DNN Partition and Offloading Strategy with Improved Particle Swarm Genetic Algorithm in VEC

Chunlin Li, Long Chai, Kun Jiang, Yong Zhang, Jun Liu, Shaohua Wan,

***Abstract*—Vehicular edge computing (VEC) is a novel computing paradigm, which is designed to satisfy the growing computation and communication needs of vehicle systems. With the assistance of VEC, vehicles can execute artificial intelligence (AI) tasks based on deep neural network (DNN), which are compute-intensive and delay-sensitive. However, it is difficult to deploy large-scale and compute-intensive DNN on resource-constrained terminal devices. Therefore, DNN model partition and offloading strategy have received a lot of attention, however, most of the researches have not taken into account the problem that the optimal partition point of a DNN model changes with the allocated computing resources. To address this problem, we propose a computing offloading strategy based on DNN model partition. This strategy selects the optimal DNN model partition points based on the computing capability of the vehicle, and then develops the optimal task offloading strategy to realize the effective distribution and execution of tasks between the edge server and the service vehicle. To minimize the task offloading delay, we propose an improved particle swarm genetic algorithm (IPSGA) to achieve the optimal offloading strategy. The algorithm uses the variable acceleration coefficient with the number of iterations and the inertia weight with the success rate as the feedback parameters to improve the particle swarm optimization algorithm (PSO), and the genetic algorithm (GA) is improved with the adaptive crossover probability and the adaptive mutation probability. Experimental results show that compared to the baselines, the IPSGA can reduce the overall system delay and increase the task completion rate.**

***Index Terms*—VEC, computing offloading, DNN model, improved particle swarm genetic algorithm, genetic algorithm**

I. INTRODUCTION

IN the VEC, the computational offloading strategy is mainly aimed at compute-intensive tasks. By offloading compute-intensive tasks to the edge server, the local computing pressure of the vehicle is reduced. In intelligent connected vehicle (ICV) navigation scenario, the tasks that intelligent vehicles need to handle are not only compute-intensive, but also very sensitive to delay, such as road signal and traffic sign recognition [1]. These tasks involve processing the information gathered by the ICV using DNN models and delivering the processed data to the driver to enhance the overall driving experience.

A. Related Work

In the VEC, it is essential for devices with constrained resources to collaborate with edge servers in order to improve processing efficiency for compute-intensive tasks. The aim of this collaboration is to minimize time and energy consumption while improving service quality. The Ref. [2] proposed a collaborative computational offloading and resource allocation optimization strategy. The Ref. [3] explored the system architecture, communication, and resource allocation in 6G autonomous intelligent transportation systems (6G-AITS), along with key features and three applications. The Ref. [4] proposed a model-free reinforcement learning for a long-term offloading strategy. The Ref. [5] proposed a strategy for computational offloading and service migration based on collaborative efforts among multiple vehicles. This strategy took into account the mobility of multiple vehicles. In order to enhanced system performance, an optimization framework for computational offloading decisions was established. By adjusting collaborative computational offloading decisions among vehicles, the strategy aimed to reduce the latency associated with cooperative computational offloading, ultimately improving the success rate of task completion. The Ref. [6] proposed a distributed computing offloading solution based on Mobile Edge Computing (MEC) with the aim of reducing task processing latency. The approach employed a Lagrangian multiplier optimization algorithm to minimize the execution cost of tasks. The Ref. [7] proposed an improved multi-objective particle swarm optimization method based on queue. The Ref. [8] proposed an edge cooperative caching strategy based on Federated Deep Reinforcement Learning (F-DRL). The Ref. [9] proposed a mixed integer nonlinear optimization problem (MINLP) problem and attempted to solve it using non-polynomial-time algorithms. The Ref. [10] and [11] proposed an autonomous computational offloading framework and a hybrid computational offloading model to further improve resource utilization. The Ref. [12] proposed a digital twin (DTs) technology into the maritime transportation system based on the Internet of Things (IoT). The Ref. [13] proposed an end-to-end automotive edge network system, FAIR, designed to offer rapid, scalable, and equitable connectivity services for intelligent vehicles equipped with edge computing. Additionally, a business-adaptive algorithm had been devised to dynamically adjust the configuration

The work was supported by the National Natural Science Foundation of China (NSFC) under grants (No.62372344, No.62171330), Key Research and Development Plan of Hubei Province (2023BAB075), National Key R&D Program of China (Grant No. 2023YFB3308701).

Chunlin Li, Long Chai, Kun Jiang, Yong Zhang, Jun Liu, are with School of Computer and Artificial Intelligence, Wuhan University of Technology, Wuhan 430063, China(e-mail: chunlinli2020@163.com).

Shaohua Wan, Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China(e-mail: shaohua.wan@uestc.edu.cn).

of environmental perception and frame resolution based on retained service cycles and user preferences.

In the VEC, diverse strategies have been proposed to tackle complex tasks. The efficient offloading strategy for DNN Inference Acceleration (EosDNN) in the Ref. [14] enhanced DNN inference acceleration. The Ref. [15] proposed the Dynamic Frame Offloading Algorithm based on Double Deep Q-Network (DFO-DDQN), excelling in identifying optimal offloading decisions for sequential subtasks. Considering both delay and energy consumption, the Ref. [16] proposed a computational offloading strategy. In urban VEC, the Ref. [17] proposed the Mobility-aware Dependent Task Offloading Strategy (MESON), incorporating a Deep Reinforcement Learning (DRL)-based algorithm. The Ref. [18] proposed an improved Double Dueling Priority Deep Q Network (DDPQN) algorithm. This algorithm, while coordinating the relationship between "latency-energy-cost" and ensuring the rational allocation of computing resources, achieves an efficient strategy for offloading DNNs. The goal is to attain low latency, low energy consumption, and low cost. The Ref. [19] proposed an efficient approximation algorithm that successfully addresses the dual-objective optimization problem proposed in the paper and achieves the algorithm's efficient convergence. The Ref. [20] proposed a model and proposed a DRL-based cost minimization strategy to tackle workload offloading decision problems. The Ref. [21] proposed the Parametrized Dueling Deep Q-Network and Linear Programming (PDDQNLP) algorithm, specifically designed to enhance offloading fairness and energy efficiency in unmanned aerial vehicles (UAVs). Additionally, The Ref. [22] formulated task scheduling as a Mixed-Integer Linear Programming (MILP) problem involving devices, edges, and clouds to minimize the total system delay. To address this MILP problem, the authors propose an Enhanced Healing Genetic Algorithm (EHGA). These diverse approaches collectively contribute to the evolving landscape of VEC, offering solutions to various facets of computational offloading challenges. The Ref. [23] proposed a Vehicle Task Scheduling Policy Optimization (VT-SPO) algorithm based on state-of-the-art policy-driven DRL. The algorithm successfully addresses the long-term scheduling policy optimization problem proposed in the paper. The Ref. [24] proposed a computational offloading (PSOCO) algorithm based on improved Particle Swarm Optimization to solve the offloading decision problem in multi-objective scenarios. The Ref. [25] proposed a suboptimal algorithm named Hierarchical Genetic Particle Swarm Optimization (HGPCA), which combines the strengths of GA and PSO to enhance performance and convergence speed. The Ref. [26] proposed an efficient offloading strategy based on an Adaptive Genetic Algorithm Particle Swarm Optimization. This strategy effectively reducing the coding dimensions and improving the algorithm's execution time. The Ref. [27] discussed the partitioning of each task into two segments, with simultaneous execution at both local and server levels, and introduced a DQN for problem resolution. The Ref. [28] proposed a multi MEC cooperative strategy for vehicle computational offloading, termed MCVCO. This strategy aimed to reduce end-to-end latency, enhanced task offloading efficiency, and improved upload quality. The Ref. [29] proposed an integrated approach that combines hybrid caching and task offloading to minimize the overall task completion delay for vehicle users within the coverage area. The optimization strategy encompasses the selection of caching data in UAV and facilitating task offloading for vehicle users through a DQN-based algorithm.

