From Offline Global Information to Online Decentralized Policies in Edge Network Scheduling

Anonymous Authors[1]

Abstract

In edge computing, end-to-end job performance depends on both client-side communication scheduling and server-side computation scheduling. However, these schedulers are physically separated and must make online decisions using only local queue observations. As a result, common heuristics such as Earliest Deadline First (EDF) and Shortest Remaining Time First (SRTF) can be globally suboptimal. We study whether offline global information can be used to train decentralized scheduling policies for online deployment. We explore two approaches: supervised imitation of Integer Linear Programming (ILP) oracle schedules, and multi-agent reinforcement learning with centralized training and decentralized execution. Across simulated edge workloads, we show that learned local policies outperform static heuristics by up to 37% despite having no access to global state at runtime. These results suggest that offline global supervision can distill coordinated scheduling behavior into decentralized policies under partial information.

1. Introduction

Distributed edge applications rely on multiple physically separated controllers to meet end-to-end latency and deadline requirements. In tandem edge scheduling, a job first waits for client-side communication and then for server-side computation. Its completion time therefore depends on queueing and scheduling decisions across both stages. An optimal scheduler pair would use a global view of the system, including both queues, job requirements, and future arrivals.

However, such global context is often unavailable during online execution in distributed settings. Controllers may be geographically separated, and exchanging state through message passing can introduce additional latency, and deployment complexity. As a result, each scheduler only observes its local queue state and workload characteristics, and relies on heuristics that optimize local priorities, such as Earliest Deadline First (EDF) (Liu & Layland, 1973) or Shortest Remaining Time First (SRTF) (Silberschatz et al., 2018). While these heuristics are simple and efficient, they are not designed to optimize global objectives such as application-level requirements that span both components.

At the same time, real-world deployments can collect abundant offline data that captures complete workload states, making it possible to compute globally optimized scheduling decisions using full-information solvers. With the help of simulators, it is also possible to train agents with full information. This raises the central question of this paper: *Can we leverage offline data with global context to train local policies that execute online using only local observations, while still optimizing for global objectives?* We study this question in the context of tandem edge scheduling, where communication and computation schedulers must coordinate implicitly without online communication. We formulate the problem as an offline-to-online decentralized decision-making task: during offline training, policies can learn from global workload information or full-information oracle schedules; during online deployment, each scheduler must make decisions independently using only its local state.

We explore two learning-based approaches. First, we train supervised policies to imitate ILP-generated oracle schedules, distilling globally optimized offline decisions into per-timestep local scheduling actions. Second, we study multi-agent reinforcement learning under the centralized-training decentralized-execution paradigm, where agents can use global state during training but are restricted to local observations during deployment. We compare these learned policies against static local heuristics and evaluate whether offline global supervision can produce decentralized policies that improve online scheduling performance.

This paper makes the following contributions:

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

- We formulate tandem edge scheduling as an offline-to-online decentralized decision-making problem.
- We train local scheduling policies from offline global decisions, using both supervised imitation of ILP oracle schedules and multi-agent reinforcement learning with centralized training and decentralized execution.
- We evaluate learned policies against local scheduling heuristics under decentralized online deployment, showing that policies trained with offline global context can better capture cross-scheduler coordination and improve global scheduling objectives.

2. Background and Motivation

2.1. Tandem Scheduling in Edge Networks

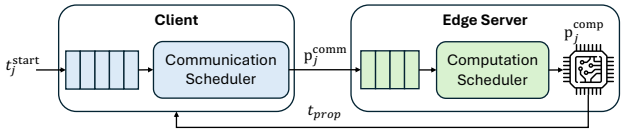


Figure 1. Edge network job flow.

The rapid growth of emerging mobile edge applications (Ren et al., 2019; Kang et al., 2022; Mao et al., 2017), creates increasing demand for ultra-low end-to-end latency while minimizing deadline misses. In these applications, a typical job workflow is shown in Figure 1. A job j arrives at the client at time t_j^{start}. The communication scheduler decides when the job is transmitted over the uplink, requiring communication time p_j^{comm}. After the job is fully received by the edge server, it enters the computation queue, where the computation scheduler decides when to process it, requiring computation time p_j^{comp}. Once computation completes, the result is returned to the client with a downlink propagation delay t_{prop}. The job is considered successful only if it completes before its deadline d_j; otherwise, it is dropped. The schedulers can perform early dropping when a job is mathematically unable to meet its deadline. In online deployment, each scheduler only observes its local queue state, making global optimization difficult to implement directly.

