Wireless Communications and Mobile Computing

Emerging Advanced Data Offloading Technologies for Urban Internet of Things

Lead Guest Editor: Pengfei Wang
Guest Editors: Chi Lin, Ning Wang, and Rongfei Zeng

Emerging Advanced Data Offloading Technologies for Urban Internet of Things

Wireless Communications and Mobile Computing

Emerging Advanced Data Offloading Technologies for Urban Internet of Things

Lead Guest Editor: Pengfei Wang
Guest Editors: Chi Lin, Ning Wang, and Rongfei Zeng

Chief Editor

Associate Editors

Academic Editors

Contents

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5055019, 8 pages
https://doi.org/10.1155/2022/5055019

Research Article

A New Way to Model the Wireless Sensor Network Maintenance Job

Ziqi Wei[1] **and Mike MacGregor**[2]

[1]*School of Software, Tsinghua University, Beijing, China 100084*
[2]*Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E8*

Correspondence should be addressed to Ziqi Wei; ziqi2@ualberta.ca

Received 24 June 2021; Revised 29 December 2021; Accepted 4 January 2022; Published 25 April 2022

Academic Editor: Ning Wang

For the maintenance of an urban wireless sensor network, the staff's travel route greatly affects the whole network's response time. Every time the network reports an error, the staff needs to find the best route to minimize the time spent on the way to the error point. The difficulty of the problem is that although the entire network fails, the error point remains unclear. In this paper, the staff's route planning is modeled as an NP-complete problem, MWLP (Minimum Weighted Latency Problem). It is a problem of finding the best route for a moving agent to satisfy multiple customers' different demands as much as possible. To solve the problem, we propose a heuristic algorithm which borrows the idea from a biological computing model called P_system. In the proposed algorithm, different classic heuristics work as separate "membranes" to accomplish their own jobs. They also collaborate under some mechanism to search for a better result. We designed the cell's structure to balance the different heuristics' time consumption and searching capacity. With this design, all the heuristics can cooperate properly in the proposed heuristic algorithm. To enhance the algorithm's efficiency, we also introduced a way to run it in parallel. The numerical experiments show that the proposed algorithm is very competitive compared with classic heuristic algorithms and helps eliminate the whole network delay as well.

1. Introduction

Koutsoupias et al. described the following situation in 1996: a treasure is hidden in one certain cave in a mountain area. An adventurer is trying to find it. However, the mountain area is full of different caves, and they all look alike. The good news is that the adventurer has a treasure map. Every cave's position and the probability of whether the treasure is placed inside it are marked on the map. Now, the adventurer is planning his search path. His goal is to minimize the distance he will travel before finding the treasure. The problem the adventurer is facing is called the Graph Searching Problem (GSP) in [1]. Wu renamed it MWLP in [2]. Two years later, Sitters proved it to be NP-hard in [3].

MWLP is formulated as follows: given n vertices $v_1, \cdots, v_n$ of a weighted (both edge and vertex) graph G. The vertices' weights are represented by $w(v)$. MWLP is looking for a tour π, starting at a fixed vertex s and visiting every other vertex of G. For this tour, the sum of all the vertices' weighted latencies (first arrival times of all the vertices $d_\pi(s, v)$ multiplied by their weights $w(v)$) must be minimized. MWLP is presented as

$$\min_{\pi} \sum_{v \in G} w(v) d_\pi(s, v). \tag{1}$$

The vertices' weights are sometimes noted as the probability of finding a treasure and limited to the interval $(0, 1]$. In this case, MWLP is formulated as follows:

$$\min_{\pi} \sum_{v \in G} pr(v) d_\pi(s, v). \tag{2}$$

By simply taking the sensor's error probability as $pr(v)$, MWLP can help to model the mobile agent's route planning problem in a wireless sensor network. To minimize the expression above, we shall minimize the time maintenance staff uses to find the failed sensor. Compared to TSP (Travelling Salesman Problem) and MLP (Minimum Latency Problem), MWLP pays more attention to the nodes' requirements in the network, which makes it more suitable for this environment.

MWLP can be seen as the vertex-weighted generalization of the Minimum Latency Problem (MLP), which is widely studied and proved to be NP-hard in [4]. Koutsoupias et al. also proposed a way to convert MWLP into MLP in polynomial time in [1]. With that approach, any algorithm designed for MLP can be used to solve MWLP. However, in practical cases, because of the shortcomings of the conversion process (weight expression form and graph structure are limited, as we explained in [5]), this method is inefficient. Wu proposed a DP (dynamic programming) algorithm to solve some special MWLPs in [2]. To the best of our knowledge, besides this, neither exact nor approximate algorithms designed for MWLP have been proposed. Though it can find the global optimum in a relatively short time, Wu's method still has the drawbacks of all DP algorithms.

Heuristic algorithms are a fast developing research field during the past several years. Different heuristic algorithms are designed for MLP, such as [6–9]. In recent years, using composite heuristic algorithms to solve NP-hard problems is becoming a trend. For example, Ha et al. proposed a metaheuristic combining tabu search (TS) and Variable Neighbourhood Search (VNS) to solve MLP in [10]. They used VNS to find the next feasible solution and TS to avoid searching one solution multiple times. Lenin et al. proposed a metaheuristic for a problem very similar to MLP which combined TS and simulated annealing (SA) in [11]. Their algorithm added the tabu list into an SA algorithm. The experimental results showed that the composite algorithm produced better results compared to either SA or TS alone. In [6], Ahmed proposed a hybrid genetic algorithm which combines the genetic algorithm (GA) and local search to solve MLP and got good results. Koutsoupias' conversion method can be used to apply these algorithms to MWLP. However, the problem scale will expand dramatically. It will cause both time and space consumption to increase exponentially, which makes a heuristic algorithm designed for MWLP necessary. In [5], we studied the heuristics used to solve MWLP. Five classic heuristic algorithms including TS, SA, GA, Particle Swarm Optimization (PSO), and ant colony optimization (ACO) were all redesigned to solve MWLP and tested effectively.

In this paper, we propose a heuristic algorithm based on our previous research. We borrowed the idea from P_system and optimized the five classic heuristics to embed them into the P_system structure. We also applied two ACO methods with different pheromone update mechanisms. We evaluated our approach with the widely used dataset for graph theory problems, TSPLib. The experiments show that this metaheuristic yields greatly improved results compared to the classic heuristics.

The rest of this paper is organized as follows. Section 2 introduces the heuristic based on P_system. Experimental results are shown in Section 3, and Section 4 gives the conclusion.

2. The Algorithm Based on P_system

P_system (also known as membrane computing) is a computational model proposed by Păun in [12]. Păun is a computer scientist whose research field is biological computing. Inspired by the biological cell's structure, he proposed the P_system model to simulate the process that a cell uses to handle compounds in a circulation system consisting of multilayer membranes. Thanks to the advantage of its inherent structure, many other algorithms can easily be embedded into a P_system, so that many metaheuristic algorithms based on P_system have been proposed recently.

For instance, Niu et al. proposed a metaheuristic algorithm based on the structure of P_system and used ant colony optimization to solve the Capacitated Vehicle Routing Problem (cVRP) in [13]. Another heuristic algorithm, which is also based on the structure of P_system, was given by Yan et al. in [14]. They used SA to communicate between different membranes to solve the weight optimization problem for case-based reasoning. In the following, we will introduce P_system and then go to the details of the proposed composite algorithm.

2.1. Introduction of P_system. Similar to other natural computation models, P_system is inspired by an existing process in the natural environment. It simulates the chemical processes that occur between different layers of the membranes within a cell. During processing, the cell is considered a frame. The chemical compounds within it are the subjects that are involved in the process. When passing through a membrane, the compounds are involved in some chemical reactions with other compounds or some prestored catalysts inside the membrane. Through iterations of this process, new compounds are produced and then involved in subsequent processing. The final output is produced after some stopping criterion is met.

In general, a P_system consists of three parts: the hierarchical structure of membranes, a multiset to represent the objects, and the evolutionary rules. The hierarchical structure of membranes defines the computing architecture. It limits the sequence and affiliation of different membranes. The object multiset represents the developing state during the computing process. The evolutionary rules define the method P_system uses to develop the results. The hierarchical membrane structure guarantees that the architecture can handle multiple heuristics in one algorithm.

The composite heuristic algorithm based on P_system has attracted much attention for two reasons. The first is that the computing model's structure is very suitable for combining multiple algorithms. The second is that the no-free-lunch (NFL) theorem guarantees its computational capacity.

The famous NFL theorem was proposed and proven by Wolpert and Macready in [15]. It proves that if an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems. That is, different algorithms have specific advantages and disadvantages when solving one certain problem. Composite algorithms can synthesize the advantages of different algorithms, which in theory gives them better searching ability than the individual components considered separately. People designed various mechanisms such as diversification in

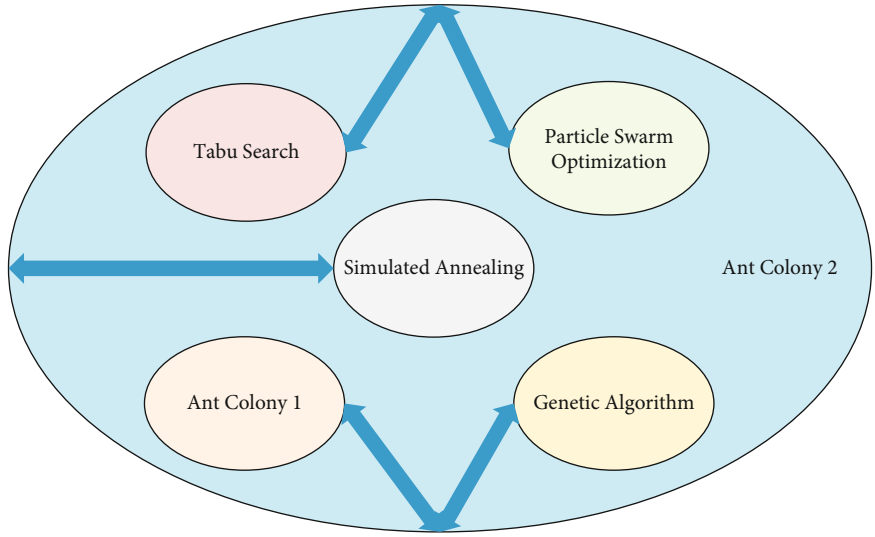


Figure 1: Illustration of the algorithm based on P_system.

TS and mutation in GA to avoid heuristic algorithms being trapped in local optima. However, being trapped in local optima is still inevitable because of the limitations of single heuristics. Composite algorithms can avoid their limitations.

2.2. Heuristic Based on P_system. Based on P_system, we designed a composite heuristic algorithm to solve MWLP. The algorithm consists of six different heuristic algorithms. They are PSO, TS, SA, GA, and two different ACOs. The two ACOs update their pheromone data in different ways. Though one of them produces better results (noted as ACO2 in this paper) than the other (noted as ACO1) in most circumstances when they are used alone to solve MWLP, as shown in [5], we choose to use both of them in the composite algorithm because of the NFL theorem. The parameters of the heuristics are the same as in [5]. The algorithm's main structure is shown in Figure 1.

As shown in the illustration, the "cell" consists of five inner membranes and an outer membrane. The five inner membranes are independent of each other, whereas they are all connected to the outer membrane. The inner membranes conduct PSO, TS, SA, GA, and ACO1, respectively, and the outer membrane runs ACO2. Previous experiments showed that ACO2 produces the best results among the six heuristic algorithms when they are running alone (same iteration number) to solve MWLP. Therefore, we consider ACO2 as the most suitable algorithm for the outer membrane.

When using this metaheuristic to solve MWLP, the mobile agent's route is stored in vertex arrays. Ten arrays are included in a group. The groups are considered the compounds to be transmitted between membranes. At the beginning, a group of vertex arrays is generated by using a greedy algorithm. This group is used as the initial solutions of the inner membranes. After the five optimization processes (inner membranes) are performed, each of them outputs a group of vertex arrays, consisting of the ten best solutions found by each heuristic. The five groups are combined as the initial solutions of the outer membrane. After the outer membrane finishes its optimization process, it outputs a group of the ten best solutions. This group of solutions is returned to the inner membranes as their initial solutions. At this time, the algorithm comes back to the starting state. We call this an iteration. The algorithm will iterate a predetermined number of times until the outer membrane outputs the final solution. The procedure's pseudocode is shown in Algorithm 1.

```
procedure P_system
  ini_seq = GreedyAlg(weighted_map, priority_seq)
  best_val = ComputeValue(ini_seq)
  iteration_number ⟵ 0
  while iteration_number < iteration_time do
    ac1_best_seq = ACO1_Alg(ini_seq)
    sa_best_seq = SA_Alg(ini_seq)
    ts_best_seq = TS_Alg(ini_seq)
    pso_best_seq = PSO_Alg(ini_seq)
    ga_best_seq = GeneticAlg(ini_seq)
    new_seq = combination of the five seqs above
    ac2_best_seq = ACO2_Alg(new_seq)
    temp_val = ComputeValue(ac2_best_seq)
    if temp_val < best_val then
      best_val = temp_val
    end if
    ini_seq = ac2_best_seq
    iteration_number + +
  end while
  return best_val
end procedure
```

Algorithm 1: P_system algorithm

2.3. Parallel Technique. It is obvious that the heuristic contains relatively many subalgorithms that make its computing scale large. Due to this, the time consumption of the P_system is its main drawback. However, thanks to the use of the membrane computing model, the whole optimization procedure

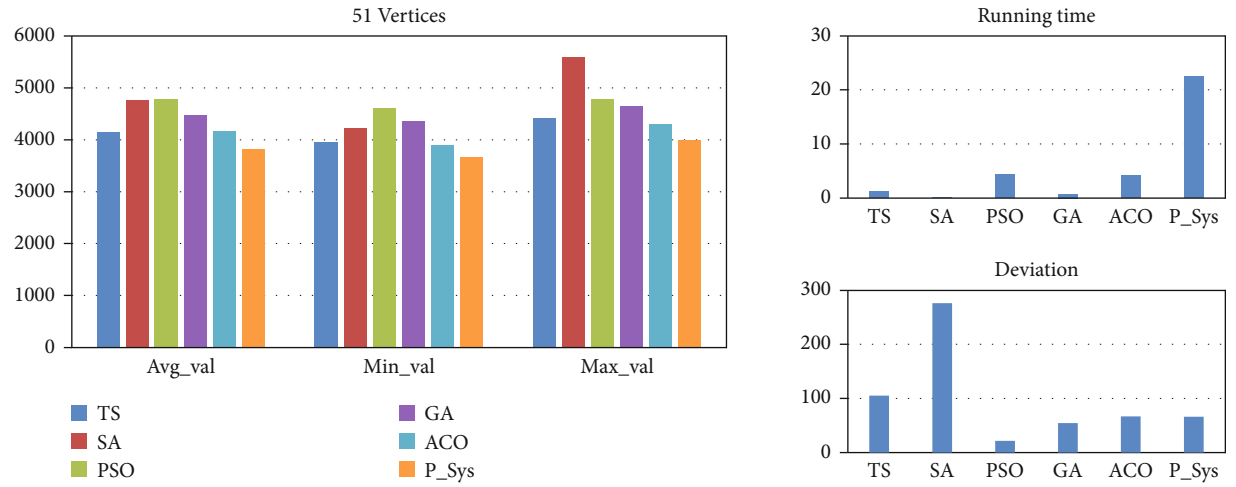


Figure 2: Experimental results of the map with 51 vertices.

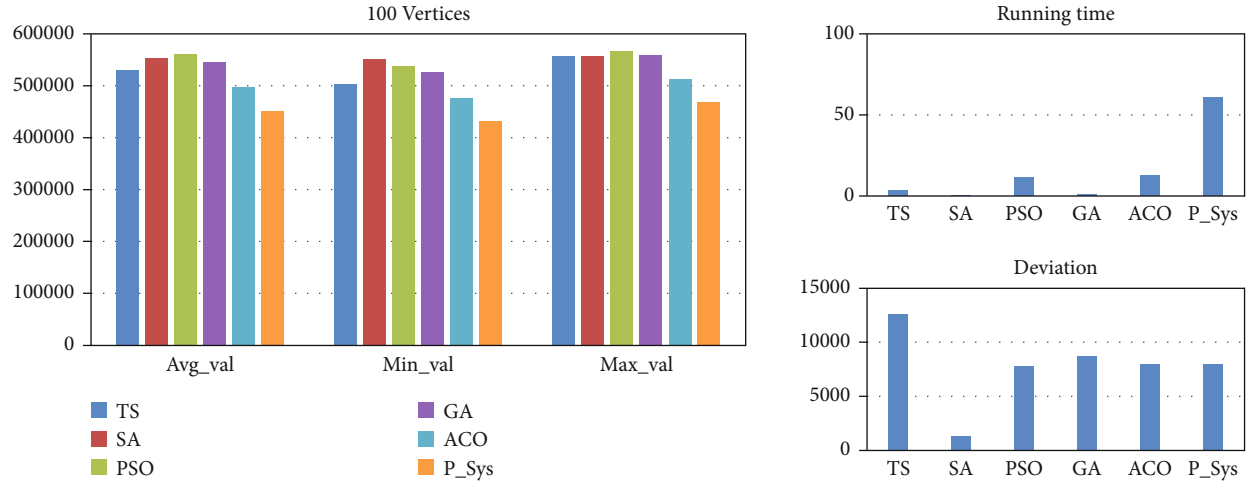


Figure 3: Experimental results of the map with 100 vertices.

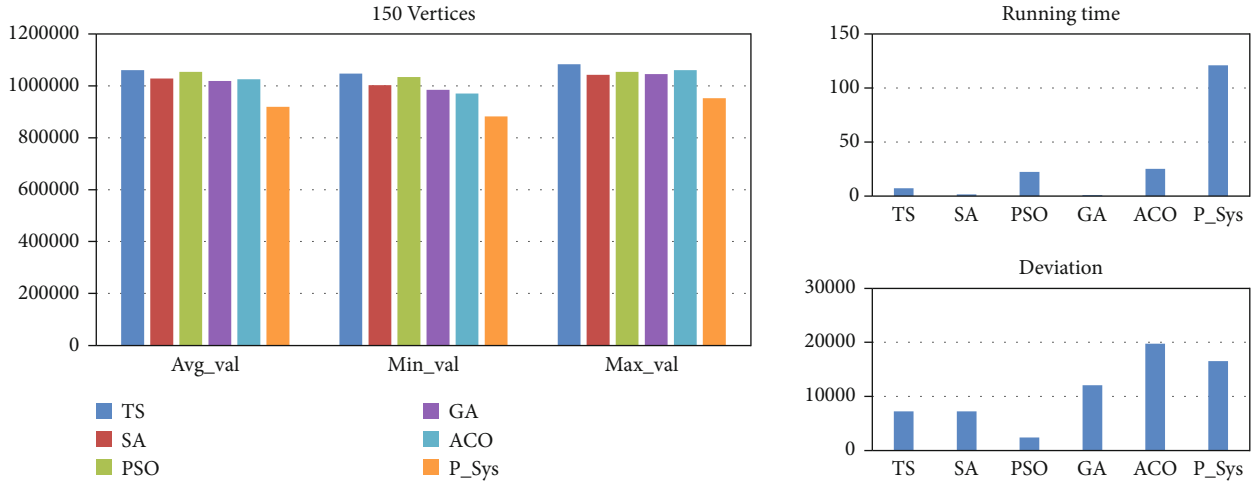


Figure 4: Experimental results of the map with 150 vertices.

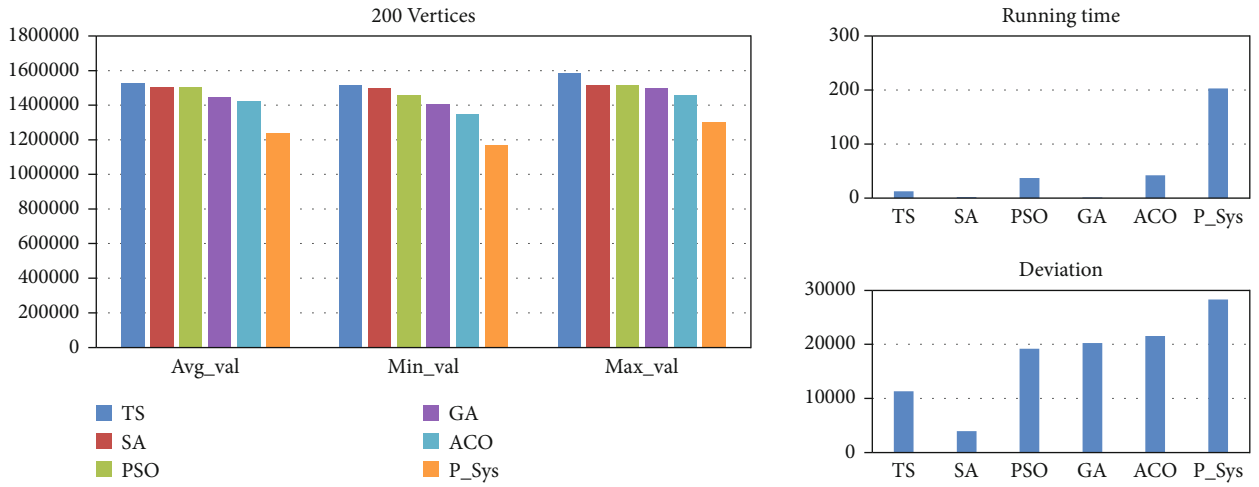


Figure 5: Experimental results of the map with 200 vertices.

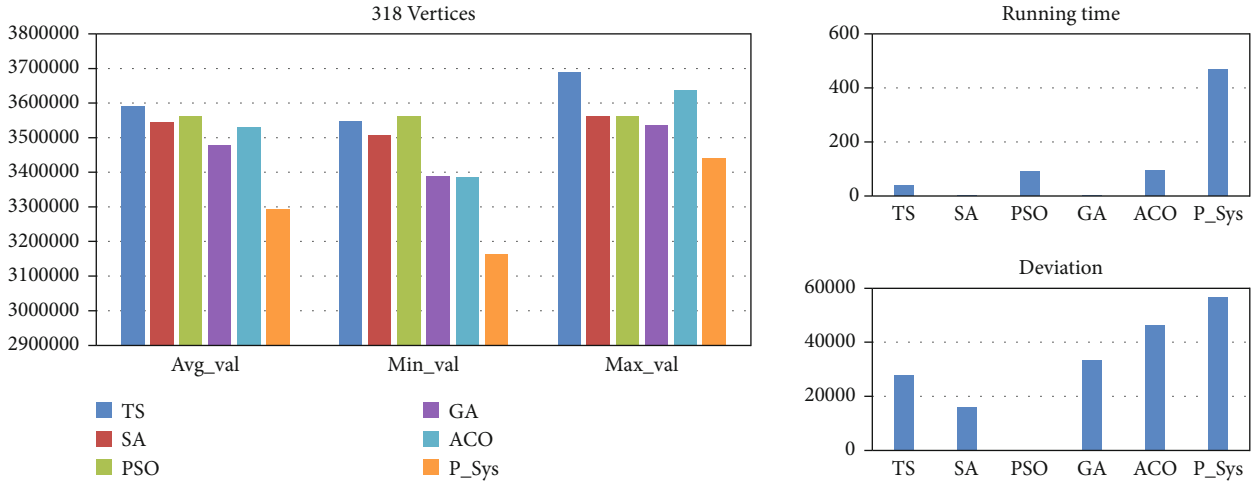


Figure 6: Experimental results of the map with 318 vertices.

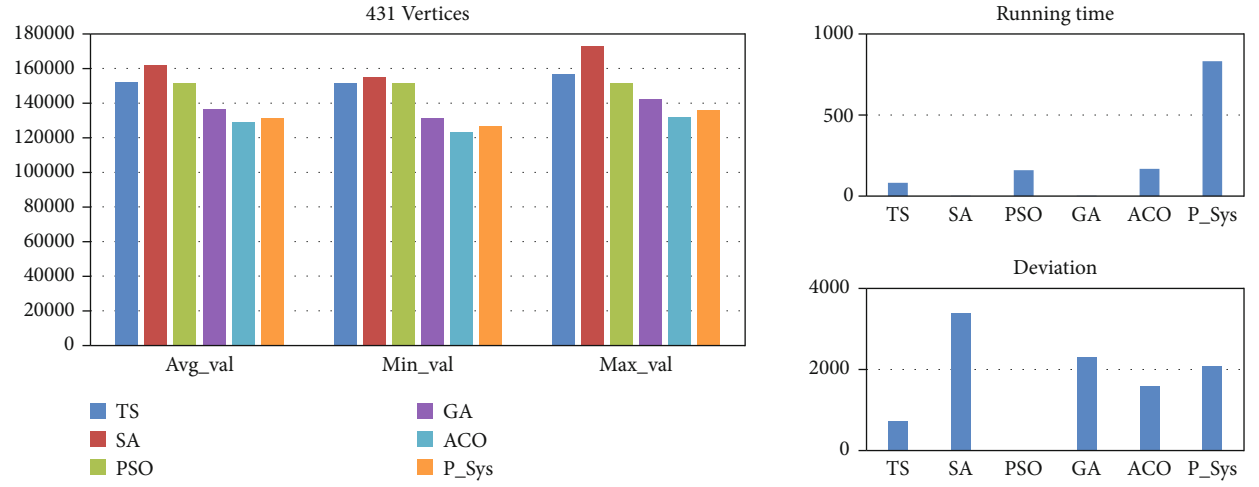


Figure 7: Experimental results of the map with 431 vertices.

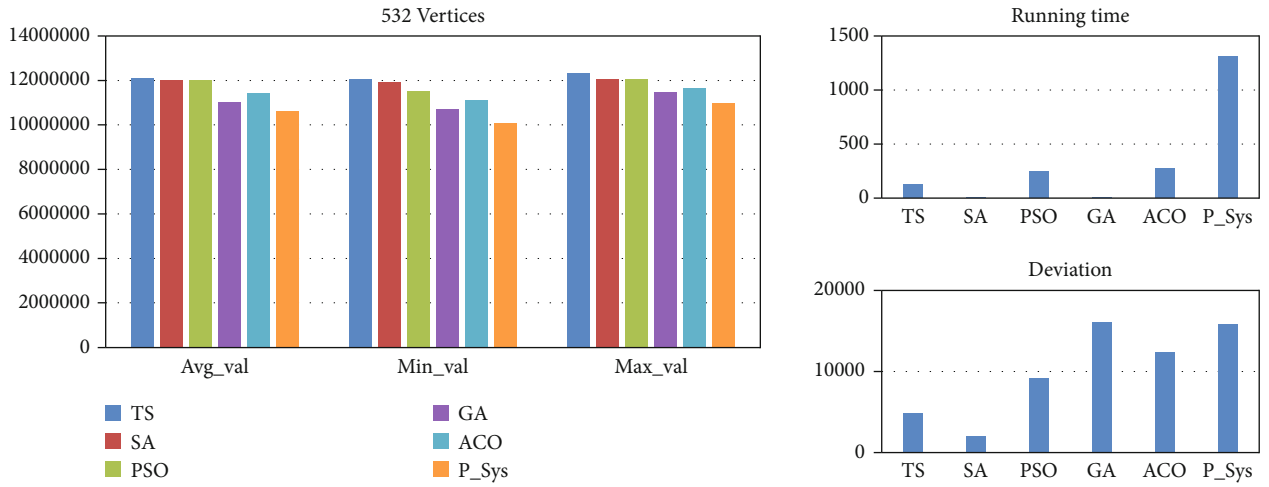


Figure 8: Experimental results of the map with 532 vertices.

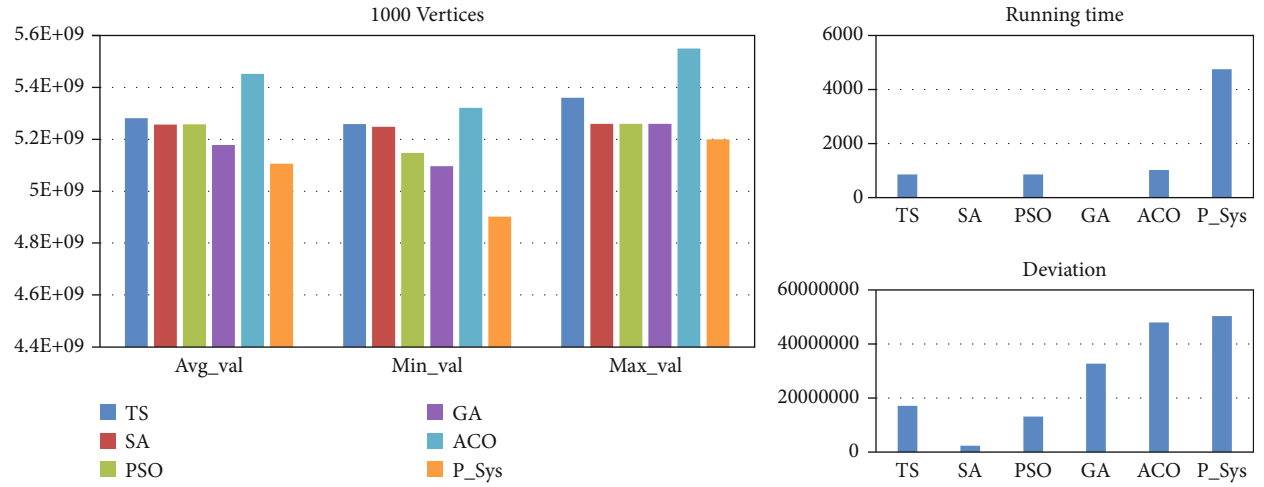


Figure 9: Experimental results of the map with 100 vertices.

Table 1: Average outputs.

Map	ACO	P_system	P_system improvement
eil51	$4.16E+03$	$3.83E+03$	7.93%
kroB100	$4.98E+05$	$4.52E+05$	9.24%
kroB150	$1.06E+06$	$9.19E+05$	13.3%
kroB200	$1.42E+06$	$1.24E+06$	12.68%
lin318	$3.53E+06$	$3.29E+06$	6.8%
gr431	$1.29E+05$	$1.31E+05$	-1.6%
att532	$1.14E+07$	$1.06E+07$	7.02%
dsj1000	$5.45E+09$	$5.11E+09$	6.24%

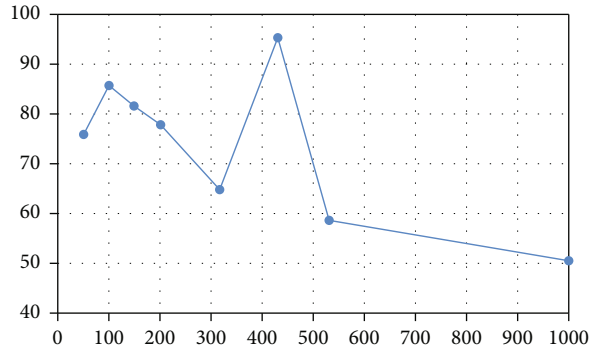


Figure 10: Iteration number at which ACO found the best result.

can be designed to run in parallel. The five inner membranes are mutually independent. None of them needs to use the others' outputs.

Based on the analysis above, we designed a parallel procedure. The main program creates five threads corresponding to the five inner-membrane heuristics. They are allocated to different CPU cores to run at the same time. When a thread completes, it sends its result to the main program and terminates. The main program waits until the five threads are all finished. Then, the main program combines the 5five threads' outputs as the outer

Table 2: Experimental results of ACO and the proposed P_system.

Map	ACO			P_system		
	Worst result	Best result	Variance	Worst result	Best result	Variance
eil51	$4.31E+03$	$3.90E+03$	$4.44E+03$	$3.99E+03$	$3.66E+03$	$4.41E+03$
kroB100	$5.13E+05$	$4.77E+05$	$6.39E+07$	$4.70E+05$	$4.32E+05$	$6.42E+07$
kroB150	$1.06E+06$	$9.70E+05$	$3.90E+08$	$9.52E+05$	$8.82E+05$	$2.72E+08$
kroB200	$1.46E+06$	$1.35E+06$	$4.63E+08$	$1.30E+06$	$1.17E+06$	$2.83E+04$
lin318	$3.64E+06$	$3.39E+06$	$2.15E+09$	$3.44E+06$	$3.16E+06$	$3.22E+09$
gr431	$1.32E+05$	$1.23E+05$	$2.51E+06$	$1.36E+05$	$1.26E+05$	$4.30E+06$
att532	$1.17E+07$	$1.11E+07$	$1.52E+10$	$1.10E+07$	$1.01E+07$	$2.49E+10$
dsj1000	$5.55E+09$	$5.32E+09$	$2.29E+15$	$5.20E+09$	$4.90E+09$	$2.52E+15$

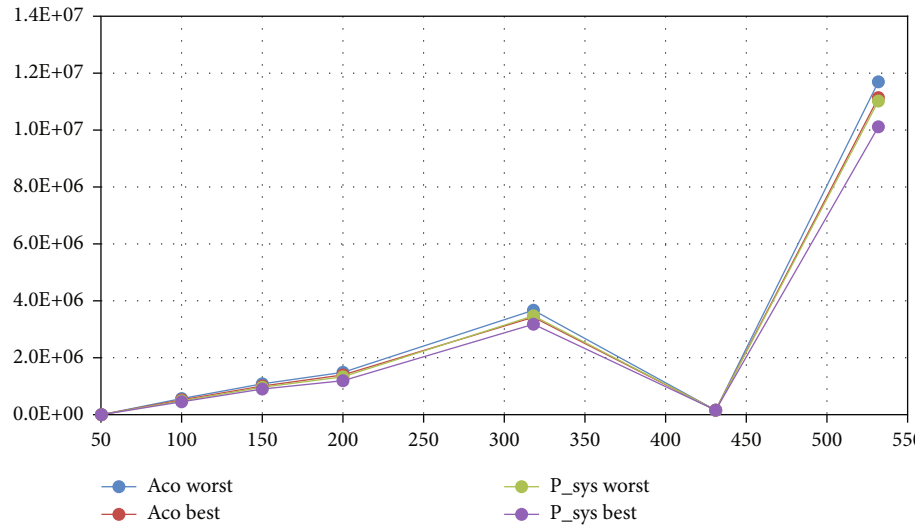


Figure 11: Best and worst results for both algorithms.

membrane's input. In other words, by running the sixth to the tenth line in the pseudo-code at the same time, some parts of the original algorithm can run in parallel. By using this strategy, the composite algorithm's running time drops impressively. The results of our experiments will be given in the fourth section.

3. Experimental Environment and Results

As in [5], the experiments were all run on a small-scale server. It is equipped with two Intel(R) Xeon(R) E5-2670 v3 @ 2.30 GHz CPUs (each has 8 cores) and 64 GB memory. The operating system is Windows® Server 2012. We used MATLAB® R2017b.

The same as the experiments in [5], we applied ACO (the outer membrane in the proposed algorithm), SA, TS, GA, and PSO as the competitive comparison for this paper. The algorithms' parameters are all the same as we set in [5]. Considering that the proposed heuristic is relatively powerful, we determined to test it in a wider range from 51 vertices to 1000 vertices. Rather than randomly generating the test map, we choose instances from TSPLib. They are *eil51*, *kroB100*, *kroB150*, *kroB200*, *lin318*, *gr431*, *att532*, and *dsj1000*, respectively. The vertices' weights are allocated randomly. We coded our P_system to run in parallel to save time. As in [5], all the problem instances are tested by the algorithms for 100 times. The outputs and running time for every map are all stored for later analysis.

We use ACO (which performed the best among the classic heuristics) and P_sys algorithms as representatives to illustrate the performance of the P_sys algorithm. The results of all the tested algorithms are shown from Figures 2–9. Table 1 shows the average outputs and running time for both algorithms to search the maps 100 times. For all but one instance, the output from the P_system was better than that from ACO by at least 6%. The performance of this proposed P_system is so good that we do not need to divide the problems into different scale groups. The P_system is able to find a better solution than ACO except for map gr431. For this map, we recorded the iteration number at which ACO found its best value and stopped improving. Then, for 100 runs on the full set of maps, we calculated the average iteration number at which the best value was found on each map. The values are shown in Figure 10.