In recent years, researches on DNN partition and offloading in VEC have gained a lot of attention, and offered various solutions from different perspectives [2] [15] [27] [28] [29]. Despite significant progress in this field, there are still many challenges that need to be addressed. Firstly, the differences between DNN recognition tasks and general tasks should be emphasized [4] [8]. In details, DNN recognition tasks typically involve the processing of massive images and videos, which need to be transmitted to cloud servers or edge nodes for processing. It will lead to higher processing costs and transmission delays [17] [30]. Secondly, complex DNN models have more parameters and deeper structures, which make features more abstract and difficult to interpret [31] [32]. Finally, how to combine the partition and compression strategies of DNN models to assist in computation offloading while reducing the completion time of DNN tasks is an problem that cannot be ignored [20] [21].

To address the above challenges, this paper considers the service vehicle (SV) as a feasible offloading target. When the computing resources of the edge server are tight, the SV will be a candidate for offloading targets. In addition, this paper proposes a computational offloading strategy based on the DNN model partition. By using DNN model partition technology and intermediate data compression to reduce intermediate data transmission delay, the processing efficiency of the DNN recognition task is improved. Furthermore, by using the improved particle swarm genetic algorithm (IPSGA), we can make optimal DNN model partition and offloading decisions for different DNN recognition tasks. The key contributions of this paper are as follows.

- To optimize the execution delay of DNN tasks, we propose a computational offloading strategy based on DNN model partition. The goal of this strategy is to reduce the execution delay of DNN tasks by using DNN model partition, and to reduce the delay of intermediate data transmission by using intermediate data compression.
- This paper proposes the IPSGA to optimize computational offloading, merging the advantages of PSO and GA. By using a selective operation approach, which enhances the exploration of the search space and particle population generation. In addition, the performance and adaptability of the algorithm are improved by adjusting the parameters in the PSO and GA.

The remainder of this paper is structured as follows. Section II proposes the system model. Section III provides a detailed description of the proposed algorithm. Experimental results are presented in Section IV. Finally, Section V concludes this paper and outlines future work.

TABLE I: SUMMARY OF PARAMETERR

Symbol	Description
r_{rsu}	Radius of coverage
P_{rsu}	Curbside unit locations
h_{rsu}	Height of curb unit
FLS_{rsu}	Total floating point computing capacity of RSU
$T^m_{tk_i}$	Maximum delay allowed for DNN task tk_i
$d_{V_i\_SV_j}$	Distance between UV V_i and SV SV_j
$d_{V_i\_rsu}$	Distance between user vehicle V_i and RSU
$R_{V_i\_SV_j}$	Maximum achievable uplink transmission rate between V_i and SV_j
Pr_{V_i}	Transmitting power of user vehicle V_i
$d^{-\theta}_{V_i\_SV_j}$	Path loss between vehicles
θ	Coefficient of route loss between vehicles
h	Uplink channel Riley decay
N_0	Gaussian white noise power
$W_{V_i\_rsu}$	Channel bandwidth between vehicle V_i and RSU
FLS_{la}	Denotes the floating point computing of the DNN recognition task tk_i at layer la
FLS_{V_i}	Floating-point computing performance of local user vehicle V_i
FLS_{conv}	Convolutional layer floating-point operations
FLS_{max}	Floating point calculation amount of pooling layer

II. SYSTEM MODEL

We can express all vehicles as $V = \{V_1, V_2, \ldots, V_n\}$, where the attributes of V_i include the DNN task, moving speed, floating-point computing performance, transmitting power, and initial position, which can be expressed as V_i=$\{tk_i, V_{V_i}, FLS_{V_i}, Pr_{V_i}, P_{V_i}\}$, the tk_i is expressed as $tk_i = \{1, 2, \ldots, l\}$. The data output by each layer of the tk_i can be expressed as $Data_{tk_i} = \{d_1, d_2, \ldots, d_l\}$, and the maximum delay allowed by the DNN task tk_i is $T^m_{tk_i}$. Each SV may process multiple DNN tasks simultaneously, we can express all SVs as $SV = \{SV_1, SV_2, \ldots, SV_m\}$. When the SV receives a DNN task tk_i , it will allocate a part of its maximum computing resource. We assume that the SV as the offloading target can provide sufficient computing resources for user tasks [33]. The moving speed, floating-point capability, and initial position of SV_j are expressed as SV_j=$\{V_{sv_j}, FLS_{SV_j}, P_{SV_j}\}$. Furthermore, for the sake of analysis, we consider the system to be quasi-static, with the wireless channels remaining unchanged during task processing [34]. The computational offloading system model is shown in Fig.1.

A. Communication Model

In a time slot t, the distance between the V_i and the SV_j is $d_{V_i\_SV_j}$, and the distance between V_i and RSU is $d_{V_i\_rsu}$, which can be expressed as formulas (1)(2).

$$d_{V_i\_SV_j} = |(P_{V_i} + v_{V_i} \times t) - (P_{SV_j} + v_{SV_j} \times t)|, \quad (1)$$

$$d_{V_i\_rsu} = \sqrt{|P_{rsu} - P_{V_i} + v_{V_i} \times t)|^2 + h^2_{rsu}}. \quad (2)$$

In the VEC, we consider V2V and V2I communication models, respectively. Since the time returned by the DNN task result is much shorter than the time required for inference and intermediate data transmission, we do not consider the result return time and do not involve the downlink. In the V2V communication model, vehicles communicate with each other using the standard IEEE 802.11p for Dedicated Short-Range Communication (DSRC) [35]. Communication between vehicles, SVs, and Roadside Units (RSUs) is implemented using Frequency Division Multiple Access (FDMA) to better isolate signals between neighboring cells. We assume that each RSU and each vehicle and SV are allocated non-overlapping frequency spectra, ensuring the spectral separation between vehicle and SVs. Therefore, this paper primarily focuses on inter-cell interference among vehicles and inter-cell interference among SVs [36].

This paper assumes that the uplink is a Rayleigh channel. According to Shannon's theorem, the maximum uplink transmission rate $R_{V_i\_SV_j}$ between the V_i and the SV_j can be expressed as formula (3).

$$R_{V\_SV_j} = W_{V_i\_SV_j} \log_2 \left(1 + \frac{Pr_{V_i} d^{-\theta}_{V_i\_SV_j} h^2}{N_0}\right), \quad (3)$$

We assume that the wireless channel between the user vehicle and the RSU also experiences flat fading and follows the Rayleigh distribution [37]. The maximum achievable uplink transmission rate $R_{V_i\_rsu}$ between the user vehicle V_i and the RSU can be expressed as formula (4).

$$R_{V\_rsu} = W_{V_i\_SV_j} \log_2 \left(1 + \frac{Pr_{V_i} d^{-\theta}_{V_i\_SV_j} h^2}{N_0}\right), \quad (4)$$

where $W_{V_i\_rsu}$ represents the channel bandwidth between V_i and the RSU, and vehicles connected to the same edge server compete fairly for the channel bandwidth of the edge server. Therefore, the channel bandwidth between the V_i and the RSU is determined by dividing the total channel bandwidth of the RSU by the number of vehicles connected to it, and $d_{V_i\_rsu}$ is the distance between the vehicle and the RSU. It is necessary to satisfy $d_{V_i\_rsu} \leq r_{rsu}$, that is, the V_i is within the RSU coverage.

B. Intermediate Data Compression

We consider the use of quantization to compress the intermediate feature data by mapping the floating point values in the intermediate feature data to a smaller set of discrete integers. The process of vehicle quantization can be expressed as formula (5).

$$y_i = round\left(\frac{(2^{c_q} - 1)[x_i - \min(x)]}{\max(x) - \min(x)}\right), \quad (5)$$

where x represents the intermediate feature data to be quantized, x_i is the ith data point in x, y_i is the integer obtained from the quantization of x_i, and c_q represents the bit width used for quantization. The maximum and minimum values of x_i can be determined using pre-collected characteristic data, and $round()$ represents rounding to the nearest integer.

