Received 22 October 2024, accepted 6 November 2024, date of publication 11 November 2024, date of current version 20 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3495615

RESEARCH ARTICLE

MAT-IGA: A Deadline and Priority-Aware Task Offloading Method in Resource-Constrained Vehicular Networks

LONG ZHANG[1,2,3], RUI CAO[4], WEI LEI[1,2,3], YAZHOU WANG[1,2,3], ZHENLONG XIE[5], AND BINGXIN NIU[4]

[1]Hebei Provincial Communications Planning, Design and Research Institute Company Ltd., Shijiazhuang 050011, China
[2]Research and Development Center of Transport Industry of Self-Driving Technology, Shijiazhuang 050011, China
[3]Vehicle-Road-Cloud Network Hebei Engineering Research Center, Shijiazhuang 050011, China
[4]School of Artificial Intelligence, Hebei Province Key Laboratory of Big Data Calculation, Hebei University of Technology, Hongqiao 300130, China
[5]Jingxiong Cloud Control (Beijing) Intelligent Transportation Technology Company Ltd., Beijing 100083, China

Corresponding author: Bingxin Niu (niubingxin666@163.com)

This work was supported in part by Hebei Province Higher Education Science and Technology Research Project under Grant QN2024201, in part by Hebei Natural Science Foundation under Grant F2024202047, and in part by the General Program of National Natural Science Foundation of China under Grant 62376088.

ABSTRACT With the developments of the intelligent vehicles, the most significant contradiction is coming to the limited computing resource and the unlimited requirements. Task offloading has been proposed to overcome this. Most of the existing task offloading methods aim to optimize the global delay. However, from the user's point of view, different in-vehicle applications (such as entertainment and autonomous driving) have different importance and delay requirements. Therefore, simply minimizing the latency of all tasks does not meet the QoS (Quality of Service) of each user. Unfortunately, very few people discuss this practical issue. In this paper, the problem of vehicular task offloading in resource-constrained scenarios is studied, and a two-stage task offloading scheme MAT-IGA based on multi-node collaboration is proposed. In the first stage, the complex tasks are pre-segmented adaptively, and the optimal matching set is solved by combining the improved multi-round deferred-acceptance algorithm, which prioritizes the resource requirements of emergency tasks and improves the reliability of the strategy. In the second stage, a chaotic genetic algorithm based on opposition-based learning is used to optimize resource allocation, and the global delay and cost are optimized on the basis of ensuring the success rate. The simulation results show that the proposed method is superior to the common baseline algorithm in different vehicle numbers and task characteristics.

INDEX TERMS Task offloading, vehicular networks, matching algorithm, genetic algorithm, mobile edge computing.

I. INTRODUCTION

With the continuous development of vehicle networking and artificial intelligence technology, people's requirements for vehicles are not limited to vehicle performance and driving safety, but also put forward higher requirements for vehicle applications. Nowadays, vehicles are given more and more computing power, and there are more and more types of in-vehicle applications, such as in-vehicle ultra-definition video, augmented reality (AR), and unmanned driving. At present, relying solely on the computing power of the vehicle itself can no longer meet the calculation requirements of a large amount of data [1].

In the past, cloud computing was the mainstream method to solve the above problems. However, with the increase of the number of on-board sensors and the explosive growth of data volume, this method has gradually become unsuitable due to its large communication delay and waste of bandwidth resources. Nowadays, the task offloading method based on Mobile Edge Computing (MEC) is considered to be an effective example to solve the above problems. By deploying Edge Server (ES) with computing and storage capabilities

The associate editor coordinating the review of this manuscript and approving it for publication was Leandros Maglaras.

on one side of the road, vehicle users can communicate with edge servers and core cloud networks through wireless transmission, and offload complex tasks that are difficult for vehicles to handle to MEC servers for processing, thus solving the problem of limited local computing power and ensuring the efficient execution of many resource-intensive tasks.

The MEC-based task offloading has the characteristics of low latency, high reliability and large data storage capacity, but the movement of vehicles, channel interference and resource constraints have brought difficulties to task offloading. Therefore, in the face of complex and diverse vehicular scenarios, how to reasonably and effectively use the limited spectrum resources and computing resources in the vehicular networks and build a reasonable offloading decision is still a research challenge.

In order to achieve efficient vehicular task offloading in resource-constrained scenarios, the main contributions of this paper are as follows:

1) a typical road vehicular task offloading model is constructed, and a variety of heterogeneous resources of cloud, roadside unit and crowd-sourcing node are introduced. Considering the computing power and communication ability of heterogeneous nodes, various types of constraints are designed, and the optimization goal is to minimize the task offloading delay and cost.

2) A two-stage task offloading method MAT-IGA is proposed. In the first stage, a new lightweight task set is obtained by slicing and reorganizing the task, and then the improved deferred-acceptance algorithm is used to solve the optimal matching set. In the second stage, the chaotic genetic algorithm based on opposition-based learning is used to solve the resource allocation problem, which further optimizes the delay and cost while ensuring the success rate of the task.

3) Simulation results show that the proposed scheme can effectively reduce the average delay and cost in the system, and improve the success rate of task offloading.

The remaining parts of this paper are organized as follows. The related works are presented in Section II. The system model is introduced in Section III. Section IV describes the problem formulation of the task offloading problem. Section V propose the task offloading method based on matching and genetic algorithm. Numerical Results are provided in Section VI. Section VII concludes this paper.

II. RELATED WORKS

Vehicular task offloading mainly solves the problem of where the task should be offloaded and how much resources should be allocated. It is a typical mixed integer nonlinear programming problem. However, edge resources are usually not rich. In order to improve the utilization of edge resources, there have been a lot of research and solutions. According to the specific resource utilization strategy, it can be divided into the following three types: task segmentation strategy, cloud-edge collaboration strategy and vehicle assistance strategy.

Task segmentation is mainly to divide complex tasks into multiple small tasks and hand them over to different nodes to improve the utilization of edge resources. Reference [2] proposed an offloading strategy based on multi-base station collaborative computing. The author used the DDPG (Deep Deterministic Policy Gradient) algorithm [3] to segment the task, and used the parallel computing of adjacent base stations to optimize the task delay. Reference [4] mainly studies the problem of task offloading and resource allocation in the Internet of Things system. The author solves the optimal task splitting ratio, transmission power and resource allocation number through an iterative algorithm to minimize the task offloading delay. Reference [5] studied a multi-user system with random wireless channels. The author proposed a partial offloading method based on DDPG to minimize user power consumption and buffer delay. Reference [6] considered the offloading problem of vehicle tasks under the edge-end two-tier architecture, in order to solve the problem of data dependence between tasks, the author proposed a task offloading method based on A3C (Asynchronous Advantage Actor-Critic) [7], which can achieve effective long-term optimization strategies. Reference [8] proposed an intelligent computing offloading scheme for IoT-dependent applications, which achieves low-latency and low-energy offloading requirements through offline training and online deployment. Reference [9] considered the privacy problem of partial offloading and proposed an offloading algorithm based on differential privacy and deep reinforcement learning, which hides the amount of real task offloading data by introducing noise.

Edge computing has strong real-time and reliability, but the disadvantage is that resources are scarce and unevenly distributed. However, the cloud computing can provide additional computing resources. Therefore, some studies have introduced cloud servers and designed related task offloading strategies based on cloud-edge collaboration. Reference [10] considered the optimization of delay and cost in data-driven scenarios, and proposed an asynchronous ADQN algorithm for task offloading based on the ideas of A3C and DQN. Reference [11] proposed a task offloading method based on deep reinforcement learning, which combines historical request records and available resources in the system to optimize task offloading decisions. Reference [12], considering the optimization requirements of different applications between service delay and result quality loss, an event-triggered dynamic task allocation framework based on linear programming optimization and binary particle swarm optimization is proposed. Reference [13] proposed a vehicular edge-cloud computing network. The author designed a value-based reinforcement learning method to minimize task execution time through the cooperation of edge nodes and cloud resources. Aiming at the security problem in vehicle networks, a trust evaluation algorithm is proposed in the [14]. This method can improve the reliability of task processing while minimizing the task offloading delay. However, the author only considers the optimization of global delay, ignoring the success rate of different types of tasks.

