symmetry

MDPI

Article

ST-HO: Symmetry-Enhanced Energy-Efficient DAG Task Offloading Algorithm in Intelligent Transport System

Zhibin Gao [1], Gaoyu Luo [2,*], Shanhao Zhan [2], Bang Liu [2], Lianfen Huang [2] and Han-Chieh Chao [3,4,*]

[1] Navigation Institute, Jimei University, Xiamen 361021, China; gaozhibin@jmu.edu.cn
[2] School of Informatics, Xiamen University, Xiamen 361005, China; shanhao@stu.xmu.edu.cn (S.Z.); 23320181154319@stu.xmu.edu.cn (B.L.); lfhuang@xmu.edu.cn (L.H.)
[3] Department of Artificial Intelligence, Tamkang University, New Taipei City 25137, Taiwan
[4] Department of Electrical Engineering, National Dong Hwa University, Hualien 974301, Taiwan
* Correspondence: gyluo@stu.xmu.edu.cn (G.L.); hcc@mail.ndhu.edu.tw (H.-C.C.)

Abstract: In Intelligent Transport Systems (ITSs), Internet of Vehicles (IoV) communications and computation offloading technology have been introduced to assist with the burdensome sensing task processing, thus prompting a new design paradigm called mobile sensing–communication–computation (MSCC) synergy. Most researchers have focused on offloading strategy design to reduce energy consumption or execution costs, but ignore the intrinsic characteristics of tasks, which may lead to poor performance. This paper studies the offloading strategy of vehicle MSCC tasks represented by a Directed Acyclic Graph (DAG) structure. According to the DAG dependency of the subtasks, this paper proposes a computation offloading strategy to optimize energy consumption under time constraints. An energy consumption model for task execution is established. Then, the Simulated Annealing and Tabu Search hybrid optimization algorithm (ST-HO) is designed to solve the problem of minimizing the energy consumption. Crucially, this research integrates the concept of symmetry into the typical DAG structure of MSCC tasks, ensuring the integrity and efficiency of task execution in ITS. The simulation results show that ST-HO reduces energy consumption by at least 5.58% compared to the conventional algorithm. Particularly, the convergence speed of ST-HO is improved by 52.63% when the replication strategy of symmetric task is considered.

Keywords: Internet of Vehicles (IoV); Directed Acyclic Graph (DAG); Vehicle to Everything (V2X); Mobile Sensing–Communication–Computation (MSCC); Mobile Edge Computing (MEC)

Citation: Gao, Z.; Luo, G.; Zhan, S.; Liu, B.; Huang, L.; Chao, H.-C. ST-HO: Symmetry-Enhanced Energy-Efficient DAG Task Offloading Algorithm in Intelligent Transport System. *Symmetry* **2024**, *16*, 164. https://doi.org/10.3390/sym16020164

Academic Editor: Vasilis K. Oikonomou

Received: 17 December 2023
Revised: 25 January 2024
Accepted: 26 January 2024
Published: 31 January 2024

1. Introduction

With the rapid evolution of computer applications, communication technologies, and the Internet of Things, Intelligent Transportation Systems (ITS) have become a crucial component of modern city infrastructure [1]. One of the standout technologies in this domain is the Internet of Vehicles (IoV) [2,3], enabling comprehensive connectivity among vehicles, roads, and people. This connectivity is foundational for enhancing traffic management and services through efficient and secure means. At the heart of IoV is vehicle–road collaboration, whereby vehicles contribute to task computation through a Road Side Unit (RSU). However, with the proliferation of IoV applications like autonomous driving and traffic monitoring, there is an increasing demand for computational and storage resources. Cloud computing has been instrumental in addressing this demand, by breaking down tasks into smaller subtasks for parallel processing on multiple servers. This approach significantly improves data processing capabilities and efficiency. Nonetheless, challenges arise when the distance between vehicles and cloud centers increases, leading to higher transmission delays and fluctuation. This situation places considerable strain on communication networks, compromising service quality [4]. To mitigate these challenges, the development of Mobile Edge Computing (MEC) has gained prominence [5], allowing vehicles to offload

computing tasks to RSUs equipped with significant computational capabilities. By providing essential computing resources at the network edge, MEC alleviates both vehicles and the core network from computational burdens. Therefore, MEC technology and offloading methods have emerged as a new paradigm for providing energy-efficient and rapid computing services, which foster enhanced cooperation between vehicles and roads, such as addressing the uncertainty and latency in IoT–eHealth systems using fog computing [6], optimizing resource placement in 5G vehicular networks for intelligent transportation [7], and developing intelligent offloading strategies in high-mobility vehicular environments [8]. In early studies, Moubayed et al. focused on service deployment in Intelligent Transportation Systems using mobile edge computing, introducing an edge-based model to optimize vehicular network services, aiming to minimize latency and maximize availability [9]. This approach addresses the challenge of distance between vehicles, devices, and servers, then resource storage limitations [10,11] and flexible data caching [12] are considered.

In the evolving landscape of ITS, Cellular Vehicle to Everything (C-V2X) technology [13] emerges as a crucial component in enhancing vehicular communication and computational offloading. C-V2X, distinguished by its dual interfaces—PC5 for direct communication and Uu for cellular network-based communication—offers a versatile and robust solution to the increasing data and computational demands seen in ITS. The PC5 interface facilitates direct, low-latency communication between vehicles and RSUs, which is essential for immediate data exchange and real-time responses in critical scenarios. Meanwhile, the Uu interface enables reliable connectivity to broader network services, playing a crucial role in accessing cloud computing resources and supporting complex IoV applications. This dual-interface approach of C-V2X not only ensures comprehensive signal coverage, but also provides flexible channels for efficient computational offloading. It effectively addresses the limitations of onboard vehicular systems.

Vehicles, as nodes within mobile edge computing, generate vast amounts of data and are required to perform complex computing tasks. Despite the support from RSUs, optimizing task offloading strategies to minimize energy consumption and improve computational efficiency remains a challenge due to the limited computing capability and energy of onboard devices. Prior research on IoV computing offloading has mainly focused on the energy consumption of the entire vehicular network, taking a holistic and systemic study [14,15]. For instance, Wang et al. proposed a task offloading decision mechanism based on cooperative game theory and Deep Reinforcement Learning (DRL) to minimize the overall task processing delay and buffer size [16]. Although reinforcement learning methods demonstrate good performance, they entail significant computational costs and cache consumption for constrained Markov decision processes. Shen et al. designed a collaborative computing framework for task offloading, integrating an improved bald eagle search optimization algorithm. This framework aims to optimize latency and energy consumption, effectively reducing the total cost of task offloading in vehicular networks [17] without considering the mobility and offloading rates of vehicles. Deng et al. investigated an edge collaboration task offloading and splitting strategy, minimizing the total cost of delay and energy consumption [18]. Feng et al. investigated processing intensive tasks in 5G cellular–V2X networks, proposing a strategy combining computational offloading and URLLC for low latency and high reliability. This approach aims to maximize throughput and optimize power consumption while maintaining network stability [19]. These studies are significantly important, but their methods may not have taken into account the execution order relationships of components inside the task. As the number of connected vehicles increases, addressing the diverse and specific needs of individual vehicles becomes increasingly vital.

