electronics

MDPI

Article

Research on Offloading Strategy for Mobile Edge Computing Based on Improved Grey Wolf Optimization Algorithm

Wenzhu Zhang and Kaihang Tuo *

School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China
* Correspondence: tkh@xauat.edu.cn

Abstract: With the development of intelligent transportation and the rapid growth of application data, the tasks of offloading vehicles in vehicle-to-vehicle communication technology are continuously increasing. To further improve the service efficiency of the computing platform, energy-efficient and low-latency mobile-edge-computing (MEC) offloading methods are urgently needed, which can solve the insufficient computing capacity of vehicle terminals. Based on an improved gray-wolf algorithm designed, an adaptive joint offloading strategy for vehicular edge computing is proposed, which does not require cloud-computing support. This strategy first establishes an offloading computing model, which takes task computing delays, computing energy consumption, and MEC server computing resources as constraints; secondly, a system-utility function is designed to transform the offloading problem into a constrained system-utility optimization problem; finally, the optimal solution to the computation offloading problem is obtained based on an improved gray-wolf optimization algorithm. The simulation results show that the proposed strategy can effectively reduce the system delay and the total energy consumption.

Keywords: mobile edge computing; vehicular edge computing; offloading strategy; gray-wolf optimization

Citation: Zhang, W.; Tuo, K. Research on Offloading Strategy for Mobile Edge Computing Based on Improved Grey Wolf Optimization Algorithm. *Electronics* **2023**, *12*, 2533. https://doi.org/10.3390/electronics12112533

Academic Editor: Martin Reisslein

Received: 5 May 2023
Revised: 31 May 2023
Accepted: 1 June 2023
Published: 4 June 2023

1. Introduction

With the growth of consumer services and the progress of science and technology, the digitalization level of the automotive industry is becoming increasingly high. Multimedia, assisted driving, and other intelligent options are gradually becoming the new normal of automotive technology [1]. The Internet of Vehicles is the application of the Internet of Things in the field of intelligent transportation, and it is also an important component of intelligent transportation systems [2,3]. In order to achieve the functions of autonomous driving and the intelligent form of cars, we need to equip various sensors and communication methods inside and outside the car. However, these growing functional requirements have approached or even exceeded the computing power limit of the vehicle itself. Vehicles themselves are no longer able to meet the requirements of substantial data volume and low-latency sensitivity in vehicle-to-vehicle communication technology. Therefore, edge-computing-based vehicle offloading technology has increasingly become a focus of research among researchers [4].

Mobile cloud-computing technology [5] is an effective means to solve the limited resources of vehicle equipment. Although cloud-computing servers have powerful computing capabilities, the transmission delay of large amounts of data is long due to the distance from the task terminal, and data transmission is greatly affected by environmental fluctuations, which also consume substantial energy. These factors make it difficult for cloud computing to be practically applied. In order to solve these problems, mobile edge computing (MEC) has emerged [6]. Mobile edge computing extends cloud-computing capabilities to the network edge, enabling it to perform tasks that traditional network infrastructure cannot accomplish [7,8]. By installing edge servers or base stations close to the

terminal devices, MEC reduces the delay and energy consumption caused by transmission, which can significantly improve user computing power and provide cloud-computing-like capabilities for end users [9,10].

Compared to resource-rich cloud-computing centers, the computing resources of MEC servers in MEC are very limited. When an edge-computing offloading platform needs to offload multiple tasks, the MEC server may not be able to handle all computing tasks simultaneously, which may result in significant computation latency [11,12]. Therefore, it is very important to study the decision-making process of computation offloading [13] and resource allocation [14] in the Internet of Vehicles. Usually, computing offloading decisions and resource allocation are NP-Hard problems [15], and these problems are generally solved using swarm-intelligence optimization algorithms. Many scholars have also conducted research on such problems. Against this backdrop, reference [16] proposed a convex optimization-based algorithm that can calculate the optimal power allocation for scenarios where the upload delay and co-channel interference of differentiated users interact with each other. The authors determined the nonorthogonal multiple-access user pairing and offloading decision via semidefinite relaxation and convex–concave iterations, thereby mitigating interference and reducing the average task response delay. Reference [17] proposed a linearization-based branch-and-bound algorithm that can solve time-varying spectral-efficiency problems caused by time-varying fading channels without considering their time-varying characteristics. To address the complexity of the algorithm, reference [17] proposed an algorithm that is closest to the rounding-integer algorithm. This algorithm can achieve the maximization of utility under the constraint of task delay requirements, and it can study the impact of time-varying channels on task offloading strategies within the task-offloading period. Reference [18] proposed a task-offloading scheme based only on vehicle-to-vehicle communication, utilizing the gathering period of vehicles in urban environments induced by traffic signals or areas of interest. The authors formulated the problem of a single task with multiple collaborative offloading modes as a minimax problem and further optimized the convergence speed and accuracy using the particle-swarm optimization (PSO) algorithm. Reference [19] focused on the impact of task-offloading decisions and edge-server execution orders on computing-offloading services. The authors investigated the offloading-decision problem for the efficient optimization of task latency and computation resource consumption in a multiuser, multiserver vehicular-edge-computing scenario, and they proposed a hybrid intelligent optimization algorithm based on a single genetic algorithm and heuristic rules. Reference [20] considered using idle resources in volunteer vehicles to process tasks in a vehicle-mounted edge server and reduce costs. The authors proposed a fast search algorithm based on a genetic algorithm to solve the pricing problem, and they observed the optimal offloading strategy based on the Stackelberg game. Reference [21] proposed a new hybrid metaheuristic algorithm of genetic simulated annealing and PSO. This algorithm aims to minimize the total energy consumption of both intelligent mobile devices and edge servers by jointly optimizing the task-offloading ratio, CPU speed, bandwidth allocation of available channels, and transmission power of each terminal device in each time slot. Reference [22] proposed a joint task-offloading and resource-allocation scheme based on V2I and V2V modes to minimize the total taken processing delay for all vehicles. The scheme takes into account task diversity, vehicle classification, and task-processing flexibility, and designed an algorithm based on generalized benders decomposition and reformed linearization method to optimally solve the optimization problem. Reference [23] proposed a task-offloading and resource-allocation scheme based on priority clustering to handle heterogeneous tasks and ensure quality of service. The scheme first assigned priorities to tasks based on their characteristics and then clustered binary tuples composed of task priorities and vehicle-edge distances to determine task-offloading strategies. Finally, the scheme utilized Lagrange multipliers to optimize the resource allocation of explored feasible strategies. Reference [24] proposed a task-offloading allocation scheme based on the Stackelberg game to improve the efficiency of vehicle-edge computing. The scheme modeled the competition and cooperation among

vehicles, edge servers, and the cloud as a Stackelberg game and used backward induction to transform the game problem into a convex optimization problem. This theoretically proved that the game had a unique Nash equilibrium, thus obtaining the optimal task allocation for the requesting vehicles. Additionally, a search algorithm based on genetic algorithms was proposed to find the optimal pricing scheme for edge servers and the cloud.

