

CoGroup: Cooperative Quality Offloading with Worker Grouping using Hierarchical Multi-Agent Deep Reinforcement Learning

Amr M. Zaki¹, Sara A. Elsayed², Khalid Elgazzar^{3,4}, Hossam S. Hassanein¹

¹School of Computing, Queen's University, Kingston, ON, Canada

²Department of Computer Science, University of Calgary, Canada

³Ontario Tech University, Oshawa, ON, Canada

⁴Canadian University Dubai, UAE

amr.zaki@queensu.ca, sara.elsayed@ucalgary.ca, khalid.elgazzar@ontariotechu.ca, hossam@cs.queensu.ca

Abstract—Task offloading in Vehicular Edge Computing (VEC) enhances cooperative perception (CP) for Autonomous Vehicles (AVs), improving traffic awareness. However, the high cost of Roadside Unit (RSU) and the underutilization of parked vehicles pose challenges. Leveraging Vehicle-to-Vehicle (V2V) communication, parked vehicles can form collaborative worker groups for efficient perception aggregation. We propose CoGroup, a two-tier framework integrating task offloading and dynamic worker grouping. Modeled as a double quadratic multiple knapsack problem, it employs Hierarchical Reinforcement Learning (HRL): QMIX for decentralized task allocation and DQN for optimized worker grouping. Experiments show that CoGroup improves traffic awareness by 21% over non-cooperative methods, reducing RSU dependence and offering a scalable, cost-effective solution for next-generation VEC systems.

Keywords: Autonomous Vehicles, Vehicular Edge Computing, Cooperative Perception, Task Offloading.

I. INTRODUCTION

Autonomous Vehicles (AVs) have the potential to improve road safety and traffic management through advanced perception and navigation systems [1]. They rely on sensors like LiDAR, cameras, radar, and GPS, but these sensors have limitations such as restricted fields of view and weather sensitivity [2]. Cooperative Perception (CP) overcomes these challenges by enabling data sharing among vehicles, improving situational awareness [3]. To meet CP demands, extensive communication among vehicles is required, which can be supported by Vehicular Edge Computing (VEC) [4], [5].

However, deploying edge servers in urban areas is costly, with Roadside Units (RSUs) priced at around \$17,680 each [6], making large-scale deployment unfeasible. This highlights the need for cost-effective alternatives that preserve the benefits of cooperative vehicular systems. Leveraging parked vehicles as Parked Roadside Units (P-RSUs) provides an affordable solution [4]. However, parked vehicles, due to their limited computational capabilities, are unable to fully accommodate the required tasks [7]. By forming Vehicle Edge Alliances (VEAs), vehicles pool resources to support overloaded edge servers [7]. Nevertheless, this introduces complexity in managing task offloading and worker cooperation, while scalability challenges necessitate efficient resource orchestration as the system grows.

This paper proposes the novel Cooperative Quality Offloading with Worker Grouping (CoGroup) which uses Hierarchical Multi-Agent Deep Reinforcement Learning to optimize offloading CP tasks to parked vehicles while efficiently forming vehicle groups to enhance resource sharing. The problem is formulated as a double quadratic multiple knapsack problem, integrating group formation [8]. We propose a two-layer hierarchical deep reinforcement learning (HRL) framework [9], where the first layer uses a multi-agent DRL model (QMIX) [10] for decentralized offloading, and the second layer employs a centralized Deep Q-Network (DQN) [11] for group formation.

Our contributions are as follows:

- CoGroup, an innovative offloading and group formation scheme that optimizes CP task distribution in VEC systems by enabling efficient worker group formation.
- Reformulation of the problem as a double quadratic multiple knapsack problem, effectively integrating offloading tasks and worker cooperation within a unified framework.
- Implementation of a hierarchical reinforcement learning (HRL) approach, utilizing QMIX for decentralized offloading and DQN for efficient group formation and coordination.

Our evaluations show that CoGroup outperforms other VEC schemes, where it improves traffic awareness by 21% over non-cooperative scheme and 12% over random worker cooperation scheme. The paper is organized as follows: Section II reviews related works. Section III details the system model. Section IV formulates the problem. Section V presents the HRL framework. Section VI provides evaluation results. Section VII concludes and discusses future research.