In addition to edge infrastructure and cloud servers, vehicles can also provide additional resources. Although the computing power of vehicles is weaker than the clouds or edge servers, a large number of storage space and computing resources can be obtained by aggregating large vehicle clusters. Therefore, some studies consider introducing vehicles into task offloading scenarios and designing related vehicle-assisted offloading methods. Reference [15] proposed an autonomous vehicle edge framework for road edge computing, and designed a scheduling algorithm based on ant colony algorithm to solve the task allocation problem. Reference [16] proposed a computing model for vehicle fog computing, mainly considering the use of vehicles in the parking lot to provide computing services, and proposed a dynamic resource allocation algorithm to solve the problem. Reference [17], considering that the MEC server is not available or the resources are insufficient, the author uses the surrounding vehicles as a resource library and proposes a distributed computing offloading method to utilize all resources. Reference [18] considered a task offloading problem in a vehicle cloud scenario. The author integrated a variety of factors such as the high mobility of the vehicle, the heterogeneity of computing power and the interdependence of tasks, and proposed an improved genetic algorithm based on integers to solve the task offloading problem. The scheme can significantly improve the utilization of computing resources while ensuring low latency and system stability. Reference [19] considered the mobility of the vehicle and the load of nodes, and proposed a task offloading method based on deep reinforcement learning to reduce the offloading cost. However, this method ignores the delay sensitivity difference of the vehicular task, and it is difficult to meet the actual application needs.

In summary, most of the existing methods aim at optimizing the global delay or energy consumption under the premise of satisfying the task delay constraint, but the delay constraint of the task is not always satisfied. For example, in the case of resource supply and demand imbalance, some tasks will inevitably have offloading timeouts. At this time, it is not feasible to only optimize the global delay. In order to solve this problem, [20] proposed a task offloading method based on Stackelberg game, which maximized the benefits through resource sharing between nodes. However, this method does not consider the importance difference of tasks, so there is still room for improvement in resource allocation. Reference [21] proposed to use the weight to adjust the attention to the task, so as to improve the success rate of offloading. However, the weight is usually difficult to determine, and the heterogeneity of the task (such as the degree of urgency and the difference of delay demand) is ignored. Reference [22] considered improving the revenue of critical tasks and selectively giving up secondary tasks, but the abandonment strategy is not desirable. Some secondary tasks cannot be completed on time, but users are usually acceptable for a small number of timeouts.

III. SYSTEM MODEL

The task offloading scenario assumed in this paper consists of Task Vehicles (TaVs), Rode Site Unit (RSU), Cloud and Crowd Computing Nodes (CCNs). Among them, RSU is a roadside infrastructure that is responsible for centralized unified task scheduling. TaVs refer to task vehicles, and each TaV will carry a vehicle application task. CCNs include SeVs (Service Vehicles) and roadside APs (Access Points), which are added to the task offloading process in the form of crowd-sourcing and provide additional computing resources. In order to accurately describe the characteristics of each vehicle, this paper models in a two-dimensional way, defining the road as the x-axis, the right as the positive direction, the direction of the RSU as the y-axis, and the upward as the positive direction. The system adopts a centralized decision-making method for unified decision-making, and the system model is shown in Figure 1.

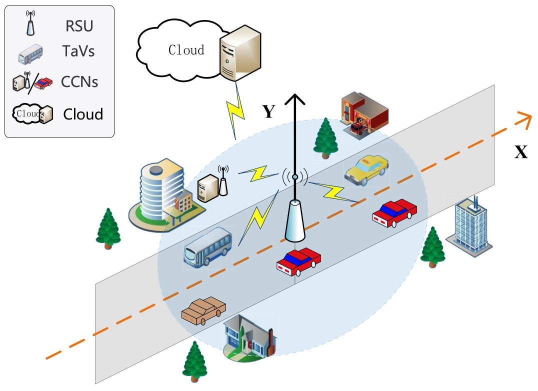

FIGURE 1. Vehicular task offloading model.

A. NETWORK DEFINITION

Define the set $M = \{1, \ldots, m, \ldots, M\}$ to represent TaVs, and the set $N = \{1, \ldots, n, \ldots, N\}$ to represent CCNs. Each TaV carries a vehicular task, the characteristics of which can be represented as a tuple $(D_m, C_m, T_m^d, \theta_m)$, where D_m is the size of the task data, including algorithm data D_m^a and computation data D_m^b. The algorithm data refers to the technical support required for the task, such as related code or models, while the computation data refers to the data to be processed by the vehicle, such as data collected by onboard sensors. C_m is the number of CPU cycles required for the task, T_m^d represents the delay requirement of the task. θ_m indicates the urgency of the task, the larger the value, the stronger the urgency of the task. By partitioning the computation data, the task can be divided into multiple subtasks. The set of subtasks for TaV m can be represented as R_m. The data volume and computational complexity of subtask r $(r \in R_m)$ can be respectively represented as D_r and C_r. All subtasks need to be processed within the time T_m^d. The computing frequency of the RSU can be defined as F_R, and the computing frequency of CCN n is defined as F_n.

Assuming the antenna height of the RSU is h, thus the RSU can be represented by the coordinates $(0, h)$. Assume that TaV m travels at a constant speed on a horizontal road, with the initial position represented as $(l_m^T, 0)$, and its speed relative to the RSU is v_m^T, where the right direction is considered as the positive direction of speed. Similarly, the initial position of CCN n can be represented as $(l_n^S, 0)$, and its speed relative to the RSU is v_n^S. Among them, the speed of APs is a constant 0.

Due to the movement of vehicles causing the relative distance to constantly change, the variable $d_{mn}(t)$ is defined to represent the distance between TaV m and CCN n at time t. Similarly, the variable $d_{mR}(t)$ is used to represent the distance between TaV m and the RSU.

According to the kinematic equations, the calculation method for the variable $d_{mn}(t)$ is as follows:

$$d_{mn}(t) = \left| l_m^T - l_n^S + (v_m^T - v_n^S)t \right| \tag{1}$$

Combining the Pythagorean theorem, the calculation method for $d_{mR}(t)$ is as follows:

$$d_{mR}(t) = \sqrt{h^2 + (l_m^T + v_m^T t)^2} \tag{2}$$

Orthogonal Frequency Division Multiplexing (OFDM) technology is used in communication. It allows for the transmission of multiple subcarriers simultaneously, which helps in achieving high data rates suitable for Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications [23]. Additionally, its ability to manage inter-symbol interference ensures reliable communication links even as vehicles travel at high speeds. These features are critical for maintaining robust V2I and V2V communications, supporting seamless connectivity in dynamic transportation environments. In this paper, $R_{rn}(t)$ is defined as the instantaneous data transmission rate of subtask r transmitted to CCN n by TaV m. $R_{rR}(t)$ is the instantaneous data transmission rate of subtask r transmitted to RSU. The rate decreases with the increase of distance. The calculation method can be obtained by Shannon formula:

$$R_{rn}(t) = B_{nr}\log_2\left[1 + \frac{\rho P}{\sigma^2\left[d_{mn}(t)\right]^\alpha}\right] \tag{3}$$

$$R_{rR}(t) = B_{Rr}\log_2\left[1 + \frac{\rho P}{\sigma^2\left[d_{mR}(t)\right]^\alpha}\right] \tag{4}$$

where ρ is the channel power gain at this distance, α is the path loss index, P is the transmission power, σ^2 is the noise power, B_{nr} andB_{Rr} are the channel bandwidth allocated by CCN n and RSU to subtask r, respectively.

B. ACTION CONSTRAINT

If the subtask r of TaV m is offloaded to CCN n, the variable $a_{rn} = 1$ is defined, otherwise $a_{rn} = 0$. Similarly, if the task r of TaV m is offloaded to the RSU, the variable $b_r = 1$, otherwise it is 0. If the task r of TaV m is offloaded to Cloud, the variable$c_r = 1$, otherwise $c_r = 0$.