Fine-grained offloading, which involves dividing tasks into multiple independent subtasks based on internal attributes or offloading environments and then making offloading decisions for these subtasks, helps reduce communication delays and system overhead. Chen et al. conceptualized the fine-grained structure of tasks as Directed Acyclic Graphs (DAG) and defined an energy and time joint optimization model that utilizes the actor–critic

framework to output optimal offloading decisions and system rewards [20]. Similarly, Cui et al. used the DAG structure to optimize task execution locations and scheduling orders, employing a genetic algorithm for solving optimal task offloading decisions [21]. However, the complexity of the algorithm execution remains relatively high. Liu et al. addressed task scheduling in dynamic Vehicular Clouds (VCs) with DAG structures, introducing the RFID scheme for dynamic task allocation and execution adjustment [22]. Similarly, our work explores a vehicular cloud auction mechanism for task allocation, considering tasks as undirected graphs with varying execution demands, with MEC acting as a broker [23]. Liwang et al. proposed an efficient future resource trading mechanism to facilitate flexible resource transactions between MECs and dynamic vehicles for multiple computation-intensive tasks [24]. Beyond the studies previously mentioned, the application of blockchain technology [25,26] and federated learning [27] in vehicular networks has demonstrated notable effectiveness in enhancing security and efficiency. Asad et al. combined blockchain and federated learning in VANETs to enhance security and efficiency [26]. Wang et al. developed an MEC-supported federated learning framework, focusing on model caching and updates in local devices [27]. These studies suggest the potential of leveraging DAGs for improved task processing in future works. In these studies, the handling of symmetric tasks is of paramount importance. Symmetric tasks, which have similar structure and processing requirements, can be efficiently identified and optimally processed in a DAG structure. This approach not only reduces resource consumption, but also minimizes execution time through centralized processing or optimal scheduling. The DAG structure's emphasis on the internal time dependency of tasks provides a framework for effectively managing symmetric tasks. Shu et al. described the typical task topology of vehicular navigation applications and proposed an efficient parallel computation offloading strategy, which reduces task execution latency [28]. This strategy is particularly adept at handling symmetric tasks and further optimizes system performance.

While previous studies are significant, a gap still exists in existing research that fails to consider the integration of DAG with symmetric and non-offloadable tasks in system resource management strategies, leading to suboptimal offloading performance. This paper focuses on offloading strategies for vehicle MSCC tasks represented by DAG. The inherent DAG structure in MSCC tasks exhibits natural symmetry in task distribution and dependencies. In particular, this paper explores how to optimize execution and offloading for tasks with symmetric structures. Therefore, an optimized energy-saving computation offloading strategy is proposed for the execution of such vehicular offloading tasks under time delay constraints. Concretely, this paper introduces a Simulated Annealing and Tabu Search hybrid optimization algorithm (ST-HO) to address the minimization of energy consumption for DAG–structure task computation. The contributions of this paper are summarized as follows:

- Considering the complexity of vehicular tasks that can be represented by DAG structures, an optimized energy-efficient computational offloading strategy is proposed to perform such tasks under time constraints. The introduction of DAG takes into account the dependencies between preceding and subsequent tasks, presenting a non-convex optimization problem. This decision framework focuses on energy savings, aligning with the emerging needs for environmentally conscious computing in the vehicular context;
- Based on the temporal dependencies of DAG structure subtasks and the limitation that some subtasks can only be computed locally, we establish an energy consumption model for executing tasks. We integrate the structural analysis of both vehicular networks and DAG tasks to propose a computational offloading solution for delay-sensitive tasks. Addressing the minimization of energy consumption for DAG structure task computation, we introduce a Simulated Annealing and Tabu Search Hybrid Optimization Algorithm (ST-HO). By considering latency as a constraint, our algorithm avoids overly prioritizing time performance at the expense of energy consumption;

- Furthermore, when computing offloading strategies for symmetric tasks, we calculate the strategies for only half of the symmetric tasks in the DAG. The offloading strategies for the other half are directly derived by replicating those of the corresponding symmetric tasks, allowing for rapid decision-making. This targeted method leverages the symmetry within tasks to reduce computational overhead, thus promoting faster and more energy-efficient outcomes in complex task offloading scenarios.

The rest of the paper is organized as follows: Section 2 presents the system model and problem formulation. Section 3 proposes a task offloading decision algorithm based on a hybrid of Simulated Annealing and Tabu Search. Section 4 shows simulation results comparing the performance of the proposed algorithm with other methods. Finally, conclusions are summarized in Section 5.

2. System Model and Problem Formulation

2.1. Scenario Description

We consider a scenario in which RSUs provide complete signal coverage along the road. As shown in Figure 1, the RSUs are interconnected through wired optical cables, facilitating efficient data sharing among MEC servers. Vehicles move at a constant speed and communicate with the RSU to which they are connected. To utilize the robust computational capabilities of MEC servers, it is essential to offload certain subtasks from vehicular device applications to the MEC servers.

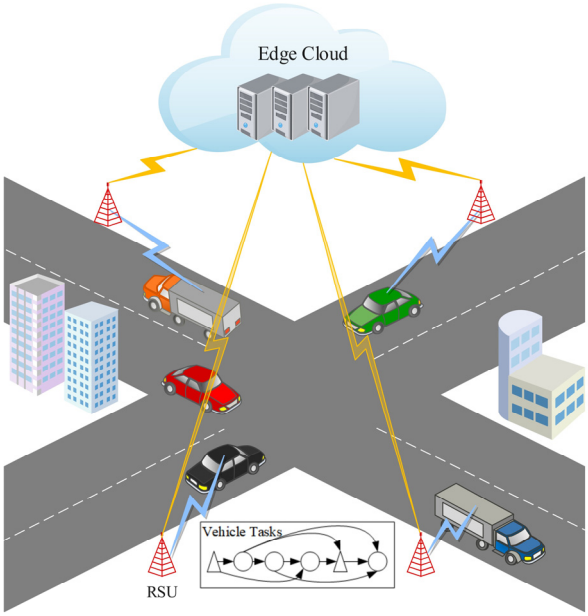

Figure 1. System scenario.

2.2. Task Model

Suppose a vehicle application task comprises several subtasks that can only be computed locally and several subtasks that can be offloaded to the MEC [12], among which the offloadable subtasks can be computed either locally in the vehicle or on the MEC server, as shown in Figure 2. The tasks are divided into two categories and a DAG $G = (V, D)$ is used to represent. $v_x \in V$ represents the subtask, and $d_{xy} \in D$ is the size of the data transferred from subtask y to subtask x on the directed edge. Triangles represent subtasks that must be executed locally, i.e., $\{v_1, v_5\} \subseteq V_{local}$. Hollow circles denote subtasks that can be offloaded, i.e., $\{v_2, v_3, v_4, v_6\} \subseteq V_{offload}$. Therefore, subtask 1 and subtask 5 are constrained to be computed locally while other subtasks offer the flexibility of being computed at optional locations, either locally or on the MEC server, and the entire process ends after the execution of subtask 6.

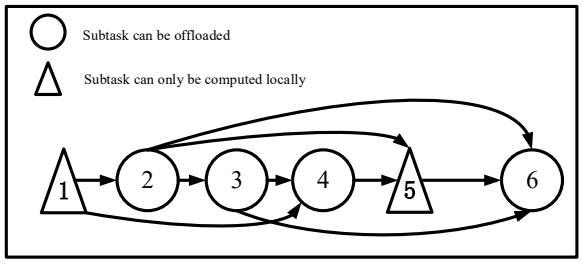

Figure 2. Task model.