The horizontal axis is the number of vertices in the map, and the vertical axis is the average iteration number at which the best value was found.

Figure 10 shows that the iteration number at which the best value was found decreases as the number of vertices increases—except in the case of map gr431. That is, as the scale of the problem increases, ACO tends to be trapped in a local optimum earlier. That may occur because as the problem scale increases, different local optima tend to get further away from each other. When ACO finds a local optimum, the algorithm is unable find a second local optimum that is far away. This leads us to hypothesize that the exceptional behaviour for map gr431 in Table 1 is because the map was relatively small.

To test this hypothesis, we captured the best and worst results for both ACO and our P_system (see Table 2 and Figure 11). These results show that the optimal value found for map gr431 is quite low, nearly matching the value for map eil51, which has 51 vertices. That is, map gr431 seems exceptional compared with the other maps in TSPlib, in terms of the results on this problem. For a map like this, we may not need an algorithm like the proposed P_system, which is able to search every possible direction of the search space. A simpler algorithm like ACO works better in this case. So in the case of map gr431, the proposed P_system did not have an advantage from using more heuristics. Instead, because of the existence of those additional heuristics, ACO did not perform the best in its task as an outer membrane.

4. Conclusion

In this paper, we propose a new way to model the urban WSN's maintenance job. A heuristic based on the structure of P_system is also designed to help solve MWLP. The proposed heuristic combines several well-known heuristics: TS, SA, GA, PSO, and ACO. Our experimental results show that this metaheuristic is more powerful than ACO in most cases, especially those with large numbers of vertices. Considering that our previous results given in [5] show that ACO gives the best results of all the individual heuristics used in our P_system, we can say that the proposed P_system has been demonstrated to be the most powerful approach overall as it yields better results. One exception has been observed in this paper where the proposed P_system performed slightly worse than ACO. Improving performance in this case is the priority in our future research. One possible solution may be finding a method to let the inner membranes cooperate with each other to empower the P_system to select better results.

Data Availability

The experimental results and the maps we used in the experiment will be found through the following link: https://drive.google.com/drive/folders/1XKDtCPHpJj2aAhiEYO3Wa4bhSIMFTJBU?usp=sharing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (No. 62102058), and part of the job was done when Ziqi Wei was pursuing for his PhD degree in the University of Alberta, which was supported by a grant from the China Scholarship Council.

References

[1] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis, "Searching a fixed graph," *In International Colloquium on Automata, Languages, and Programming, pages*, vol. 1099, 1996.

[2] B. Y. Wu, "Polynomial time algorithms for some minimum latency problems," *Information Processing Letters*, vol. 75, no. 5, pp. 225–229, 2000.

[3] R. Sitters, "The minimum latency problem is NP-hard for weighted trees," *Lecture Notes in Computer Science*, vol. 2337, pp. 230–239, 2002.

[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "On the minimum latency problem," in *Proc 26thSymp Theory of Computing STOC*, p. 9, 1994.

[5] Z. Wei and M. H. MacGregor, "Heuristic approaches for minimum weighted latency problem," in *2016 4th International Conference on Machinery, Materials and Information Technology Applications*, pp. 552–556, Atlantis Press, 2017.

[6] Z. H. Ahmed, "The minimum latency problem: a hybrid genetic algorithm," *IJCSNS International Journal of Computer Science and Network Security*, vol. 18, no. 11, pp. 153–158, 2018.

[7] H. B. Ban and N. D. Nghia, "Improved genetic algorithm for minimum latency problem," in *Proceedings of the 2010 Symposium on Information and Communication Technology*, pp. 9–15, 2010.

[8] T. Dewilde, D. Cattrysse, S. Coene, F. C. R. Spieksma, and P. Vansteenwegen, "Heuristics for the traveling repairman problem with profits," *Computers and Operations Research*, vol. 40, no. 7, pp. 1700–1707, 2013.

[9] M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi, "A simple and effective metaheuristic for the minimum latency problem," *European Journal of Operational Research*, vol. 221, no. 3, pp. 513–520, 2012.

[10] B. B. Ha and N. N. Duc, "A meta-heuristic algorithm combining between tabu and variable neighborhood search for the minimum latency problem," *Fundamenta Informaticae*, vol. 156, no. 1, pp. 21–41, 2017.

[11] K. Lenin, B. R. Reddy, and M. Suryakalavathi, "Hybrid tabu search-simulated annealing method to solve optimal reactive power problem," *International Journal of Electrical Power and Energy Systems*, vol. 82, pp. 87–91, 2016.

[12] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[13] Y. Niu, S. Wang, J. He, and J. Xiao, "A novel membrane algorithm for capacitated vehicle routing problem," *Soft Computing*, vol. 19, no. 2, pp. 471–482, 2015.

[14] A. Yan, H. Shao, and Z. Guo, "Weight optimization for case-based reasoning using membrane computing," *Information Sciences*, vol. 287, pp. 109–120, 2014.

[15] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1514437, 14 pages
https://doi.org/10.1155/2022/1514437

Research Article

An Incentive Mechanism for Computation Offloading in Satellite-Terrestrial Internet of Vehicles

Xingyu Zhang,[1] **Heyang Zhang**,[1] **Sida Dai**,[2] **and Yang Liu**[2]

[1]*International School, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Yang Liu; liu.yang@bupt.edu.cn

Received 30 September 2021; Accepted 7 December 2021; Published 19 January 2022

Academic Editor: Chi Lin

With the development of satellite-terrestrial technology and the popularity of the Internet of Vehicles, how to improve the efficiency of mobile cloud computing (MCC) has become the next concern. However, the resource of cloudlets is not sufficient to perform large-scale computation tasks, or some applications designed to run on vehicles have more efficiency executed on vehicles than executed on cloudlets. Additionally, it is still challenging for platforms to motivate mobile vehicle owners to join in the process while the existing mechanisms cannot provide all the desired properties in cloudlet scenarios. To this end, we design a satellite-terrestrial IoV based on an incentive mechanism for computation offloading (IMCO) in mobile edge computing to motivate vehicle owners to perform computation offloading tasks so as to offload certain kinds of tasks to the mobile vehicles. By optimizing the MCC model, we integrate auction theory into the mechanism to ensure individual rationality, budget balance, system efficiency, and truthfulness for both sellers and buyers. Through rigorous theoretical analysis, we prove that our mechanism can achieve computational efficiency under the condition that all algorithm outputs be computed in polynomial time. Both theoretical derivations and numerical calculations prove that all the desired properties of the mechanism hold.

1. Introduction

Along with advances in computing and communications technology, the Internet of Vehicles (IoV), which is gradually replacing conventional Vehicle Ad-hoc Networks, is becoming popular worldwide and creating a promising vision for transport systems. According to a recent report by Allied Market Research, the global IoV market is expected to exceed 200 billion by 2024. IoV is a distributed network [1] of huge commercial and research value that supports the use of data created by connected cars and vehicular ad-hoc networks (VANETs). Today, the design, implementation, and management of IoV technology rely heavily on modern cloud computing technologies, which play a central role in linking isolated vehicles into an organic network in which data are being processed primarily. However, considering limitations such as the size of the data and the stability of the channel, cloud computing struggles to process efficiently and could cause fatal latency [2]. According to a McKinsey & Company estimate, connected cars create up to 25 GB of data per hour. With such huge amounts of data, we need to consider new computing technologies [3]. Mobile edge computing is an emerging paradigm in IoT systems [4] that offloads data processing to the mobile which localizes data processing and enabling services to be functional [5] even if connectivity between the vehicle and cloud is absent. In addition, some applications which are suitable for running on vehicles have more efficiency executed on powerful mobile vehicles than executed on cloudlet.

However, implementing mobile edge computing for computation offloading still faces several challenges [6]. First, requisitioning a viable and suitable mobile vehicle for computing takes up a user's personal resources, and it is difficult to convince users to volunteer their mobile vehicles without rewards. Therefore, we need to design a flexible and reasonable incentive mechanism to encourage users to

willingly provide their vehicles as computation resources [7]. Second, the complex physical environment in which cars travel [8], such as unstable signals and frequent congestion or crashes, places high demands on the stability and capacity of communications. Therefore, we design to adopt a satellite-terrestrial network [9] on IoV [8] to overcome the implementation difficulties. With seamless accessibility to overcome the coverage and distribution limitations of RSUs and BSs, it could provide macroscopic management to improve overall network efficiency and resource utilization at a low cost. Third, resource allocations [10] and computation offloading transactions require low time delay or there is no necessity to conduct it. Therefore, we design costumed algorithms during different stages which guarantee system and computation efficiency [11]. Lastly, the battery capacity is still a limitation of supporting energy consumption [12]. Luckily, with the popularity of fast charging technology [13], the charging efficiency of vehicles is greatly improved [14], since the development of low delay wireless charging technology [15] and the convenience of using wireless charging [16]. By using wireless charging technology [17], users can charge their mobile vehicles anytime and anywhere [18]. Therefore, with the development of fast charging technology and wireless charging technology, the problem of energy constraints can be relieved [19].

In order to overcome the above challenges, we propose an incentive mechanism for computation offloading (IMCO) that maximizes the motivation of vehicle owners to perform computation offloading tasks while guaranteeing the system stability and the privacy of participants. More specifically, we first investigate the auction theory and integrate it into Mobile Cloud Computing (MCC) scenarios for computing offloads, where the problem of resource allocation was redesigned according to demand. Then, we propose the sealed-bid auction model which achieves consistent asks of mobile vehicles and variational offer of cloudlets as well as many desirable properties such as individual rationality, budget balance, truthfulness, and computational efficiency. To further illustrate the mechanism of IMCO, we design three algorithms for different phases within an auction and further use an example to demonstrate the process in detail. By rigorous theoretical analysis, our proposed mechanism is proved to achieve the four proposed desirable properties and close-to-optimal performance. Finally, we carry out extensive simulations of our proposed mechanism by testing the performance on a dataset generated by distribution functions by analyzing six indexes of different aspects.

The main contributions of this paper are highlighted below.

(1) We propose an incentive mechanism for computation offloading (IMCO) based on satellite-terrestrial IoV. The feasibility and the advantages of our design are also discussed

(2) IMCO solves the problem that remote cloud server is not that sufficient to perform large scale computation tasks which will greatly encourage users to join in with computation offloading even further helps to shape a good user environment and provides the basis for widespread commercialization

(3) We apply the auction theory in economics to the scenarios in which computation tasks are offloaded to the vehicle owners. Since the existing auction algorithms cannot achieve all the properties when they are applied to computation offloading in MCC with cloudlets and vehicles as homogeneous items

(4) We theoretically prove that our proposed mechanism is more suitable for real-life scenarios as it achieves truthfulness. Extensive simulations verify that all previous proposed desirable properties are satisfied

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes our system model and formulates problems. Section 4 designs the algorithm and the mechanism as well as carries out a theoretical analysis. Section 5 presents simulation and performance results of our proposed mechanism. Section 6 concludes the paper.

2. Related Work

In this part, the related work is divided into two groups: incentive mechanisms especially for mobile cloud computing and auction mechanisms used in economics.

Mobile edge computing is an emerging paradigm in the current IoT system. Based on the concepts of cloud computing and mobile computing, MCC was built relying on wireless networks and focusing on the construction and interaction of physical devices and virtual resources. So, there are a number of studies and researches about different applications in MCC, such as online translation service with MCC and big data storage strategy for MCC. Nevertheless, there is little research about the incentive mechanism. In the scenario, the auction must deal with items with complementary and replacement relationships.

As what we illustrated in Section 1, even if the auction theory is well developed and applied, the existing auction algorithms are not suitable to apply in computation offloading from cloudlets to mobile vehicles. Because of incompletely satisfying the desirable properties, Vickrey auction [20] has the properties of individually rational, budget-balanced. However, it cannot deal with cheating behavior; therefore, it is susceptible to market manipulation. McAfee in [21] focuses on the auction of nonidentical items. In other words, buyers do not lean to auction commodities. When the auction begins, every buyer makes their offer only once and each seller asks for a price. After that, the auctioneer ranks the offers in a nonascending way and asks in a nondescending order to find out the minimum possible offer and maximal possible ask; then, determining the winners in buyers according to their bids. That is, their bids are larger than or equal to the maximum possible ask. And also determining the winning sellers relies on the same rule. Next, the auctioneer generates two prices: clearing price and clearing

payment. Every buyer must submit the clearing price to the auctioneer. In the meantime, each seller gets rewards that are equal to clearing payment. Notwithstanding, McAfee cannot be applicable to the computation offloading scenario, since the mechanism aims at auction of homogeneous items. Even if it achieves all the desirable properties, in the scenario, each mobile vehicle will be rated differently from cloudlets rely on different factors, such as latency [6], quality of service, communication overhead, and computational capabilities.

Considering the homogeneous commodities, TASC [22] includes two phases: winner determination phase and pricing phase. In the first stage, the auctioneer filtrates inapposite sellers and buyers to determine the first-stage winners. In the second stage, the auctioneer implements the McAfee mechanism to determine the clearing price and payment. Although the scenario of TASC is close to mobile cloud computing with cloudlets, there are still some defects by applying TASC in the cloudlet scenario. Because of the distinct feature of cloudlets, TASC cannot support truthfulness for buyers even if it can guarantee truthfulness for sellers and other properties.

Our previous work proposes a computation offloading incentive mechanism [23]. Compared to [23], this paper discusses how to unload some types of applications to mobile vehicles by encouraging vehicle owners to perform computing unload tasks. By optimizing the MCC model, we integrate auction theory into the mechanism to ensure individual rationality, budget balance, system efficiency, and truthfulness for both sellers and buyers. IMCO maximizes the motivation of vehicle owners to perform computation offloading tasks. In addition, IMCO can assign more than one mobile vehicle for a single cloudlet in a temporary state.

3. System Model and Problem Formulation

In our system architecture shown in Figure 1, there are three main parts: the satellite-terrestrial networks (STNs), the vehicles, and the cloudlets. STNs are the integration of satellite networks and terrestrial networks [24], consisting of one or more satellite systems and land-based stations, which efficiently disseminate information through wireless channels [25]. In the architecture, computational tasks on cloudlets can be offloaded to vehicles within coverage [26], while vehicles are simultaneously free to choose to perform computational tasks from all cloudlets covered by their current locations via STNs.

There are lots of vehicles around one cloudlet, the performance of some vehicles may be too weak to carry out computation-intensive tasks. To lift efficiency, the cloudlet can use auction mechanisms to offload computation tasks to other high-performance vehicles so as to reduce the communication overhead and improve efficiency [27]. Each vehicle will be rated differently from cloudlets based on different factors, such as latency, quality of service, communication overhead, and computational capabilities [28]. On the other side, vehicle owners can be rewarded for providing computation resources and compensated for communication costs [29]. As in most real cases in the auction market, both the buyers and the sellers can get benefits from the transactions, that is, car owners can get paid to be incentivized to share its resource [30]. When the computation offloading is applied in the mobile edge computing, every cloudlet bids when it needs extra computational performance, and it submits the offers to the auctioneer [31]. Then, the asks of vehicles are also submitted. The auctioneer, also known as a control center, will allocate m vehicles among n cloudlets according to the outcome of IMCO, consequently reducing the communication overhead and the latency so that the services will be affordable.

3.1. Resource Allocation for Mobile Vehicles. As the structure shown in Figure 1, there are lots of mobile vehicles around one cloudlet, and some performance of vehicles may be too weak to carry out some computation-incentive applications. The cloudlet can provide service to running applications for those vehicles. However, some tasks rely on particular hardware such as DSP, ISP, and NPU. The virtual machines may solve the problem by hardware virtualization. But the efficiency of virtualization is very low due to the structure of vehicles. The integration level of vehicles is much higher than cloudlet, so there are some applications using heterogeneous computing combining the advantages of CPU, GPU, NPU, ISP, and DSP. And lots of hardware are designed to process specific tasks. Hence, using virtualization is not a suitable solution. One cloudlet may serve several low-end mobile vehicles at one time; to lift efficiency, cloudlet can use auction mechanism to offload computation tasks to other high-performance mobile vehicles. The close range of high-performance mobile vehicles can be exploited to reduce the communication overhead and improve efficiency. Each mobile vehicle will be rated differently from cloudlets rely on different factors, such as latency, quality of service, communication overhead, and computational capabilities. Some cloudlet may need more powerful computation capabilities to achieve computation-intensive tasks, and others may prefer a low communication latency to respond to their clients more quickly.

On the other side, vehicle owners can be rewarded for providing computation resource and compensated for communication cost. As most real cases in the auction market, both the buyers and sellers can get benefits from the trading, that is, vehicle owners can get paid such as bitcoins to be incentivized to share its resource, and the tasks which are offloaded to the mobile vehicles should be done in given times. Especially, cloudlet must pay no less than the cost of the vehicle owner providing such computation resource.

In an area, the more computation offloading tasks, the higher the resource utilization of the mobile vehicles. In order to maximize the use of resources, the auctioneer should accurately allocate the mapping mobile vehicles' resources to the assigned tasks from cloudlets. Mobile vehicles only care about the cost of renting their resources. And the cloudlets' offers are adjustable. The performance of mobile vehicles varies from field to field and according to the auction model will be calculated to determine the right winning bidder.

3.2. Auction Model. The computation offloading is applied in the MCC scenario which is illustrated in Figure 1. Every

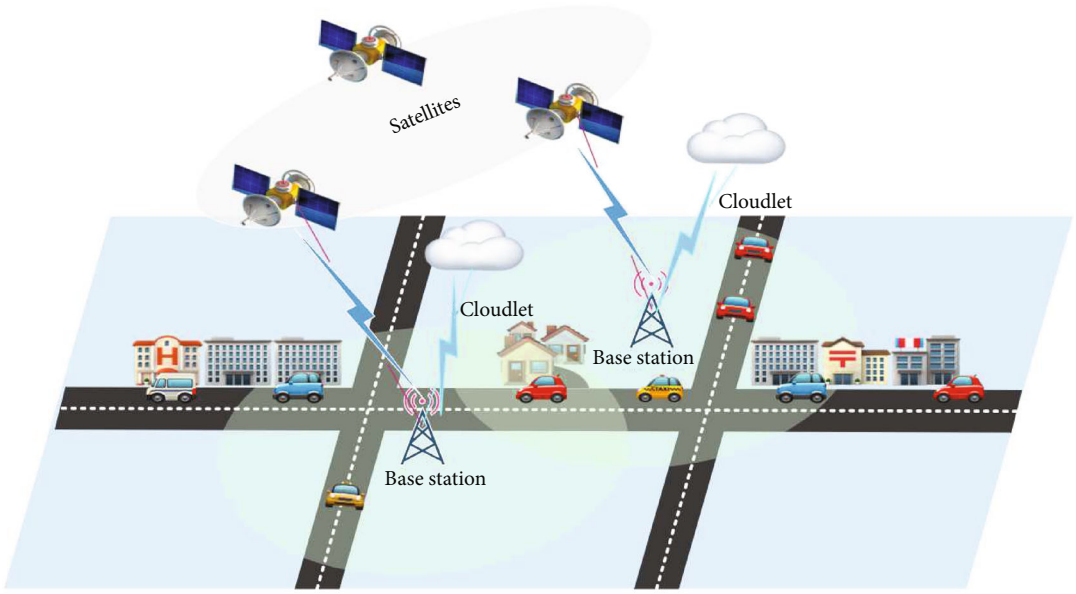


Figure 1: The system architecture of satellite-terrestrial IoV.

cloudlet bids when they need extra computational performance, and they submit their offers to the auctioneer. Then, the asks of mobile vehicles are also submitted. The auctioneer also known as the control center will allocate m mobile vehicles among n cloudlet according to IMCO, consequently reducing the communication overhead and the latency so that the services will be affordable.

To achieve truthfulness, the sealed-bid auction model is necessary. That is to say, each cloudlet (mobile vehicle) can upload its bid (ask) secretly to the auctioneer so that everyone's information including buyers' bids and sellers' asks is invisible to others. As Table 1 depicted:

(i) For each cloudlet $c_i \in C$, $C = \{c_1, c_2, \cdots, c_n\}$. $B_i = \{B_i^1, B_i^2, \cdots, B_i^m\}$ is defined as bid vector, where B_i^j is the bid for mobile vehicle $m_j \in M$, $M = \{m_1, m_2, \cdots, m_m\}$. The matrix of bids including all the bid vectors of all cloudlets is denoted by $B = \{B_1; B_2; \cdots; B_m\}$.

(ii) For each mobile vehicle $m_j \in M$, its ask vector is defined as $A = \{A_1, A_2, \cdots, A_m\}$, where A_j is the ask of mobile vehicles $m_j \in M$

As you can see, the asks of mobile vehicles are consistent no matter which cloudlets are since the mobile vehicles only care about how much payment for renting their resources.

On the contrary, the offer of cloudlets is variational respecting different mobile vehicles due to the weights of different tasks are distinct, and the performance of different mobile vehicles of different fields is different.

The auctioneer will calculate to determine the winning mobile vehicles $C_c \in C$ and winning cloudlets $M_w \in M$, also define a mapping σ between C_c and M_w, the price P_i^c is charged from winning cloudlets $M_w \in M \in M$, and the reward P_j^m to $C_c \in C$. In particular, P_{ij}^c is denoted as the price charged to cloudlet c_i from mobile vehicle m_j, and P_{ij}^m is defined as payment rewarded to mobile vehicle m_j from cloudlet c_i.

We are also concerned about the utilities of cloudlets and mobile vehicles in the mechanism. There are two factors that affect the utilities. The first one is the evaluation of cloudlets toward the services provided by mobile vehicles, and the next one is the cost of providing such computation resources. So, we defined V_i^j as the valuation toward mobile vehicle m_j rated by c_i. $V_i = (V_i^1, V_i^2, \cdots, V_i^m)$. Hence, we can have the utility of c_i and m_j as below.

$$U_c^i = \begin{cases} V_i^j - P_i^c, & \text{if } c_i \in C_w, \\ 0. & \text{otherwise,} \end{cases} \tag{1}$$

$$U_m^j = \begin{cases} P_j^m - \text{Cost}_j, & \text{if } m_j \in M_w, \\ 0. & \text{otherwise.} \end{cases} \tag{2}$$

As what (1) and (2) expressed, the utility $U_c^i > 0$ only when cloudlet c_i is allocated to a mobile vehicle m_j. The higher U_c^i is, the more satisfying the cloudlet will be. On the other side, the utility U_m^j illustrates the profit the mobile vehicles received, since it is equals to the remainder of the received reward over its cost.

3.3. Desired Properties. As what we have depicted, the auction model has several inputs: C, M, B, and A. Correspondingly, the auctioneer adopts an auction mechanism to calculate a series of results: C_w, M_w, and σ. A well-designed auction mechanism must have the properties as follows.

(i) Individual rationality:

Table 1: Notions.

Symbol	Definition
c_i	Cloudlet
c_{ij}	Cloudlets with positive evaluation toward mobile vehicles
n	Total number of cloudlets
m	Total number of mobile vehicles
m_j	Mobile vehicle
C	Set of cloudlets
C^+	Cloudlets with positive evaluation
M	Set of mobile vehicles
C_c	Set of winning cloudlets candidates
M_c	Set of winning mobile vehicles candidates
C_{bf}	Set of winning cloudlets before filtering
M_{bf}	Set of winning mobile vehicles before filtering
C_w	Set of winning cloudlets
M_w	Set of winning mobile vehicles
$\mathbb{C}$	Nonincreasing order of C^+
$\mathbb{M}$	Nondecreasing order of mobile vehicles
$\sigma'()$	A function that maps from M_{bf} to C_{bf}
$\sigma()$	A function that maps from M_w to C_w
B_i^j	Cloudlet c_i biding on mobile vehicle m_j
B_i	Cloudlet c_i's bid vector
B	Bid matrix of C
A_j	Ask of mobile vehicles m_j
A	Ask vector of M
A_{-j}	Ask vector of all sellers not including m_j
V_i^j	Valuation of m_j rated by c_i
V_i	Valuation vector of c_i
Cost_j	Cost of m_j performing computation offloading tasks
P_i^c	Price that is charged to c_i
P_j^m	Reward price paid to m_j
P_{ij}^c	Price that is charged to c_i for m_j
P_{ij}^m	Reward price paid to m_j with allocated c_i
U_i^c	Utility of c_i
U_j^m	Utility of m_j
U_{ij}^c	Utility of c_i with allocated m_j
U_{ij}^m	Utility of m_j with allocated c_i
P_w^c	The set of clearing price that is charged to C_w
P_w^m	The set of clearing rewards to M_w

$$P_i^c \le B_i^j, \forall c_i \in C_w, \tag{3}$$

$$P_j^m \ge A_j, \forall c_i \in C_w, \forall m_j \in M_w. \tag{4}$$

The formula shown in (3) means that the winning cloudlet is charged less than or equal to its offer. The formula in (4) shows that the winning vehicle is paid more or equal to its ask.

(ii) Budget balance:

$$\sum_{c_i \in C_w} P_i^c \ge \sum_{m_j \in M_w} P_j^m. \tag{5}$$

The formula above shows that the total rewards that the auctioneer gives to all winning vehicles are more or equal to the sum of prices that all winning buyers submit to the auctioneer.

(iii) *Truthfulness*. An auction algorithm is said to be truthful if and only if

$$U_i^c(c_{i'}, B_{-i}) \ge U_i^c(c_i, B_{-i}), \tag{6}$$

$$U_j^m\left(m_{j'}, A_{-j}\right) \ge U_j^m\left(m_{j'}, A_{-j}\right). \tag{7}$$

B_{-i} means that the bid vector without the cloudlet c_i and A_{-j} means that the ask vector of all sellers not including m_j. Such that a cloudlet cannot improve its utility by faking a bid different from the true valuation of a vehicle and no vehicle can fake the ask different from its cost. U_i^c achieves its maximum only if

$$B_i = V_i, \forall c_i \in C. \tag{8}$$

The U_j^m achieves its maximum only if

$$A_j = C_j, \forall m_j \in M. \tag{9}$$

(iv) *Computational Efficiency*. The outputs of the auction mechanism, i.e., C_w, M_w, σ, P_w^c, and P_w^m should be calculated in a polynomial time concerning n and m

As what we have depicted, the auction model has those input: C, M, B, and A. Correspondingly, the auctioneer adopts auction algorithms to calculate a series of result: C_w, M_w, σ, and (referring to Table 1). A well-designed auction mechanism must have the properties as follows.

3.4. Technical Challenges. As the what we argue in Section 2, the existing auction algorithms cannot achieve all the properties when they are applied to computation offloading in MCC with cloudlets and mobile vehicles as homogeneous items. The example shown below illustrates that TASC auction mechanism in [22] cannot support truthfulness for buyers despite the support of the other desirable properties.

Table 2 shows a matrix of bid in which the value of different mobile vehicles rated by different cloudlets are

Table 2: Bid matrix.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
c_1	0	7	0	0	10	5	0
c_2	0	1	2	2	0	0	8
c_3	0	0	0	6	9	0	0
c_4	10	0	0	0	0	0	6
c_5	6	0	9	8	0	0	0

different. And Table 3 shows the ask vector of mobile vehicles which has the same value of the cost of providing computation resources. To illustrate the lacking truthfulness of TASC applied in this scenario, supposing that the auctioneer uses an algorithm to maximize the overall QoS. The TASC mechanism has two stages: the first step is to determine the winner and the next one is to determine the price. The mapping of winning cloudlet candidates and winning mobile vehicle candidates is shown in Figure 2.

In the second stage, we get the result of final winning cloudlet: c_1 and c_4, the winning mobile vehicles: m_1 and m_5, the $P_w^c = 8$ and $P_w^m = 6$. It is clear to see that the utility of c_3 is 0 according to equation (1).

If cloudlet c_3 cheats the auctioneer by submitting bid dishonestly. That is, c_3 increases its offer B_3^5 by adding a variable Δ which is larger than 1, since $9 + \Delta > 10$, the new result of winner mapping is shown in Figure 3. The winning cloudlets are changed to c_4 and c_3 which is still the same value as in Figure 2. The utility of c_3 is changed to $9-8 = 1$. Hence, c_3 can improve its utility by faking its bid. It means that TASC auction mechanism is not suitable to the scenario that we are interested in. In Section 4, we propose a truthful double auction mechanism called IMCO, which can achieve all the desirable properties illustrated in Section 1.

4. Mechanism Design

As we illustrated in the background, truthful Vickrey auction [20] cannot guarantee truthfulness and other properties at the same time. However, the McAfee auction mechanism cannot be directly used in MCC with cloudlets and mobile vehicles as heterogeneous items. TASC auction mechanism is extending the McAfee mechanism in [22]. Unlike the McAfee auction, TASC is a heterogeneous auction mechanism. While TASC cannot achieve truthfulness to the scene as the example we present in the technical challenges. The result shows that TASC cannot counter market manipulation of untruthful cloudlets in stage 1.

To solve this problem, we propose IMCO. In IMCO, we change the order of the general stage. The auctioneer firstly selects the winning candidates both the cloudlets and the mobile vehicles referring to some rules. And then, each mobile vehicle that is the winning candidate will be allocated to one winning cloudlet candidate. And the P_w^c and P_w^m will be determined as well. In addition, IMCO can assign more than one mobile vehicle for a single cloudlet in a temporary state; in the last stage, the winner redundant mobile vehicles will be eliminated to ensure the efficiency of the mechanism.

Table 3: Ask vector of mobile vehicles.

Mobile vehicle	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Ask	2	3	9	5	1	4	7

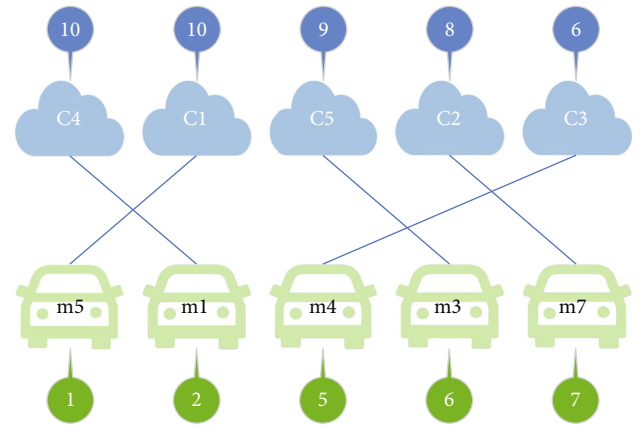


Figure 2: The bidirectional mapping graph of winning mobile vehicles and cloudlets.

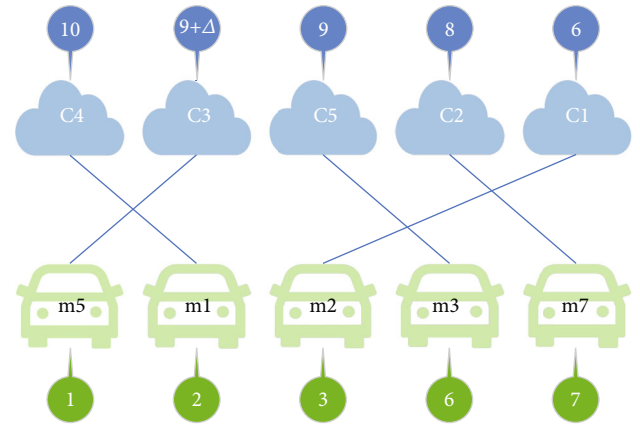


Figure 3: Different results of bidirectional mapping graph of winning mobile vehicles and cloudlets.

In the next part of this paper, we illustrate the mechanism of IMCO minutely. Then, we give an example to present the IMCO more clearly. The properties of IMCO will be discussed in the following chapter.

4.1. Details of IMCO. The IMCO consists of three phases: winner candidates determination phase, assignment and pricing phase, and winner elimination phase to achieve the preceding desirable properties.

To be specific, Algorithm 1 is used to filter some unqualified cloudlets and mobile vehicles so as to select winner candidates. To begin with, Algorithm 1 creates C^+ from the original cloudlet set C which consists of bidders whose bid is greater than zero in accordance to B. That is to say, a cloudlet can appear several times in regard to the mobile vehicles for which the cloudlet's offer is larger than zero. Next, the C^+ is sorted to C in which the bids are sorted in a nonascending sequence. Also, mobile vehicles are sorted

Input: C,M,B,A;
Output: C_c,M_c,$B_{p\alpha}^{q\alpha}$,$A_{j\beta}$;
1: $C_c \longleftarrow \varnothing$, $M_c \longleftarrow \varnothing$;
2: Create C^+ which is consist of bidders whose bid is greater than zero in accordance to B, i.e., $C^+ = \{c_p^q: B_p^q > 0, c_p \in C\}$;
3: Sort all cloudlets in C^+ to a non-ascending sequence $\mathbb{C} = (c_{p1}^{q1}, c_{p2}^{q2}, \cdots, c_{p1\gamma}^{q\gamma})$ in which $B_{p1}^{q1} \geq B_{p2}^{q2} \geq \cdots \geq B_{p\gamma}^{q\gamma}$;
4: Sort all mobile vehicles in M into a non-descending order $M = (m_{j1}, m_{j1}, \cdots, m_{jm})$ such that $A_{j1} \leq A_{j2} \leq \cdots \leq A_{jm}$;
5: Find the median ask $A_{j\beta}$ of M, where $\beta = \lceil 1 + m/2 \rceil$;
6: Find the minimized α, so that $B_{p\alpha+1}^{q\alpha+1} < A_{j\beta}$ and $B_{p\alpha}^{q\alpha} \geq A_{j\beta}$;
7: $C_c \longleftarrow$ the first α cloudlets in $\mathbb{C}$;
8: **For** $c_p^q \in C_c$ **do**
9: **If** $A_q > A_{j\beta}$ **then**
10: $C_c \longleftarrow C_c \setminus \{c_p^q\}$;
11: **Else**
12: **If** $m_q \notin M_c$
13: $M_c \longleftarrow M_c \cup \{m_q\}$;
14: **End if**
15: **End if**
16: **End for**
17: **Return** $(C_c, M_c, B_{p\alpha}^{q\alpha}, A_{j\beta})$;

Algorithm 1: Winner candidates determination phase.

in a nondescending list according to ask A, denoted by M. Particularly, the $A_{j\beta}$ is the median ask of mobile vehicles in M, where $\beta = \lceil 1 + m/2 \rceil$, is to find the minimize value α, so that $B_{p\alpha+1}^{q\alpha+1} < A_{j\beta}$ and $B_{p\alpha}^{q\alpha} \geq A_{j\beta}$. The $A_{j\beta}$ and $B_{p\alpha}^{q\alpha}$ are two standards to identify the winning candidates. For example, if c_p^q is a winning cloudlet candidate in C_c, it should satisfy $A_q < A_{j\beta}$ and $B_{p\alpha}^{q\alpha} \geq A_{j\beta}$. The mobile vehicle m_α is said to be winning candidate if and only if and at least one cloudlet from $\mathbb{C}$ has a bid for m_α. The β can be other value no necessary the median value. The value of β directly affect the number of M_c so to affect the number of C_c.

