offloading action changes when varying the exponent α of the penalty function $d(n) = (n-1)^{\alpha}$. The experiment was carried out with 20 devices, since in the case of 10 devices the offloading frequency remains essentially unchanged across all values of α. This is likely because, with fewer devices, it is rare for many of them to simultaneously use the shared resource, making the penalty less pronounced. Moreover, the penalty is independent of α when exactly two devices offload at the same time, so differences only emerge when three or more devices compete for the shared resource—a situation that occurs more frequently with 20 devices.

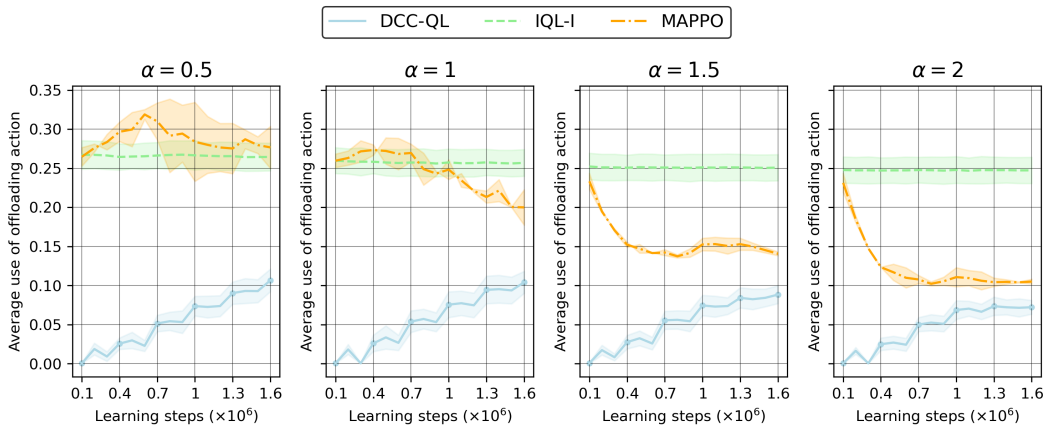


Figure 10: Evolution of the offloading action frequency for different values of the penalty exponent α. Increasing α leads to a consistent decrease in offloading frequency across algorithms, reflecting the stronger penalization of simultaneous offloading.

Figure 10 shows that, across all algorithms, the offloading frequency decreases as α increases, consistent with the expectation that higher exponents strengthen the penalty and discourage simultaneous offloading. The effect is particularly pronounced for DCC-QL and MAPPO.

C.7 Evaluation gradient objective function

We conducted an additional experiment to empirically validate the gradient properties derived in Proposition 2. For each value of the exponent of the penalty function, we generated 15 small random environments and solved each CMDP with the linear program to obtain the optimal policy and its discounted reward. First we perturbed the constraint vector by a finite noise ϵ and estimated gradients with respect to both the local component θ_i and the coupling term θ_{-i}. The results (Fig. 11) confirm the theoretical prediction and the intuition: the local gradient is consistently negative, while the coupling gradient is positive.

Then, using very small ϵ, we compared left and right finite-difference estimates, which matched closely, confirming differentiability (not reported). Moreover, in fig. 12 we verified that the local derivative equals the negative of the optimal Lagrange multiplier, and that the coupling derivative matches the expression in Proposition 2.

These findings validate proposition 2 and suggest a possible refinement of DCC-QL : instead of estimating local gradients via finite differences, the algorithm could directly substitute the Lagrange multiplier, potentially leading to a more efficient implementation. We leave this extension to future work.

C.8 Verification of the Approximate Reward Bound

In this experiment, we considered small environments with 10 devices, where the linear program (LP) can be applied reliably to compute the optimal policy. For each device, we compared the discounted reward obtained using the approximated reward definition in (6) with the corresponding component

