Received December 17, 2021, accepted December 30, 2021, date of publication January 7, 2022, date of current version January 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141456

OPTOS: A Strategy of Online Pre-Filtering Task Offloading System in Vehicular Ad Hoc Networks

JUNJING HE[1], YUJIE WANG[1], XIN DU[1], ZHIHUI LU[1,2], (Member, IEEE), QIANG DUAN[3], (Senior Member, IEEE), AND JIE WU[1,4]

[1]School of Computer Science, Fudan University, Shanghai 200433, China
[2]Shanghai Blockchain Engineering Research Center, Shanghai 200433, China
[3]College of Information Sciences and Technology, The Pennsylvania State University, Abington, PA 19001, USA
[4]Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai 200433, China

Corresponding authors: Zhihui Lu (lzh@fudan.edu.cn) and Jie Wu (jwu@fudan.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1405000, in part by the National Natural Science Foundation of China under Grant 61873309 and Grant 92046024, and in part by the Shanghai Science and Technology Innovation Action Plan Project under Grant 19510710500 and Grant 18510732000.

ABSTRACT The advanced services provided by vehicle ad hoc networks (VANETs) often require vehicles to process complex computing tasks that may not be completed by individual vehicles within a required delay limit. Offloading tasks to road side units (RSUs) is a typical approach to enhancing service performance in VANET; however, RSUs may not always have sufficient resources for handling all task offloading requests. With the increasing amount of computing capacities available on vehicles, offloading tasks to other vehicles offers a promising alternative to RSU-based task offloading. However, vehicle-to-vehicle task offloading in VANET faces some new challenges that have not been fully addressed, among which is the degraded delay performance caused by vehicle mobility. In order to solve this problem, we propose an Online Pre-filtering Task Offloading System (OPTOS) that is able to mitigate the impact of vehicle mobility on task offloading performance. OPTOS comprises a process that selects candidate vehicles for hosting offloaded tasks and an HGSA algorithm that assigns tasks to vehicles for minimizing task completion delay while balancing utilization of computing capacities on different vehicles. We have conducted extensive experiments using a real-world dataset for evaluating the performance of the proposed OPTOS. Obtained results indicate that OPTOS is effective for reducing task completion delay and increasing task success rate in various VANET scenarios with different levels of vehicle mobility.

INDEX TERMS Task offloading, vehicular ad hoc network (VANET), vehicular-to-vehicular (V2V), vehicle mobility.

I. INTRODUCTION

In recent years, mobile vehicles are expected to provide drivers with more and more advanced services, including traffic accident analysis, traffic flow forecasting, route planning and traffic light management, etc [1]–[3]. These services are often composed of massive and complex computing tasks that cannot be completed by vehicles independently with an acceptable response delay. Vehicle ad hoc network (VANET) has been developed to provide communication connections between vehicles and infrastructures and also among vehicles [4], [5]. With the support of VANET, tasks can be offloaded to roadside units (RSU) with available computing capabilities to increase task processing speed and reduce response delay [6], [7]. However, with the increasing number of vehicles, the phenomenon of vehicle aggregation and computing task concentration occurs frequently [8]. Therefore, the strategy of offloading all computing tasks from vehicles to RSUs may cause some RSUs to be overloaded thus degrading the service delay performance.

In order to solve the problem of insufficient RSU computing resources, one possible solution is to deploy more RSUs and/or increase the computing capacity of each RSU, which might not be feasible due to the associated costs, especially in large-scale transportation systems. Another solution is to fully utilize the available computational resources on all vehicles by offloading tasks to other vehicles for processing [9]. When multiple vehicles are connected together via VANET, they can form a distributed "vehicular cloud" [10] in which each vehicle works as a computing

The associate editor coordinating the review of this manuscript and approving it for publication was Asad Waqar Malik.

node. In a vehicular cloud, computing tasks generated on one vehicle may be offloaded to other vehicles in the same cloud, which enables the idle resources on a group of vehicles to be fully utilized to reduce the overall task completion delay. The increasing computing capacities on vehicles together with rapid development in 5G wireless communication technologies [11] make inter-vehicle offloading an attractive alternative to RSU-based offloading.

The existing technologies for offloading tasks to other vehicles in a vehicular cloud mainly consider matching task requirements and vehicle resources [12], [13]. Although such offloading strategies achieve good results in stationary vehicular clouds, the impact of vehicle mobility has not been sufficiently addressed thus may cause unstable performance in some practical applications. Formed by a group of moving vehicles, a vehicular cloud is a dynamic structure that a vehicle may leave at any time (e.g., when the vehicle moves beyond the maximum communication scope of other vehicles in the cloud). All the uncompleted tasks offloaded to a left vehicle will have to be reassigned to other vehicles for processing, which prolongs the task completion delay. The main objective of our work is to investigate task offloading in a vehicular cloud for addressing this challenge introduced by vehicle mobility. The specific problem we want to solve in this paper is to find a task offloading strategy in a vehicular cloud that minimizes task completion delay with the negative impact of vehicle mobility eliminated.

In this paper, we first formulate vehicle-to-vehicle task offloading as an optimization problem based on our analysis on communication and computing delay in vehicular clouds. Then we design an online pre-filtered task offloading system (OPTOS) in order to obtain reliable task offloading in a vehicular cloud. The OPTOS addresses the impact of vehicle mobility by adding a step for identifying a set of candidate vehicles before making the offloading decision. We also develop a half-greedy simulated annealing (HGSA) algorithm that is employed in OPTOS for assigning tasks to candidate vehicles and minimizing the total task completion delay. More specifically, the main contributions of this article are summarized as follows.

1) We proposed OPTOS, an online pre-filtering task offloading system that fully considers the impact of vehicle mobility to obtain a reliable vehicle-to-vehicle offloading strategy. A key element of OPTOS is a step for filtering out vehicles that may leave the vehicular cloud before completing the tasks offloaded to them. This step ensures that tasks are only assigned to reliable vehicles that can successfully complete their tasks without being interrupted by mobility thus avoiding the extra delay in task completion caused by task reassignment.

2) We designed the HGSA algorithm to solve the task allocation problem to minimize task completion delay. The HGSA algorithm considers multiple factors, including task complexity, data volume, available resources on candidate vehicles, and inter-vehicle communication capacities, to make decisions on task-to-vehicle assignment for minimizing task completion delay while balancing task load across the candidate vehicles.

3) We conducted extensive experiments using a real vehicle data set to evaluate the performance of the proposed scheme. The experiment results indicate that our method can achieve reliable task offloading that increases the task success rate from 70% to near 100%. The obtained results also show that the proposed scheme can reduce task completion delay by about 19% compared to existing methods.

The remainder of this paper is organized as follows. Section II reviews the related work on task offloading. Section III describes the system model and gives the problem formulation. The structure of the OPTOS and the task offloading algorithm HGSA are presented in Section IV. Section V reports the evaluation experiments and analyzes the obtained results. The paper is concluded in Section VI.

II. RELATED WORK

In this section, we review state-of-the-art technologies for task offloading in vehicular networks.

Over the past decade, most research in this area focuses on offloading tasks to RSUs or cloud servers due to their rich computing and storage resources [14]–[18]. The optimization goals of the proposed methods include delay performance, energy consumption, and bandwidth utilization. The techniques for achieving the goals can be divided into two categories – mathematical optimization and artificial intelligence.

In the works based on mathematical optimization, task offloading is formulated as an optimization problem by defining an objective function with a set of constraints. Then the optimal solution is found using techniques including mixed-integer programming (MIP), heuristic algorithms, and game theory. Y. Wang *et al.* designed a payoff function and constructed a distributed optimal response algorithm for computing offloading games in [17]. It is proved that the offloading probability of each vehicle can converge to a unique equilibrium under certain conditions. C. Shu *et al.* [18] studied the offloading of dependent tasks to edge servers. Considering the dependency between subtasks and the competition among multiple edge servers, they proposed a new offloading scheme based on game theory to reduce the overall completion time of applications.