In the next stage, to protect the system from biding and asking untruthfully, we closely combine assigning and pricing. The procedure is shown above in Algorithm 2. First, the control center assigns the winning cloudlet candidates for winning mobile vehicle candidates. There are three situations needed to be discussed. The first situation is that there is only one cloudlet biding for one mobile vehicle, so the cloudlet is added to the C_{bf} waiting for processing Algorithm 2 and charged $B_{p\alpha}^{q\alpha}$. The second situation is that there are two or above cloudlets scrambling for one mobile vehicle, the auctioneer would choose the cloudlet with the highest bid from the cloudlet candidates and add it to C_{bf}. The last situation is when there are two or more cloudlets that offer the highest bids, the control center will randomly choose one cloudlet among the cloudlets with the highest bid and add to $B_{p\alpha}^{q\alpha}$. Different from the first situation, the cloudlet is charged for the 2nd highest bid. To make it easier to understand, supposing that the clearing price $B_{p\alpha}^{q\alpha}$ is 6 and two highest bids are $B_a^j = B_b^j = 12$, and the next bid in nonascending order is $B_c^j = 10$, then, c_{aj} and c_{bj} have a 50-50 chance of becoming member of C_{bf}. And the winner will be charged 10 to avoid untruthfully biding since the 2nd highest bid is 10.

Input: C_c, M_c, $B_{p\alpha}^{q\alpha}$, $A_{j\beta}$, B;
Output: C_{bf}, M_{bf}, σ', P_{bf}^m, P_{bf}^c;
1: $C_{bf} \longleftarrow \varnothing$, $M_{bf} \longleftarrow M_c$, $P_{bf}^m \longleftarrow \varnothing$, $P_{bf}^c \longleftarrow \varnothing$;
2: **For** $m_j \in M_{bf}$ **do**
3: $P_j^m = A_{j\beta}$, $P_{bf}^m \longleftarrow P_{bf}^m \cup \{P_j^m\}$;
4: $C^j = \{c_{ij} : c_{ij} \in C_c\}$;
5: **If** $|C^j| = 1$ **then**
6: $C_{bf} \longleftarrow C_{bf} \cup \{c_{ij}\}$, $\sigma'(j) = i$
7: $P_{bf}^c = B_{p\alpha}^{q\alpha}$, $P_{bf}^c \longleftarrow P_{bf}^c \cup \{P_{ij}^c\}$
8: **Else**
9: Sort C^j into a non-ascending list C^j such that
10: $B_i^j(1) \geq B_i^j(2) \geq \cdots \geq B_{p\alpha}^{q\alpha}$
11: **If** the first k $(k \geq 2)$ bids in C^j are equal **then**
12: Randomly choose one buyer from the first k
13: Buyers in C^j;
14: **Else**
15: Select the 1_{st} buyer of C^j whose bid is the
16: Highest;
17: **End if**
18: $C_{bf} \longleftarrow C_{bf} \cup \{c_{ij}\}$, $\sigma'(j) = i$;
19: $P_{bf}^c = B_i^j(2)$, $P_{bf}^c \longleftarrow P_{bf}^c \cup \{P_{ij}^c\}$;
20: **End if**
21: **End for**
22: **Return** $(C_{bf}, M_{bf}, \sigma', P_{bf}^m, P_{bf}^c)$;

Algorithm 2: Assignment and pricing phase.

In the final phase, if one cloudlet in C_{bf} who wins at least two mobile vehicles in M_{bf}, the auctioneer relies on the

efficiency of utilization to decide only one mobile vehicle for a cloudlet. For example, if $c_{i\theta}$ and $c_{i\vartheta}$ are both in the C_{bf}, in other words, c_i as a cloudlet wins 2 mobile vehicles, m_θ and m_ϑ in M_{bf}. The auctioneer would choose one cloudlet according to whose utility is higher. Just as we illustrated above, if there are at least 2 utilities equal, the auctioneer will select only one mobile vehicle stochastically, when algorithm 3 is finished, there should be a one-to-one mapping relationship between the mobile vehicle in M_w and cloudlet in C_w since the redundant mobile vehicles can join another auction.

The idea that cloudlet offloads one task to multiple mobile vehicles seems an excellent way to raise the efficiency of cloudlets, but it may be too ideal to adopt. The major tasks offloaded to mobile vehicles are applications run on mobile vehicles, and the applications may not design for running parallelly or running on heterogeneous hardware.

4.2. A Walk through Example. In this section, we use an example to demonstrate the process of IMCO by using cloudlets' bid in Table 2 and mobile vehicles' ask in Table 3.

The first stage is to determine the winner candidates.

(i) Create the set C^+ from Table 2 $C^+ = \{c_{12}, c_{15}, c_{16}, c_{22}, c_{23}, c_{24}, c_{27}, c_{34}, c_{35}, c_{41}, c_{47}, c_{51}, c_{53}, c_{54}\}$

(ii) Sequencing the cloudlets in C^+ in a nonincreasing order to get $\mathbb{C} = \{c_{15}, c_{41}, c_{35}, c_{53}, c_{27}, c_{54}, c_{12}, c_{34}, c_{47}, c_{51}, c_{16}, c_{23}, c_{24}, c_{22}\}$

(iii) Sorting the mobile vehicles in M in a nondecreasing sequence to obtain $\mathbb{M} = \{m_5, m_1, m_2, m_6, m_4, m_3, m_7\}$

(iv) Drawing the mapping relationship graphic between cloudlets in $\mathbb{C}$ and mobile vehicles in $\mathbb{M}$ in Figure 4

(v) Calculating the two standards: $B_{p\alpha}^{q\alpha} = B_1^6 = 5$ and $A_{j\beta} = A_6 = 4$

(vi) Determining the winner cloudlet and mobile vehicle candidates:

$C_c = \{c_{15}, c_{41}, c_{35}, c_{53}, c_{27}, c_{54}, c_{12}, c_{34}, c_{47}, c_{51}, c_{16}\}$;
$M_c = \{m_5, m_1, m_2\}$

(vii) Drawing the mapping relationship graphic between cloudlets in C_c and mobile vehicles in M_c in Figure 4

(viii) According to Algorithm 2, we can obtain

$C_{bf} = \{c_{15}, c_{41}, c_{12}\}$; $M_{bf} = \{m_5, m_1, m_2\}$;
so we can see $\sigma'(5) = 1, \sigma'(1) = 4, \sigma'(2) = 1$

(ix) Computing the price of cloudlets and determining the clearing price:

$P_{bf}^c = \{P_{15}^c = B_3^5 = 9, P_{41}^c = B_5^1 = 6, P_{12}^c = B_{p\alpha}^{q\alpha} = 5\}$

(x) Determining the payment paid to winning mobile vehicles:

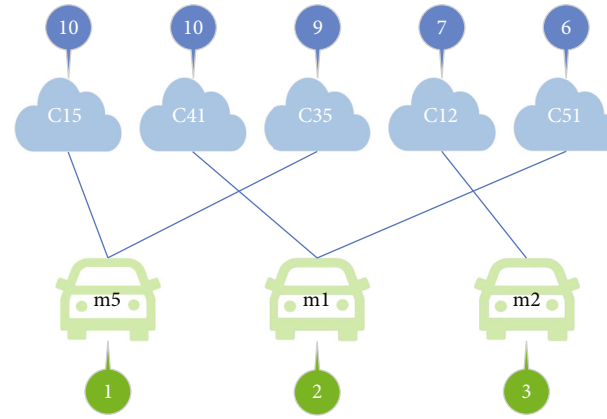


Figure 4: Mapping relationship graphic between C_c and M_c.

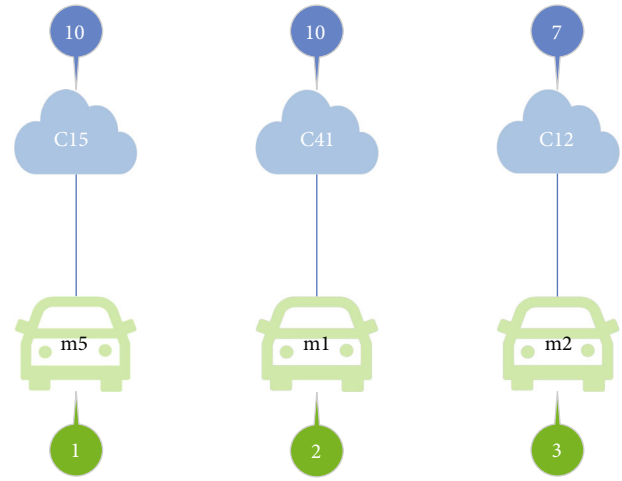


Figure 5: Mapping relationship graphic between C_{bf} and M_{bf}.

$P_{bf}^m = \{P_5^m = P_1^m = P_2^m = A_{j\beta} = 4\}$

(xi) Drawing the mapping relationship graphic between cloudlets in C_{bf} and mobile vehicles in M_{bf} in Figure 5

However, there still has some mobile vehicles server one cloudlet at the same time; auctioneer needs to filter the redundant vehicles.

So, auctioneer executes the elimination algorithm depending on the utilities of cloudlet according to different mobile vehicles.

(i) Calculating the utilities of cloudlet c_1 according to mobile vehicle m_5 and m_2:

$U_{15}^c = B_1^5 - P_{15}^c = 10 - 9 = 1,$
$U_{12}^c = B_1^2 - P_{12}^c = 7 - 5 = 2.$

(ii) It is clear to see that $U_{12}^c > U_{15}^c$, so mobile vehicle m_5 is filtered to achieve higher utility of cloudlet c_1 with m_2. The winning cloudlet is $C_w = \{c_1, c_4\}$

(iii) Determining the winning mobile vehicles: $M_w = \{m_5, m_1\}$

$P_w^c = \{P_1^c = 5, P_4^c = 6\}$, $P_w^m = 4$, $\sigma(1) = 4, \sigma(2) = 1$.

Remember what we illustrate about the TASC mechanism which fails to satisfy the truthfulness. The IMCO overcomes the shortcomings of TASC. As what you can see below.

If C_3 increases its offer B_3^5 from 9 to $9 + \delta$ to cheat auctioneer. ($\delta > 1$). Then, Figure 6 shows the situation when assignment and pricing phase finish. In consequence, the utility of C_3 is 9−10 = -1 which is lower than 0. So, it will be filtered in the next stage. To win the auction, C_3 should bid truthfully.

5. Theoretical Analysis

In this chapter, we analyze the IMCO that we designed with the attributes that we are interested in, including computational efficiency, individual rationality, budget balance, and truthfulness. If and if only those properties are achieved, the auction mechanism can be applied to the scenario.

Theorem 1. *IMCO is computational efficiency.*

Proof. In Algorithm 1, there are not more than $n \times m$ cloudlet customers in the filtered set of cloudlets with positive valuation C^+. Merge-sort sorts the cloudlets from C^+ into C using $O(nm \log (nm))$ times and sorting the mobile vehicles in M using $O(m \log (m))$ time. Then when auctioneer sorts the cloudlets into C_c, the number of cloudlets in C_c is $n \times \lceil 1 + m/2 \rceil$. The remain part of the algorithm, the following for loops, takes $O(\lceil 1 + m/2 \rceil \times (1 + n))$ times. Thus, the time complexity of Algorithm 1 is $O(nm^2 + nm \log n)$.

In Algorithm 2, there exist not more than $\lceil 1 + m/2 \rceil$ cloudlet customers in C_c and mobile vehicles in M_c. To find out the subset of $C_c C_j$, the auctioneer needs to take $O(nm)$ time, since auctioneer has already sort the cloudlets in non-increasing order, so the auctioneer can save some time without sorting again. There are no more than n mobile vehicles, so calculate the winning mobile vehicles only takes $O(n)$ time. The following loops take $O(\lceil 1 + m/2 \rceil) = O(n^2 m^2)$.

In Algorithm 2, there exist no more than $\lceil 1 + m/2 \rceil$ cloudlets in C_{bf} and mobile vehicles in M_{bf}, since $|C_{bf}| = |M_{bf}|$. The loop has a time complexity of $O(|C_{bf}| \times |C_{bf} - 1/2|) = O(m^2)$.

All in all, since the algorithms in IMCO is sequential execution, so the time complexity of IMCO mechanism is $O(nm^2 + nm \log n)$. It means that, IMCO is computational efficiency. □

Theorem 2. *IMCO is individually rational.*

Proof. Each winning mobile vehicle $m_j \in M_w$, the price offered to mobile vehicles m_j is P_w^j which is larger than the ask of m_j on the basis of IMCO. Therefore, mobile vehicles accommodate the demand of desirable property.

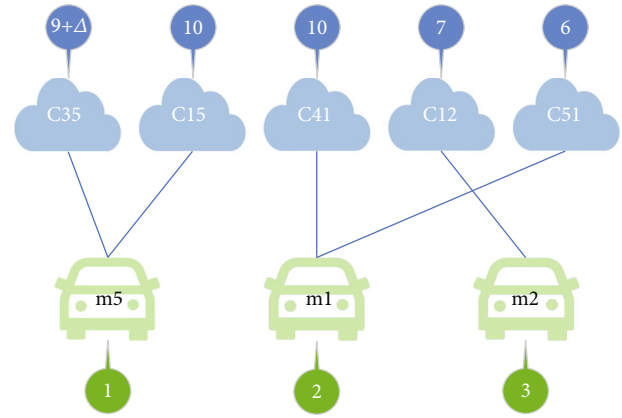


Figure 6: Mapping relationship graphic between C_c and M_c with untruthful biding.

In Algorithm 2, for every cloudlet $c_{ij} \in C_{bf}$, there are two different situations needed to be discussed.

In the former situation, there is only one cloudlet c_{ij} selects one mobile vehicle m_j. According to Algorithm 2, the price that cloudlet c_i needed to submit to auctioneer is the clearing price $B_{p\alpha}^{q\alpha}$ which is less than or equal to the price B_i^j.

In the last case, there are more than one cloudlet competing for one mobile vehicle. Supposing that cloudlet c_i wins the auction; in other words, it has the highest bid valued B_i^j. Referring to Algorithm 2, the c_{ij} is charged the 2nd highest bid in C_j. It is easy to see that P_{ij}^c is no more than B_i^j.

As what we illustrated above, both the cloudlets and mobile vehicles meet the desirable property. In the next stage, auctioneer filters some redundant mobile vehicles and according to the utility of cloudlet. Algorithm 3 will not affect the P_{ij}^m and P_{ij}^c, so it can still has the desirable property. □

Theorem 3. *IMCO is budget-balanced.*

Proof. After the auctioneer filters all redundant mobile vehicles, only one $c_i \in C_w$ mapped to one $m_j \in M_w$. According to the mapping relationship $\sigma(j) = i$, it is easy to see that

$$P_i^c \geq B_{p\alpha}^{q\alpha} \geq A_{j\beta} = P_j^m. \tag{10}$$

That is

$$\sum_{c_i \in C_w} P_i^c - \sum_{m_j \in M_w} P_j^m \geq 0. \tag{11}$$

□

Lemma 4. *IMCO is truthful for mobile vehicles.*

Referring to Algorithm 2, all the winning mobile vehicles are rewarded $A_{j\beta}$. According to the utility formula (2)

Input: C_{bf}, M_{bf}, σ', P^m_{bf}, P^c_{bf};
Output: C_w, M_w, σ, P^m_w, P^c_w;
1: $C_w \longleftarrow C_{bf}$, $M_w \longleftarrow M_{bf}$, $\sigma \longleftarrow \sigma'$, $P^m_w \longleftarrow P^m_{bf}$, $P^c_w \longleftarrow P^c_{bf}$;
2: **For** any two cloudlets $c_{\sigma(a),a}, c_{\sigma(b),b} \in C_w$ & $a \neq b$ **do**
3: **If** $\sigma(a) = \sigma(b)$ **then**
4: $U^c_{\sigma(j)j} = B^j_{\sigma(j)} - P^c_{\sigma(j),j}$, $j = \{a, b\}$
5: **If** $U^c_{\sigma(a),a} = U^c_{\sigma(b),b}$ **then**
6: $j' =$ select from $\{a, b\}$ randomly;
7: **Else**
8: $j' = \min(U^c_{\sigma(a),a}, U^c_{\sigma(b),b})$;
9: **End if**
10: $C_w \longleftarrow C_w \setminus \{c_{\sigma(j'),j'}\}$, $M_w \longleftarrow M_w \setminus \{s_{j'}\}$;
11: $P^c_w \longleftarrow P^c_w \setminus \{P^c_{\sigma(j'),j'}\}$, $P^m_w \longleftarrow P^m_w \setminus P^m_{j'}$,
12: $\sigma(j') = \varnothing$;
13: **End if**
14: **End for**
15: **Return** $(C_w, M_w, \sigma, P^m_w, P^c_w)$;

Algorithm 3: Winner elimination phase.

of mobile vehicles:

$$U^m_j = P^m_j - \text{Cost}_j. \tag{12}$$

Since the cost is regarded as constant and the reward is also constant, the utility of winning mobile vehicles can be improved, in the contrary, if the ask is higher than its cost in some degree (higher than $A_{j\beta}$), the utility will decrease to 0.

Lemma 5. *IMCO is truthful for cloudlets.*

According to Algorithm 2, cloudlet cannot fake biding higher than the actual value of mobile vehicle to compete the mobile vehicle, because the price charging to winning cloudlets is variable. So, the utility of faking cloudlet is below zero. From formula (1), it is easy to obtain

$$U^c_i = V^j_i - P^c_i < 0. \tag{13}$$

Since there are competition in the auction, the winning cloudlet cannot improve its utility by increasing its bid, because the price charged to it depends on the second-highest biding.

Theorem 6. *IMCO is truthful.*

Proof. According to the above lemma, we can get that IMCO is truthful for both mobile vehicles and cloudlets, that is to say, IMCO is truthful. □

6. Performance Evaluation

In this part, we use the dataset to analyze the IMCO to find out whether it satisfies the desirable properties that are proved in the section above. Also, we validate the effectiveness of the proposed approach. Because there has no statistics about the cost of mobile vehicles and the demands of cloudlets. So we use the built-in uniform distribution function to generate the bids of cloudlet and ask of mobile vehicles in the scope from 0 to 1. That is the conclusions hold true no matter what dataset is used.

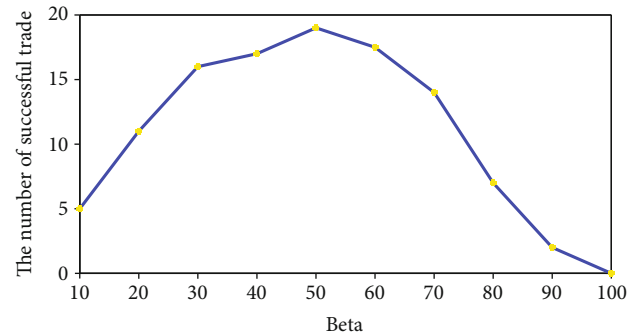


Figure 7: The relationship between the number of successful trades and different β.

6.1. The Influence of Parameter β. First, we randomly generate 100 cloudlets and 100 mobile vehicles, which means there are 10000 bids in the auction. The number of winning cloudlet candidates and winning mobile vehicle candidates is closely related to the value of β. We found out that the largest number of winning cloudlet candidates appears when β is closed to the $\lceil 1 + m/2 \rceil$. So, we are interested in successful trades. Figure 7 shows the relationship between the number of successful trades and different β.

From what we observe from Figure 7 when the β is small, the threshold $B^{q\alpha}_{p\alpha}$ is small enough. However, the number of sellers is too small, so it limited the total number of cloudlet buyers. When the β is large, the threshold $B^{q\alpha}_{p\alpha}$ is

Table 4: Computation time.

$n = 100$	m	50	100	150	200	250	300
	Time(s)	0.160	0.456	0.742	1.034	1.325	1.617

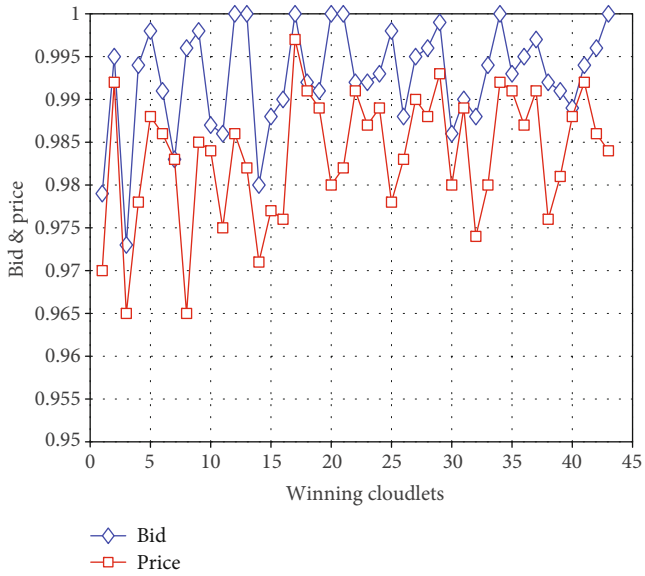


Figure 8: The relationship between the bids and prices of winning cloudlets.

so large that there are few cloudlet buyers who meet the conditions. And the β value is closely related to different values of n and m.

6.2. Computational Efficiency. We randomly generate different numbers of cloudlets and vehicles to test the running time. The computation time is shown in Table 4.

6.3. Individual Rationality. To verify this property, we present both the winning cloudlets' bids and the prices, which are shown in Figure 8. In addition, Figure 9 shows the asks and the payments of winning vehicles. We can see that the bids are always no less than the price charged to cloudlets, and the rewards to vehicles are no less than the asks of vehicles. So, IMCO meets individual rationality. In other words, both the cloudlets and the vehicles can achieve positive utilities. The auctioneer can stimulate vehicles to carry out computation offloading tasks by paying them no less than their asks while the cloudlets can make use of the computation resources of the vehicles.

As what we observe from the figures, the bids are always no less than the price charged to cloudlets, and the payment rewards to mobile vehicles are no less than the asks of mobile vehicles. So, the IMCO meets the individual rationality. In other words, both the cloudlets and mobile vehicles can achieve positive utilities, and it is a win-win game. The auctioneer can stimulate mobile vehicles to carry out computation offloading tasks by paying them no less than their asks, and the cloudlets can make use of the computation resources of mobile vehicles.

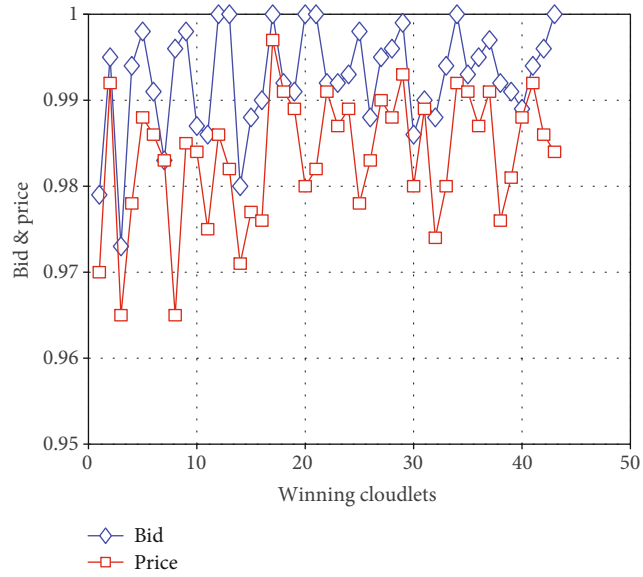


Figure 9: The relationship between the asks and payments of winning mobile vehicles.

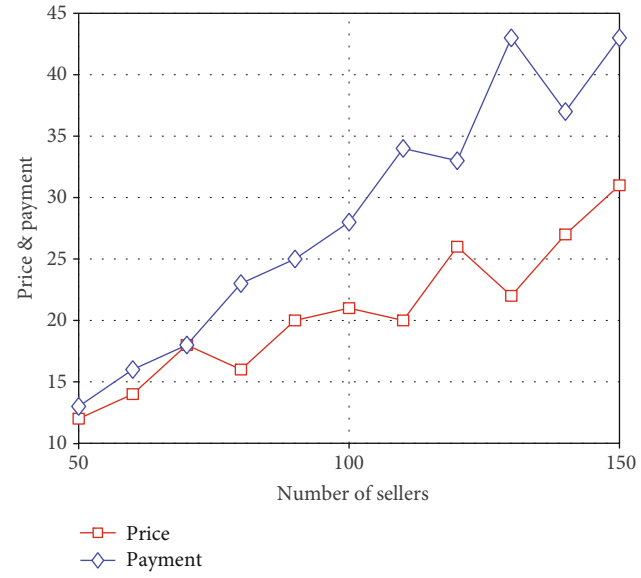
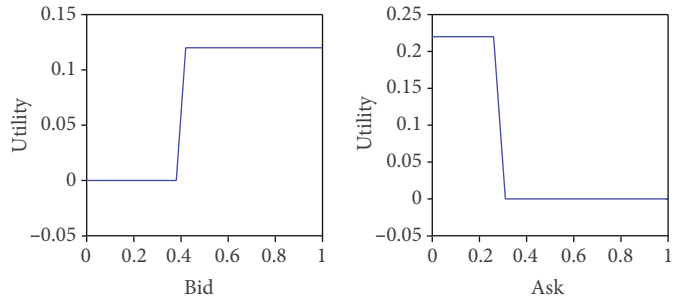


Figure 10: The relationship between the prices and payments of IMCO.

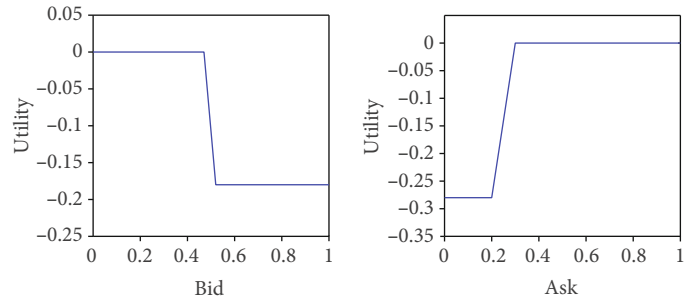
6.4. Budget Balance. IMCO is only of budget balance when the total fund that the cloudlets submit to the auctioneer is larger than or equal to the payments which are rewarded to the winning vehicles. We set the number of total cloudlets to 100 consistently. By changing the number of vehicles from 50 to 150, we obtain the total prices and payments. The result is shown in Figure 10 showing each time the total number of vehicles increases by 10. It is clear that the total prices are always larger than the total payments.

As the result shows, the total prices are always larger than the total payments.



(a) Buyer (b) Seller

Figure 11: The utilities of vehicles and cloudlets in the winning set.



(a) Buyer (b) Seller

Figure 12: The utilities of vehicles and cloudlets not in the winning set.

6.5. Truthfulness. We focus on two different cloudlets and mobile vehicles groups which contain both winners and losers. Then, by changing their bids and asks from 0 to 1, we obtain the changing trends of utilities. The result of those in the winning set is shown in Figure 11. Figure 11(a) shows that when the cloudlet offers a truthful bid of 0.42, it has a utility of 0.12. However, no matter how higher a bid the cloudlet selects, the utility still stays the same, unless it reduces the bid under the original price. Figure 11(b) shows that the utility of mobile vehicles cannot be improved by asking untruthfully. Offering asks lower than 0.25 cannot achieve higher utilities while the utilities are always positive. The result of those not in the winning set is shown in Figure 12. Figure 12(a) shows that the utility of the losing cloudlet is always no larger than 0 since it loses the auction. However, even if the losing cloudlet fakes its bid, it still cannot make the utility larger than 0. Figure 12(b) shows that the utility of a losing mobile vehicle can only be 0 or below 0. Overall, IMCO can achieve truthfulness for both cloudlets and mobile vehicles.

In conclusion, IMCO can achieve truthfulness for both cloudlets and mobile vehicles. They cannot improve their utilities by lying.

6.6. System Efficiency. In Section 5, we have proved that IMCO meets the desirable properties in theoretical, and Section 6 uses the numerical results shows the above properties. As what we address, the TASC auction mechanism cannot guarantee the property of truthfulness applied in the cloudlet scenario. In Figure 13, the system efficiency of IMCO and TASC is evaluated as the number of successful trades.

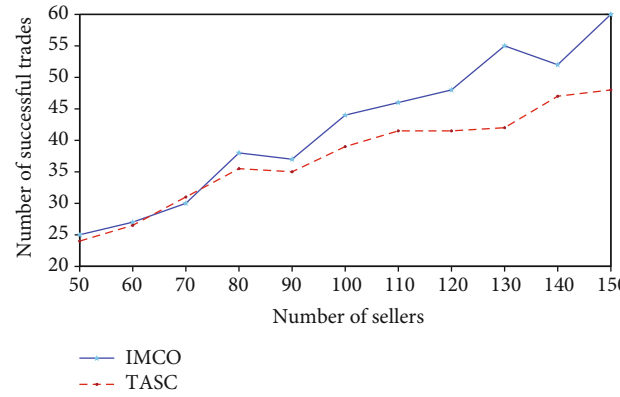


Figure 13: System efficiency with uniformly distributed datasets.

Figure 14 shows the efficiency of TASC and IMCO according to a number of successful trades. The bids and asks are generated by the MATLAB built-in unified function which is used to generate uniformly distributed datasets. Some of the efficiency of IMCO is sacrificed to guarantee truthfulness. Thus, the total efficiency of IMCO still has an advantage over TASC since IMCO maintains more winning cloudlet candidates than TASC, and IMCO only filters a few winning cloudlets in the third stage. Hence, IMCO can achieve a reasonable efficiency with all the desirable properties.

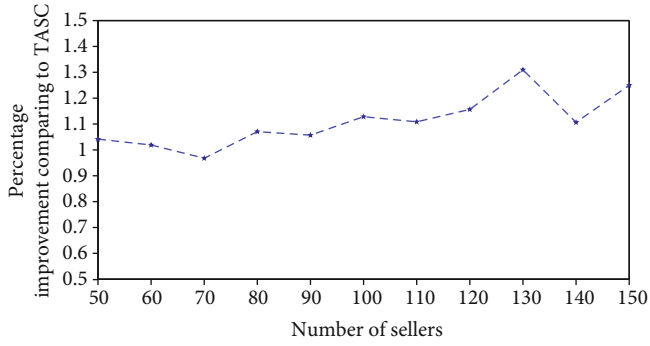


Figure 14: Percentage improvement of normalized system efficiency comparing to TASC with uniformly distributed datasets.

7. Conclusion

In this paper, we concentrate on an emerging paradigm in the current IoT systems, in which computation is offloaded to vehicle owners. Because of the distance between mobile vehicles and cloudlets and the distinct computational capabilities of mobile vehicles, the mobile vehicles can be seen as heterogeneous sellers to cloudlets, so different cloudlets may evaluate each mobile vehicle due to different demands and give different valuations toward different vehicles. In order to make mobile vehicles offer the computation offloading servers to cloudlets, we have proposed the IMCO which deals with the resource exchanging between mobile vehicles and cloudlets. IMCO can meet the properties such as individual rationality, budget balance, system efficiency, and truthfulness for both cloudlets and mobile vehicles.

There are some aspects that may be improved in future work, such as the system efficiency is not that satisfying even it is better than TASC as a whole since IMCO sacrifices some efficiency to maintain the truthfulness for both cloudlets and mobile vehicles. The other problem is that even if the time complexity of the mechanism is acceptable, the program in MATLAB could not be seen as of high-efficiency. So we could optimize the programs to execute faster as we expected in further study.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by Research Innovation Fund for College Students of Beijing University of Posts and Telecommunications.

References

[1] S. Xia, F. Lin, Z. Chen, C. Tang, Y. Ma, and X. Yu, "A Bayesian game based vehicle-to-vehicle electricity trading scheme for blockchain-enabled internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 6856–6868, 2020.

[2] M. Chen, S. Guo, K. Liu, X. Liao, and B. Xiao, "Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 2025–2040, 2021.

[3] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial iot in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2020.

[4] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[5] L. Li, T. Q. Quek, J. Ren, H. H. Yang, Z. Chen, and Y. Zhang, "An incentive-aware job offloading control framework for multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 63–75, 2021.

[6] A. Mukherjee, D. De, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *Computing*, vol. 7, no. 1, pp. 141–154, 2019.

[7] B. Zhao, S. Tang, X. Liu, and X. Zhang, "PACE: privacy-preserving and quality-aware incentive mechanism for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1924–1939, 2021.

[8] R. Han, Q. Guan, F. R. Yu, J. Shi, and F. Ji, "Congestion and position aware dynamic routing for the internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16082–16094, 2020.

[9] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5871–5883, 2019.

[10] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[11] G. Pan, J. Ye, Y. Tian, and M.-S. Alouini, "On HARQ schemes in satellite-terrestrial transmissions," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7998–8010, 2020.

[12] C. Lin, Y. Yu, J. Xiong et al., "Shrimp: a robust underwater visible light communication system," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 134–146, New York, NY, USA, 2021.

[13] C. Lin, F. Gao, H. Dai, J. Ren, L. Wang, and G. Wu, "Maximizing charging utility with obstacles through Fresnel diffraction model," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 2046–2055, Toronto, ON, Canada, 2020.

[14] C. Lin, Z. Wang, J. Deng, L. Wang, J. Ren, and G. Wu, "mts: temporal- and spatial-collaborative charging for wireless rechargeable sensor networks with multiple vehicles," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, USA, 2018.

[15] C. Lin, Z. Yang, H. Dai, L. Wang, and G. Wu, "Minimizing charging delay for directional charging," *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2478–2493, 2021.

[16] C. Lin, Y. Zhou, F. Ma, J. Deng, L. Wang, and G. Wu, "Minimizing charging delay for directional charging," *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2478–2493, 2019.

[17] C. Lin, J. Zhou, C. Guo, H. Song, G. Wu, and M. S. Obaidat, "TSCA: a temporal-spatial real-time charging scheduling algorithm for on-demand architecture in wireless rechargeable sensor networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 211–224, 2018.

[18] Y. Sun, C. Lin, H. Dai et al., "Trading off charging and sensing for stochastic events monitoring in wrsns," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2021.

[19] C. Lin, Z. Shang, W. Du, J. Ren, L. Wang, and G. Wu, "Codoc: a novel attack for wireless rechargeable sensor networks through denial of charge," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 856–864, Paris, France, 2019.

[20] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[21] R. P. McAfee, "A dominant strategy double auction," *Journal of Economic Theory*, vol. 56, no. 2, pp. 434–450, 1992.

[22] D. Yang, X. Fang, and G. Xue, "Truthful auction for cooperative communications," in *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 1–9, Paris, France, 2011.

[23] X. Zhang, H. Zhang, S. Dai, and Y. Liu, "An incentive mechanism for computation offloading in satellite-terrestrial internet of vehicles," in *2021 the 7th International Conference on Computer and Communications*, Chengdu, Sichuan, China, 2021.

[24] A. Guidotti, A. Vanelli-Coralli, M. Conti et al., "Architectures and key technical challenges for 5G systems incorporating satellites," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2624–2639, 2019.

[25] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[26] Z. Wang, D. Zhao, M. Ni, L. Li, and C. Li, "Collaborative mobile computation offloading to vehicle-based cloudlets," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 768–781, 2021.

[27] V. Krishna, *Auction Theory*, Academic Press, 2009.

[28] A. Yassine, M. S. Hossain, G. Muhammad, and M. Guizani, "Cloudlet-based intelligent auctioning agents for truthful autonomous electric vehicles energy crowdsourcing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5457–5466, 2020.

[29] X. Chen, T. Zhang, W. Ye, Z. Wang, and H. H.-C. Iu, "Blockchain-based electric vehicle incentive system for renewable energy consumption," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 396–400, 2021.

[30] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2019.