2.2. Why Local Heuristics Fall Short

In practice, online schedulers often rely on simple local heuristics. We assume preemptive schedulers, where both communication and computation can be interrupted and resumed. Two common scheduling heuristics are Earliest Deadline First (EDF), which prioritizes the job with the nearest deadline, and Shortest Remaining Time First (SRTF), which prioritizes the job with the smallest remaining processing time. However, EDF and SRTF optimize local priorities rather than the global objective of the tandem system. We also run an Integer Linear Programming (ILP) solver assuming perfect information to serve as an oracle baseline. It minimizes $C_{\max} + \lambda N_{\text{drop}}$, where $C_{\max}$ is the makespan (maximum completion time among all jobs), N_{drop} is the number of dropped jobs, and λ is a large drop penalty.

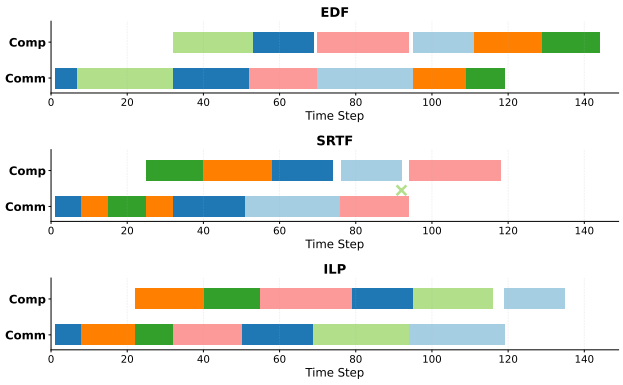


Figure 2. Timeline of heuristic schedulers versus the ILP oracle. Cross markers indicate missed deadlines

Figure 2 shows an example comparing EDF, SRTF, and ILP. Under EDF, the communication scheduler prioritizes transmitting the light green job because it has the tightest deadline. However, this job requires long communication time p_j^{comm}, causing head-of-line blocking in the communication queue and leaving the computation resource underutilized for a long period. Under SRTF, the scheduler prioritizes shorter jobs, improving local resource usage, but it ignores cross-stage deadline constraints and causes the tight-deadline light green job to miss its deadline. In contrast, the ILP schedule uses full workload information to coordinate communication and computation decisions, reducing both deadline misses and inefficient pipeline behavior.

These examples show that local priority rules are poor proxies for the global objective: one scheduler's decision directly changes the future state observed by the other. Tandem edge scheduling therefore requires coordinated behavior across both stages.

3. Design

As shown in Section 2, a full-information ILP oracle can produce high-quality schedules but requires knowledge of the full workload trace in advance, making it infeasible for online deployment. In this section, we formulate the online decentralized scheduling problem and describe two approaches for using offline global information to train local policies: imitation learning from ILP oracle schedules and multi-agent reinforcement learning with centralized training and decentralized execution.

3.1. Online Decentralized Tandem Scheduling

Online deployment requires each scheduler to make decisions using only local observations. We formulate the tan-

dem edge scheduling problem as a decentralized partially observable Markov decision process (Dec-POMDP) (Bernstein et al., 2002), where the communication and computation schedulers are two agents that act independently at runtime.

At each decision time t, each agent $i \in \{\text{comm}, \text{comp}\}$ observes only its local queue Q_t^i. Its observation is represented as $o_t^i \in \mathbb{R}^{N\times 5}$, where $N = |Q_t^i|$ is the number of jobs in the local queue. Each job $j \in Q_t^i$ is encoded by the feature vector $[d_j - t, u_j^{\text{comm}}, u_j^{\text{comp}}, w_j^{\text{comm}}, w_j^{\text{comp}}]^\top$, where $d_j - t$ is the remaining time until the deadline, u_j^{comm} and u_j^{comp} are the remaining communication and computation times, and w_j^{comm} and w_j^{comp} are the cumulative queueing times in the two stages.