For any TaV m, each subtask can only be offloaded to one position, so the action constraint can be obtained:

$$b_r + \sum_{n \in N} a_{rn} + c_r = 1, \quad m \in \mathcal{M}, \quad r \in \mathcal{R}_m \tag{5}$$

Since the vehicle is constantly moving, the communication time of the vehicle is limited. Define τ_{mn} as the maximum duration of TaV m and CCN n that can maintain the connection. According to the kinematic formula, the calculation method of τ_{mn} is defined as:

$$\tau_{mn} = Q\left(\left|\left(l_m^T - l_n^S\right)\right| \le R\right) \frac{R - \left(l_m^T - l_n^S\right) sign\left(v_m^T - v_n^S\right)}{\left|\left(v_m^T - v_n^S\right)\right|} \tag{6}$$

Among them, R is the V2V transmission range at a fixed transmission power P. Q is an indicator function. When the discriminant is true, the value is 1, otherwise it is 0. *Sign* is a symbol function, and a positive value indicates that the two nodes are gradually separating, otherwise it means that they are close. In τ_{mn} time, if the task can be offloaded, then $\int_0^{\tau_{mn}} R_{rn}(t)dt \ge D_r$, then the offloading action constraint can be formulated as:

$$a_{rn} \le \frac{\int_0^{\tau_{mn}} R_{rn}(t)dt}{D_r} \tag{7}$$

Because the result data of the task is relatively small, the backhaul delay of the task is not considered. When a task is offloaded to RSU, T_{rR}^{com} is defined as the delay of subtask r transmitted to RSU by TaV m, T_{rR}^{cmp} is the calculation delay of subtask r in RSU, f_{Rr} is the CPU resource allocated by RSU to subtask r, B_{Rr} is the bandwidth allocated by RSU to subtask r, and the sum of allocated resources cannot be greater than the total resources B_R and F_R, so there are:

$$\int_0^{T_{rR}^{com}} R_{rR}(t)dt = D_r, \quad m \in \mathcal{M}, \quad r \in \mathcal{R}_m \tag{8}$$

$$T_{rR}^{cmp} = \frac{C_r}{f_{Rr}}, \quad m \in \mathcal{M}, \quad r \in \mathcal{R}_m \tag{9}$$

$$\sum_{m \in \mathcal{M}} \sum_{r \in \mathcal{R}_m} b_r f_{Rr} \le F_R \tag{10}$$

$$\sum_{m \in \mathcal{M}} \sum_{r \in \mathcal{R}_m} b_r B_{Rr} \le B_R \tag{11}$$

When the task is offloaded to CCNs, T_{rn}^{com} is defined as the delay of TaV m sending subtask r to CCN n, T_{rn}^{cmp} is the calculation delay of TaV m subtask r at CCN n, f_{nr} is the CPU frequency allocated to r by CCN n, B_{nr} is the bandwidth allocated to r by CCN n, and the sum of allocated resources cannot be greater than the total resources B_n and F_n of CCN n, so there are:

$$\int_0^{T_{rn}^{com}} R_{rn}(t)dt = D_r, \quad m \in \mathcal{M}, \quad n \in \mathcal{N}, \quad r \in \mathcal{R}_m \tag{12}$$

$$T_{rn}^{cmp} = \frac{C_r}{f_{nr}}, \quad m \in \mathcal{M}, \quad n \in \mathcal{N}, \quad r \in \mathcal{R}_m \tag{13}$$

$$\sum_{m\in M}\sum_{r\in\mathcal{R}_m} a_{rn}f_{nr} \le F_n, \quad n\in\mathcal{N} \tag{14}$$

$$\sum_{m\in M}\sum_{r\in\mathcal{R}_m} a_{rn}B_{nr} \le B_n, \quad n\in\mathcal{N} \tag{15}$$

When task r is offloaded to Cloud, T_{rc}^{com} is defined as the transmission delay of TaV m sending subtask r to Cloud, $T_{rc}^{cmp}$is the computing delay of subtask r of TaV m in Cloud, f_{cr} is the computing resource allocated to the task by the cloud server, and R_c is the uplink channel data rate of the task transmitted from the cellular interface to Cloud, so we can get:

$$T_{rc}^{com} = \frac{D_r}{R_c}, \quad m\in\mathcal{M},\ r\in\mathcal{R}_m \tag{16}$$

$$T_{rc}^{cmp} = \frac{C_r}{f_{cr}}, \quad m\in\mathcal{M},\ r\in\mathcal{R}_m \tag{17}$$

The task is offloaded to RSU, CCN or Cloud, so the offloading delay and computing delay of subtask r can be expressed as:

$$T_r^{off} = b_rT_{rR}^{com} + \sum_{n\in\mathcal{N}} a_{rn}T_{rn}^{com} + c_rT_{rc}^{com} \tag{18}$$

$$T_r^{cal} = b_rT_{rR}^{cmp} + \sum_{n\in\mathcal{N}} a_{rn}T_{rn}^{cmp} + c_rT_{rc}^{cmp} \tag{19}$$

In order to constrain the proportion of communication resource allocation, this paper defines different transmission costs for nodes. It is assumed that the cost of unit data transmission to RSU, CCNs and Cloud is φ_R^{com}, φ_n^{com} and φ_c^{com}, respectively, where each CCN can have different costs, so the transmission cost of subtask r is:

$$Cost_r^{off} = b_r\varphi_R^{com}D_r + \sum_{n\in\mathcal{N}} a_{rn}\varphi_n^{com}D_r + c_r\varphi_c^{com}D_r \tag{20}$$

Similarly, in order to constrain the allocation of computing resources, the unit computing resource costs of RSU, CCN and Cloud are defined as φ_R^{cmp}, φ_n^{cmp} and φ_c^{cmp}, so the computing cost of subtask r can be obtained as:

$$Cost_r^{cal} = b_r\varphi_R^{cmp}f_{Rr}C_r + \sum_{n\in\mathcal{N}} a_{rn}\varphi_n^{cmp}f_{nr}C_r + c_r\varphi_c^{cmp}f_{cr}C_r \tag{21}$$

The transmission and computing costs of the cloud are much higher than the other two. In summary, the total offloading delay and cost of subtask r can be expressed as:

$$T_r^{total} = T_r^{off} + T_r^{cal} \tag{22}$$

$$Cost_r^{total} = Cost_r^{off} + Cost_r^{cal} \tag{23}$$

IV. PROBLEM FORMULATION

Through the definition and analysis of the problem, this paper aims to minimize the delay and cost of the task, and the final optimization problem can be expressed in the form of P1.

$$\begin{aligned} P1 :& \min_{E,S} \sum_{m\in\mathcal{M}}\sum_{r\in\mathcal{R}_m} \mu Cost_r^{total} + \theta_m(1-\mu)T_r^{total} \\ s.t.\ & C1 : b_r + \sum_{n\in N} a_{rn} + c_r = 1, \quad m\in\mathcal{M},\ r\in\mathcal{R}_m \\ & C2 : \int_0^{\tau_{mn}} R_{rn}(t)dt \ge D_r, T_r^{total} \le T_m^d \\ & C3 : \sum_{m\in\mathcal{M}}\sum_{r\in\mathcal{R}_m} b_rf_{Rr} \le F_R, \sum_{m\in\mathcal{M}}\sum_{r\in\mathcal{R}_m} b_rB_{Rr} \le B_R \\ & C4 : \sum_{m\in M}\sum_{r\in\mathcal{R}_m} a_{rn}f_{nr} \le F_n, \sum_{m\in M}\sum_{r\in\mathcal{R}_m} a_{rn}B_{nr} \le B_n \\ & C5 : f_{cr}\in R^+, \quad m\in\mathcal{M},\ n\in\mathcal{N},\ r\in\mathcal{R}_m \end{aligned} \tag{24}$$

where $E = \{a_{rn}, b_r, c_r | m \in \mathcal{M}, n \in \mathcal{N}, r \in \mathcal{R}_m\}$, $S = \{B_{nr}, B_{Rr}, f_{nr}, f_{Rr}, f_{cr} | m \in \mathcal{M}, n \in \mathcal{N}, r \in \mathcal{R}_m\}$, μ is the weight of the cost, C1 is the offloading action constraint, C2 is the offloading delay constraint, C3 is the RSU resource allocation constraint, C4 is the CCNs resource allocation constraint, and C5 is the cloud server resource allocation constraint. In the actual scenario, due to resource constraints, the constraint C2 may be difficult to meet. Therefore, in the algorithm design process, priority should be given to ensuring the successful execution of emergency tasks, and on this basis, the global delay and cost should be optimized. Based on this idea, this paper decomposes the original problem into two sub-problems: offloading decision and resource allocation, and solves them respectively.

V. ALGORITHM DESIGN

A. TASK OFFLOADING DECISION

Due to the heterogeneity of vehicle tasks and edge nodes, the complexity of tasks may exceed the computing power of edge nodes, resulting in insufficient utilization of resources. In order to solve this problem, this paper first pre-segments the task to improve the utilization of edge resources. Suppose that TaV m has only one task r at first, the specific segmentation process is expressed as the following steps:

1) Calculate the segmentation flag ω. An edge node i (i.e., $a_{ri} = 1$ or $b_r = 1$) is randomly selected for the current task r, and the expected offloading delay T_{pre} of the task is calculated. Then, according to T_{pre} and the maximum tolerable delay T_m^d of the task, the segmentation flag ω is calculated. The calculation method can be formulated as:

$$T_{pre} = T_r^{off} + T_r^{cal} \tag{25}$$

$$\omega = \frac{T_{pre}}{T_m^d} \tag{26}$$

2) Average task segmentation. If $\omega \ge \delta$, the task is divided equally, and the calculation process is shown in formulas (27) and (28). If $\omega < \delta$, the task is not segmented.

$$D_{r_v} = D_r^a + \frac{D_r^b}{2} \tag{27}$$

$$C_{r_v} = \frac{C_r}{2} \tag{28}$$

3) Recursive subtask. Traverse each subtask r_v, repeat steps 1) and 2) until the termination condition is satisfied.