[31] J. Du, E. Gelenbe, C. Jiang, Z. Han, and Y. Ren, "Auction-based data transaction in mobile networks: data allocation design and performance analysis," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1040–1055, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8804530, 12 pages
https://doi.org/10.1155/2022/8804530

Research Article

An Adaptive Task Migration Scheduling Approach for Edge-Cloud Collaborative Inference

Boyin Zhang,[1] Yinggang Li,[1] Shigeng Zhang,[1,2] Yue Zhang,[3] and Bing Zhu[1]

[1]*School of Computer Science and Engineering, Central South University, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[3]*Hunan Provincial Key Laboratory of Network Investigational Technology, Hunan Police Academy, China*

Correspondence should be addressed to Shigeng Zhang; sgzhang@csu.edu.cn

Received 29 September 2021; Accepted 25 November 2021; Published 17 January 2022

Academic Editor: Pengfei Wang

Deep Neural Network (DNN) models have achieved excellent performance in many inference tasks and have been widely used in many intelligent applications. However, DNN models often require a lot of computational resources to complete the inference tasks, which hinders the deployment of such models to resource-constrained edge devices. In order to extend the application scenarios of DNN models, the edge-cloud collaborative inference methods, represented by model partition, have attracted much research attention in recent years. In scenarios that have multiple edge devices deployed, the edge-cloud collaborative inference method requires partial migration of tasks, but traditional scheduling methods only migrate tasks at the task level. In this paper, we propose two task scheduling methods, which can solve the problem of partial migration of tasks in multiedge scenarios. The first scheduling method is based on the optimal cutting of a single DNN. The cutting positions of all the models are the same, regardless of the influence of external factors. This method is suitable for chain and directed acyclic graph- (DAG-) type DNNs. The second scheduling method takes external factors such as congestion and queuing delay at the cloud side into consideration, which dynamically selects the cutting position of each DNN to optimize the overall delay and thus is applicable to chain DNN models. The experimental results show that, compared with the baseline method, our proposed scheduling method can reduce the delay by up to 6.48x.

1. Introduction

With the rapid development of Deep Neural Network (DNN), edge intelligence has been widely applied in the Internet of Things (IoT) scenarios in recent years [1, 2]. The growing proliferation of IoT devices with high-quality sensors will result in massive data streaming to the edge or the cloud. Edge devices deployed generally have some constraints including energy and computational capacity [3]. Thus, process data and inference only at edge ends will cause high delays that cannot meet the requirements of most applications. Cloud servers have high capacities of computation; however, transferring all data to the cloud will suffer double network delay that is intolerant in bad network conditions. Moreover, transferring whole data to the cloud takes the risk of privacy leakage [4]. Edge-cloud collaborative inference, which assigns tasks between the edge and the cloud on the basis of constraints in applications, has become a research focus nowadays.

In the edge computing environment, there are usually multiple edge devices collecting data at the same time, but the number of corresponding cloud servers is very small [5]. If a lot of tasks are generated at the edge at the same time, how to better schedule these tasks to reduce the overall delay is a problem we need to study.

The edge-cloud collaborative inference can adaptively cut the DNN according to the network bandwidth without changing the original model parameters of the DNN [6], so as to minimize the delay or maximize the throughput. For the deployment of a single edge, determine the cutting position by network bandwidth is obviously the optimal solution. However, in the real edge environment, there are

often multiple edge devices generating tasks. Due to a large number of edge tasks, the tasks that the cloud can process at the same time are limited. Therefore, how to schedule these tasks is very important. In this paper, we design a system that can perform task scheduling in multiple edge scenarios.

For a single-edge system, we determine how the DNN is cut based on the current network status and the fixed configuration parameters of the system (such as the execution time of the DNN on the edge and the cloud), through the goal of minimizing delay or maximizing throughput, find the optimal cutting point. However, in a multiedge scenario, the cutting point for minimizing the delay for a single task is not necessarily optimal for the entire system. Due to the limited processing capacity of the cloud, when the number of tasks generated at the edge is much greater than the processing capacity of the cloud, many tasks will face the problem of blocking. When the edge end completes its own inference task, it transmits the intermediate processing result to the cloud, and the cloud cannot process it in time, which causes the task to be blocked. Although the execution time of each task at the edge is optimal, the waiting time in the cloud increases the processing time of the entire system.

In this scenario, minimizing the delay of each DNN may not be the optimal solution for the entire system. If cloud task congestion is detected, the position of the DNN is appropriately adjusted so that the task it is waiting for is executed at the edge. Although the delay cannot be minimized for a single DNN, it will make the overall delay of the system less. To our knowledge, most of the scheduling schemes for the DNN model at present are scheduling the entire DNN model. Due to the dependency and complexity of the edge-cloud collaborative system scheduling problem, few researches focus on its scheduling strategy. In [7], the author studied the scheduling system in the scenario of multiple IoT devices; it suggested that each type of IoT device has different computing capabilities and designed an online scheduling (Online) system. Online can decide whether DNN is deployed locally or in the cloud, and it can also adjust the scheduling sequence. The author compares Online with first-come, first-served (FIFO) and low-bandwidth first deployment (LBF) strategies. Compared with the other two methods, Online can improve the overall quality of service. In [8], the author proposed a migration scheduling problem for DNN tasks in edge environment, gave the formal definition and evaluation criteria of the problem, and proposed a greedy algorithm and a genetic algorithm for the migration. The problem-solving is approximately optimal, which can effectively solve the migration and deployment problems of DNN.

In this paper, we designed two task scheduling strategies for edge-cloud collaborative inference. The first strategy finds the optimal cutting position of DNN models for every single task which is capable for DNN models with DAG and chain topologies. The second strategy decides cutting position on the basis of network condition to reach global optimal for all tasks, which is capable for DNN models with chain topology.

The remainder of this paper is organized as follows. In Section 2, we conduct a brief literature review of related studies. In Section 3, we define the problem to be addressed. In Section 4, we introduce our approach in detail. Section 5 shows the results of the experiments. We conclude in Section 6.

2. Related Work

Much work has been done to accelerate the inference of DNN models in IoT scenarios. In order to reduce the delay of inference at edge devices, the main research interests are divided into three aspects: model compression [9], distributed model deployment [10], and computation offloading [11].

Compression of DNN models uses technics such as prune [12], quantization [13], and knowledge distillation [14] to reduce the computation operations in model inference without significant accuracy reduction [15]. Although model compression speed up the inference on edge devices significantly, the compression itself needs lots of computation to complete and the inference of a compressed model may need specific hardware to complete. For the application environments that contain heterogeneous architecture edge devices, this method is not competent. There are multiple studies about process DNN models on resource-constrained devices effectively [16, 17]. To inference DNN models effectively on resource-constrained devices, researchers design the hardware architecture deliberately [18] and hardware-accelerating methods usually combined with model compression methods which can improve the efficiency of specific model inference significantly [19]. However, these methods are not universal to various applications compared to other methods. Edge devices in real application scenarios are generally heterogeneous in architectures; hardware accelerating requires specific computation units or ICs to execute the model effectively. Distributed deployment of DNN models can make full use of the computing resources of devices [10] and is capable for large-scale applications. However, for edge intelligence environments, the number of devices may vary dynamically; distributed deployment cannot handle this well. Moreover, distributed deployment on multiple edge devices may cause network congestion, especially under bad network conditions.

Edge-cloud collaborative inference has unique advantages over the first two technologies; it does not change the original model compared with lightweight model and distributed network deployment. Edge-cloud collaborative inference has high scalability and can be combined with the other two technologies. For edge-cloud collaborative inference, a lot of previous work has been done to achieve the purpose of reducing overall inference delay [8, 20–23] or to satisfy resource constraints [24–27] (bandwidth, power consumption, etc.) or to protect privacy [28]. Applying traditional artificial intelligence technology to edge computing, which is usually resource-constrained, researchers' ideas are mainly divided into the following three kinds: DNN model selection depending on sample [29], design lightweight DNN architectures [30] or DNN model compression [15],

and edge-cloud collaborative inference by cutting DNN models and scheduling tasks between edge and cloud [25].

In [29], the authors apply an autoencoder to compress the data transmitted to cloud platforms. Kang et al. [6] first proposed Neurosurgeon, a method that partitions the DNN model to execute on end devices and cloud platforms simultaneously to improve the efficiency of the model inference. It cuts a DNN into two parts and executes on a mobile device and the cloud platform, respectively, to accelerate the inference of the model. However, Neurosurgeon can only partition chain-like DNNs; it cannot handle DAG structure DNNs, which limits its application. What is more, Neurosurgeon does not have enough accuracy on execution time prediction because of the linear regression method it used. This leads to a nonoptimal partition of the model. Teerapittayanon et al. proposed DDNN [31] to speed up the latency of inference of a DNN model over distributed computing hierarchies, consisting of the cloud, the edge (fog), and end devices. However, DDNN is designed for BranchyNet [32] and is hard to be extended to other types of DNN models. In [33], the authors presented a DNN as an encoding pipeline that encodes the feature space and transmits it to the clouds. It improves the energy efficiency and throughput of the model inference. Edgent [21] exploits two knobs: DNN partitioning and DNN right-sizing to find the optimal cutting point in a dynamic network environment. Hu et al. [22] proposed DADS, a partition scheme that optimally cut the DNN with DAG topology under different network conditions. However, it fails to reduce the overall delay because of the high time complexity of the algorithm. This method cannot guarantee real-time application. In reference [34], the authors studied the mobile Web AR scenario for edge-cloud collaboration and proposed a fine-grained adaptive DNN partition mechanism. In [35], the authors studied the edge-cloud inference of RNN models. It can decide whether to offload to clouds depending on network condition and input size. Wang et al. [36] proposed a task scheduling algorithm for tasks that need to be transferred to the cloud based on the catastrophic genetic algorithm (CGA) to satisfy the latency constraint. In [37], a novel DNN architecture was design for edge-cloud collaborative inference. However, this method is difficult to apply to currently running IoT applications. In [38], Auto-Split was proposed as an industry solution of DNN splitting for edge-cloud collaborative inference.

3. Problem Definition

First, we consider the chain topology DNN. For a given chain DNN model, we construct it as a chain $L = <V, E>$. Each vertex $v_i \in V$ corresponds to one layer in the model, and $<v_{i-1}, v_i> \in E$ corresponds to the two layers of model M with data transmission. $T^e(j, i)$ and $T^c(j, i)$ represent the delays from the j-th layer to the i-th layer at the edge and the cloud, respectively. For a single DNN, after using the edge-cloud collaborative inference strategy, if the cutting is performed on the i-th layer of the DNN, the first to i-th layers will be deployed on the edge, and the $i+1$th~Nth layer will be deployed in the cloud. The output d_i of vertex v_i will be transmitted to the cloud through the network. When the bandwidth is B, the total inference delay is $T^e(1, i) + d_i/B + T^c(i+1, N)$. The best cutting layer of a single DNN is the point where the total inference delay is minimized. The optimization objective of a single DNN is

$$T_{\min} = \operatorname{argmin}\left(T^e(1, i) + \frac{d_i}{B} + T^c(i+1, N)\right)(i = 0, 2, \cdots, N). \tag{1}$$

For chain DNN, we apply an iterative algorithm to enumerate the inference delay required by the DNN segmented in each layer and take the minimum value. In the above equation, $i = 0$ means that the entire DNN is deployed at the edge, and $i = N$ represents that the entire DNN is deployed in the cloud. When multiple edge devices generate tasks at the same time, at this time, each DNN has inference delay T_e, transmission delay T_t, and cloud inference delay T_c at the edge. Since the cloud is not always able to process the tasks sent from the edge in time, the waiting delay T_w of each task needs to be considered. At this time, the goal of optimization has changed from minimizing the delay of a single DNN to minimizing the overall delay of the system.

As shown in Figure 1, there are E edge servers and one cloud server in the multiedge scenario. Assume that E edge servers have a total of M DNN inference tasks, and all tasks use the same type of DNN for inference.

4. Method Design

In this section, we designed two scheduling strategies to solve the problem of partial migration of tasks in multiedge scenarios. The first strategy named Single Cutting treats each DNN as the same; this strategy is based on finding the optimal cutting position of a single DNN. The second strategy named Scheduling with Queuing takes other external factors into account; it adjusts the cutting position dynamic according to the network condition.

4.1. Single Cutting. For the scheduling strategy for each DNN inference task, we use the strategy that minimizes the delay of a single DNN for scheduling. Since we are using the same type of DNN, for each task, when the bandwidth is constant at B, $T_{\{\min\}}$ is the same. We use an iterative algorithm to find the edge processing time t_e and data transmission time at this time t_t, cloud processing time t_c, and optimal cutting point v_{best}.

We use FIFO to schedule all tasks. When there are tasks in task set S, the scheduler will not terminate. Each edge end e maintains a variable, which represents the earliest time at which the edge end can process the next task edgetime$_e$. In each poll, if there is still a task that has not been processed on the edge end e, we will update the current edgetime$_e$ to the maximum value between the arrival time of the current task and edgetime$_e$, and finally, add the current task processing time t_e.

After updating the time at which all edges can process the next task, we select the edge that can be processed

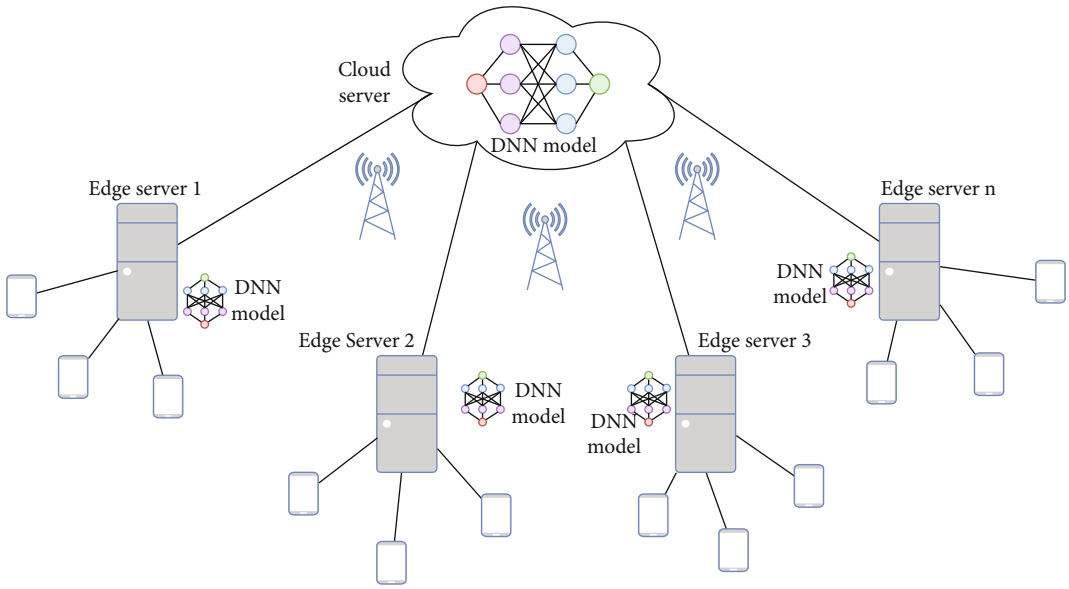


Figure 1: Model architecture in multiedge scenarios.

earliest among them. Next, we check whether the task queue C in the cloud is full. If the task queue is not full, we transfer the intermediate variables of the current task to the cloud task queue. Otherwise, the current task will enter a blocking state, that is, after waiting for the cloud to process the first task of the team, we will transfer the current task to the cloud for processing. After the cloud finishes processing a task, it updates the processing time of the cloud and wakes up the blocked task to start the next poll. The whole process is shown in Algorithm 1.

4.2. Scheduling with Queuing. Scheduling strategy 1 uses the optimal division of a single DNN for each task, but it is not necessarily the optimal solution globally, because no matter whether the cloud is currently congested or not, each DNN will choose the optimal cutting point under noncongested conditions for segmentation. It may leave the edge end in an idle state, and the cloud has been waiting for tasks, which will cause the overall time to become longer.

Suppose the current DNN inference task to be processed is task, and the start time at which the edge end can process it is t_{start}. The task is divided according to the optimal division of a single DNN, the time to reach the cloud is $t_{\text{start}} + t_e + t_t$, and the time when the previous task pretask is executed is t_{end}. If $t_{\text{start}} + t_e + t_t \geq t_{\text{end}}$, then there will be no waiting time for the DNN to reach the cloud. At this time, v_{best} is cut into the optimal solution of the task. If $t_{\text{start}} + t_e + t_t < t_{\text{end}}$, after the task arrives in the cloud, the waiting time is $t_{\text{wait}} = t_{\text{end}} - t_{\text{start}} - t_e - t_t$. At this time, a new cutting point can be selected after v_{best} to divide the task. In this case, you can select a new cutting point after v_{best} to divide the task. At the new cutting point, the edge end processing time is t'_e, the data transmission time is t'_t, the cloud processing time is t'_c, and the waiting time $t'_{\text{wait}} = t_{\text{end}} - t_{\text{start}} - t'_e - t'_t$. If $t'_{\text{wait}} < t_{\text{wait}}$, then for the task, the new cutting point is better than v_{best}, the waiting time will be less, and because the new cutting point is after the optimal cutting point, there will be fewer layers executed in the cloud.

However, when selecting a new cutting point, the execution time of the edge is $t'_e > t_e$, which causes the waiting time of the next DNN task to be processed too long. This may not be conducive to the overall delay reduction; we need to take into account the waiting delay of the next task in the edge device. Assume that the next task that needs to be inferred at the current edge is nexttask. At the original cutting point, we can get that the waiting time for nexttask is $t_{\text{start}} + t_e - \text{arrivetime}_{\text{nexttask}}$, then the overall waiting time of the system is

$$t_{\text{wait}} + t_{\text{start}} + t_e - \text{arrivetime}_{\text{nexttask}}. \tag{2}$$

$\text{arrivetime}_{\text{nexttask}}$ represents the arrival time of the next task. Similarly, the overall waiting delay of the system at the new cutting point is

$$t'_{\text{wait}} + t_{\text{start}} + t'_e - \text{arrivetime}_{\text{nexttask}}. \tag{3}$$

It is necessary to ensure that the overall delay of Equation (3) is less than Equation (2), that is, $t_{\text{wait}} - t'_{\text{wait}} > t'_e - t_e$. Based on the above analysis, we design scheduling strategy 2, and Algorithm 2 describes the process of scheduling strategy 2.

4.3. DAG-Type DNN Scheduling Method. In the above sections, we took the chain DNN as an example and proposed two scheduling algorithms. The first method is that for each DNN, we cut at the optimal cutting point of a single DNN and then schedule. The second method is to consider the waiting delay of the DNN in the cloud and find the cutting point that can make the overall delay smaller on the basis of the optimal cutting point of a single DNN to cut and then perform scheduling.

The cutting point of each DNN of Single Cutting is determined, so for DAG-type DNNs, we can modify the

Input: DNN topology L, layer N; bandwidth B; edge number E; task set S; task queue capacity C
Output: total time to execute all tasks cloudtime
$t_e, t_t, t_c, v_{\text{best}} \longleftarrow \text{BestPartition}(L, N, B)$
for $e = 1, 2, \cdots, E$ **do**
edgetime$[e] \longleftarrow 0$
end for
cloudtime $\longleftarrow 0$
$\text{cloud}_{\text{queue}} \longleftarrow \{\}$
needupdate $\longleftarrow$ true
while $S! = \varnothing$ **then**
  minedgetime $\longleftarrow +\infty$
  nowtask $\longleftarrow \varnothing$
  for $e \longleftarrow 1, 2, \cdots, E$ **do**
    task $\longleftarrow$ getLastArriveTask(S, e)
    if task! $= \varnothing$ and needupdate **then**
      edgetime$[e] \longleftarrow \max$ (edgetime$[e]$, task.arrivetime) $+ t_e$
    end if
    if minedgetime > edgetime$[e]$ **then**
      edgetime$[e] \longleftarrow$ minedgetime
      nowtask $\longleftarrow$ task
    end if
    if length$(\text{cloud}_{\text{queue}}) < C$ **then**
      add {nowtask, t_t, t_c} to $\text{cloud}_{\text{queue}}$
      delete nowtask from S
      needupdate $\longleftarrow$ true
    else then
      needupdate $\longleftarrow$ false
    end if
    process_update$(\text{cloud}_{\text{queue}}$.front, cloudtime$)$
  end for
end while
return cloudtime

Algorithm 1: Single Cutting.

BestPartition(L, N, B) function in Single Cutting to QDMP algorithm. The Scheduling with Queuing method needs to find the point that can make the waiting delay smaller by enumerating the cutting points to shorten the overall delay of the system. For the chain model, we can find the optimal cutting point by enumerating the cutting points sequentially. But for the DAG model, the optimal edge cut set often corresponds to a set of vertices and edges, and enumerating all points and edges is an NP-hard problem. We cannot find a better set of cut edges by enumerating vertices sequentially. Therefore, in this article, we only discuss Single Cutting on DAG-type DNNs.

5. Experimental Evaluation

5.1. Experimental Environment Settings. We use multiple edge devices and a single cloud for scheduling simulation. For the edge, we use 5 Raspberry Pi 3B platforms, which are equipped with a 4-core ARM Cortex-A53@1.2 GHz processor and 1 G RAM. For the cloud, we used a laboratory server equipped with an 8-core Intel core i7-9700 k@3.60 GHz processor and a NVIDIA RTX 2080Ti GPU. For the dataset, we use the self-acquired video dataset to evaluate our proposed scheduling algorithms. Each edge will sample the video frame, extract 5~15 frames of pictures from the video every second, and use the DNN model for inference. In the experiment, we considered six different DNN models, including three chain DNN models and three DAG-type DNN models. The three chain models are AlexNet, Tiny-YOLO, and DarkNet19, respectively. The three DAG models are AlexNet-Parallel, ResNet-18, and GoogLeNet. We evaluated the edge-only method, the cloud-only method, Single Cutting method, and Scheduling with Queuing method.

5.2. Inference Delay under 3G and 4G Networks. We first compare the inference delays of different methods in different network states. We use 3G and 4G as the default transmission technology, and the theoretical maximum uplink bandwidth is 1.1 Mbps and 5.85 Mbps, respectively. We use 5 Raspberry Pi 3B, each Raspberry Pi 3B will input a video stream, each edge end samples 30 frames of pictures in the video, and the cloud cache queue size is 20. In Table 1, we list the total inference delay of the system under different scheduling methods using different DNNs. For chain DNN, we use both two scheduling methods for scheduling. For DAG-type DNN, we use Single Cutting for scheduling.

Input: DNN topology L, layer N; bandwidth B; edge number E; task set S; task queue capacity C
Output: total time to execute all tasks cloudtime

$t_e, t_t, t_c, v_{\text{best}} \longleftarrow \text{BestPartition}(L, N, B)$
for $e = 1, 2, \cdots, E$ **do**
edgetime$[e] \longleftarrow 0$
end for
cloudtime $\longleftarrow 0$
$\text{cloud}_{\text{queue}} \longleftarrow \{\}$
needupdate $\longleftarrow$ true
while $S! = \varnothing$ **then**
  minedgetime $\longleftarrow +\infty$
  nowtask $\longleftarrow \varnothing$
    for $e \longleftarrow 1, 2, \cdots, E$ **do**
      task $\longleftarrow$ getLastArriveTask(S, e)
      next_task $\longleftarrow$ getSecondLastArriveTask(S, e)
      if task! $= \varnothing$ **then**
        edgetime$[e] \longleftarrow$ max (edgetime$[e]$, task.arrivetime)
    end if
    $t_{\text{wait}} \longleftarrow$ cloudtime $-$ edgetime$[e] - t_e - t_c$
    if $t_{\text{wait}} \le 0$ **then**
      $t_t' \longleftarrow t_t, t_e' \longleftarrow t_e, t_c' \longleftarrow t_c$
    else then
    for $i \longleftarrow v_{\text{best}} + 1, \cdots, N$ **do**
      $t'_{\text{wait}} \longleftarrow$ cloudtime $-$ edgetime$[e] - T^e(1, i) - d_i/B$
      if $t'_{\text{wait}} \ge 0$ and $t'_{\text{wait}} < t_{\text{wait}}$ and $t_{\text{wait}} - t'_{\text{wait}} > t_e' - t_e$ **then**
      $t_{\text{wait}} \longleftarrow t'_{\text{wait}}, t_e' \longleftarrow T^e(1, i), t_t' \longleftarrow d_i/B, t_c' \longleftarrow T^c(i + 1, N)$
      end if
    end for
  end if
  if needupdate **then**
    edgetime$[e] \longleftarrow$ edgetime$[e] + t_e'$
  end if
  if minedgetime > edgetime$[e]$ **then**
  edgetime$[e] \longleftarrow$ minedgetime
    nowtask $\longleftarrow$ task
  end if
  if length$(\text{cloud}_{\text{queue}}) < C$ **then**
    add {nowtask, t_t, t_c} to $\text{cloud}_{\text{queue}}$
    delete nowtask from S
    needupdate $\longleftarrow$ true
  else then
    needupdate $\longleftarrow$ false
  end if
    process_update($\text{cloud}_{\text{queue}}$.front(), cloudtime)
  end for
end while
return cloudtime

Algorithm 2: Scheduling with Queuing.

We further evaluated the speedup of different scheduling methods with the "edge-only" delay as a baseline. S-C and S-Q represent the inference delay of Single Cutting strategy and Scheduling with Queuing, respectively, and the speedup is defined as

$$\kappa = \frac{L_{\text{EdgeOnly}}}{L_{\text{ALGO}}}, \tag{4}$$

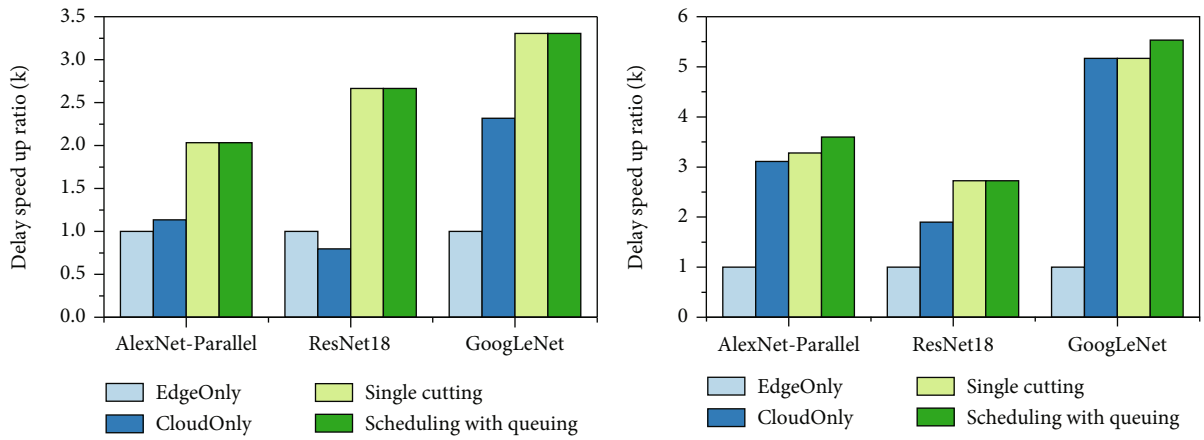
where L_{EdgeOnly} represents the inference delay of only the edge and L_{ALGO} represents the inference delay of all comparison methods. "Edge-only" is used as the baseline, and its speedup ratio is 1. The edge-only method executes the whole task on edge devices while the cloud-only method executes the whole task on the cloud server.

Figure 2 compares the chain DNN. Figure 2(a) shows the latency acceleration ratios of the four methods in the case of 3G. Both Single Cutting and Scheduling with

Table 1: The total inference delay of several typical DNN models under different scheduling methods (seconds).

DNN	3G				4G			
	C-O[1]	E-O[2]	S-C[3]	S-Q[4]	C-O[1]	E-O[2]	S-C[3]	S-Q[4]
AlexNet	78.3	88.9	43.7	43.7	28.6	88.9	27.1	24.7
Tiny-YOLO	200.8	159.7	59.9	59.9	84.2	159.7	58.6	58.6
DarkNet19	77.8	180.4	54.6	54.6	34.9	180.4	34.9	32.6
AlexNet-Parallel	70.2	44.9	40.6	—	27.9	44.9	27.9	—
ResNet18	84.3	81.7	32.0	—	34.4	81.7	25.5	—
GoogLeNet	91.2	121.1	63.3	—	43.2	121.1	43.2	—

[1]Cloud-only, [2]edge-only, [3]Single Cutting, and [4]Scheduling with Queuing.



(a) 3G Network (b) 4G Network

Figure 2: Inference delay acceleration ratio of 4 methods: Single Cutting, Scheduling with Queuing, cloud-only, and edge-only.

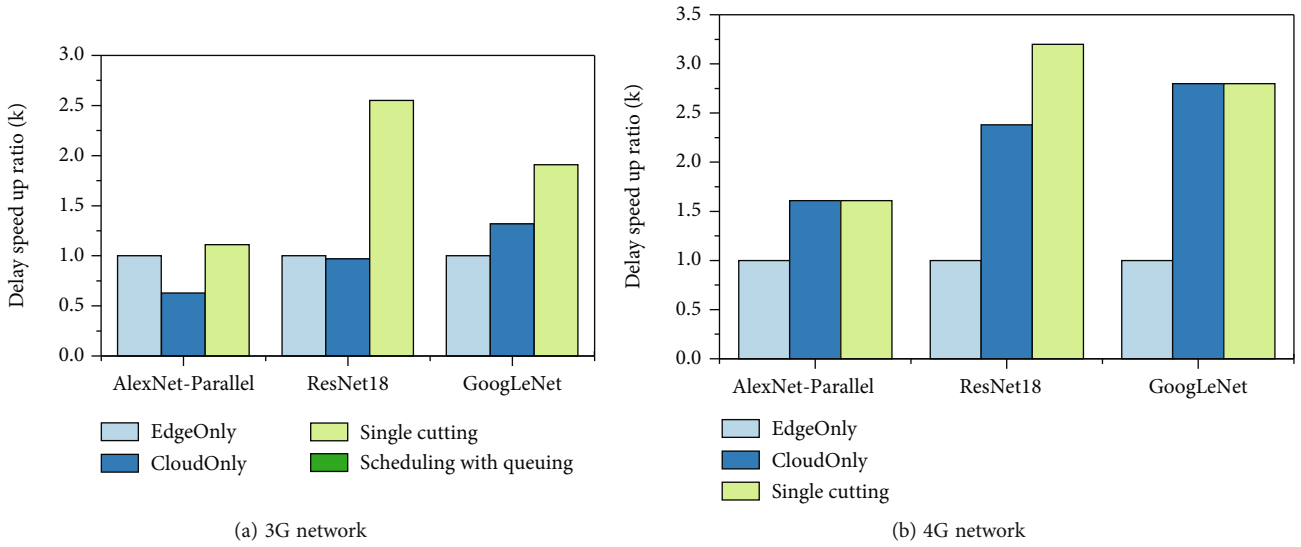
Queuing methods surpass the edge-only and cloud-only methods. In the case of 3G, the efficiency of Single Cutting and Scheduling with Queuing is the same. Compared with the edge-only and cloud-only methods, they achieve a delay acceleration of 2.03 to 3.30 times and 2.22 to 6.48 times, respectively. Figure 2(b) shows the delay acceleration ratios of the four methods in the case of 4G. Compared with only the edge end and only the cloud, the Single Cutting and the Scheduling with Queuing achieve 1.9~5.48 and 1.05~1.57 times acceleration, respectively. In both AlexNet and DarkNet19 models, Scheduling with Queuing achieves the optimal delay speedup ratio. On Tiny-YOLO, Single Cutting and Scheduling with Queuing have the same efficiency.

Figure 3 compares the DAG-type DNN. Figure 3(a) shows the latency acceleration ratios of the three methods in the case of 3G. The cloud-only method performs the worst due to network bandwidth limitations. Compared with the edge-only and cloud-only methods, the Single Cutting achieves 1.11~2.72 times and 2.07~4.12 times delay acceleration, respectively. Figure 3(b) shows the latency acceleration ratios of the three methods in the case of 4G. The cloud-only method surpasses the edge-only method on the three models. On AlexNet-Parallel and GoogLeNet, the efficiency of Single Cutting is the same as that of the cloud. On ResNet18, the efficiency of Single Cutting is 1.34 times higher than that of the cloud alone. Compared with the edge-only method, the Single Cutting achieves a speedup of 1.11~2.72 times.

5.3. The Influence of the Number of Edge Devices on the Inference Delay. In this section, we compare the performance of different methods with different numbers of edge ends. In the same way, we use "edge-only" as the baseline to evaluate the speedup ratio (k) of different methods. We use AlexNet and ResNet18 to verify the effectiveness of our scheduling algorithm under different edge numbers.

As shown in Figure 4, we deployed AlexNet on the edge and compared the speedup ratios of the four algorithms on the chain DNN in the case of 1 to 5 edge ends. Single Cutting and Scheduling with Queuing are better than the edge-only and cloud-only methods under different numbers of edge terminals. Compared with the edge-only method, the Single Cutting can achieve a delay acceleration of 2.88 to 3.27 times, and the Scheduling with Queuing can achieve a delay acceleration of 3.12 to 3.75 times.

As shown in Figure 5, we deploy ResNet18 on the edge and compare the speedup ratios of the three algorithms on DAG-type DNNs in the case of 1 to 5 edge ends. Compared with the edge-only method, scheduling Algorithm 1 can achieve a delay acceleration of 2.94 to 3.20 times. Compared with the edge-only method, scheduling Algorithm 1 can achieve a delay acceleration of 2.94~3.20 times. Compared with the cloud-only method,



(a) 3G network (b) 4G network

Figure 3: Inference delay acceleration ratio of 3 methods: Single Cutting, cloud-only, and edge-only.

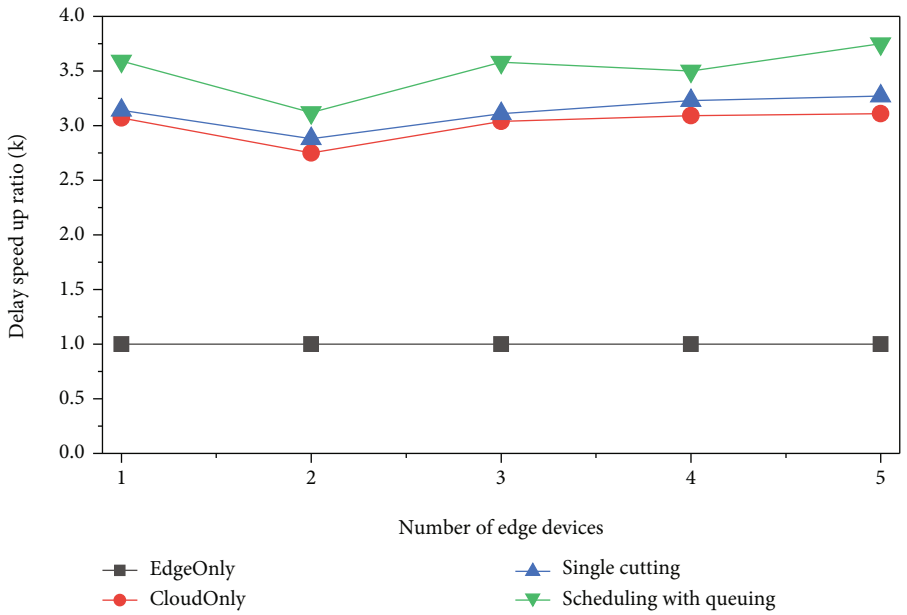


Figure 4: The delay acceleration ratio of AlexNet under different methods under different number of edge devices.

scheduling Algorithm 1 can achieve a delay acceleration of 1.25~1.43 times.

5.4. The Influence of Network Bandwidth on Inference Delay. We compared the impact of network bandwidth on the performance of different methods. We use "edge-only" as the baseline to evaluate the speedup (k) of different methods. During the experiment, we accelerated the network bandwidth from 0 Mbps to 12 Mbps. Similarly, we used AlexNet and ResNet18 to verify the influence of network bandwidth on our scheduling algorithm.

As shown in Figure 6, we deployed AlexNet at the edge and experimented with different methods on the impact of AlexNet's inference delay under different network conditions. When the network condition is 0 Mbps, the cloud-only method performs the worst, and the efficiency of the two scheduling algorithms and the edge-only method is the same. When the network condition is 1 Mbps, Single Cutting and Scheduling with Queuing have the same efficiency, which is better than the two baseline methods. When the network condition is between 2 Mbps and 7 Mbps, the efficiency of Scheduling with Queuing is higher than that of Single Cutting and the edge-only method. When the network condition is greater than 7 Mbps, Single Cutting, Scheduling with Queuing, and the edge-only method have the same efficiency.