Given its local observation, agent i selects an action a_t^i, which indexes a job from Q_t^i for processing, or idles when the queue is empty. The communication scheduler and computation scheduler therefore execute decentralized policies: $\pi^{\text{comm}}(a_t^{\text{comm}} \mid o_t^{\text{comm}})$, $\pi^{\text{comp}}(a_t^{\text{comp}} \mid o_t^{\text{comp}})$. We optimize the same objective used by the ILP oracle: $C_{\max} + \lambda N_{\text{drop}}$.

This creates an offline-to-online learning problem: during training, we can use global workload traces, oracle schedules, or simulator feedback; during deployment, each scheduler must act independently using only its local observation.

3.2. Imitation Learning from ILP Oracle Schedules

The imitation-learning approach treats the ILP oracle schedule as the label for supervised training. For each offline workload trace, we first solve the full-information ILP to obtain an oracle schedule. The oracle specifies, at each time step, which job should be processed by both schedulers. We train separate supervised models for communication and computation.

We consider two supervised imitation formulations: *SL-E2E* and *SL-Step*. In *SL-E2E*, the model inputs a full job set and learns to predict the entire communication or computation schedule in a single forward pass. This formulation is efficient for offline planning, but is less suitable for dynamic online deployment because it assumes the full workload is known in advance.

In contrast, *SL-Step* converts each ILP solution into per-timestep labeled data. At each time step t, we record the local observation and the oracle-selected action for each scheduler: $(o_t^{\text{comm}}, a_t^{\text{comm},*}), (o_t^{\text{comp}}, a_t^{\text{comp},*})$. The trained policy is then deployed online by observing the current local queue state and predicting the next job to process. It forces the learned policy to approximate globally coordinated ILP behavior from local observations, while allowing real-time adaptation to changing queue states.

Imitation learning has the advantage of directly leveraging high-quality ILP schedules. However, it also has two limitations. First, ILP schedules may be expensive to generate for large workloads. Second, supervised imitation relies on the model's ability to generalize from oracle-visited states to new states encountered during deployment, and its behavior becomes less reliable when observations are far from training data.

3.3. Multi-Agent Reinforcement Learning

We also study multi-agent reinforcement learning (MARL) that bypasses the need for pre-generated ILP labels. Instead, agents collect training data by interacting with the simulator and learning from the consequences of their own actions.

We follow the centralized training with decentralized execution (CTDE) paradigm. During training, the critic or value-mixing module can access centralized state information as the concatenation of both scheduler local observations: $s_t = [o_t^{\text{comm}} \parallel o_t^{\text{comp}}]$. During execution, however, each scheduler observes only its local queue state. At each decision step, the simulator advances the system state.

To provide denser training feedback while still optimizing the end-to-end scheduling objective, we use both step-level and episodic rewards. The step-level reward penalizes local queueing delay and projected deadline violations. For each agent i, we define the base immediate reward as $\hat{r}_t^i = -\sum_{j\in Q_i} \left(\alpha_1 w_j^i + \alpha_2 \cdot \mathbf{1}(e_j > d_j)\right)$, where $e_j = t + u_j^{\text{comm}} + u_j^{\text{comp}} + t_{\text{prop}}$ is the earliest possible completion time of job j. The indicator term thus penalizes jobs that are already projected to miss their deadline (and therefore dropped).

Because the communication scheduler affects the computation scheduler's future workload, we add an additional downstream penalty to the communication reward when jobs in the computation queue are dropped: $r_t^{\text{comm}} = \hat{r}_t^{\text{comm}} - \sum_{j\in Q_{\text{comp}}} \alpha_3 \cdot \mathbf{1}(e_j > d_j)$. The computation scheduler uses only its base reward: $r_t^{\text{comp}} = \hat{r}_t^{\text{comp}}$. This reward design encourages the communication agent to account for downstream computation congestion rather than greedily optimizing its local queue alone.

At the end of each episode, we also apply an episodic reward aligned with the global scheduling objective: $R = -(\beta_1 C_{\max} + \beta_2 N_{\text{drop}})$, The parameters α, β control the relative weights of queueing delay, makespan, and dropped jobs.