The above studies in the literature mostly approached the vehicle offloading problem of different scenarios and used different algorithms for solving it. Currently, there is limited research on concurrent offloading of multiple vehicles considering latency and energy consumption when cloud-computing servers are scarce. Furthermore, the computational platforms are not fully utilized, and the effectiveness of the proposed algorithms is not significantly improved. The gray-wolf optimization (GWO) algorithm is a swarm-intelligence algorithm inspired by the hunting behavior and social hierarchy of gray wolves. Compared to other intelligent optimization algorithms, it has the following characteristics: relatively simple structure, fewer adjustable parameters, and ease of implementation. The algorithm has a convergence factor and an information feedback mechanism, and it has good performance in both solution accuracy and convergence speed. The GWO algorithm has been applied to solve problems such as array sorting [25], scheduling [26], and load control [27]. However, its application in the field of MEC has not received substantial attention from scholars. Therefore, the GWO algorithm may provide a new solution approach for the vehicle task offloading and allocation problem.

Based on the above analysis, this paper proposes an edge-computing offloading strategy based on an improved GWO. The main arrangement of this paper is as follows.

(1) This paper establishes a model for computing latency and energy consumption in the context of MEC, and it uses task computing latency, computing energy consumption, and MEC server-computing resources as constraints. We designed a system-utility function to evaluate the computing offloading strategy and convert the offloading problem into a constrained system-utility optimization problem.
(2) Based on the gray-wolf optimizer algorithm, this paper introduces the whale optimization algorithm (WOA) and the levy flight algorithm to improve the population initialization and alpha-wolf selection steps of the GWO algorithm. Finally, we realize the optimization solution to the computation-offloading-strategy problem.
(3) This paper designs experiments to verify the effectiveness of the edge computing offloading strategy based on the hybrid gray-wolf–whale algorithm. We evaluate and analyze the performance of the designed edge computing offloading strategy from the aspects of computation delay, computation energy consumption, and convergence.

The remaining chapters of this paper are organized as follows. The second section introduces the system model, communication model, computing models, etc. The third section introduces the MEC offloading method on the Internet of Vehicles environment. Section 5 carries out the simulation experiment design and obtains results from the analysis of the method in this paper. Section 5 concludes with a comprehensive summary of this paper.

2. System Model

2.1. Network Model

The deployment scenario of the MEC system studied in this paper is shown in Figure 1. The system includes multiple mobile vehicles and MEC servers. The computing tasks that need to be offloaded are initiated by the corresponding mobile vehicles. The computing methods for tasks include three types, namely, local computing, offloading to idle vehicles for computing, and offloading to edge servers for computing. After the latter two computing methods are completed, the computing results will be returned to the task vehicle. Roadside edge servers have abundant computing resources, but the distance between the vehicles and roadside edge servers is uncertain, resulting in high communication latency and related energy consumption for the vehicles. In contrast, idle vehicles have relatively

weak computing capabilities, but the communication latency between the task vehicle and the constrained vehicle is lower.

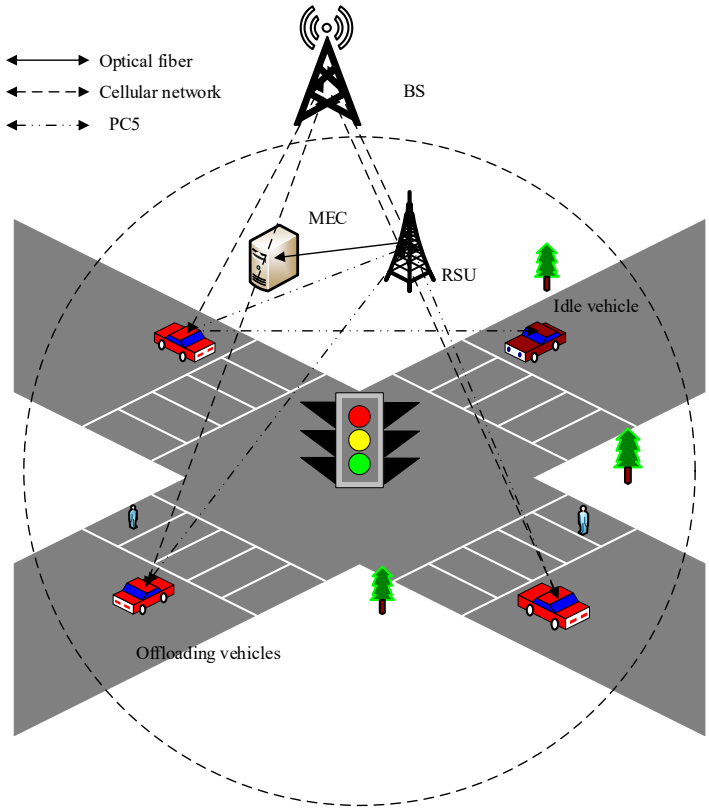

Figure 1. System model.

Assuming that multiple vehicles unload tasks at the same time, this paper defines the set of vehicles unloading computing tasks under the base station coverage, such as $V = \{v_1, v_2, \cdots, v_n\}$. Each task vehicle has its task, which is defined as T. The set of idle computing resource vehicles around the task vehicle is defined as $D = \{\mathrm{d}_1, \mathrm{d}_2, \cdots, \mathrm{d}_x\}$. The upload channel model for task vehicles is the Rayleigh channel model [28].

The transmission rates between vehicle v_i and idle vehicles or MEC can be shown in Equation (1):

$$R_i = B \log_2 \left(1 + \frac{P_i^{up} h_i}{B N_0} \right) \tag{1}$$

where P_i^{up} represents the transmission power of vehicle v_i, h_i represents the channel gain between the task vehicle and the edge server or idle vehicle, B represents the channel bandwidth of the vehicle, and N_0 represents the background channel noise power.

2.2. Computational Offloading Model

Assuming that the set of tasks to be offloaded by mobile vehicles is defined as $Task = \{T_1, T_2, \cdots, T_n\}$, each task has four properties: $T_i = \{data_i, \rho, f_i, t_{\max.i}\}$. $data_i$ represents the amount of data to be offloaded for task i; ρ represents the computational intensity of task i, which is the number of CPU cycles required to compute 1 bit of data; f_i represents the computing capability of vehicle v_i; and $t_{\max,i}$ represents the maximum delay time required to complete task i. In addition, the computing power of the edge server is denoted as f_{MEC}, the allocated computing resource for vehicle v_i is denoted as $f_{MEC,i}$, and the computing power provided by the surrounding idle vehicles is denoted as f_{idle}.

Assuming that the task is divisible, the task can be offloaded to the edge server, nearby idle vehicles, and locally for computation. Each task can be partially or completely offloaded

to any computing device. Therefore, it is necessary to partition the task and determine the partition size and location. The set of all offloading decisions on task vehicles is denoted as set $X = \{x_1, x_2, \cdots, x_n\}$, where $x_i = [a_i^1, a_i^2, a_i^3]$, $a_i^1, a_i^2, a_i^3 \in [0,1]$ and $a_i^1 + a_i^2 + a_i^3 = 1$. a_i^1, a_i^2 and a_i^3, respectively, represent the proportion of task i that is offloaded to local, nearby idle vehicles, and edge servers. If the unloading ratio of a certain task is $a_i^x = 0$ ($i \in \{1, 2, 3\}$), this means that the task will not be unloaded to the corresponding computing device. When the unloading ratio of a task is equal to one, this means that the mobile device performs the computation entirely on the corresponding computing device.