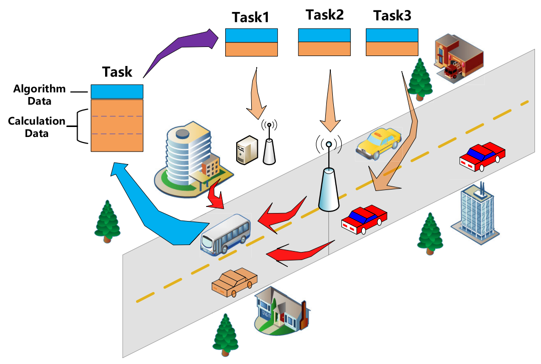

FIGURE 2. **Task offloading procedure.**

Among them, δ is the segmentation threshold, which can be determined according to the experiment.

Matching algorithm is an effective method to solve the problem of task offloading. This kind of algorithm has the advantages of low complexity, strong scalability and strong fairness [24], [25]. It can comprehensively consider multiple factors, including delay, energy consumption, and task urgency, to achieve overall optimization. This multidimensional consideration gives the matching algorithm a significant advantage in resource utilization efficiency. Additionally, compared to common methods like linear programming and ant colony algorithms, the matching algorithm has lower computational complexity, making it suitable for highly dynamic scenarios like vehicular networks. It can also handle competitive and cooperative relationships among users, ensuring fairness in resource allocation. Deferred acceptance (DA) algorithm is one of two-sided matching algorithm, this paper uses DA algorithm to solve the optimal matching set on the basis of comprehensively considering the user's offloading requirements and node performance differences.

The DA algorithm belongs to the one-to-one matching algorithm, but in the vehicle task offloading scenario, the number of tasks is much larger than the number of nodes. In order to solve this problem, this paper proposes a multi-round DA algorithm to realize the many-to-one matching strategy. Specifically, in the initial matching, the requester will think that the service providers are all idle nodes, and the maximum resource is the expected resource, and the two-way matching is performed according to the initial preference. If the request is rejected, the user will update the preference size of the candidate node according to the matching result of the previous round, and perform a new round of matching. By repeating the above process until all tasks are matched. The specific task segmentation and offloading process is shown in Figure 2.

In order to accurately describe the degree of demand for computing resources by the task generator, the preference values of the subtask r ($r \in \mathcal{R}_m$) for CCN n, RSU and Cloud are defined as $P_r(n)$, $P_r(RSU)$ and $P_r(Cloud)$, respectively. For users, the more computing resources are expected to be allocated and the shorter the distance from the vehicle during communication, the more users prefer the node. Due to the high transmission delay and cost of the cloud server, $P_r(Cloud)$ is always defined as a minimum value, so the following definition can be obtained:

$$P_r(n) = f_{nr} - \beta\left(d_{mn}(0) + d_{mn}\left(T_{rn}^{com}\right)\right) \tag{29}$$

$$P_r\left(RSU\right) = f_{Rr} - \beta\left(d_{mR}(0) + d_{mR}\left(T_{rR}^{com}\right)\right) \tag{30}$$

$$P_r\left(Cloud\right) = min\left\{P_r\left(RSU\right), P_r(n)|\forall n \in \mathcal{N}\right\} \tag{31}$$

where β is the weight coefficient of distance, and the relative distance between users and nodes is used to measure the quality of communication.

The service provider will conduct a comprehensive evaluation of the incoming offloading request from two aspects: task level and task complexity. For tasks at different levels, nodes will preferentially accept tasks with high urgency to ensure the resource requirements of critical tasks. For tasks at the same level, nodes will preferentially accept more complex tasks to improve resource utilization. Therefore, the preference calculation methods of CCN n, RSU and Cloud for subtask r are shown in (32)-(34).

$$P_n(r) = \theta_m + \xi\left(\varphi_n^{com} D_r + \varphi_n^{cmp} C_r\right) \tag{32}$$

$$P_{RSU}(r) = \theta_m + \xi\left(\varphi_R^{com} D_r + \varphi_R^{cmp} C_r\right) \tag{33}$$

$$P_{Cloud}(r) = \mathrm{K} \tag{34}$$

where ξ is the adjustment coefficient, and the influence of the second part is adjusted by setting a smaller value. The cloud server has unlimited resources and will accept all incoming tasks, so a constant K is used to represent the preference value for the task.

To ensure that $T_r^{total} \leq T_m^d$ and the resource constraints C3-C4 of P1, the edge node cannot accept too many tasks and needs to meet certain computing resource requirements. To achieve this goal, we defined constraints (35) and (36).

$$\sum_{m \in M} \sum_{r \in \mathcal{R}_m} a_{rn} \frac{C_r}{T_m^d - \hat{t_r}} < F_n, \quad n \in \mathcal{N} \tag{35}$$

$$\sum_{m \in M} \sum_{r \in \mathcal{R}_m} b_r \frac{C_r}{T_m^d - \hat{t_r}} < F_R \tag{36}$$

where $\hat{t_r}$ is the expected transmission delay of task r, which is calculated according to the average bandwidth allocation method. At the same time, in order to avoid idle nodes, each node is allowed to accept at least one task. Based on the above definition, the improved matching algorithm for task offloading (MAT) proposed in this paper is divided into seven steps, which are described as follows:

1) Initialization. The TaVs set $\mathcal{M}$, CCNs set $\mathcal{N}$, RSU and Cloud are defined, and the pre-allocation resources $f_{Rr} = F_R$, $B_{Rr} = B_R$ of RSU for r and $f_{nr} = F_n$, $B_{nr} = B_n$ of CCN n for r are defined.

2) Task segmentation. Adaptive task segmentation is performed on any TaV m to obtain the corresponding sub-task

set $\mathcal{R}_m$. Define the set of rejected subtasks $\mathcal{Q} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \ldots \cup \mathcal{R}_M$.

3) Positive preference. By traversing the $\mathcal{Q}$ subtask, we construct a preference list List1 for Cloud, RSU and CCNs satisfying $\int_0^{\tau_{mn}} R_{rn}(t)dt \geq D_r$ (Corresponding to the constraint C2 of P1), and calculate the corresponding ranking according to the preference.

4) Reverse preference. For each CCN, RSU, and Cloud, a reverse preference list List2 for subtask r is constructed, and r is ranked according to the preference value.

5) Send requests. By traversing the set $\mathcal{Q}$, each subtask sends an offloading request to the most preferred node (Corresponding to the constraint C1 of P1).

6) Accept the request. By clearing the set $\mathcal{Q}$, the service party will accept the tasks according to the task offloading request in the order of preference value from high to low, and reject the tasks that cannot meet the resource constraints (35) and (36). If the load of the node is full and there are subsequent offloading requests with higher preference values, the accepted low preference value requests are replaced.

7) Preference update. All rejected tasks are added to the set $\mathcal{Q}$, if $\mathcal{Q} = \varnothing$, the algorithm ends; otherwise, according to the formulas (37) and (38), the expected resources of CCNs for $r \in \mathcal{Q}$ are updated (f_{Rr} and B_{Rr} are updated in the same way), and then List1 is reconstructed for the candidate nodes of r, back to step 5).

$$f_{nr} = F_n - \sum_{m' \in M} \sum_{r' \in \mathcal{R}_{m'}} \frac{a_{r'n} C_{r'}}{T_{m'}^{d} - \hat{t_{r'}}} \tag{37}$$

$$B_{nr} = \frac{B_n}{1 + \sum\limits_{m' \in M} \sum\limits_{r' \in \mathcal{R}_{m'}} a_{r'n}} \tag{38}$$

Since the task will not send repeated requests to the node, and the cloud server will accept all incoming tasks, the algorithm will eventually converge. The complexity of the algorithm increases with the number of subtasks and the number of service nodes. In the worst case, the complexity of the algorithm is $O(|\mathcal{Q}| \cdot |\mathcal{N}|)$.

B. RESOURCE ALLOCATION

After completing the offloading decision, it is necessary to allocate an appropriate number of computing and communication resources for each task to further optimize the delay and cost. In this paper, the resource allocation problem is divided into two cases: cloud-side resource allocation and edge-side resource allocation, and solved separately.

