Article

Design and Application of a Resource Allocation Method for CAEVs Internet of Things Based on Swarm Intelligence Computing

Yibo Han [1,*], Zheng Zhang [2,*], Pu Han [3], Bo Yuan [4], Lu Liu [4] and John Panneerselvam [4]

1 Nanyang Research Institute of Big Data, Nanyang Institute of Technology, Nanyang 473004, China
2 School of Computer and Software, Nanyang Institute of Technology, Nanyang 473004, China
3 School of Information Engineering, Nanyang Institute of Technology, Nanyang 473004, China; hanpu@nyist.edu.cn
4 Department of Informatics, University of Leicester, University Rd, Leicester LE1 7RH, UK; b.yuan@leicester.ac.uk (B.Y.); j.panneerselvam@leicester.ac.uk (J.P.)
* Correspondence: hanyibo@nyist.edu.cn (Y.H.); zhangzheng@nyist.edu.cn (Z.Z.)

Abstract: The Internet of Things (IoT) faces significant challenges in the requirements of sensitive task latency, reasonable resource allocation and reliability for resource transactions. This paper introduces a novel method for road resource allocation in the IoT context of connected and autonomous electric vehicles (CAEVs). The proposed algorithm leverages the ant colony algorithm (ACA) to effectively allocate and coordinate road resources within groups of CAEVs. By considering the energy consumption and pheromone volatilization, the allocation and coordination process of road resources are optimized. To improve the linear packet loss of RED, we adopt the advanced ACA and CRED in the NS2 platform. The experimental results demonstrate that the proposed method outperforms the RED algorithm in packet loss rate and delay time, significantly enhancing system efficiency and performance. Furthermore, the combination of the CRED algorithm and ant colony algorithm successfully mitigates short-term congestion and identifies optimized paths with minimal delay.

Keywords: ant colony algorithm; CAEVs; resource allocation

check for updates

Citation: Han, Y.; Zhang, Z.; Han, P.; Yuan, B.; Liu, L.; Panneerselvam, J. Design and Application of a Resource Allocation Method for CAEVs Internet of Things Based on Swarm Intelligence Computing. *Electronics* **2023**, *12*, 2997. https://doi.org/10.3390/electronics12132997

Academic Editor: Shiho Kim

Received: 22 May 2023
Revised: 4 July 2023
Accepted: 5 July 2023
Published: 7 July 2023

1. Introduction

The intelligent transportation system (ITS) has made significant strides in complex environments, accompanied by extensive research into optimization strategies. Among the pivotal components of ITS, the Internet of Things (IoT) for connected and autonomous electric vehicles (CAEVs) has emerged as a focal point for investigation [1]. The CAEVs Internet of Things (IoT) entails leveraging internet connectivity to facilitate the seamless connection and management of transportation facilities and equipments, including vehicles, roads, and charging stations [2]. To achieve crucial functionalities such as movement and charging, CAEVs rely on the collaborative utilization of resources such as charging stations and roads. Effective resource allocation and coordination play a vital role in improving the overall efficiency and performance of the system [3].

Resource allocation in CAEVs IoT encounters various challenges in practical applications, including vehicle mobility, resource constraints, and allocation methods. CAEVs IoT faces serious challenges in terms of unbalanced deployment, unbalanced allocation of computing resources, and real-time computing latency due to the existence of roadside computing nodes. It is important to study how CAEVs IoT can effectively use limited resources to achieve reliable and scalable wireless transmission is of great importance to the development of CAEVs IoT. Vehicle density may change dramatically with dramatic changes in time and position, while also being in high-speed motion. The tasks generated by CAEVs IoT also have very different requirements for quality of service (QoS). Emergency messages and real-time collaborative control messages have strict delay constraints, while related entertainment applications can tolerate a certain degree of delay. Thousands of vehicles need to exchange information in real time, but the spectrum of resources available for CAEVs IoT communication is very limited, and the high-speed movement of vehicles in CAEVs IoT environments and the heterogeneity of latency requirements of different applications are the main challenges that hinder the rapid development of CAEVs IoT. Therefore, it is necessary to design a reasonable and effective resource allocation strategy for CAEVs IoT environments. To address these issues, swarm intelligent computing has become an effective approach [4]. Swarm intelligent computing involves forming a group of individuals and equipping them with intelligence and behavioural rules, enabling the collective intelligence of the entire group. Through cooperation, communication and competition, individuals in swarm intelligence computing can collaboratively accomplish tasks and achieve swarm optimization and coordination [5]. By employing resource allocation methods based on swarm intelligent computing, vehicles can achieve collaborative resource utilization, leading to enhanced system efficiency and performance. Swarm intelligent computing refers to a class of intelligent algorithms with distributed intelligent behavioural characteristics inspired by the group behaviour of insects, herds of animals, flocks of birds, schools of fish, etc. As an emerging computational technology, intelligent computing has received increasing attention from researchers and has special connections with artificial life, evolutionary strategies, and genetic algorithms, being widely used in research. Swarm intelligent computing provides the basis for finding solutions to complex distributed problems without centralized control or providing a global model. In this paper, we optimized and improved the bionic swarm intelligence algorithm, i.e., ant colony algorithm (ACA), and proposed a resource allocation method for CAEVs IoT.

To effectively reduce energy consumption and improve the performance of avionics systems, Du et al. proposed an energy-aware resource allocation method based on the ACA [6]. Firstly, they established a mathematical model for resource allocation, reflecting the relationship between task requirements and resource energy consumption. Then, the ACA allocates heterogeneous resources to minimize completion time and achieve low-energy consumption. Xu et al. introduced a proportional fair resource allocation problem based on chance constrained programming, which maximizes the average and rate within an adaptive time window and ensures the Jain fairness index (JFI) requirement under the target probability [7]. To solve the resource allocation problem with opportunity constraints, they utilized hybrid ant colony optimization and a support vector machine (SVM). Al-Masri et al. suggested a collaborative energy-aware resource allocation and scheduling strategy based on the TOPSIS multi-criteria decision-making method to maximize resource sharing and utilization efficiency in offloading IoT tasks [8]. Ari et al. used an improved ACA to address resource allocation for 5G C-RAN [9]. The challenge is to design logical joint mappings between user equipment (UE) and RRH, as well as between RRH and BBU. This was performed adaptively based on network load conditions to reduce overall network costs while maintaining user QoS and QoE. From the above discussion, we can see that there is no report on the ACA on resource allocation for CAEVs. the ACA is a swarm intelligence optimization algorithm that can solve discrete optimization problems, so it is suitable for the problem.

This paper presents a road resource allocation method based on swarm intelligent computing to address the challenge of road resource sharing in CAEVs IoT. The method's fundamental principle and implementation process are thoroughly explained, followed by comprehensive experimental verification and performance testing. Furthermore, this paper discusses and analyses the potential applications and future research directions of this method, providing valuable insights into its prospects and future developments. The approach presented in the paper has two main contributions: First, we not only consider the time, cost and load factors under the resource allocation task, but also consider the security, reliability and latency issues specific to CAEVs IoT scenarios to effectively improve the performance of all aspects of resource scheduling. Secondly, we try to introduce other algorithms to integrate and complement each other with the ACA to optimize the pheromone of the probability transfer formula in the ACA, so as to find the optimized path solution. The proposed method provides some reference for the next step of resource allocation optimization.

2. Related Works

With the development of the economy, the level of car ownership is rising, the number of vehicle users is increasing, and the rate requirements for CAEVs IoT are also increasing. To meet the demand of various application services of CAEVs IoT and to utilize the available communication resources more effectively, more and more researchers focus on resource allocation optimization in the field of CAEVs IoT.

In the literature [10], the authors build an SDN-assisted MEC network architecture for in-vehicle networks and improve the efficiency and flexibility of in-vehicle networks by introducing SDN controllers. The optimal offloading decision, transmission power control, sub-channel allocation and computational resource allocation schemes are also given for the V2X offloading and resource allocation problems. The literature [11] proposed a DNN model partitioning strategy, then proposed a content-driven joint resource allocation scheme based on vehicle-side collaboration, and designed a multi-intelligent distributed Q-learning algorithm to solve the multi-constrained non-linear planning arising from the vehicle network resource allocation A multi-intelligence distributed Q-learning algorithm is designed to solve the multi-constrained non-linear planning problem arising from the resource allocation of vehicular networks.

The literature [12] reported increasing the system capacity by introducing NOMA and improved the network performance by centralized resource management that allows D2D resource sharing based on spatial reuse among all V2X communication groups. The literature [13] decomposed the problem of optimizing the average delay of downloading files from a system vehicle into a communication resource optimization allocation subproblem and a cache resource optimization allocation and file placement selection subproblem, and proposed a joint resource allocation algorithm based on network slicing. In the literature [14], in order to satisfy the maximum delay and residence time constraints of vehicles, a fog-based vehicle network was used to balance the number of shared messages and constrained resources, and decouple the original problem into two independent subproblems and design solutions to solve them. To address the highly dynamic nature of the V2X environment, the literature [15] introduced a mapping fuzzy space to effectively avoid the deterioration of this dynamic environment on the stability and convergence of the optimization mechanism, and constructed a bilateral many-to-many fuzzy matching game (MM-FMG) to formulate the optimization problem with dynamic and uncertain information on this basis, finally proposing a vehicle resource dynamic matching algorithm to solve the problem.

Most of the current resource allocation algorithms for CAEVs IoT are focused on the allocation of on-board information resources, and the scope of research is relatively limited. However, with the continuous development of communication technology, new technologies such as millimetre wave communication, radar and satellite communication are emerging, providing a wide creative space for the development of CAEVs research scholars and unlimited possibilities for the development of telematics technology.