2.2.1. Local Computing

The amount of computation and time required for task i to be computed on a local vehicle depends on the computing power of the task vehicle itself. The amount of task data allocated for computation is denoted as $a_i^1 \times data_i$, and the corresponding execution delay and energy consumption are $t_{local,i}$ and $e_{local.i}$, and the corresponding equations are shown in Equations (2) and (3). Task i does not require data transmission during local vehicle computing, so there is no transmission delay and only computation delay.

$$t_{local,i} = \frac{a_i^1 \times data_i \times \rho}{f_i} \tag{2}$$

$$e_{local.i} = P_i \times t_i^{local} \tag{3}$$

where P_i represents the device power of task vehicle v_i.

2.2.2. Idle-Vehicle Computing

Task i's computing-task data size of idle vehicles is represented by $a_i^2 \times data_i$, the time delay for idle vehicles to perform the unloading task is $t_{idle.i}^{exe}$, the transmission delay for the task to be unloaded to idle vehicles is $t_{idle.i}^{trans}$ (vehicles may not be able to communicate directly with each other), and the average delay of relaying between vehicles is t_{r}. λ is an output data coefficient, representing the relationship between output data volume and input data volume. The calculation's resulting data transmission time is $t_{idle.i}^{back}$. The total time delay for the task vehicle to transmit to the idle vehicle is $t_{idle,i}$, and the total energy consumption is $e_{idle.i}$. The corresponding expressions for $t_{idle.i}^{exe}$, $t_{idle.i}^{trans}$, $t_{idle.i}^{back}$, $t_{idle,i}$, and $e_{idle.i}$ are shown in Equations (4)–(8).

$$t_{idle.i}^{exe} = \frac{a_i^2 \times data_i \times \rho}{f_{idle}} \tag{4}$$

$$t_{idle.i}^{trans} = \frac{a_i^2 \times data_i}{R_i} + t_{\mathrm{r}} \tag{5}$$

$$t_{idle.i}^{back} = \frac{a_i^2 \times data_i \times \lambda}{R_i} \tag{6}$$

$$t_{idle,i} = t_{idle,i}^{trans} + t_{idle,i}^{exe} + t_{idle,i}^{back} \tag{7}$$

$$e_{idle.i} = P_{idle} \times t_{idle,i}^{exe} + P_i^{up} \times \left(t_{idle,i}^{trans} + t_{idle,i}^{back} \right) \tag{8}$$

where P_i^{up} is the upload power of the task vehicle v_i, and P_{idle} represents the device power of the idle vehicle.

2.2.3. Edge-Server Computing

For task i, the allocation of offloading computing data to the edge server depends on the busyness of the server. Let the computation data amount on the edge server be

denoted as $a_i^3 \times data_i$, the offloading task-execution delay on the edge server be $t_{MEC.i}^{exe}$, the transmission delay for offloading the task to the edge server be $t_{MEC.i}^{trans}$, the transmission delay of returning computation results be $t_{MEC.i}^{back}$, the total transmission delay from the task vehicle to the idle vehicle be $t_{MEC,i}$, and the total energy consumption be $e_{MEC.i}$. The corresponding expressions for $t_{MEC.i}^{exe}$, $t_{MEC.i}^{trans}$, $t_{MEC.i}^{back}$, $t_{MEC,i}$, and $e_{MEC.i}$ are shown in Equations (9)–(13).

$$t_{MEC.i}^{exe} = \frac{a_i^3 \times data_i \times \rho}{f_{MEC,i}} \tag{9}$$

$$t_{MEC.i}^{trans} = \frac{a_i^3 \times data_i}{R_i} \tag{10}$$

$$t_{MEC.i}^{back} = \frac{a_i^3 \times data_i \times \lambda}{R_i} \tag{11}$$

$$t_{MEC,i} = t_{MEC,i}^{trans} + t_{MEC,i}^{exe} + t_{MEC,i}^{back} \tag{12}$$

$$e_{MEC.i} = P_{MEC} \times t_{MEC,i}^{exe} + P_i^{up} \times \left(t_{MEC,i}^{trans} + t_{MEC,i}^{back} \right) \tag{13}$$

where P_{MEC} represents the device power of the edge server.

3. Improving the Gray-Wolf Algorithm

This article models the task offloading of multiple vehicles and the allocation of MEC computing resources as a multiconstraint optimization problem. Based on the above models, total task computation delay t_{sum} and energy consumption e_{sum} in the edge computing system are represented as shown in Equations (14) and (15).

$$t_{sum} = \sum_{i=1}^{n} t_{local.i} + t_{idle,i} + t_{MEC,i} \tag{14}$$

$$e_{sum} = \sum_{i=1}^{n} e_{local.i} + e_{idle,i} + e_{MEC,i} \tag{15}$$

To achieve the joint optimization of computation time and energy consumption in a collaborative offloading mode between edge servers, idle vehicles, and local devices, this paper designs a system-utility function Q to evaluate the offloading effectiveness of tasks. Q represents the total cost of the system. In this case, t_{sum} and e_{sum} represent the total computation time delay and energy consumption when all tasks are completed, and η and θ ($\eta + \theta = 1$) represent the weight of computation time delay and energy consumption in the system's utility function, respectively. η and θ can be set according to the service demand and status of the mobile vehicles. Therefore, the offloading optimization model can be described as shown in Equations (16) and (17).

$$Q = \eta \times t_{sum} + \theta \times e_{sum} \tag{16}$$

$$s.t. \begin{cases} \text{a} : a_i^1, a_i^2, a_i^3 \in [0,1], \forall i \in [1,n] \\ \text{b} : a_i^1 + a_i^2 + a_i^3 = 1, \forall i \in [1,n] \\ \text{c} : 0 \le f_{MEC,i} \le f_{MEC}, \forall i \in [1,n] \\ \text{d} : \sum_{i=1}^{n} f_{MEC,i} \le f_{MEC}, \forall i \in [1,n] \\ \text{e} : f_{idle,i} \le f_{idle}, \forall i \in [1,n] \end{cases} \tag{17}$$

Constraint conditions (17a) and (17b) indicate the sum of computation task data: task i is offloaded to the local, idle vehicles, and edge servers and equals the entire computation-task data of the task vehicle. Constraints (17c) and (17d) indicate that the allocated comput-

ing resources to each task on the edge computing processor do not exceed its upper limit, and the total allocated resources to all tasks do not exceed the computing resources of the edge device itself. Constraint (17e) indicates that the computing resources allocated to each task on the idle vehicle do not exceed its own computing resources.

In order to solve the proposed unloading model and optimization objective function faster, this paper proposes an algorithm inspired by the gray-wolf optimization algorithm, improved for its shortcomings. The following content is the improvement steps.