1) If only the resource allocation on the cloud side is considered, the optimization problem can be expressed as:

$$\begin{aligned} P2: \min_{f_{cr}} (1-\mu)\, & \frac{\theta_m C_r}{f_{cr}} + \mu \varphi_c^{cmp} f_{cr} C_r \\ s.t.\ & f_{cr} \in R^{+}, \quad \forall r \in \mathcal{R}_m \end{aligned} \tag{39}$$

The problem is a convex optimization problem. Since the most cloud side tasks are non urgent and have high task offloading costs, the optimal resource allocation solution can be obtained by $f_{cr}^{*} = \sqrt{\frac{\theta_m (1-\mu)}{\mu \varphi_c^{cmp}}}$.

2) If only the edge-side resource allocation is considered, any edge node can be expressed as $i (i \in \mathcal{H})$, where $\mathcal{H} = \mathcal{N} \cup \{RSU\}$, then the optimization objective of node i can be expressed as V_i:

$$V_i = \sum_{m \in \mathcal{M}} \sum_{r \in \mathcal{R}_m} a_{ri} \left[\mu Cost_r^{total} + \theta_m (1-\mu) T_r^{total} \right] \tag{40}$$

When task r is offloaded to RSU, there is $a_{rRSU} = 1$ and $b_r = 1$. The optimization problem at this time can be reconstructed into the following form:

$$\begin{aligned} P3: & \min_{B_{ir},\ f_{ir}} \sum_{\forall i \in \mathcal{H}} V_i \\ s.t.\ & C1: \sum_{m \in \mathcal{M}} \sum_{r \in \mathcal{R}_m} a_{ri} f_{ir} \leq F_i \\ & C2: \sum_{m \in \mathcal{M}} \sum_{r \in \mathcal{R}_m} a_{ri} B_{ir} \leq B_i \end{aligned} \tag{41}$$

This problem has high computational complexity and no explicit analytical solution. Genetic algorithm has the advantages of strong flexibility and easy implementation. It is a feasible method to solve this kind of problem. However, the disadvantage is that the calculation is time-consuming and easy to fall into local optimum.

In order to solve this problem, this paper uses chaotic mapping to select crossover and mutation points [26]. Compared with the completely random selection method, the chaotic mapping has the combined characteristics of randomness and certainty in mathematics, and can generate a numerical sequence that is evenly distributed in a specific interval. In this way, the genetic algorithm can cover all possible solutions more evenly when searching the solution space, which is helpful to jump out of the local optimal solution. In addition, in order to accelerate the convergence speed of the algorithm, this paper uses the opposition-based learning strategy to improve the quality of the initial population [27]. This strategy can generate two individuals with symmetrical values according to the interval range. One of the two individuals must be closer to the global optimum. Therefore, the initial population after optimization can be obtained by eliminating the worse one and retaining the better one. Compared with the completely randomly generated population, this strategy can make the algorithm converge faster.

Therefore, this paper uses opposition-based learning and chaotic mapping to improve its search ability and convergence speed. Since the resource allocation between nodes is not related, it can be considered to solve the resource allocation problem separately for each node, so that the search space is smaller and the complexity of problem solving can be reduced.

Assuming that each channel is transmitted in OFDM mode, the bandwidth resources to be allocated can be composed of multiple sub-bandwidths, and CPU resources can be allocated according to the time slice. Therefore, the optional bandwidth allocation ratio is defined as $B_{ij} \in \{B_1, B_2, \ldots, B_l\}$, and the allocated computing resource ratio is $F_{ij} \in \{F_1, F_2, \ldots, F_c\}$. The value range of each element is in the interval (a, b) (this paper assumes $a = 0, b = 1$). In the process of resource allocation, it is necessary to optimize the global delay and cost, and ensure the success rate of the task. Therefore, the fitness function of node i can be expressed as follows:

$$J_i = \frac{1}{V_i}\left[1 + \sum_{m\in\mathcal{M}} \sum_{r\in\mathcal{R}_m} a_{ri}\theta_m \cdot \chi\left(T_r^{total} \leq T_m^d\right)\right] \tag{42}$$

Among them, χ is the indicator function, when the discriminant is true, the value is 1, otherwise it is 0. When the target value is smaller and the number of urgent tasks completed is larger, the individual's fitness is correspondingly larger. The iterative process of the algorithm mainly includes six steps: coding, population optimization, selection, chaotic crossover, chaotic mutation and elimination, as shown in Figure 3. The specific process of improved genetic algorithm (IGA) is as follows:

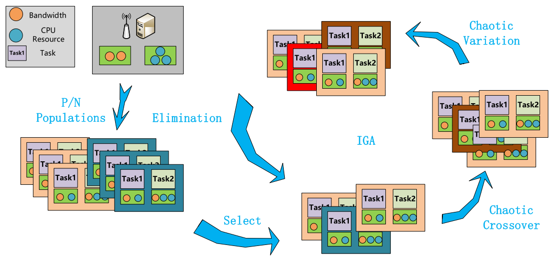

FIGURE 3. IGA algorithm procedure.

1) Coding. Individual $X = \{x_1, x_2, \ldots, x_k\}$ is defined to represent a resource allocation solution of node i. Here, any element $x_q \in X$ is represented in the form of a binary string, which means the number of resource allocations of the node i for the $\left\lfloor \frac{(q-1)}{2} \right\rfloor + 1$st tasks. If $q\%2 = 0$ is satisfied, then x_q represents the solution of bandwidth allocation, otherwise, it represents the solution of computing resource allocation.

2) Population optimization. In order to optimize the quality of the initial population, the reverse individual $X' = \{x_1', x_2', \ldots, x_k'\}$ corresponding to X is obtained by using the opposition-based method, and any element $x_q' = a + b - x_q$. According to the size of the fitness, the optimal one is retained as a member of the initial population in a pair of positive and negative individuals.

3) Selection. On the basis of the initial population, the selected probability is obtained according to the fitness of each individual, and a part is selected as an excellent individual by the roulette method.

4) Chaotic crossover. Through the given hybridization probability, some chromosome fragments of each individual are exchanged. When the probability hits, the individual is cross-operated, and logistics mapping is used to select the crossover point. Specifically, firstly, the initial value $o_n \in (0, 1)$ is selected, and the chaotic sequence is obtained according to the formula $o_{n+1} = 4o_n(1 - o_n)$. Then, the sequence is mapped to the chromosome gene space according to $z = \lfloor o_{n+1} \rfloor * |X|$, so that the crossover occurs at this position to form a new offspring. Only by replacing some point genes, the change is small, which can avoid the problem of optimal chattering in the process of generating offspring.

5) Chaotic variation. Through the given mutation probability, some gene values are randomly changed from 1 to 0, or from 0 to 1. Similar to the previous step, the chaotic sequence is also used to obtain the mutation point and change the value at the point.

6) Elimination. In order to meet the resource allocation constraints, the individuals beyond the sum of resources are eliminated, and the optimal individuals in this cycle are recorded. Due to the timeliness requirements of the vehicle network, it is assumed that the algorithm iteration process is terminated when a certain number of cycles is satisfied, and the optimal individual is used as an approximate solution for resource allocation.

C. GLOBAL ALGORITHM

In this paper, task offloading decision and resource allocation are modeled as mixed integer nonlinear programming problems. A task offloading scheme based on MAT and IGA is proposed and named as MAT-IGA. In this paper, the problem is divided into two sub-problems: offloading decision and resource allocation. Firstly, the original task is divided by adaptive task segmentation algorithm, and the task offloading decision is obtained by using the improved matching algorithm. Then the task offloading decision is input into the original problem, and the resource allocation problem is solved by applying IGA algorithm to each node. The specific steps are as follows:

1) Initialize the task set of TaVs and the starting resources of CCNs, RSU and Cloud.

2) Roadside unit RSU collects environmental data and uses MAT algorithm to obtain task offloading decision E.

3) According to the decision result E, the IGA algorithm is applied to the edge-side CCNs and RSU, and the convex optimization method is applied to Cloud to obtain the resource allocation scheme S.

4) RSU publishes task offloading schemes E and S, and all nodes offload or calculate tasks according to the specified scheme.

5) Each node returns the calculation results to the task vehicle TaVs.

In summary, the entire process of the MAT-IGA algorithm is shown in Figure 4 and Algorithm 1.