In addition, subtask 5 is executed only after receiving data from subtask 2 and subtask 4. Thus, we define a binary vector $r_{xy} \in \{0,1\}$ to represent the dependency between subtask x and subtask y, as follows:

$$r_{xy} = \begin{cases} 1 & \text{subtask } x \text{ depends on subtask } y \\ 0 & \text{subtask } x \text{ doesn't depend on subtask } y \end{cases} \quad (1)$$

This formula declares the execution dependencies of the individual subtasks. If a subtask has multiple prerequisite subtasks, it can only commence after the completion of all its preceding subtasks. For instance, subtask 6 requires the completion of subtasks 2, 3, and 5, and it must also receive the data generated by these subtasks before execution.

2.3. Energy Consumption Model

Assuming that the supplementary noise present in vehicle-to-infrastructure communication is Gaussian noise with zero mean and variance σ^2, the transmission rate between the vehicle and edge server can be expressed as follows:

$$R = B\, log_2(1 + \frac{ph}{BN_0}) \quad (2)$$

where N_0 is the noise power spectral density. B is the communication bandwidth between the vehicle and the RSU. p represents the transmission power of the signal sender and h is the channel gain.

For a subtask x, $s_x \in \{0,1\}$ is the decision identity, as follows:

$$s_x = \begin{cases} 0 & \text{subtask } x \text{ is computed at MEC} \\ 1 & \text{subtask } x \text{ is computed at local} \end{cases} \quad (3)$$

2.3.1. Local Computing

In this case, only the computation process needs to be considered. The local computation time of subtask x can be given by

$$t_x^{local} = \frac{C_x}{f_{local}} \quad (4)$$

where C_x (M cycles) is the computation amount of subtask x, and f_{local} represents the local computation capacity of the vehicle device. The energy consumption of subtask x computed locally is

$$e_x^{local} = p_{local} t_x^{local} \quad (5)$$

2.3.2. MEC Computing

In this case, it is indispensable to consider the calculation process and communication process. The subtask will contain two quantities. W_x (KB) represents the amount of data corresponding to the basic information that subtask x needs to provide when it is calculated by the MEC server. The computation time of subtask x at the MEC server can be expressed as $t_x^{mec} = C_x / f_{mec}$, where f_{mec} is the computational capability of MEC. The

transmission time of offloading subtask x to MEC server is $t_x^{trans} = W_x / R_{vg}$, where R_{vg} is the communication transmission rate from the vehicle to the MEC server. When subtask x is offloaded to MEC server, the onboard device is idle. However, the energy consumption is also generated to maintain basic work, which can be given by

$$e_x^{basic} = p_{basic} t_x^{mec} \tag{6}$$

where p_{basic} is the power of the onboard devices to maintain basic operation. The energy consumption that subtask x offloads to the MEC server during transmission is

$$e_x^{trans} = p_{up} t_x^{trans} \tag{7}$$

where p_{up} denotes the communication power when the onboard device is uploading. Therefore, the total time t_x^{edge} spent by subtask x in the MEC server's computation is $t_x^{edge} = t_x^{trans} + t_x^{mec}$.

The total energy consumption of on-board equipment is denoted as $e_x^{edge} = e_x^{trans} + e_x^{basic}$.

2.3.3. Results Transmission

In the process of task computation, if $s_x = s_y$, both subtask x and subtask y are executed locally or by the MEC server. Thus $t_{xy} = e_{xy} = 0$, where t_{xy} is the total time of transmission, and e_{xy} is the total energy of transmission. Except for the above case, we see the transmission of calculation results between the onboard device and the MEC server. If $s_x = 0$ and $s_y = 1$, the subtask x is executed on the MEC server and its preceding subtask y is executed on the vehicle terminal, which is represented as the calculation uploading process. The time of uploading data in the data transmission process is $t_{xy}^{up} = d_{xy} / R_{vg}$.

The energy consumption during uploading is

$$e_{xy}^{up} = p_{up} t_{xy}^{up} \tag{8}$$

where p_{up} is the communication power during uploading. If $s_x = 1$ and $s_y = 0$, the subtask x is executed locally and its preceding subtask y is executed on the MEC server, which is manifested as a download process. The download time of the data transfer can be given by $t_{xy}^{down} = d_{xy} / R_{gv}$, where R_{gv} is the communication transmission rate from the MEC server to the vehicle. Therefore, the energy consumption during the downloading is

$$e_{xy}^{down} = p_{down} t_{xy}^{down} \tag{9}$$

where P_{down} is the communication power of downloading. Hence, the energy consumption of the onboard device during the whole transmission process is

$$e_{xy} = (1 - s_x) e_{xy}^{up} + s_x e_{xy}^{down} \tag{10}$$

The transmission time is

$$t_{xy} = (1 - s_x) t_{xy}^{up} + s_x t_{xy}^{down} \tag{11}$$

2.4. Problem Formulation

If the total execution time of the task after offloading to the MEC server is greater than the time of all local execution, then the decision of task offloading is unacceptable. Therefore, t_{total} is defined as the total time required for the local execution of all subtasks, which determines the time threshold of the task offloading process. The optimization objective is to find an offloading strategy where the execution time of the task is less than the time limit and the energy consumption of the onboard device is minimized. The complete time of subtask x is

$$tm_x^{com} = t_x^{exec} + tm_x^{start} \tag{12}$$

where tm_x^{start} is the start time of computing, and t_x^{exec} is the execution time, which can be denoted as

$$t_x^{exec} = s_x t_x^{local} + (1 - s_x) t_x^{edge} \tag{13}$$

where the right side of the equation represents the time taken by the onboard device to execute the task locally and the task to be computed by the MEC server, respectively. However, subtask x may depend on the computation results of multiple preceding subtasks, and must receive all the results before it can be executed. Therefore, tm_x^{start} can be given by

$$tm_x^{start} = \max_{(x,y) \in G} r_{xy} \left(tm_y^{com} + |s_x - s_y| t_{xy} \right) \tag{14}$$

which means that subtask x cannot execute until it receives the calculation results of all the preceding subtasks. The start time for subtask x is determined by the latest completion moment among its multiple preceding subtasks.

For a DAG structure task with N subtasks, the time required to complete the task is

$$t_{total} = tm_N^{com} - tm_1^{start} \tag{15}$$

where tm_N^{com} is the end moment of the last subtask, and tm_1^{start} is the start moment of the first subtask.

We define $S = [s_1, s_2, \ldots, s_N]$ to represent the offloading decision for each subtask. For subtasks that can only be computed locally, $s_x = 1$. The time limit t_{total} is calculated by the time when all subtasks are executed locally, i.e., when $S = [1, 1, \ldots, 1]$.

The energy consumption of the whole system can be given by

$$\begin{aligned} & E(S) = \sum_{v_x \in V} \left[s_x e_x^{local} + (1 - s_x) e_x^{edge} \right] + \sum_{(x,y) \in G} \left| s_x - s_y \right| e_{xy} \\ & s.t. \\ & s_x \in \{0, 1\} \\ & s_x = 1, \forall v_x \in V_{local} \\ & S = [s_1, s_2, \ldots, s_N] \end{aligned} \tag{16}$$

which means that the system's energy consumption is derived from three main components: local computing, offloading to MEC computing, and transmitting the processing results of subtasks. The former sum term represents the energy consumption of local or MEC server computing. The latter sum term accounts for the energy consumption arising from transferring calculation results when interdependent subtasks are computed at different locations. $(x, y) \in G$ denotes the dependency between tasks in the DAG structure.