Some other studies are based on reinforcement learning [7], [19], [20]. H. Ke *et al.* [19] designed a heterogeneous vehicle network MEC system considering the changes in channel state and available bandwidth. They proposed an adaptive offloading method based on deep reinforcement learning (ACORL). ACORL can intelligently learn a policy to solve the trade-off between energy consumption, bandwidth allocation, and execution delay. Y. Liu *et al.* [20] regard the vehicles as mobile edge servers (VES) combined with fixed edge servers (FES) to provide computing services to users. They formulated task offloading and resource allocation as Markov processes and proposed two offloading strategies based on Q-Learning and deep reinforcement learning, respectively. Their scheme achieves better performance than pure VES or FES methods.

Although RSUs and cloud servers can provide high computing capacities, their deployments are often expensive. At the same time, the computing power of vehicles has been greatly improved in recent years, which enables vehicles to work as computing nodes for task processing. Therefore, offloading tasks to vehicles has started attracting research attention and interesting progress toward this direction has been reported in the literature.

M. LiWang *et al.* [21] proposed a randomized graph job allocation mechanism using hierarchical tree-based subgraph isomorphism with low complexity. However, the graph-based operation mechanism cannot adapt to the rapid changes in the topological structure of the Internet of Vehicles environment. At the same time, they did not consider the problem of imbalanced resource utilization of vehicle nodes. Based on game theory, J. Zhao *et al.* [22] designed a collaborative computation offloading and resource allocation optimization (CCORAO) scheme that can effectively improve system utilization and reduce computing time. J. Zhang *et al.* [6] proposed an approximate computing offloading scheme ALBOA that can utilize computing resources of the edge server more effectively and reduce the processing delay. Q. Qi *et al.* [23] proposed a knowledge-driven (KD) IoV service offloading decision framework, which provides an optimal policy directly from the environment. Deep reinforcement learning is used to obtain the optimal solution and continuous online learning is carried out during vehicle service execution to adapt to environmental changes. Y. Sun *et al.* [24] proposed an adaptive learning-based task offloading (ALTO) algorithm to minimize the average task completion delay. At the same time, they modified the existing multi-armed bandit (MAB) algorithm to input sensing and event sensing so that the ALTO algorithm can adapt to the dynamic vehicle task offloading environment. G. Qiao *et al.* [25] proposed a collaborative task offloading and output transmission mechanism that guarantees low latency as well as application-level performance.

However, the aforementioned works have not sufficiently considered the effect of vehicle mobility on task offloading to vehicles and reasonable allocation of vehicle resources. To the best of our knowledge, this article is the first effort to jointly consider the effect of vehicle mobility and the balanced utilization of the vehicle resource to offload tasks to vehicles.

III. SYSTEM MODELING AND PROBLEM FORMULATION

In this section, we first describe the system architecture for task offloading in a vehicular cloud and analyze the task completion delay, including communication delay and computing delay, in such a system. Then based on the delay analysis, we formulate the task offloading in a vehicular cloud as an optimization problem.

A. SYSTEM OVERVIEW

The high-level architecture of a smart transportation system with vehicular clouds is shown in Figure 1, which is composed of three layers: the cloud layer, the edge layer, and the vehicle layer. The cloud layer comprises data centers in the Internet. The cloud data centers have substantial computing capabilities but are located far from vehicles; therefore, they are often used for processing computing-intensive tasks that can tolerate long delays. The edge layer comprises RSUs located on the roadside, which have certain amounts of computational resources for completing lightweight tasks with shorter delays. RSUs are interconnected via the network infrastructure. The vehicle layer consists of vehicles clustered in vehicular clouds. Each vehicle in the system is assumed to have a certain amount of computing capacity for processing tasks that are either generated locally or offloaded from other vehicles. The vehicular ad hoc network provides direct communication connections among vehicles in the same vehicular cloud.

The task offloading process in this three-layer architecture can be described as follows.

1) The basic information of vehicles, such as their current positions, speeds, moving directions, and available computing capacities, are collected and sent to RSUs. Such information may be provided by vehicles and/or measured by road-side sensors.

2) RSUs receive and process the vehicle information to maintain an information base about the surrounding vehicles.

3) The cloud layer might also be involved in assisting RSUs for processing the vehicle information, especially in a large-scale system with a large number of vehicles.

4) In the vehicular cloud, a vehicle (referred to as task-vehicle) generates a series of tasks with different sizes and types, which can not be completed by the task-vehicle alone.

5) The task-vehicle sends a task offloading request to the nearest RSU. If the RSU accepts the offloading request, the task-vehicle offloads its tasks to the RSU and the RSU returns the processed results back to the task-vehicle.

6) If the RSU cannot accept the offloading request for some reason, such as lack of sufficient available resources to complete the tasks within the required time limit, it will decline the offloading request and returns information about surrounding vehicles to the task-vehicle.

7) The task-vehicle then makes decisions on offloading its tasks to other vehicles in the same vehicular cloud based on the vehicle information provided by RSUs. This is the stage where the proposed scheme – the proposed OPTOS scheme and HGSA algorithm – takes place.

8) The task-vehicle offloads its tasks to chosen vehicles according to the offloading decisions made in step-8.

9) The chosen vehicles (for receiving offloaded tasks) process the tasks assigned to them and return the results back to the task-vehicle.

10) If some vehicles leave the vehicular cloud before completing all their assigned tasks, then these unfinished

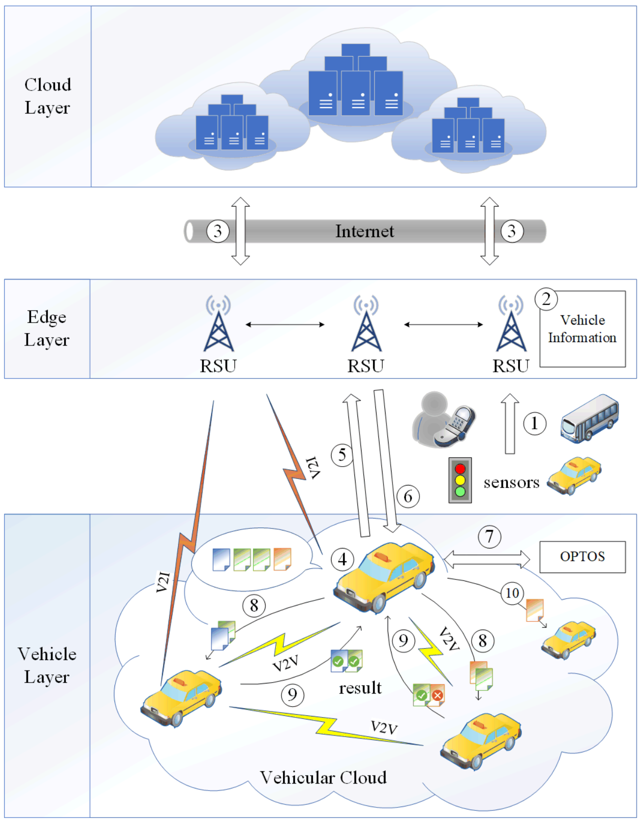


FIGURE 1. The three-layer architecture for task offloading in vehicular networks.

tasks have to be reassigned by the task-vehicle to other vehicles in the cloud by repeating the steps 7-9, which introduces extra delays in task completion.