The intermediate data compression process can be expressed as formula (6).

$$x_i{}' = \frac{y_i[\max(x) - \min(x)]}{2^{c_q} - 1} + \min(x), \quad (6)$$

where $x_i{}'$ represents the recovered value.The rounding operation performed during the quantization process introduces a

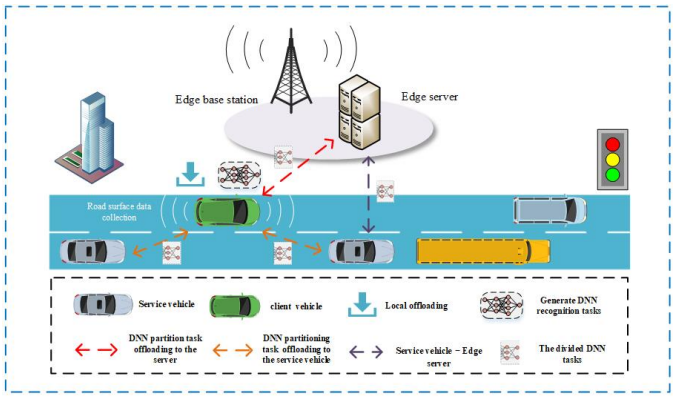


Fig. 1: System model of the computational offloading system based on DNN model partition.

rounding error, so $x_i{}'$ is usually not exactly equal to x_i. Since the original intermediate features are represented by 32-bit wide floating point numbers, the compression rate (CR) of the quantization process can be expressed as formula (7).

$$CR = \frac{c_q}{32}. \tag{7}$$

C. DNN recognition task partition and offloading model

In the previous discussion, each vehicle generates only one DNN task $tk_i = \{1, 2, \ldots, l\}$ within a time slot. Due to the limited computing capacity of vehicles, the DNN task is divided into two parts. The first part is used for local computing, and then the locally generated data is transmitted to the RSU or SV. This represents that the second part will be processed by the server or a SV, taking advantage of its computing capabilities. The DNN recognition task consists of l layers. We assume that these layers are relatively independent, that is, to process one layer of tasks, only the output of the previous layer of tasks is required. Therefore, we consider the task at each level as a subtask. We assume that the mth task is selected as the partition point, so the tasks before the mth task will be offloaded to the vehicle, and the tasks after the mth task will be offloaded to the RSU.

D. computational Model

The Ref. [38] presents the relationship between the computing time of the convolutional layer and the pooling layer and the floating point operations ($FLOPS$), and subsequently introduces a formula for calculating $FLOPS$ for both the convolutional layer and the pooling layer. The calculation of FLS_{conv} can be expressed as formula (8).

$$FLS_{conv} = 2 \cdot H \cdot W \left(C_{in} K^2 + 1\right) C_{out}, \tag{8}$$

where H represents the height of the input feature map to the convolutional layer, while W represents its width, C_{in}, C_{out} represents the number of input channels and output channels for the convolution calculation, respectively. K represents the size of the convolution kernel. FLS_{max} can be expressed as formula (9).

$$FLS_{max} = H \cdot W \cdot C_{in}. \tag{9}$$

For the DNN task $tk_i = \{1, 2, \ldots, l\}$ of the V_i, the RSU selects the partition point l_i, where the initial l_i layers of the DNN task are processed locally, while the subsequent layers from $l_i + 1$ to l are processed in the SV or edge server. The execution time $T_{tk_i}^{local}$ of tk_i on the V_i can be expressed as formula (10).

$$T_{tk_i}^{local} = \sum_{la=1}^{l_i} \frac{FLS_{la}}{FLS_{V_i}}, \tag{10}$$

where FLS_{la} represents the floating-point computing capability of layer la in tk_i, which can be expressed as formula (11), and FLS_{V_i} represents the floating point capability of V_i.

$$FLS_{la} = \begin{cases} FLS_{la}, & la_{max} \\ FLS_{conv}, \text{la is a convolution layer} \end{cases}. \tag{11}$$

The time $T_{tk_i}^{SV_j}$ required for processing the DNN task on the SV_j after partition can be expressed as formula (12), where FLS_{SV_j} represents the floating-point computing capability of the SV_j.

$$T_{tk_i}^{SV_j} = \sum_{la=l_j}^{l} \frac{FLS_{la}}{FLS_{SV_j}}, \tag{12}$$

The time $T_{tk_i}^{rsu}$ of DNN tk_i processed on the edge server after partition can be expressed as formula (13).

$$T_{tk_i}^{rsu} = \sum_{la=l_i}^{l} \frac{FLS_{la}}{fl_{tk_i}^{rsu}},$$
$$s.t. \sum_{i=1}^{n} fl_{tk_i}^{rsu} \leq FLS_{rsu}. \tag{13}$$

where FLS_{rsu} represents the floating-point computing capacity of the overall computing resources within the edge server, while $fl_{tk_i}^{rsu}$ represents the floating-point computing capacity allocated by the edge server to task tk_i.

When the DNN task partition point is at layer l_i, $T_{tk_i}^{V_i\_tran\_SV_j}$ and $T_{tk_i}^{V_i\_tran\_rsu}$ represent the time required for intermediate data transmission to the SV or the edge server, respectively.

$$T_{tk_i}^{V_i\_tran\_SV_j} = \frac{d_{l_i} \cdot CR}{R_{V_i\_SV_j}}, \tag{14}$$

$$T_{tk_i}^{V_i\_tran\_rsu} = \frac{d_{l_i} \cdot CR}{R_{V_i\_rsu}}, \tag{15}$$

When the DNN task of the V_i is offloaded to local, the required task execution time is expressed as $T_{V_i}^{local}$, as shown in formula (16), which includes only the execution time on the local.

$$T_{V_i}^{local} = T_{tk_i}^{local}. \tag{16}$$

The task execution time required when the DNN task of the user vehicle V_i is selected to be executed on the edge server can be expressed as $T_{V_i}^{MEC}$, as shown in formula (17). This includes the local execution time $T_{tk_i}^{local}$, the intermediate data transmission time between the user vehicle and the edge server

$T_{tk_i}^{V_i\_tran\_rsu}$ and the execution time of the DNN task on the edge server $T_{tk_i}^{rsu}$.

$$T_{V_i}^{MEC} = T_{tk_i}^{local} + T_{tk_i}^{V_i\_tran\_rsu} + T_{tk_i}^{rsu}, \tag{17}$$

The task execution time required when the DNN task of the V_i is selected to execute on the SV_j can be expressed as $T_{V_i}^{SV_j}$, as shown in formula (19), including the local execution time $T_{tk_i}^{local}$, the intermediate data transmission time $T_{tk_i}^{V_i\_tran\_SV_j}$ between the user vehicle and the SV, and the execution time of the DNN task on the SV $T_{tk_i}^{SV_j}$.

$$T_{V_i}^{SV_j} = T_{tk_i}^{local} + T_{tk_i}^{V_i\_tran\_SV_j} + T_{tk_i}^{SV_j}. \tag{18}$$

where $D_{V_i}^{local}$ represents the local offloading decision executed by user vehicle V_i, where $D_{V_i}^{local} = 1$ indicates that user vehicle V_i executes the DNN task locally. $D_{V_i,rsu}^{rsu}$ is defined as the offloading decision from the user vehicle V_i to the RSU. $D_{V_i,rsu}^{rsu} = 1$ means that the partitioned DNN task is offloaded to the RSU. Define D_{V_i,SV_j}^{rsu} as the offloading decision from user vehicle V_i to SV_j, where $D_{V_i,SV_j}^{SV} = 1$ indicates that user vehicle V_i offloads the partitioned DNN task to SV_j. Since the partitioned DNN task can only be offloaded to one service node, the three offloading decisions $D_{V_i}^{local}$, $D_{V_i,rsu}^{rsu}$, D_{V_i,SV_j}^{SV} satisfy the following constraints.

$$D_{V_i}^{local} + D_{V_i,rsu}^{rsu} + \sum_{j=1}^{m} D_{V_i,SV_j}^{SV} = 1,$$

$$D_{V_i}^{local}, D_{V_i,rsu}^{rsu}, D_{V_i,SV_j}^{SV} \in 0, 1. \tag{19}$$

In summary, the total time consumption T_{tk_i} of the DNN task tk_i of the V_i can be expressed as formula (20).