As shown in Figure 7, we deployed ResNet18 at the edge and experimented with different methods on the inference

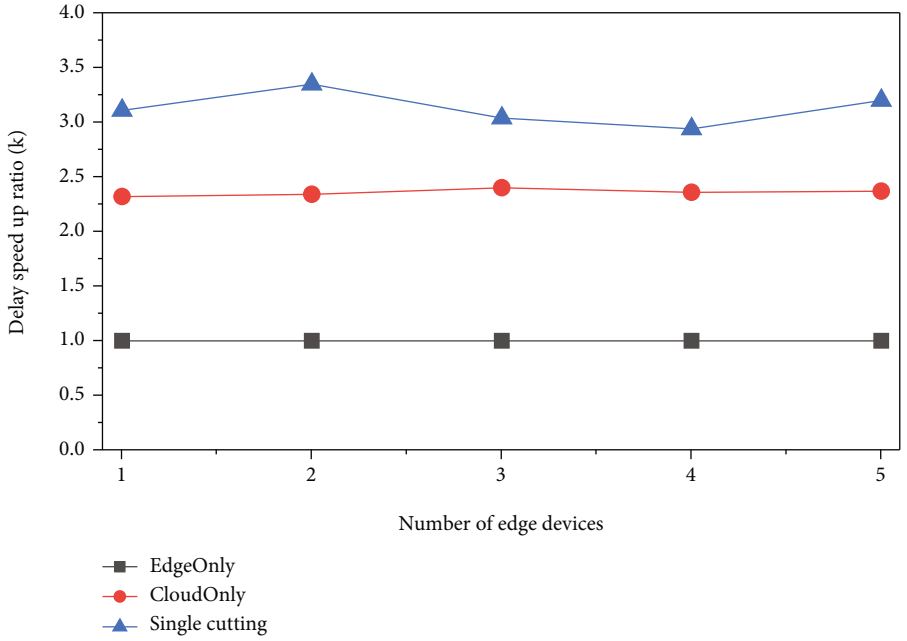


Figure 5: Delay acceleration ratio of ResNet under different methods under different numbers of edge ends.

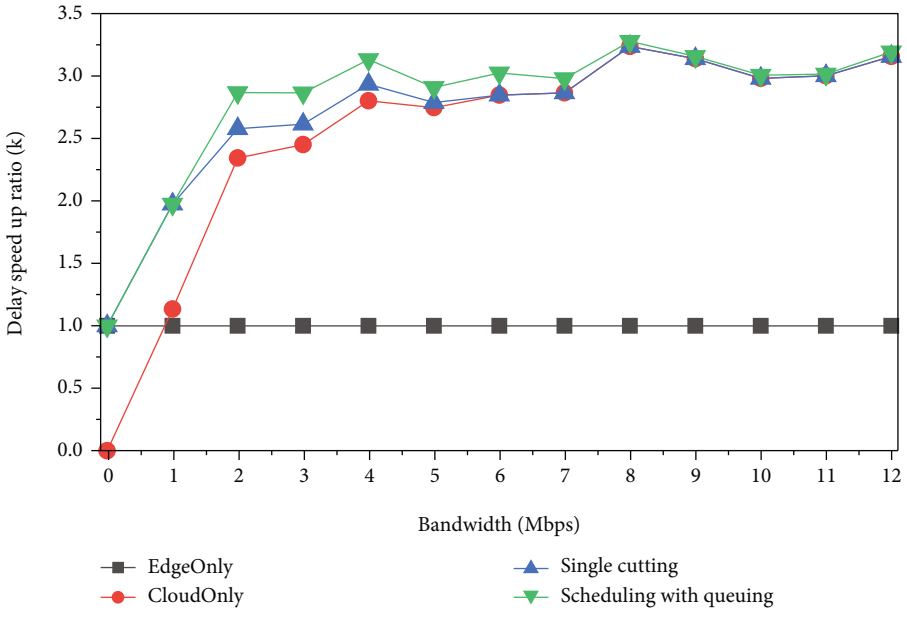


Figure 6: The impact of different methods of bandwidth changes on AlexNet inference delay.

delay of ResNet18 under different network conditions. When the network condition is 0 Mbps, the cloud-only method performs the worst, and the scheduling algorithm is the same as the edge-only method. When the network conditions are between 1 Mbps and 8 Mbps, scheduling Algorithm 1 performs optimally, at most 2.85 times faster than the edge-only method and at most 1.93 times faster than the cloud-only method. When the network speed is greater than 8 Mbps, scheduling Algorithm 1 has the same efficiency as the edge-only method.

6. Discussion

In Section 5, we implement the experiments using five Raspberry Pi platforms to mimic edge devices. According to the experimental results, the proposed scheduling algorithm can handle the circumstances that have more than 5 edge devices. With the increase of the number of edge devices, the number of tasks will be enormous, and the network resources are constrained; the first scheduling algorithm, i.e., Single Cutting, may not be able to find the optimal

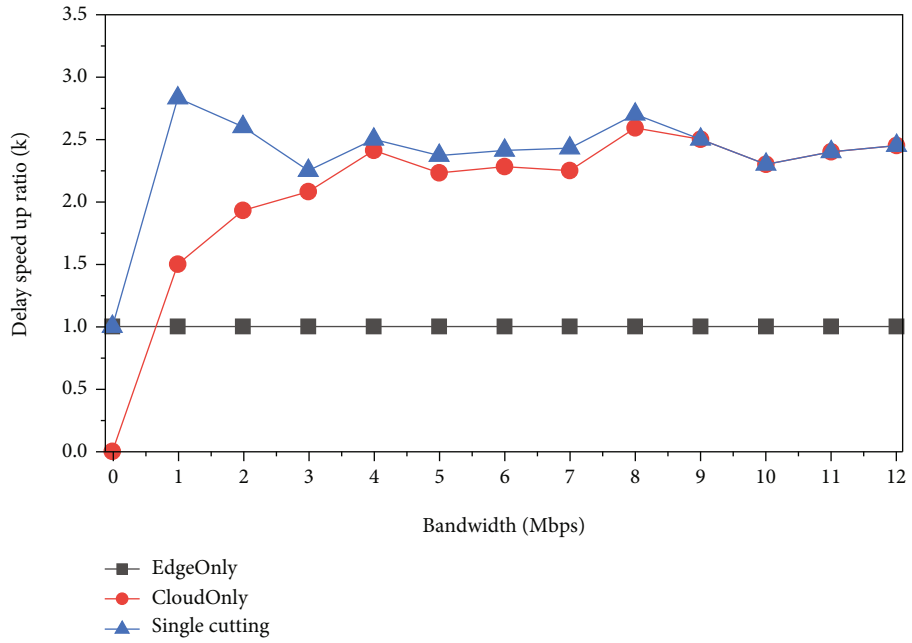


Figure 7: The impact of different methods of bandwidth changes on ResNet inference delay.

cutting point and fail to accelerate the inference, while the second scheduling algorithm, considering the network delay and queuing delay, can still find good cutting point to reduce the overall delay. The impact of cloud server is reflected in the queuing delay in the second scheduling algorithm; the larger the capacity of the cloud server, the less the queuing delay when scheduling. Moreover, different capacity of edge devices and cloud servers will cause different DNN model partitioning in proposed algorithms while the overall delay keeps low.

7. Conclusion

In this paper, we propose two DNN scheduling algorithms for edge-cloud collaborative inference systems. The difference from previous work is that, in the case of multiple edges, we considered partial migration of tasks instead of whole migration. The first scheduling algorithm performs scheduling based on the optimal decision of a single DNN, and each task performs task migration at the same split point. The algorithm combined with the QDMP algorithm can be applied to the scheduling of chain DNN and DAG-type DNN and has a wide range of applicability. The second scheduling algorithm can search for a partition point that can make the overall delay smaller for scheduling based on factors such as the waiting time for tasks in the cloud and the execution interval between tasks. This scheduling algorithm can be used for chain DNN scheduling.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Shigeng Zhang and Yue Zhang are the corresponding authors of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61772559, 61901529, 61902434 and the Natural Science Foundation of Hunan under Grant 2020JJ5776 and 2019JJ50826.

References

[1] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5g beyond for the industrial internet of things," *IEEE Network*, vol. 33, no. 5, pp. 12–19, 2019.

[2] S. Zhang, X. Liu, Y. Liu, B. Ding, S. Guo, and J. Wang, "Accurate respiration monitoring for mobile users with commercial rfid devices," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 513–525, 2020.

[3] Z. Zhou, E. L. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[4] S. Pearson, "Privacy, security and trust in cloud computing," in *Privacy and Security for Cloud Computing*, pp. 3–42, Springer, 2013.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] Y. Kang, J. Hauswald, C. Gao et al., "Neurosurgeon," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[7] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[8] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, "Computation offloading and task scheduling for dnn-based applications in cloud-edge computing," *IEEE Access*, vol. 8, pp. 115537–115547, 2020.

[9] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: automl for model compression and acceleration on mobile devices," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 784–800, Munich, Germany, 2018.

[10] J. Dean, G. S. Corrado, R. Monga et al., "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems*, pp. 1232–1240, Lake Tahoe, NV, USA, December 2012.

[11] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[12] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, April 2018.

[13] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[14] W. Park, D. Kim, L. Yan, and M. Cho, "Relational knowledge distillation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3967–3976, Long Beach, CA, USA, June 2019.

[15] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, http://arxiv.org/abs/1710.09282.

[16] S. Han, X. Liu, H. Mao et al., "EIE: efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[17] W. Jiang, Z. He, S. Zhang et al., "Microrec: accelerating deep recommendation systems to microseconds by hardware and data structure solutions," 2020, http://arxiv.org/abs/2010.05894.

[18] D. Moolchandani, A. Kumar, and S. R. Sarangi, "Accelerating cnn inference on asics: a survey," *Journal of Systems Architecture*, vol. 113, article 101887, 2021.

[19] K. Guo, L. Sui, J. Qiu et al., "From model to fpga: software-hardware co-design for efficient neural network acceleration," in *2016 IEEE Hot Chips 28 Symposium (HCS)*, pp. 1–27, Cupertino, CA, USA, August 2016.

[20] S. Zhang, Y. Li, X. Liu et al., "Towards real-time cooperative deep inference over the cloud and edge end devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–24, 2020.

[21] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: on-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[22] H. Chuang, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1423–1431, Paris, France, April 2019.

[23] L. Hu, G. Sun, and Y. Ren, "Coedge: exploiting the edge-cloud collaboration for faster deep learning," *IEEE Access*, vol. 8, pp. 100533–100541, 2020.

[24] X. Liu, J. Yin, S. Zhang, B. Xiao, and B. Ou, "Time-efficient target tags information collection in large-scale RFID systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 9, pp. 2891–2905, 2021.

[25] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint dnn partition deployment and resource allocation for delay-sensitive deep learning inference in iot," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9241–9254, 2020.

[26] X. Liu, J. Yin, J. Liu, S. Zhang, and B. Xiao, "Time efficient tag searching in large-scale RFID systems: a compact exclusive validation method," *IEEE Transactions on Mobile Computing*, vol. 1, p. 1, 2020.

[27] X. Liu, Q. Yang, J. Luo, B. Ding, and S. Zhang, "An energy-aware offloading framework for edge-augmented mobile rfid systems," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 3994–4004, 2018.

[28] R. Gao, H. Yang, S. Huang et al., "Pripro: towards effective privacy protection on edge-cloud system running dnn inference," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 334–343, Melbourne, Australia, May 2021.

[29] Y. Dong, P. Zhao, H. Yu, C. Zhao, and S. Yang, "CDC: classification driven compression for bandwidth efficient edge-cloud collaborative deep learning," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3378–3384, 2020.

[30] A. G. Howard, M. Zhu, B. Chen et al., *Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, 2017, http://arxiv.org/abs/1704.04861.

[31] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 328–339, Atlanta, GA, USA, June 2017.

[32] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, Cancun, Mexico, December 2016.

[33] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Auckland, New Zealand, November 2018.

[34] P. Ren, X. Qiao, Y. Huang, L. Liu, S. Dustdar, and J. Chen, "Edge-assisted distributed dnn collaborative computing approach for mobile web augmented reality in 5g networks," *IEEE Network*, vol. 34, no. 2, pp. 254–261, 2020.

[35] D. J. Pagliari, R. Chiaro, Y. Chen, E. Macii, and M. Poncino, "Optimal input-dependent edge-cloud partitioning for rnn inference," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 442–445, Genoa, Italy, November 2019.

[36] S. Wang, Y. Li, S. Pang, Q. Lu, S. Wang, and J. Zhao, "A task scheduling strategy in edge-cloud collaborative scenario based

on deadline," *Scientific Programming*, vol. 2020, Article ID 3967847, 9 pages, 2020.

[37] H. Zhou, W. Zhang, C. Wang, X. Ma, and H. Yu, "Bbnet: a novel convolutional neural network structure in edge-cloud collaborative inference," *Sensors*, vol. 21, no. 13, p. 4494, 2021.

[38] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-split: a general framework of collaborative edge-cloud ai," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2543–2553, Singapore, August 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5331712, 15 pages
https://doi.org/10.1155/2022/5331712

Research Article

OffFog: An Approach to Support the Definition of Offloading Policies on Fog Computing

Sávio Melo,[1] **Felipe Oliveira**,[1] **Cícero Silva**,[2] **Paulo Lopes**,[1] **and Gibeon Aquino**[1]

[1]*Department of Informatics and Applied Mathematics (DIMAp), Federal University of Rio Grande do Norte, Natal, 1524 RN, Brazil*
[2]*Federal Institute of Education, Science and Technology of Paraíba, Catolé do Rocha, 227 PB, Brazil*

Correspondence should be addressed to Sávio Melo; saviorennan@ufrn.edu.br

Received 15 July 2021; Revised 25 November 2021; Accepted 8 December 2021; Published 4 January 2022

Academic Editor: Pengfei Wang

IoT devices deployed in Smart Cities usually have significant resource limitations. For this reason, offload tasks or data to other layers such as fog or cloud is regularly adopted to smooth out this issue. Although data offloading is a well-known aspect of fog computing, the specification of offloading policies is still an open issue due to the lack of clear guidelines. Therefore, we propose OffFog—an approach to guide the definition of data offloading policies in the context of fog computing. In order to evaluate OffFog, we extended the well-known simulator *iFogSim* and conducted an experimental study based on an urban surveillance system. The results demonstrated the benefits of implementing data offloading based on OffFog recommended policies. Furthermore, we identified the best configuration involving design decisions such as data compression, data criticality, and storage thresholds. The best configuration produced at least 76% improvement in network latency and 5% in the average execution time compared to the *iFogSim* default strategy. We believe these results represent a significant step towards establishing a systematic decision framework for data offloading policies in the context of fog computing.

1. Introduction

Fog computing plays an essential role in building a sustainable IoT infrastructure for smart cities [1]. Many research efforts have been made on fog computing in smart cities, but there are still several open challenges towards its concrete realization [1, 2]. Moreover, despite Fog computing brings advantages to the development of IoT applications such as latency improvement, new challenges related to storage, and local processing power emerges in the context of Smart Cities [3, 4]. Indeed, IoT devices deployed in Smart Cities usually have significant resource limitations. Therefore, some strategies must be used to mitigate the problems caused by this issue. A usual solution is to transfer the responsibility to entities with more resources, known as offloading. One particular category is the data offloading between the fog and cloud. The need to transfer the data to the cloud emerges in several cases, being very usual when there is the need for long-term storage of data produced on the network border [5].

Data offloading affords several benefits to IoT applications, including energy-saving, storage reduction, decreased network usage, reduced decision-making time, among others. However, the correct execution of this task requires following specific rules (or policies) related to transferring data between persistence entities. These policies should be defined based on criteria such as load balancing, data volume, long-term storage, latency, data management and privacy, among others [5, 6]. Furthermore, they are context-dependent, i.e., they are strongly influenced by the application domain and the characteristics of the fog devices. Thus, defining and implementing effective data offloading strategies in the fog context is a challenging issue.

Therefore, our work proposes an approach to assist the definition of data offloading strategies in fog contexts. This approach is called OffFog and aims to facilitate decisions for selecting data offloading criteria. It consolidates the main concerns related to data offloading in fog contexts and offers proper guidelines in the form of a set of activities aiming at conducting the solution's architect to an effective offloading strategy. To the best of our knowledge, no other previous work provides clear guidelines on how to perform the data offloading process in fog computing-based applications.

As a case study, we use OffFog in the context of an urban security camera application, basing on the scenario proposed by Gupta et al. [7]. This case study attempts to demonstrate the feasibility of using OffFog in applications based on fog computing and, at the same time, to verify the offloading criteria of selected data influence in this scenario. For this, we extended the work of Naas et al. [8], including new offloading capabilities, such as data compression, and performed simulations in order to evaluate the impact of different levels of choice for these criteria.

The main contribution of this paper is threefold. (1) A taxonomy that compiles and structures the knowledge regarding concepts and techniques of data offloading in fog contexts. (2) The OffFog approach aiming to guide the definition of data offloading policies in the context of fog computing. (3) The extension of the well-known simulator *iFogSim* [7], including new data offloading capabilities. We believe these results represent a significant step towards establishing a systematic decision framework for data offloading policies in the context of fog computing.

The remainder of this article is organized as follows: Section 2 describes the related works. Section 3 presents the background related to data offloading on Fog computing. Section 4 describes the OffFog approach. A case study is described in Section 5, aiming at demonstrating the execution of OffFog in a scenario of a Surveillance Camera System (SCS). Section 6 presents the results of an experimental study involving simulations of the SCS scenario. Finally, Section 7 discusses the concluding remarks.

2. Related Works

Studies related to this work can be divided into two groups: studies dealing with data offloading in fog computing (Section 2.1); solutions that enable the simulation of applications based on fog computing (Section 2.2).

2.1. Data Offloading. According to Farahani et al. [9], Silva [10], Silva and Aquino Jr. [11], there are several challenges related to managing stored data in the fog layer. Furthermore, the storage capacity on Fog is usually limited when compared to Cloud [3, 11]. Thus, several works propose strategies to deal with these issues.

Silva et al. [12] propose a fog computing-based architecture to manage patients' medical data. In this architecture, the authors define a component particularly designed to store metadata related to medical records kept in the fog. Thus, this metadata helps decide which data to free or keep on the fog nodes. However, the proposed solution defines a generic behaviour without specify the offloading process's concrete rules (or policies).

Alelaiwi [13] and Pisani et al. [14] use the data offloading process to meet the requirements of their applications better. Thus, these authors consider strategies and policies that set the course for such a process, serving to mitigate the limitations of the fog environment. On the other hand, considering a dynamic fog computing network, Shan et al. [15] use the heterogeneity of sensor nodes as contextual information to promote the data offloading between sensors and fog nodes while improving the algorithm for stable performance in urban areas environments.

The data offloading process in fog computing is a topic of constant discussion. Alelaiwi [13] uses machine learning algorithms that select the data to be sent to the cloud. In this approach, the offloading of edge data is only performed after carrying out deep learning techniques. Pisani et al. [14] consider data that needs to be balanced across fog nodes to data offloading according to available storage. However, these studies do not discuss criteria related to the volume and nature of the data.

Aazam et al. [5] classify the types of offloading that exist, defining different situations where the process can occur and showing its variations. He et al. [16] consider that security and load balancing aspects are significant in the data offloading process. However, neither of the two works relates the properties of data and infrastructure.

Still dealing with security, He et al. [16] propose a security model for storing fog data, highlighting the possible vulnerabilities that this data may be subject to. Verma et al. [17] propose a processing balancing model adaptable to the context of fog data. However, the authors do not specify the techniques used in the processes and their possible impacts.

Mukherjee et al. [18] present a fog data offloading strategy to minimize energy consumption and the total delay related to the processing task for the end-user. The work formulates an optimization problem and provides a solution through the CVX tools, a modelling system based on MATLAB for convex optimization [18]. The simulation results demonstrated that the proposed solution offers minimal end-to-end latency than running all tasks on end-user devices and running all tasks on primary fog nodes. However, the study does not discuss criteria related to data volume and nature.

Zhu et al. [19] propose task offloading policy to help decide whether to process tasks locally and send them to the appropriate fog node or cloud. This approach considers calculating the execution time and energy consumption when running the task on the device compared to the execution time and energy consumed when transferring and receiving the processed task at the appropriate fog node. Dinh et al. [20] propose an offloading framework to optimize the task allocation decision and the allocation of computational resources.

In summary, several works in the literature propose solutions for data offloading in fog, which addresses different problems. The work of Bali et al. [21], for instance, provides a systematic literature review (SLR) where approaches on data offloading in IoT networks at edge and fog nodes can be found. In that same work, the authors propose an intelligent architecture for data offloading and propose a data offloading workflow for industrial IoT. However, there are no clear directions regarding the selection criteria to assist data offloading decision making. Therefore, unlike other works, this paper classifies the criteria of the data offloading process to guide in defining policies to enable the execution of data offloading in fog-based solutions.

2.2. Simulation of Fog Computing Environments. Some tools simulate fog computing environments. Margariti et al.'s [22]

work investigates the simulation tools used to develop fog-based solutions. Among the simulation tools investigated, *iFogSim* stands out. *iFogSim* (https://github.com/cloudslab/ifogsim) is currently the most used tool to simulate fog computing-based environments. Described in Gupta et al. [7], this simulator implements resource management techniques on fog computing environments in order to measure latency and throughput.

Some *iFogSim* extensions were implemented, aiming to add additional features to this simulator. Lopes et al. [23] developed *MyiFogSim* (http://www.lrc.ic.unicamp.br/fogcomputing), an extension of *iFogSim* to support the mobility requirement of fog computing-based applications. In addition, Naas et al. [24] extended *iFogSim* to simulate scenarios with strategies aimed at optimizing data placement, called *iFogSimWithDataPlacement* (https://github.com/medislam/iFogSimWithDataPlacement).

An important extension of *iFogSim* that allows simulating environments with the requirement of mobility is the so-called *MobFogSim*. Proposed by Puliafito et al. [25], this extension allows overcoming the limitations of environments that require mobility. It also extends *iFogSim* to allow device mobility modelling. However, its distinguishing feature is support for service migration in fog computing.

Another well-known simulation environment for fog computing is the Edge-Fog cloud simulator [26]. This simulator is a Python implementation consisting of two layers: an outer layer consisting of the edge devices and an inner layer consisting of the fog devices. Furthermore, this implementation focuses on implementing the Least Processing Cost First (LPCF) algorithm, which reduces processing time and network costs.

Overall, none of the implementations mentioned above brings the possibility of simulating data offloading policies. Thus, *iFogSimWithDataPlacement* was extended in our work, generating a new implementation called *iFogSimOffload* (https://github.com/labcomu/iFogSimOffload), which differs from the others by providing some techniques related to the data offloading processes, such as compression and data criticality.

3. Data Offloading in Fog

The offloading process is responsible for performing migration of data or processing tasks, helping devices with limited computational capabilities [27]. Regarding data offloading, this is a technique that has been widely discussed in the fog computing paradigm [5]. Furthermore, it is generally applied due to the resource constraints of the entities that make up the fog. Data offloading is controlled through rules, which indicate how data is managed during this process. Finally, these rules are commonly called policies.

To organize the essential concepts involved in the data offloading process, we defined taxonomy. In particular, it compiles essential aspects for data offloading policies in the fog contexts. Moreover, the taxonomy describes the main properties to be considered when designing data offloading policies for the most diverse fog scenarios, considering the characteristic of the data and the infrastructure of the environments involved. Therefore, the section classifies some concepts about the data offloading process and exemplifies the main techniques used in such a process. Thus, this classification can serve as a basis for selecting techniques to define data offloading policies since it provides many of the concepts necessary for this definition.

3.1. Taxonomy. In our taxonomy, the criteria were divided into two main views: data and environment. Therefore, the classes and subclasses that compose the taxonomy are distributed in these views.

The Data View brings together the characteristics directly related to data generation, transmission, and processing in a fog computing environment. Thus, it is formed by the Security, Management, Nature, and Volume classes. On the other hand, the Environment View encompasses the physical characteristics related to the data offloading process in a fog environment. This view is related to the requirements of the environment in which the data resides, that is, the physical properties of the fog nodes. Therefore, it is formed only by the Infrastructure class.

Figure 1 shows the taxonomy proposed by Melo et al. [6]. It summarizes the classes that help design policies for data offloading in fog computing. Therefore, these classes will be detailed below.

3.1.1. Security. The use of inappropriate approaches creates many security problems in the nodes that form the fog [28]. However, applying the four pillars of information security (confidentiality, availability, integrity, and privacy) guarantees the security of information transferred in the fog.

As far as confidentiality is concerned, the application of authentication protocols guarantees the identity of the fog connected nodes [29]. In addition, the use of this type of protocol can prevent unauthorized people from accessing the data transmitted between these nodes. On the other hand, technologies that promote data availability must be applied to ensure that fog nodes or cloud servers are always available [29]. The use of techniques such as data replication and load balancing can help achieve this goal.

In the same context, data integrity ensures the detection of any unauthorized attempts to modify them. In fog computing, Blockchain technology is often used to avoid this type of problem [30]. The privacy attribute is intended to protect sensitive personal information [31]. To meet this requirement, [32] propose a Blockchain-based method to ensure the patient's medical data privacy in the fog computing environment. In this perspective, it is common to use Blockchain technology to provide the data anonymization feature.

3.1.2. Management. Transferring data from one device to another is a common task in fog computing. Therefore, this process needs to be well managed [5]. The primary motivation to consider this aspect in the data offloading process is that fog nodes are geographically distributed, making it challenging to identify the location of the data. In addition, several techniques can be used during data management, such as optimization and replication. Thus, using these

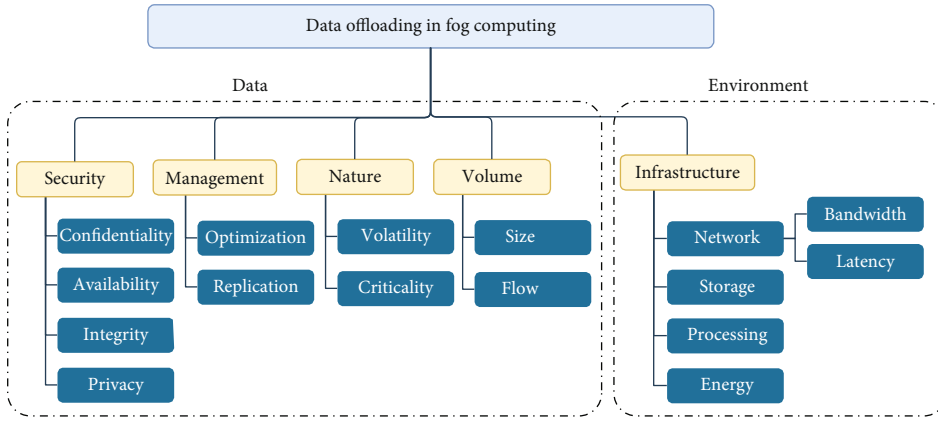


Figure 1: Taxonomy for data offloading in fog computing.

techniques will help distribute the data between the fog nodes more efficiently since the requested data may not be available in nearby nodes.

About data optimization, this process is directly related to the storage capacity and processing constraints of a fog node [33]. According to Rahmani et al. [34], two optimization techniques stand out in the fog computing paradigm: data filtering and compression.

3.1.3. Nature. Generally, environments in which fog computing is applied do not use their hardware or communication technologies as a solution for nonvolatile data storage [35]. On the other hand, most resources (storage and network) are provided by devices near the edge of the network, such as smartphones, smart TVs, or vehicles. Thus, this paradigm faces a challenge linked to the resources provided by its nodes, as they may at some point become unavailable for access [36].

In addition, many applications make use of fog computing to meet their real-time operational needs. At the same time, large-scale computing infrastructures are increasingly dealing with critical data, which is also stored in cloud databases [37]. Therefore, when operating/transmitting the data generated by the fog layer device, it is necessary to consider its different degrees of criticality. For example, critical data may be found in the scenario of monitoring patient health information. In contrast, some other data in the same solution may be less relevant and volatile.

3.1.4. Volume. The volume class is related to two aspects: data size and flow [38]. These two aspects are the most mentioned in the literature and the most relevant when the data analysis is done in the nodes that make up the fog or in external entities. In this context, analyzing these two factors in this paradigm allows data to be transferred to the cloud only when necessary.

The size aspect is related to the amount of data that needs to be processed and stored in the fog. Thus, many researchers point to this feature as one of the essential requirements for meeting requests [39]. In addition, this aspect must be considered to mitigate the storage limitation characteristic of the nodes that make up the fog. Concerning data flow, this aspect is related to the intensity at which data is sent from a source to a fog node. Thus, the flow can occur in two ways: event-driven [40] or real-time [41]. In real-time, streaming data is streamed into the fog immediately after generation, resulting in nodes being overloaded. Therefore, understanding the size and flow of data in the fog during the data offloading process is essential to keep the system running smoothly.

3.1.5. Infrastructure. The fog is composed of several heterogeneous devices, which use different communication protocols. Furthermore, the nodes that make up the fog have several hardware limitations. However, with the knowledge of the limitations of the fog computing environment, the cloud can contribute with its capabilities and resources. Therefore, the process of offloading data from the fog nodes to the cloud makes it possible to use the capabilities of this last paradigm.

The storage in the fog computing paradigm is a topic widely discussed in the literature [3, 11]. It is known that data storage in fog nodes is limited, even with the application of data management techniques and the elimination of unnecessary transactions. Furthermore, this paradigm also applies techniques related to the processing aspect, which aim to reduce the amount of data sent to the cloud [42]. On the other hand, applying these techniques can overload the infrastructure that makes up the fog. It is noteworthy that in the fog, network devices constitute the nodes in the fog, and they are mostly battery powered. Therefore, the application of data compression and processing techniques, while helping to alleviate the restrictions of the fog environment, can overload and worsen the energy autonomy of this paradigm's infrastructure.

In summary, the assessment of infrastructure resources is a requirement that contributes when designing policies for data offloading in the fog. Such resource availability analysis will indicate whether the data flow will proceed normally or whether data offloading will occur [43].

4. OffFog Approach

In order to guide the designing of data offloading policies in the context of fog, we propose OffFog. It is an approach that describes essential activities for defining data offloading policies from fog to cloud layers. This approach guides the

Table 1: Example of a security-focused policy.

Requirements	Objective	Criteria	Techniques
Data	Security	Confidentiality	Authentication and encryption
		Availability	Replication and load balancing
		Integrity	Blockchain
		Privacy	Anonymization

selection of valuable criteria for data offloading decisions and suggests techniques that support the fulfilment of these criteria in fog computing environments.

Policies play an essential role in the data offloading process. They define how the data will be managed during the offloading process, which security procedures will be necessary, when the process should be executed, which techniques will be used, among other factors. For better understanding, Table 1 presents an example of the basic structure of a data offloading policy. In this example, the policy meets data requirements, has a security-oriented objective, and is based on data offloading criteria given the haze system problems. Finally, the techniques to be used as decision aids in the data offloading process are defined.

Policies are critical for scenarios that require the efficient execution of the data offloading process. The efficient execution of this process brings some benefits to fog computing solutions. Among the benefits, some are remarkable, such as network usage decrease [44], latency reduction [45], energy efficiency [44], and privacy [32, 46]. In this context, a guidance mechanism is required for a taxonomy to be instantiated and used to support policy definition. Therefore, this section presents a methodology to guide the definition of policies for data offloading in fog computing. Therefore, the following activities define actions to be done in the policy design phase.

The purpose of the activities is to ensure that the policy definition accommodates the main offloading criteria classified by the taxonomy. To this end, the guide proposed here seeks to provide an initial methodology that consists of five (5) activities, as presented by Figure 2.

The activity #1 has as the main action understanding the system requirements. The requirement elicitation of the fog system and other software involved is essential to obtain knowledge of the characteristics of the data and the organization's infrastructure. That said, this knowledge serves to support the understanding of the general business requirements and the criteria that influence the offloading process.

As a support for the survey of these requirements, the IoT system requirement engineering support framework proposed by Silva et al. [47] was adopted. The tool was adapted to the fog computing context to consider the requirements of the fog system and the software involved in its solutions.

In this scope, these requirements must be elicited and analyzed to know the system behaviour and the interactions between the components that can help in the offloading decision. In order to provide a tool for this elicitation, we designed a template that shows how this requirements elicitation can be done practically. In summary, based on the Silva et al. [47] method, the suggestion is that the template to identify requirements for fog systems should have an identity field, a description, and a field to define the degree of priority of the requirement under specification.

At the end of the first activity, the architect designing the data offloading policies will have requirements which serve as the foundation for the offloading decisions required by the following activities.

The purpose of activity #2 is to identify the possible data offloading problems. Before defining a policy, it is also necessary to identify the problems that will influence the data offloading process.

There are two situations that must be considered in this activity. The first is the identification of data offloading problems considering that the policy design is for a new system and the second keeping in mind the policy design is for existing systems. For the case of new systems, the problem identification process will be done based on requirements. On the other hand, when the policy design is for existing systems, the problem identification process flow should also consider existing policies and possible problems. In this sense, to assist in identifying possible problems arising from requirements and policies, Table 2 provides a grouping of requirements and relates them to common problems of the fog environment when there is a need for data offloading.

At the end of the second activity, the architect designing the policies will have a list of common problems in fog computing systems. Therefore, the execution of this activity adds to the policy design stage the identification of the main problems that will motivate the definition of the objectives of the data offloading policies.

Activity #3 will serve to define the policy objectives. The definition of these objectives is a central action in the design of policies for data offloading. Therefore, this activity helps the architect to identify the main criteria that may impact the system's cloud data offloading process.

Table 3 guides this identification and organizes the criteria in the data view, i.e., considering the characteristics of the data trafficked in the fog. This support tool will serve to guide the architect to define the system's data offloading criteria and consequently define its objectives. According to Tables 3 and 4 by answering some questions about the system based on its requirements, it is possible to identify which data offloading criteria are related. Therefore, as these criteria refer to a certain objective, at the end of the analysis of the questions, we have two definitions: criteria and objectives.

Similarly, the environment view perspective (Table 4), i.e., helps in the identification of data offloading criteria

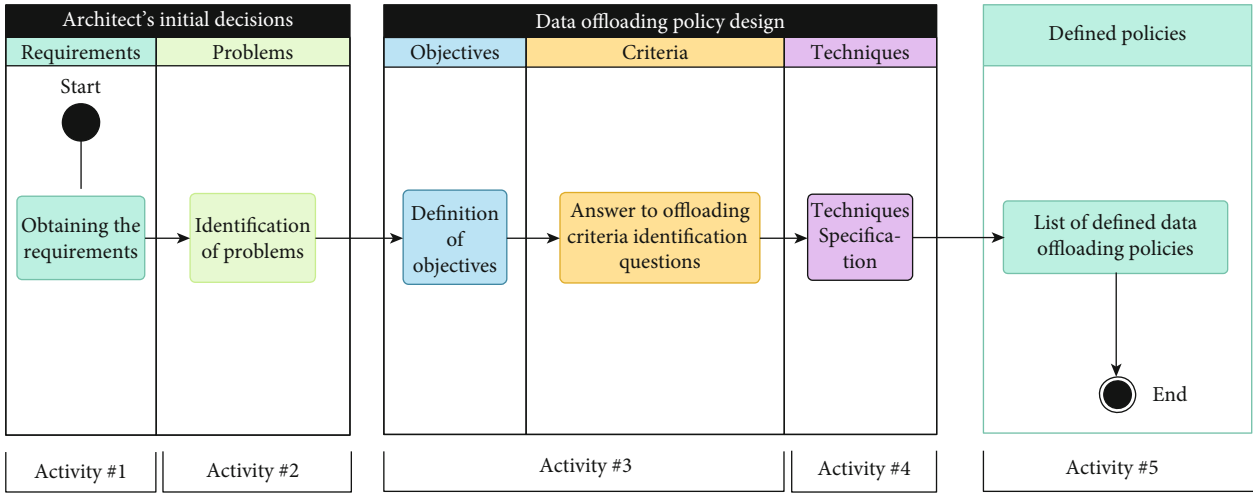


Figure 2: Activities for defining data offloading policies.