For the whole system, the goal is to identify the decision that corresponds to the minimum energy consumption, while satisfying the delay requirement, as follows:

$$\begin{aligned} & \min_S E(S) \\ & s.t. \\ & t_{total} < t_{local} \\ & s_x \in \{0, 1\} \\ & s_x = 1, \forall v_x \in V_{local} \\ & S = [s_1, s_2, \ldots, s_N] \end{aligned} \tag{17}$$

However, different computational decisions correspond to different system completion times and overall system energy consumption. For the task with k offloadable subtasks, there are 2^k corresponding decisions. If the exhaustive method is used for optimization, it will cause high computational complexity.

3. Optimization Scheme

We introduce a hybrid algorithm that merges the Simulated Annealing (SA) algorithm's capability to escape local optima with the Tabu Search (TS) algorithm's advantage of avoiding repetitive searches. This optimization scheme capitalizes on the strengths of both algorithms.

3.1. Simulated Annealing Algorithm

The SA algorithm is inspired by the annealing process of solid materials in physics, where the internal energy of a solid minimizes as it gradually cools from a high temperature to a low one. This process bears a striking resemblance to the approach of solving combinatorial optimization problems. The SA algorithm mainly consists of two parts: the Metropolis criterion and the annealing process. The Metropolis criterion is as follows: During the search for the solution that minimizes the system's objective function, assume the system transitions from a previous state $x(n)$ to the current state $x(n+1)$. As the state changes, the system's objective function changes from $E(n)$ to $E(n+1)$. In this case, the probability of the current optimal state transitioning from state $x(n)$ to $x(n+1)$ is P.

As can be seen from the above formulas, if the value of the objective function decreases, then the state $x(n+1)$ is considered as the current optimal, and is directly taken as the criterion for subsequent calculations. On the other hand, if the value of the objective function increases, this indicates that the current state $x(n+1)$ is deviating from the global optimal state in comparison with $x(n)$. Instead, the algorithm calculates the probability P according to [29], and then decides the next step. Specifically, the system generates a random number ε in the interval $[0, 1]$; if $\varepsilon <$ P, the algorithm transitions to the new state $x(n+1)$ for further computation. Otherwise, it continues with state $x(n)$ as the criterion. The value of P is influenced by both the change value of the objective function and the temperature, which changes dynamically during the calculation. The annealing process is controlled by the initial temperature and the temperature decay factor.

3.2. Tabu Search Algorithm

The Tabu Search (TS) algorithm starts from an initial feasible solution, and then iteratively improves the solution until it meets the termination condition. In each iteration, TS first conducts a neighborhood search from the current solution. It then selects the best solution within this neighborhood based on the objective function. Finally, the algorithm updates the current optimal solution according to the optimal neighborhood solution. To escape from the local optimum, TS utilizes a tabu table, which marks the local optimal solutions and corresponding neighborhood actions that have been explored, thereby avoiding repetition in subsequent searches. The evaluation function defaults to the objective function in combinatorial optimization problems. The override criterion allows, in cases where all options are in the tabu list, the selection of the least detrimental tabu option for the solution.

3.3. ST-HO Algorithm

When implementing the Tabu Search algorithm, the offloading decisions define the solution space with energy consumption as the objective function. Initially, an offloading decision that satisfies the constraints is randomly generated as both the current and optimal solution. Under the guidance of the tabu list, the algorithm explores the neighborhood of the current solution for a better candidate. If an optimal candidate solution is superior to the current best solution, or if it meets the criteria for amnesty, it becomes the new optimum. Otherwise, the second-best non-tabu candidate is chosen, and the tabu tenure is updated. This process continues until the termination criteria are met, at which point the optimal solution is selected.

Subsequently, the Simulated Annealing algorithm is employed for global optimization. It starts with a predefined initial temperature, cooling factor, maximum iterations, and termination temperature. At each temperature level, the algorithm generates a random

offloading decision within the constraints and applies the Metropolis criterion. If the termination condition at the current temperature is satisfied, the system is cooled down; otherwise, the iteration continues. This process is repeated until the temperature drops below the termination threshold, yielding the final solution. The details are described in Algorithm 1.

Algorithm 1: Task offloading decision algorithm based on ST-HO.

Input:
DAG structure tasks and vehicular network parameters.
Output:
Optimal offloading decision and its corresponding minimum energy consumption.

1. **Initiate**: tabu object, tabu length, initial temperature T, cooling factor a and end temperature T_{end}.
2. The delay constraint t_{total} is computed iteratively.
3. Randomly generate a solution S_{new} that satisfies the delay constraint.
4. Set the current solution and the optimal solution: $S_{now} = S_{new}$, $S_{best} = S_{now}$.
5. **While** the Tabu Search termination condition is not reached **do**
6. Perform neighborhood actions on S_{now} to generate a neighborhood set that meets the delay constraints;
7. Update S_{now} and the tabu list according to tabu rules. Simultaneously, compare to obtain the solution S_{best} corresponding to the minimum energy consumption in the current iteration;
8. **End while**
9. The optimal solution of Tabu Search S_{best} is used as the Simulated Annealing initial solution S_{now} .
10. **While** the temperature T is greater than the temperature T_{end} at the end **do**
11. **While** the termination condition of current temperature is not reached **do**
12. Perturb a solution S_{new} that satisfies the delay constraint;
13. Calculate the energy consumption E_{s_new} under S_{new} using Equation (16);
14. Calculate the difference in energy consumption dE $= E_{s\_new} - E_{s\_now}$;
15. **If** $dE < 0$ then
16. Accept the new solution $S_{now} = S_{new}$ and update $S_{best} = S_{now}$;
17. **Else**
18. **If** $exp(-dE/T) > random(0,1)$ then
19. Accept the new solution $S_{now} = S_{new}$.
20. **End if**
21. **End if**
22. **End while**
23. The cooling annealing $T =$ a*T.
24. **End while**
25. Return the optimal offloading strategy $S(i)$, and the corresponding minimum energy consumption $E_{s(i)}$.

4. The Simulation Analysis

We first give the relevant parameter value settings for this simulation in Table 1, and compare the proposed algorithm with some traditional algorithms to verify the performance advantages of the proposed algorithm.

The DAG structure task is represented in Figure 3, where subtasks 1, 5, 6, and 10 can only be computed using the onboard device, and the remaining subtasks have the flexibility to be computed either using the onboard device or the MEC server. The arrows represent the dependencies between subtasks. The computation of a subtask commences only after the completion of all its preceding dependent subtasks. Accordingly, each subtask request operates independently, ensuring that there is no interference in the computation times due to communication.

Table 1. Parameters for simulation.