We regard offloading to vehicles as a complementary strategy to traditional offloading to RSUs. In this way, when an RSU is available with sufficient resources for meeting the performance requirements of task processing, the task-vehicle offloads its tasks to the RSU. When the RSU is not available, the task-vehicle has an alternative solution – offloading tasks to other vehicles. Considering that task offloading to RSUs has been studied extensively, our research in this article focuses on the case of task offloading to vehicles assuming the task-vehicle has decided to do so. The specific problem we study here is: when the task-vehicle chooses to offload its tasks to other vehicles, how to assign these tasks to appropriate target vehicles in order to minimize the completion delay of all tasks (i.e., total task completion delay).

B. TASK COMPLETION DELAY

We now analyze the communication delay and computation delay associated with task offloading to vehicles to determine the complete task completion time. The main notations we used in our analysis are summarized in Table 1.

Let $V = \{v_1, v_2, \dots, v_n\}$ denote the set of n vehicles in a vehicular cloud. The information for each vehicle v_i can be represented by a tuple $\{f_i, l_i, s_i\}$, in which f_i gives the computing capacity (CPU cycles/s) of the vehicle, l_i is the current vehicle position, and s_i represents the instant speed of the vehicle. Suppose that vehicle v_k is the only task-vehicle that generates a set of m tasks denoted as $Q = \{q_1, q_2, \dots, q_m\}$. Each task q_i can be presented by a tuple $\{S_i, C_i, L_i\}$, where S_i is the size of data associated with the task, C_i is the computing complexity of the task (measured by CPU cycles/bit; i.e., the average number of CPU cycles needed for processing each bit of data of the task), L_i is the maximum tolerable completion delay of the task.

TABLE 1. Summary of Main Notations.

Notation	Description
V	The set of vehicles
n	The number of vehicles
f_i	The computing capability of the vehicle i
p_i	The position of vehicle i
s_i	The speed of the vehicle i
$st(i)$	The stay-time of the vehicle i
k	The task-vehicle
Q	The set of tasks
m	The number of tasks
S_q	The data size of the task q
C_q	The computational complexity of the task q(cycles/bit)
L_q	The delay limitation of the task q
e	The minimum size of a subtask
m'	The number of all subtasks
m'_q	The total number of subtasks generated from task q
x_{ij}	The offloading strategy
r	The transmission rate between two vehicles
D_{comm}	The communication delay of a subtask
D_{comp}	The computing delay of a subtask
D_{ij}	The completion delay of subtask i executed on vehicle j
$T_{task}(q)$	The time required for task q to be completed
$T_{vehicle}(j)$	The time required for vehicle j to complete all subtasks
T_{total}	The total time required for all tasks to be completed

Due to the differences in data sizes and computing complexity between tasks, offloading tasks directly to vehicles will lead to imbalanced utilization of vehicle resources. Therefore, we divide tasks into equal size subtasks and perform offloading to vehicles in the unit of subtasks. We suppose that there is no dependency between subtasks and all subtasks can be processed in parallel on different vehicles. Then, the total number of subtasks generated from m tasks can be calculated as

$$m' = \sum_{i=1}^{m} \frac{S_i}{e}, \tag{1}$$

where S_i is the data size of task q_i and e is the data size of a subtask.

Each subtask is either executed locally on the task-vehicle or offloaded to one of the $n-1$ vehicles. Each vehicle can have multiple subtasks simultaneously. We express the task offloading assignment as an $m' \times n$ matrix X

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & & & \\ x_{m'1} & x_{m'2} & \dots & x_{m'n} \end{bmatrix} \tag{2}$$

in which $x_{ij} = 1$ when the i-th subtask is assigned to vehicle v_j otherwise $x_{ij} = 0$. When $x_{ik} = 1$, it indicates that the task-vehicle v_k executes the i-th subtask locally.

When a subtask is executed locally the task completion delay equals computing delay. If the subtask is offloaded to another vehicle then the task completion delay is the sum of communication delay and computing delay. We assume that direct communication channels exist between all vehicles in the same vehicular cloud. The communication delay is the time taken to transmit subtask data from the task-vehicle to the target vehicle, which is mainly determined by the throughput of the wireless channel between these two vehicles. According to Shannon formula [26], the transmission rate between two vehicles can be calculated as

$$r = \frac{B}{n-1} log_2(1 + \frac{p \cdot h}{N}), \tag{3}$$

where B denotes the channel bandwidth, p represents the channel power, h means the channel gain, and N denotes the noise power. Since the task-vehicle transmits subtasks outward to all other vehicles simultaneously, the channel bandwidth will be divided into $n-1$ segments one for each target-vehicle. Therefore, the communication delay can be obtained by

$$D_{comm} = \frac{e}{r} = \frac{e(n-1)}{B \cdot log_2(1 + \frac{p \cdot h}{N})}. \tag{4}$$

After a subtask is executed, its result will be transmitted back to the task-vehicle. However, since the size of the result is much smaller than the size of the subtask, its transmission delay is negligible relative to the transmission delay of the subtask.

The computing delay for a subtask is the time for completing execution of the subtask, which is mainly determined by the amount of computation required by the task and the computing capacity of the vehicle where the task is processed, either the task-vehicle where the task is generated or a vehicle to which the task is offloaded. Therefore, the computing delay D_{comp} for the i-th subtask on vehicle v_j can be obtained as

$$D_{comp} = \frac{C_i \cdot e}{f_j}, \quad i = 1, 2, \ldots, m', \ j = 1, 2, \ldots, n, \tag{5}$$

where C_i is the computational complexity of the i-th subtask and f_j is the computing capacity of vehicle v_j.

Then completion delay D_{ij} for the i-th subtask executed on vehicle v_j can be determined as

$$D_{ij} = \begin{cases} \frac{C_i \cdot e}{f_j}, & j = k \\ \frac{C_i \cdot e}{f_j} + \frac{e}{r}, & j \neq k \end{cases} \tag{6}$$

where $j = k$ represents the case that this subtask is executed locally (i.e., without being offloaded).

Since a task is composed of many subtasks that are assigned to and processed on multiple vehicles, a task is completed when all its subtasks have been successfully processed by the vehicles they are assigned to. Therefore, the completion delay of a task q can be determined as

$$T_{task}(q) = max\{\sum_{i=1}^{m'_q} x_{ij} \cdot D_{ij}\}, \quad j = 1, 2, \ldots, n. \tag{7}$$

where, m'_q is the total number of subtasks generated from this task. Each task q needs to be completed within its corresponding maximum tolerable delay L; that is,

$$max\{\sum_{i=1}^{m'_q} x_{ij} \cdot D_{ij}\} \leq L_q, \quad j = 1, 2, \ldots, n. \tag{8}$$

Each vehicle may receive multiple offloaded subtasks. The total time for vehicle v_j to complete all the subtasks offloaded to it, denoted as $T_{vehicle}(j)$, is the sum of the completion delays of these subtasks; that is

$$T_{vehicle}(j) = \sum_{i=1}^{m'} x_{ij} \cdot D_{ij}, \quad j = 1, 2, \ldots, n. \tag{9}$$

If the vehicle leaves the vehicular cloud before finishing all the offloaded subtasks, then the unfinished subtasks will have to be reassigned to other vehicles, which causes additional communication and computing delay. Therefore, although some vehicles have strong computing power, they are not suitable for being assigned too many subtasks. The task offloading strategy should ensure that $T_{vehicle}(j)$ is no greater than the stay-time of the vehicle, $st(j)$, which is the residual time of the vehicle staying in the vehicular cloud. That is,

$$\sum_{i=1}^{m'} x_{ij} \cdot D_{ij} \leq st(j), \quad j = 1, 2, \ldots, n. \tag{10}$$