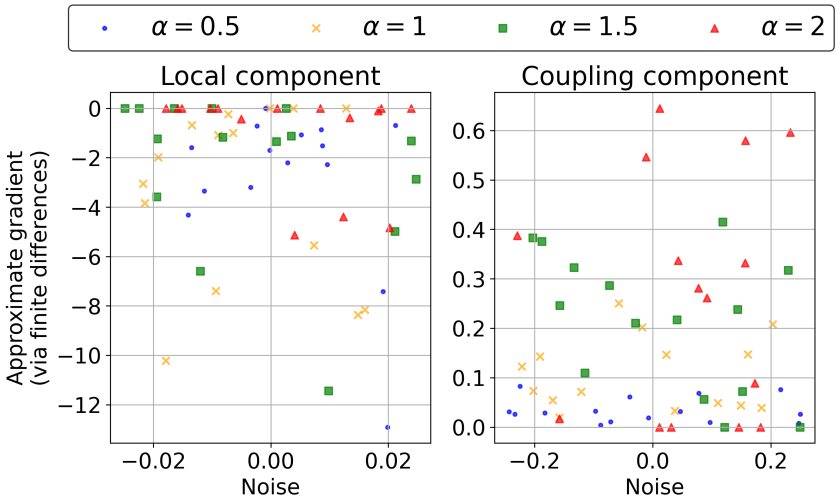
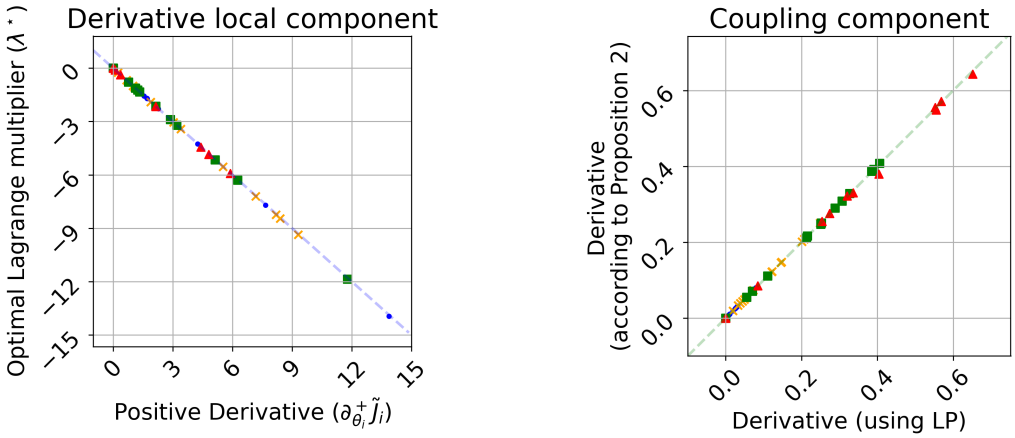


Figure 11: In these figures we evaluate an approximation of the gradient of $\tilde{J}_i^\ell(\theta)$ using the finite difference method. We considered a noise $\epsilon \in (0.01, 0.25)$ for both cases, In the left figure, representing the local component of the noise, we expected negative values, while in the right figure we expected positive values.



(a) Numerical verification that the finite-difference derivative of the objective with respect to the local noise coincides with the negative of the optimal Lagrange multiplier, confirming exact correspondence up to solver precision.

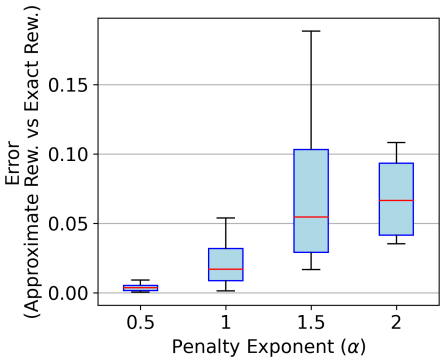
(b) Numerical verification that the finite-difference derivative of the objective with respect to the local noise coincides with the value given by proposition 2, confirming exact correspondence up to solver precision.

Figure 12: In these figures we evaluate an exact gradient of $\tilde{J}_i^\ell(\theta)$ using the finite difference method with a very small noise $\epsilon \in \{-0.00001, 0.00001\}$.

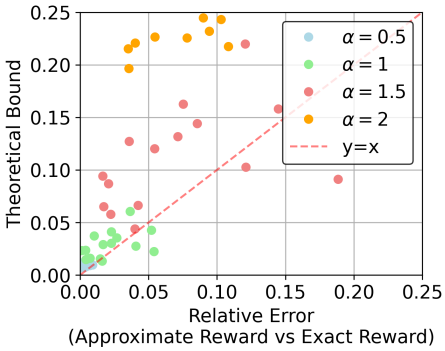
of the true global reward defined in (5). This comparison allows us to quantify the error introduced by the approximation and to assess its alignment with the theoretical analysis.

To this end, we measured the relative error between the two evaluations and compared it with the theoretical bound established in lemma 1. The results confirm that the observed error is consistently within the predicted range, up to small numerical deviations due to stochastic evaluation. Importantly, the special case $\alpha = 1$, where the penalty function is linear and the theoretical bound reduces to zero, was verified exactly in our experiments. For all nonlinear values of α, the empirical errors remained strictly below the bound, thereby validating the theoretical result in practice.

These findings provide additional evidence for the robustness of the approximation framework, showing that the theoretical guarantees extend to practical instances solved via the LP in small environments.



(a) Boxplot of the relative error as a function of the penalty exponent α



(b) Relative error compared with the theoretical bound from lemma 1, showing that the error remains within the predicted range in most cases.

Figure 13: Evaluation of the approximation error between the discounted reward computed with the exact reward and with the approximated reward.