II. RELATED WORK

Offloading vehicular tasks to the edge is essential for AVs, which face computation-intensive and latency-sensitive demands [5], [12]–[14]. Despite advancements in edge computing, the high costs associated with deploying RSUs continue to pose significant economic and practical challenges. To address this, alternative solutions, such as utilizing idle vehicles for task offloading, have been explored [15]. However, due to the limited computational capabilities of parked vehicles,

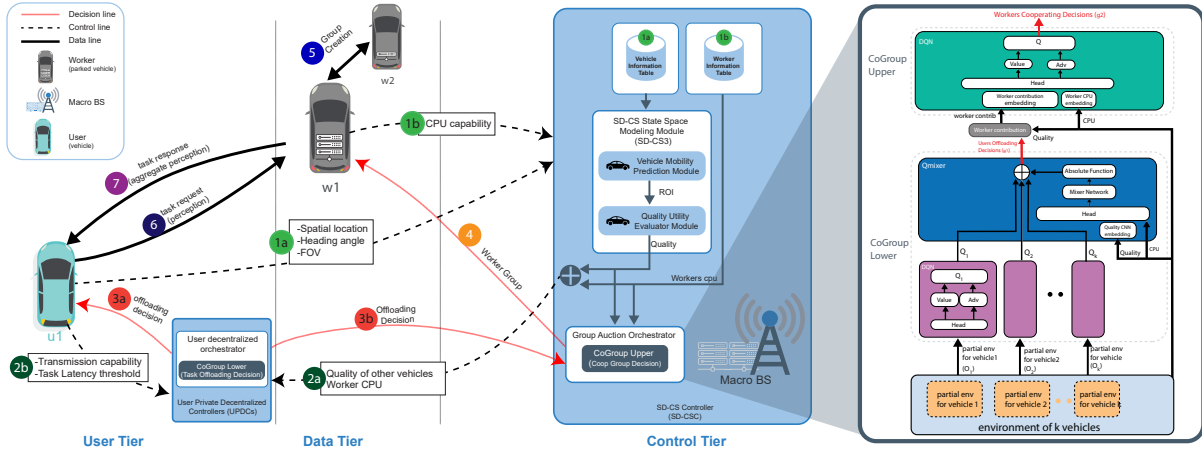


Fig. 1: CoGroup Architecture

cooperative task processing has been proposed as a means to enable dynamic resource sharing, thereby enhancing overall system efficiency [16]. The complexity of navigating solution spaces in resource allocation within VEC environments further complicates this, which is why Multi-Agent Reinforcement Learning (MARL) has been employed to efficiently explore these complex spaces [17]. Additionally, to improve scalability and decision-making efficiency in dynamic VEC networks, Hierarchical Reinforcement Learning (HRL) has been integrated, further enhancing network performance and adaptability [9]

III. SYSTEM MODEL

A. System Architecture

In the CoGroup architecture, a Macro Base Station (MBS) serves intersection vehicles (users) and coordinates parked vehicles (workers) for cooperative perception (CP). Users transmit perception data to the MBS, while parked vehicles form worker groups to process and aggregate data. The system follows a hybrid centralized-decentralized model inspired by SDN [18], enabling decentralized task offloading with centralized global computations. As shown in Fig. 1, the framework has three tiers: user, data, and control. Users offload tasks via User Private Decentralized Controllers (UPDCs), while the MBS's Cooperative Services Controller (SD-CSC) manages global computations. The process involves users and workers transmitting data (Step 1), SD-CSC evaluating task quality and worker capabilities (Step 2), and then UPDCs making offloading decisions (Step 3). The Group Orchestrator forms worker groups, assigning the highest-capacity worker within a group as leader (Steps 4,5), subsequently users offload tasks (Step 6) and receive fused perceptions (Step 7). This design enhances computational efficiency, CP quality, and user privacy.

B. System Model and Overview

Consider a set of W cooperating workers, $\mathcal{W} = \{w_1, w_2, \dots, w_W\}$, and a set of U users, $\mathcal{U} = \{u_1, u_2, \dots, u_U\}$, operating in discrete time slots $\mathcal{T} = \{1, \dots, T\}$. In each slot $t \in \mathcal{T}$, each user $u_i \in \mathcal{U}$ offloads its CP task to an assigned worker $w_j \in \mathcal{W}$. Tasks are defined by $\Psi_i = \{l_i, \lambda_i, \kappa_i\}$, where l_i is the computation workload (CPU cycles), λ_i the data size (bits), and κ_i

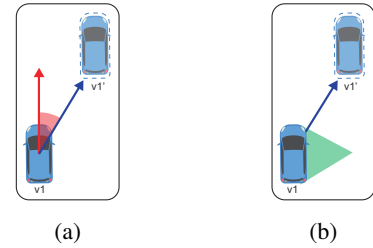


Fig. 2: RoI estimation, a) v_1 predicts the next location v_1' . b) Accordingly the RoI is estimated (green triangle).

the latency deadline (ms). Workers are characterized by $\varpi_j = \{c_j\}$, where c_j is the maximum CPU frequency (cycles/s). The overall system latency T is governed by worker cooperation, with the max function in Eq. 1 capturing the impact of the slowest component, reflecting the bottleneck in collaborative task processing.

$$T = \max(T_{\text{trans}} + T_{\text{receiveback}} + T_{\text{propagation}} + T_{\text{cpu}}) + T_{\text{agg}}, \quad (1)$$

where T_{trans} , $T_{\text{receiveback}}$, $T_{\text{propagation}}$, T_{cpu} , and T_{agg} denote transmission, result return, propagation, computation, and aggregation latencies, respectively.