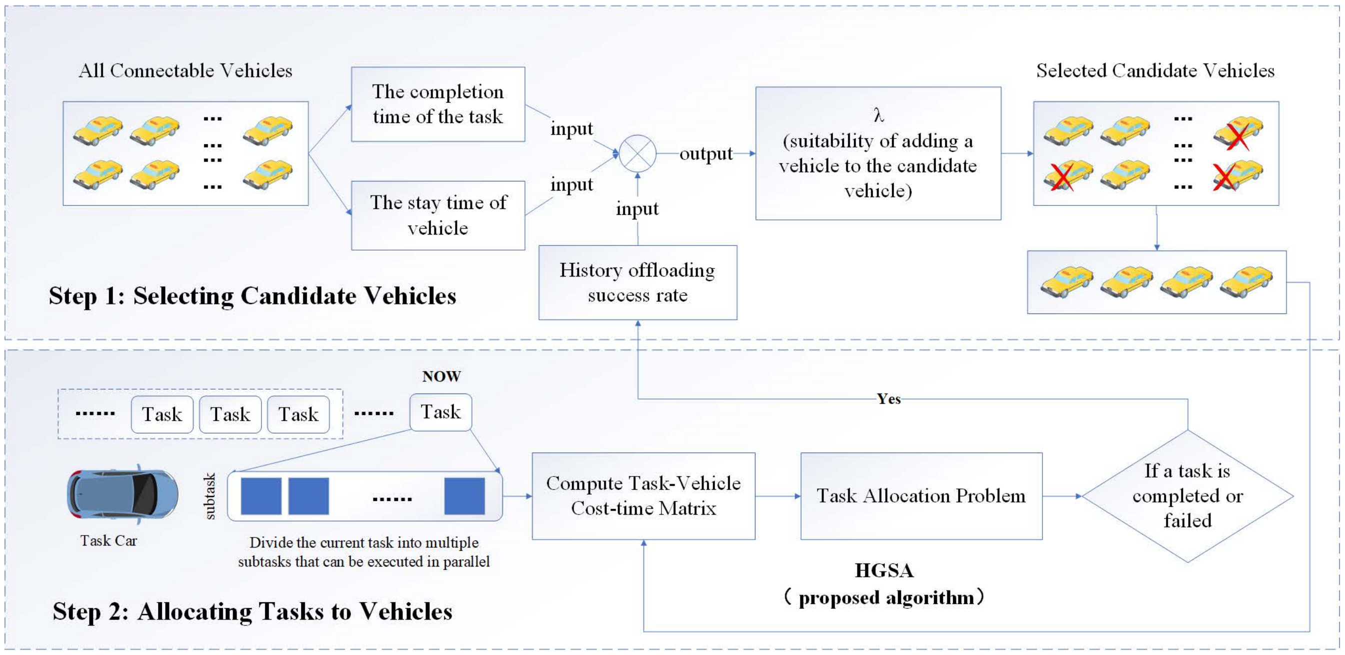
The total task completion delay depends on the last vehicle that completes all its subtasks. In other words, the total completion time of all the tasks is equal to the maximum completion time of all the vehicles. That is,

$$T_{total} = max\{\sum_{i=1}^{m'} x_{ij} \cdot D_{ij}\}, \quad j = 1, 2, \ldots, n. \tag{11}$$

C. PROBLEM FORMATION

Based on the above analysis, we formulate the problem of task offloading in a vehicular cloud as an optimization problem as (12). Given the number of subtasks m' and the number of vehicles n, the system uses an offloading decision matrix X that minimizes the total task completion time as the objective function while satisfying the constraints given by $C1$-$C4$.

$$min\{max\{\sum_{i=1}^{m'} x_{ij} \cdot D_{ij}\}, j = 1, 2, \ldots, n.\}$$
$$s.t. \ C1 : x_{i1} + x_{i2} + \cdots + x_{in} = 1, i = 1, 2, \ldots, m'$$



All Connectable Vehicles
The completion time of the task
The stay time of vehicle
input
input
input
output
History offloading success rate
λ (suitability of adding a vehicle to the candidate vehicle)
Selected Candidate Vehicles
Step 1: Selecting Candidate Vehicles
NOW
Task
Task
Task
Task
subtask
Task Car
Divide the current task into multiple subtasks that can be executed in parallel
Compute Task-Vehicle Cost-time Matrix
Task Allocation Problem
If a task is completed or failed
Yes
HGSA (proposed algorithm)
Step 2: Allocating Tasks to Vehicles

FIGURE 2. The structure of OPTOS. It contains two main steps: a) selecting candidate vehicles for each subtask; b) allocating subtasks to vehicles by HGSA Algorithm.

$$C2: x_{ij} = 0 \; or \; x_{ij} = 1$$

$$C3: max\{\sum_{i=1}^{m'_q} x_{ij} \cdot D_{ij}\} \leq L_q, \quad j = 1, 2, \ldots, n.$$

$$C4: \sum_{i=1}^{m'} x_{ij} \cdot D_{ij} \leq st(j), \quad j = 1, 2, \ldots, n \qquad (12)$$

Constraint $C1$ limits each subtask to be offloaded to only one vehicle. $C2$ indicates that $x_{ij} = 1$ when subtask i is offloaded to vehicle j, otherwise $x_{ij} = 0$. $C3$ requires each task to be completed within its delay budget. $C4$ requires each vehicle to complete all the subtasks offloaded to it within its residual time in the vehicular cloud (i.e., before the vehicle leaves the current vehicular cloud).

IV. TASK OFFLOADING STRATEGY

In this section, we first introduce the overall process of the OPTOS and then discuss the details of candidate vehicle selection and task allocation algorithm HGSA in OPTOS.

A. OPTOS SYSTEM

OPTFO is an online pre-filtering task offloading system, which includes a filtering process before offloading. It can obtain the optimal offloading strategy and significantly reduce the impact of vehicle mobility on task completion delay. The structure of the proposed OPTOS is shown in Figure 2, which comprises two steps: selecting candidate vehicles and allocating tasks to vehicles.

The first step selects reliable candidate vehicles for each subtask, which is designed to mitigate the negative impact of vehicle mobility on task offloading performance. Due to mobility, a vehicle may leave the vehicular cloud before completing all the subtasks offloaded to it – an event referred to as "early-leaving" in this paper. When this occurs, the uncompleted subtasks will need to be reassigned to other vehicles thus causing an additional delay. Therefore, OPTOS attempts to avoid early-leaving as much as possible. A vehicle that processes many subtasks or has a short stay-time in the vehicular cloud is more likely to cause early-leaving; therefore, it is less desirable to consider such a vehicle as a candidate when making offloading decisions. In order to evaluate whether a vehicle can be a candidate for a subtask, three factors are comprehensively considered in OPTOS – the stay-time of the vehicle in the vehicular cloud, the completion time of the subtask on this vehicle, and the task success rate of the vehicle. The OPTOS uses these three factors to calculate a value to guide the selection of candidate vehicles for each subtask.

In the second step, all subtasks are allocated to the appropriate vehicles. All tasks are divided into multiple independent subtasks that can be executed on different vehicles in parallel. This step aims to find the most suitable vehicle for each subtask to minimize the total completion time of all tasks. OPTOS first calculates the completion time of each subtask on all its candidate vehicles and then uses the proposed algorithm HGSA to obtain an optimal subtask-to-vehicle assignment in a limited time.