VI. SIMULATION DESIGN

A. PARAMETER SETTING

This paper uses Python 3.8 for simulation verification on Windows 10 systems. Consider a road scenario where there

Algorithm 1 MAT-IGA Algorithm

```
Input:
    The task vehicles $\mathcal{M}$;
    The crowd computing nodes $\mathcal{N}$;
    The roadside unit $RSU$;
Output:
    Task offloading decision $E$;
    Resource allocation schema $S$;
1:  RSU collects environmental information including task
    vehicle information and service node information;
2:  RSU uses MAT algorithm to solve $E$;
3:  $\mathcal{H} = \mathcal{N} \cup \{RSU\}$;
4:  $S = \emptyset$
5:  for each $i \in \mathcal{H}$ do
6:      RSU uses IGA algorithm to solve the resource
        allocation schema $s_i$ of node $i$;
7:      $S$.append($s_i$);
8:  end for
9:  RSU publishes task offloading strategy $\{E, S\}$;
10: for each $j \in \mathcal{M}$ do
11:     TaV $j$ offloads the task to the corresponding CCN;
12: end for
13: for each $z \in \mathcal{H}$ do
14:     Node $z$ processes the received task according to
        scheme $S$;
15:     Node $z$ returns the calculation result of the task to the
        corresponding TaV;
16: end for
```

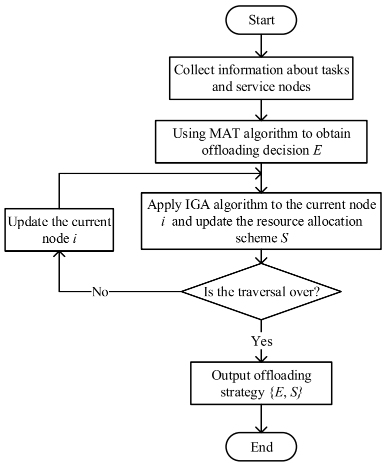
Start
Collect information about tasks and service nodes
Using MAT algorithm to obtain offloading decision E
Apply IGA algorithm to the current node i and update the resource allocation scheme S
Update the current node i
Is the traversal over?
No
Yes
Output offloading strategy {E, S}
End

FIGURE 4. **MAT-IGA algorithm flow chart.**

is a RSU with computing and communication capabilities, a cloud server, and different numbers of TaVs and CCNs. Each TaV has a randomly generated in-vehicle task, and each service node has a different number of resources and unit costs. It is assumed that the number of TaVs is between [20,25], the number of SeVs is between [8,13], the RSU communication radius is 300 m, the number of roadside APs is between [3,6], and the distribution interval is between [10,70]m. The simulation scenario is shown in Figure 1.

In order to ensure the reliability of the experimental results, the initial position and speed of the vehicle will change randomly in each iteration period. At the same time, in order to eliminate randomness, each experiment carries out 100 iterations in different scenarios. At each iteration, the vehicle position, vehicle speed and other characteristics are changed, and the target cumulative value of the optimization problem P1 is taken as the final result. The target cumulative value can be defined as the sum of the target value for each iteration. The relevant parameter settings are shown in Table 1.

TABLE 1. **Units for magnetic properties.**

Symbol	Description	Value
α	path loss exponent	4.0
μ	cost weight	0.5
σ^2	noise power	-104 dBm
B_n	node bandwidth	10-20 MHz
B_R	RSU channel bandwidth	20 MHz
R	V2V maximum transmission range	200 m
V_m^T	Vehicle speed	$\pm$5-10 m/s
D_m^a	Task algorithm data	50-100 KB
T_m^d	Task delay requirements	0.2-1.0 s
D_m^b	Task calculation data	0.5-3.0 MB
C_m	Task computational complexity	1.0-3.5 Gcycles
θ_m	Task urgency	$\{1, 2, 3\}$
F_n	CCN computing power	4-8 GHz
F_R	RSU computing power	12 GHz

In order to verify the effect of the MAT-IGA algorithm proposed in this paper, four benchmark schemes are selected for simulation comparison:

1) MAT-AVG scheme: The MAT algorithm is used to segment and offload tasks, and the communication and computing resources are evenly allocated to each task.

2) DIS-IGA scheme: Taking distance as the main reference factor, the sub-task is preferentially offloaded to the node closest to the current vehicle. If the load of the offloaded node is full, it is offloaded to the next nearest node. Finally, the IGA algorithm is used to allocate resources.

3) SMRETO scheme [28]: A partial offloading algorithm based on bilateral matching. This method uses task deadline, channel rate as evaluation indicators to make decisions, but does not consider the importance of tasks and cloud server resources.

4) SA-PSO scheme [29]: An improved particle swarm optimization algorithm based on the Metropolis Criterion. This method considers the residence time of the vehicle and uses the improved particle swarm optimization algorithm to search the offloading strategy, but does not consider task segmentation and node collaboration.

B. SIMULATION ANALYSIS

Due to the difference in the number of resources and the unit cost of nodes, too much task segmentation will lead to increased transmission delay and cost, and too little task segmentation will lead to insufficient resource utilization. Therefore, this paper first compares the effects of MAT-IGA algorithm under different δ parameters, as shown in Figure 5. In the scenario of this paper, with the increase of δ, the target value decreases first and then increases. When δ is 0.8, the best global optimization effect can be achieved.

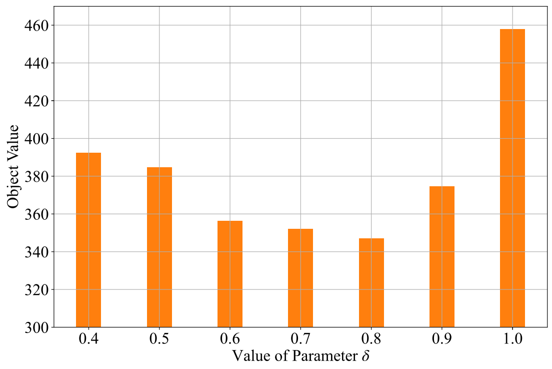

FIGURE 5. The effect of MAT-IGA under different δ.

For the IGA algorithm, too large population size will increase the amount of calculation, too small can not provide enough sampling points, easy to fall into local optimum, so this paper compares the effects of the algorithm under different population sizes, as shown in Figure 6.

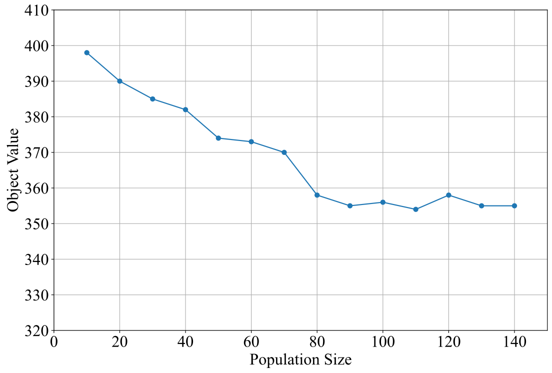

FIGURE 6. Comparison of IGA under different populations.

The simulation results show that when the initial population size is 90, the target value can reach the minimum. Continuing to increase the population size does not significantly improve the optimization effect, but will lead to an increase in computational complexity. In order to verify that the improved IGA is better than the original genetic algorithm, this paper compares the differences between the four algorithms without any improvement (IGA-NoALL), only using opposition-based learning (IGA-NoChaos), only using chaotic mapping (IGA-NoOBL) and the proposed algorithm (IGA), as shown in Figure 7.

It can be found that the lack of any improvement strategy will lead to a decrease in the convergence speed or optimization effect of the algorithm. The opposition-based learning and chaotic mapping strategies have a certain effect

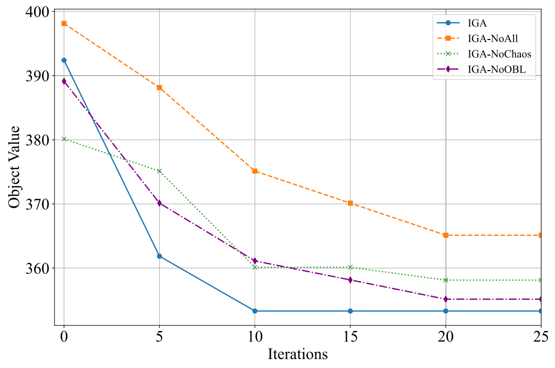

FIGURE 7. The convergence effect of different genetic algorithms.

on improving the convergence speed and final optimization effect of the algorithm.

In order to adjust the influence of delay and cost, this paper defines the parameter μ. However, the setting of μ value will affect the procession of resource allocation. Therefore, in order to explore the influence of this parameter on the performance of the algorithm, this paper compares the change trend of the delay and cost of task offloading and the average task success rate under different μ, as shown in Figure 8 and Figure 9.

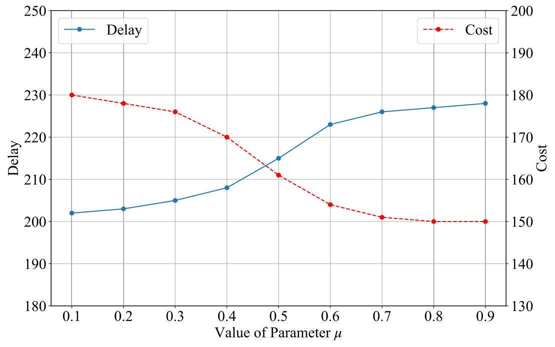

FIGURE 8. The effect of MAT-IGA under different μ.