$$\begin{aligned} Ttk_i = D_{V_i}^{local} \cdot T_{V_i}^{local} + D_{V_i,rsu}^{rsu} \cdot \\ T_{V_i}^{MEC} + \sum_{SV_j \in SV} D_{V_i,SV_j}^{SV} \cdot T_{V_i}^{SV_j}, \\ s.t. T_{tk_i} \leq T_{tk_i}^{m}. \end{aligned} \tag{20}$$

III. ALGORITHM DESCRIPTION

In this section, we modeled the offloading of recognition tasks in the VEC. The goal is to minimize the total execution time of all DNN tasks in each single time-slice. The objective function can be expressed as formula(21).

$$\begin{aligned} &\min_{D_{V_i}^{local}, D_{V_i,rsu}^{rsu}, D_{V_i,SV_j}^{SV}, l_i, fl_{tk_i}} \sum_{i=1}^{n} T_{tk_i} \\ &s.t. C_1 : D_{V_i}^{local}, D_{V_i,rsu}^{rsu}, D_{V_i,SV_j}^{SV} \in \{0, 1\}, \\ &C_2 : D_{V_i}^{local} + D_{V_i,rsu}^{rsu} + \sum_{j=i}^{m} D_{V_i,SV_j}^{SV} = 1, \\ &C_3 : T_{tk_i} \leq T_{tk_i}^{m}, \\ &C_4 : 0 \leq l_i \leq l, \\ &C_5 : \sum_{i=1}^{n} fl_{tk_i}^{rsu} \leq FLS_{rsu}. \end{aligned} \tag{21}$$

where constraints C_1 and C_2 ensure that each user vehicle's DNN task is offloaded to only one service node, either a RSU or a SV. Constraint C_3 guarantees that the total execution time of each task after offloading does not exceed the maximum allowable delay for that task. Constraint C_4 represents that the partition point l_i for each DNN task can range from layers $0 \sim l$, where l_i=0 means the task is completely offloaded to the service node and not executed on the local, l_i=l represents that all tasks are executed locally. Constraints C_5 represent that the total computing resources allocated to all tasks should not surpass the maximum computing resources available.

A. Problem Description and Algorithm Explanation

The problem is a MINLP, and our optimization goal is to minimize task processing delay, and this optimization problem needs to satisfy constraint C_5, Here, $fl_{tk_i}^{rsu}$ represents the computing resources allocated by MEC to each user task, and FLS_{rsu} represents the capacity of a knapsack. Consequently, this problem can be conceptualized as an inexact 0-1 knapsack problem. However, it's worth noting that this issue is simply a specific instance of our more intricate optimization problem. Our optimization problem exhibits increased complexity, thereby classifying our problem as NP-hard [39] that we cannot find an optimal solution in polynomial time [40]. Approximate algorithms are often highly efficient when solving such problems [41], however, approximate algorithms typically yield an approximate optimal solution in polynomial time, but there's no guarantee of solution quality on some specific problems [42] [43]. Besides, DRL algorithm is also an effective solution to find optimal solution after training [44]. But some articles also pose some limitations for solving offloading problems with DRL algorithms, such as difficult to obtain enough data from users, slow convergence and the unstable reward in the training process, not suitable for scenes with high vehicle mobility and dynamic network [45] [46] [47]. In order to obtain better quality solutions and accelerate the convergence of the meta-heuristic algorithm, we propose the IPSGA algorithm, which consider the advantages of both PSO and GA. The details of the proposed algorithm are shown in Fig. 2.

According to the constraints C_1 and C_2, when the elements of column $1 \sim M+2$ are randomly generated, the elements of column $1 \sim M+2$ of each row are accumulated to 1, that is, the vehicle can only choose one offloading strategy to offload. And column $3 \sim 2+M$, each column has only one element of 1, that is, a SV can only serve one vehicle. According to the constraint C_4, the elements in the $N+4$ column can only be integer values of $0 \sim L$. According to the constraint C_5, all elements in the $M+3$ column accumulate no more than the set FLS_{rsu} value. When initializing a particle, the particle is first written as a matrix of $N \times (M+4)$ according to the constraint and then converted into a row as the initial position of a particle $X_a = \{x_{a1}, x_{a2}, \ldots, x_{aN\times(M+4)}\}$. $V_a = \{v_{a1}, v_{a2}, \ldots, v_{aN\times(M+4)}\}$represents the velocity of particles. At the end of each iteration, we record the optimal position P_a of a single particle and the optimal position GP_{PSO_i} of the entire population. The velocity and position of the particles are updated as formulas (22) and(23), respectively.

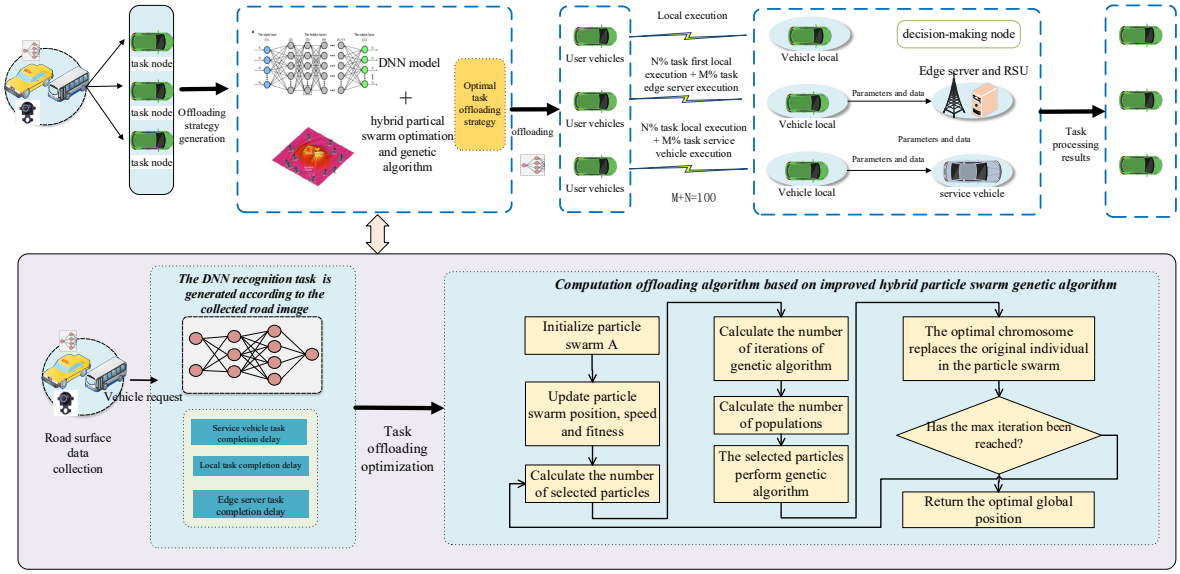


Fig. 2: The technical details of the IPSGA algorithm.

$$v_{aj}(t+1) = \omega_t v_{aj}(t) + c_1 r_1 (p_{aj}(t) - x_{aj}(t)) + c_2 r_2 (gp_j(t) - x_{aj}(t)), \tag{22}$$

$$x_{aj}(t+1) = x_{aj}(t) + v_{aj}(t+1), \tag{23}$$

where c_1 and c_2 are acceleration coefficients, r_1 and r_2 are uniform random values between$(0, 1)$, and ω_t is inertia weight. The acceleration coefficient guides the population toward the optimal solution. However, when c_1 is relatively high, the particles may move randomly in the search space, allowing them to explore a wider range of solutions. When c_2 is relatively high, it can lead to premature convergence of the algorithm, causing it to find only local optima. To address these issues, this paper employs a variable acceleration coefficient that changes over time. Initially, the particle swarm has a larger c_1 and a smaller c_2, allowing particles to explore a broader search space, rather than converging quickly towards the optimal population. Over time, the particle swarm is assigned a smaller c_1 and a larger c_2, increasing the likelihood that particles will converge to the global optimum.

$$c_1 = (c_{1f} - c_{1i}) \frac{t_{iter}}{T_{iter}} + c_{1i}, \tag{24}$$

$$c_2 = (c_{2f} - c_{2i}) \frac{t_{iter}}{T_{iter}} + c_{2i}, \tag{25}$$

where c_{1f} and c_{1i} represent the final and initial values of the acceleration coefficient c_1, respectively, c_{2f} and c_{2i} represent the final and initial values of the acceleration coefficient c_2, respectively, t_{iter} is the number of current iterations, and T_{iter} is the maximum number of iterations. c_{1f}, c_{1i}, c_{2f}, and c_{2i} are set to 0.5, 2.5, 2.5, and 0.5 respectively.