In addition, whenever a vehicle completes all its subtasks or leaves the vehicular cloud, the task success rate of the vehicle will be updated. If the task success rate increases, the vehicle will get more offloading opportunities. If the task

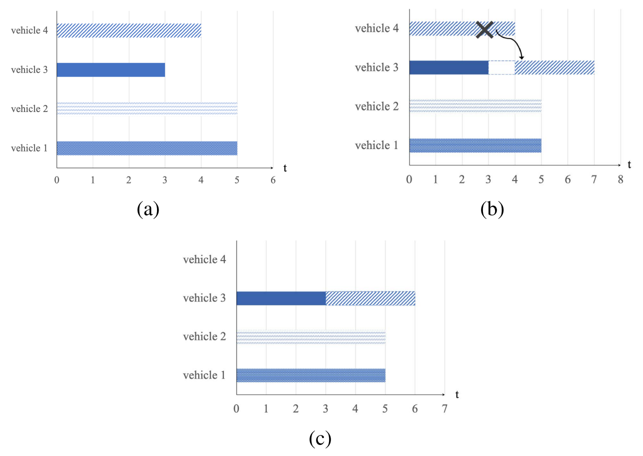


FIGURE 3. **The effect of candidate selection on the overall task completion delay. (a),(b) Offloading situation before and after the vehicle leaves without candidate processing. (c) Offloading strategy with candidate processing.**

success rate decreases, it indicates that the risk of offloading to this vehicle increases. The two-way feedback can improve the effectiveness of offloading decisions and reduce the total completion delay.

B. CANDIDATE VEHICLES

Selecting candidate vehicles for each subtask is essential because it facilitates task offloading to avoid the additional delay caused by early-leaving phenomenon thus mitigating the impact of vehicle mobility on delay performance. Figure 3 shows an example scenario that illustrates the effect of candidate selection on overall task completion delay, which demonstrates the necessity of this step.

Suppose there are four equal size subtasks and four vehicles with different computing capacities. The completion time of a single subtask on vehicle-1/2/3/4 is 5, 5, 3, 4 respectively. Figure 3(a) gives the subtask assignment without candidate selection – one subtask on each vehicle with a total completion time of 5. Figure 3(b) shows that early-leaving occurs to vehicle-4, which leaves the cloud at time instant 4 before it completes the subtask. Then the subtask assigned to vehicle-4 has to be reassigned to vehicle-3 to restart its execution. The total completion time of the four subtasks now becomes 7 due to the extra delay introduced by vehicle-4's early-leaving. Figure 3(c) gives the task offloading with candidate vehicle selection. Because vehicle-4 has a stay-time that is less than 4, which is the completion delay for a single subtask on this vehicle, vehicle-4 is removed from the set of candidate vehicles. Then two subtasks are assigned to vehicle-3 to achieve a shorter completion time than the case shown in Figure 3 (b).

In order to correctly decide whether a vehicle can be a candidate for subtasks, OPTOS comprehensively considers the following three factors:

- The completion time of the subtask: According to the analysis in Section III, the completion time of each subtask on each vehicle can be calculated using (6).

TABLE 2. **The training set for predicting the suitability of adding a vehicle to the candidate vehicle.**

Task type	stay-time	Completion time	History success rate	Success / failure
type-A	30	20	0.85	success
type-A	25	15	0.80	failure
type-B	28	17	0.75	failure
...	...	...	...	...
type-D	34	23	0.46	success

- The stay-time of the vehicle: The stay-time refers to the residual time of a vehicle in a vehicular cloud. Vehicle stay-time can be estimated using a vehicle mobility model together with vehicle information including the positions and speeds. We use the Freeway Mobility Model [27] to mimic vehicles' movements on highways. In this model, each mobile node is restricted to a specific lane and their movement must obey certain rules of relative speeds and distance between vehicles. Then the trajectory of the vehicles in a larger time range can be predicted and the stay-time can be estimated. We focus our research on highway scenarios because vehicles moving on highways are more likely to form stable vehicular clouds, in which task offloading to vehicles becomes a reasonable alternative to the conventional RSU-based task offloading. Please note that the proposed OPTOS may work with different mobility models for estimating vehicle stay-time.
- Offloading success rate: Assuming that the total number of tasks that have been offloaded to the i-th vehicle is y_i and the number of tasks completed by the vehicle is x_i, the offloading success rate of this vehicle can be calculated by x_i/y_i.

The relationship between task completion time and vehicle stay-time reflects whether the vehicle is capable of completing the task in time and the offloading success rate reflects the possibility that the vehicle fails the task due to other factors.

The historical data of the above three factors can be used to train a machine learning model that outputs the parameter λ that indicates the suitability of a vehicle to be a candidate vehicle. The training set format is shown in Table. 2. The value of λ is between [0,1], giving the probability that a vehicle may successfully complete all the subtasks offloaded to it thus being an appropriate candidate vehicle. This model is an online model. Therefore, whenever a new task is completed or failed, OPTOS updates the "history offloading success rate" and carries out new training on the model.

C. HGSA ALGORITHM

After selecting candidate vehicles for each task, the next step of OPTOS is to find the optimal task allocation strategy that minimizes the overall task completion delay.

Before assigning tasks to vehicles, OPTOS calculates the completion time of each subtask on its candidate vehicles and forms a cost-delay matrix D. The matrix D has m' rows and n columns, and each element D_{ij} represents the completion time of subtask-i on vehicle v_j. If v_j is identified as a candidate vehicle for subtask-i, then D_{ij} is calculated using (6); otherwise, D_{ij} is set to be infinity.

The goal of this step is to allocate m' independent subtasks to n vehicles to minimize the total task completion time. This problem has been formulated as a 0-1 integer linear programming problem given by (12), proven to be an NP-hard problem. This problem can be solved by a greedy algorithm to find the optimal solution; however, the complexity of a greedy algorithm increases greatly with the number of subtasks.

In this work, we propose a half greedy simulated annealing (HGSA) algorithm (Algorithm 1) that assigns subtasks to vehicles for minimizing the task completion delay. This algorithm combines the advantages of the greedy algorithm and simulated annealing, which can gain an optimal strategy in a limited time. The inputs of the algorithm include the number of subtasks m', the number of vehicles n, the cost-delay matrix D, and the parameters T_{max}, T_{min}, α for controlling the iterative annealing process. The algorithm outputs an $m' \times n$ task offloading matrix X whose element $x_{ij} = 1$ when the subtask-i is offloaded to vehicle-j otherwise $x_{ij} = 0$. Another output of the algorithm is a vector of vehicle total time $\vec{\tau}$ with n elements, each giving the total time for a vehicle to complete all the subtasks offloaded to it.

Steps 1-7 of the HGSA algorithm find a feasible task offloading matrix X. The algorithm first initializes the matrix X and $\vec{\tau}$ to 0 and then finds an available vehicle from the candidates for each subtask one by one. In order to obtain the optimal result in a short time, the algorithm does not randomly assign subtasks to vehicles in this phase but seeks a sub-optimal solution as much as possible. The algorithm ensures the overall delay is minimized in every offloading operation with the greed principle. After offloading each subtask to each candidate vehicle, the algorithm calculates the vehicle's total completion time and then selects the smallest one in step-3. Then the algorithm checks the constraints, such as whether the total completion time of this vehicle exceeds its stay-time. If the minimum vehicle meets the constraint, the algorithm selects it as the target vehicle for offloading (step-4 and step-5). After each subtask is assigned to a vehicle, the offloading matrix X and the corresponding vehicle's total time τ are updated. After the above operations are performed on all subtasks, the maximum total time in vehicles is the total completion delay $f(X)$ of all subtasks.