Parameter	Value	Parameter	Value
T	100	p_{up}	0.5 W
T_{end}	0.01	p_{down}	0.2 W
a	0.992	B	10 MHz [12]
$limit$	5	N_0	−174 dBm/Hz
f_{local}	1 GHz [25]	f_{mec}	5 GHz
p_{local}	1 W	p_{basic}	0.1 W [30]

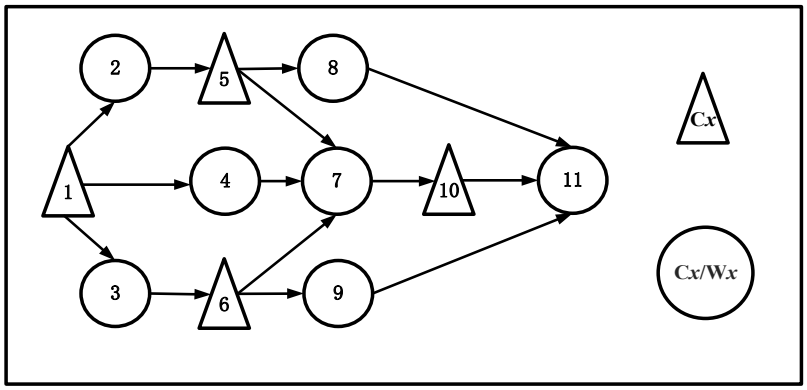

Figure 3. Simulation task topology diagram.

4.1. Convergence of ST-HO

To evaluate the optimization effects and convergence of the algorithms proposed in this paper, two different sizes of DAG structure task models are established. The smaller task consists of 11 subtasks, with computational tasks, transmission data and the offloading data between subtasks adhering to the uniform distribution. Approximately 20–30% of these subtasks can only be computed locally. The larger task is configured with 30 subtasks, following the same setup rules as the smaller model.

In the experiment, the proposed ST-HO algorithm and the following algorithms are selected for comparison:

- Random algorithm—This approach involves offloading the tasks that can be offloaded at random, either to the vehicle terminal or to the MEC server for execution;
- Greedy algorithm—During each iteration, the algorithm compares the current solution with the optimal one. If the current solution is superior, it is then set as the new optimal solution;
- Simulated Annealing algorithm—Mainly consists of the Metropolis criterion and the annealing process.

Figure 4 shows the performance of the proposed ST-HO algorithm in finding optimal solutions for smaller-scale DAG structured tasks, compared to other algorithms during the iterative solving process. The ST-HO algorithm requires fewer iterations to reach the same level of fitness as the other algorithms, which indicates that it has a lower time complexity. Moreover, the ST-HO algorithm exhibits the highest efficiency in terms of energy consumption throughout the iterative process.

Figure 5 shows the performance of the ST-HO algorithm and the other algorithms in determining optimal solutions for the larger DAG structure task. The results indicate that the ST-HO algorithm effectively scales with the increasing size of the task, maintaining its efficiency. It also shows superior convergence performance compared to other algorithms.

4.2. Performance of ST-HO

In this section, we analyze the performance of the ST-HO algorithm. Figure 6 shows a comparison of the energy consumptions of four algorithms—the ST-HO algorithm, the Simulated Annealing algorithm, the Greedy algorithm, and the Exhaustive algorithm

(EA)—when processing 10 to 14 tasks. The energy consumption of all algorithms shows an upward trend as the number of tasks increases. Notably, the performance curves of the ST-HO algorithm closely align with those of the optimal solution, indicating that the ST-HO algorithm is very effective in finding solutions that approximate optimal energy consumption. Although the energy consumption of the ST-HO algorithm is slightly higher than that of the optimal solution, it provides a good balance between computational complexity and optimization performance.

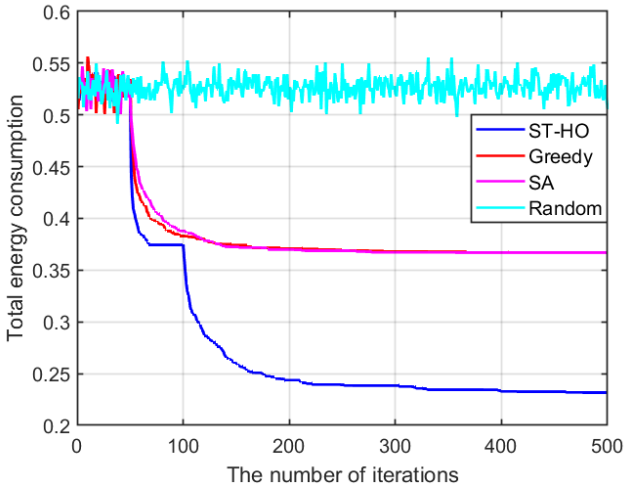

Figure 4. Convergence performance of ST-HO algorithm in terms of total energy consumption under different numbers of iterations with task number 11.

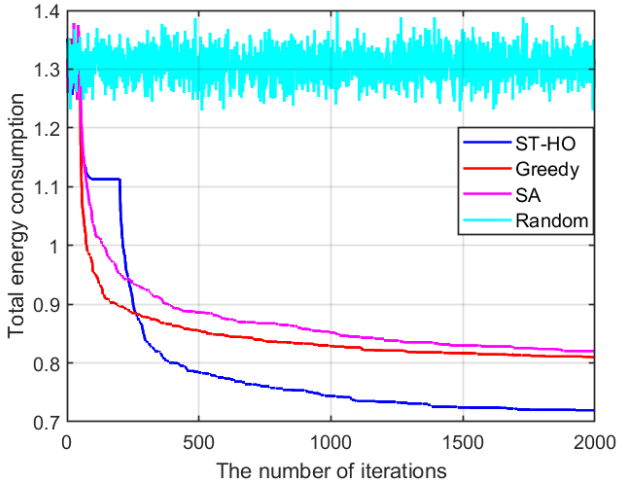

Figure 5. Convergence performance of ST-HO algorithm in terms of total energy consumption under different numbers of iterations with task number 30.

Figure 7 shows the variations in system energy consumption for different algorithms during the task offloading process. This variation is observed as the computation rate of the MEC server changes for smaller and larger DAG structure tasks. As shown in Figure 7a,b, the Random algorithm exhibits irregular variations in energy consumption, attributed to the lack of comparison and optimization in offloading decisions. In contrast, the ST-HO, SA, and Greedy algorithms exhibit a decrease in energy consumption when the computation rate of the MEC server increases. Of the four algorithms, the ST-HO algorithm is the most effective in optimizing energy for task offloading decisions, particularly in minimizing energy consumption with time constraints. With the increasing MEC computation rate, the offloading strategies from the SA, Greedy, and ST-HO algorithms achieve modest reductions in system energy consumption, remaining below 0.05 w. Changes in MEC

computing capability have little effect on these algorithms' ability to find the optimal offloading strategy.

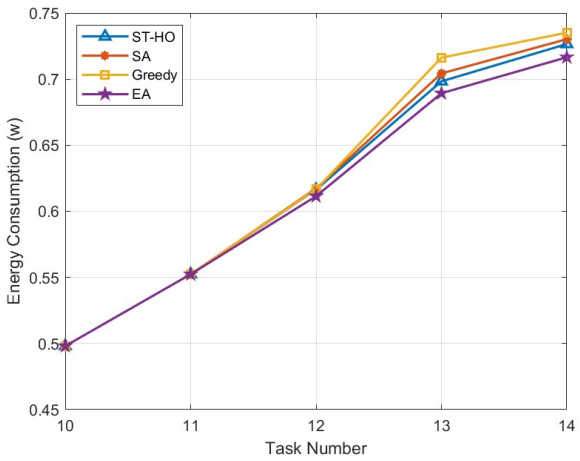

Figure 6. Comparison of energy consumption of different algorithms.