Using this formulation, we evaluate three MARL algorithms under CTDE: Value-Decomposition Networks (VDN), QMIX, and Multi-Agent Proximal Policy Optimization (MAPPO). VDN estimates the joint action-value function as the sum of individual agent Q-values, enabling decentralized policies to learn from a shared team objec-

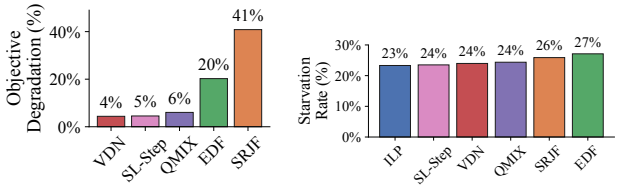
tive (Sunehag et al., 2017). QMIX generalizes VDN by using a monotonic mixing network, allowing richer joint value functions while preserving decentralized action selection (Rashid et al., 2018). MAPPO uses decentralized actor policies with a centralized critic that observes the global state during training (Yu et al., 2022).

MARL policies can discover decentralized strategies from a more flexible search space. However, MARL introduces its own challenges. The action of one scheduler changes the future state observed by the other scheduler, creating a non-stationary learning problem. The reward can also remain sparse or delayed, since final deadline misses and makespan are only fully known after all jobs complete.

4. Evaluation

4.1. Data Collection

We construct workload traces from two applications in Zhang & Kim (2026): file transfer and video transcoding. Each trace contains six jobs, with three from each application. Computation times are measured from cloud runs, uplink communication times are estimated from packet sizes using the reported linear model, and downlink delay is modeled as a constant. We collect 1,000 traces and split them into 700 training and 300 testing traces.



(a) **Normalized objective score:** Objective degradation over the ILP baseline; lower is better.

(b) **Pipeline starvation rate:** Percent of timesteps where the computation queue was empty while the communication scheduler was busy; lower is better.

Figure 3. In-distribution performance comparison.

4.2. In-Distribution Evaluation

We first evaluate policies on held-out test workloads sampled from the same distribution as the training workloads. Figure 3a compares the objective value across EDF, SRTF, imitation learning, MARL, and ILP. The learned policies outperform static local heuristics, showing that offline global information can be distilled into decentralized online policies.

We also analyze the pipeline starvation rate, which measures how often the computation scheduler is idle while waiting on the upstream communication scheduler. Figure 3b shows that SL-Step, VDN, and QMIX had lower starvation rates than the heuristics. SL-E2E and MAPPO performed poorly and are excluded for visual clarity (see Appendix for full results).

4.3. Generalization Experiments

We next evaluate whether learned policies generalized beyond the training distribution. For evaluation, in addition to the previous 3-3 test set, we separately sample 300 job workloads from each of the following file-video splits: 1-5, 2-4, 4-2, 5-1.

SL-Step outperforms the heuristic baselines on the 4-2 and 2-4 splits, which are closest to the original 3-3 training distribution, but performs worse than the baselines on the more shifted 5-1 and 1-5 splits. This suggests that SL-Step can tolerate moderate changes in workload composition, but its learned policy remains sensitive to larger distribution shifts. The MARL approaches show a similar trend: they perform better near the training distribution but degrade as the job-type mix becomes more imbalanced. Among them, QMIX exhibits the largest performance variation, suggesting that its learned value decomposition may be more sensitive to changes in the relative importance of communication and computation across workload types.

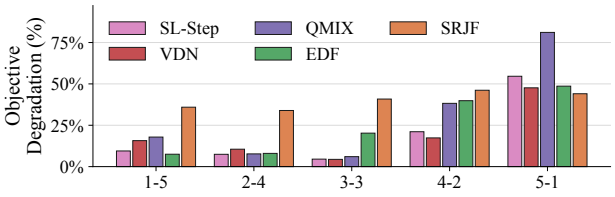


Figure 4. **Zero-shot evaluation:** Models trained only on the 3-3 split and evaluated on all splits.

5. Conclusion and Future Work

This paper studies whether offline global scheduling knowledge can be distilled into online decentralized policies for tandem edge workloads. We formulate the problem as a Dec-POMDP and evaluate ILP imitation and CTDE-based MARL. Our results show that learned policies, especially SL-Step and value-decomposition-based MARL, can outperform static local heuristics and reduce pipeline starvation, while mixed-distribution training improves generalization across workload compositions.