After initialization, the algorithm will perform simulated annealing. Similar to the metal annealing process, this part of the algorithm starts from the initial temperature T_{max}, with the continuous decline of temperature T, combined with the probability jump characteristics, randomly finds the optimal global solution of the objective function in the solution space until the temperature is lower than the termination temperature T_{min}, when the annealing process ends. In each iteration from step-9 to step-24, the algorithm first generates a new strategy by randomly exchanging the subtasks with different types, and then checks whether the new strategy meets the constraints. If the constraints are met,

Algorithm 1 HGSA Algorithm

```
Input: Number of subtasks $m'$; Number of vehicles $n$;
    Cost-delay matrix $D$; Initial temperature $T_{max}$; Terminal
    temperature $T_{min}$; Cooling rate $\alpha$
Output: Task offloading matrix $X$, Vehicle total time vector
    $\vec{\tau}$
 1: initial $X = 0, \vec{\tau} = 0$
 2: for $i = 1$ to $m'$ do
 3:     $j^* \leftarrow min\{\tau_j + D_{ij}\}, j = 1, 2, \ldots, n$
 4:     Check constraints of (12)
 5:     $x_{ij^*} = 1, \tau_j = \tau_j + D_{ij}$
 6: end for
 7: $f(X) = max\{\vec{\tau}\}$, according to (11)
 8: $T = T_{max}, X_{old} = X, f(X)_{old} = f(X)$
 9: while $T < T_{min}$ do
10:     $X_{old} = X, f(X)_{old} = f(X)$
11:     Randomly exchange two subtasks with different types,
        get $X_{new}$ and $\vec{\tau}_{new}$
12:     Check constraints of (12)
13:     $f(X)_{new} = max\{\vec{\tau}_{new}\}$, according to (11)
14:     $\Delta = f(X)_{new} - f(X)_{old}$
15:     if $\Delta < 0$ then
16:         $X = X_{new}$ and $\vec{\tau} = \vec{\tau}_{new}$
17:     else
18:         $p = \exp(\Delta / T)$
19:     end if
20:     if $ramdom.p < p$ then
21:         $X = X_{new}$ and $\vec{\tau} = \vec{\tau}_{new}$
22:     end if
23:     $T = T \cdot \alpha$
24: end while
25: return $X$ and $\vec{\tau}$
```

the corresponding $f(X)$ is calculated according to (11) to obtain the total delay required to complete all tasks (i.e., the maximum of the cumulative delay of each vehicle). Next, step-14 compares the value of the total completion delay $f(X)$ of the new and old strategies. If $f(X)_{new}$ is smaller than $f(X)_{old}$, the new strategy is better than the old one, then the new strategy X is accepted and vehicle total time $\vec{\tau}$ is updated. When $f(X)_{new}$ is larger, it may be a case of jumping out of the local optimum. Therefore, the new strategy is accepted according to a certain probability $p = \exp(\Delta/T)$ related to the current temperature. Finally, the current temperature is multiplied by the cooling factor to obtain the new temperature used in the next iteration. When the temperature drops to the lowest temperature, the iteration ends and the algorithm returns the optimal strategy X and the vehicle total time $\vec{\tau}$. Based on the above analysis, the HGSA algorithm can quickly obtain an offloading strategy that guarantees the optimal delay.

The HGSA algorithm also facilitates balancing the utilization of computing resources on different vehicles. In the process of finding a feasible solution, the algorithm assigns the current subtask to the vehicle with the lightest task load

TABLE 3. Summary of Simulation Parameters.

Parameters	Value
Number of vehicles in the vehicular cloud	10-20
Number of subtasks	100-1500
Range of V2V communication	150 m
Size of subtask	10^6 bit
Speed of vehicles	60-100 km/h
Vehicle computing power	4×10^6-2×10^7 cycles/s
Vehicle transmission power	1.3 W
Gaussian white noise power	3×10^{-13} W
Channel gain	4
V2V Link bandwidth	2 MHz
Data size of vehicular task	100 kb
Task computational complexity	30/15/40/20 cycles/bit

(the vehicle with the shortest completion time for all its subtasks), ensuring the completion time of different vehicles to be as close to each other as possible. By running the HGSA algorithm as a core part of OPTOS, the task-vehicle is not only responsible for task allocation but also in charge of load balancing.

V. EXPERIMENT AND EVALUATION

In this section, we first introduce the experiment environment, datasets, and parameter settings. Then, we evaluate the delay performance of the proposed algorithm by comparing it with the baseline algorithms. Finally, we discuss the effectiveness of the candidate selection process in OPTOS for handling vehicle mobility.

A. SIMULATION SETUP

We simulated our experiment environment in MATLAB by implementing a 10 km long three-lane highway and some moving vehicles. The trajectories of vehicles are selected from the Madrid trace [28], which is a set of synthetic traces containing real information from three highways (A6, M40, and M30) in Madrid, Spain. The vehicles have different computing capacities ranging from 4×10^6 cycles/sec to 2×10^7 cycles/sec. The tasks are classified into four categories – image processing, video processing, interactive game, and augmented reality, and their computational complexity parameters are 30, 15, 40, 20 respectively. Each task can be divided into 50-100 subtasks. The main simulation parameters are summarized in Table. 3.

In order to simulate the natural movement of the vehicles, we selected the track records of 20 vehicles on the A6 highway within a 30-minute time period. Furthermore, we set the vehicle communication range to be 150 m and every vehicle can communicate with all other vehicles at the beginning of the experiment. Figure 4 shows the connectivity among these 20 vehicles at the initial state.

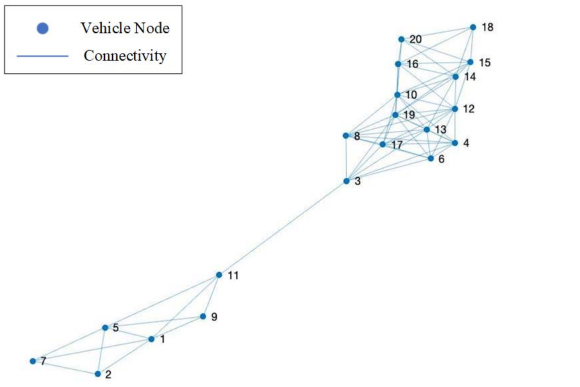


FIGURE 4. The connectivity among 20 vehicles in the vehicular cloud at the initial state.

B. PERFORMANCE COMPARISON

For performance comparison, we consider the following four algorithms as baseline algorithms:

Random: Each subtask will be randomly assigned to any vehicle that the task-vehicle can connect to.

Distance First (DF): Sort all vehicles according to their distance from the task-vehicle and then offload subtasks to the closest vehicle with a sufficient stay-time, as in [7].

Strongest Computing power First (SCF): Sort all vehicles according to their computing capacities and then offload subtasks to the vehicles with the strongest computing power.

Shortest Stay-time First (SSF): Subtasks are preferentially assigned to the earliest departing vehicle with the shortest stay-time.

1) IMPACT OF VEHICLE NUMBER

In order to study the influence of the number of vehicles on delay, we fixed the number of subtasks to 400 and then set the number of vehicles to 10 to 20.

Figure 5 shows the average completion delay of a subtask when the number of vehicles increases from 10 to 20, with the total number of subtasks is set to 400. It can be seen that the average delay of subtasks obtained by all the five algorithms increases with the number of vehicles. However, HGSA, SSF, and RANDOM algorithms have shorter average completion delay than SCF and DF do. This is because the SCF algorithm focuses on computing power and the DF algorithm focuses on communication power. Both of them consider only one part of the task completion delay.