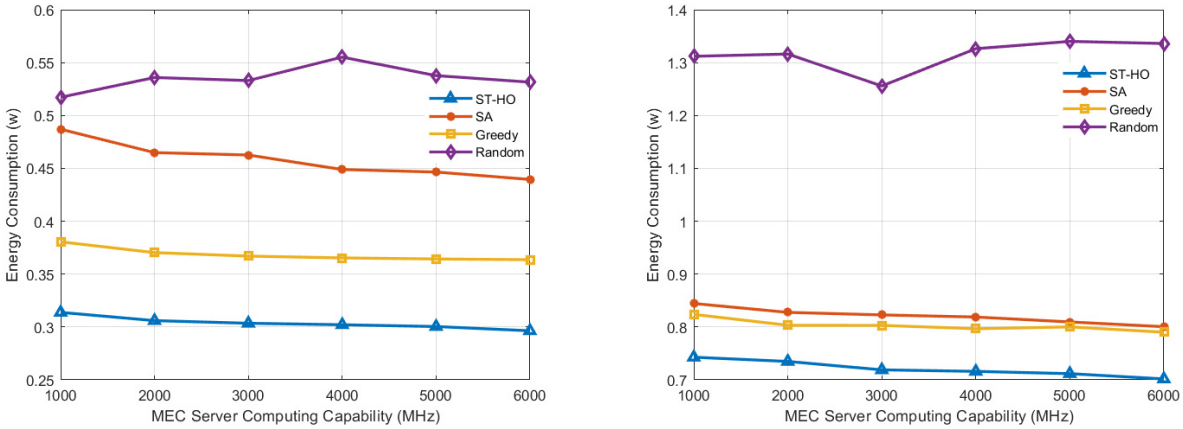

Figure 7. Changes in system energy consumption as the computing capability of the MEC server varies (**a**) when the DAG structure task number is 11 and (**b**) when the DAG structure task number is 30.

Figure 8 shows the performance of ST-HO compared to the SA, Greedy and Random algorithms with different local computing capabilities in both smaller- and larger-scale tasks. Figure 8a illustrates that the ST-HO algorithm outperforms all other algorithms under different local computational capacities, reflecting its ability to utilize local resources efficiently. The ST-HO excels across a broader range of computational capacities and can effectively handle different computational resource conditions. Although the Greedy and SA algorithms improve with increased local capacity, their potential for enhancement is somewhat limited. Especially the Greedy algorithm, with its slow convergence speed and increasing complexity when the number of tasks rises, falls short in the car networking environment, which has a high demand for fast response. Thus, the ST-HO algorithm is more balanced and reliable, particularly in complex scenarios where real-time performance and task size are critical factors. Figure 8b shows that, with large tasks, it performs similarly to how it does with smaller tasks; ST-HO demonstrates a significant decrease in energy consumption as the local computing capability increases, indicating that it scales well with resource availability. The SA algorithm shows a decrease in energy consumption with enhanced local capabilities, but its improvements plateau and do not match the efficiency gains seen with ST-HO. The Greedy algorithm demonstrates an energy consumption decrease that is comparable to the SA algorithm, indicating similar behavior in leveraging

local computing resources. In contrast, the Random algorithm exhibits the highest energy consumption, with a slight decrease as local capability improves, remaining the least efficient option among the algorithms evaluated.

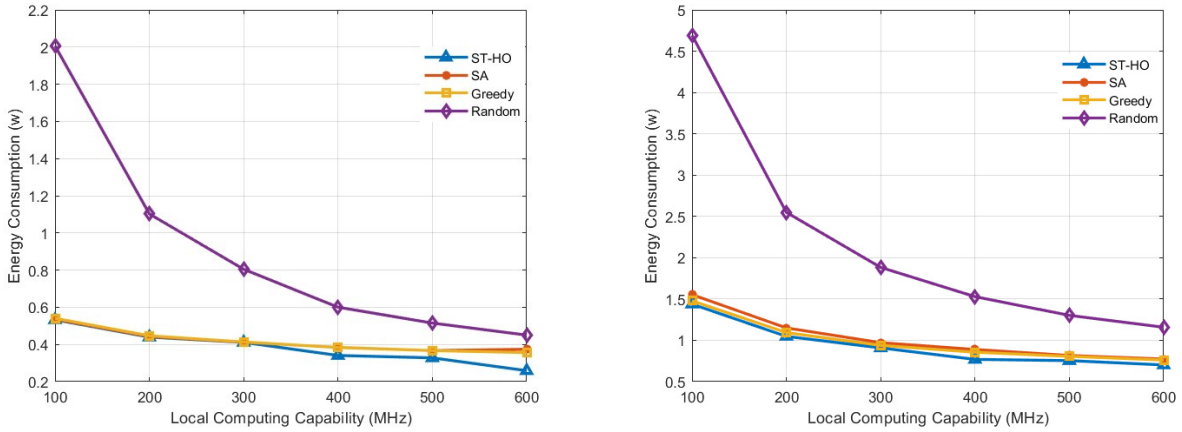

Figure 8. Changes in system energy consumption as the local computing capability varies (**a**) when the DAG structure task number is 11 and (**b**) when the DAG structure task number is 30.

Figure 9 shows the performance of ST-HO compared to the SA, Greedy and Random algorithms with different communication rates between the vehicle and MEC server in both smaller- and larger-scale tasks. Figure 9a illustrates that, for smaller tasks, the energy consumption of all the algorithms decreases with the increase in communication rate, which indicates that higher communication efficiency reduces the energy consumption. Notably, the ST-HO algorithm exhibits the lowest energy consumption across the entire communication rate, achieving a significant reduction in energy consumption of 23.6% as the communication rate increases. This indicates that the ST-HO algorithm is particularly responsive to the improvement in communication efficiency when the task size is small. The SA and Greedy algorithms also show notable reductions in energy consumption, but they still consume more energy compared to the ST-HO algorithm. The Random algorithm, while showing a decreasing trend in energy consumption with higher communication rates, remains the least efficient overall. Compared to other algorithms, the ST-HO algorithm shows a 5.5% improvement over the Greedy algorithm, 16% over the SA algorithm, and 23.5% over the Random algorithm in terms of energy consumption reduction. Figure 9b shows that for larger task sizes, the ST-HO algorithm maintains its strong performance, with a significant decrease in energy consumption as the communication rate increases, demonstrating good scalability. The energy consumption of all algorithms is higher for larger tasks than for smaller ones, suggesting that increased task size leads to a higher communication load and consequently affects the overall system energy consumption. Despite this, the ST-HO algorithm still maintains the lowest energy consumption under larger task sizes. The SA and Greedy algorithms show a similar trend in energy reduction, but the Greedy algorithm's energy reduction is more consistent than the SA algorithm's at high communication rates. The Random algorithm still performs poorly with larger task sizes, where it consistently consumes the highest amount of energy. Overall, the ST-HO algorithm improves by 5% over the Greedy algorithm, 6% over the SA algorithm, and 48% over the Random algorithm in terms of reduced energy consumption.

In addition, by analyzing the symmetric relationship shown in the DAG task graph, our approach simplifies the offloading strategy's computation. Specifically, we only consider tasks that represent half of the symmetric relationship. The offloading strategy for the other half is not recalculated, but is directly replicated from the offloading strategy of the corresponding symmetric task.