The computation latency T_{cpu} , essential to worker cooperation, is given by:

$$T_{\text{cpu}(ij)} = \frac{l_i}{c_j/\eta_j}, \quad (2)$$

where l_i is the task's workload, c_j the worker's computational capacity, and η_j the number of assigned tasks. Workers share resources equally among tasks.

LiDAR-based applications, essential for CP, generate approximately 4 MB per frame [19], but CNN-based feature extraction reduces it to 200 KB, minimizing T_{trans} . Short inter-vehicle distances and 5G low-latency [20] further reduce transmission latency. $T_{\text{receiveback}}$ and $T_{\text{propagation}}$ are negligible due to small result sizes [21] and short propagation distances. Therefore, this study focuses on T_{cpu} and T_{agg} as key latency factors, with aggregation latency discussed in later sections.

C. Vehicle Mobility Prediction Module

CoGroup predicts each vehicle's next movement to estimate its Region of Interest (RoI), denoted \mathcal{Z}_i for user u_i . The SD-CS3 module uses the vehicle's current location and heading

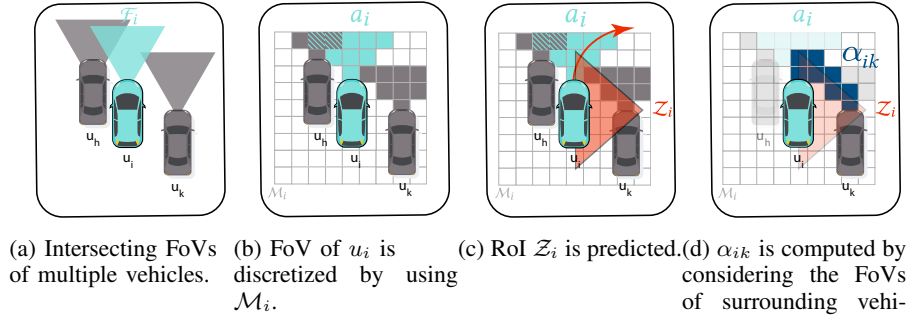


Fig. 3: Shared-Interest utility function

to predict its next turn, employing Kalman filtering (KF) [22] for accurate estimation. KF estimates the state of a linear dynamic system from noisy measurements. After predicting the vehicle's next location, as shown in Fig.2, SD-CS3 module compares it with the current heading helps determine if the vehicle will turn left or right, aiding in RoI estimation. The RoI is modeled as a triangular region aligned with the predicted turning direction.

D. Quality Utility Evaluation Module

Once the RoI is estimated for each vehicle, CoGroup evaluates traffic awareness, a_i , as shown in Fig.3, based on the non-redundant coverage of user u_i 's sensors. This is done by projecting the triangular field of view (FoV) \mathcal{F}_i onto a matrix \mathcal{M}_i , then applying rasterization to determine which cells are covered. The resulting metric, a_i , quantifies the perception area. CoGroup also incorporates shared interest, α_{ik} , to improve traffic awareness. α_{ik} is calculated by multiplying user u_i 's RoI with the perceptions of all other users whose FoVs overlap with \mathcal{Z}_i . This reduces redundancy by considering only the intersecting areas once.

E. Group Worker cooperation

CP applications require substantial computational resources, often exceeding parked vehicles capacities. To balance workloads, worker collaboration is essential [16]. CoGroup forms mutually exclusive groups G of workers. Each group's aggregator, w_{\max} , is the worker with the highest computational capacity. The aggregation latency T_{agg} , accounting for the workload consolidation at w_{\max} , is given by Eq. 3:

$$T_{\text{agg}} = \frac{l_{\text{agg}}}{c_{\text{max}}} \quad (3)$$

where l_{agg} is the aggregation workload, and c_{max} is the aggregator's computational capacity. This latency represents the additional processing burden on w_{\max} .