The existing resource allocation method based on the ACA has the disadvantages of a long span, high cost and easy to produce local optimal solutions, making its application in resource-scheduling problems much less valuable. Therefore, most scholars optimize the ACA to improve the performance of the algorithm in three aspects: time, cost, and load by adding constraints. In addition to considering time, cost, and load, the improvement of the ACA should also consider the security, reliability, and delay of resource allocation for CAEVs IoT. However, existing studies rarely consider these factors together and propose effective optimization improvement methods. Our proposed method tries to introduce the RED algorithm and ACA to complement each other, thus improving the drawbacks of the ACA. Specifically, we define the objective function based on the task processing time, network bandwidth, and network delay, use the allocation results as pheromones of the probabilistic transfer formula in the ACA, improve the ACA, and then reallocate according to the constraints to achieve the optimal effect.

3. Congestion Recognition Based on the ACA

3.1. Ant Colony Algorithm (ACA)

The ACA (ant colony algorithm) is a self-organizing and swarm intelligence-based algorithm that mimics the food search behaviour of ants to find optimal paths in a network [16,17]. It effectively addresses path-planning problems in both static and dynamic environments. In dynamic environments, traditional path-planning algorithms struggle to handle the constant changes in obstacles. However, the ACA leverages the concept of pheromones to guide ants in finding new shortest paths, thereby enhancing the efficiency and accuracy of path planning. ACA's adaptability to dynamic environments makes it a powerful tool in path-planning applications [18,19].

The ACA finds application in various domains of intelligent transportation systems. It enables path planning for intelligent vehicles, facilitates the identification and resolution of traffic congestion, and facilitates traffic light control. In the aerospace sector, the ACA proves valuable for UAV path planning and flight control [20]. Additionally, the ACA finds utility in optimizing routing protocols and enhancing transmission quality in the field of data transmission [21]. Its versatility and effectiveness make the ACA a versatile tool across multiple domains [22].

3.2. Improved ACA and Its Steps

(1) Modify the heuristic function

In congestion identification, time delay is chosen as the evaluation index for the optimal path. The traditional ACA, which considers the shortest Euclidean mile distance, no longer meets the desired outcome. In this context, the aim is to assess the path's merits and drawbacks based on the total time delay [23]. To achieve this, the network propagation delay is determined by calculating the ratio of the location distance of the communication network nodes to the packet propagation speed in the network topology system. The network queuing delay and processing delay are obtained using the M/M/1 queuing theory model. The total network delay is composed of these three components [24]. To simplify calculations, the propagation delay and transmission delay can be combined as the transmission delay. Consequently, the original heuristic function is modified, with P_t representing the transmission delay and S_t denoting the queuing delay. During congestion, the value of P_t is significantly lower than S_t. The modified heuristic function is as follows:

$$\eta_{ij}(t) = \frac{1}{P_t + S_t} \tag{1}$$

(2) Improved pheromone updating formula

Upon shifting the focus of the ACA from solving the shortest distance path to solving the shortest delay path, it becomes necessary to update the pheromone formulas [25]. These formulas play a crucial role in guiding the ants' search for optimal paths. To ensure accuracy in representing the desired objective, it is essential to revise the pheromone formulas accordingly. The following pheromone formulas need to be updated:

$$\tau_{ij}(t+1) = (1-\rho_0) \times \tau_{ij}(t) + \rho_0 \times \tau_{ij}(0) \tag{2}$$

The given formula represents the local update formula for pheromones. However, to complete the pheromone update process, the global update formula is also required. The global update formula is as follows:

$$\tau_{ij}(t+1) = (1-\rho_1) \times \tau_{ij}(t) + \rho_1 \times \frac{1}{T_{\text{best}}} \tag{3}$$

where $\tau_{ij}(0)$ is the initial concentration of pheromones and T_{best} is the total delay of the shortest path in each cyclic search. The steps of the improved ACA are as follows:

Step 1: To conduct the search for the designated network termination node in the total network delay matrix, which includes the transmission delay matrix and the queuing delay matrix, the ACA is employed with Q ants placed on the starting node of the network. The number of ants traversing the network, denoted as q, is typically calculated as 0.6 times the number of network nodes with the result rounded to the nearest whole number. Initially, the number of ants in each cycle is set to 0. Additionally, the cycle number, represented by k, starts at 0 and can reach a maximum value of N. It is important to note that the starting node and destination node of the network must be manually designated for each re-experiment.

Step 2: If the current cycle number of the ACA is less than N, make $k = k + 1$ and enter the next cycle.

Step 3: If the number of ants sent in this cycle is less than q, then $q = q + 1$, and continue to send the next ant.

Step 4: During each iteration, every ant in the ACA calculates the transition probability based on the pheromone concentration on the link connecting the starting node to its adjacent nodes. Subsequently, the ant checks whether the next hop node is present in its taboo table, denoted as taboo(k). If the node is not in the taboo table, it is added to the ant's taboo list. However, if the node is already in the taboo table, the ant reselects the next jump node to ensure that it does not revisit previously visited nodes. This process ensures that ants explore new paths while avoiding revisiting nodes in their path selection.

Step 5: The process described in step 4 is iterated until the current network node reaches the desired termination node. During this process, the traversal path of ant q is continuously updated, keeping track of the number of nodes traversed along the path. Additionally, the total time delay of the recorded traversal path is calculated. After completing the traversal, the local pheromone concentration is updated according to Equation (2). This iterative process ensures that ants explore and update their paths while taking into account the local pheromone information.

Step 6: Returning to step 3, the algorithm continues until the condition where Q is greater than Q is satisfied, and all ants in the current cycle have completed their traversal.

Step 7: By comparing the total delay of all paths in the current cycle, the algorithm identifies the optimal path, including the number of nodes and the total delay associated with it. Subsequently, the global pheromone concentration is updated using the formula specified in Equation (3).

Step 8: Steps 2–7 are repeated until the current cycle number k is greater than N.

Step 9: The program outputs the optimized path and the total delay associated with it. The flowchart of the optimized ACA in the network total delay matrix is depicted in Figure 1.

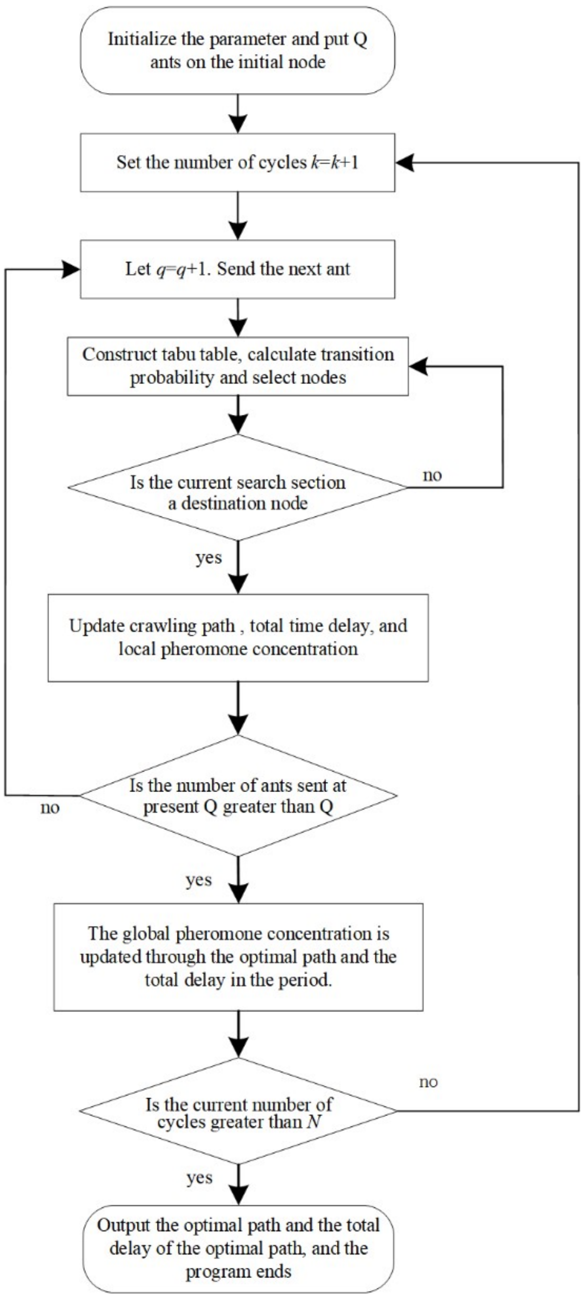
Initialize the parameter and put Q ants on the initial node
Set the number of cycles *k*=*k*+1
Let *q*=*q*+1. Send the next ant
Construct tabu table, calculate transition probability and select nodes
Is the current search section a destination node
no
yes
Update crawling path , total time delay, and local pheromone concentration
Is the number of ants sent at present Q greater than Q
no
yes
The global pheromone concentration is updated through the optimal path and the total delay in the period.
Is the current number of cycles greater than *N*
no
yes
Output the optimal path and the total delay of the optimal path, and the program ends

Figure 1. The flowchart of the optimized ACA.

3.3. Overview of Congestion Identification Algorithms

This paper introduces a method for identifying congested nodes using reverse reasoning. The method relies on the observation that if two optimal paths intersect and subsequently deviate from their initial overlapping section after a period of stability, this suggests the presence of congestion during that timeframe. Such congestion triggers a change in the initial path selection by the ACA. Therefore, the skipped section of the initial overlapping path is considered the congestion path, with its nodes identified as congestion nodes. The network congestion identification method based on the ACA involves the following main steps:

To begin with, the initialization and assignment of parameters for the ant colony model and M/M/1 queuing model are carried out. Then, utilizing the Waxman model proposed by B.M. Waxman [26], the network topology model and the corresponding distance information between nodes are randomly generated [27]. This allows for the determination of the propagation delay between network nodes and the calculation of data transmission delay. Consequently, a network transmission delay matrix is constructed [28]. The propagation delay is defined as the ratio of the distance between network nodes to the data propagation speed. Additionally, employing the M/M/1 queuing model, the data processing delay and queuing delay of each network node in the network topology are generated, resulting in the construction of a network queuing delay matrix. To amplify the effect, the data can be scaled according to a specified proportion. Finally, the total network delay matrix is constructed by combining the network transmission delay matrix and the network queuing delay matrix. It is important to note that in cases of congestion, the network transmission delay becomes significantly smaller or even negligible compared to the network queuing delay. The queuing delay at network nodes becomes the crucial factor in determining the optimal path.