3.1. Gray-Wolf Optimization Algorithm

The gray-wolf optimizer is a swarm-intelligence algorithm inspired by the hunting behavior and social hierarchy of gray wolves in the natural world. Gray wolves typically live in a social hierarchy with strict dominance levels. Wolf packs can be divided into four categories: α, β, δ, and ω. In the GWO algorithm, the best solution is regarded as α, the second-best and third-best solutions are regarded as β and δ, respectively, and the rest of the solutions are regarded as ω. The hunting behavior of gray wolves is abstracted into three stages in the algorithm: searching for prey, surrounding prey, and attacking prey. However, the basic gray-wolf optimizer algorithm often struggles to escape local optimal solutions when solving complex problems, resulting in long computation times, poor convergence, and suboptimal optimization results. Therefore, in this paper, we adopt an improved GWO algorithm to solve the model. This algorithm can improve the unloading strategy, accelerate computation time, and reduce task computation delay and corresponding computing consumption, thereby improving the efficiency and accuracy of the solution.

3.2. Improved Gray-Wolf Optimization Algorithm

The WOA is a population-based metaheuristic algorithm that mimics the hunting behavior of humpback whales for problem optimization. This section introduces an improved method that incorporates the WOA into the gray-wolf optimization algorithm, called the hybrid gray-wolf optimization with the whale optimization algorithm (HGWOWOA). This algorithm can improve the global optimal search ability and avoid becoming trapped in locally optimal solutions.

3.2.1. Improvements Incorporated into the Whale Algorithm

To prevent the gray-wolf optimization algorithm from becoming trapped in local optima, we have made the following improvements. First, we introduced the spiral mesh hunting behavior and random probability factor from the WOA into the GWO algorithm and compared them for selection. Second, we introduced the Levy flight algorithm to extend the algorithm's search range with random step lengths, thereby enhancing the algorithm's search ability and diversifying the gray-wolf population. The specific improvement methods are shown as follows.

First, $\boldsymbol{Levy}(d)$ is generated according to the random-step formula proposed by Mantegna [29], as shown in Equation (18).

$$\boldsymbol{Levy}(d) = \frac{u}{|v|^{1/d}} \tag{18}$$

Parameter d is usually a constant within the range of [0, 2]; here, we take it as 1.5. u and v follow the distribution of $u \sim N(0, \sigma^2)$ and $v \sim N(0, 1)$, where the expression for σ is shown in Equation (19).

$$\sigma = \left[\frac{\Gamma(1+d) \times \sin(\frac{d \times \pi}{2})}{d \times \Gamma(\frac{1+d}{2}) \times 2^{\frac{d-1}{2}}} \right]^{1/d} \tag{19}$$

The improved gray-wolf optimization algorithm introduces a random probability factor, denoted as p, $p \in [0, 1]$. When $p \geq 0.5$, the wolf population updates their positions based on the optimal solution's position, utilizing both the Levy flight algorithm and spiral bubble-net hunting behavior, as shown in Equation (20).

$$\boldsymbol{X}'(t) = \boldsymbol{X}_{best}(t) + e^{vl} \times \cos(2\pi l) \times |\boldsymbol{X}_{best} - \boldsymbol{X}(t)| \oplus \boldsymbol{Levy}(d) \tag{20}$$

In the equation, $\boldsymbol{X}_{best}$ is the global best solution; $\boldsymbol{X}$ represents the position vector of the individual gray wolf; $\boldsymbol{X}_\alpha$, $\boldsymbol{X}_\beta$, and $\boldsymbol{X}_\delta$ represent the three types of leading wolves. The $\oplus$ symbol represents the dot product, and v represents a constant coefficient in the spiral equation (usually taken as l), and l is a random number within the range of $[-1, 1]$. In the improved GWO, when $p < 0.5$, the individual wolves in the algorithm are surrounded according to Equations (21)–(24), where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are randomly generated vectors within the range of [0, 1], $m = 2 - 2t/T$, t is the current iteration number, T is the maximum iteration number, and m linearly decreases from 2 to 0 as the iteration number increases.

$$\begin{cases} \boldsymbol{A} = 2 \times m \times \boldsymbol{n}_1 - m \\ \boldsymbol{C} = 2 \times \boldsymbol{n}_2 \end{cases} \tag{21}$$

$$\begin{cases} D_\alpha = |\boldsymbol{C}_1 \cdot \boldsymbol{X}_\alpha - \boldsymbol{X}(\mathrm{t})| \\ D_\beta = |\boldsymbol{C}_2 \cdot \boldsymbol{X}_\beta - \boldsymbol{X}(\mathrm{t})| \\ D_\delta = |\boldsymbol{C}_3 \cdot \boldsymbol{X}_\delta - \boldsymbol{X}(\mathrm{t})| \end{cases} \tag{22}$$

$$\begin{cases} \boldsymbol{X}_1(t+1) = \boldsymbol{X}_\alpha(t) - \boldsymbol{A}_1 \cdot D_a \\ \boldsymbol{X}_2(t+1) = \boldsymbol{X}_\beta(t) - \boldsymbol{A}_2 \cdot D_\beta \\ \boldsymbol{X}_3(t+1) = \boldsymbol{X}_\delta(t) - \boldsymbol{A}_3 \cdot D_\delta \end{cases} \tag{23}$$

$$\boldsymbol{X}(t+1) = \frac{1}{3}(\boldsymbol{X}_1(t+1) + \boldsymbol{X}_2(t+1) + \boldsymbol{X}_3(t+1)) \tag{24}$$

3.2.2. Location Update Improvements

When searching for the optimal solution, the position corresponding to the α wolf in the GWO algorithm may not necessarily be the global optimal solution, and other graded individuals of the gray wolf also have difficulty jumping out of their local optimal solutions. This situation can lead to a significant increase in processing time and a decrease in the convergence speed of the algorithm. To improve the solution accuracy and convergence speed of the GWO algorithm, this paper proposes a new proportional weight. Inspired by the custom weights, different individual gray wolves have different proportional effects on the optimal solution, and the expression of dynamic weights is shown in Equations (25) and (26).

$$\begin{cases} W_1 = \frac{|\boldsymbol{X}_1|}{|\boldsymbol{X}_1| + |\boldsymbol{X}_2| + |\boldsymbol{X}_3|} \\ W_2 = \frac{|\boldsymbol{X}_2|}{|\boldsymbol{X}_1| + |\boldsymbol{X}_2| + |\boldsymbol{X}_3|} \\ W_3 = \frac{|\boldsymbol{X}_3|}{|\boldsymbol{X}_1| + |\boldsymbol{X}_2| + |\boldsymbol{X}_3|} \end{cases} \tag{25}$$

$$\boldsymbol{X}'(t) = W_1 \cdot (\boldsymbol{X}_1 - \boldsymbol{X}_\alpha) + W_2 \cdot (\boldsymbol{X}_1 - \boldsymbol{X}_\beta) + W_3 \cdot (\boldsymbol{X}_1 - \boldsymbol{X}_\delta) \tag{26}$$