Future work will scale the evaluation to larger workloads, more application types, and more realistic edge deployments. We are also interested in learning offline global state representations that expose cross-scheduler dependencies more explicitly, and in studying simulation-to-real adaptation under latent system factors and measurement noise.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

Kang, D., Romero, F., Bailis, P., Kozyrakis, C., and Zaharia, M. VIVA: An end-to-end system for interactive video analytics. In *12th Annual Conference on Innovative Data Systems Research (CIDR '22)*, 2022.

Liu, C. L. and Layland, J. W. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):40–61, 1973. doi: 10.1145/321738.321743.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017. doi: 10.1109/COMST.2017.2745201.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 4295–4304. PMLR, 2018.

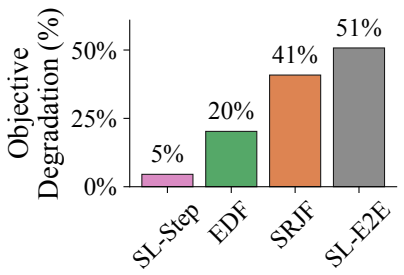
Ren, J., He, Y., Huang, G., Yu, G., Cai, Y., and Zhang, Z. An edge-computing based architecture for mobile augmented reality. *bibtexIEEE Networkbibtex*, 33(4):162–169, 2019. doi: 10.1109/MNET.2018.1800132.

Silberschatz, A., Galvin, P. B., and Gagne, G. *Operating System Concepts*, volume 10. Wiley, 2018.

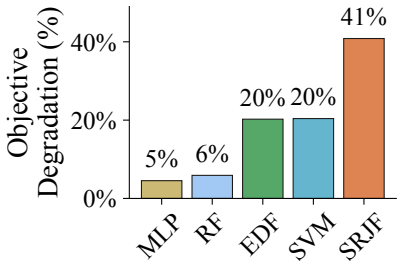
Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of PPO in cooperative, multi-agent games. In *Advances in Neural Information Processing Systems*, 2022.

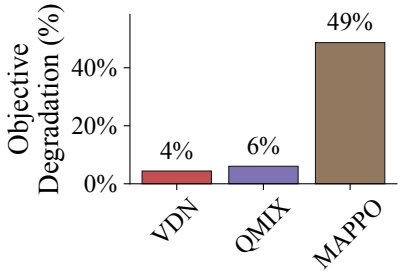
Zhang, X. and Kim, D. Enabling SLO-aware 5G multi-access edge computing with SMEC. In *Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. USENIX Association, 2026.



(a) **SL-E2E:** SL-E2E performed worse than the baseline heuristics on the base 3-3 split, even after tuning hyperparameters like model size.



(b) **Supervised Learning Classifiers:** MLP performed best among the three classifiers, with Random Forest close behind and Linear SVM lagging substantially. The gap suggests the decision boundary between scheduling actions is non-linear, which Linear SVM cannot capture.



(c) **MARL:** Objective degradation graph for VDN, QMIX, MAPPO. MAPPO performs poorly due to its simpler joint value approximations.

Figure 5. In-distribution performance comparison.

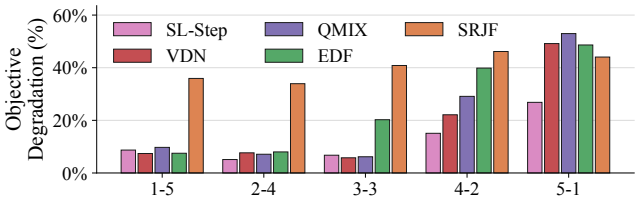


Figure 6. **Mixed-distribution retraining:** Models retrained on all splits and re-evaluated on the hold-out sets. After retraining, both models improve across nearly all held-out splits.

A. Appendix

We show additional comparison results of in-distribution performance comparison with SL-E2E, various supervised learning classifiers, and MAPPO in Figure 5.

We also show generalization results with retrain in Figure 6. After retraining, both types of models improve across nearly all held-out splits. Performance in the 3-3 split decreases slightly relative to the base models, but still outperforms both baseline heuristics, reflecting the expected tradeoff from training on a more diverse distribution.