By comparing several pairs of data from the original optimal path set and the current optimal path set, it becomes evident that each pair of current optimal paths avoids or bypasses the previous intersection point or a segment of the previous intersection path. These bypassed sections precisely correspond to the previously adjusted and simulated congestion paths. This observation indicates that the congested nodes are located along these bypassed paths, ultimately leading to the ACA selecting a new path. This reverse reasoning approach provides a logical and reasonable method for determining the location of congestion paths. By carefully comparing and analysing the nodes along the bypassed paths, it becomes possible to infer the specific locations of the congested nodes. Figure 2 illustrates the flowchart of the congestion identification method.

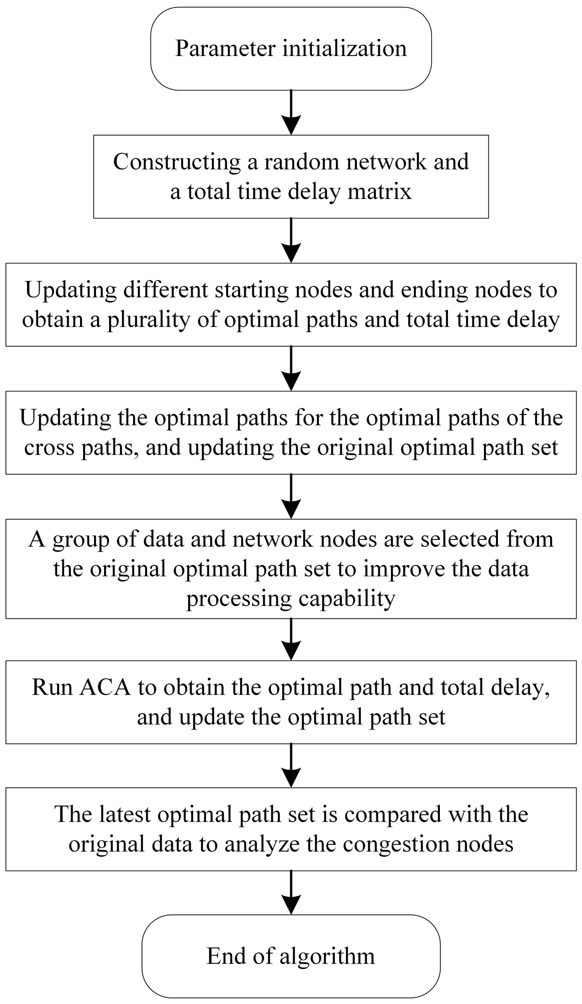
Parameter initialization
Constructing a random network and a total time delay matrix
Updating different starting nodes and ending nodes to obtain a plurality of optimal paths and total time delay
Updating the optimal paths for the optimal paths of the cross paths, and updating the original optimal path set
A group of data and network nodes are selected from the original optimal path set to improve the data processing capability
Run ACA to obtain the optimal path and total delay, and update the optimal path set
The latest optimal path set is compared with the original data to analyze the congestion nodes
End of algorithm

Figure 2. The flowchart of the congestion identification method.

4. Congestion Control Based on the RED Algorithm

4.1. Principle of the CRED Algorithm

The random early detection (RED) algorithm, widely used in network routers and switches, is a network traffic management algorithm for congestion control. Key parameters of the RED algorithm include minimum and maximum thresholds. When the number of packets in the queue is below the minimum threshold, no packets are dropped. When the number of packets is between the minimum and maximum thresholds, packets are dropped probabilistically based on a certain probability. When the number of packets in the queue exceeds the maximum threshold, more packets are discarded.

It is important to note that the RED algorithm is a passive congestion control mechanism. It takes action only when network congestion has already occurred, rather than actively predicting and avoiding congestion. Therefore, in practical networks, it is common to combine other congestion control algorithms (such as TCP congestion control) to provide better performance and stability. The RED algorithm is designed to overcome the global synchronization issue inherent in traditional TCP congestion control algorithms and alleviate biases against burst traffic. To comprehend the subsequently improve the model, it is essential to grasp the underlying principles of the original RED algorithm [29]. The core of the RED algorithm comprises two main components: calculating the average queue length in routers and determining the packet drop probability [30].

(1) Calculate the average queue length

$$\text{avg} = (1 - \text{Wq}) \times \text{avg} + \text{Wq} \times \text{Lsamp} \tag{4}$$

where avg represents the average length of the weighted dynamic queue. Wq is the weight factor that represents the weight of the latest sample. It typically ranges from 0 to 1. Lsamp represents the current length of the queue. This formula calculates the average length of the weighted dynamic queue by performing a weighted average between the length of the latest sample and the previous average length. The weight factor Wq determines the influence of the latest sample, with larger weights giving more importance to the latest sample, while smaller weights provide a more averaged result.

Weighted dynamic averaging analyses congestion by measuring the queue length over a specific period, rather than relying on the instantaneous queue length. This approach provides a more comprehensive understanding of congestion and its fluctuations, as illustrated in Figure 3.

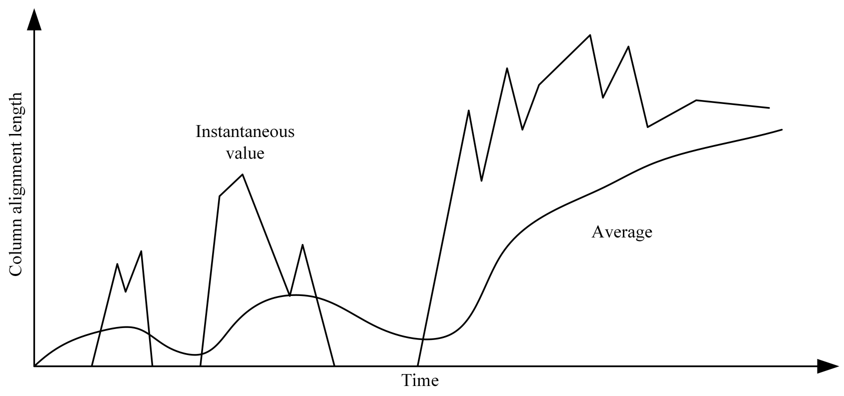

Figure 3. Average length of a weighted dynamic queue.

(2) How to discard packets

The RED algorithm utilizes minimum min_{th} and maximum threshold max_{th} values as benchmarks to determine the appropriate handling of incoming packets [31]. Additionally, a discard probability P is established. The following rules outline the specific guidelines:

If $\text{avg} \leq \text{min}_{\text{th}}$, there is no packet processing and arrival is normal. When the number of packets in the queue is below the min_{th}, no packet loss occurs;

If $\text{max}_{\text{th}} \leq \text{avg}$, all incoming packet data is discarded. When the number of packets in the queue surpasses max_{th}, a greater number of packets are discarded;

If $\text{min}_{\text{th}} < \text{avg} < \text{max}_{\text{th}}$, packets are discarded according to the discard probability P. The discard probability P determines how packets are discarded. The schematic diagram illustrates the relationship between the discard probability P and the average length of packets.

(3) Calculate discarding probability

As mentioned above, the RED algorithm assigns a dropping probability P to the router when the average length of packets falls within the range of the minimum and maximum thresholds [32]. This allows the router to make informed decisions on packet dropping based on the assigned probability. The value of the dropping probability P is typically determined by two key elements, which are:

Step 1: As the average length of packets gradually approaches the maximum threshold, the discarding probability P will gradually increase. When the average length of packets equals the maximum threshold, the discarding probability P immediately becomes 1.

Step 2: In the RED algorithm, when the average length of packets falls within the critical zone between max_{th} and min_{th}, the discarding probability is adjusted to ensure equal discarding intervals for all incoming packets. This adjustment involves modifying the original discarding probability P_b to obtain a new marking probability P_a. The determination of packet discarding is based on the value of P_a. Specifically, as the number of successive successful packets transmitted c increases, the probability of discarding the next packet also increases. The discarding probability is calculated using the following formula:

$$P_b = \max_p(\text{avg} - \text{min}_{\text{th}})/(\text{max}_{\text{th}} - \text{min}_{\text{th}}) \tag{5}$$

$$P_a = P_b/(1 - \text{c} * P_b) \tag{6}$$

In the above formula, P_b represents a very small positive number. The term $1 - c * P_b$ is nearly equal to l. It is important to note that P_b plays a decisive role in the formula, while P_a is used to make slight adjustments to P_b. This correction is implemented to prevent routers from continuously losing packets. The discard probability P_b is defined as a piecewise function with different values in different intervals, as depicted in Figure 4. The specific functional form of P_b is as follows:

$$P_b = \begin{cases} 0 & \text{avg} <= \text{min}_{\text{th}} \\ \text{max}_p(\text{avg} - \text{min}_{\text{th}})/(\text{max}_{\text{th}} - \text{min}_{\text{th}}) & \text{min}_{\text{th}} < \text{avg} < \text{max}_{\text{th}} \\ 1 & \text{avg} >= \text{max}_{\text{th}} \end{cases} \tag{7}$$

In summary, we can conclude that as the average value avg starts from min_{th} and gradually approaches max_{th}, the discarding probability P_b also increases linearly from 0 to max_p. Once avg reaches max_{th}, P_b immediately becomes 1. Throughout this process, the dropping probability P_a increases slowly with the increase of c. Eventually, when avg reaches max_{th} or later, P_a further increases, resulting in the dropping of all newly arrived packets, regardless of their importance.