Table 2: Relationship between requirements and most common problems.

Requirements	Problems
Data communication	(i) Data redundancy compromises fog node storage [17] (ii) Continuous flow of sensor data compromises storage components [41] (iii) Local data processing compromises processing components [42] (iv) Loss or modification of data [48]
Security	(i) Incorrect approaches [49] (ii) Data theft by malicious users [28] (iii) Display of sensitive personal information [50]
Availability	(i) Loss of critical data [51]
Communication between systems	(i) Network unavailability [52] (ii) Power unavailability [53]

Table 3: Identifying data offloading criteria in the data view.

ID	Questions	Criteria	Objectives
Q1	Does the system handle sensitive or confidential data?	Confidentiality	Security
Q2	Is the data accessed frequently?	Availability	
Q3	Does the loss or modification of part of the data cause any harm?	Integrity	
Q4	Should you control data sharing?	Privacy	
Q5	Is the data prefiltered and analyzed beforehand?	Optimization	Management
Q6	Is there data redundancy?	Replication	
Q7	Is there a need for long-term data storage?	Volatility	Nature
Q8	Can data availability tolerate delays?	Criticality	
Q9	Is there a need to reduce the space occupied by data on a particular device?	Size	Volume
Q10	Is there simultaneity between data generation and data storage?	Flow	

based on the characteristics of the organization's infrastructure. It is important to note that the choice of objectives and criteria for offloading of the policy under development will occur from the favorable response to each question. Thus, each positive answer to the questions in relation to the system requirements will define a criterion to be considered in the policy objective that is being designed in the infrastructure view.

As a result of this activity, based on the answers to the criteria identification questions, the architect will have the definition of their policy objective as a result. Therefore, performing this activity adds to the policy design

Table 4: Identifying data offloading criteria in the environment view.

ID	Questions	Criteria	Objectives
Q11	Is the system sensitive to the latency of requests?	Network	Infrastructure
Q12	Does the infrastructure of the scenario have its own storage resources?	Storage	
Q13	Does the system have favorable energy autonomy?	Energy	
Q14	Do the fog devices have good processing power?	Processing	

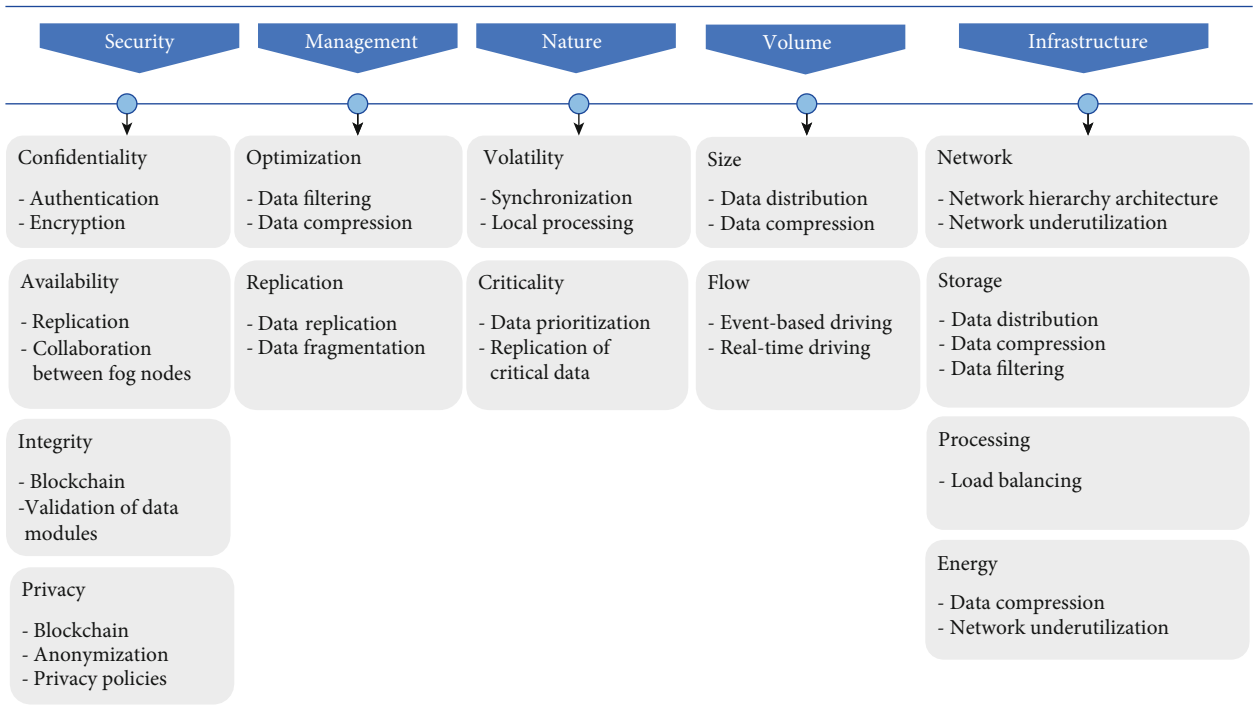


Figure 3: Data offloading criteria and techniques.

phase the ability to establish a relationship between the actual system needs and requirements and the policy objectives.

However, it is still necessary to specify the techniques related to each data offloading criterion for the organized requirements, problems, and criteria to support policy design. With this in mind, the actions of this activity #4 will be focused on selecting methods and techniques to support the data offloading process. This selection will be made based on the highest priority requirements. Therefore, this activity will specify the techniques that will be part of the system's data offloading policies, usually considering the highest priority problems.

Initially, this activity strives to understand the criteria of each policy data offloading objective. With this in mind, the next step is to specify which techniques are essential in executing the fog-cloud data offloading process. Figure 3 simply lists some techniques that were discovered through the study organized by the data offloading taxonomy present in Section 3. By looking at it, it is possible to highlight the most commonly used techniques in each data offloading criterion, which will make it easier to define the actions of the policy being defined.

Activity #5 assists in analyzing the output of the process specified here, that is, the list of policies defined for the solution. The main goal of this activity is to select the conflicting policies. The action defined by this activity is necessary due to the possibility of fog systems having more than one data offloading policy that conflict. In this activity, one must consider that policies can result in different solutions, benefiting or worsening the data offloading in the system. Therefore, it is a crucial decision to be made.

When there are conflicts between policies, the architect should look for alternatives based on his experience and knowledge of the system requirements. Activity #1 can support the decisions regarding requirements, which helps gather requirements based on priority level. So an alternative is to consider this priority level to select which policy is most relevant for the system.

In summary, within the activities required to define data offloading policies, many actions are performed. The first action is formed by the architect's decisions about the requirements and problems addressed in the policies. After these decisions are made, the definition of policies begins. Some questions are raised to identify the criteria that will motivate the data offloading process with the objective

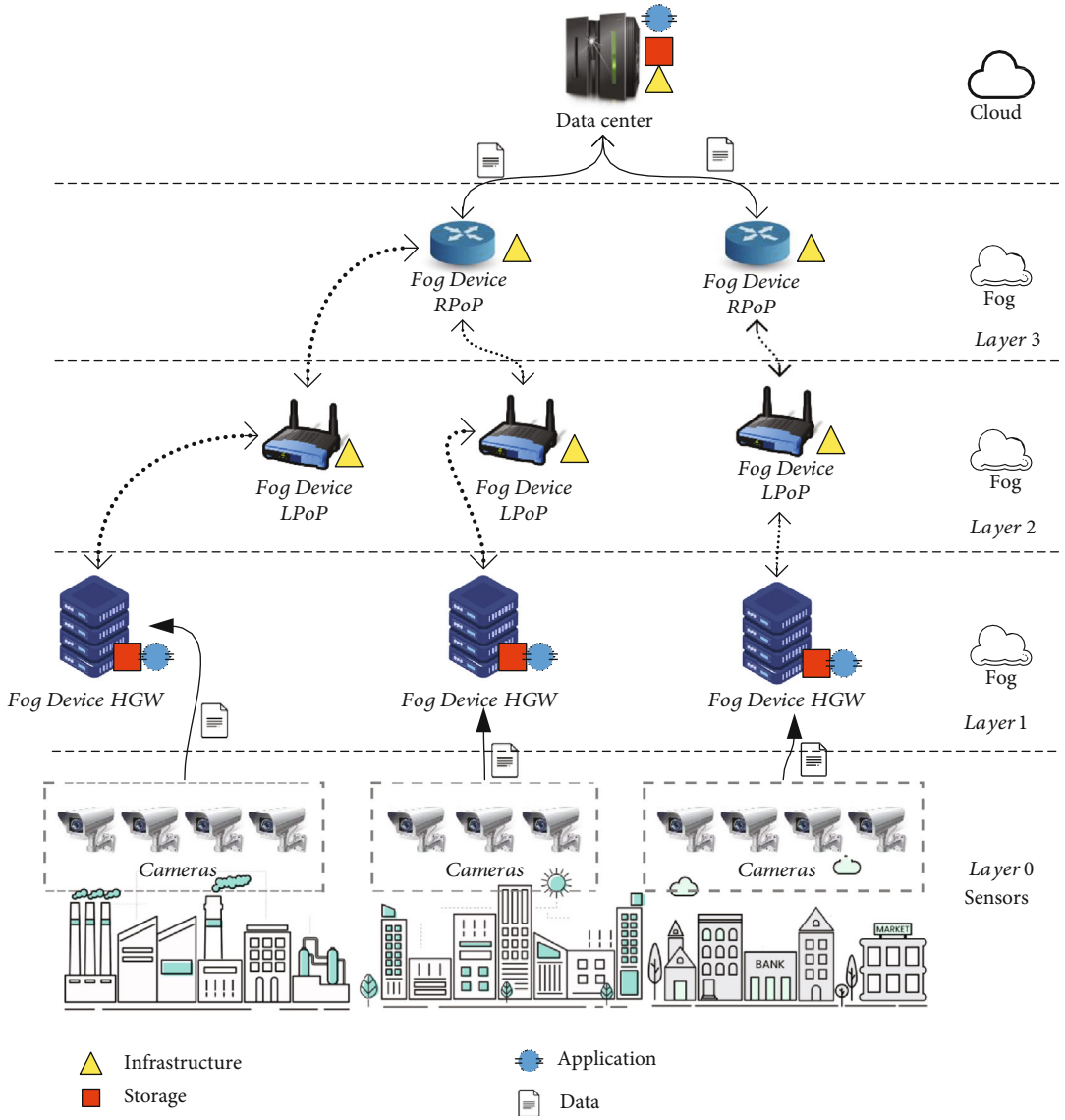


Figure 4: System architecture (adapted from Naas et al. [24]).

defined. Then, after the questions are answered, the techniques used to solve the solution's offloading problem are specified. Finally, there is an output with a policy for each identified problem, which allows decision-making to be performed on conflicts of choice between policies.

5. Case Study

In order to demonstrate the practical use of the OffFog approach, we applied it in an urban security application, particularly in a surveillance camera system (SCS). The study is based on the scenario proposed by Gupta et al. [7] but also includes requirements related to high-scale IoT infrastructure for smart cities. This case study attempts to demonstrate the feasibility of using OffFog in fog computing applications.

Intelligent cameras send a detection alert and video stream to a designated controller upon detecting motion in their field of view. As a result, SCS must handle motion detection alerts as quickly as perceived by multiple sensors. Furthermore, a long-term data storage capacity from the systems is required, driven by the nature of several uses for captured images. For that, SCS demands low latency communication to report activities that are currently taking place and also synchronise with other cameras. Therefore, a cloud-based centralised system scenario suffers from a disadvantage because all sensors must forward data to the cloud to handle all processing, causing higher network latency and bandwidth consumption.

For our analysis, we considered a distributed architecture for the SCS, where cameras are responsible for monitoring public places based on the scenario proposed by Hong et al. [54]. Camera sensors detect activities in the context and send the video stream to intermediary devices previously configured. In turn, these devices perform tasks like data compression, partial store, data forward, and data criticality.

Table 5: Data offloading policies for the camera surveillance system.

Policy ID	Requirements	Objectives	Criteria	Techniques
P1	Long-term image storage	Nature	Volatility	Storing video surveillance images in the cloud
	Video stream management for emergency response		Criticality	Storing data in fog nodes
P2	Real-time data flow management	Volume	Flow	Event-driven data flow for noncritical data
	Camera file data size management		Size	Data compression in fog nodes
P3	Object and motion detection	Infrastructure	Storage	Storing data in fog nodes
	Security systems with decision making without delay		Latency	

5.1. Surveillance Camera System Architecture. Devices LPoP and RPoP organisationally cover infrastructure nodes disposal in the local or regional network. They are responsible for the communication between devices. Data collected by the sensors (cameras) are stored and disposed of on devices installed on fog (HGW) and cloud (DC) layers. Figure 4 shows the arrangement of components present in the SCS architecture.

Intermediate devices HGW are responsible for receiving data captured by sensors or transmitted from another device via LPoP node. Also, they can serve data to system applications. Furthermore, they operate under established data offloading policies. For this, they send data to other HGW devices or to the cloud, according to resource availability, such as storage capacity and data relevance. A set of sensors strictly communicates with a node in the fog set on the topology definition. Sensors are responsible for inserting data obtained and its properties such as size, compression ratio, and relevance in the simulated scenario.

5.2. Data Offloading Policies for the SCS. The data offloading policies for the SCS scenario accord to the activities specified by OffFog. Table 5 presents the structure of this design and highlights the three policies defined. According to OffFog guidelines, the policies were organized considering the priorities of the scenario requirements, their objectives, the definition of data offloading criteria, and the specification of techniques that support the offloading decision.

According to Table 5, policy P1 defines that the nature of the data in this scenario should be considered. Data offloading criteria such as data volatility and criticality must be considered to meet two system requirements. The first requirement concerns the need for long-term persistence of the monitoring images. For this requirement, the suggested technique is to store nonvolatile data in the cloud. The second requirement is the management and analysis of the video streams for emergency service response. For this requirement, the suggested technique is to store data in the nearest fog nodes to decrease the latency for decision making in emergencies.

The P2 policy aims to control the volume of data traffic in the system. The system requirements that motivate this policy are the need to manage the data flow in real-time and manage the size of the video files captured by the camera. For the data flow control requirement, the suggested technique is task scheduling (for noncritical data), i.e., data offloading will only occur after some conditional factor, such as when the storage reaches its total capacity. Similarly, the suggested technique is data compression for the data size requirement, aiming to decrease the disk space occupied by the monitoring videos.

Regarding the P3 policy, the focus is to consider the management impact of some resources of the environment infrastructure on the data offloading process. To this end, the first requirement to be considered concerns the storage of object detection and motion data made by the cameras. For this requirement, the suggested technique is to store it in fog nodes near the device, reducing communication latency and allowing the system to recognise objects and their movements faster. Finally, emergency response systems should work without delays that impact their operation, so the suggestion is to prioritise the storage in fog nodes.

6. Experimental Analysis

Aiming to evaluate the resulting policies of OffFog adoption on the scenario presented in Section 5, we executed an experimental study based on the well-known framework *iFogSim* [7]. In particular, we provide a fog computing simulation environment to reproduce the SCS behaviour following the resulting offloading policies from the OffFog execution.

Furthermore, we seek to provide insights and also evaluate the trade-offs related to the resulting policies. We evaluated from a latency-sensitive perspective, based on the Naas et al. [24] architecture. We performed several simulations using different values to relevant attributes for the offloading techniques resulting from OffFog (device storage capacity, level of data compression, and data relevance). We offer a topology for this use case, including cloud devices, fog devices, and sensors.

6.1. Implementation. Despite the *iFogSim* allows the elaboration of complex and heterogeneous evaluation scenarios, as pointed out by Naas et al. [8], the base framework lacks good data abstraction. The Tuple class represents a data fragment and elementary information, but it does not elaborate on storage abstractions. In their study, Naas et al. [8] introduced improvements to the base framework to evaluate different aspects of data placement, i.e., how to better distribute the data, using different strategies, thus, reducing overall latency.

We extended the *iFogSimWithDataPlacement* (https://github.com/medislam/iFogSimWithDataPlacement) by including new capabilities related to data offloading. Figure 5

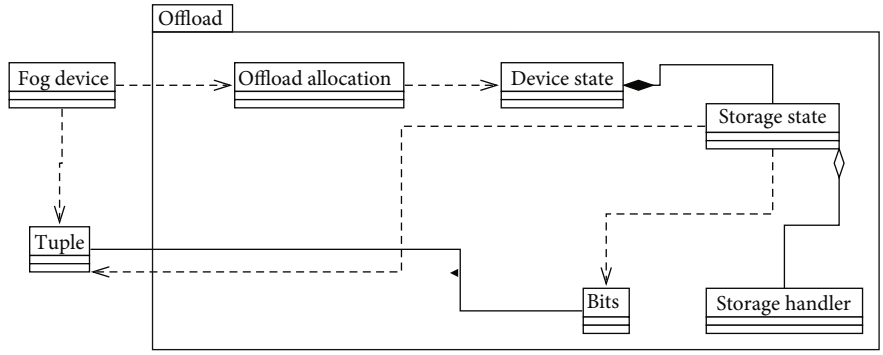


Figure 5: Data offloading implementation components.

Table 6: Data offloading simulation parameters.

Parameter	Description	Values
<*DV*>_TUPLE_FILE_SIZE [a]	Data fragment size	102400 (100 KB)
<*DV*>_Storage [a]	Storage capacity	1048576 (1 MB)
<*DV*>_Storage_Min_Threshold [a]	Storage lower-level threshold percentage	10, 20
<*DV*>_Storage_Max_Threshold [a]	Storage upper-level threshold percentage	20, 30, 40
<*DV*>_Storage_Compression [a]	Data fragment compress level	20, 40, 60
<*DV*>_Critical_Selection [a]	Probability of data fragment CRITICAL	0.3, 0.5, 0.7
<*DV*>_Compression_Selection [a]	Probability of data fragment COMPRESS	0.3, 0.5, 0.7

[a] <*DV*> represents the infrastructure device type (DC, RPoP, LPoP, or HGW).

describes the main components used by our extension. The OffloadAllocation acts as the data offloading entry point, reasoning which host should store the given data. It answers whether it should be kept in the current host or be offloaded to a corresponding node, i.e., to the Cloud (other nodes in the infrastructure may be considered, but it was still not implemented in this version). Other essential components are DeviceState and StorageState, which abstract a device and its current storage state. These components keep track of the stored data and the storage free space, acting as the building blocks to the implementation of data offloading policies. All resources, including source codes, are available at *iFogSimOffload* on Github (https://github.com/labcomu/iFogSimOffload).

Once a new Tuple (data fragment) arrives at OffloadAllocation, it is wrapped with data headers to support the data offloading policies. The Bit component represents these headers. Currently, it provides the options of COMPRESS and CRITICAL. These options indicate if the StorageState should compress the fragment size and if the data fragment should never be offloaded, respectively. These characteristics are selected randomly according to the simulation parameters.

Requests to OffloadAllocation initiate a store attempt into the device storage—performed on StorageState. First, if the COMPRESS bit is activated, it compresses according to the compress level passed as a simulation parameter. Then, it tries to store it, failing when there is no space left on the device. If it succeeds, but the currently used space reaches an upper-level threshold, the store operation initiates data offloading to the corresponding node (for instance, Cloud node). The StorageState starts moving data to the corresponding node, halting when the used space reaches a lower-level threshold. The StorageHandler component handles data offloading and halts logic.

Each simulation takes into consideration parameters that influence the offloading behaviour, as detailed in Table 6. These parameters are configured on the DataPlacement component, which initiates the framework execution. The <*DV*>_TUPLE_FILE_SIZE parameter is closely related to the corresponding <*DV*>_Storage parameter and should be defined in conjunction. Otherwise, the data offloading operation might not be triggered. In other words, if a storage free space is too large compared to the Tuple data fragment size, the used space will remain under the upper threshold—hence, data offloading will not be required. The <*DV*>_Compression_Selection and <*DV*>_Critical_Selection parameters define the probabilities where a data fragment has the COMPRESS and CRITICAL bits selected, respectively. For instance, if we want to select only 20% of data fragments arriving at HGW devices as critical, we should define HGW_Critical_Selection to 0.2.

6.2. Experimental Setup. All simulations were performed on an Intel Core i7 4 CPU 1.8 GHz with 16 GB of RAM. Since the *iFogSim* base framework does not benefit from parallel processing and uses only up to 3 GB of RAM, the simulation duration is only influenced by the unitary core frequency (1.8 GHz). Moreover, we defined a topology with 5 DC, 10 RPoP, 50 LPoP, and 500 HGW. Table 6 details the main

parameter values varied on simulations. In order to achieve better confidence, we performed three full executions of all possible combinations, producing a total of 405 executions (135 per full simulation) and combined the results by averaging the corresponding values of each configuration. Because we focus on investigating data offloading operations and due to the high number of simulations, we kept the HGW_TUPLE_FILE_SIZE parameter as a fraction of 0.1 of HGW_Storage, and both as low as possible. In this way, we observed more data offloading occurring within practical execution times (see Section 6.3).

6.3. Evaluation. We grouped the executions by HGW_Storage_Min_Threshold and HGW_Storage_Max_Threshold, i.e., the storage lower-level and upper-level thresholds, respectively. Section 6.1 explains these thresholds, which determine when the offloading starts and halts, according to a storage usage percentage. The groups were tagged with *T*_<Min>_<Max>, where <Min> and <Max> determine minimal and maximal thresholds for offloading, respectively. For instance, the execution T_10_20 starts offloading when storage usage reaches 20% of its capacity and halts at 10%. This storage threshold would trigger more data offloading than T_20_40, which accepts up to 40% usage and halts at 20%.

In order to understand how the OffFog chosen policies affected the system behaviour, we measured the network latencies under different experiment configurations. In particular, we evaluated the average network latencies, which represent the average time spent, in milliseconds, on all network communications between any pair of devices in the SCS topology (Section 5.1). Having lower latencies means that, on average, the simulations triggered fewer communications.

Figure 6(a) shows the average network latency when we vary the HGW_Storage_Compression and keep both HGW_Compression_Selection and HGW_Critical_Selection fixed in the minimum value (0.3). In all simulations, we can notice a reduction in network latencies with higher compression levels. On the T_10_40 configuration occurs the most considerable variation. It represents an excellent example of how the compression policy, notably the more strong compression techniques, could significantly impact the offloading performance. Also, we can notice that with HGW_Storage_Min_Threshold at 20 (T_20_30 and T_20_40), the latency is significantly lower than the others configuration. It demonstrates the positive influence of more tolerant storage thresholds, where fewer data fragments are offloaded, resulting in fewer network communications.

Figure 6(b) shows the results for HGW_Compression_Selection when we keep fixed the HGW_Storage_Compression and HGW_Critical_Selection to the minimum values (20 and 0.3, respectively). Again, the T_10_40 configuration presents the most considerable variation when HGW_Compression_Selection is between 0.5 and 0.7. It repeats the same pattern observed in Figure 6(a). To some extent, both cases are related to the compression policy. With more data fragments selected for compression, the storage threshold will delay reaching the upper level, resulting in fewer data offloading. Once more, with HGW_Storage_Min_Threshold at 20, the latency is significantly reduced to approximately half due to fewer data offloading.

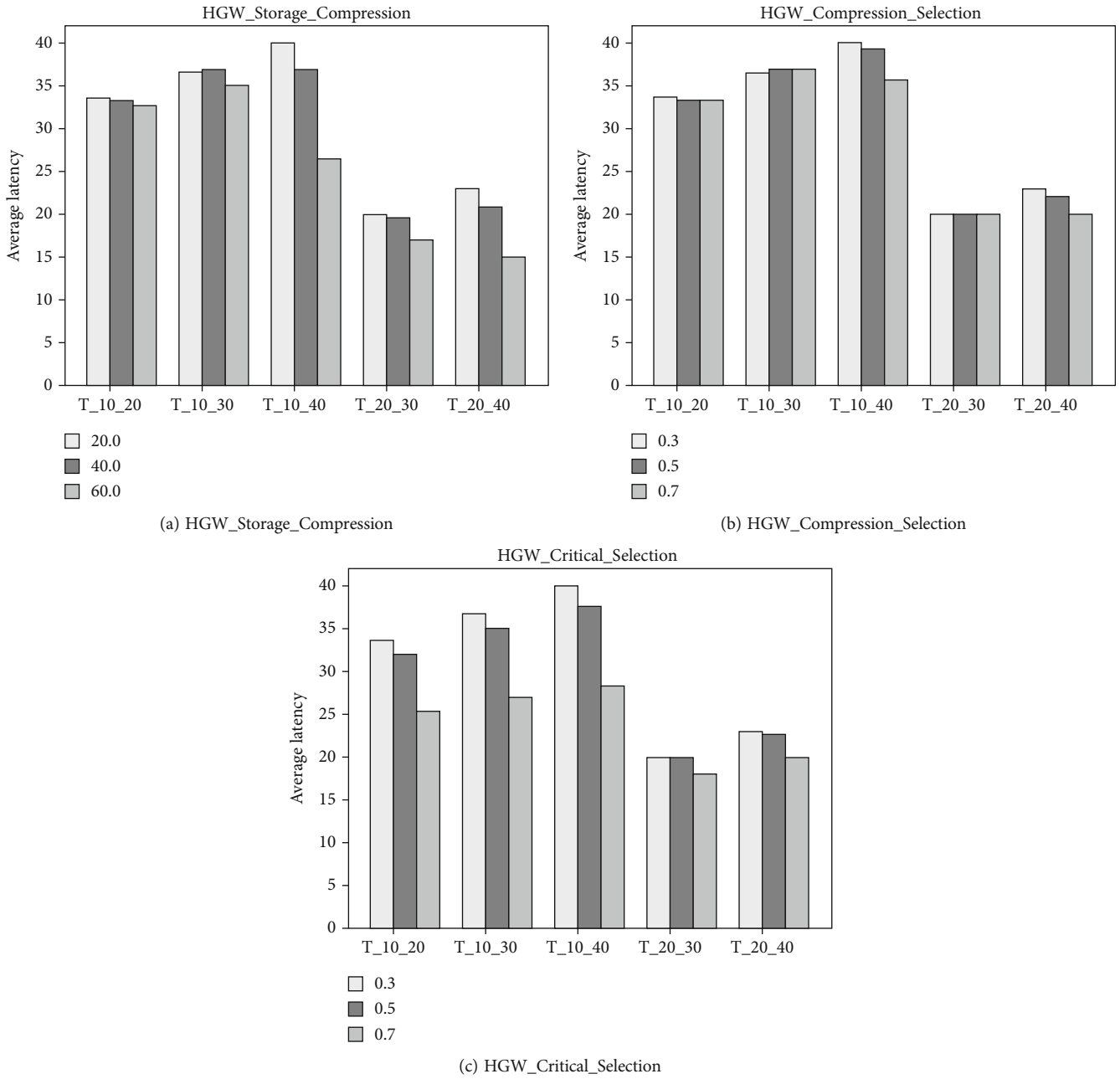
Figure 6(c) shows the results for HGW_Critical_Selection when we keep fixed the HGW_Storage_Compression and HGW_Compression_Selection to the minimum values (20 and 0.3, respectively). Changes in HGW_Critical_Selection produce significant latency variation, especially with values between 0.5 and 0.7. Indeed, the criticality policy directly affects offloading behaviour, given that a data fragment tagged with a CRITICAL bit would never be offloaded. More critical data fragments indicate that a smaller amount of data stored is available for offloading. However, it is relevant to note that this is not always feasible in practice since a critical data fragment will never be sent to the Cloud, which can be undesirable in many real-world scenarios. Finally, we can observe again a reduction in the latency (approximately 50%) when the HGW_Storage_Min_Threshold is the minimum (i.e., 20).

Table 7 presents the evaluation of the results of the two strategies: data offloading (offload, i.e., our extension) and data sent directly to the cloud host (cloud, i.e., without data offloading activated). The two strategy latencies are presented regarding their absolute values in milliseconds (time) and the ratio between offload and cloud (ratio). Data offloading outperforms cloud strategy in all latency statistics by at least 76% (max. ratio 0.24). On average, data offloading reduces the network latencies by 87%. These represent a remarkable improvement, demonstrating the efficacy of selecting significative data offloading policies.

One also can note that the minimum average latency is equal to 1 ms (Table 7). Very low latencies occurred when we simulated HGW_Compression_Selection at 0.7 and HGW_Storage_Compression at 60. It means that with a high number of compressed data fragments and at an elevated compress level, data offloading is reduced drastically. However, such behaviour can compromise data share between devices, which can be undesirable in real-world situations. Therefore, these data offloading policy configurations should be balanced according to the application scenario.

Although the average execution time presented a modest improvement of 5% (Avg. Ration 0.95), the execution was faster in all 405 simulations. With data offloading enabled, less processing occurs when Tuple components (data fragments) are not transferred to other hosts in the simulated topology. It is an interesting behaviour because the same is expected to happen in real-world scenarios.

In summary, the experiments with *iFogSimOffload* revealed the OffFog's selected policies effectiveness (Table 5). Grouping results by minimal and maximal thresholds (<*DV*>_Storage_Min_Threshold and <*DV*>_Storage_Max_Threshold) demonstrated a direct influence on network latencies. With more tolerant ranges, fewer data fragments are offloaded. Nevertheless, it should be selected with caution. Otherwise, it might even lead to no data sharing at all. The same considerations apply to data criticality (<*DV*>_Critical_Selection). Simulations with data compression parameters (<*DV*>_Storage_Compression and <*DV*>_Compression_



(a) HGW_Storage_Compression

(b) HGW_Compression_Selection

(c) HGW_Critical_Selection

Figure 6: HGW_Storage_Compression latency variation per storage threshold.

Table 7: Comparison between cloud and offload statistics.

Cloud		Average latency		Execution time	
		Time (ms)	Ratio	Time (ms)	Ratio
		165	—	53,776	—
Offload	Min.	1	0.01	48,145	0.90
	Max.	40	0.24	53,536	1.00
	Avg.	22	0.13	51,230	0.95
	Median	22	0.13	51,910	0.97

Selection) also demonstrate direct influence on latency reduction. Again, it should be considered carefully due to the need for more processing related to compression tasks.

It is important to note that *iFogSim* considers a constant data fragment size. It represents a limitation of our work since we cannot perform more refined analysis with

dynamic data fragment sizes. Moreover, because the storage capacity directly influences the time to trigger data offloading, our experiments become unfeasible to execute with large storage sizes. Thus, we kept this configuration more tangible, i.e., 100 KB data fragments and HGW storage capacity of 1 MB.

7. Conclusions

Although data offloading is a well-known aspect of fog computing applications, the specification of offloading policies is still an open issue due to the lack of clear guidelines. Therefore, software teams must have knowledge bases arranged in an organized and structured manner to support data offloading decision making for such specifications. OffFog provides this support and, in addition, it also offers a set of activities that guides the policy design from requirements prioritisation to conflicting requirements resolution.

The experiments with *iFogSimOffload* demonstrated the benefits of implementing data offloading based on OffFog recommended policies in an urban surveillance system. Once a real-world scenario topology is implemented, the *iFogSimOffload* can help to identify the best relation between configurations. An optimal combination of compression level, compression selection, critical selection, and offloading thresholds significantly reduces the amount of data offloaded. Indeed, our experiments show an improvement of at least 76% of network latencies compared to cloud strategy. Also, *iFogSimOffload* is executed faster in all simulations, another positive effect of implementing OffFog policies.

In future work, we plan to extend *iFogSimOffload* in order to analyse power-related policies. This feature would allow the inclusion of power-sensitive offloading policies, e.g., data offloading to the cloud during low-battery conditions, thus, avoiding data loss. In particular, we can extend the existing *iFogSim* ability to track device power consumption.

Data Availability

The data used to support the findings of this study are included in the article. More detailed resources were also publicly published, including source code and simulation results. These are available at the iFogSimOffload on Github repository (https://github.com/labcomu/iFogSimOffload).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank the Department of Informatics and Applied Mathematics (DIMAp) and the Coordination of Superior Level Staff Improvement (CAPES) for the assistance and support provided in all stages of this work. This study was financed, in part, by the Coordination of Superior Level Staff Improvement—Brazil (CAPES)—Finance Code 001.

References

[1] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–43, 2017.

[2] G. Javadzadeh and A. M. Rahmani, "Fog computing applications in smart cities: a systematic survey," *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, 2020.

[3] I. Azimi, A. Anzanpour, A. M. Rahmani, P. Liljeberg, and T. Salakoski, "Medical warning system based on internet of things using fog computing," in *2016 International Workshop on Big Data and Information Security (IWBIS)*, pp. 19–24, Jakarta, Indonesia, 2016.

[4] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli, "A simulation platform for largescale internet of things scenarios in urban environments," in *Proceedings of the First International Conference on IoT in Urban Space*, pp. 50–55, Rome, Italy, 2014.

[5] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[6] S. Melo, C. Silva, and G. Aquino, "Classification aspects of the data offloading process applied to fog computing," in *International Conference on Computational Science and Its Applications*, pp. 340–353, Cagliari, Italy, 2021.

[7] H. Gupta, A. Vahid Dastjerdi, S. Ghosh, and R. Buyya, "ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software - Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[8] M. Naas, P. Parvedy, J. Boukhobza, and L. Lemarchand, "IFogStor: an IoT data placement strategy for fog infrastructure," in *2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 97–104, Madrid, Spain, 2017.

[9] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: promises and challenges of IoT in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.

[10] C. A. Silva, *A Fog Computing-Based Software Architecture for Patient-Centered Management of Medical Records*, PhD thesis, Federal University of Rio Grande do Norte, 2020.

[11] C. A. Silva and G. S. Aquino Jr., "Fog computing in healthcare: a review," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1126–1131, Natal, Brazil, 2018.

[12] C. A. Silva, G. S. Aquino Jr., S. R. M. Melo, and D. J. B. Egídio, "A fog computing-based architecture for medical records management," *Wireless Communications and Mobile Computing*, vol. 2019, 16 pages, 2019.

[13] A. Alelaiwi, "An efficient method of computation offloading in an edge cloud platform," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 58–64, 2019.

[14] F. Pisani, V. Martins do Rosario, and E. Borin, "Fog vs. cloud computing: should i stay or should i go?," *Future Internet*, vol. 11, no. 2, p. 34, 2019.

[15] Y. Shan, H. Wang, Z. Cao, and K. Yury, "Data offloading in heterogeneous dynamic fog computing network: a contextual bandit approach," in *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*, pp. 73–77, Nagoya, Japan, 2021.

[16] S. He, B. Cheng, H. Wang, X. Xiao, Y. Cao, and J. Chen, "Data security storage model for fog computing in large-scale iot application," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 39–44, Honolulu, HI, USA, 2018.

[17] S. Verma, A. K. Yadav, D. Motwani, R. Raw, and H. K. Singh, "An efficient data replication and load balancing technique for fog computing environment," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2888–2895, New Delhi, India, 2016.

[18] M. Mukherjee, V. Kumar, S. Kumar et al., "Computation offloading strategy in heterogeneous fog computing with energy and delay constraints," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–5, Dublin, Ireland, 2020.

[19] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, vol. 14, no. 11, pp. 59–68, 2017.