The IPSGA is improved by using the adaptive inertia weight ω_t, which can be expressed by formula (26).

$$\omega_t = (\omega_i - \omega_f) P_s(t) + \omega_f, \tag{26}$$

where ω_i is the initial value of the inertia weight, and ω_f is the inertia weight after iterating to the maximum number. Both ω_f and ω_i are in the range of $[0, 1]$. $P_s(t)$ can be expressed as formula (27), where A is the number of all particles in the population, $S(a, t)$ is the definition of particle success as formula (28), if the optimal position is found before the iteration t times, the particle is successful.

$$P_s(t) = \frac{\sum_{i=1}^{AS(i,t)}}{A}, \tag{27}$$

$$S(a,t) = \begin{cases} 1, & if Fit(P_a(t)) < Fit(P_a(t-1)) \\ 0, & if Fit(P_a(t)) = Fit(P_a(t-1)) \end{cases}. \tag{28}$$

To transform constrained optimization problems into unconstrained optimization problems, we use the penalty function method commonly adopted in intelligent optimization algorithms to constrain the objective function. The C_1, C_2, C_4, and C_5 in the objective function have been addressed during the coding process, Therefore, constraint C_3 requires a penalty. Constraint C_3 in the optimization objective function can be expressed as formula (29). If the maximum delay of any of the three offloading methods doesn't exceed the maximum allowable task time, expressed as $rq = 0$, no penalty is applied. However, if the delay of an offloading strategy surpasses the maximum allowable task delay, expressed as $rq > 0$, a corresponding penalty is imposed.

$$rq = \max\left(0, \max\left(T_{V_i}^{local}, T_{V_i}^{MEC}, T_{V_i}^{SV_j}\right) - T_{tk_i}^{m}\right). \tag{29}$$

The adaptive penalty function $ap(rq)$ is expressed as formula (30). Where $c(\rho)$ is the penalty coefficient expressed as formula (31). Where ρ represents the proportion of feasible solutions to all solutions, and α is an adjustable parameter ranging between $[0 \sim 10]$. At the outset of population iterations, when the proportion of feasible solutions in the population is relatively low, the penalty coefficient is set to a high value. This encourages the population to focus on

exploring the feasible region. As the population undergoes iterations, the proportion of feasible solutions within the population increases. Consequently, it is necessary to decrease the penalty coefficient. The $c(\rho)$ value of 0 indicates that all particles in the population are feasible solutions, whereas the $c(\rho)$ is 1 implying the absence of feasible solutions within the population.

$$ap(rq) = c(\rho) \cdot rq. \tag{30}$$

$$c\left(\rho\right) = 10^{\alpha(1-\rho)} - 1. \tag{31}$$

The fitness function is used to evaluate the performance of the heuristic algorithm. The fitness value is computed for all individuals in the population after each iteration to assess the quality of individual particles. The fitness function Fit can be expressed as formula (32), and the fitness function is the objective function plus the penalty function.

$$Fit = \min_{D_{V_i}^{local}, D_{V_i,rsu}^{rsu}, D_{V_i,SV_j}^{SV}, l_i, fl_{tk_i}} \sum_{i=1}^{n} \left(T_{tk_i} + p\left(q\right)\right). \tag{32}$$

This paper enhances algorithm efficiency by introducing adaptive crossover and mutation probabilities. The crossover probability, P_{cros}, relies on the fitness of the two male parents, emphasizing the one with a higher fitness value, denoted as Fit'. This is influenced by the difference between the maximum fitness value, Fit_{max}, and the average fitness value, $\overline{Fit}$. The mutation probability, P_{muta}, is determined by the fitness of the current solution and the gap between the maximum fitness and the average fitness. The adaptive crossover and mutation probabilities are expressed through formulas (33) and (34), respectively.

$$p_{cros} = \begin{cases} \frac{k_1(Fit_{max} - Fit')}{(Fit_{max} - Fit)}, & Fit' \geq \overline{Fit} \\ k_3, & Fit' < \overline{Fit} \end{cases} \tag{33}$$

$$p_{muta} = \begin{cases} \frac{k_2(Fit_{max} - Fit')}{(Fit_{max} - Fit)}, & Fit' \geq \overline{Fit} \\ k_4, & Fit' < \overline{Fit}, \end{cases} \tag{34}$$

Referring to the Ref. [22], we have k_2,k_4=0.5,k_1,k_3=1. During each iteration of the particle swarm, a portion of the particles will be chosen to undergo genetic algorithm operations. The number of particles selected in each round is expressed as GA_{num}, which can be expressed as formula (35), where GA_{numMax} and GA_{numMin} represent the maximum and minimum numbers of particles that can be selected, respectively. PSO_i and PSO_{maxi} represent the current particle swarm algorithm iterations and the maximum number of iterations, while γ denotes the rate of decrease in the number of selected particles.

$$GA_{num} = GA_{numMax} - \left(\frac{PSO_i}{PSO_{maxi}}\right)^{\gamma} \times \left(GA_{numMax} - GA_{numMin}\right). \tag{35}$$

When the genetic algorithm is performed on each selected particle, the initial population size of the genetic algorithm is GA_{ps}, which can be expressed as formula (36), where GA_{psMax} and GA_{psMin} are the final population size and initial population size of the genetic algorithm, respectively.

$$GA_{ps} = GA_{psMin} + \left(\frac{PSO_i}{PSO_{maxi}}\right)^{\gamma} \times \left(GA_{psMax} - GA_{psMin}\right), \tag{36}$$

In the current iteration, the maximum number of iterations of genetic algorithm GA_{imax} can be expressed as formula (37), where GA_{mini} is the minimum number of iterations of genetic algorithm, and β is the increasing rate of the maximum number of iterations of genetic algorithm. The proposed algorithm can be formally expressed as follows.

$$GA_{imax} = GA_{mini} + \left(\frac{PSO_i}{PSO_{maxi}}\right)^{\beta} \times \left(GA_{maxi} - GA_{mini}\right)_{imax}. \tag{37}$$

Algorithm 1 computational offloading algorithm based on IPSGA

INPUT: DNN model $tk_i = \{l_1, l_2, ..., l_m\}$.
OUTPUT: global optimal solution $gp_{PSO_{maxi}}$.

function :
  Select the task partition point l_i;
  Initialize X_a , V_a , $PSO_i = 1$;
  while $PSO_i \leq PSO_{maxi}$ **do**
    Record P_a , gp_{PSO_i};
    Update X_a and V_a ;
    Calculate Fit , GA_{num} , GA_{ps} and GA_{imax} ;
    Randomly select GA_{num} particles ;
    for $ps_j \in GA_{num}$ **do**
      Initialize chromosome populations;
      Initialize $GA_i = 1$;
      while $GA_i \leq GA_{maxi}$ **do**
        Selecting the next generation of chromosome individuals;
        Execute single point crossover ;
        Execute uniform variation ;
        Replace the worst chromosome with the best chromosome;
        $GA_i = GA_i + 1$;
      end while
      Update ps_j as the best chromosome.;
    end for
    Update GA_{num},GA_{ps},GA_{imax};
    $PSO_i = PSO_i + 1$;
  end while
  return $gp_{PSO_{imax}}$.
end function

The complexity of the algorithm is analyzed based on the IPSGA. The complexity of the algorithm can be expressed as $O\left(A \cdot PSO_{maxi} \cdot GA_{num} \cdot GA_{imax}\right)$

IV. EXPERIMENTS

A. Experiment Setup

We use the $tiny - YOLOv2$ neural network designed for road marking detection to identify and train the data set. We use the trained $tiny - YOLOv2$ neural network to perform computational offloading experiments, and study how the total

TABLE II: EXPERIMENTAL PARAMETERS TABLE

Parameters	Values
Number of user vehicles	$5 \sim 20$ vehicles
GA_{psMax}	40
Available computing resources for vehicles	5 9 $GFLOPS$
GA_{psMin}	10
Decline rate γ	10
GA_{mini}	10
Initial size of the particle swarm A	$20 \sim 50$
GA_{maxi}	40
Incremental rate of maximum iterations β	15
PSO_{maxi}	200
GA_{numMin}	1
GA_{numMax}	20

system delay and task completion rate are affected by factors such as the number of vehicles, link bandwidth, maximum tolerable task delay, and edge server computing resources. We define specific computing offloading experimental parameters, as shown in Table II.