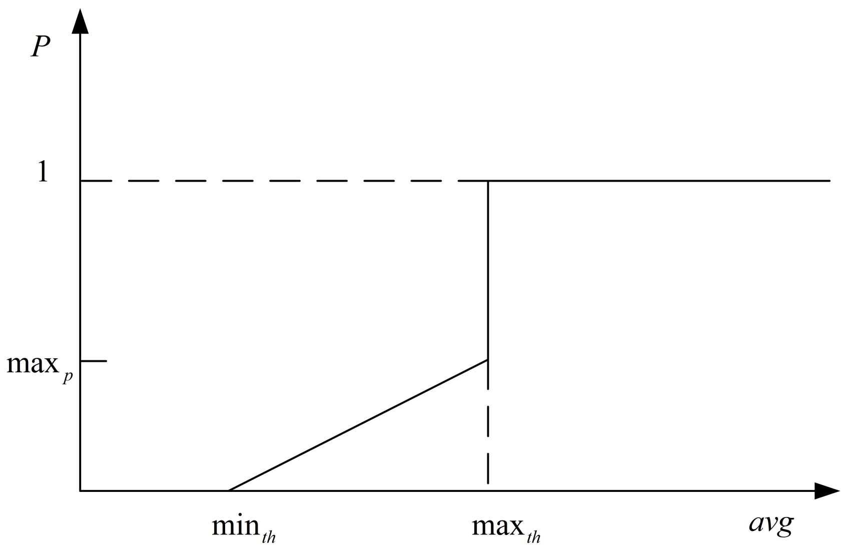

Figure 4. The discarding probability function of the RED algorithm.

(4) The relevant parameters and pseudo-codes of the RED algorithm (Algorithm 1)

Algorithm 1 The RED algorithm.

```
foreach packet arrival do
    calculate the new average queue size avg;
end
if $min_{th} < avg < max_{th}$ then
    increcent c;
    calculate packet dropping probability;
    $P_b = max_p(avg - min_{th})/(max_{th} - min_{th})$;
    $P_a = P_b/(1 - c^* P_b)$;
    with probability $P_a$, drop the arriving packet;
    c=0;
else
    if $max_{th} <= avg$ then
        drop the arriving packet;
    end
end
```

The execution flow of the RED algorithm is shown in Figure 5.

4.2. Analysis of the Advantages and Disadvantages of RED

The RED algorithm proves to be highly effective in congestion control, offering significant contributions to this area. Its main contributions can be summarized as follows [33]:

- The RED algorithm has the capability of congestion identification. It can determine whether congestion occurs by detecting the average queue length and can request the source to reduce the sending rate.
- The RED algorithm has the capability of congestion control and can choose how to discard packets based on the discard probability, which is very effective in relieving the current congestion state.
- The RED algorithm makes the interval of dropped packets evenly distributed, avoiding the global synchronization problems often caused by traditional congestion control algorithms in TCP.

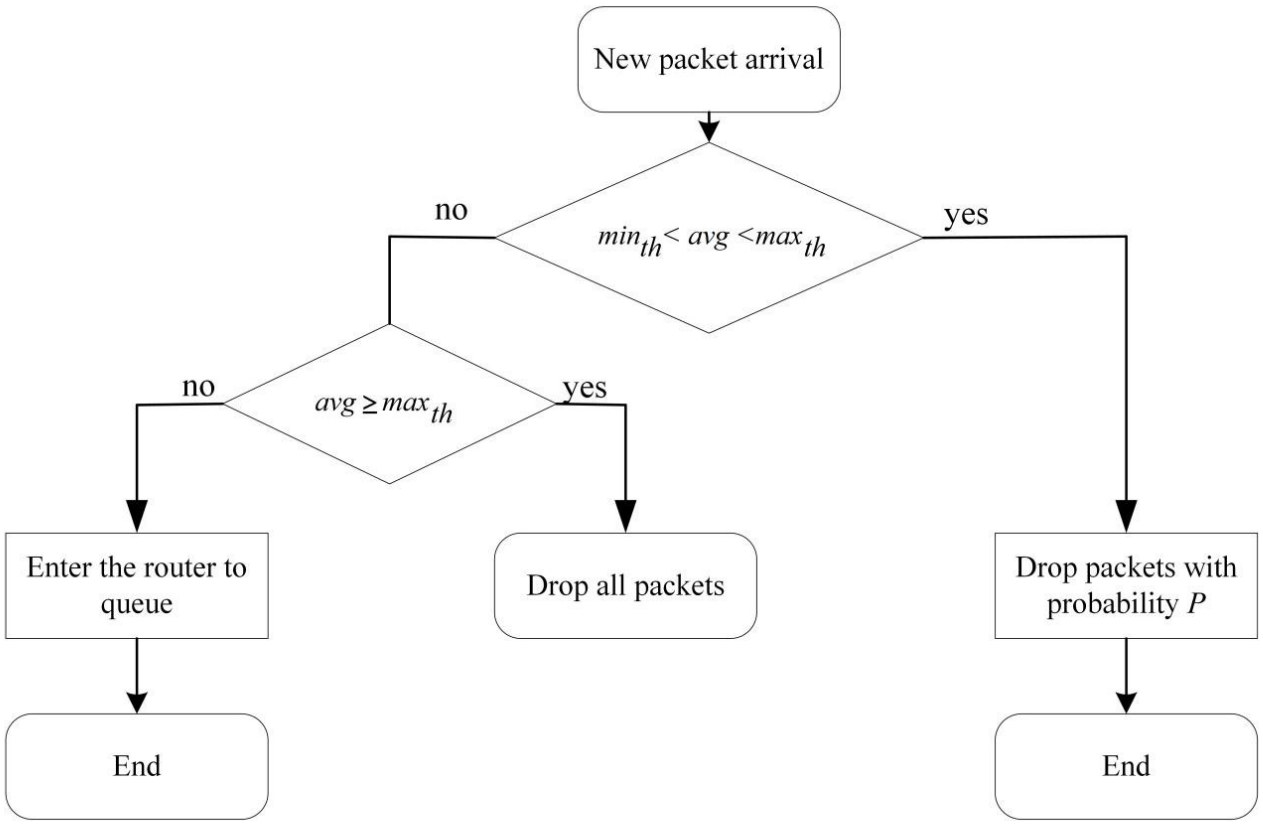
New packet arrival
$min_{th} < avg < max_{th}$
no
yes
$avg \geq max_{th}$
no
yes
Enter the router to queue
Drop all packets
Drop packets with probability *P*
End
End

Figure 5. Flow chart of the RED algorithm.

The RED algorithm has a significant drawback related to its static packet dropping strategy and predetermined parameter settings. This limitation prevents the algorithm from effectively adapting to the dynamic nature of network traffic [34]. In practice, the RED algorithm may perform well when the parameter settings align with the current network load changes. However, it can also result in poor congestion control outcomes, highlighting the algorithm's limited parameter sensitivity and lack of dynamic adaptability [35].

Furthermore, the RED algorithm exhibits linear changes in the packet loss rate regardless of whether the average queue length is near the minimum or maximum threshold. This approach is not practical since the severity of congestion varies at different thresholds. When the average queue length is close to the minimum threshold, congestion levels are typically less severe, suggesting that the packet loss rate should change gradually [36]. Conversely, when the average queue length approaches the maximum threshold, congestion is more critical, warranting a rapid increase in the packet loss rate [37].

To address these issues, an improvement to the RED algorithm can be introduced by adopting the curve RED (CRED) algorithm. CRED addresses the limitations of the original RED algorithm by incorporating non-linear changes into the packet loss rate. This means that the packet loss rate changes at different speeds depending on the proximity to the minimum or maximum threshold. The CRED algorithm aims to provide a more realistic and effective approach to congestion control by adapting the packet loss rate based on the current congestion severity. In summary, the static packet-dropping strategy and predetermined parameter settings of the RED algorithm hinder its adaptability to dynamic network conditions. To overcome these limitations, the CRED algorithm can be implemented, offering non-linear changes in the packet loss rate and improved congestion control based on the current congestion severity.

The advantages of combining congestion recognition based on the ACA and congestion control based on the RED algorithm include: (1) Efficient congestion recognition. The ACA accurately identifies congestion in the network. By monitoring indicators such as traffic, latency, and packet loss, it can quickly detect signs of congestion and respond promptly. (2) Dynamic congestion control. The RED algorithm-based congestion control adjusts the packet drop rate dynamically based on the level of congestion in the network. It uses a lower drop rate to alleviate network load when congestion is mild and gradually increases the drop rate to mitigate congestion as it intensifies. (3) Quick congestion response. With the timely congestion detection provided by the ACA, combining it with the RED-based congestion control enables a fast congestion response. Once congestion is identified, the system can immediately adjust the packet drop rate to prevent further deterioration of congestion and maintain network stability and reliability. (4) Efficient network performance. The combination of congestion recognition based on the ACA and congestion control based on the RED algorithm enables efficient congestion management. It rapidly identifies congestion and dynamically adjusts the packet drop rate based on the actual congestion level, ensuring network stability and reliability. This combined algorithm optimizes network performance, enhances transmission efficiency, and provides a superior user experience.