IV. PROBLEM FORMULATION

To tackle the task offloading group cooperation challenge, we model it as a Dual Quadratic Multiple Knapsack Problem (D-QMKP) [8]. This model incorporates two types of binary decision variables; namely x_{ij} and y_{ij} . Specifically, x_{ij} is set to 1 if user u_i is assigned to worker w_j , and 0 otherwise, while y_{ij} governs the collaboration between workers w_i and w_j . The objective function, comprises of two main components:

- 1) **Single Cooperation:** which evaluates the benefit of users u_a and u_b sharing resources within the same worker w_i ,

weighted by the quality metric α_{ab} , reflecting the quality relationship between both users, as shown in Eq. 4. Here y_{ii} signals that w_i is utilized (i.e., $y_{ii} = 1$).

$$\sigma_{\text{single}} = \sum_{i=1}^W \sum_{a=1}^U \left(\sum_{b \neq a}^U y_{ii} \cdot x_{ai} \cdot x_{bi} \cdot \alpha_{ab} \right) \quad (4)$$

- 2) **Dual Cooperation:** which accounts for the advantages of collaboration between users u_a and u_b across different workers w_i and w_j within the same group g , mediated by the cooperative relationship y_{ij} , as shown in Eq. 5

$$\sigma_{\text{Double}} = \sum_{i=1}^W \sum_{a=1}^U \left(\sum_{j \neq i}^W \sum_{b \neq a}^U y_{ij} \cdot x_{ai} \cdot x_{bj} \cdot \alpha_{ab} \right) \quad (5)$$

These components form the objective function in Eq. 6. The model constraints are as follows: Eq. 6b ensures each user u_a is assigned to at most one worker, while Eq. 6c mandates a minimum of two users per worker. Eq. 6d activates the self-cooperation variable $y_{i,i}$ if a worker has at least one assigned user. Eq. 6e limits CPU delay to meet the user offloading deadline κ , while Eq. 6f constrains worker aggregation time within ϱ . Eq. 6g enforces symmetric cooperation ($y_{ij} = y_{ji}$), and Eq. 6h ensures transitivity ($y_{ij} = 1$ and $y_{jk} = 1$ then $y_{ik} = 1$). Finally, Eq. 6i defines x_{ij} and y_{ij} as binary variables, ensuring discrete user assignments and worker cooperation decisions.

$$P1: \max_{x,y} \vartheta(x,y) = \sigma_{\text{single}} + \sigma_{\text{Double}} \quad (6a)$$

$$\sum_{i=1}^W x_{a,i} \leq 1 \quad \forall a \in U \quad \forall i \in W \quad (6b)$$

$$\sum_{a=1}^U x_{a,i} + (1 - x_{a,i}) \cdot 2 \cdot 2 \geq 2 \quad \forall i \in W \quad \forall a \in U \quad (6c)$$

$$y_{i,i} = \left(\sum_{i=1}^U x_{a,i} \geq 1 \right) \quad \forall i \in W \quad (6d)$$

$$\sum_{a=1}^U x_{a,i} \cdot T_{\text{cpu}(a,i)} \leq \kappa \quad \forall i \in W \quad (6e)$$

$$\sum_{i=1}^W y_{i,j} \cdot T_{\text{agg}(i)} \leq \varrho \quad \forall j \in W \quad (6f)$$

$$y_{i,j} = y_{j,i} \quad \forall i, j \in W, i \neq j \quad (6g)$$

$$y_{i,j} + y_{j,k} \leq 1 + y_{i,k} \quad \forall i, j, k \in W, i \neq j \neq k \quad (6h)$$

$$x_{a,i} \in \{0, 1\}, \quad y_{i,j} \in \{0, 1\} \quad \forall a \in U, \forall i, j \in W \quad (6i)$$

CoGroup must be solved periodically to adapt to the dynamic nature of VEC. However, solving P1, which integrates

a Double Quadratic Multiple Knapsack Problem (D-QMKP), poses significant computational challenges due to its NP-Hard complexity [8]. To overcome this, we propose CoGroup-Heuristic, a hierarchical multi-agent Deep Reinforcement Learning (HRL) framework.

V. COGROUP-H (COOPERATIVE QUALITY OFFLOADING GROUP COOPERATION HEURISTIC)

A. CoGroup-H Framework

The CoGroup-H scheme tackles task offloading using a two-layer HRL approach. The first layer (CoGroup-lower) employs a decentralized VDN-based QMIX [10] to optimize user offloading decisions, enhancing scalability for large user sets. The second layer (CoGroup-upper) uses a centralized DQN [11] to manage worker cooperation. To structure decision-making in CoGroup, we define the MDP formulations for both layers before introducing the HRL mechanism.