Figure 6 shows how the total task completion delay changes when the number of vehicles in the vehicular cloud increases. It can be seen that the total completion delays obtained by the five algorithms all intend to decrease with the increase of the number of vehicles, among which the delay performance of SCF and DF is inferior and basically remains unchanged. This is because SCF and DF permanently offload tasks to the same vehicle, which weakens the benefit of executing subtasks on different vehicles in parallel. In addition, although HGSA has similar performance with SSF and Random in average task completion delay, HGSA performs better in total task completion delay. This is because our HGSA algorithm is able to balance

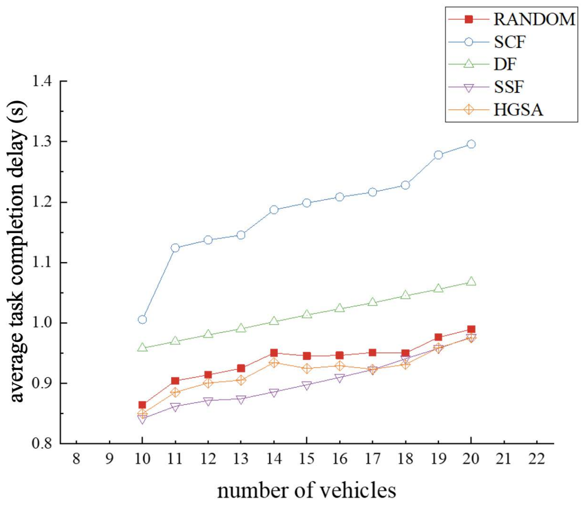


FIGURE 5. The average task completion delay versus the number of vehicles ($m' = 400$).

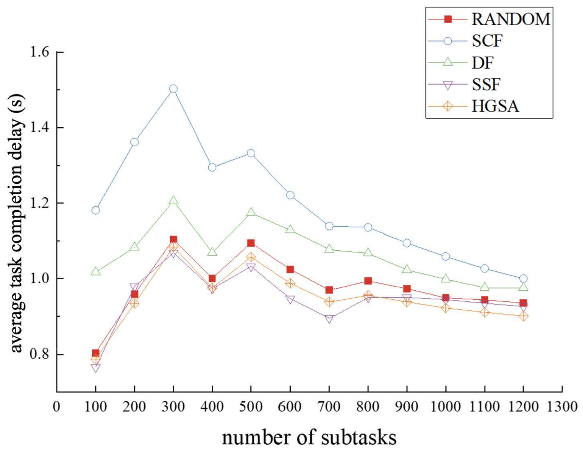


FIGURE 7. The average task completion delay versus the number of subtasks ($n = 20$).

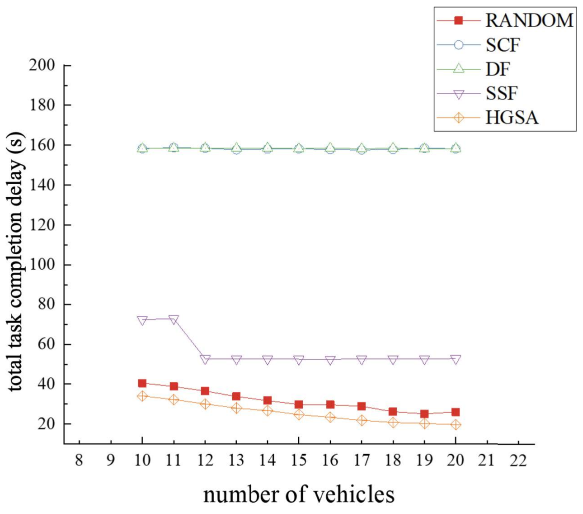


FIGURE 6. The total task completion delay versus the number of vehicles ($m' = 400$).

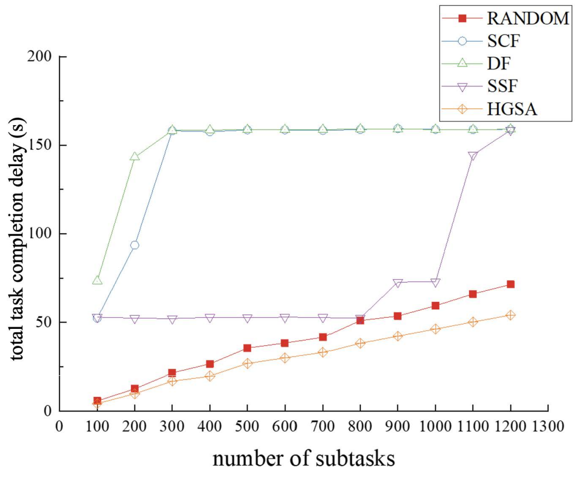


FIGURE 8. The total task completion delay versus the number of subtasks ($n = 20$).

the task loads across vehicles thus making full use of the vehicle resources. In conclusion, the HGSA algorithm outperforms the other four algorithms in total task completion delay. HGSA achieves 54% delay reduction compared to DF, SF, and SSF, and 19% delay reduction compared to Random.

2) IMPACT OF TASK NUMBER

Next, we discuss the impact of the number of tasks on delay performance when the number of vehicles is set to be 20.

Figure 7 shows the average completion delay of subtasks as the task number increases. We can find that the delay performance of the five algorithms have similar trends and HGSA always has the least delay regardless of the number of subtasks. The advantage of HGSA is more obvious when the number of subtasks is small. With a small number of subtasks, the priority offloading of other algorithms to a vehicle will lead to more idle vehicles while HGSA balances the computing resources of all vehicles to obtain the shortest delay.

Figure 8 shows how the total task completion delay changes with the increasing number of subtasks. It can be seen from this figure that the curves for HGSA and RANDOM increase linearly with the subtask number; the curves for SCF and DF first rise rapidly and then remain unchanged; while the curve SSF first remains unchanged and then rises rapidly. This is because the total completion delay depends on the vehicle with the longest completion time. When the number of tasks is large enough, SCF, DF, and SSF need more vehicles to join the offloading. They inevitably select a vehicle with a long stay-time, resulting in longer total completion delays. As a result, HGSA offloads tasks to all vehicles so that more tasks can run in parallel. The proposed HGSA algorithm reduces the total completion delay by 20% compared to the best performer among the four baseline algorithms.

TABLE 4. The distribution of stay-time of vehicles in three different scenarios.

Scenarios	min	P50	P75	P99	mean
Scenario 1 (high mobility)	53	53	74	118	60.8
Scenario 2 (low mobility)	53	174.5	290	580.5	206.7
Scenario 3 (medium mobility)	53	174.5	174.5	355.5	152.2

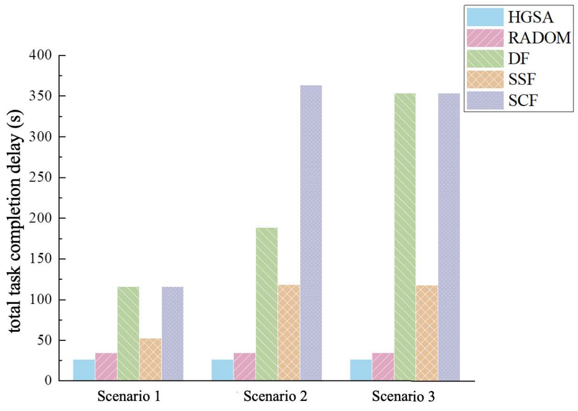


FIGURE 9. The total task completion delay in three scenarios with different mobility levels.

C. IMPACT OF VEHICLE MOBILITY

We conducted our experiments in three scenarios with different levels of vehicle mobility. The settings of these three scenarios are shown in Table 4.

- Scenario 1: The vehicular cloud comprises vehicles with high mobility and most of the vehicles may leave the cloud soon (having short stay-time).
- Scenario 2: The vehicular cloud comprises vehicles with low mobility and most of the vehicles stay in the cloud for a long time (having long stay-time).
- Scenario 3: The vehicular cloud has a medium level of mobility between scenario 1 and scenario 2.

Figure 9 shows the total completion delay for the three scenarios with different lengths of stay-time. It can be seen that the delays of DF, SSF, and SCF differ significantly in different scenarios, which indicates that the delay performance of these algorithms is greatly impacted by vehicle mobility. On the other hand, this figure shows that the proposed HGSA algorithm achieves the same level of delay in all three scenarios with different stay-time lengths. Therefore, the obtained experimental results indicate that HSGA is very stable with respect to vehicle mobility, mainly due to the candidate vehicle selection mechanism in OPTOS that mitigates the impact of mobility on task offloading performance.