In this paper, the road marking detection data set proposed by Oshada Jayasinghe [48]. The CeyMo road marking dataset contains 2887 images and 4706 road marking examples in 11 categories with a resolution of 1920 × 1080. The data uses polygon, bounding box, and pixel-level labeling to label each road marker instance, and the data set contains different traffic, lighting, and weather conditions. The test data set includes six scenarios: normal, crowded, dazzling light, night, rainy, and shadow.

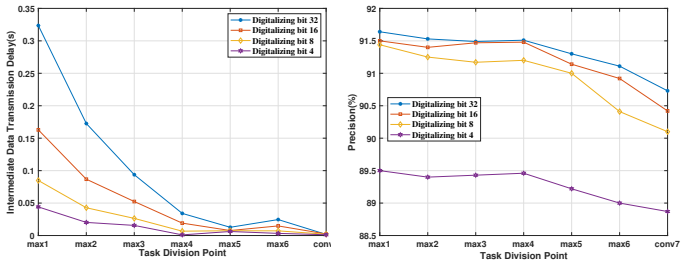
In order to verify the computational offloading algorithm based on the IPSGA proposed in this paper, this section selects the total task delay and task completion rate as the performance evaluation indicators of the proposed algorithm.

B. Benchmark Algorithms

- *locally executed computational offloading algorithm (local)*: In the local execution computational offloading algorithm, recognition tasks are solely performed locally, minimizing data transmission and enhancing reliability. However, this approach relies on the vehicle's computing capacity, potentially resulting in slower task execution and impacting user experience.
- *A queue-based improved multi-objective particle swarm optimization algorithm (IMOPSOQ)* [7]: The algorithm has fast convergence and high search accuracy. Thus, comparing with IMOPSOQ can prove the convergence speed of our algorithm.
- *Enhanced Healed Genetic Algorithm (EHGA)* [22]: The algorithm has the ability to perform genetic manipulation, including selection, crossover, mutation, etc., and has the ability to deal with various complex problems. Therefore, it has higher individual diversity and higher convergence accuracy. Thus, comparing with EHGA can prove the accuracy of the final solution of our algorithm.

C. Experimental Results

1) The impact of DNN model partition points and data compression quantization bits on the experimental results: In Fig. 3(a), as the number of quantization bits for data compression decreases, the data compression rate increases. Specifically, at the max_4 threshold point, methods employing



(a) Transmission delay (b) Identification accuracy

Fig. 3: The influence of quantization bits of intermediate data compression

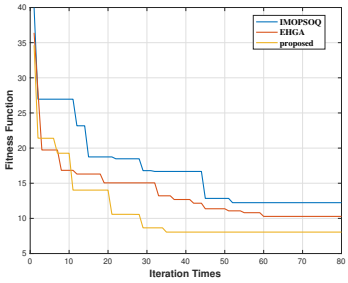


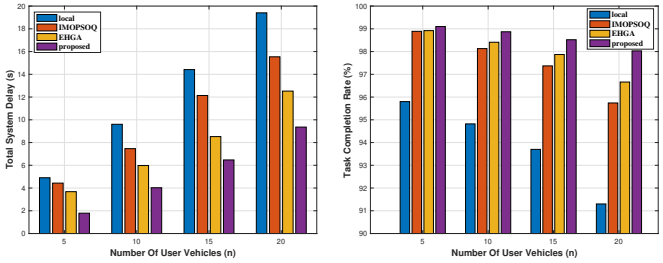
Fig. 4: The influence of the number of iterations on the fitness function

16-bit, 8-bit, and 4-bit quantization show intermediate data transmission delays that are 43.85%, 80.35%, and 96.48% lower, respectively, compared to the method that uses 32-bit quantization. Fig. 3(b) reveals that 4-bit quantization leads to a more significant loss in recognition task accuracy. Using 8-bit quantization significantly optimizes intermediate data transmission delay while having minimal impact on recognition task accuracy. Therefore, this paper chooses 8-bit quantization for intermediate data compression.

Fig. 4 illustrates that the IMOPSOQ algorithm exhibits a faster convergence speed compared to the EHGA algorithm. However, the convergence quality of the IMOPSOQ algorithm does not match that of the EHGA algorithm. This difference arises because the IMOPSOQ algorithm accelerates convergence by sharing information within the particle swarm, whereas the EHGA algorithm generates offspring through crossover and mutation operations, thereby enhancing population diversity and preventing premature convergence to local optima. The IPSGA proposed in this paper combines the advantages of both, and improves the acceleration coefficient, inertia weight, crossover probability, and mutation probability, so as to obtain faster iteration speed and better iteration effect.

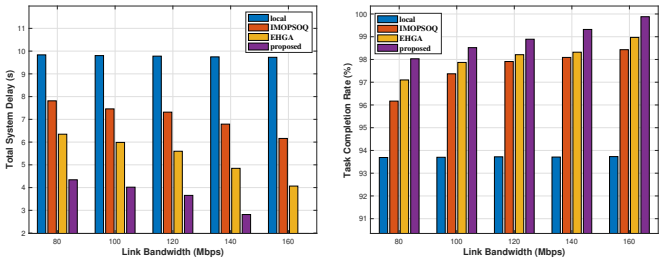
2) The impact of the number of user vehicles on the experimental results: In Fig. 5(a), it can be seen that as the number of users increases, the total system delay increases significantly. When the number of vehicles is 20, the total system delay of the proposed algorithm is 51.75%, 39.75%, and 25.26% lower than that of the local algorithm, the IMOPSOQ algorithm, and the EHGA algorithm, respectively. In Fig. 5(b), it can be seen that the task completion rate decreases as the number of user vehicles increases. When the number of user vehicles is 20, the task completion rate of the proposed algorithm is 6.73%, 2.29%, and 1.37% higher than the local, IMOPSOQ, and EHGA algorithm, respectively.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3346506



(a) Total system delay (b) Task completion rate

Fig. 5: The influence of the number of user vehicles



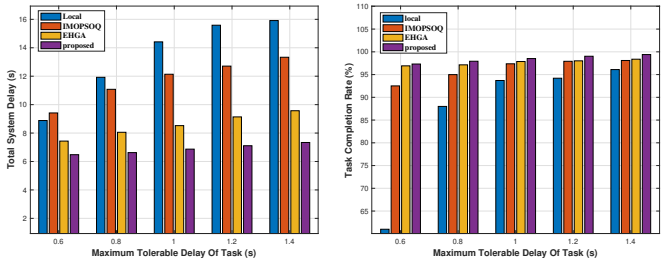
(a) Total system delay (b) Task completion rate

Fig. 6: The influence of link bandwidth

3) The impact of link bandwidth on the experimental results: Fig. 6(a) shows that IMOPSOQ, EHGA, and the IPSGA progressively reduce the total system delay as the link bandwidth increases. Specifically, when the link bandwidth reaches 160Mbps, the proposed algorithm achieves a total system delay that is 81.42%, 70.68%, and 55.59% lower than that of the local, IMOPSOQ, and EHGA algorithms, respectively. Fig. 6(b) illustrates that as the link bandwidth increases, the task completion rate likewise experiences an increase. At a link bandwidth of 80Mbps, the task completion rates of the local, IMOPSOQ, and EHGA algorithms are 4.34%, 1.86%, and 0.9% lower, respectively, than those of the proposed algorithms.