B. CoGroup Markov decision process formulation

1) CoGroup-Lower Dec-POMDP formulation

In the lower layer, CoGroup-H models the problem as a Dec-POMDP [10], where agents make decentralized decisions with limited information. The Dec-POMDP is defined as $\langle \mathcal{K}, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{G}, \mathcal{P}, \mathcal{R} \rangle$, with \mathcal{K} as the set of agents (vehicles), \mathcal{S} as the global state, \mathcal{O} as the observation space, \mathcal{A} as the action space, \mathcal{G} as the joint action space, \mathcal{P} as the state transition function, and \mathcal{R} as the reward function. Agents aim to maximize the cumulative discounted reward $R_k = \sum_{t=1}^T \gamma r_k(t)$, where $\gamma \in (0, 1]$ is the discount factor.

Global State: The global state $s(t)$ at time t as shown in Eq. 7, consists of the shared interest quality matrix $\alpha(t) = \{\alpha_{kl}(t) \mid l \in U, l \neq k\}$, capturing user collaboration potential, and worker computation capabilities $C = \{c_j \mid j \in M\}$:

$$s(t) = [\alpha(t), C] \quad (7)$$

Managing $s(t)$ is challenging due to the $U \times U$ size of $\alpha(t)$, increasing computational costs and privacy concerns. A multi-agent framework enables decentralized processing, reducing overhead and enhancing privacy.

Observation: To mitigate these challenges, the global state is decomposed as shown in Eq. 8. $\alpha(t)$ is compacted into a $U \times 1$ vector (i.e., $\alpha_k(t)$), ensuring users access only relevant data, reducing transmission overhead, and improving convergence. Worker capabilities C remain broadcasted to all users:

$$O_k(t) = [\alpha_k(t), C] \quad (8)$$

Action: The action for each agent k involves either offloading the task to a specific worker $w_j \in W$ or choosing not to offload ($a_k(t) = -1$) as defined in Eq. 9.

$$a_k(t) \in \{(j, j \in W), (-1)\} \quad (9)$$

Reward: After executing action $a_k(t)$ at time t , agent k receives a reward $r_k(t)$, assessing its effectiveness. The reward function (Eq. 10a) integrates multiple performance factors:

$$r_k(t) = \begin{cases} \omega_a & \text{if offloads alone} \\ \omega_m & \text{if cooperation is possible but not utilized} \\ \sigma_{\text{single}} & \text{if } T_{\text{cpu}(k)} \leq \kappa, \alpha_k(t) > 0 \\ \omega_d + \sigma_{\text{single}} & \text{if } T_{\text{cpu}(k)} > \kappa \end{cases} \quad (10a)$$

Where:

- ω_a : Penalty for independent offloading, missing perception aggregation.
- ω_m : Penalty for missing cooperation despite availability.
- σ_{single} : Reward for cooperative offloading within deadline.
- ω_d : Penalty for exceeding latency deadline κ .

2) CoGroup-Upper MDP formulation

In the upper layer, CoGroup-H models the problem as a centralized MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, integrating results from CoGroup-Lower (i.e., offloading decisions) with worker attributes like CPU capacity.

Global State: The global state $s(t)$, as defined in Eq. 11, consists of three components: $\alpha(t)$, the global shared interest quality matrix; $\mathcal{A}_{\text{lower}}$, the offloading actions from CoGroup-Lower; and C , the workers' CPU capacity.

$$s(t) = [\alpha(t), \mathcal{A}_{\text{lower}}, C] \quad (11)$$

However, this representation is challenging due to the dynamic interaction between lower-layer actions and the shared interest matrix $\alpha(t)$, both scaling with the number of users. To address this, we define the *worker contribution* $\phi_j(t)$, which quantifies how cooperation potential and offloading decisions influence each worker:

$$\phi_j(t) = \sum_{k \in \mathcal{K}} \mathbf{1} \left(a_k^{\text{lower}}(t) \cdot \alpha_{kj}(t) > 0 \right) \quad (12)$$

Here, $\phi_j(t)$ counts the number of users offloading to worker j while considering cooperation potential, reducing the search space complexity. With this modification, the global state $s'(t)$ becomes:

$$s'(t) = [\Phi(t), C] \quad (13)$$

where $\Phi(t) = \{\phi_j(t) \mid j \in M\}$ represents worker contributions, and C remains as defined. This compact form mitigates dimensionality issues while preserving interdependencies.

Action: In the upper layer, the action y is a $W \times W$ matrix defining worker cooperation, constrained by no self-cooperation, mutual agreement (Eq. 6g), and transitivity (Eq. 6h). Since the action space grows exponentially, we simplify it by enumerating valid cooperation patterns along with a "no cooperation" option. Agents select only from this predefined set, ensuring constraint compliance:

$$a_y = \begin{cases} \mathcal{Y}_i & \text{if } i \in \{1, 2, \dots, P\}, \\ \mathcal{Y}_{\text{no coop}} & \text{if } i = 0. \end{cases} \quad (14)$$

This structured approach significantly reduces the action space while maintaining validity.