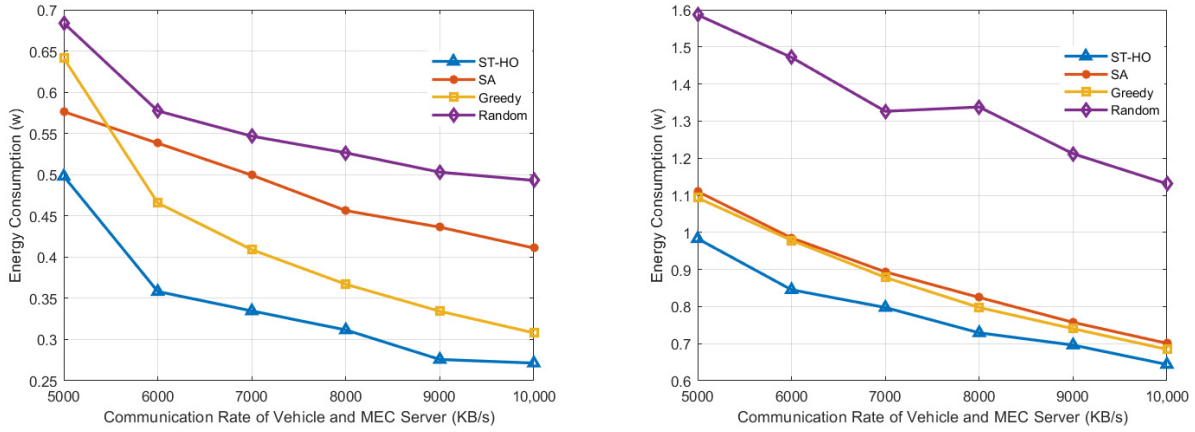

Figure 9. Changes in system energy consumption as the communication rate of the vehicle and the MEC server varies (**a**) when the DAG structure task number is 11 and (**b**) when the DAG structure task number is 30.

Figure 10 shows the energy consumption performance based on the replication offloading strategy for symmetric tasks. The ST-HO algorithm is compared with its variant, the ST-HO-Symmetry, which utilizes the replication offloading strategy for symmetric tasks. The results demonstrate that although the ST-HO-Symmetry algorithm has lower space complexity by considering only half of the symmetric tasks, its performance remains robust. The offloading strategy for the other half of the tasks, once replicated, effectively completes the offloading process for the entire task. The ST-HO-Symmetry algorithm converges faster and has a more guaranteed performance. In terms of adaptability, the ST-HO-Symmetry algorithm is comparable to the ST-HO algorithm that performs the overall computations for the offloading strategy. This comparison validates the effectiveness of the replication strategy for symmetric tasks in terms of computation efficiency and energy optimization.

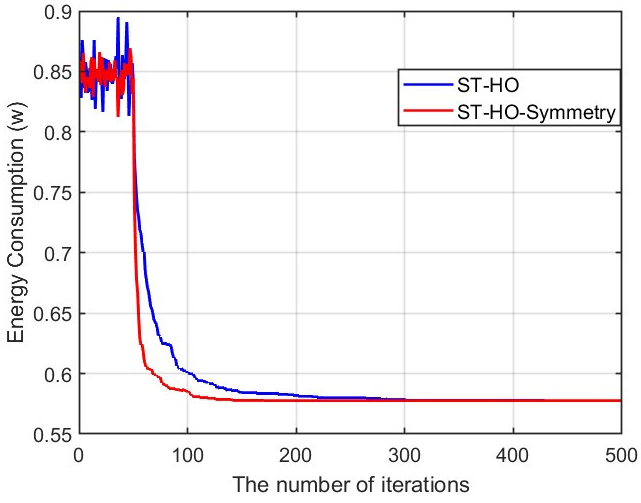

Figure 10. Performance of replicating offloading strategy in symmetric tasks.

In summary, in small-scale tasks, the ST-HO algorithm decreased its energy consumption by an average of 17.58% compared to the Greedy algorithm, 34% compared to the Simulated Annealing algorithm, and 43.74% compared to the Random algorithm under the variation of the computing power of the MEC server. Under the variation of local computing power, the ST-HO algorithm showed an average decrease in energy consumption of 9.80% over the Greedy algorithm, 9.97% over the SA algorithm, and 51.27% over the Random algorithm. In the case of different communication rates between the vehicle and

the MEC server, the ST-HO algorithm decreased its energy consumption by 19.76% on average compared to the Greedy algorithm, 31.74% compared to the SA algorithm, and 40.39% compared to the Random algorithm.

In large-scale tasks, the ST-HO algorithm shows an average decrease in energy consumption of 10.28% over the Greedy algorithm, 12.43% over the more SA algorithm, and 45.36% over the Random algorithm under variations in the computing power of the MEC server. Under variations in local computing power, the ST-HO algorithm shows an average decrease in energy consumption of 5.56% compared to the Greedy algorithm, 8.51% compared to the SA algorithm, and 53.32% compared to the Random algorithm. With different communication rates between the vehicle and the MEC server, the ST-HO algorithm decreases energy consumption by an average of 8.88% compared to the Greedy algorithm, 10.25% compared to the SA algorithm, and 41.66% compared to the Random algorithm.

5. Conclusions

With the development of communications technologies and computer applications, ITS has been merged with MSCC synergy in a task-oriented manner in IoV. This paper studies the offloading strategy of the vehicle MSCC task represented by the DAG structure, and proposes an optimized energy-efficient computational offloading strategy for executing such tasks under time constraints. Based on the temporal dependencies of DAG structure subtasks and the constraints of local computation, we establish an energy consumption model for executing DAG tasks and design the ST-HO algorithm to address this optimization. Simulation results indicate that the ST-HO algorithm significantly reduces energy consumption during task execution, while guaranteeing the running speed. Especially in large-scale tasks, the ST-HO algorithm achieves at least 5.56% improvement in energy reduction. Moreover, the replication offloading strategy of symmetric tasks has been explored, showing a 52.63% improvement in convergence speed, while ensuring performance.

Our future work will expand task offloading beyond MEC, leveraging underutilized vehicular computational resources in the V2X network to achieve a robust network topology and reliable task execution. Additionally, a key area of enhancement will be incentivizing the collaborative use of distributed vehicular resources, which has the potential to revolutionize task offloading strategies and further advance ITS applications.