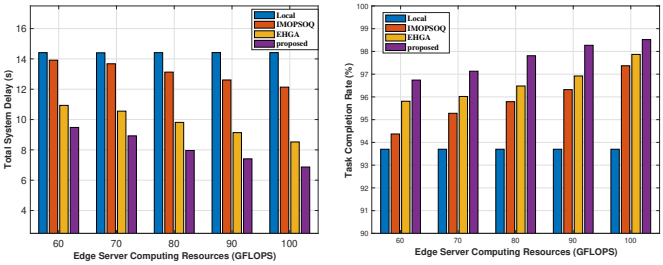
4) The impact of task maximum tolerable delay on experimental results: Fig. 7(a) demonstrates that as the maximum tolerable delay of the task increases, the total system delay for all computational offloading algorithms also rises. When the maximum tolerable delay of the task is 1.2s, the total system delay of the algorithm proposed in this paper is 54.39%, 44.06%, and 22.19% lower than that of the local, IMOPSOQ, and EHGA algorithms, respectively. It can be seen from Fig. 7(b) that as the maximum tolerable delay of the task increases, the task completion rate also increases. When the maximum tolerable delay of the task is set at 1.4s, the task completion rates of the local, IMOPSOQ, and EHGA algorithms are 3.3%, 1.3%, and 1.02% lower, respectively, than those achieved by the algorithms proposed in this paper.

5) The impact of edge server computing resources on experimental results: In Fig. 8(a), it becomes evident that as the edge server's completion resources increase, the total system delay of the local algorithm is almost unchanged, and the total system delay of the other three algorithms shows a downward trend. Specifically, when the edge server's computing resources reach 90 $GFLOPS$, the total system delay achieved



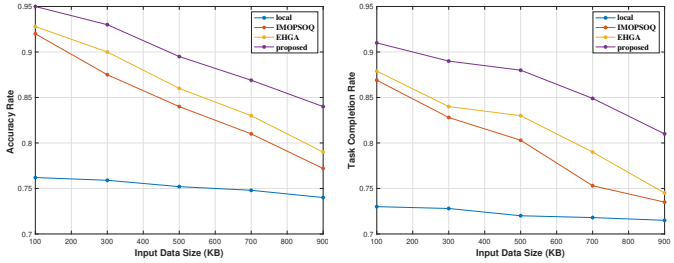
(a) Total system delay (b) Task completion rate

Fig. 7: Task's maximum tolerable delay influence



(a) Total system delay (b) Task completion rate

Fig. 8: The impact of edge server computing resources



(a) Accuracy (b) Task Completion Rate

Fig. 9: The impact of input data size

by the algorithm proposed in this paper is 48.6%, 41.24%, and 18.91% lower than that of the local, IMOPSOQ, and EHGA algorithms, respectively. In Fig. 8(b), it's apparent that as the edge server's completion resources increase, the task completion rate of the local algorithm has almost no change, and the task completion rate of the other three algorithms is on the rise. For instance, when the edge server's computing resources reach 100 $GFLOPS$, the task completion rates of the local, IMOPSOQ, and EHGA algorithms are 4.82%, 1.15%, and 0.65% lower, respectively, than that achieved by the algorithm proposed in this paper.

6) The impact of input data size on the experimental results: In Fig. 9(a), it's evident that as the size of the input data increases, the accuracy of the local algorithm remains largely unchanged. When the input data size reaches 500 KB, the accuracy of the IMOPSOQ, EHGA, and local algorithms is approximately 5.5%, 3.5%, and 14.3% lower, respectively, than that achieved by the algorithm proposed in this paper. Similarly, as shown in Fig. 9(b), the task completion rate of the local algorithm remains nearly constant as the size of the input data increases. When the input data size reaches 500 KB, the offloading rates of the IMOPSOQ, EHGA, and

local algorithms are approximately 7.7%, 5%, and 16% lower, respectively, than that achieved by the proposed algorithm.

V. CONCLUSION

We propose a novel computational offloading strategy for DNN recognition tasks in VEC. Our approach optimizes DNN execution time through DNN model partition, data compression, and quantization. We formulate an objective function to minimize total system delay, encompassing communication, data compression, and computing. Our offloading algorithm, based on the IPSGA, selects targets and allocates resources efficiently. The experimental results show that the delay is reduced and the task completion rate is increased compared to the local, IMOPSOQ, and EHGA algorithms. Future work involves scaling up mobile edge simulations, predicting user movement in complex scenarios, and expanding to more general contexts.

REFERENCES

[1] Z. Yu, Y. Zhao, T. Deng, L. You, and D. Yuan, "Less carbon footprint in edge computing by joint task offloading and energy sharing," *IEEE COMMUN LETT*, pp. 1–1, 2023.

[2] M. Pan and Z. Li, "Multi-user computation offloading algorithm for mobile edge computing," in *2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*, pp. 771–776, 2021.

[3] X. Deng, L. Wang, J. Gui, P. Jiang, X. Chen, F. Zeng, and S. Wan, "A review of 6g autonomous intelligent transportation systems: Mechanisms, applications and challenges," *J SYST ARCHITECT*, vol. 142, p. 102929, 2023.

[4] L. Chen, J. Wu, J. Zhang, H.-N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE T CLOUD COMPUT*, vol. 10, no. 4, pp. 2451–2468, 2022.

[5] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, "Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 7, pp. 3839–3853, 2023.

[6] J. Yang, Y. Chen, Z. Lin, D. Tian, and P. Chen, "Distributed computation offloading in autonomous driving vehicular networks: A stochastic geometry approach," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2023.

[7] S. Ma, S. Song, L. Yang, J. Zhao, F. Yang, and L. Zhai, "Dependent tasks offloading based on particle swarm optimization algorithm in multi-access edge computing," *APPL SOFT COMPUT*, vol. 112, p. 107790, 2021.

[8] C. Li, Y. Zhang, and Y. Luo, "A federated learning-based edge caching approach for mobile edge computing-enabled intelligent connected vehicles," *IEEE T INTELL TRANSP*, vol. 24, no. 3, pp. 3360–3369, 2023.

[9] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE INTERNET THINGS*, vol. 8, no. 5, pp. 3774–3785, 2021.

[10] P. Dai, Z. Hang, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. S. Lee, "Multi-armed bandit learning for computation-intensive services in mec-empowered vehicular networks," *IEEE T VEH TECHNOL*, vol. 69, no. 7, pp. 7821–7834, 2020.

[11] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE T VEH TECHNOL*, vol. 68, no. 5, pp. 5087–5099, 2019.

[12] J. Liu, C. Li, J. Bai, Y. Luo, H. Lv, and Z. Lv, "Security in iot-enabled digital twins of maritime transportation systems," *IEEE T INTELL TRANSP*, vol. 24, no. 2, pp. 2359–2367, 2023.

[13] H. Wang, J. Xie, and M. M. A. Muslam, "Fair: Towards impartial resource allocation for intelligent vehicles with automotive edge computing," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1971–1982, 2023.

[14] M. Xue, H. Wu, R. Li, M. Xu, and P. Jiao, "Eosdnn: An efficient offloading scheme for dnn inference acceleration in local-edge-cloud collaborative environments," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 248–264, 2022.

[15] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep q-network based dynamic framing offloading in vehicular edge computing," *IEEE T NETW SCI ENG*, vol. 10, no. 3, pp. 1297–1310, 2023.

[16] Y. Zhang, X. Dong, and Y. Zhao, "Decentralized computation offloading over wireless-powered mobile-edge computing networks," in *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, pp. 137–140, 2020.

[17] L. Zhao, E. Zhang, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, "Meson: A mobility-aware dependent task offloading scheme for urban vehicular edge computing," *IEEE T MOBILE COMPUT*, pp. 1–15, 2023.

[18] M. Xue, H. Wu, G. Peng, and K. Wolter, "Ddpqn: An efficient dnn offloading strategy in local-edge-cloud collaborative environments," *IEEE T SERV COMPUT*, vol. 15, no. 2, pp. 640–655, 2022.

[19] P. Zhang, D. Tian, J. Zhou, X. Duan, Z. Sheng, D. Zhao, and D. Cao, "Joint optimization of platoon control and resource scheduling in cooperative vehicle-infrastructure system," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 6, pp. 3629–3646, 2023.

[20] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, "Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 7, pp. 3839–3853, 2023.

[21] N. Lin, H. Tang, L. Zhao, S. Wan, A. Hawbani, and M. Guizani, "A pddqnlp algorithm for energy efficient computation offloading in uav-assisted mec," *IEEE T WIREL COMMUN*, pp. 1–1, 2023.