3.2.3. Comparing Selection Strategies

To select the better result, the algorithm needs to compare the fitness values of $\boldsymbol{X}(t)$ and $\boldsymbol{X}'(t)$, keep the corresponding minimum fitness function values as the result of iteration $\boldsymbol{X}(t+1)$, and record the position of the minimum fitness function value. The chosen strategy is shown in Equation (27).

$$\begin{cases} \boldsymbol{X}(t+1) = \boldsymbol{X}'(t) & \textbf{\textit{if}}\ \mathrm{fitness}\left(\boldsymbol{X}'(t)\right) < \mathrm{fitness}(\boldsymbol{X}(t)) \\ \boldsymbol{X}(t+1) = \boldsymbol{X}(t) & \textbf{\textit{else}} \end{cases} \tag{27}$$

3.3. Individual Gray-Wolf Amendment

Each vehicle contains four parameters to be optimized, which are a_i^1, a_i^2, a_i^3, and $f_{MEC,i}$. Assuming that there are a vehicles that need to be offloaded, the encoding matrix of a gray wolf is $A \in \mathbb{R}^{a\times 4}$, where its first three columns represent the offloading ratios of the task vehicle to the local vehicles, nearby idle vehicles, and edge servers, respectively, and the fourth column represents the computing resources allocated to the current task vehicle by the MEC server. The entire wolf pack is stored in matrix B, where each wolf's encoding matrix A is first converted into a row and stored in matrix B. Here, $B \in \mathbb{R}^{C\times(a\times 4)}$, and C represents the size of the wolf population.

As the randomly initialized values of the gray-wolf population may not necessarily meet the constraints, corresponding improvements need to be made to the coding of individual gray wolves. Specifically, the sum of the first three values in each row of matrix A should be one, the sum of the values in the last column of matrix A should be less than one, and the number of nonzero values in the second column of matrix A should not exceed the current set number of idle vehicles b. To this end, we have designed a gray-wolf-individual correction algorithm to ensure that the encoding matrix satisfies the constraints. The description of the gray-wolf-individual correction algorithm is shown in Algorithm 1.

Algorithm 1 Gray-Wolf-Individual Correction Algorithm

Input: Matrix B, number of vehicles on task a, number of idle vehicles b, size of gray-wolf population C
Output: Gray-wolf-individual correction algorithm corrected calculation matrix B

```
for i = 1, 2, ... , C do
Take the ith row of matrix B and transform it into a matrix A with ax4;
    for j = 1, 2, ... , a do
        if Amount of tasks to offload to free vehicles > Local offload task volume then
            A(j,3) = A(j,1) × rand(0,1);
                /* Perform randomization */
        end if
      end for
        Select the b largest numbers from the 4th column of A and assign the rest to 0;
      for k = 1, 2, ... , a do
            if A(k,1) + A(k,3) > 1 then
              A(k,1) = (1 − A(k,3) ) × rand(0,1);
            end if
            A(k,2) = 1 − A(k,1) − A(k,4);
                /* Ensure that the unloading ratio sums to 1*/
    end for
    Add the 2nd column of matrix A to get sum
    for c = 1, 2, ... , a do
      A(c, 4) = A(c, 2)/sum;
    end for
    Transform the matrix A into 1 row and put it into row i of matrix B;
        /* Each line of B is an offloading possibility */
end for
Return B
```

3.4. Network Model

Based on the Sections 3.1–3.3, this paper proposes a hybrid algorithm, called the HGWOWOA. The algorithm is described as follows:

Step 1: Initialize algorithm parameters, including the number of wolves in the wolf pack, maximum number of iterations, dimension and value ranges of independent variables, and the generation of random computing-task information and the computing-capacity information of each device.

Step 2: Randomly generate the initial positions of the gray-wolf population and generate probability factors and random numbers. Set parameter b and calculate the corresponding $\boldsymbol{A}$ and $\boldsymbol{C}$.

Step 3: Compute the fitness value of each individual in the gray-wolf population based on the given fitness function and select the three individuals with the lowest fitness values as the leading three wolves.

Step 4: Depending on different probability factors, determine which search weight position formula to execute in order to obtain a new position.

Step 5: Calculate the fitness value of the new position, compare it with the fitness value of the original position, and choose the one with a better fitness value as the position for the next iteration.

Step 6: Run the gray-wolf-individual correction on its result.

Step 7: Check whether the set number of iterations has been reached. If it has, proceed to the next step; otherwise, go back to Step 2.

Step 8: Output the optimal offloading strategy.

Further pseudocode descriptions of the HGWOWOA are presented in Algorithm 2.

Algorithm 2 HGWOWOA

Input: Gray-wolf population size C, maximum number of iterations T, number of idle vehicles b, information on random computing tasks, information on the computing power of the device
Output: Optimal offloading strategy

Initialization: gray-wolf population location $X_i(0)(i = 1, 2, \cdots, N)$, t←0;
for t = 1, 2, . . . , T do
  $a \leftarrow 2 - 2t/T$;
  $\boldsymbol{A} \leftarrow 2 \times a \times \boldsymbol{n}_1 - a$;
  $\boldsymbol{C} \leftarrow 2 \times \boldsymbol{n}_2$;
  if $p < 0.5$ then
    $D_{\mathrm{a}} \leftarrow |\boldsymbol{C}_1 \cdot \boldsymbol{X}_{\mathrm{a}} - \boldsymbol{X}(\mathrm{t})|$;
  $D_\beta \leftarrow \left|\boldsymbol{C}_2 \cdot \boldsymbol{X}_\beta - \boldsymbol{X}(\mathrm{t})\right|$;
  $D_\delta \leftarrow |\boldsymbol{C}_3 \cdot \boldsymbol{X}_\delta - \boldsymbol{X}(\mathrm{t})|$;
  $\boldsymbol{X}_1(t+1) \leftarrow \boldsymbol{X}_\alpha(t) - \boldsymbol{A}_1 \cdot D_a$;
  $\boldsymbol{X}_2(t+1) \leftarrow \boldsymbol{X}_\beta(t) - \boldsymbol{A}_2 \cdot D_\beta$;
  $\boldsymbol{X}_3(t+1) \leftarrow \boldsymbol{X}_\delta(t) - \boldsymbol{A}_3 \cdot D_\delta$;
  $\boldsymbol{X}(t+1) \leftarrow \frac{1}{3}(\boldsymbol{X}_1(t+1) + \boldsymbol{X}_2(t+1) + \boldsymbol{X}_3(t+1))$;
  else
    $\boldsymbol{Levy}(d) = s \leftarrow \frac{u}{|v|^{1/d}}$;
    $\sigma \leftarrow \left[\frac{\Gamma(1+d) \times \sin(\frac{d \times \pi}{2})}{d \times \Gamma(\frac{1+d}{2}) \times 2^{\frac{d-1}{2}}}\right]^{1/d}$;
    $\boldsymbol{X}'(t) \leftarrow \boldsymbol{X}_{best}(t) + e^{bl} \times \cos(2\pi l) \times |\boldsymbol{X}_{best} - \boldsymbol{X}(t)| \oplus \boldsymbol{Levy}(d)$;
  end if
  if fitness$\left(\boldsymbol{X}'(t)\right) <$ fitness$(\boldsymbol{X}(t))$ then
    $\boldsymbol{X}(t+1) = \boldsymbol{X}'(t)$;
  else
    $\boldsymbol{X}(t+1) = \boldsymbol{X}(t)$;
end for
Return $\boldsymbol{X}$