Reward: After action y , CoGroup-upper receives a reward r_y assessing its effectiveness in problem $P1$. The reward function (Eq. 15) accounts for perception aggregation and latency penalties, where w_{total} accounts for the weight of perception, while w_{delay} accounts for the weight of exceeding the deadline:

$$r_y(t) = w_{\text{total}} \cdot \sigma_{\text{Double}} - w_{\text{delay}} \quad (15)$$

C. CoGroup Hierarchical Reinforcement Learning

1) CoGroup-Lower and CoGroup-Upper Architecture

CoGroup integrates CoGroup-Lower and CoGroup-Upper, as shown in Fig.1 for distributed scheduling and cooperative

decision-making under centralized training decentralized execution (CTDE) [10]. At the lower layer, each agent k estimates its state-action value $Q_k(o_k, a_k)$, aggregated via QMIX to compute $Q_{\text{QMIX}}(s, g)$, guiding user offloading g_1 . At the upper layer, these decisions, and CPU capacities feed into DQN to determine cooperation g_2 . After training, decentralized agents are deployed to UPDCs.

a) CoGroup-Lower (Agents)

Each agent k uses multi-layer perceptions (MLPs) to compute $Q_k(o_k, a_k; \theta_k)$, with parallel advantage $A_k(o_k, a_k; \theta_k)$ and value $V_k(o_k; \theta_k)$ streams, combined as:

$$Q_k(o_k, a_k) = V_k(o_k) + \left(A_k(o_k, a_k) - \frac{1}{|\mathcal{A}|} \sum_{a'} A_k(o_k, a') \right). \quad (16)$$

b) CoGroup-Lower (Mixing Network)

QMIX aggregates individual Q-values into $Q_{\text{QMIX}}(s, g)$, ensuring monotonic decomposition:

$$\arg \max_{\{a_k\}} Q_{\text{tot}}(s, g) = \begin{pmatrix} \arg \max_{a_1} q_{a_1} \\ \vdots \\ \arg \max_{a_k} q_{a_k} \end{pmatrix}. \quad (17)$$

Actions follow an epsilon-greedy strategy to balance exploration and convergence.

c) CoGroup-Upper

CoGroup-Upper employs DQN for high-level cooperation, processing worker contributions and CPU capacities through embedding layers to compute $Q_{\text{upper}}(s, a)$, guiding worker cooperation.

2) CoGroup Joint Training

The joint training process, minimizes loss functions $L_{\text{lower}}(\theta)$ and $L_{\text{upper}}(\theta)$, using experience replay for stability.

a) Loss Functions:

For CoGroup-Lower and CoGroup-Upper:

$$L_{\text{lower}}(\theta) = \frac{1}{N_b} \sum_j \sum_t (y_{\text{tot},j}(t) - Q_{\text{QMIX}}(s, g; \theta))^2. \quad (18a)$$

$$L_{\text{upper}}(\theta) = \frac{1}{N_b} \sum_j (y_j(t) - Q_{\text{upper}}(s_j, a_j; \theta))^2. \quad (18b)$$

b) Target Values:

For CoGroup-Lower and CoGroup-Upper:

$$y_{\text{tot},j}(t) = r_j(t) + \gamma \max_{\mathbf{g}_{t+1}} Q_{\text{QMIX}}(s, g; \theta). \quad (19a)$$

$$y_j(t) = r_j(t) + \gamma \max_{a'} Q_{\text{upper}}(s_j, a'; \theta^-). \quad (19b)$$

c) Parameter Updates:

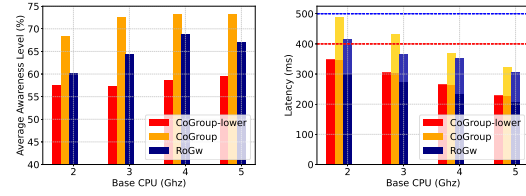
Network parameters are updated using gradient descent:

$$\theta \leftarrow \theta - \text{lr} \cdot \frac{\partial L}{\partial \theta}. \quad (20)$$

This architecture balances centralized guidance with decentralized decision-making for efficient resource allocation and cooperation.

VI. PERFORMANCE EVALUATION

We evaluate the performance of CoGroup by comparing it against multiple baselines. The evaluation is structured as follows: we first provide an overview of the comparison algorithms, followed by a description of the performance metrics. Next, we present the simulation setup and parameter settings. Finally, we discuss the experimental results.