[20] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[21] M. S. Bali, K. Gupta, D. Koundal, A. Zaguia, S. Mahajan, and A. K. Pandit, "Smart architectural framework for symmetrical data offloading in iot," *Symmetry*, vol. 13, no. 10, p. 1889, 2021.

[22] S. V. Margariti, V. V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective," *Future Internet*, vol. 12, no. 5, p. 89, 2020.

[23] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: a simulator for virtual machine migration in fog computing," in *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pp. 47–52, Austin, Texas, USA, 2017.

[24] M. Naas, J. Boukhobza, P. Raipin Parvedy, and L. Lemarchand, "An extension to ifogsim to enable the design of data placement strategies," in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pp. 1–8, Washington, DC, USA, 2018.

[25] C. Puliafito, D. M. Gonçalves, M. M. Lopes et al., "Mobfogsim: simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, article 102062, 2020.

[26] N. Mohan and J. Kangasharju, "Edge-fog cloud: a distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–6, Paris, France, 2016.

[27] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[28] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported internet of things environment," in *2015 6th International Conference on the Network of the Future (NOF)*, pp. 1–3, Montreal, QC, Canada, 2015.

[29] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: architecture, use case, and security and forensic challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, 2017.

[30] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: a blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.

[31] V. Moysiadis, P. Sarigiannidis, and I. Moscholios, "Towards distributed data management in fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, 14 pages, 2018.

[32] C. A. Silva, G. S. de Aquino Júnior, and S. R. M. Melo, "A blockchain-based approach for privacy control of patient's medical records in the fog layer," in *Anais Principais do XXV Simpósio Brasileiro de Multimídia e Web*, pp. 133–136, Porto Alegre, RS, Brasil, 2019, https://sol.sbc.org.br/index.php/webmedia/article/view/8011.

[33] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, Helsinki, Finland, 2012.

[34] A. M. Rahmani, T. N. Gia, B. Negash et al., "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[35] A. Yousefpour, C. Fung, T. Nguyen et al., "All one needs to know about fog computing and related edge computing paradigms: a complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[36] P. H. Kuo, A. Mourad, C. Lu et al., "An integrated edge and fog system for future communication networks," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 338–343, Barcelona, Spain, 2018.

[37] S. Esteves, J. Silva, and L. Veiga, "Quality-of-service for consistency of data geo-replication in cloud computing," in *European Conference on Parallel Processing*, pp. 285–297, Rhodes Island, Greece, 2012.

[38] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: a taxonomy, survey and future directions," in *Internet of Everything*, pp. 103–130, Springer, 2018.

[39] H. Shi, N. Chen, and R. Deters, "Combining mobile and fog computing: using coap to link mobile device clouds with fog computing," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pp. 564–571, Sydney, NSW, Australia, 2015.

[40] W. Lee, K. Nam, H.-G. Roh, and S.-H. Kim, "A gateway based fog computing architecture for wireless sensors and actuator networks," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 210–213, Pyeong-Chang, Korea (South), 2016.

[41] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing iot applications in the fog: a distributed dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, pp. 155–162, Seoul, Korea (South), 2015.

[42] M. Maksimovic, "Improving computing issues in internet of things driven e-health systems," in *Proceedings of the International Conference for Young Researchers in Informatics, Mathematics and Engineering'17*, pp. 14–17, Kaunas, Lithuania, 2017.

[43] N. I. M. Enzai and M. Tang, "A taxonomy of computation offloading in mobile cloud computing," in *2014 2nd IEEE*

international conference on mobile cloud computing, Services, and Engineering, pp. 19–28, Oxford, UK, 2014.

[44] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[45] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (Per Com Workshops)*, pp. 1–4, Sydney, NSW, Australia, 2016.

[46] F. A. Gomes, W. Viana, L. S. Rocha, and F. Trinta, "A contextual data offloading service with privacy support," in *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*, pp. 23–30, Teresina, Piauí, Brazil, 2016.

[47] D. V. Silva, T. G. Gonçalves, and G. H. Travassos, "A technology to support the building of requirements documents for iot software systems," in *19th Brazilian Symposium on Software Quality*, pp. 1–10, São Luís, Brazil, 2020.

[48] A. Alazeb and B. Panda, "Ensuring data integrity in fog computing based health-care systems," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pp. 63–77, Springer, 2019.

[49] M. Mukherjee, R. Matam, L. Shu et al., "Security and privacy in fog computing: challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[50] A. Heidari, M. A. Jabraeil Jamali, N. Jafari Navimipour, and S. Akbarpour, "Internet of things offloading: ongoing issues, opportunities, and future challenges," *International Journal of Communication Systems*, vol. 33, no. 14, 2020.

[51] U. O. Delaware, *Understanding Data Criticality*, 2018, http://www1.udel.edu/security/data/criticality.html.

[52] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, 2017.

[53] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, "Fog data: enhancing telehealth big data through fog computing," in *Proceedings of the ASE bigdata & socialinformatics 2015*, pp. 1–6, Kaohsiung, Taiwan, 2015.

[54] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, pp. 15–20, Hong Kong, China, 2013.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 1951095, 14 pages
https://doi.org/10.1155/2021/1951095

Research Article

A Privacy-Preserving Incentive Mechanism for Data Offloading in Satellite-Terrestrial Crowdsensing

Boxiang Zhu,[1] **Jiarui Li**,[2] **Zhongkai Liu**,[3] **and Yang Liu**[4]

[1]*School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Department of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[3]*International School, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[4]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Yang Liu; liu.yang@bupt.edu.cn

Received 30 September 2021; Accepted 7 December 2021; Published 31 December 2021

Academic Editor: Chi Lin

Data offloading algorithm is the foundation of urban Internet of Things, which has gained attention for its large size of user engagement, low cost, and wide range of data sources, replacing traditional crowdsensing in areas such as intelligent vehicles, spectrum sensing, and environmental surveillance. In data offloading tasks, users' location information is usually required for optimal task assignment, while some users in remote areas are unable to access base station signals, making them incapable of performing sensing tasks, and at the same time, there are serious concerns about users' privacy leakage about their locations. Until today, location protection for task assignment in data offloading has not been well explored. In addition, existing privacy protection algorithms and data offloading task assignment mechanisms cannot provide personalized protection for different users' privacy protection needs. To this end, we propose an algorithm known as differential private long-term privacy-preserving auction with Lyapunov stochastic theory (DP-LAL) for data offloading based on satellite-terrestrial architecture that minimizes the total payment. This not only gives an approximate optimal total payment in polynomial time but also improves the issue of poor signal in remote areas. Meanwhile, satellite-terrestrial data offloading architecture integrates wireless sensor networks and cloud computing to provide real-time data processing. What is more, we have considered long-term privacy protection goals. We employ reverse combinatorial auction and Lyapunov optimization theorem to jointly optimize queue stability and total payment. More importantly, we use Lyapunov optimization theorem to jointly optimize queue stability and total payment. We prove that our algorithm is of high efficiency in computing and has good performance in various economic attributes. For example, our algorithms are personally rational, budget-balanced, and true to the buyer and seller. We use large-scale simulations to evaluate the proposed algorithm, and compare our algorithm with existing algorithms, our algorithm shows higher efficiency and better economic properties.

1. Introduction

Advances in network technology and industrial production capabilities have led to a proliferation of mobile devices, e.g., smartphones, smart watches, and tablets, which are embedded with multiple sensors, called user equipments (UEs), e.g., global positioning system (GPS), gyroscopes, and microphones. How to offload data properly and efficiently has become a hot topic. Owing to data offloading platform provide high-quality services, this technology has been maturely applied in various fields, such as intelligent transportation [1], spectrum sensing [2], and environmental monitoring [3].

The typical data offloading platform system usually consists of three layers: (1) Sensing Layer, (2) Network Layer, and (3) Cloud Computing Layer, which is shown in Figure 1. In Sensing Layer, users use built-in sensors to accomplish the sensing tasks from the platform. These

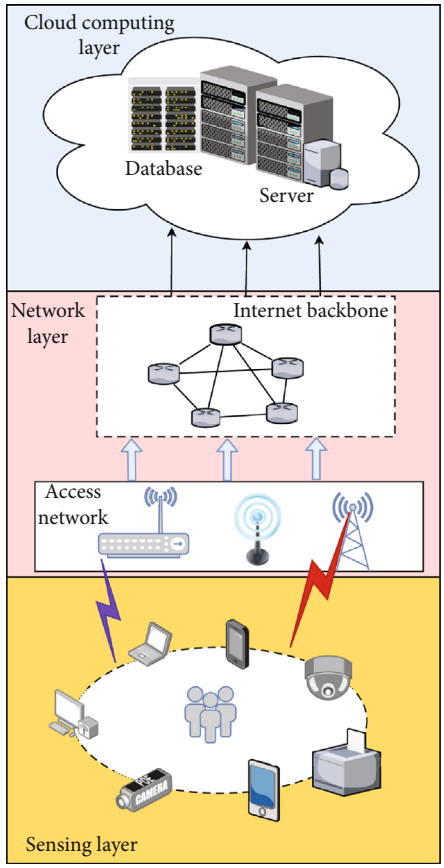


Figure 1: The typical data offloading platform system.

tasks are performed by the sensors built into the terminals, such as the camera of a phone, GPS, Bluetooth, and infrared sensing. In Network Layer, users send the data to the platform, which is composed of a backbone of centralised servers, by means of different networks, such as cellular network, sensor network, and Wi-Fi networks, for sensing. In Cloud Computing Layer, it consists of databases and servers to analyze and process the sensing data returned by users to set up intelligent systems driven by different applications.

However, there are still some challenges to such a data offloading platform system architecture. First, it requires precise real-time data aggregation across a large number of user terminals. For the sensing platform, participating in real-time data aggregation implies huge consumption, because it not only consumes the power of terminals, users' time, and system resources, e.g., power and memory, but also requires the platform to encourage users by designing incentives. At the same time, in real-time data aggregation, if the expected benefits are not achieved, it may lead users to quit the system midway, such as closing their accounts, resulting in a decrease in service quality of service (QoS) for providers. Second, people are now more concerned about their privacy, and considering that collecting and uploading sensing data may cause privacy leakage and users may even be maliciously attacked, they will not continue to perform sensing tasks. Third, in typical sensing tasks, we mainly initiate sensing requests to users through cellular networks, Wi-Fi networks, sensor networks, etc. However, in remote areas with weak signals, users cannot use traditional access networks to complete sensing tasks.

To overcome the above challenges, we introduce a differential privacy-based satellite-terrestrial architecture to address this problem. We emphasize that the incentives are designed for real-time data aggregation and also need to ensure that each user does not earn less than the expected benefits while also protecting their privacy, thus maintaining long-term user participation [4], while the satellite-terrestrial network allows users in all regions of the globe to be able to complete sensing tasks [5]. In this paper, we quantify users' location information through differential privacy [6] and precise data aggregation based on satellite-terrestrial networks and propose an auction-based long-term online incentive mechanism to handle location information, sensing task posting, bidding, and task completion involving both buyers and sellers. We also give an effective solution to the strategy coupling problem that arises in the process.

The major contributions of this paper are listed below:

(i) We use satellite-terrestrial network architecture, which can well handle the problem of weak signals in remote areas and promote the integration of global sensing networks

(ii) We propose differential privacy single-minded reverse combinatorial auction with heterogeneous cost, which can minimize the total payment. This algorithm means that we cannot only give approximate optimal total payment in polynomial time but also guarantee the approximate ratio to the optimal total payment

(iii) To achieve long-term privacy protection, we propose to use reverse combination auctions to minimize the total long-term payment to the user. Also, we use Lyapunov optimization theorem to jointly optimize queue stability and total payment

(iv) We analyze our proposed algorithm theoretically and use a large number of simulations for evaluating and analyzing our proposed mechanism

The remainder of the paper is organized as follows. The related work is discussed in Section 2. Section 3 describes system model in detail. Section 5 designs the DP-LAL mechanisms and proves its feasibility. Section 7 shows simulation results of the DP-LAL mechanism. Section 8 concludes the paper.

2. Related Work

2.1. Privacy-Preserving in Crowdsensing. There has been so much work made great effort to guarantee privacy in data offloading. Security and privacy are the basic attributes of crowdsensing, which requires security research method and privacy threat analysis to provide strong privacy

protection barrier. For example, in location privacy protection, users upload obscure distances and personal privacy levels in a new framework of personalized privacy protection [7]. With multiple levels of agents [8] and protocols, users can finish crowdsensing tasks without any data linkability [9]. For instance, in [10], Cai et al. proposed a knowledge market framework based on crowdsensing data offloading. Likely, SPOON [11] considers task allocation and privacy preservation using the credit of users. They also propose a framework for fog computing [12] that can improve the accuracy of task allocation. A cryptography-based perspective is also one important way to protect privacy. Zhang et al. [13] present RPTD scheme, which consists of the homomorphic Paillier encryption, and superincreasing sequence, one-way hash chain. In [14], Stackelberg game is used in the process of data collection in crowdsensing to preserve privacy.

2.2. Incentive Mechanism in Crowdsensing. Incentive mechanism is an essential part of the crowdsensing system, the specific performance of which is either maximize the overall profit of the system or minimize the payment of the users. In crowdsensing, there are different incentive methods. In brief, incentives can be divided into monetary and nonmonetary incentives. Monetary incentives are mainly paid to motivate users to participate. One of the most important monetary incentives is the auction, which selects the subset of participants who pay less to complete a sensing task by making an offer on their sensing data. Monetary payment incentive returns participants' sensing data, which is the most direct and currently the main incentive method. Stackelberg game is an important method. Nie et al. model a two-stage game [15] and a multiple leaders and multiple followers Stackelberg game approach [16]. IBE [17] consists of participant selection scheme and payment decision scheme, which improve task fulfillment rates, participant utility, and platform benefits. Sun et al. [18] are offering personalized payments to users as compensation for the cost of privacy. They both increase the user's income to incentivise them. A dynamic demand-driven incentive mechanism [19] is proposed that dynamically varies the reward for each round of sensing tasks in an on-demand way to balance its popularity. Reducing user costs is also an effective incentive approach. Hu et al. also design novel privacy incentives to protect users' real bid information from the platform [20]. Zhan et al. [21] propose a policy gradient-based approach that efficiently learns optimal pricing policies directly from history. Duan et al. [22] design preference-based auction mechanisms (PreAM) that guarantee at the same time individual reasonableness, budgetary viability, authenticity of preferences, and realism of prices. In vehicular crowdsourcing, Chen et al. [23] propose a time-sensitive incentive mechanism that takes into account the uncertainty of vehicle travel times.

2.3. Satellite-Terrestrial Networks. Satellite-terrestrial network has better performance in time delay, QoS, etc. Meanwhile, satellite-terrestrial network is different from the traditional cellular network, which can get rid of the geographical limitation and achieve global network integration. Satellite-terrestrial network for 6G field has a better performance role. For example, Fu et al. [24] propose a tapped water-filling algorithm to solve the problem about the hot air balloons. Jia et al. [25] design a new algorithm that can efficiently allocate SDLSN resources to provide services to the users. Fang et al. [26] design a smart, agile, and safe HSTN for the omnipresent Internet of Things in 6G. A satellite terrestrial network architecture [27] is proposed to integrate superdense LEO and terrestrial networks for efficient data allocation. Meanwhile, Li et al. [28] propose an integral satellite-terrestrial transport solution that offloads base station traffic via broadcast transmissions from satellites, enabling energy-efficient wireless access networks. Ruan et al. [5] focus on how to provide services with different latency QoS requirements in energy-limited systems and propose an efficient power allocation mechanism for satellite-terrestrial networks. Wang et al. [29] study the problem of channel gain and distance between terrestrial users and BSs and propose a user association scheme to provide complete coverage for terrestrial users. Jiang et al. [30] propose a block division algorithm based on density, which divides users into a series different sized blocks based on their density, in order to solve the problem of poor quality due to uneven distribution of users. Huang et al. [31] implement an online scheduling framework that maximizes the volume of data accepted with minimal energy usage when files are downloaded from LEO data centers to satisfy user demand. Li et al. [32] enhance the reach of the hybrid satellite-terrestrial marine telecommunication network by deploying drones. Lin et al. propose a temporal-spatial charging scheduling algorithm [33]. Meanwhile, they introduce linear constraints to improve an energy transfer model [34], focus on fee exclusivity in random event monitoring [35], study of online collaborative charging schedules for multiple WCVs [36], and address the charging of the WRSN in the case of obstructions in order to maximize the charging utilities within a given energy constraint [37]. Also, they focus on the problem of attacks and develop a new attack for WRSN aimed at maximizing disruption [38]. In their research on underwater visible light communication systems, they propose a dedicated coding scheme based on a dual CPL design [39].

However, none of the works above uses privacy-preserving auction with Lyapunov to enhance the performance of computing and thus reducing the payment of users, and no one has solved the problem of weak signals from base stations in remote areas. In short, our work incorporates all the strands of the above work in an innovative way. Our previous work [40] is mainly on the building and solving of our model, which includes the definitions of the model and the solution and optimization of the model, as well as a theoretical analysis of the lemmas of our model. In this paper, we summarize related work, including privacy-preserving mechanisms, incentive mechanisms, and satellite-terrestrial networks. Meanwhile, our work gives a more detailed system design model and a more detailed explanation of our model, not only that, but we also give the technical roadmap, which can facilitate the reader to

implement our work more easily, and finally provide more convincing simulations to illustrate the advancement of our work.

3. System Model

We describe the data offloading in satellite-terrestrial architecture as Figure 2. There are still many unsolved problems in the typical data offloading platform system. We need a network that can cover the world, especially remote areas, to achieve global data offloading integration. Satellite-terrestrial network is a great way to achieve global network integration. In general, a satellite network consists of several satellites, ground stations (GS), and a Network Operations Control Centre (NOCC), providing services for navigation, emergency relief, communication/relay, global resources, and geographic information management. Depending on their altitude, satellites can be categorized as low earth orbit (LEO), medium earth orbit (MEO), or geostationary orbit (GSO).

Users use sensors built into mobile terminals, such as the phone's camera, GPS, Bluetooth, and infrared, to finish the tasks allocated by the platform. Urban users send data to the platform via cellular networks and remote users via satellite-terrestrial network. The backbone network transmits the data to the SDN controller center [41], which consists of databases and servers for analyzing and processing the sensing data returned by users to set up smart sensing systems powered by different applications.

Meanwhile, it requires precise real-time data aggregation across a large number of user terminals, and the sensing platform continuously collects sensing data from users to provide real-time services. For the sensing platform, participating in real-time data aggregation implies a huge consumption, because it not only consumes the power of terminals, users' time, and system resources (e.g., power and memory) but also requires the platform to encourage users by designing incentives. At the same time, in real-time data aggregation, if the expected revenue is not achieved, it may lead users to quit the system midway, such as closing their accounts or even uninstalling mobile applications, resulting in a decrease in service quality of service providers. Therefore, we emphasize that the incentive mechanism design for real-time data aggregation needs to make sure that each user's income is not lower than expected, thus maintaining its long-term participation.

Auctions are an effective way to design incentive mechanisms. In the data offloading auction mechanism, user's private information is often included in their bids, and this part is usually private. Therefore, in order to make sure the user's bid privacy is not being disclosed, this project intends to protect the privacy of users by submitting bid data with differential privacy noise and design a near-real, individual-reasonable, computationally effective long-term incentive mechanism that satisfies differential privacy. We suppose that all of the users are connected to a satellite-terrestrial network. Our goal is to ensure that the total payment of the data offloading platform is minimized, that is, to maximize the utility of the data offloading platform. We list the notations in Table 1.

Suppose there are a set of N users $\mathbb{N} = \{1, \cdots, N\}$. The set of access points (APs) is $\mathbb{M} = \{1, \cdots, M\}$. The sensing platform issues K tasks. Each task $k \in [1, K]$ contains $\mu_k \geq 1$ sensing locations and therefore contains μ_k subtasks. We use $t_{k,j}$ represents the j^{th} subtask of task k, all μ_k subtasks of task k are represented by $\mathbb{T}_k = \{t_{k,j} \mid j \in [1, \mu_k]\}$, and all subtasks are represented by $\mathbb{T} = \{t_{k,j} \mid k \in [1, k], j \in [1, \mu_k]\}$. But for a single sensing task, all of its subtasks are required to be completed within the same time slot. Therefore, we require each user to bid for a maximum of one subtask per sensing task at a time.

4. Problem Formulation

The APs are essential in the network, and each user must be linked to only one AP. We denote $a_{i,m}$ as

$$a_{i,m} = \begin{cases} 1, & \text{user } i \text{ is associated with AP } m \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

We assume the power and bandwidth of each AP is P_m and B_m, $m \in \mathbb{M}$, and the lost during transmission including rayleigh fading and path loss, etc., as $\mathbb{L}_m$. The power of interference received at user i is I_i, and we use a constant η_0 to represent the zero mean and unit variance additive white Gaussian noise (AWGN) power. Thus, we have the signal to interference plus noise ratio (SINR) of the channel between user i and AP m and the service rate $S_{i,m}$ as:

$$\text{SINR}_{i,m} = \frac{P_m \mathbb{L}_m}{I_i + \eta_0}. \tag{2}$$

According to the Shannons formula, the service rate $S_{i,m}$ for each user i that is associated with AP m can be described as:

$$S_{i,m} = B_m \log_2^{(1+\text{SINR}_{i,m})}. \tag{3}$$

We also suppose the bidding data of each user is A_i. Actually, the amount of data $\sum_{i \in \mathbb{N}} A_i$ is constrained by the capacity of backhual. We suppose the system is strong enough to carry that much of data. The delay $t_{i,m}$ between user i and the corresponding AP m is

$$t_{i,m} = \frac{A_i}{S_{i,m}}. \tag{4}$$

What is more, we have to ensure that each user is associated with only one AP, and the delay is less than a constant T_c as described in the following two constraints.

$$\sum_{m \in \mathbb{M}} a_{i,m} = 1, \forall i \in \mathbb{N}, \tag{5}$$

$$t_{i,m} < T_c, \forall i \in \mathbb{N}, \forall m \in \mathbb{M}. \tag{6}$$

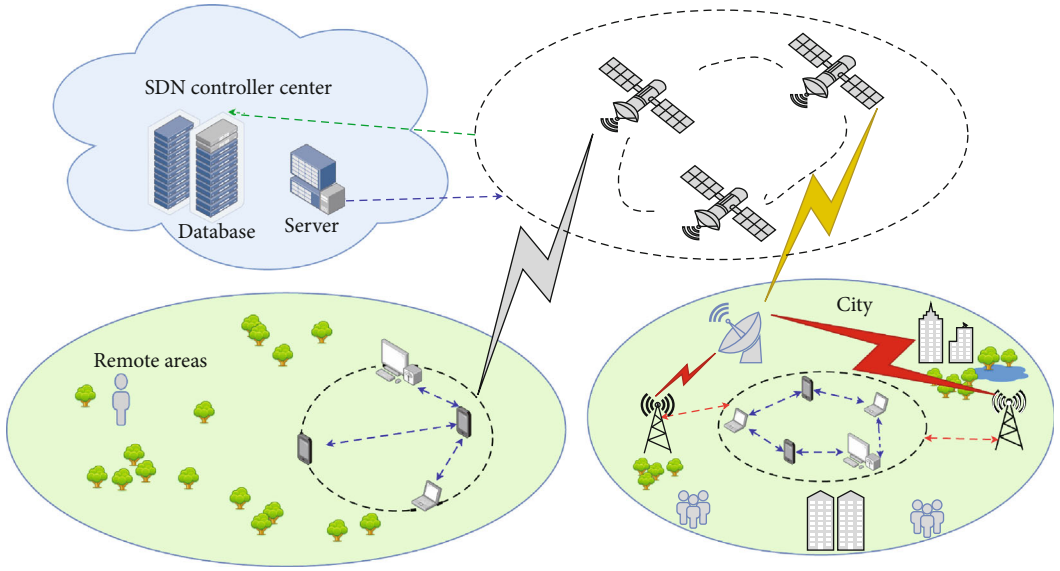


Figure 2: System architecture for data offloading in satellite-terrestrial crowdsensing.

Table 1: The notations in this paper.

N	The set of users
K	The amount of sensing tasks from the platform
μ_k	The amount of subtasks of task k
$t_{k,j}$	The jth subtasks of task k
T_k	All μ_k subtasks of task k
T	The set of all of the subtasks
n_k	The amount of users for task k
i	A specific user
b_i	The bid of user i
L_i	A subset of T, which represents the subtasks user i is interested in
C_i	The cost user i claimed of finishing task set L_i
W	The winner set of users L_i
u_i	The profit of user i
c_i	The true value of the cost of user i needs to finish his task set L_i

The sensing platform is aimed at selecting $n_k, k \in [1, k]$ user equipments(UEs) for each sensing task. Since users have to compete for the sensing tasks in exchange for reward, we intend to model the user selection in the data offloading task as a reverse auction; the sensing platform plays a role of auctioneer in the auction of sensing task; every user $i \in [1, N]$ serves as a tenderer for the sensing task. The sensing platform broadcasts a list which include subtasks called τ, and each potential user i who was interested will give a bid $b_i = (L_i, C_i)$, $L_i \subset \mathbb{T}$, where C_i is the cost he claimed of finishing sensing subtask set L_i. $x_i = 1$ indicates that the user i wins the subtask set L_i. We, respectively, assume that C_i are limited in the range of $[C_{\min}, C_{\max}]$ where $C_{\max}$ and $C_{\min}$ represent the maximum and minimum sensing cost. Each user needs to follow two rules during the bidding. First, for a sensing task, the user can only bid for at most one of the subtasks. Second, the user may bid for several different sensing tasks at the same time. The first rule is to protect users from using strategies to control bids. For instance, there are two users A and B who bid subtasks $t_{1,1}$ and $t_{1,2}$. If this is the real bid, A will be assigned $t_{1,1}$, and B will be assigned $t_{1,2}$. But A may find out that if he had $t_{1,2}$, he could get more reward. Therefore, A may misrepresent the cost of $t_{1,1}$ and give the chance to B. Thus, A deliberately misrepresents the cost in order to win another subtask and get more awards. The first rule can effectively block solve the problem. The second rule is intended to reduce the cost of performing a sensing task, that is, allowing a single user to simultaneously perform multiple sensing tasks that do not interfere with each other in the same time slot.

The sensing platform selects the winner set W in order to execute all subtasks in τ. User i maintains a true value of the cost of subtask set L_i represented by c_i.

Definition 1 (user's utility). If the sensing platform accepts user's bid b_i, then the utility of the user is defined as

$$u_i = p_i x_i - c_i, \tag{7}$$

where p_i is the reward received from the platform. If the user i is not the winner, we will set u_i to 0. Each user knows the allocation algorithm and payment method before bidding and actively uses their own strategies to maximize profits. Therefore, the cost C_i he claimed by users may be unequal to each participant's c_i.

Different users have different sensing costs regarding the same sensing task. As each user determines his own subtask set, we have to design a new mechanism in case he lies about his cost C_i.

Definition 2 (authenticity). The auction is authentic if and only if for any user i, the utility when he bids the real *value* c_i is no less than any other bid C_i, i.e.,

$$u_i(c_i, C_{-i}) \geq u_i(C_i, C_{-i}), \tag{8}$$

where c_{-i} represents the cost vector of all users except user i.

Now, we suppose that each user's cost C_i equals the real cost c_i. The sensing cost comes from both finishing the sensing tasks and power and traffic of UE and AP used during the process. We use $b_i^p(t)$ to represent the privacy and power and traffic cost of UE i at time slot t, and the cost of sensing tasks is $b_i^{L_i}(t)$. We have $c_i(t)$, i.e.,

$$c_i(t) = b_i^p(t) + b_i^{L_i}(t). \tag{9}$$

Definition 3 (long-term participation). An auction meets long-term participation constraint if and only if the user's choice probability is

$$\frac{1}{T}\sum_{t\in T} x_i(t) \geq D, \forall i \in \mathbb{N}, \tag{10}$$

where D is the threshold for the time the user was selected. In practice, we can conduct a questionnaire to determine the minimum threshold to maintain long-term user participation.

Our goal is to maximize user's long-term total profit. The power cost is relevant to the power of AP connected to the user, i.e.,

$$\max \sum_{i\in\mathbb{N}} u_i(t), \tag{11}$$

$$\text{s.t.} \left| \bigcup_{i\in N} \cap T_k \right| = \mu_k, \forall k \in [1, k], \tag{12}$$

$$\frac{1}{T}\sum_{t\in T} x_i(t) \geq D, \forall i \in \mathbb{N}, \tag{13}$$

$$\sum_{m\in\mathbb{M}} a_{i,m} = 1, \forall i \in \mathbb{N}, \tag{14}$$

$$t_{i,m} < T_c, \forall i \in \mathbb{N}, \forall m \in \mathbb{M}, a_{i,m} = 1. \tag{15}$$

The first constraint states that all K tasks will be executed by users in the winner set. The second constraint limits that in a sensing task, each user can bid up to at most one subtask. The third constraint limits the number of sensing tasks that the user performs in each round of bidding, where γ is a constant decided by the sensing platform. The fourth constraint is to ensure long-term user participation. The last two constraints ensures the user can get stable connection to AP and low latency. Our overall goal is to select the user set which has the lowest sensing cost under the above constraints. In addition, our mechanism should also satisfy the authenticity and individual rationality.

The formulated problem is NP hard. We intend to solve this problem by using heuristic algorithms. Firstly, the winner set will be initialized to an empty set, after that the users who can make the greatest contribution to the selected set of sensing tasks will be able to join the winner set. If there are several users contributing the same, we will randomly select one. When all tasks are selected, this process ends.

5. Mechanism Design

The above heuristic algorithm does not protect user's location privacy. We intend to protect the privacy of users by using differential privacy. Differential privacy is an excellent tool which provides a statistical guarantee of privacy leaks caused by the release of sensitive input datasets. Differential privacy's basic idea is when the two input data sets are basically the same, the output of the two sets is almost the same under this mechanism.

Definition 4 (differential privacy private). For a random function M, if there is only one different input between any two data sets D_1 and D_2 and for any output set $O \subseteq \text{Range}(M)$, M is of differential privacy, i.e.,

$$\Pr\left|M(D_1) \in S\right| \leq \exp(\varepsilon) \times \Pr\left|M(D_2) \in S\right|. \tag{16}$$

Random function M corresponds to our mechanism, and $\text{Range}(M)$ is the result space of the mechanism. Approximate differential privacy allows users to have a small privacy budget ε, i.e.,

$$\Pr\left|M(D_1) \in S\right| \leq \exp(\varepsilon) \times \Pr\left|M(D_2) \in S\right| + \delta. \tag{17}$$

For the above heuristic algorithm, we intend to randomise the price choice result in the algorithm by using the exponential mechanism. Firstly, we define a ranking index

that applies to every user $i \in [1, N]$ and use this index to indicate the platform's preference for the user.

$$r_i = \frac{C_i}{|(T - T_w) \cap L_i|}. \tag{18}$$

The collection T_w represents the currently winning bid subtask set, i.e., $T_w = \cup_{i \in N} L_i$. The basic principle of the above definition is that the platform always prefers to select those users that are not yet included in the set T_w and those claimed the lowest cost of the subtask. In each iteration, we calculate the user's preference ranking. For those excluded from the winning bid list, we will use the indexing mechanism shown below to calculate the quality score.

$$q(C_i, x_i) = -r(C_i). \tag{19}$$

In the above formula, we use the negative sign to point to the index mechanism in the reverse auction model. Obviously, the smaller $r(C_i)$ the higher quality score of user i. The probability that a user is selected as the winner in the index mechanism is

$$\Pr\,(x_i = 1) \propto \exp\left(-\varepsilon' r(C_i)\right), \tag{20}$$

where $\varepsilon' = \varepsilon/\Delta \cdot e \ln(e/\delta)$ and Δ means the maximum input difference of C_i, which is equal to $C_{\max} - C_{\min}$. As for ε and δ, these two parameters are used to balance privacy leakage and social cost. We randomly determine the winner from the remaining users based on the probability in the iteration. Then, we remove the user's bid from the original set and add it to the winner set. The model diagram for this system is shown in Figure 3. The user bid data will be protected by differential privacy, and this data is then added with noise, after which it enters the sensing platform. Finally, the winner is decided based on the evaluation index.

Note that our algorithm is subject to time average, where the current strategy is coupled to future strategies. However, a user's bid in the future is unknown, making the selection in the user's current time slot challengeable. In order to solve this challenge, we intend to transform the long-term participation requirements into queue stability requirements and use Lyapunov optimization theorem [42] to jointly optimize the queue stability and platform benefits.

We use leverage Lyapunov optimization theorem and transform the long-term participation constraint into queue stability requirements. Assume there is a virtual request queue Q with arrival rate D. When $x_i(t) = 1$, a virtual request leaves the queue. We can describe the backlog of the queue as follows:

$$Q_i(t+1) = \max\{Q_i(t) - x_i(t), 0\} + D. \tag{21}$$

Then, we consider the Lyapunov function and the Lyapunov drift:

$$L(t) \triangleq \frac{1}{2}\sum_{i \in \mathbb{N}} (q_i(t))^2, \tag{22}$$

$$\Delta(t) \triangleq L(t+1) - L(t). \tag{23}$$

Through the Lyapunov drift theorem, by stabilizing the virtual queue Q, we can satisfy the long-term participation requirements. However, another goal of us is to maximize user's long-term total profit, which includes the sensing cost and the power cost during bidding data transmission. The power cost is relevant to the power of satellite connected to the user, i.e.,

$$\max \sum_{i \in \mathbb{N}} u_i(t). \tag{24}$$

We can jointly stabilize the queue and minimize the long-term total payments by minimizing the following drift-plus-penalty function, i.e.,

$$\Delta(t) + V\sum_{i \in N} u_i(t), \tag{25}$$

where V is a tradeoff parameter to balance the importance of queue stability and total payments. Note that $\Delta(t)$ has a upper bound as follows:

$$\Delta(t) = L(t+1) - L(t), \tag{26}$$

$$= \frac{1}{2}\sum_{i \in N} (\max\{q_i(t) - x_i(t), 0\} + D)^2 - \frac{1}{2}\sum_{i \in N} q_i(t)^2, \tag{27}$$

$$\leq \frac{1}{2}\left(\sum_{i \in N}\left\{q_i(t)^2 + x_i(t)^2 + 2q_i(t)(D - x_i(t))\right\} - \sum_{i \in N} q_i(t)^2\right), \tag{28}$$

$$\leq \sum_{i \in N} \frac{D^2 + 1}{2} + \sum_{i \in N} q_i(t) D - \sum_{i \in N} q_i(t) x_i(t). \tag{29}$$

In (28), we omit the item $-2Dx_i(t)$, thus scaling the inequality, and we have (29) because $x_i(t) \in \{0, 1\}$.

Considering that the first two items in (29) are constants, if we want to minimize the drift-plus-penalty function, we can minimize the following function instead.

$$\max \sum_{i \in N} V u_i(t) - q_i(t) x_i(t). \tag{30}$$

Since the algorithm we designed involves the NP-hard problem of the total payment minimization (TPM) problem and gives the specified price in time, we cannot calculate the winner set with the smallest total payment amount in polynomial time. In the case where P is not equal to NP, we are unable to find the group with the best price and make a payment based on the aggregated price collection and ensure that the selected payment method and expenditure ratio are optimal. In addition, in order to defend user's privacy

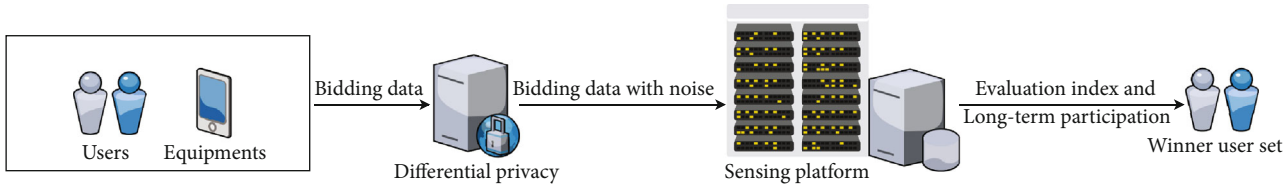


Figure 3: System design model.