4. Simulation Design and Result Analysis

4.1. Simulation Parameter Settings

Experimental verification was conducted on the MATLAB 2020b simulation platform, and a vehicle-unloading model was constructed, which included task vehicles, idle vehicles, and vehicles equipped with edge-server base stations. This section presents a performance simulation of the HGWOWOA and compares it with the following five unloading strategies: local, MEC, random, PSO, and GWO. The local strategy represents the calculation's

unloading only on the task vehicle itself, whereas the MEC strategy represents unloading only on the base station equipped with MEC. The random strategy selects a random location for unloading between the task vehicle itself, nearby idle vehicles, or base stations with MEC. The PSO strategy is based on the particle-swarm algorithm, and it solves for the optimal unloading strategy and resource allocation, whereas the GWO strategy is based on the gray-wolf algorithm, and it solves for the optimal unloading strategy and resource allocation. The specific simulation parameter settings are shown in Table 1.

Table 1. Simulation parameter settings.

Parameters	Value
Number of vehicles offloaded	10~30
Number of idle vehicles	5~10
Task data size $data_i$	10~50 kb
Computational intensity ρ	20~40 cycles/bit
Channel gain between vehicle and MEC/idle vehicle h_i	1.3×10^{-4} Hz
Total channel bandwidth between vehicle and MEC/idle vehicle	8×10^4 Hz
Transmission power of vehicle P_i^{up}	1.5~2 w
Computing capability of vehicle f_i	3.2×10^5 cycles/s
Computing capability of idle vehicles f_{idle}	3.2×10^5 cycles/s
Computing capability of edge server f_{MEC}	$7\sim9 \times 10^6$ cycles/s
Equipment power of vehicles/idle vehicles P_i/P_{idle}	4~8 w
Equipment power of edge server P_{MEC}	25 w
Background channel noise power N_0	5×10^{-14} w
Allocation of bandwidth	$3.2\sim8 \times 10^4$ Hz
Maximum tolerated delay for tasks	400~500 ms
Average delay of relaying between vehicles t_r	4~6 ms
Output data coefficient λ	0.01~0.3

4.2. Analysis of Simulation Results

4.2.1. Experiments on the Values of η and θ

Figure 2 shows the impact of delay weight factor η on the total cost of the system when there are ten task vehicles and two idle vehicles in the system. It can be observed from Figure 2 that the total cost of the system decreases as η increases. Among all algorithms, the HGWOWOA obtains the minimum total cost; therefore, the HGWOWOA has the best performance. In addition, when the delay weight coefficient η is within the range from 0.8 to 0.9, the total cost of the six unloading strategies is relatively close. Therefore, in order to reduce the sensitivity of the algorithm itself relative to the delay weight coefficient, we chose to set the value of η to 0.85 in the subsequent simulation.

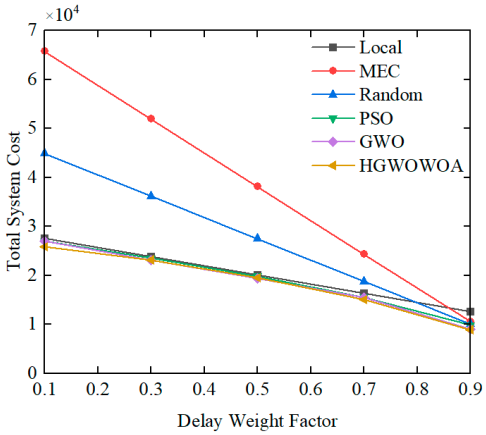

Figure 2. Influence of the value of η on system cost.

4.2.2. Impact of Factors on System Costs

Figure 3 shows the results of the impact on the number of idle vehicles for the system's total cost. As observed in Figure 3, the average system cost per task vehicle decreases with an increasing number of idle vehicles. Specifically, when the number of idle vehicles is 16, the total system cost decreases most significantly; in contrast, when the number of idle vehicles is 10, the decreasing trend of the total system cost is the smallest. This is because, with fewer vehicles at the base station, each task vehicle can obtain more computing resources from the server; thus, the increase in idle vehicles has less impact on reducing the system's cost. However, as the number of task vehicles increases, and each task vehicle can obtain fewer computing resources from the server, increasing the number of idle vehicles can significantly reduce the system's cost.

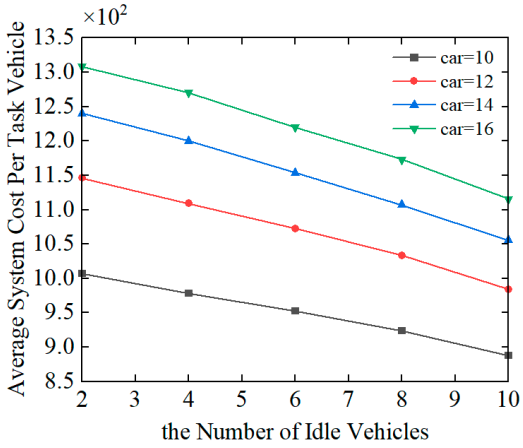

Figure 3. Influence of idle vehicles on system cost.

Figure 4 shows the impact of task volume per vehicle on the system cost for different offloading strategies. As the computational task volume per vehicle increases, the system cost generally increases in all offloading strategies. However, the HGWOWOA has the smallest increase in system cost, indicating that it has the best offloading performance among all algorithms. Additionally, overall, the HGWOWOA has the smallest system total cost of the same conditions. When the task volume was 50 kb, the system total cost of the HGWOWOA was 66.72% lower than that of local, 41.84% lower than that of MEC, 36.03% lower than that of random, 16.87% lower than that of PSO, and 11.78% lower than that of GWO. This demonstrates that the HGWOWOA has the best performance among the six offloading strategies.

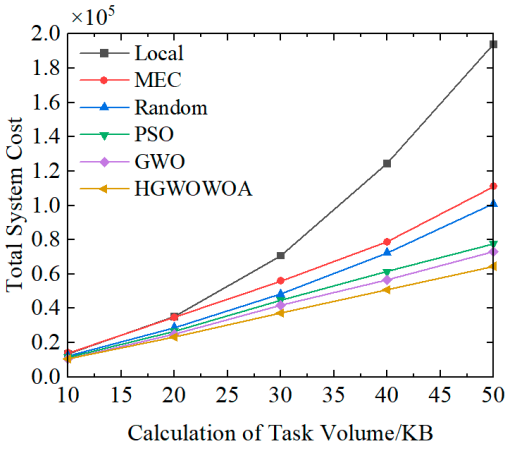

Figure 4. Influence of calculation task on system cost.