(a) Average Awareness (b) Average Latency
Fig. 4: Performance results of CoGroup-lower, CoGroup-W/A, CoGroup and RoGw over varying computation capability

A. Baselines

Several existing task offloading algorithms are used as baselines. CoGroup-Lower evaluates the lower layer of CoGroup, which performs task offloading without involving worker cooperation, using a decentralized QMIX algorithm based on the baseline architectures in [5]. Random-Offloading-Greedy-Worker-Cooperation (RoGw) combines greedy task offloading with random worker cooperation to assess CoGroup's effectiveness in cooperative offloading.

B. Performance Metrics

We focus on two key performance indicators: Average awareness level, measured as the ROI coverage from user collaboration with the same worker and the vehicle's own perception, and the response delay, measured from task offloading to worker cooperation aggregation until result return.

C. Simulation Setup

We implement the schemes using Python and PyTorch. Vehicular mobility traces are generated using the SUMO simulator [23]. Simulations occur in a 200×200 intersection with four lanes, where vehicles travel at a maximum speed of 40 km/h. Four parked workers have computational resources randomly selected from $\{c + 1\delta, c + 2\delta\}$. The number of users is fixed at 20. The computational intensity for tasks is 3×10^8 cycles [5], with 2×10^8 cycles allocated for worker cooperation. The ROI is defined as a 10m-high equilateral triangle in the vehicle's turn direction. Perception latency is set to 0.4 seconds, and worker cooperation has a 0.1-second deadline [24]. The simulation runs for 10 seconds, with updates occurring every second.

D. Model Architecture

The CoGroup-Lower agent: architecture consists of shallow online and target networks, featuring a linear head module with ReLU activation for state processing and advantage/value streams. Parameters are initialized using Kalman normal initialization for weights and zeros for biases. The Adam optimizer is used for training updates. **CoGroup-Lower (QMIX):** The QMIXer processes the shared-interest matrix using a CNN embedding. Worker CPU embeddings and other projections are passed to a two-layer mixing network, producing weights for Q-values, which are combined into the joint Q-value. **CoGroup-Upper:** A DQN-based model where worker contribution, CPU, and budget embeddings are processed through distinct layers before merging into a linear head layer. The merged embeddings feed into advantage and value streams to produce the final output. Training is conducted over 40,000 iterations.

E. Simulation Results and Analysis

This experiment, illustrated in Fig. 4, examines the effect of varying computational capacity C (from 2.0 to 5.0 GHz) on different offloading schemes, with other parameters held constant. The results in Fig. 4a show that increasing C significantly improves traffic awareness (i.e., coverage of the RoI) by enabling more offloading tasks and increasing cooperating workers, due to enhanced computational resources. Worker cooperation schemes of CoGroup consistently outperform non-cooperative ones, with traffic awareness improvements of 21% and 12%, compared to CoGroup-lower and RoGw respectively, underscoring the crucial role of the worker cooperation step in optimizing performance. Figs. 4b shows the total system latency (dark shade: CPU latency; light shade: aggregation latency), with the lower (400ms, red) and upper (100ms, blue) deadlines indicated. Latency decreases across all schemes as computational resources of parked vehicles increase, while additional latency from worker cooperation steps is also highlighted. In Fig. 4b, all schemes meet the 400ms deadline for lower-level task offloading, while all worker cooperating schemes meet the added 100ms deadline. In Fig. 4b, CoGroup incurs a higher latency due to worker cooperation but remains within the deadline, ensuring timely task execution. This demonstrates its ability to effectively balance the benefits of cooperation with stringent latency constraints.

VII. CONCLUSION

This paper introduces the Cooperative Quality Offloading with Worker Grouping (CoGroup) framework, using Hierarchical Multi-Agent Deep Reinforcement Learning (HRL) to address task offloading and worker cooperation in Vehicular Edge Computing (VEC) systems. We model the problem as a double quadratic multiple knapsack and solve it with a two-level HRL approach. In the lower layer (CoGroup-Lower), QMIX enables decentralized task offloading, ensuring privacy and efficiency. At the upper layer, a Deep Q-Network (DQN) optimizes the state and action spaces, enabling scalable worker cooperation. Our results show that CoGroup improves traffic awareness by 21% over non-cooperative scheme and 12% over random worker cooperation scheme. In conclusion, CoGroup offers a scalable and real-time solution for task management in next-generation 5G and beyond VEC systems.

ACKNOWLEDGMENT

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number ALLRP 549919-20, and a grant from Distributive, ltd.