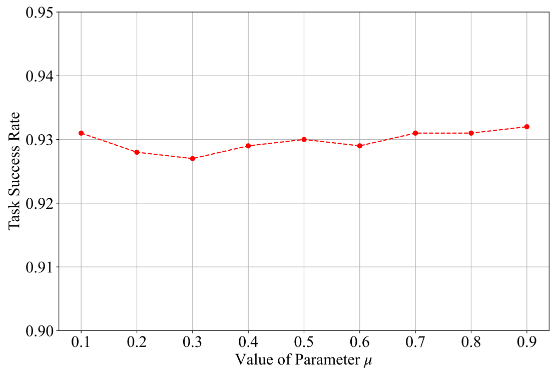

FIGURE 9. The success rate of MAT-IGA under different μ.

From the simulation results, it can be seen that with the increase of μ, the delay increases gradually, the cost decreases gradually, and the success rate of task offloading is almost unchanged. The reason is that when the cost weight μ increases, the more resources the node allocates to the task, the higher the offloading cost of the task. Therefore, nodes tend to allocate fewer resources to tasks, but at the same time, due to the definition of fitness function, nodes must meet the delay requirements of receiving tasks. Therefore, the node ultimately only allocates the necessary number of resources to the task. When the cost weight μ is small, the task offloading delay gradually becomes the main component of the optimization goal. At this time, the node will allocate as much resources as possible to reduce the total processing delay of the task, so as to obtain a smaller object value. However, due to the limited resources of nodes, the optimization degree of delay will gradually approach a limit.

Since the number of vehicles in the actual scene often changes greatly, in order to verify the optimization effect of the algorithm in different scenarios, this paper changes the number of TaVs and SeVs and performs simulation tests. The optimization effect of different task offloading schemes is shown in Figure 10.

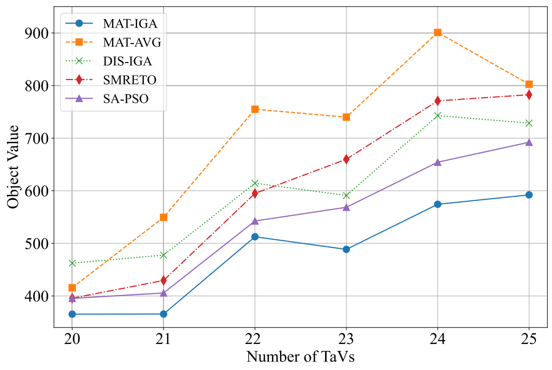

FIGURE 10. Comparison under different TaVs.

It can be found from Figure 10 that due to the heterogeneity of tasks, the curves have certain fluctuations. However, as the number of TaVs increases, the target values of all algorithms show an upward trend. The reason is that the increase in the number of tasks leads to an increase in latency and cost. In Figure 11, as the number of service vehicles increases, the target values of most algorithms show a downward trend, because the increase of crowdsourcing nodes can provide more computing resources.

From the perspective of optimization effect, this method can achieve the lowest delay and cost loss under different vehicle numbers. Since SA-PSO algorithm does not use task segmentation to reduce task complexity, it is slightly worse than MAT-IGA in resource utilization. The SMRETO algorithm considers the deadline of the task and uses the matching strategy to allocate the task. However, the algorithm ignores the difference in the urgency of the task, which leads to the lack of delay optimization of important tasks, and then makes the global optimization effect worse. It can be found

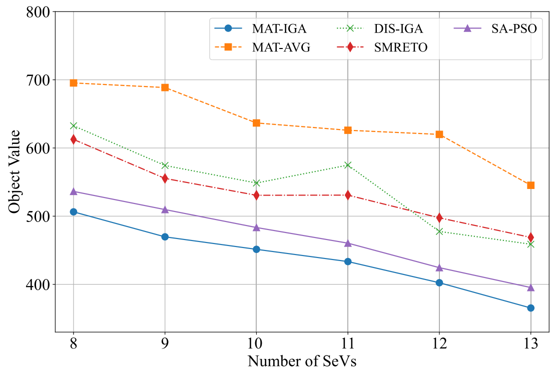

FIGURE 11. Comparison under different SeVs.

from Figure 11 that the optimization effect of the algorithm is not as good as that of SA-PSO, and it is close to DIS-IGA as a whole.

Considering the impact of task characteristics on the algorithm, this paper conducts simulation tests under different task data volume and computational complexity, and compares the optimization effects of each scheme. The simulation results are shown in Figure 12 and Figure 13.

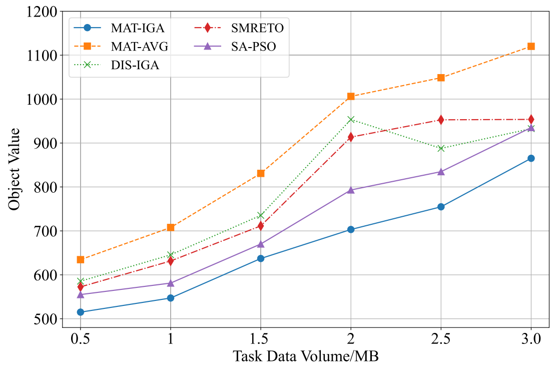

FIGURE 12. Comparison under different task data.

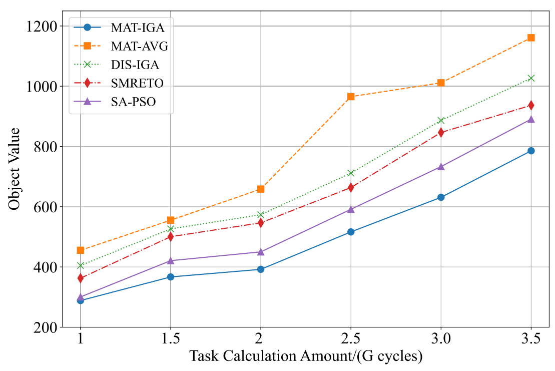

FIGURE 13. Comparison under different task computation amount.

Figure 12 and Figure 13 shows the comparison results of the algorithms under different task characteristics. As the complexity of the task increases, the transmission delay and calculation delay of the task will increase, so the target values of all algorithms are on the rise. Similarly, with the increase of computational complexity, the target value also shows an upward trend. However, in contrast, the MAT-IGA algorithm still maintains the lowest delay and cost.

In addition to the comparison of the object value, the task success rate is also an important indicator to measure the advantages and disadvantages of the algorithm. Therefore, this paper compares the average task success rate of three levels of tasks under different schemes, as shown in Figure 14. Since the MAT-AVG algorithm uses the average resource allocation method and does not consider the delay bound of the task, it has the worst effect in terms of task success rate. The MAT-IGA algorithm can give priority to the resource requirements of emergency tasks. Therefore, for tasks with high urgency, the offloading success rate is significantly higher than that of other methods, but for tasks with the lowest level, the offloading success rate is slightly worse than that of SA-PSO algorithm.

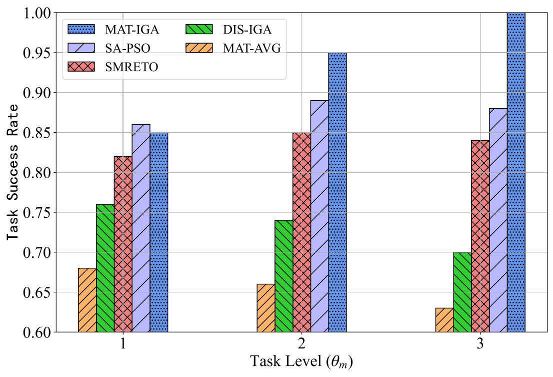

FIGURE 14. Success rate of different levels of tasks.

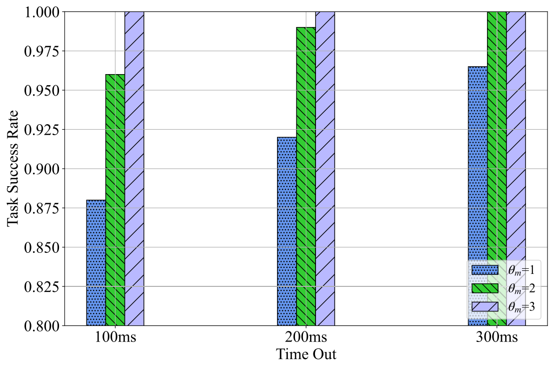

FIGURE 15. Task success rate under a small amount of timeout.