4.3. The CRED Algorithm

The CRED algorithm is an enhancement to the RED algorithm that specifically addresses its linear packet loss method [38]. This improvement aims to modify the packet loss rate behaviour of the original RED algorithm. In the CRED algorithm, the packet loss rate increases gradually and becomes less sensitive when the congestion level is low. However, once congestion is detected, the packet loss rate accelerates rapidly. This adjustment allows for a more nuanced response to varying congestion degrees, providing a more effective congestion control mechanism. To mathematically model the CRED algorithm, a graphical representation is utilized. This representation includes three points, A, B, and C, placed along the diagonal line. These points are assigned specific coordinates as follows:

$$(\mathrm{min}_{\mathrm{th}}, 0); \left(\frac{\mathrm{min}_{\mathrm{th}} + \mathrm{max}_{\mathrm{th}}}{2}, \frac{\mathrm{max}_p}{2}\right); (\mathrm{max}_{\mathrm{th}}, \mathrm{max}_p) \tag{8}$$

There are two curves with the following expressions for the quadratic equation:

$$P = a_1 \mathrm{avg}^2 + b_1 \mathrm{avg} + c_1 \tag{9}$$

$$P = a_2 \mathrm{avg}^2 + b_2 \mathrm{avg} + \mathrm{c}_2 \tag{10}$$

Let us denote point A as the vertex of the curve represented by Equation (9). In this case, the equation for the symmetry axis can be expressed as follows:

$$-b_1/2a_1 = \mathrm{min}_{\mathrm{th}} \tag{11}$$

Similarly, if we consider point C as the vertex of the curve described by Equation (10), we can determine the equation of the symmetry axis as follows:

$$-b_2/2a_2 = \mathrm{max}_{\mathrm{th}} \tag{12}$$

$$\frac{\mathrm{min}_{\mathrm{th}} + \mathrm{max}_{\mathrm{th}}}{2} = \delta \tag{13}$$

By substituting points A and B into Equation (9) and combining them with Equations (11) and (13), we can obtain the following result:

$$P_a = 0.5 \times \mathrm{max}_p \times \left(2 \times \frac{\mathrm{avg} - \mathrm{min}_{\mathrm{th}}}{\mathrm{max}_{\mathrm{th}} - \mathrm{min}_{\mathrm{th}}}\right)^2 \quad \mathrm{avg} \in (\mathrm{min}_{\mathrm{th}}, \delta_{\mathrm{th}}) \tag{14}$$

By substituting points B and C into Equation (10) and combining them with Equations (12) and (13), we obtain the following results:

$$P_a = max_p - 2max_p \times \frac{(\mathrm{avg} - \mathrm{max_{th}})^2}{(\mathrm{max_{th}} - \mathrm{min_{th}})^2} \quad \mathrm{avg} \in (\mathrm{ffi_{th}}, \mathrm{max_{th}}) \tag{15}$$

Based on the description of the CRED algorithm above, the relevant pseudocode is as follows (Algorithm 2):

Algorithm 2 The CRED algorithm.

```
foreach packet arrival do
    calculate the new average queue size avg;
end
if $\mathrm{min_{th}} < \mathrm{avg} < \mathrm{max_{th}}$ then
    calculate probability $P_a = 0.5 \times max_p \times \frac{(\mathrm{avg}-\mathrm{max_{th}})^2}{(\mathrm{max_{th}}-\mathrm{min_{th}})^2}$;
    with probability $P$, mark the arriving packet;
    if $\delta_{\mathrm{th}} < \mathrm{avg} < \mathrm{max_{th}}$ then
        calculate probability $P_a = max_p - 2max_p \times \frac{(\mathrm{avg}-\mathrm{max_{th}})^2}{(\mathrm{max_{th}}-\mathrm{min_{th}})^2}$;
        with probability $P$, mark the arriving packet;
    else
        if $\mathrm{max}_p \leq \mathrm{avg}$ then
            mark the arriving packet;
        end
    end
end
```

Based on the provided formulas and pseudo-code, we can determine the specific relationship between the packet loss rate and the average queue length in the CRED algorithm. This relationship is illustrated in Figure 6.

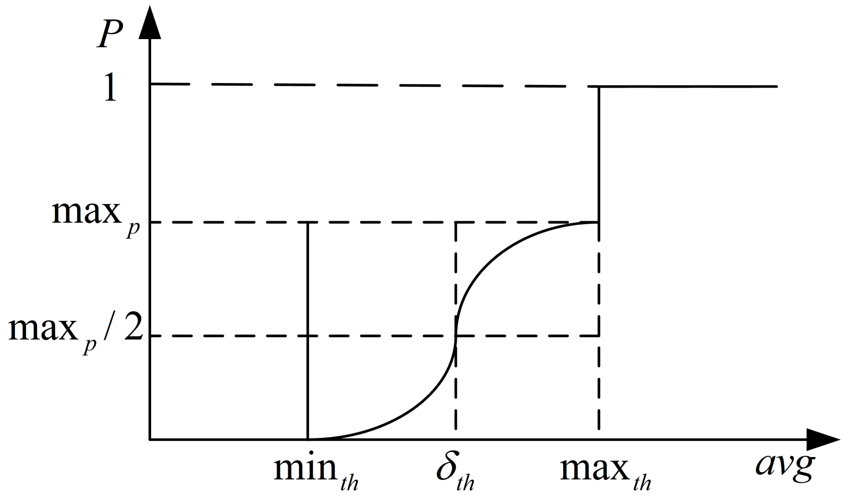

Figure 6. Discarding probability function of the CRED algorithm.

From the figure we can observe the behaviour of the CRED algorithm regarding the relationship between the average queue length avg and the packet loss rate. When avg is in proximity to the minimum threshold, the curve appears smooth, indicating a gradual increase in the packet loss rate. This behaviour allows for a slower response to congestion when the congestion level is low. On the other hand, when avg approaches the maximum threshold, the curve shows a steeper ascent, indicating a rapid increase in the packet loss rate. This characteristic enables the CRED algorithm to effectively handle burst traffic and promptly control congestion in the network.

The improved behaviour of the CRED algorithm aligns better with the actual variations in network traffic. It offers greater practical value in managing congestion and enhancing overall network performance.

5. Simulation Experiment and Result Analysis

5.1. Comparison and Simulation of the RED and CRED Algorithms

The preceding analysis examines the RED and CRED algorithms, highlighting their effectiveness in alleviating congestion. However, determining the algorithm for controlling the average queue length in combination with the ACA and achieving a low-delay optimized path requires simulation. Theoretical comparisons and analyses alone are insufficient. Therefore, the NS2 platform is used here to simulate and analyse these two algorithms. After conducting the simulation analysis, we concluded that the CRED algorithm outperforms the RED algorithm. Considering its superior performance in terms of packet loss rate and delay, the CRED algorithm is ultimately selected for congestion control.

5.1.1. Platform Introduction

NS2 is an object-oriented programming environment and a TCL-OTCL script interpreter. It combines C++ and Otcl, which is referred to as the split object model [39]. In this model, CH (C++ and Otcl) represents the compiled languages, while OTcl corresponds to an interpretive language. TclCL facilitates the association of objects and variables between C++ and OTcl [40]. To understand the specific connection between these languages, the NS2 architecture diagram depicted in Figure 7 provides valuable insights. Additionally, Figure 8 presents the simulation flow chart of NS2, offering a comprehensive overview of the operational flow within NS2.

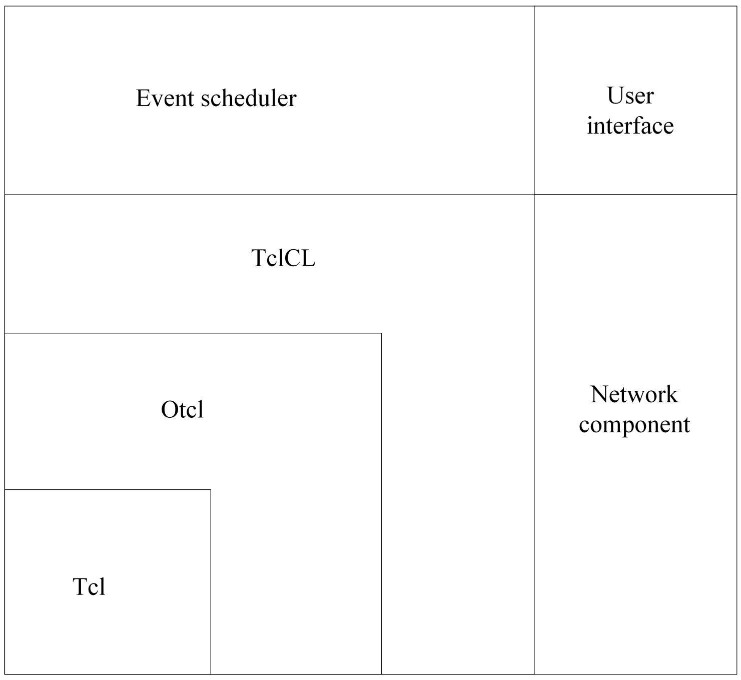

Figure 7. NS2 architecture diagram.

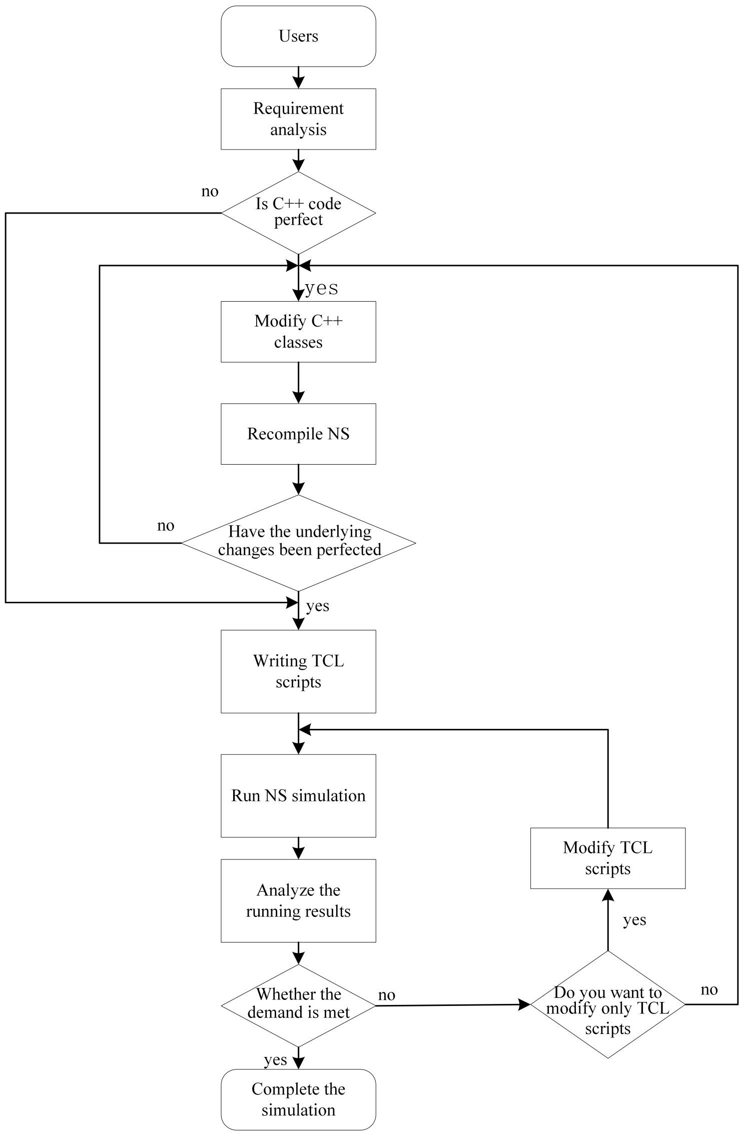

Figure 8. Simulation flow chart of NS2.