Figure 5 shows the impact curve of the number of vehicle terminals on the system's total cost. It can be observed in Figure 5 that, as the number of task vehicles increases, the total task volume of the system also increases; therefore, the total system cost also increases. Among them, the MEC offloading strategy performs most noticeably, and the total cost of the system increases significantly when the number of task vehicles is greater than 15. This is because the total amount of MEC computing resources is fixed, and when the number of offloaded task vehicles gradually increases, the allocated computing resources will decrease. Overall, the HGWOWOA offloading strategy performs better and has lower costs than other strategies. When the number of vehicle terminals was 30, the HGWOWOA strategy reduced the total cost of the system by 85.8% compared to MEC, 51.7% compared to local, 36.08% compared to random, 19.6% compared to GWO, and 19.03% compared to PSO.

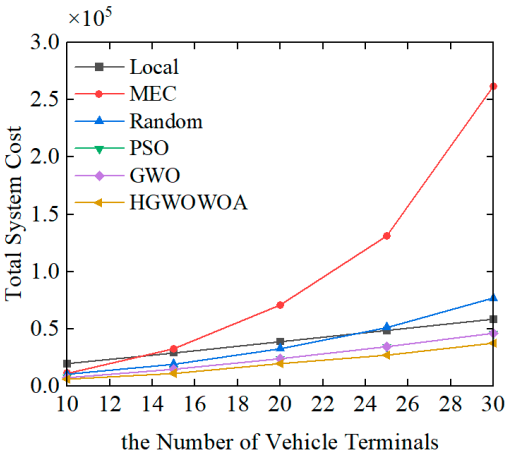

Figure 5. Influence of vehicle terminals on system cost.

Figure 6 shows the impact of the transmission bandwidth on the total system cost. As shown in the graph, as the bandwidth allocation increases, the transmission time of task data decreases, and the energy consumption of task vehicles also decreases, resulting in a decreasing trend in the total system cost. However, when the transmission bandwidth increased to around 50 KHz, the downward trend of the system's total cost was not significant, because the 50 KHz bandwidth was already sufficient for meeting the transmission of the vast majority of task data. Compared with other offloading strategies, the HGWOWOA strategy has the lowest overall system cost. When the transmission bandwidth was 80 KHz, compared with the MEC strategy, the HGWOWOA strategy reduced the total system cost by 30.8%, 28.5% compared with the local strategy, 20.8% compared with the random strategy, 8.5% compared with the GWO strategy, and 8.2% compared with the PSO strategy.

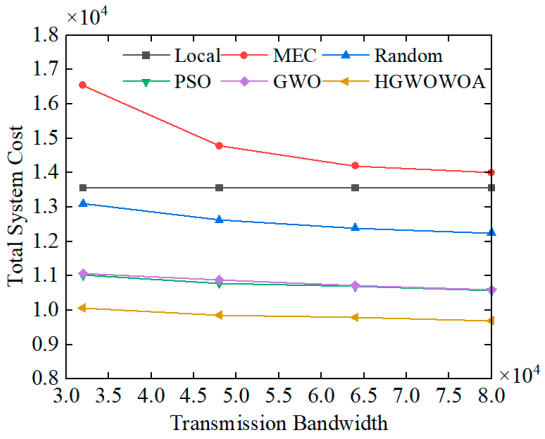

Figure 6. Impact of the transmission bandwidth on system cost.

Figure 7 displays the impact of the output data coefficient on the system cost. As shown in the graph, the system cost of the local offloading strategy remains unchanged with the increase in the output data coefficient, whereas other offloading strategies have a slight increase. This is because the local offloading strategy does not require data transmission, so its total cost is independent of the output data coefficient. For other offloading strategies, the impact of the output data coefficient is very small, so many edge-computing offloading studies directly omit it [30].

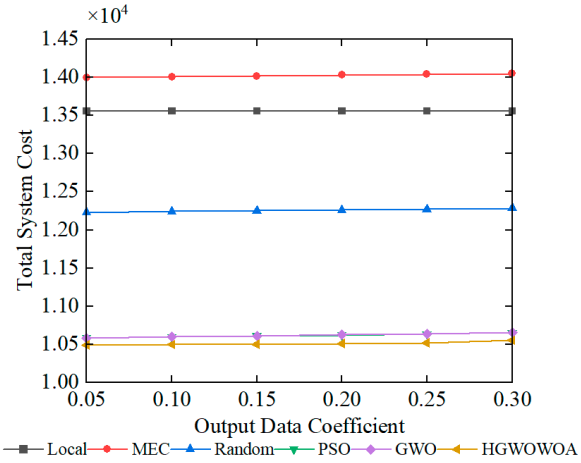

Figure 7. Impact of output data volume on system cost.

4.2.3. Convergence Comparison

In order to compare the convergence performance of the three decision-making methods—GWO, PSO, and HGWOWOA—we set the number of vehicles to ten, the number of idle vehicles to two, and the computing task to 10 kbit in the experiment. We tested the convergence of the total system cost under the three methods. The experimental results are shown in Figure 8. Overall, the GWO algorithm has the fastest convergence speed, reaching stability after 10 iterations, but with a generally poorer convergence value. The PSO algorithm has the slowest convergence speed, reaching stability after 25 iterations, but with a better convergence value. The convergence speed of the HGWOWOA method is between that of the PSO method and the GWO method, and the convergence value is the best. Therefore, the proposed HGWOWOA in this paper has excellent performances in both convergence speed and convergence accuracy; it can converge faster than the PSO algorithm, and it has a higher convergence accuracy than both the GWO algorithm and the PSO algorithm.

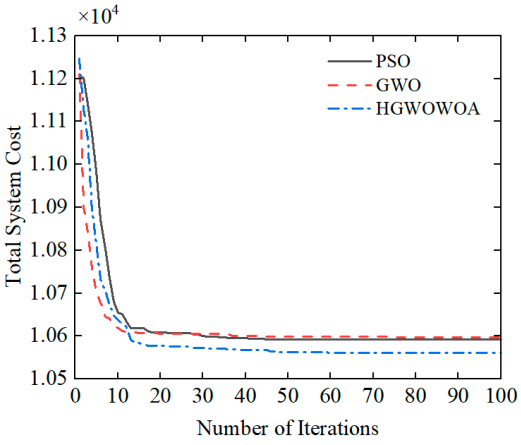

Figure 8. Comparison of convergence of the PSO algorithm, GWO algorithm, and HGWOWOA.

5. Conclusions

In this paper, we have presented an adaptive joint offloading strategy for vehicular edge computing, which does not require cloud-computing support. An offloading computing model has been proposed for computing latency and energy consumption in the context of MEC, and it uses task computing latency, computing energy consumption, and MEC server resources as constraints. Further, a system-utility function has been designed to transform the offloading problem into a constrained system-utility optimization problem. In addition, an improved gray-wolf optimization algorithm has been proposed to obtain the optimal solution to the computation offloading problem. Compared with local, MEC, random, PSO, and GWO offloading strategies, the proposed adaptive joint offloading strategy can achieve the minimum computation delay and the lowest energy consumption.