D Extended Algorithm

Algorithm 1 Three-Timescale Decentralized Constrained MARL

Initialize: Initial constraints θ_i^0, multipliers λ_i^0, policies π_i^0
function EVALUATE_OPTIMAL_CONSTRAINED_POLICY$(i, \theta_i, \theta_{-i}, n)$
  for $m = 0, 1, \ldots, M(n)$ **do** ▷ Intermediate timescale — λ update
    for $t = 0, 1, \ldots, T(m, n)$ **do** ▷ Fastest timescale — policy update
      Collect trajectories using policy $\pi_i^{t,m,n}$
      Compute total reward: $r_i^{\text{total}}(s_i, a_i; \theta_{-i}) = \hat{r}_i(s_i, a_i; \theta_{-i}) + \lambda_i c_i(s_i, a_i)$

      Update policy: $\pi_i^{t+1,m,n} \leftarrow$ RL step on r_i^{total}
    end for
    $\hat{K}_i \leftarrow K_i(\pi_i^{\cdot,m,n})$ ▷ Estimate long term cost of current policy
    $\lambda_i^{m+1,n} \leftarrow \lambda_i^{m,n} + \alpha_k(\hat{K}_i - \theta_i)$
  end for
  return $\hat{J}_i(\theta_i, \theta_{-i}) \leftarrow$ evaluate local policy $\pi_i^{T(M(n),n),M(n)}$
end function

for $n = 0, 1, 2, \ldots$ **do** ▷ Slowest timescale — constraint update
  for each agent $i = 1, \ldots, N$ **do**
    $\hat{J}_i(\theta_i^n, \theta_{-i}^n) \leftarrow$ EVALUATE_OPTIMAL_CONSTRAINED_POLICY$(i, \theta_i^n, \theta_{-i}^n, n)$
  end for
  $\epsilon \leftarrow$ RandomNoise(0, σ)
  for each agent $i = 1, \ldots, N$ **do**
    $\hat{J}_i(\theta_i^n + \epsilon_i, \theta_{-i}^n) \leftarrow$EVALUATE_OPTIMAL_CONSTRAINED_POLICY$(i, \theta_i^n + \epsilon_i, \theta_{-i}, n)$
    $\hat{J}(\theta_i^n, \theta_{-i}^n + \epsilon_i) \leftarrow$EVALUATE_OPTIMAL_CONSTRAINED_POLICY$(i, \theta_i^n, \theta_{-i} + \epsilon_i, n)$
  end for
  for each agent i **do**
    $\frac{\partial}{\partial \theta_i} \hat{J}_i(\theta_i^n, \theta_{-i}^n) \leftarrow \left(\hat{J}_i(\theta_i^n + \epsilon_i, \theta_{-i}^n) - \hat{J}_i(\theta_i^n, \theta_{-i}^n) \right) / \epsilon_i$
    $\frac{\partial}{\partial \theta_{-i}} \hat{J}_i(\theta_i^n, \theta_{-i}^n) \leftarrow \left(\hat{J}_i(\theta_i^n, \theta_{-i}^n + \epsilon_i) - \hat{J}_i(\theta_i^n, \theta_{-i}^n) \right) / \epsilon_i$
    $\nabla \frac{\partial}{\partial \theta_i} \hat{J}(\theta) = \frac{\partial \hat{J}_i(\theta)}{\partial \theta_i} + \sum_{j \neq i} \frac{\partial \hat{J}_j(\theta)}{\partial \theta_{-j}}$
    $\theta_i^{n+1} \leftarrow \theta_i^n + \eta_n \nabla_{\theta_i} \hat{J}_i(\theta_i^n)$
  end for
end for

E Numerical experiments hyperparameters

Parameters of the environment To keep a simple system, we considered both harvesting probability and processing cost probability, in an interval respectively $[min_H, max_H]$ and $[min_C, max_C]$. The following table describes the values among which we sampled the values for each experiment.

Parameter	Sample set
M	15
B	15
min_H	{0, 1}
min_H	{1, 2, 3}
min_H	{1}
max_C	{5, 7, 10}
γ	0.95

E.1 Learning parameters

Table 1 reports the learning parameters used in the experiments for IQL and DCC-QL . We tested several learning and exploration rates, but the values shown here yielded the best performance. Symbols in parentheses correspond to those used in algorithm 1. For DCC-QL , the parameters α_k, σ, and η_n denote the initial values of sequences that decay according to the convergence conditions of the algorithm; these are not applicable to IQL.

Parameter	DCC-QL	IQL
Learning rate Q-learning	0.5	0.05
Exploration rate Q-learning	0.05	0.05
Exploration decaly Q-learning	0.95	0.95
LM learning rate (α_k)	1	
Standard deviation constraint noise (σ)	0.05	
Constraint learning rate (η_n)	0.25	

Table 1: Learning parameters for DCC-QL and IQL

For MAPPO, we used the implementation in [10] for the custom environment described in appendix B. |lc|

	MAPPO
LR	0.001
NUM_ENVS	8
TOTAL_TIMESTEPS	1.6e7
UPDATE_EPOCHS	2
NUM_MINIBATCHES	64
GAMMA	0.95
GAE_LAMBDA	0.95
CLIP_EPS	0.2
ENT_COEF	0.01
VF_COEF	0.5
MAX_GRAD_NORM	0.5
ACTIVATION	relu

Table 2: Learning parameters MAPPO