5.1.2. Design of Simulation Experiments

We primarily utilize NS2 as the simulation tool to analyse the performance of the proposed algorithms. The simulation network topology diagram is presented in Figure 9. From the depicted topology model, it is evident that S1, S2, and S3 function as data-sending nodes, while RO, R2, and R4 serve as data-receiving nodes. Router1 and Router2 are the intermediary routers in the network. The connection between Router1 and the sending nodes exhibits a delay of 10 ms and a bandwidth of 15 Mb/s. On the other hand, the connection between Router2 and each receiving node experiences a delay of 30 ms and a bandwidth of 15 Mb/s. Router1 and 2 possess a delay of 20 ms and a bandwidth of 1 Mb/s. Additional parameters are set as follows: the minimum threshold min_{th} is set to 10, the maximum threshold max_{th} is set to 30, the maximum packet loss rate max_p is 0.16, and δ is 20. The emulation time is set to 6000 s, and the packet length is fixed at 1000 bytes. Furthermore, the queue length between the routers varies from 10 to 100 packets.

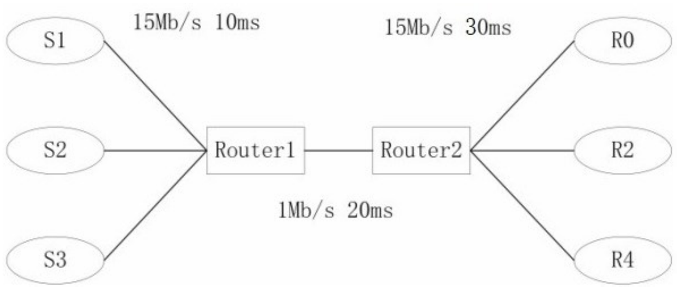

Figure 9. Network topology model.

5.1.3. Simulation Results and Analysis

The performance simulation data graphs of the RED and CRED algorithms are presented below. Figure 10 clearly demonstrates that as the queue length reaches 25 packets, the packet loss rate of both algorithms starts to diverge. As the queue length continues to increase, the packet loss rate of the CRED algorithm consistently remains lower than that of the RED algorithm.

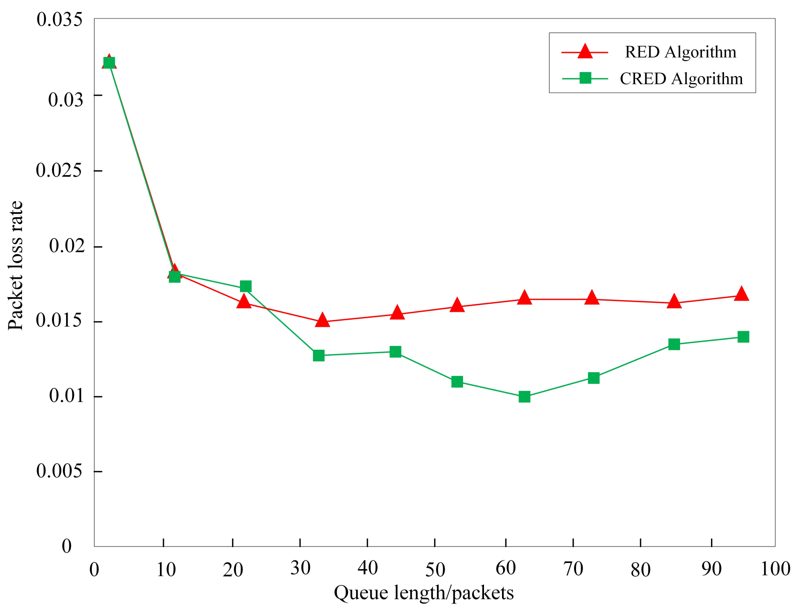

Figure 10. Packet loss rate versus queue length.

The simulation results depicted in Figure 11 highlight an observable delay disparity between the RED algorithm and the CRED algorithm as the queue length reaches 15 packets. Furthermore, as the queue length continues to increase, the delay time of the CRED algorithm consistently remains lower than that of the RED algorithm. The delay of the CRED algorithm remains stable at around 100 ms and does not exceed this threshold. Based on the NS2 simulation analysis, it becomes apparent that the CRED algorithm outperforms the RED algorithm in terms of both packet loss rate and end-to-end delay. The CRED algorithm consistently demonstrates improved performance compared to the RED algorithm. Therefore, in the subsequent simulation experiments, the CRED algorithm will be utilized in conjunction with the ACA to effectively alleviate short-term congestion, allowing the ACA to continue selecting an optimized path with low delay even during congestion scenarios.

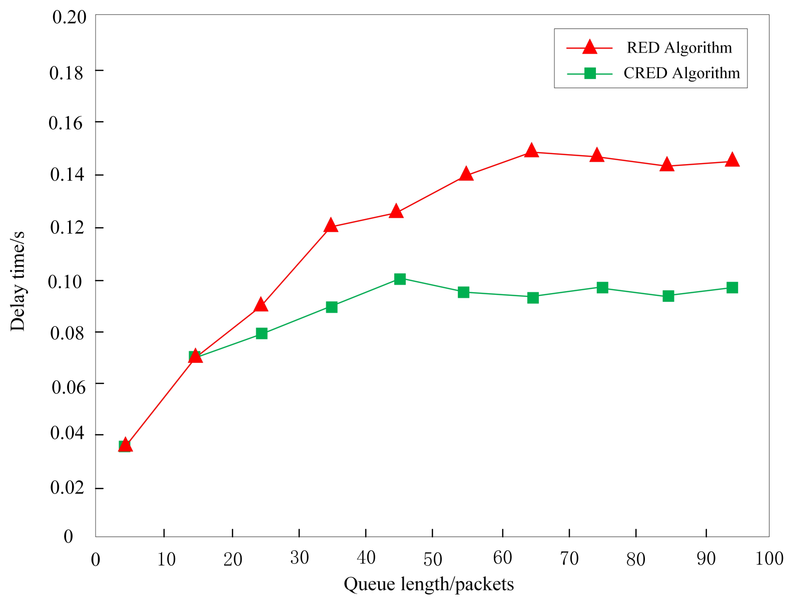

Figure 11. Delay versus queue length.

5.2. Simulation of Congestion Control Based on the CRED Algorithm

The ACA serves as a mechanism to assess changes in cross paths by comparing the total delay of the optimal path before and after congestion. Upon congestion, it becomes evident that the total delay of the optimal path significantly increases, rendering the suboptimal path inadequate for meeting user requirements. Consequently, congestion must be addressed to establish a new optimal path with lower delay. The primary cause of congestion lies in the diminished data processing capacity of the network nodes, leading to a substantial increase in the average queue length of data packets and a subsequent surge in queue delay. This congestion may propagate, resulting in a considerable increase in the optimal path delay obtained through the ACA.

Based on the comparative analysis conducted in the previous section, the enhanced CRED algorithm aligns more closely with actual traffic patterns compared to the RED algorithm. Thus, it is feasible to combine the CRED algorithm with the ACA. By leveraging the discard strategy of the CRED algorithm, even during sudden congestion events, the queuing delay can be controlled to a reasonable level, allowing the ACA to identify an optimized path with lower total delay that fulfils user requirements. In the following sections, we will present a simulation experiment involving the combination of the CRED algorithm and ACA for network congestion control. We will subsequently analyse the corresponding results to evaluate the effectiveness of this approach.

5.2.1. Design of the Simulation Experiments

In this experiment, the MATLAB platform is utilized. The first step involves constructing a random network topology model with N nodes using the Waxman algorithm. In this case, N is set to 19, but it is important to note that a larger number of nodes would yield more realistic data. Based on the randomly generated link lengths, a node location distance matrix T is constructed, with unconnected nodes designated as such. The resulting network topology model is illustrated in Figure 12.

In the MM/1 queuing model, the following parameters are defined: the number of data packets to be sent, which should ideally exceed 1000 to observe significant queuing delay effects at the node. For this experiment, the number of data packets is set to 1000, with each packet having a length of 1000 bytes. Additionally, the service rate u and packet arrival rate λ are determined for each node. By employing the M/M/1 queuing model, the queuing processing delay of each network node in the network topology is generated, resulting in the construction of the network queuing delay matrix Q. The total network delay matrix W is then obtained by combining the network transmission delay matrix T and the network queuing delay matrix Q. In the CRED algorithm, the following parameters are set: $\min_{\text{th}} = 30$, $\delta = 20$, and an initial value of $max_p = 0.16$.

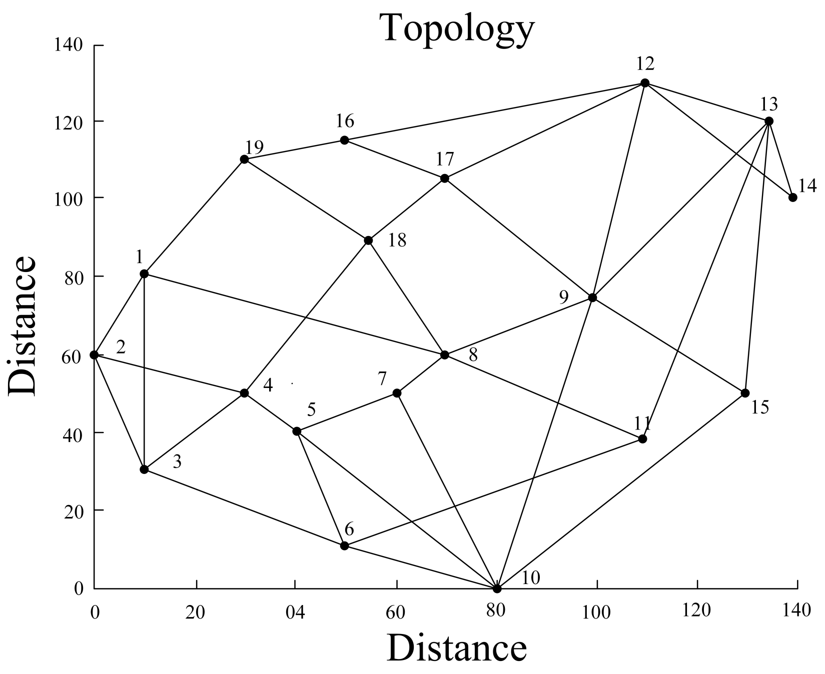

Figure 12. Random network topology model.