In future work, we will design an improved computational offloading model that can reflect vehicle mobility and the actual connections of vehicles in the real state. Meanwhile, it would be meaningful to consider the transmission loss due to packet loss in fast-moving vehicles and MEC devices, which is more similar to the real scene. A practical offloading strategy must be promising in the field of Internet of Vehicles.

Author Contributions: W.Z. and K.T. contributed to the conception and design of the proposed strategy. All authors wrote and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Foundation of China (Grant 72071153) and the Key Research and Development Program in Shaanxi Province of China (Grant 2021GY-066).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Phadke, A.; Medrano, F.A.; Ustymenko, S. A Review of Vehicular Micro-Clouds. In Proceedings of the 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2021; pp. 411–417.
2. Qureshi, K.N.; Alhudhaif, A.; Haidar, S.W.; Majeed, S.; Jeon, G. Secure data communication for wireless mobile nodes in intelligent transportation systems. *Microprocess. Microsyst.* **2022**, *90*, 104501. [CrossRef]
3. Qureshi, K.N.; Din, S.; Jeon, G.; Piccialli, F. Internet of vehicles: Key technologies, network model, solutions and challenges with future aspects. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 1777–1786. [CrossRef]
4. Verschoor, T.; Charpentier, V.; Slamnik-Kriještorac, N.; Marquez-Barja, J. The testing framework for Vehicular Edge Computing and Communications on the Smart Highway. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2023; pp. 1147–1150.
5. Jin, X.; Hua, W.; Wang, Z.; Chen, Y. A survey of research on computation offloading in mobile cloud computing. *Wirel. Netw.* **2022**, *28*, 1563–1585. [CrossRef]
6. Zhang, J.; Zhao, X. An overview of user-oriented computation offloading in mobile edge computing. In Proceedings of the 2020 IEEE World Congress on Services (SERVICES), Beijing, China, 18–23 October 2020; pp. 75–76.
7. Zhang, K.; Zhu, Y.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet Things J.* **2019**, *6*, 7635–7647. [CrossRef]
8. Zhang, K.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. Mobile edge computing and networking for green and low-latency Internet of Things. *IEEE Commun. Mag.* **2018**, *56*, 39–45. [CrossRef]
9. Shen, X.; Chang, Z.; Niu, S. Mobile Edge Computing Task Offloading Strategy Based on Parking Cooperation in the Internet of Vehicles. *Sensors* **2022**, *22*, 4959. [CrossRef]
10. Fang, J.; Zhang, M.; Ye, Z.; Shi, J.; Wei, J. Smart collaborative optimizations strategy for mobile edge computing based on deep reinforcement learning. *Comput. Electr. Eng.* **2021**, *96*, 107539. [CrossRef]
11. Huang, X.; Zhang, W.; Yang, J.; Yang, L.; Yeo, C.K. Market-based dynamic resource allocation in Mobile Edge Computing systems with multi-server and multi-user. *Comput. Commun.* **2021**, *165*, 43–52. [CrossRef]
12. Lu, J.; Jiang, J.; Balasubramanian, V.; Khosravi, M.R.; Xu, X. Deep reinforcement learning-based multi-objective edge server placement in Internet of Vehicles. *Comput. Commun.* **2022**, *187*, 172–180. [CrossRef]
13. Li, W.; Zhang, N.; Liu, Q.; Feng, W.; Ning, R.; Lin, S. Scalable modulation based computation offloading in vehicular edge computing system. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, 18 November–16 December 2020; pp. 1–5.

14. Gao, J.; Kuang, Z.; Gao, J.; Zhao, L. Joint Offloading Scheduling and Resource Allocation in Vehicular Edge Computing: A Two Layer Solution. *IEEE Trans. Veh. Technol.* **2022**, *72*, 3999–4009. [CrossRef]
15. Sun, G.; Zhang, J.; Sun, Z.; He, L.; Li, J. Collaborative Task Offloading in Vehicular Edge Computing Networks. In Proceedings of the 2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Denver, CO, USA, 19–23 October 2022; pp. 592–598.
16. Qian, L.; Wu, Y.; Ouyang, J.; Shi, Z.; Lin, B.; Jia, W. Latency optimization for cellular assisted mobile edge computing via non-orthogonal multiple access. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5494–5507. [CrossRef]
17. Li, S.; Lin, S.; Cai, L.; Li, W.; Zhu, G. Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3384–3398. [CrossRef]
18. Chen, C.; Chen, L.; Liu, L.; He, S.; Yuan, X.; Lan, D.; Chen, Z. Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks. *IEEE Access* **2020**, *8*, 18863–18873. [CrossRef]
19. Sun, J.; Gu, Q.; Zheng, T.; Dong, P.; Valera, A.; Qin, Y. Joint optimization of computation offloading and task scheduling in vehicular edge computing networks. *IEEE Access* **2020**, *8*, 10466–10477. [CrossRef]
20. Zeng, F.; Chen, Q.; Meng, L.; Wu, J. Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 3247–3257. [CrossRef]
21. Bi, J.; Yuan, H.; Duanmu, S.; Zhou, M.; Abusorrah, A. Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. *IEEE Internet Things J.* **2020**, *8*, 3774–3785. [CrossRef]
22. Fan, W.; Su, Y.; Liu, J.; Li, S.; Huang, W.; Wu, F.; Liu, Y.A. Joint Task Offloading and Resource Allocation for Vehicular Edge Computing Based on V2I and V2V Modes. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 4277–4292. [CrossRef]
23. Liu, J.; Wang, Y.; Zhang, W.; Tian, K. A Novel Offloading and Resource Allocation Scheme for Time-critical Tasks in Heterogeneous Internet of Vehicles. In Proceedings of the 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India, 3–5 March 2023; pp. 1–7.
24. Zhang, Z.; Zeng, F. Efficient Task Allocation for Computation Offloading in Vehicular Edge Computing. *IEEE Internet Things J.* **2022**, *10*, 5595–5606. [CrossRef]
25. Zhao, K.; Liu, Y.; Hu, K. Optimal Pattern Synthesis of Array Antennas Using Improved Grey Wolf Algorithm. In Proceedings of the 2022 IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 15–17 July 2022; pp. 172–175.
26. Mahdi, M.A.; Dawood, L.M. A Grey Wolf Optimization Algorithm for Integrating Process Planning and Scheduling Problem. In Proceedings of the 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 9–11 June 2022; pp. 1–5.
27. Ramesh, M.; Yadav, A.K. Wind contributed load frequency control scheme for standalone microgrid using grey wolf optimization. In Proceedings of the 2022 IEEE Delhi Section Conference (DELCON), New Delhi, India, 11–13 February 2022; pp. 1–6.
28. Deng, X.; Sun, Z.; Li, D.; Luo, J.; Wan, S. User-centric computation offloading for edge computing. *IEEE Internet Things J.* **2021**, *8*, 12559–12568. [CrossRef]
29. Mantegna, R.N. Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. *Phys. Rev. E* **1994**, *49*, 4677. [CrossRef]
30. Chen, L.; Zhou, S.; Xu, J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. *IEEE/ACM Trans. Netw.* **2018**, *26*, 1619–1632. [CrossRef]