Author Contributions: Conceptualization, Z.G.; methodology, Z.G.; software, G.L.; validation, Z.G.; formal analysis, G.L. and B.L.; investigation, G.L. and S.Z.; resources, G.L. and B.L.; data curation, G.L.; writing—original draft preparation, G.L.; writing—review and editing, S.Z.; visualization, H.-C.C.; supervision, L.H.; project administration, Z.G.; funding acquisition, L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 62371406, 61971365), Key Science and Technology Project of Fujian Province (No. 2021H6001), and Xiamen Major Science and Technology Project (No. 3502Z20221026).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Garg, T.; Kaur, G. A Systematic Review on Intelligent Transport Systems. *J. Comput. Cogn. Eng.* **2022**, *2*, 175–188. [CrossRef]
2. Sharma, S.; Kaushik, B. A Survey on Internet of Vehicles: Applications, Security Issues & Solutions. *Veh. Commun.* **2019**, *20*, 100182. [CrossRef]
3. Ji, B.; Zhang, X.; Mumtaz, S.; Han, C.; Li, C.; Wen, H.; Wang, D. Survey on the Internet of Vehicles: Network Architectures and Applications. *IEEE Commun. Stand. Mag.* **2020**, *4*, 34–41. [CrossRef]
4. Hao, L.; Le, C.; Ting, P.; Qing, D.; Jinag, Z. A Fast Algorithm for Energy-Saving Offloading with Reliability and Latency Requirements in Multi-Access Edge Computing. *IEEE Access* **2020**, *8*, 151–161. [CrossRef]
5. Liu, Y.; Peng, M.; Shou, G.; Chen, Y.; Chen, S. Towards Edge Intelligence: Multi-access Edge Computing for 5G and Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 6722–6747. [CrossRef]
6. Zhang, L.; Cao, B.; Li, Y.; Peng, M.; Feng, G. A Multi-Stage Stochastic Programming-Based Offloading Policy for Fog Enabled IoT-eHealth. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 411–425. [CrossRef]

7. Lin, B.; Zhou, X.; Duan, J. Dimensioning and Layout Planning of 5G-Based Vehicular Edge Computing Networks Towards Intelligent Transportation. *IEEE Open J. Veh. Technol.* **2020**, *1*, 146–155. [CrossRef]
8. Guo, H.; Liu, J.; Ren, J.; Zhang, Y. Intelligent Task Offloading in Vehicular Edge Computing Networks. *IEEE Wirel. Commun.* **2020**, *27*, 126–132. [CrossRef]
9. Moubayed, A.; Shami, A.; Heidari, P.; Larabi, A.; Brunner, R. Edge-Enabled V2X Service Placement for Intelligent Transportation Systems. *IEEE Trans. Mob. Comput.* **2021**, *20*, 1380–1392. [CrossRef]
10. Wang, S.; Xin, N.; Luo, Z.; Lin, T. An Efficient Computation Offloading Strategy Based on Cloud-Edge Collaboration in Vehicular Edge Computing. In Proceedings of the 2022 International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), Xiamen, China, 5–7 August 2022; pp. 193–197. [CrossRef]
11. Yuan, S.; Zhao, H.; Geng, L. An Offloading Algorithm Based on Deep Reinforcement Learning for UAV-Aided Vehicular Edge Computing Networks. In Proceedings of the 2022 IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)/2022 IEEE 8th International Conference on Edge Computing and Scalable Cloud (EdgeCom), Xi'an, China, 25–27 June 2022; pp. 153–159. [CrossRef]
12. Dai, W. Joint Task Offloading, Resource Allocation and Data Caching in MEC-Assisted Vehicular Network. In Proceedings of the 2023 4th International Conference on Computer Engineering and Application (ICCEA), Hangzhou, China, 7–9 April 2023; pp. 70–76. [CrossRef]
13. *3GPP TS 23.287 V17.6.0*; Architecture Enhancements for 5G System (5GS) to Support Vehicle-to-Everything (V2X) Services (Release 17). ETSI: Sophia Antipolis, France, 2023.
14. Li, X.; Dang, Y.; Aazam, M.; Peng, x.; Cheng, T.; Chen, C. Energy-Efficient Computation Offloading in Vehicular Edge Cloud Computing. *IEEE Access* **2020**, *7*, 37632–37644. [CrossRef]
15. Cho, H.; Cui, Y.; Lee, J. Energy-Efficient Computation Task Splitting for Edge Computing-Enabled Vehicular Networks. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
16. Wang, L.; Zhou, W.; Xu, H.; Li, L.; Cai, L.; Zhou, X. Research on Task Offloading Optimization Strategies for Vehicular Networks based on Game Theory and Deep Reinforcement Learning. *Front. Phys.* **2023**, *11*, 1292702. [CrossRef]
17. Shen, X.; Chang, Z.; Xie, X.; Niu, S. Task Offloading Strategy of Vehicular Networks Based on Improved Bald Eagle Search Optimization Algorithm. *Appl. Sci.* **2022**, *12*, 9308. [CrossRef]
18. Deng, T.; Chen, Y.; Chen, G.; Yang, M.; Du, L. Task Offloading Based on Edge Collaboration in MEC-Enabled IoV Networks. *J. Commun. Netw.* **2023**, *25*, 197–207. [CrossRef]
19. Feng, L.; Li, W.; Lin, Y.; Zhu, L.; Guo, S.; Zhen, Z. Joint Computation Offloading and URLLC Resource Allocation for Collaborative MEC Assisted Cellular-V2X Networks. *IEEE Access* **2020**, *8*, 24914–24926. [CrossRef]
20. Chen, J.; Yang, Y.; Wang, C.; Zhang, H.; Qiu, C.; Wang, X. Multitask Offloading Strategy Optimization Based on Directed Acyclic Graphs for Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 9367–9378. [CrossRef]
21. Cui, Y.; Zhang, D.; Zhang, T.; Yang, P.; Zhu, H. A Multi-User Fine-Grained Task Offloading Scheduling Approach of Mobile Edge Computing. *Acta Electonica Sin.* **2021**, *49*, 2202–2207. [CrossRef]
22. Liu, Z.; Liwang, M.; Hosseinalipour, S.; Dai, H.; Gao, Z.; Huang, L. RFID: Towards Low Latency and Reliable DAG Task Scheduling Over Dynamic Vehicular Clouds. *IEEE Trans. Veh. Technol.* **2023**, *72*, 12139–12153. [CrossRef]
23. Gao, Z.; Liwang, M.; Hosseinalipour, S.; Dai, H.; Wang, X. A Truthful Auction for Graph Job Allocation in Vehicular Cloud-Assisted Networks. *IEEE Trans. Mob. Comput.* **2022**, *21*, 3455–3469. [CrossRef]
24. Liwang, M.; Gao, Z.; Wang, X. Let's Trade in the Future! A Futures-Enabled Fast Resource Trading Mechanism in Edge Computing-Assisted UAV Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3252–3270. [CrossRef]
25. Lang, P.; Tian, D.; Duan, X.; Zhou, J.; Sheng, Z.; Leung, V.C. Blockchain-Based Cooperative Computation Offloading and Secure Handover in Vehicular Edge Computing Networks. *IEEE Trans. Intell. Veh.* **2023**, *8*, 3839–3853. [CrossRef]
26. Asad, M.; Shaukat, S.; Javanmardi, E.; Nakazato, J.; Bao, N.; Tsukada, M. Secure and Efficient Blockchain-Based Federated Learning Approach for VANETs. *IEEE Internet Things J.* **2023**. [CrossRef]
27. Wang, Z.; Nakazato, J.; Asad, M.; Javanmardi, E.; Tsukada, M. Overcoming Environmental Challenges in CAVs through MEC-based Federated Learning. In Proceedings of the 2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN), Paris, France, 4–7 July 2023; pp. 151–156. [CrossRef]
28. Shu, C.; Zhao, Z.; Han, Y.; Min, G.; Duan, H. Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach. *IEEE Internet Things J.* **2020**, *7*, 1678–1689. [CrossRef]
29. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
30. Pratap, A.; Misra, R.; Das, S.K. Maximizing Fairness for Resource Allocation in Heterogeneous 5g Networks. *IEEE Trans. Mob. Comput.* **2021**, *20*, 603–619. [CrossRef]