5.2.2. Simulation Results and Analysis

By analysing several sets of experimental data, it becomes evident that the integration of the CRED algorithm enables the ACA to obtain an optimized path with low delay even in the presence of congestion. Two specific datasets are selected for further analysis. In these experiments, the source node is set to 1 and the destination node to 14. By running the ACA, the optimized path from 1 to 14 and its corresponding total delay are obtained. To simulate congestion, the node parameters on the cross path, specifically the service rate u, are adjusted to significantly reduce the data processing capacity. This allows us to observe the total path delay after congestion. Finally, with the aid of the CRED algorithm, the optimized total path delay after congestion alleviation is determined, resulting in the third set of data. The results of these analyses are depicted in Figure 13.

In Figure 13, the vertical axis represents the total delay in milliseconds, while the horizontal axis represents the iteration times of the ACA. After incorporating the CRED algorithm, the ACA reaches a stable state of convergence after approximately 48 iterations. At this point, there is still a noticeable difference of around 100 ms between the optimized path delay after congestion relief and the initial path delay. However, compared to the optimized path delay without any congestion control, the total delay is significantly reduced by approximately 300 ms. This demonstrates the effectiveness of the CRED algorithm in controlling congestion, enabling the ACA to obtain a path with a total delay that closely matches the original optimized path even after congestion occurs.

From the experimental data, we can select a specific group and set the source node as 3 and the destination node as 13. By running the ACA, we can obtain an optimized path from node 3 to node 13 and calculate the corresponding total delay. After this, the node parameters on the cross path, specifically the service rate u, are adjusted to simulate a sharp decrease in data processing capacity, resulting in congestion. This allows us to determine the second optimized path total delay after congestion. Finally, by introducing the CRED algorithm, we can alleviate the congestion and obtain the third optimized path total delay after congestion relief. The results of these simulations are illustrated in Figure 14.

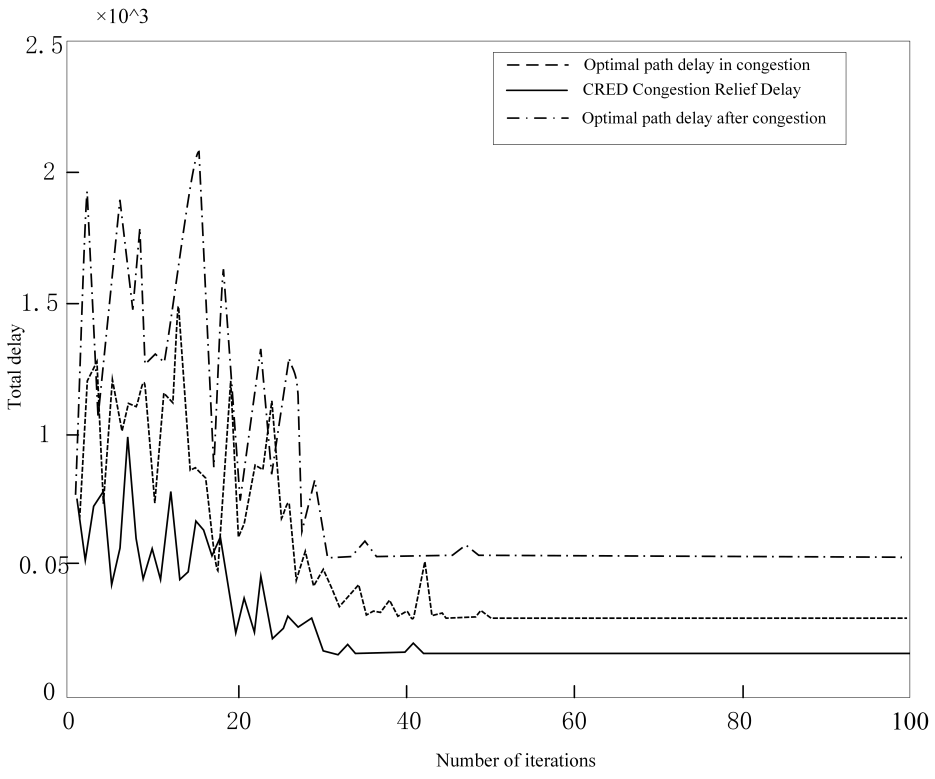

Figure 13. Comparison of delay before and after congestion of the optimized path (1–14).

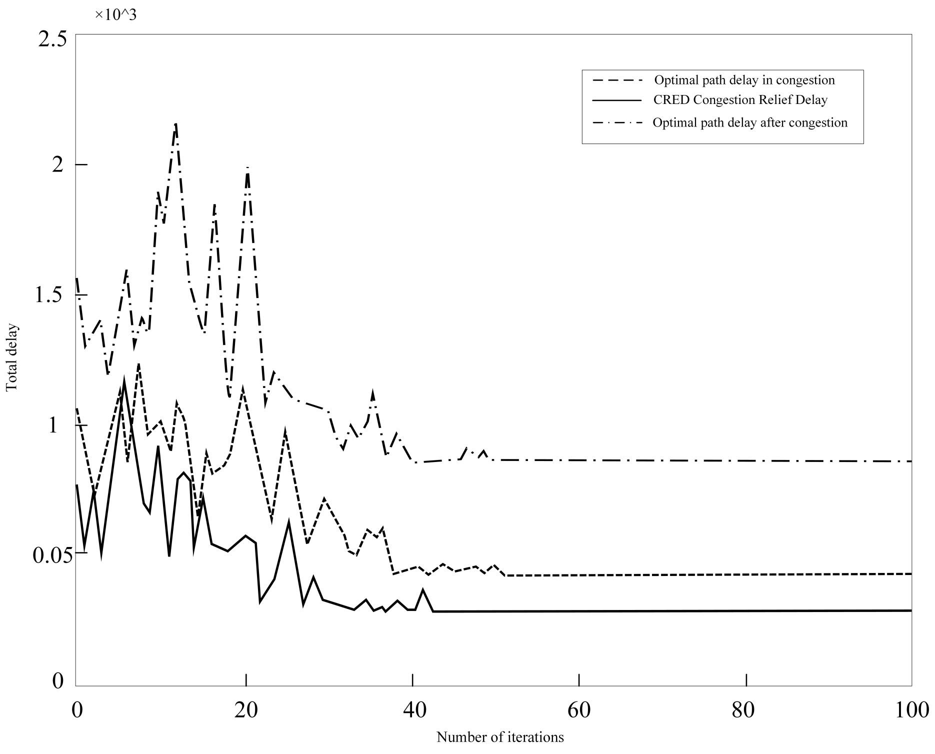

Figure 14. Comparison of delay before and after congestion of the optimized path (3–13).

Figure 14 illustrates the results of the simulation. After introducing the CRED algorithm, the ACA reaches a stable state after approximately 50 iterations. In this state, the gap between the path delay before congestion and the optimized path delay after congestion is relatively small, with an increase of around 70 ms. However, compared to the path delay without congestion relief, the total delay is significantly reduced by approximately 500 ms. These findings demonstrate the effectiveness of the CRED algorithm in controlling congestion, further confirming that the ACA can still achieve a path with lower delay even after congestion occurs.

The proposed ACA and CRED algorithms are designed to be applicable in real-time applications. The time required to find a solution depends on several factors, including the problem's size and complexity, the efficiency of the algorithm implementation, and the availability of computational resources. To demonstrate the computational efficiency of the proposed method, the execution time of our algorithm on different node sizes is given in Table 1. From the table, it can be seen that the computation time of the proposed method increases linearly as the number of nodes increases, verifying that the algorithm has a better time complexity and computational efficiency. Furthermore, the existing resource allocation algorithm fails to fully solve the problem of the non-linear increase in computation time caused by the increase in the number of nodes. The main reason for the improvement in computational efficiency of the proposed algorithm is the optimization of the RED algorithm, thus solving the problem of inconsistency in describing the actual traffic. Therefore, the proposed algorithm has good scalability in real-time application scenarios.

Table 1. Execution time of the algorithm for different node sizes.

Serial Number	Nodes Count	Computational Time (ms)
1	10	12
2	20	27
3	30	35
4	40	48
5	50	69
6	60	72
7	70	100

6. Conclusions

This paper focuses on the application of swarm intelligent computing in the context of CAEVs IoT. It begins by discussing the current development status of CAEVs and highlights the issue of resource allocation in their applications. To address this problem, the paper introduces a detailed resource allocation method based on the ACA. Furthermore, the paper provides a comprehensive explanation of the principles behind the RED algorithm, highlighting its advantages and disadvantages. Recognizing the inconsistency in the RED algorithm for describing actual traffic, an improved variant called the CRED algorithm is proposed and its principles are elaborated upon. Through simulation and comparison using the NS2 platform, the performance of the CRED algorithm is demonstrated to be superior. Consequently, the paper concludes by advocating for the combination of the CRED algorithm and ACA to effectively alleviate short-term congestion. Subsequent simulations further validate that this combination successfully mitigates congestion and enables the discovery of optimized paths with lower delay. In summary, this paper presents a feasible and practical solution based on the ACA for the resource allocation challenge in CAEVs IoT applications. Compared with existing resource allocation methods based on ACAs, our proposed method has the following three improvements. First, in resource allocation, the proposed method integrates security, reliability and latency issues specific to CAEVs IoT scenarios. Second, we improve the inconsistency problem of the RED algorithm in describing actual traffic and integrate it with the ACA to improve the path optimization. Finally, we optimize the pheromone of the probabilistic transfer formula in the ACA to effectively improve the shortcomings of resource allocation in many aspects.