REFERENCES

- [1] K. Edward. (2023, Jul) Goodbye gridlock: How autonomous vehicles can revolutionize city living. Accessed: Nov 19, 2024. [Online]. Available: <https://www.forbes.com/sites/kyleedward/2023/07/29/goodbyegridlock-how-autonomous-vehicles-can-revolutionize-city-living/>
- [2] A. Mahmoud, A. Nouredin, and H. S. Hassanein, "Integrated positioning for connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 397–409, 2020.
- [3] J. Guo, D. Carrillo, Q. Chen, Q. Yang, S. Fu, H. Lu, and R. Guo, "Slim-fcp: Lightweight-feature-based cooperative perception for connected automated vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 630–15 638, 2022.
- [4] Y. Chen, J. Wu, J. Han, H. Zhao, and S. Deng, "A game-theoretic approach-based task offloading and resource pricing method for idle vehicle devices assisted vec," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 21 954–21 969, 2024.
- [5] A. M. Zaki, S. A. Elsayed, K. Elgazzar, and H. S. Hassanein, "Quality-aware task offloading for cooperative perception in vehicular edge computing," in *arXiv 2405.20587*, 2024.
- [6] A. B. Reis, S. Sargento, and O. K. Tonguz, "Parked cars are excellent roadside units," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2490–2502, Sep. 2017.
- [7] P. Qin, Y. Fu, X. Feng, X. Zhao, S. Wang, and Z. Zhou, "Energy-efficient resource allocation for parked-cars-based cellular-v2v heterogeneous networks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 3046–3061, 2022.
- [8] M. Aider, O. Gacem, and M. Hifi, "Branch and solve strategies-based algorithm for the quadratic multiple knapsack problem," *Journal of the Operational Research Society*, vol. 71, 12 2020.
- [9] Y.-C. Wu, C. Lin, and T. Q. S. Quek, "A robust distributed hierarchical online learning approach for dynamic mec networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 641–656, 2022.
- [10] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.
- [11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
- [12] J. Yan, X. Zhao, and Z. Li, "Deep-reinforcement-learning-based computation offloading in uav-assisted vehicular edge computing networks," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 882–19 897, 2024.
- [13] A. M. Zaki, S. A. Elsayed, K. Elgazzar, and H. S. Hassanein, "Plto: Path loss-aware task offloading for vehicular cooperative perception," in *7TH IEEE INTERNATIONAL CONFERENCE ON FOG AND EDGE COMPUTING*, 2023.
- [14] A. M. Zaki, S. A. Elsayed, K. Elgazzar, and H. S. Hassanein, "Multi-vehicle task offloading for cooperative perception in vehicular edge computing," in *IEEE International Conference on Communications*, 2023.
- [15] B. Lu, B. Fan, Y. Wu, L. Qian, H. Zhang, and R. Lu, "Predictive computation offloading and resource allocation in dt-empowered vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 6, pp. 5474–5487, 2024.
- [16] Y. Hou, Z. Wei, R. Zhang, X. Cheng, and L. Yang, "Hierarchical task offloading for vehicular fog computing based on multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 23, no. 4, pp. 3074–3085, 2024.
- [17] K. Qu, W. Zhuang, Q. Ye, W. Wu, and X. Shen, "Model-assisted learning for adaptive cooperative perception of connected autonomous vehicles," *IEEE Transactions on Wireless Communications*, vol. 23, no. 8, pp. 8820–8835, 2024.
- [18] Y. Li, Q. Zhang, H. Yao, R. Gao, X. Xin, and F. R. Yu, "Stigmergy and hierarchical learning for routing optimization in multi-domain collaborative satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1188–1203, 2024.
- [19] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," 11 2019, pp. 88–100.
- [20] T. Qin, J. Hou, P. Yang, X. Cao, and Y. Wu, "Edge-assisted low-latency object detection for networked vehicles using point cloud," in *2024 International Conference on Cloud and Network Computing (ICNC)*, 2024, pp. 49–56.
- [21] E. Krijestorac, A. Memedi, T. Higuchi, S. Ucar, O. Altintas, and D. Cabric, "Hybrid vehicular and cloud distributed computing: A case for cooperative perception," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.
- [22] E. Maggio and A. Cavallaro, "Video tracking: Theory and practice," *Video Tracking: Theory and Practice*, 12 2010.
- [23] M. Behrisch, L. Bieker-Walz, J. Erdmann, and D. Krajzewicz, "Sumo – simulation of urban mobility: An overview," vol. 2011, 10 2011.
- [24] J. Tan, R. Khalili, and H. Karl, "Multi-objective optimization using adaptive distributed reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 10 777–10 789, 2024.