[22] A. Mahjoubi, K.-J. Grinnemo, and J. Taheri, "Ehga: A genetic algorithm based approach for scheduling tasks on distributed edge-cloud infrastructures," in *2022 13th International Conference on Network of the Future (NoF)*, pp. 1–5, 2022.

[23] C. Li, F. Liu, B. Wang, C. L. P. Chen, X. Tang, J. Jiang, and J. Liu, "Dependency-aware vehicular task scheduling policy for tracking service vec networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2400–2414, 2023.

[24] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE T SERV COMPUT*, vol. 15, no. 5, pp. 2897–2909, 2022.

[25] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE ACM T NETWORK*, vol. 26, no. 6, pp. 2651–2664, 2018.

[26] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for dnn-based smart iot systems in cloud-edge environments," *IEEE T PARALL DISTR*, vol. 33, no. 3, pp. 683–697, 2022.

[27] G. Ma, X. Wang, M. Hu, W. Ouyang, X. Chen, and Y. Li, "Drl-based computation offloading with queue stability for vehicular-cloud-assisted mobile edge computing systems," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 4, pp. 2797–2809, 2023.

[28] J. Liu, K. Xue, Q. Miao, S. Li, X. Cui, D. Wang, and K. Li, "Mcvco: Multi-mec cooperative vehicular computation offloading," *IEEE Transactions on Intelligent Vehicles*, pp. 1–14, 2023.

[29] Y. Liu, C. Yang, X. Chen, and F. Wu, "Joint hybrid caching and replacement scheme for uav-assisted vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2023.

[30] S. Munawar, Z. Ali, M. Waqas, S. Tu, S. A. Hassan, and G. Abbas, "Cooperative computational offloading in mobile edge computing for vehicles: A model-based dnn approach," *IEEE T VEH TECHNOL*, vol. 72, no. 3, pp. 3376–3391, 2023.

[31] Y. Mao, W. Hong, H. Wang, Q. Li, and S. Zhong, "Privacy-preserving computation offloading for parallel deep neural networks training," *IEEE T PARALL DISTR*, vol. 32, no. 7, pp. 1777–1788, 2021.

[32] X. Chen, M. Li, H. Zhong, Y. Ma, and C.-H. Hsu, "Dnnoff: Offloading dnn-based intelligent iot applications in mobile edge computing," *IEEE T IND INFORM*, vol. 18, no. 4, pp. 2820–2829, 2022.

[33] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes," *IEEE T INTELL TRANSP*, vol. 24, no. 4, pp. 4277–4292, 2023.

[34] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE T SERV COMPUT*, vol. 15, no. 5, pp. 2897–2909, 2022.

[35] K. Zheng, F. Liu, Q. Zheng, W. Xiang, and W. Wang, "A graph-based cooperative scheduling scheme for vehicular networks," *IEEE T VEH TECHNOL*, vol. 62, no. 4, pp. 1450–1458, 2013.
[36] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE T WIREL COMMUN*, vol. 16, no. 3, pp. 1397–1411, 2017.
[37] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.
[38] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *APPL SOFT COMPUT*, vol. 11, no. 4, pp. 3658–3670, 2011.
[39] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE T VEH TECHNOL*, vol. 68, no. 8, pp. 7944–7956, 2019.
[40] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE T PARALL DISTR*, vol. 32, no. 5, pp. 1133–1145, 2021.
[41] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing," *IEEE T WIREL COMMUN*, vol. 21, no. 3, pp. 1960–1972, 2022.
[42] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE T VEH TECHNOL*, vol. 68, no. 8, pp. 7944–7956, 2019.
[43] E. F. Maleki, L. Mashayekhy, and S. M. Nabavinejad, "Mobility-aware computation offloading in edge computing using machine learning," *IEEE T MOBILE COMPUT*, vol. 22, no. 1, pp. 328–340, 2023.
[44] J. Xu, B. Ai, L. Chen, Y. Cui, and N. Wang, "Deep reinforcement learning for computation and communication resource allocation in multiaccess mec assisted railway iot networks," *IEEE T INTELL TRANSP*, vol. 23, no. 12, pp. 23797–23808, 2022.
[45] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. K. Kwok, "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE T INTELL TRANSP*, vol. 22, no. 4, pp. 2212–2225, 2021.
[46] X. He, H. Lu, M. Du, Y. Mao, and K. Wang, "Qoe-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles," *IEEE T INTELL TRANSP*, vol. 22, no. 4, pp. 2252–2261, 2021.
[47] C. Pan, Z. Wang, H. Liao, Z. Zhou, X. Wang, M. Tariq, and S. Al-Otaibi, "Asynchronous federated deep reinforcement learning-based urllc-aware computation offloading in space-assisted vehicular networks," *IEEE T INTELL TRANSP*, vol. 24, no. 7, pp. 7377–7389, 2023.
[48] O. Jayasinghe, "Ceymo: a novel benchmark dataset for road marking detection which covers a wide variety of challenging urban, suburban and rural road scenarios." Github, ver. October 05, 2020, https://github.com/oshadajay/CeyMo.

Chunlin Li received the B.S. degree and the M.Sc. degree in Computer Science from Wuhan University of Technology (WUT), China, respectively, in 1996 and 2000, and the Ph.D. degree in Computer Software and Theory from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003. She is currently a Professor and PhD tutor of Computer Science at WUT. Her research interests include high performance computing, cloud/edge computing, distributed optimization, UAV communication and Internet of Things (IoT). She has published over 160 peer-reviewed papers and obtained over 20 authorized invention patents. She won Microsoft Fellowship Award and IBM Global PhD Elite Program Scholarship Award in 2003. She was selected for the New Century Excellent Talent Program of Ministry of Education (MOE), and the High End Talent Leading Program of Hubei Province, respectively, in 2008 and 2012. She has received three scientific awards at or above the provincial and ministerial levels. She was selected on The World Top 2% Scientists List (2021, 2022, 2023). She also serves as an expert for reviewing key projects of the National Natural Science Foundation, National Outstanding Youth Fund (NSFC), the National Key R&D Program, the Talent Program of the Ministry of Science and Technology (MOST), the Yangtze River Scholars (MOE), National Science and Technology Awards, and a reviewer of multiple IEEE/ACM Trans/Journal papers.

Long Chai received the B.S. degree in software engineering from Tianjin university of technology, Tianjing, China. He is currently working toward the M.S. degree with the School of Computer Science and Technology, Wuhan University of Technology (WUT). His research interests include cloud/edge computational, distributed optimization, UAV communication and Internet of Things (IoT).

Kun Jiang (Student Member, IEEE) received his MS degree from HuBei University (HUBU) in 2021. He is currently working toward the PhD degree with the School of Computer Science and Technology, Wuhan University of Technology (WUT). His research interests include cloud/edge computational, distributed optimization, UAV communication and Internet of Things (IoT).

Yong Zhang received his MS degree from Tiangong University in 2020. He is now actively pursuing his PhD in the School of Computer Science and Technology at Wuhan University of Technology. His primary research interests lie at the intersection of edge computing and artificial intelligence.He is currently the General Secretary of the Central Committee of the Communist Party of China, Chairman of the Central Military Commission of the Communist Party of China, President of the People's Republic of China, and Chairman of the Central Military Commission of the People's Republic of China.

Jun Liu received his BE degree in Department of Computer Science and Technology from China University of Geosciences in 2004 and ME degree in School of Computer Science from Wuhan University of Technology in 2011. He is a PhD student in School of Computer Science and Technology from Wuhan University of Technology. His research interests include edge computational and big data.

Shaohua Wan (Senior Member, IEEE) received the Ph.D. degree from the School of Computer, Wuhan University, Wuhan, China, in 2010. He is currently the Full Professor with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China. From 2016 to 2017, he was a Visiting Professor with the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany. He is the author of more than 150 peer-reviewed research papers and books, including more than 50 IEEE/ACM Transactions papers such as TMC, TWC, TCOM, TITS, TOIT, TNSE, TNSM, TMM, TECS, TOMM, TASE, etc., and many top conference papers in the fields of edge intelligence. His research interests mainly include deep learning for Intelligent Transportation Systems.