In the future, we will expand the proposed algorithm to edge and fog computing. Furthermore, we will also consider the challenges of resource allocation in specific IoT scenarios such as smart cities and smart driving. Meanwhile, we plan to introduce deep reinforcement learning to detect and multiplex link resources to reduce the network signalling overhead and achieve effective V2V communications.

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

Funding: This research was funded by the Henan Science and Technology Plan Project under Grant No. 232102210112, Grant No. 232102520004.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no known competing interests or personal relationships that could appear to influence the work reported in this paper.

References

1. Smuts, M.; Scholtz, B.; Wesson, J. Issues in implementing a data integration platform for electric vehicles using the internet of things. In *Internet of Things. Information Processing in an Increasingly Connected World: First IFIP International Cross-Domain Conference, IFIPIoT 2018, Held at the 24th IFIP World Computer Congress, WCC 2018, Poznan, Poland, 18–19 September 2018, Revised Selected Papers 1*; Springer: Cham, Switzerland, 2019; pp. 160–177.
2. Wu, T.; Qu, D.; Zhang, G. Research on lora adaptability in the leo satellites internet of things. In Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 131–135.
3. Zhang X.; Dahu, W. Application of artificial intelligence algorithms in image processing. *J. Vis. Commun. Image Represent.* **2019**, *61*, 42–49. [CrossRef]
4. Chen, M. Automatic console image processing aided by improved particle swarm computing intelligent algorithm. *Math. Probl. Eng.* **2022**, *2022*, 3475806. [CrossRef]
5. Li, S.; Li, W.; Lu, J. Research on logistics service transaction blockchain and ant colony smart contract algorithm. *Comput. Eng. Appl.* **2019**, *55*, 28–34.
6. Du, X.; Du, C.; Chen, J.; Liu, Y. An energy-aware resource allocation method for avionics systems based on improved ant colony optimization algorithm. *Comput. Electr. Eng.* **2023**, *105*, 108515. [CrossRef]
7. Xu, L.; Li, Y.-P.; Li, Q.-M.; Yang, Y.-W.; Tang, Z.-M.; Zhang, X.-F. Proportional fair resource allocation based on hybrid ant colony optimization for slow adaptive ofdma system. *Inf. Sci.* **2015**, *293*, 1–10. [CrossRef]
8. Al-Masri, E.; Souri, A.; Mohamed, H.; Yang, W.; Olmsted, J.; Zhang, M.; Kotevska, O. Energy-efficient cooperative resource allocation and task scheduling for internet of things environments. *Internet Things* **2023**, *23*, 100832. [CrossRef]
9. Ari, A.A.A.; Gueroui, A.; Titouna, C.; Thiare, O.; Aliouat, Z. Resource allocation scheme for 5 g c-ran: A swarm intelligence based approach. *Comput. Netw.* **2019**, *165*, 106957. [CrossRef]
10. Zhang, H.; Wang, Z.; Liu, K. V2x offloading and resource allocation in sdn-assisted mec-based vehicular networks. *China Commun.* **2020**, *17*, 266–283. [CrossRef]
11. Zhu, X.; Liu, F.; Zeng, Z.; Guo, C.; Chen, J. Content-driven joint resource allocation based on vehicle-edge synergy in vehicular networks. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021; pp. 1–7.
12. Wang, B.; Zhang, R.; Chen, C.; Cheng, X.; Yang, L.; Jin, Y. Interference hypergraph-based 3d matching resource allocation protocol for noma-v2x networks. *IEEE Access* **2019**, *7*, 90789–90800. [CrossRef]
13. Wu, W.; Dong, J.; Sun, Y.; Yu, F.R. Heterogeneous markov decision process model for joint resource allocation and task scheduling in network slicing enabled internet of vehicles. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 1118–1122. [CrossRef]
14. Zhang, X.; He, Z.; Sun, Y.; Yuan, S.; Peng, M. Joint sensing, communication, and computation resource allocation for cooperative perception in fog-based vehicular networks. In Proceedings of the 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP), Changsha, China, 20–22 October 2021; pp. 1–6.
15. Fan, C.; Li, B.; Wu, Y.; Zhang, J.; Yang, Z.; Zhao, C. Fuzzy matching learning for dynamic resource allocation in cellular v2x network. *IEEE Trans. Veh. Technol.* **2021**, *70*, 3479–3492. [CrossRef]
16. Zheng, K.; Zhang, Z.; Gauthier, J. Blockchain-based intelligent contract for factoring business in supply chains. *Ann. Oper. Res.* **2020**, *308*, 777–797. [CrossRef]
17. Gong, W.; Mingyang, S.; Huiyang, S.; Zhang, Y.; Ziming, T. An adaptive threshold of remaining energy based ant colony routing algorithm. *Xibei Gongye Daxue Xuebao/J. Northwestern Polytech. Univ.* **2022**, *40*, 442–449.

18. Cui, Y.; Ren, J.; Zhang, Y. Path planning algorithm for unmanned surface vehicle based on optimized ant colony algorithm. *IEEJ Trans. Electr. Electron. Eng.* **2022**, *17*, 1027–1037. [CrossRef]
19. Ntakolia, C.; Lyridis, D.V. A comparative study on ant colony optimization algorithm approaches for solving multi-objective path planning problems in case of unmanned surface vehicles. *Ocean Eng.* **2022**, *255*, 111418. [CrossRef]
20. Huan, L.; Ning, Z.; Qiang, L. Uav path planning based on an improved ant colony algorithm. In Proceedings of the 2021 4th International Conference on Intelligent Autonomous Systems (ICoIAS), Wuhan, China, 14–16 May 2021; pp. 357–360.
21. Li, F.; Liu, M.; Xu, G. A quantum ant colony multi-objective routing algorithm in wsn and its application in a manufacturing environment. *Sensors* **2019**, *19*, 3334. [CrossRef]
22. Buniyamin, N.; Ngah, W.W.; Sariff, N.; Mohamad, Z. A simple local path planning algorithm for autonomous mobile robots. *Int. J. Syst. Appl. Eng. Dev.* **2011**, *5*, 151–159.
23. Senthilkumar, M.; Kavitha, V.; Kumar, M.S.; Raj, P.A.C.; Shirley, D.R.A. Routing in a wireless sensor network using a hybrid algorithm to improve the lifetime of the nodes. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1084*, 012051. [CrossRef]
24. Zhang, R.; Zhang, J.; Liu, L.; Yan, Z.; Dong, M. Research on active detection method of network congestion. In Proceedings of the GLOBECOM 2022—2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 4685–4690.
25. Abdolhosseinzadeh M.; Alipour, M.M. Design of experiment for tuning parameters of an ant colony optimization method for the constrained shortest hamiltonian path problem in the grid networks. *Numer. Algebr. Control. Optim.* **2021**, *11*, 321. [CrossRef]
26. Waxman, B.M. Routing of multipoint connections. *IEEE J. Sel. Areas Commun.* **1988**, *6*, 1617–1622. [CrossRef]
27. Dolfing, A.G.; Leuven, J.R.; Dermody, B.J. The effects of network topology, climate variability and shocks on the evolution and resilience of a food trade network. *PLoS ONE* **2019**, *14*, e0213378. [CrossRef]
28. Zhang, X.; Jing, C.; Tang, F.; Fowler, S.; Cui, H.; Dong, X. Joint redundant and random network coding for robust video transmission over lossy networks. *Mob. Inf. Syst.* **2012**, *8*, 213–230. [CrossRef]
29. Hu, H.; Huang, L. Linear stability and hopf bifurcation in an exponential red algorithm model. *Nonlinear Dyn.* **2010**, *59*, 463–475. [CrossRef]
30. Xu, L.; Pan, D. An improved genetic ant colony algorithm for solving tsp problem. *Intell. Comput. Appl.* **2017**, *7*, 34–36.
31. Singha, S.; Jana, B.; Mandal, N.K. Active queue management in red considering critical point on target queue. *J. Interconnect. Netw.* **2021**, *21*, 2150017. [CrossRef]
32. Jiang, M.; Liu, F. Pbred: An improved red algorithm based on priority. *Comput. Eng. Sci.* **2015**, *37*, 245–251.
33. Nguyen, T.-H.; Jung, J.J. Ant colony optimization-based traffic routing with intersection negotiation for connected vehicles. *Appl. Soft Comput.* **2021**, *112*, 107828. [CrossRef]
34. Zhang, Z.; Ning, H.; Shi, F.; Farha, F.; Xu, Y.; Xu, J.; Zhang, F.; Choo, K.-K.R. Artificial intelligence in cyber security: Research advances, challenges, and opportunities. *Artif. Intell. Rev.* **2022**, *55*, 1029–1053. [CrossRef]
35. Hou, W.; Xiong, Z.; Wang, C.; Chen, H. Enhanced ant colony algorithm with communication mechanism for mobile robot path planning. *Robot. Auton. Syst.* **2022**, *148*, 103949. [CrossRef]
36. Mahi, M.; Baykan, Ö.K.; Kodaz, H. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Appl. Soft Comput.* **2015**, *30*, 484–490. [CrossRef]
37. Arivarasan, S. An energy efficient qos routing protocol based on red deer algorithm in manet. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 1461–1471.
38. Yuan, J.; Zhao, L.; Huang, C.; Xiao, M. A novel hybrid control technique for bifurcation in an exponential red algorithm. *Int. J. Circuit Theory Appl.* **2020**, *48*, 1476–1492. [CrossRef]
39. Ousterhout, J.K. *Tcl: An Embeddable Command Language*; Citeseer: Princeton, NJ, USA, 1989.
40. Nawaz Jadoon, R.; Fayyaz, M.; Zhou, W.; Khan, M.A.; Mujtaba, G. Pcoi: Packet classification-based optical interconnect for data centre networks. *Math. Probl. Eng.* **2020**, *2020*, 2903157. [CrossRef]