In the proposed OPTOS, selecting candidate vehicles for each subtask plays an essential role in eliminating the impact of vehicle mobility. In order to demonstrate the importance of this step, we compared the total completion delay for each vehicle under the HGSA algorithm with and without the candidate selection process (HGSA-noCandicate). The obtained results are reported in Figure 10.

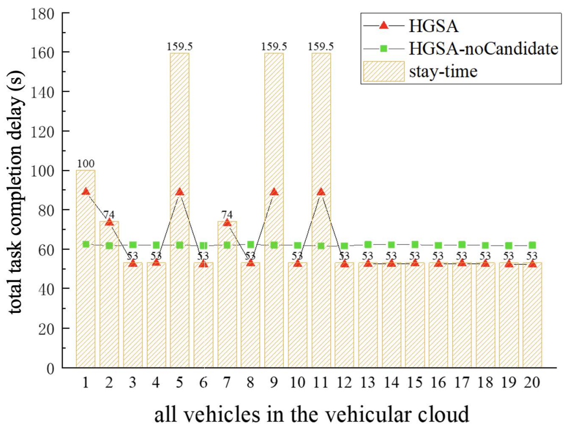


FIGURE 10. Comparison of HGSA algorithms with candidate selection and without candidate selection.

We set the number of subtasks to 1400 and the number of vehicles to 20. As can be seen from this figure, when HGSA without candidate selection is used for task offloading, all vehicles have a similar completion time. However, in this case, many vehicles, including vehicles 3, 4, 6, 8, 12-20, have a stay-time that is less than their completion time. It means that these vehicles will leave the vehicular cloud before completing all the subtasks offloaded onto them, which causes the failure of 70% of the offloaded tasks.

With candidate selection, the HGSA algorithm first guarantees that the total completion time of all the subtasks offloaded to a vehicle is less than the vehicle's stay-time, and then the algorithm balances the task loads across all candidate vehicles to make the completion time of all vehicles as close as possible. The results show that by sacrificing a small amount of delay, the algorithm with candidate selection can obtain an offloading strategy that achieves a 100% success rate; that is, all offloaded tasks are successfully completed. Without candidate selection, the failure rate of offloaded tasks reaches 70%, which will significantly increase the total task completion delay due to re-assignment and re-execution of the failed subtasks.

VI. CONCLUSION

In this paper, we designed the OPTOS system that is able to eliminate the negative impact of vehicle mobility on task offloading in vehicular clouds for achieving optimal delay performance and balanced vehicle resource utilization. The OPTOS comprises two key components – candidate vehicle selection and task-to-vehicle allocation. The online filtering process for selecting candidate vehicles protects the execution of offloaded tasks from being interrupted by vehicle mobility thus avoiding the extra delay caused by re-assignment and re-execution of the interrupted tasks. The task allocation algorithm HGSA identifies the most appropriate vehicle for each task in order to minimize the total task completion delay while balancing the utilization of computing capacities on different vehicles. We have conducted extensive experiments using a real-world dataset for evaluating the performance of the proposed OPTOS. Experiment results show that OPTOS outperforms the baseline algorithms by about 20% in task completion delay and significantly increases the task success rate. The obtained results also indicate that OPTOS is effective in various VANET scenarios with different levels of vehicle mobility.

In the future, we plan to take subtask inter-dependency and execution priority into consideration to further enhance OPTOS. We also plan to study the collaboration between vehicle clouds to provide more scalable task offloading for intelligent transportation systems.

REFERENCES

[1] B. N. Silva, M. Khan, and K. Han, "Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities," *Sustain. Soc.*, vol. 38, pp. 697–713, Apr. 2018.

[2] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 383–398, Jan. 2019.

[3] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. Shen, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.

[4] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *J. Netw. Comput. Appl.*, vol. 37, pp. 380–392, Jan. 2014.

[5] H. Hartenstein and L. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Commun. Mag.*, vol. 46, no. 6, pp. 164–171, Jun. 2008.

[6] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

[7] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint optimization of computation offloading and task scheduling in vehicular edge computing networks," *IEEE Access*, vol. 8, pp. 10466–10477, 2020.

[8] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5G networks for the Internet of Things: Communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619–3647, 2018.

[9] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[10] B. Ahmed, A. W. Malik, T. Hafeez, and N. Ahmed, "Services and simulation frameworks for vehicular cloud computing: A contemporary survey," *EURASIP J. Wireless Commun. Netw.*, vol. 4, pp. 1–21, Jan. 2019.

[11] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE Standard 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009), Jul. 2010, pp. 1–51.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[13] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[14] 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 Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7821–7834, Jul. 2020.

[15] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[16] V. D. Nguyen, T. T. Khanh, N. H. Tran, E.-N. Huh, and C. S. Hong, "Joint offloading and IEEE 802.11p-based contention control in vehicular edge computing," *IEEE Wireless Commun. Lett.*, vol. 9, no. 7, pp. 1014–1018, Jul. 2020.

[17] Y. Wang, P. Lang, D. Tian, J. Zhou, X. Duan, Y. Cao, and D. Zhao, "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[18] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1678–1689, Mar. 2020.

[19] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[20] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[21] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai, "Allocation of computation-intensive graph jobs over vehicular clouds in IoV," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 311–324, Jan. 2020.

[22] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[23] 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.

[24] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[25] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 48–54, Aug. 2018.

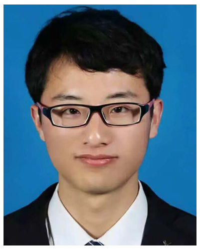
[26] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jun. 1948.

[27] F. Bai, N. Sadagopan, and A. Helmy, "The IMPORTANT framework for analyzing the impact of mobility on performance of RouTing protocols for Adhoc NeTworks," *Ad Hoc Netw.*, vol. 1, no. 4, pp. 383–403, 2003.

[28] M. Gramaglia, O. Trullols-Cruces, D. Naboulsi, M. Fiore, and M. Calderon, "Mobility and connectivity in highway vehicular networks: A case study in Madrid," *Comput. Commun.*, vol. 78, pp. 28–44, Dec. 2015.

JUNJING HE is currently pursuing the master's degree in computer science with Fudan University, Shanghai, China. Her research interest includes edge computing.

YUJIE WANG is currently pursuing the master's degree in computer science with Fudan University, Shanghai, China. His research interests include edge computing, machine learning, and service computing.



XIN DU is currently pursuing the Ph.D. degree with the Department of Computer Science, Fudan University, China. His research interests include edge computing, service computing, and machine learning.

ZHIHUI LU (Member, IEEE) received the Ph.D. degree in computer science from Fudan University, in 2004. He is currently a Professor at the School of Computer Science, Fudan University. His research interests include cloud computing and service computing technology, big data architecture, edge computing, and the IoT distributed systems. He is a member of the China computer federation's service computing specialized committee.

QIANG DUAN (Senior Member, IEEE) is currently a Professor with the College of Information Sciences and Technology, The Pennsylvania State University, Abington, PA, USA. His current research interests include network virtualization and softwarization, cognitive and autonomous networking, and edge computing-based ubiquitous intelligence. He has published three books and more than 100 research papers in these areas. He has served as an editor/associate editor for multiple research journals.

JIE WU received the Ph.D. degree in computer science from Fudan University, in 2008. He is currently a Professor at the School of Computer Science, Fudan University. His research interests include internet technology, big data architecture, edge computing, cloud computing, and the IoT distributed systems.