Figure 15 compares the changes in the task success rate index of the MAT-IGA algorithm when a small amount of timeout is allowed. It can be seen that with the reduction of delay requirements, the success rate of all tasks is increasing, and the low-level tasks change the most. In the actual scenario, for some common types of tasks, users often do not have strict delay requirements. Therefore, this method can prioritize the resource requirements of emergency tasks while completing secondary tasks as much as possible, thereby reducing the computational burden of vehicles.

VII. CONCLUSION

This paper proposes a vehicular task offloading and resource allocation scheme based on mobile edge computing. Based on the three-tier computing offloading framework of MEC, this scheme fully considers the dynamics of environmental resources, especially the task offloading problem in resource-constrained scenarios. Firstly, this paper uses the adaptive task segmentation mechanism to divide the tasks to adapt to the computing power in the region and improve the resource utilization. Then, combined with the improved DA algorithm, the preference value is defined by the factors such as task urgency, relative distance and communication delay, and the optimal matching set is solved by the two-way selection of users and service methods. Finally, the chaotic genetic algorithm is used to optimize resource allocation. The simulation results show that the proposed method is superior to the common baseline algorithm under different task characteristics and vehicle numbers. Therefore, the method proposed in this paper makes up for some shortcomings in the research of task offloading to a certain extent, and also promotes the further development of the Internet of Vehicles.

REFERENCES

[1] Z. Li, Q. Wang, and Y. Chen, "A survey on task offloading research in vehicular edge computing," *Chin. J. Comput.*, vol. 44, no. 5, pp. 963–982, 2021.

[2] M. Li, J. Gao, N. Zhang, L. Zhao, and X. Shen, "Collaborative computing in vehicular networks: A deep reinforcement learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2016, *arXiv:1509.02971*.

[4] A. Mahmood, Y. Hong, M. K. Ehsan, and S. Mumtaz, "Optimal resource allocation and task segmentation in IoT enabled mobile edge cloud," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13294–13303, Dec. 2021.

[5] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 188–208, Dec. 2020.

[6] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1928–1937.

[8] H. Xiao, C. Xu, Y. Ma, S. Yang, L. Zhong, and G.-M. Muntean, "Edge intelligence: A computational task offloading scheme for dependent IoT application," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7222–7237, Sep. 2022.

[9] X. Pang, Z. Wang, J. Li, R. Zhou, J. Ren, and Z. Li, "Towards online privacy-preserving computation offloading in mobile edge computing," in *Proc. IEEE INFOCOM*, Jun. 2022, pp. 1179–1188.

[10] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[11] C. Fang, Z. Hu, X. Meng, S. Tu, Z. Wang, D. Zeng, W. Ni, S. Guo, and Z. Han, "DRL-driven joint task offloading and resource allocation for energy-efficient content delivery in cloud-edge cooperation networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16195–16207, Dec. 2023.

[12] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

[13] T. H. Binh, D. B. Son, H. K. Vo, B. M. Nguyen, and H. T. T. Binh, "Reinforcement learning for optimizing delay-sensitive task offloading in vehicular edge-cloud computing," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2058–2069, Sep. 2024.

[14] H. Guo, X. Chen, X. Zhou, and J. Liu, "Trusted and efficient task offloading in vehicular edge computing networks," *IEEE Trans. Cogn. Commun. Netw.*, early access, Jun. 11, 2024, doi: 10.1109/TCCN.2024.3412394.

[15] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.

[16] J. Klaimi, S. Senouci, and M. A. Messous, "Theoretical game approach for mobile users resource management in a vehicular fog computing environment," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Limassol, Cyprus, Jun. 2018, pp. 452–457.

[17] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107108.

[18] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[19] W. Fan, Y. Zhang, G. Zhou, and Y. Liu, "Deep reinforcement learning-based task offloading for vehicular edge computing with flexible RSU-RSU cooperation," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 7712–7725, Jul. 2024.

[20] L. Yin, J. Luo, C. Qiu, C. Wang, and Y. Qiao, "Joint task offloading and resources allocation for hybrid vehicle edge computing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 8, pp. 10355–10368, Aug. 2024.

[21] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021.

[22] K. Li, X. Wang, Q. He, B. Yi, A. Morichetta, and M. Huang, "Cooperative multiagent deep reinforcement learning for computation offloading: A mobile network operator perspective," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24161–24173, Dec. 2022.

[23] Y. Guo, D. Ma, H. She, G. Gui, C. Yuen, H. Sari, and F. Adachi, "Deep deterministic policy gradient-based intelligent task offloading for vehicular computing with priority experience playback," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 10655–10667, Jul. 2024.

[24] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5G-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1352–1361, Feb. 2020.

[25] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, A. Aziz, and Z. Han, "RL/DRL meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8315–8338, Jun. 2022.

[26] J. Zhang, J. Wang, R. Dong, X. Qiao, and S. Shen, "Optimization method for remote inspection path planning of secondary equipment based on chaotic genetic algorithm," in *Proc. 5th Int. Symp. Robot. Intell. Manuf. Technol. (ISRIMT)*, 2023, pp. 192–195.

[27] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Comput. Math. Appl.*, vol. 53, no. 10, pp. 1605–1614, May 2007.

[28] U. M. Malik, M. A. Javed, J. Frnda, and J. Nedoma, "SMRETO: Stable matching for reliable and efficient task offloading in fog-enabled IoT networks," *IEEE Access*, vol. 10, pp. 111579–111590, 2022.

[29] M. R. Raju, S. K. Mothku, and M. K. Somesula, "MITS: Mobility-aware intelligent task scheduling in vehicular fog networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 3079–3093, Mar. 2024.

LONG ZHANG received the bachelor's degree in civil engineering and the master's degree in bridge and tunnel engineering from Shijiazhuang Railway University, in 2011 and 2014, respectively. He is currently with the Minister of Test and Development, Intelligent Transportation Research and Development Center, Hebei Provincial Communications Planning, Design and Research Institute Company Ltd. A total of eight invention patents, three landmarks in Hebei, one EI article, and 11 software works were published. His research interests include intelligent transportation, 5G, Beidou, large model, and other information application fields. His awards and honors include four certificates of scientific and technological achievements in Hebei, all of which have reached the international advanced level. The provincial and ministerial awards include the first prize of the Science and Technology Award of China Highway Society, in 2023, the bronze award of the third national highway 'micro innovation' competition, and the second prize of the digital scene innovation professional competition of state-owned enterprises.

RUI CAO received the bachelor's degree from Nanchang University, in 2021. He is currently pursuing the master's degree with Hebei University of Technology. His research interests include edge computing and machine learning.

WEI LEI received the bachelor's degree from Hebei University of Technology, in 1993. He is currently the Chief Engineer with Hebei Provincial Communications Planning, Design and Research Institute Company Ltd. A total of 19 patents and articles have been published. His research interests include intelligent transportation, intelligent monitoring, and early warning of infrastructure groups. His awards and honors include one first-class award of science and technology from China Highway Society, one special award of tunnel and underground space engineering innovation from China Highway Society, one third-class award of scientific and technological progress from China Highway Construction Industry Association, and two outstanding achievements of science and technology of Hebei Provincial Transportation Department.

YAZHOU WANG received the bachelor's degree from Chang'an University, in 2017, and the master's degree from Southeast University, in 2020. He is currently an Intelligent Transportation Research, Development, and Test Engineer with Hebei Provincial Communications Planning, Design and Research Institute Company Ltd. He has published a total of ten patents. His research interests include traffic flow simulation deduction technology research, road traffic conflict risk identification and early warning, intelligent transportation key technology research and development, and testing.

ZHENLONG XIE received the bachelor's degree in civil engineering from Hebei University of Technology, in 2007, and the master's degree in road and railway engineering from Hebei University of Technology, in 2010. He is currently the Director of the Intelligent Transportation Engineering and Operation and Maintenance Department, Jingxiong Cloud Control (Beijing) Intelligent Transportation Technology Company Ltd. His research interests include the development and application of traffic cloud control platform, digital twin technology, integrated command and dispatch technology, and road risk early warning technology. The awards and honors obtained him include the honors of advanced individuals in the labor competition of the key construction projects of Beijing-Tianjin-Hebei transportation integration (Hebei competition area), in 2023, and the traffic support workers of the 2022 Beijing Winter Olympics. He has won a total of 14 first-class, second-class, and third-class awards for the scientific and technological achievements of Hebei Science and Technology Department and Transportation Department. He has published five articles, eight patents, and the formulation of participation standards and norms.

BINGXIN NIU received the Ph.D. degree from Dalian University of Technology, in 2019. He is currently with Hebei University of Technology. His main research interests include wireless sensor networks, the Internet of Things, and mobile sensing. He is a member of the Chinese Computer Society (CCF).