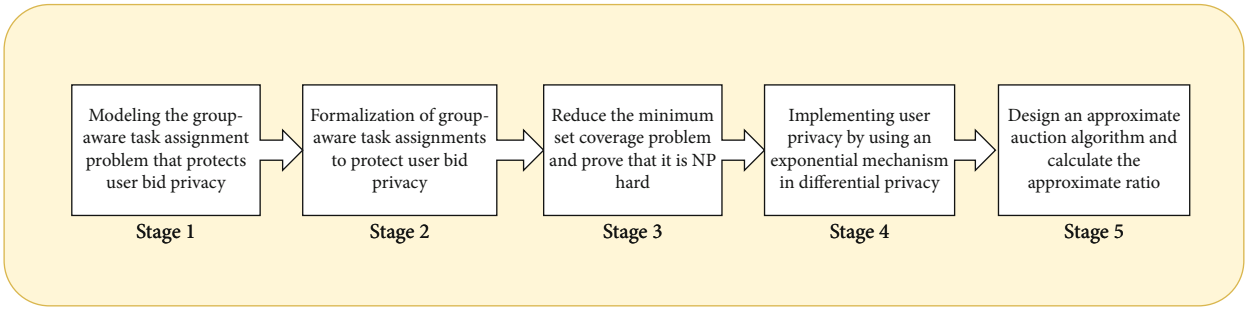


Figure 4: Partial technical roadmap for long-term incentives for group-aware users with bid privacy protection.

information unrelated to the auction during bidding process and to maintain the long-term participation of the user, we consider Lyapunov stochastic optimization to ensure the stability of the queue. Finally, according to the technical roadmap shown in Figure 4 and model diagram, we obtain the DP-LAL auction algorithm shown in Algorithm 1. As for the input, N is set as the number of users, T indicates the list of tasks and T_w shows the subtask lists, C shows the cost when users perform this task, i shows the index mechanism ranking value, $\{b_n\}$ implies workers' bid profile, Γ_i is the matrix, and Q means vector.

6. Theoretical Analysis

We prove that DP-LAL meets the four attributes required for a general auction. That is, DP-LAL is computationally efficient, with individual rationality, budget balance, and authenticity for both sellers and buyers.

Lemma 5. *DP-LAL is computationally efficient.*

Proof. We assume that all buyers are sorted in a nonincremental order for display, and for all sellers, we sort them in a nondecreasing order according to their requirements. It takes $\emptyset(n \log n + m \log m)$ time to sort the buyers and sellers, and the rest of the DP-LAL takes time to be linear. The sum of the two is the total time spent on implementing DP-LAL, which is $\emptyset(n \log n + m \log m)$.□ □

Lemma 6. *DP-LAL is individually rational.*

Proof. We assume that each winning buyer i pays b_i. When he bids truthfully, his profit is $u_i = p_i x_i - c_i > = 0$. As for seller k, when he charges truthfully his utility which is $u_k = p_u x_u - c_u > = 0$, where p represents income, x represents user's index, and v represents the estimated cost of user. It shows that both sellers and buyers are individually rational in DP-LAL.□ □

Lemma 7. *DP-LAL is budget-balanced.*

Proof. If the auctioneer does not select any user as the winner, he need not have to pay any sellers. If $n \geq k+1$, all the winning buyers' payment is $p^b = (n-1)b_n \geq 0$. The profit in the auction is nonnegative because there is no winner. When $n < k+1$, the overall payment paid by the winning buyers is still $p^b = (n-1)b_n$. The all income earned by the winning sellers is $p^s = (k-n+1)a_{k-n+2}$. The auctioneer's profit is $p^b - p^s \geq 0$.□ □

Lemma 8. *DP-LAL is authentic only when the buyers bid truthfully.*

Proof. We assume that b_i and u_i are the payment and profit for user i, when the bids are truthfully, i.e., $b_i = c_i$. If the user lies during budding, we have $b_i' \neq c_i'$. When the tie-breaking occurs, we suppose that i is at the forefront of all users. We prove that $u_i' \geq u_i'$ for any $b_i \neq c_i$. Consider the following case: if a buyer wins, we have p_i as its minimum value and $u_i = p_i x_i - c_i \geq 0$ by Lemma 6. We know that by bidding $b_i > c_i$, the outcome of the auction will not be changed, and it results in $u_i' = c_i - p_i' = c_i - p_i = u_i$. When the user loses in the auction using his truthful bid, his profit $u_i = 0$. It is obvious that bidding $b_i < c_i$ will not influence the result of the auction. By bidding $b_i > b_{\min}$, i becomes a winner. However, his utility becomes $u_i' = c_i - p_i = c_i - b_{\min} = u_i$. Based on the case, we obtain that if and only if bids are truthful, the users can earn maximum utility.□ □

Lemma 9. *DP-LAL is truthful for the sellers.*

Proof. For every seller s, we suppose that p_s and U_s are the payment and profit, when he charges truthfully, i.e., $a_s = c_s$. Let p_s' and u_s' be his payment and profit, when he lies, i.e.,

Input: $N, T, T_w, C, i, \Gamma_i, \gamma$;
Output: W, p
1: Sort all users by bid from high to low, $b_1 \geq b_2 \geq \cdots \geq b_n \min_b \in \{b_1 \cdots b_n\}$
2: Ranking index mechanism:
3: $i = C_i / |(T - T_w) \cap \Gamma_i|$;
4: arg $\max_{i:b_{\min} \leq p} \longrightarrow b_{\min}$;
5: $\{i_{\min}, i_{\min+1}, \cdots, N\} \longrightarrow I$;
6: Add the constant δ as an appendix to C;
7: $\delta + C \longrightarrow b_{N+1}$;
8: Fill up the winner set;
9: **for** $i \in I$ **do**
10: $W' \longrightarrow 0, Q \longrightarrow Q', \{s_k \mid b_k \leq b_i\} \longrightarrow N'$;
11: **end for**
12: **while** $\sum_{T_j \in T} Q' \neq 0$ **do**
13: $\text{argmax} x_{i:s_k \in N}{}' sum_{U:T_M \in T_W}$;
14: $W_i \cup S_{k\,\max} \longrightarrow W_i$;
15: $N'/S_{k\,\max} \longrightarrow N'$;
16: **end while**
17: **for** $j \in T_j$ **do**
18: $do\ Q_j' - \min\{Q_j', \Gamma_{\text{imax},j}\}$;
19: **end for**
20: **for** $b \in C \cap [b_i, b_{i+1})$ **do**
21: $W_i \longrightarrow W(p)$;
22: **end for**
23: Pick a bid called p from $[p = x] = \exp(-\partial x \mid W(x)/2NC_{\max}) / \sum_{y \in \{b_n\}} \exp(-\partial x \mid W(x)/2NC_{\max}), \forall x \in \{b_n\} W(p) \longrightarrow W$;
24: **return** $\{W, p\}$;

Algorithm 1: DP-LAL auction with Lyapunov stability.

$a_s \neq c_s$. In the proof, to simplify the description, we assume that S ranks ahead of the others after the tie-break. We prove that $u_s \geq u_s'$ for any $a_s \neq c_s$. If seller wins, we have p_s as its minmum value and $u_s = p_s x_s - v_s \geq 0$ by Lemma 6. We know that the $b_s > v_s$ will not influence the result of the auction, and it results in $u_s' = v_s - p_s' = v_s - p_s = u_s$. If he loses by bidding truthfully, his utility is $u_s = 0$. It is obvious that bidding $b_s < v_s$ will not influence the result of auction. By bidding $b_s > b_{\min}$, s becomes a winner. However, its utility becomes $u_s' = v_s - p_s = v_s - b_{\min} < 0 = u_s$. Therefore U maximizes his profit by charging truthfully. From the proof of Lemmas 5–9, we obtain that DP-LAL is computationally valid, personally justifiable, budget-balanced, and realistic.□ □

7. Simulation

In this section, we perform a wide range of simulations to simulations to show the effectiveness of our DP-LAL mechanism.

7.1. Simulation Setup. The following are the initial parameters we set:

(i) The initial differential privacy of each user is 0

(ii) The threshold of maintaining user's long-term participation is 10

(iii) The number of buyers is 10

(iv) The number of sellers is 1000

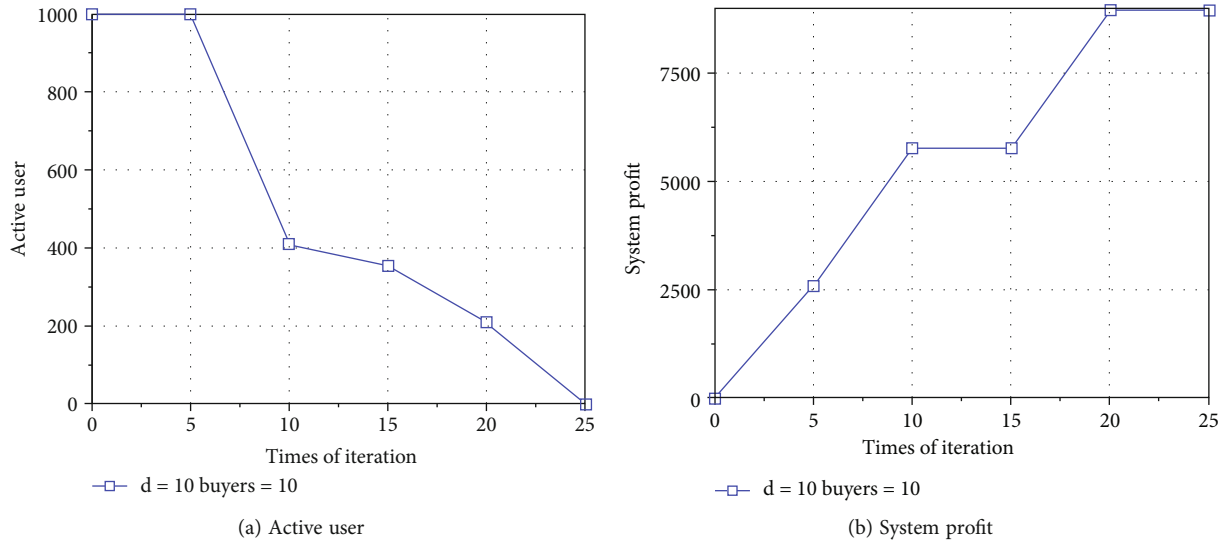
(v) The bid of the buyer user is controlled within the range of [5000, 10000]

(vi) The seller's bid is controlled within the range of [1, 50]

(vii) The simulated auction was conducted for a total of 500 rounds

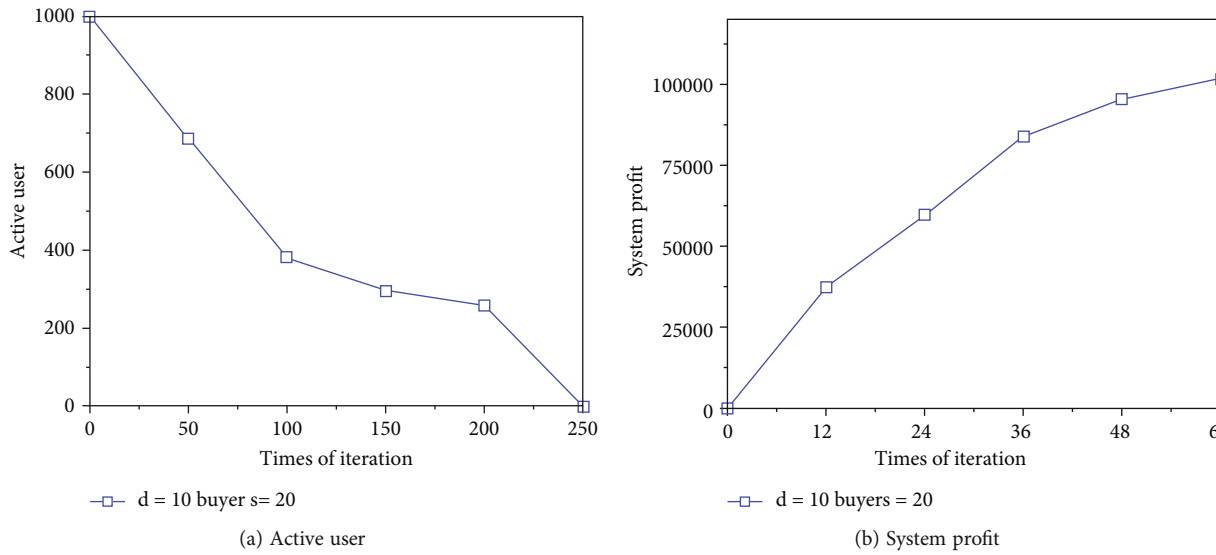
(viii) The adjustment threshold between the long-term and efficiency values of the system is 1

After the setting up the parameters, we initialize the buyer and seller variables. The seller queue is initialized to 0, and each seller's bid is set to a random number. We initialize the buyer parameter as a random number within the threshold. The buyer groups the sellers according to differential privacy, and each group of sellers is sorted by their bids, and the sellers with the highest bids in all groups are sorted using index mechanism. After that, the sensing platform looks for seller groups that offer the largest profit by traversing the bids of all sellers. If there is no group that satisfies this condition, then this group is set to −1. Then, the sensing platform charges the seller with the highest bid and searches for the highest price for the seller from the remaining buyers as the winning group for the winning group analysis:



(a) Active user (b) System profit

Figure 5: System performance when the number of buyers is 10.



(a) Active user (b) System profit

Figure 6: System performance when the number of buyers is 20.

(1) Whether the number of winning sellers meets the seller's demand

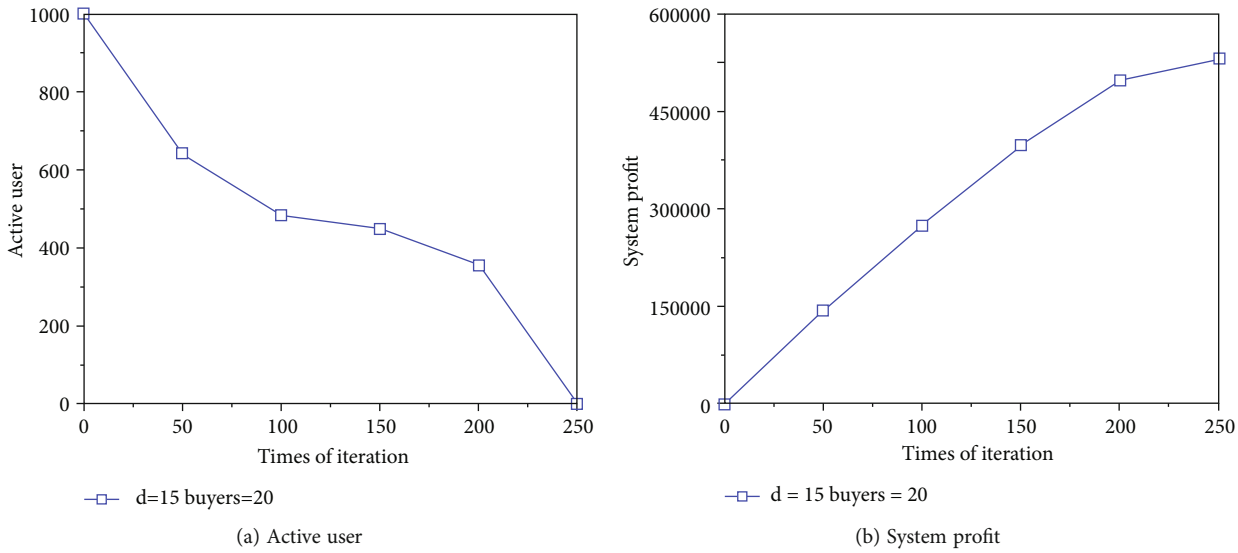
(2) Whether the buyer can pay the seller's total cost based on the above two principles

At last, the sensing platform completes the collection of buyers' fees and pays the sellers' profit. The remaining expenses are used as sensing platform's profit.

7.2. Performance Evaluation. Firstly, we introduce a baseline to demonstrate the effectiveness of the differential private long-term privacy-preserving auction with Lyapunov (DP-LAL). The baseline is static auction without long-term participation constraints; platform is only aimed at minimizing its total payment at current slot. In the static auction, platform chooses the winner users by their accuracy bidding ratio and pay them with their critical payments at each time slot.

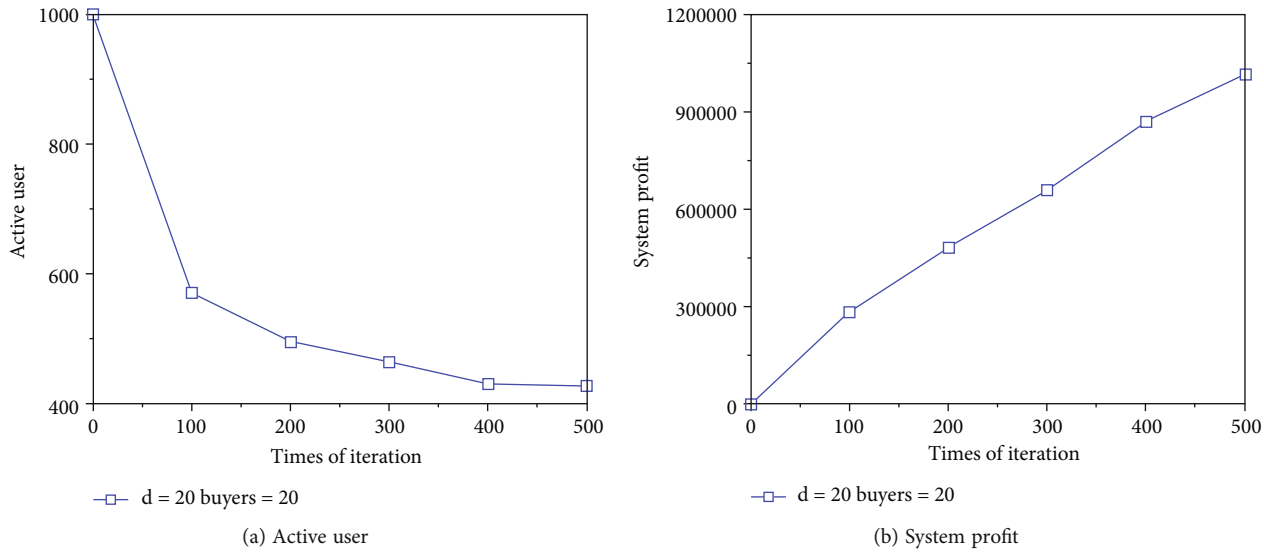
We use the matplotlib to simulate the auction results and compare the simulation results we obtained with those obtained by traditional auction algorithms. The comparison results are shown in the figures below.

We perform 100 simulated auctions and gather the simulation results. However, we find out that there is a sudden drop of user number after 10 auctions in Figure 5(a), which means that most of the users left without winning a single auction and most auctions are failed. We can verify it by Figure 5(b), which shows the profit of the system, and we can see that there are only a few successful auctions. Then, we set buyers to 20, as shown in Figures 6(a) and 6(b); it is obvious that the system works stabler but not stable enough, and more buyers will make it easier for users to win in an auction and thus maintaining the stability of the system.



(a) Active user (b) System profit

Figure 7: System performance when d is 15.



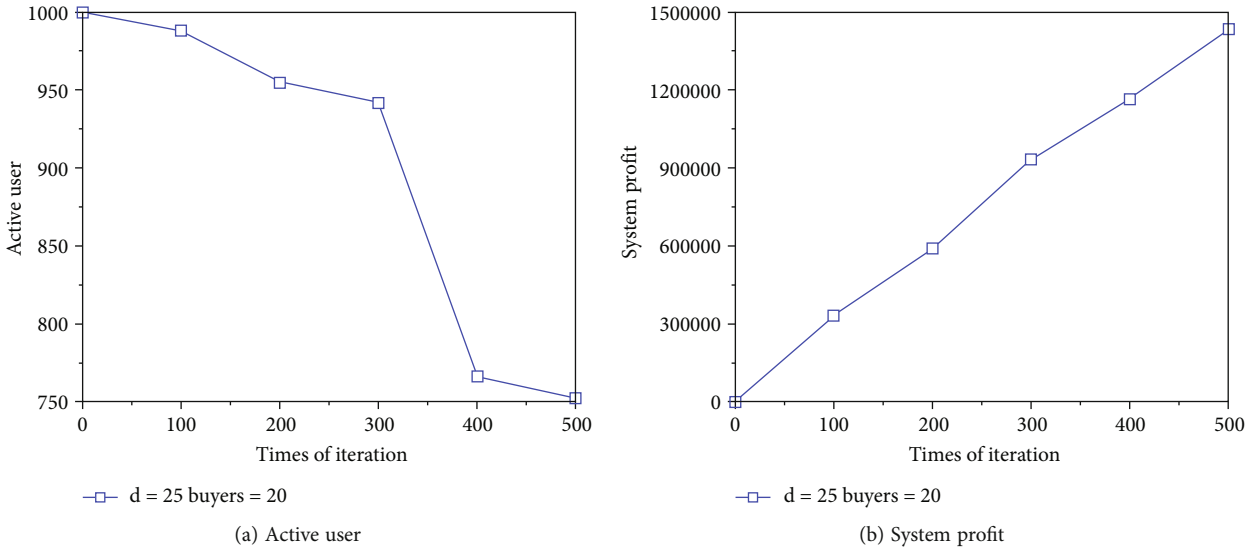
(a) Active user (b) System profit

Figure 8: System performance when d is 20.

Now we set buyers to 20 to make sure the system is stable. Then, we consider parameter d to estimate user's long-term participation in our algorithm and how it will affect the stability of the system. We run the auction 500 times this time and change the value of d several times. At first, we set d to 15, which means a user can lose at most 15 auctions before he quit participating in an auction; we can see from Figure 7(a) that all of the users leave when the number of simulations reaches about 250 times. Obviously, the impact of parameter d is deeper than the number of buyers. Then, we add d to 20; as shown in Figure 8(a), half of the users leave when it reaches 500 times, but the situation is better since there are still users left in the system after 500 auctions. But comparing to Figure 7(b), we can learn from Figure 8(b) that when d is 20, the system makes almost twice the profit when d is 15. Figure 9(a) shows that only about 200 users leave after 500 auctions when d is 25; we perform more auctions and less than 50 users will leave if d is up to 30. When d is larger than 30, nobody will quit the system. We also draw the conclusion from Figures 8(b) and 9(b) that, the larger d, the faster the system makes profit. It is obviously that we can guarantee user's long-term participation, which means no user will quit the system, with a winning rate of proximately 93.5%.

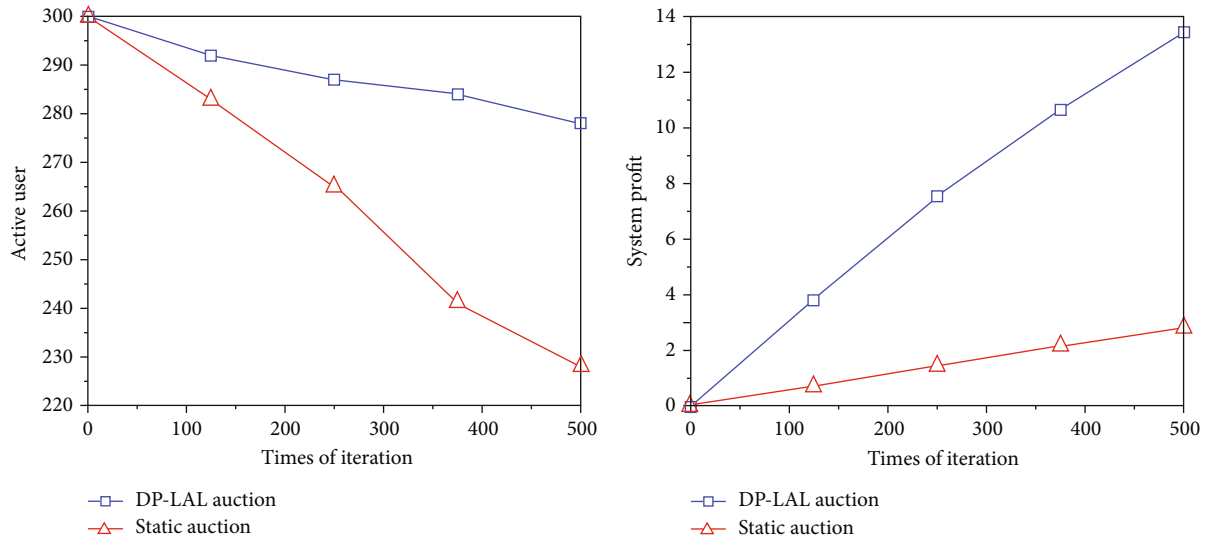
The comparison results are shown in Figure 10. Figure 10(a) simulates the amount of active users after multiple auctions. Figure 10(b) simulates the profit of the system. With the growing of number of simulations, the number of users leave in our system is much more than those in a static auction; the system profit is also increasing at a faster rate. This proves that our algorithm guarantees a better return for users after multiple auctions.

Since our algorithms need to consider long-term participation, real bidding, tie-breaking, and individual rationality,



(a) Active user (b) System profit

Figure 9: System performance when d is 25.



(a) Active user (b) System profit

Figure 10: Comparing to static auction algorithm.

there is no guarantee that the DP-LAL algorithm will achieve the theoretical maximum benefit. However, according to our analysis, our algorithm can maintain the advantage of the number of users compared to the static algorithm after multiple auctions and can choose higher bids and better yield among more users.

We can draw a conclusion that the static algorithm does not take into account the long-term involvement of users, and many users will quit the system after they have not been selected as winners for a long time. On the whole, our algorithm better solves the problem of maintaining long-term user participation.

8. Conclusion

In this paper, we design a differential private long-term privacy preserving auction with Lyapunov (DP-LAL) auction, which is used to perform sensing task release, auction, and task completion. Our algorithm considers long-term participation constraints and user differential privacy, which is essential for motivating participation in real-time data aggregation. Due to the combination of the user's strategic behavior and task nature, we propose a computational efficiency mechanism with near-optimal performance to jointly optimize the participation of users' payment and sensing platforms and ensure that users are encouraged to participate for a long time. In addition, we ensure that the proposed auction meets other desirable attributes, including authenticity and truthfulness. The validity of the proposed auction is verified by theoretical analysis and extensive simulation. At present, our algorithm has been able to guarantee the user's motivation to participate in the auction for a long time, and to ensure that after multiple rounds of auction, the

average profit of the user is much larger than an existing algorithm.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by Research Innovation Fund for College Students of Beijing University of Posts and Telecommunications.

References

[1] Z. Ning, S. Sun, X. Wang et al., "Blockchain-enabled intelligent transportation systems: a distributed crowdsensing framework," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[2] S. Lin, J. Zhang, and L. Ying, "Crowdsensing for spectrum discovery: a waze-inspired design via smartphone sensing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 750–763, 2020.

[3] D. Kim, J. Jung, Y. Koo, and Y. Yi, "Bird-mac: energy-efficient mac for quasi-periodic IoT applications by avoiding early wake-up," *IEEE Transactions on Mobile Computing*, vol. 19, no. 4, pp. 788–802, 2020.

[4] B. Niu, Y. Chen, Z. Wang, F. Li, B. Wang, and H. Li, "Eclipse: preserving differential location privacy against long-term observation attacks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 125–138, 2022.

[5] Y. Ruan, Y. Li, C.-X. Wang, R. Zhang, and H. Zhang, "Energy efficient power allocation for delay constrained cognitive satellite terrestrial networks under interference constraints," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4957–4969, 2019.

[6] P. Huang, X. Zhang, L. Guo, and M. Li, "Incentivizing crowdsensing-based noise monitoring with differentially-private locations," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 519–532, 2021.

[7] Z. Wang, J. Hu, R. Lv et al., "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2019.

[8] Z. Wang, X. Pang, Y. Chen et al., "Privacy-preserving crowdsourced statistical data publishing with an untrusted server," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1356–1367, 2019.

[9] H. Wu, L. Wang, G. Xue, J. Tang, and D. Yang, "Enabling data trustworthiness and user privacy in mobile crowdsensing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2294–2307, 2019.

[10] C. Cai, Y. Zheng, A. Zhou, and C. Wang, "Building a secure knowledge marketplace over crowdsensed data streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, 2019.

[11] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. S. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1317–1331, 2020.

[12] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 581–594, 2020.

[13] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif, "Reliable and privacypreserving truth discovery for mobile crowdsensing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1245–1260, 2021.

[14] G. Yang, Z. Shi, S. He, and J. Zhang, "Socially privacy-preserving data collection for crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 851–861, 2019.

[15] J. Nie, J. Luo, Z. Xiong, D. Niyato, and P. Wang, "A stackelberg game approach toward socially-aware incentive mechanisms for mobile crowdsensing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 724–738, 2019.

[16] J. Nie, J. Luo, Z. Xiong, D. Niyato, P. Wang, and H. V. Poor, "A multileader multi-follower game-based analysis for incentive mechanisms in socially-aware mobile crowdsensing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1457–1471, 2021.

[17] J. Liu, Y. Yang, D. Li, deng, S. Huang, and H. Liu, "An incentive mechanism based on behavioural economics in location-based crowdsensing considering an uneven distribution of participants," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 44–62, 2022.

[18] P. Sun, Z. Wang, L. Wu et al., "Towards personalized privacy-preserving incentive for truth discovery in mobile crowdsensing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 352–365, 2022.

[19] J. Hu, Z. Wang, J. Wei et al., "Towards demand-driven dynamic incentive for mobile crowdsensing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4907–4918, 2020.

[20] Z. Wang, J. Li, J. Hu et al., "Towards privacy-driven truthful incentives for mobile crowdsensing under untrusted platform," *IEEE Transactions on Mobile Computing*, 2021.

[21] Y. Zhan, C. H. Liu, Y. Zhao, J. Zhang, and J. Tang, "Free market of multi-leader multi-follower mobile crowdsensing: an incentive mechanism design by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2316–2329, 2020.

[22] Z. Duan, W. Li, X. Zheng, and Z. Cai, "Mutual-preference driven truthful auction mechanism in mobile crowdsensing," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1233–1242, Dallas, TX, USA, 2019.

[23] X. Chen, L. Zhang, Y. Pang, B. Lin, and Y. Fang, "Timeliness-aware incentive mechanism for vehicular crowdsourcing in smart cities," *IEEE Transactions on Mobile Computing*, 2021.

[24] S. Fu, J. Gao, and L. Zhao, "Collaborative multi-resource allocation in terrestrial-satellite network towards 6G," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7057–7071, 2021.

[25] Z. Jia, M. Sheng, J. Li, D. Zhou, and Z. Han, "VNF-based service provision in software defined LEO satellite networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 6139–6153, 2021.

[26] X. Fang, W. Feng, T. Wei, Y. Chen, N. Ge, and C.-X. Wang, "5G embraces satellites for 6G ubiquitous IoT: basic models

for integrated satellite terrestrial networks," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 14399–14417, 2021.

[27] B. Di, H. Zhang, L. Song, Y. Li, and G. Y. Li, "Ultra-dense LEO: integrating terrestrial-satellite networks into 5G and beyond for data offloading," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 47–62, 2019.

[28] J. Li, K. Xue, D. S. L. Wei, J. Liu, and Y. Zhang, "Energy efficiency and traffic offloading optimization in integrated satellite/terrestrial radio access networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2367–2381, 2020.

[29] L. Wang, Y. Wu, H. Zhang, S. Choi, and V. C. M. Leung, "Resource allocation for NOMA based space-terrestrial satellite networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1065–1075, 2021.

[30] D. Jiang, F. Wang, Z. Lv et al., "QoE-aware efficient content distribution scheme for satellite-terrestrial networks," *IEEE Transactions on Mobile Computing*, p. 1, 2021.

[31] H. Huang, S. Guo, W. Liang, K. Wang, and Y. Okabe, "Coflow-like online data acquisition from low-earth-orbit datacenters," *IEEETransactions on Mobile Computing*, vol. 19, no. 12, pp. 2743–2760, 2020.

[32] X. Li, W. Feng, Y. Chen, C.-X. Wang, and N. Ge, "Maritime coverage enhancement using UAVs coordinated with hybrid satellite-terrestrial networks," *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2355–2369, 2020.

[33] C. Lin, J. Zhou, C. Guo, H. Song, G. Wu, and M. S. Obaidat, "TSCA: a temporal-spatial real-time charging scheduling algorithm for ondemand architecture in wireless rechargeable sensor networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 211–224, 2018.

[34] C. Lin, Z. Yang, H. Dai, L. Cui, L. Wang, and G. Wu, "Minimizing charging delay for directional charging," *IEEE/ACM Transactions on Networking*, vol. 29, no. 6, pp. 2478–2493, 2021.

[35] Y. Sun, C. Lin, H. Dai, Q. Lin, L. Wang, and G. Wu, "Trading off charging and sensing for stochastic events monitoring in WRSNs," *IEEE/ACM Transactions on Networking*, 2020.

[36] C. Lin, Z. Wang, J. Deng, L. Wang, J. Ren, and G. Wu, "mTS: temporal-and spatial-collaborative charging for wireless rechargeable sensor networks with multiple vehicles," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 99–107, Honolulu, HI, USA, 2018.

[37] C. Lin, F. Gao, H. Dai, J. Ren, L. Wang, and G. Wu, "Maximizing charging utility with obstacles through Fresnel diffraction model," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 2046–2055, Toronto, ON, Canada, 2020.

[38] C. Lin, Z. Shang, W. Du, J. Ren, L. Wang, and G. Wu, "Codoc: a novel attack for wireless rechargeable sensor networks through denial of charge," in *IEEE INFOCOM 2019- IEEE Conference on Computer Communications*, pp. 856–864, Paris, France, 2019.

[39] C. Lin, Y. Yu, J. Xiong et al., "Shrimp: a robust underwater visible light communication system," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, p. 134146, New Orleans, United States, 2021.

[40] B. Zhu, J. Li, Z. Liu, and Y. Liu, "A privacy-preserving incentive mechanism in satellite-terrestrial crowdsensing," in *IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2021.

[41] W. Aljoby, X. Wang, T. Z. J. Fu, and R. T. B. Ma, "On SDN-enabled online and dynamic bandwidth allocation for stream analytics," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1688–1702, 2019.

[42] A. Asheralieva and D. Niyato, "Combining contract theory and Lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1213–1226, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5599155, 10 pages
https://doi.org/10.1155/2021/5599155

Research Article

CrowdBox: Crowdsourced Network-in-Box Recruitment for Edge Computing-Enabled Industrial Internet of Things

Pengfei Wang,[1] **Chi Lin**,[2] **Zhen Yu**,[1] **Leyou Yang**,[3] **and Qiang Zhang**[1]

[1]*School of Computer Science and Technology, Dalian University of Technology, Dalian, 116024 Liaoning, China*
[2]*School of Software Technology, Dalian University of Technology, Dalian, 116600 Liaoning, China*
[3]*School of Computer Science and Engineering, Northeastern University (NEU), Shenyang, 110819 Liaoning, China*

Correspondence should be addressed to Pengfei Wang; wangpf@dlut.edu.cn and Qiang Zhang; zhangq@dlut.edu.cn

Received 22 August 2021; Revised 10 November 2021; Accepted 17 November 2021; Published 13 December 2021

Academic Editor: Enrico M. Vitucci

The rapidly increasing number of smart devices deployed in the Industrial Internet of Things (IIoT) environment has been witnessed. To improve communication efficiency, edge computing-enabled Industrial Internet of Things (E-IIoT) has gained attention recently. Nevertheless, E-IIoT still cannot conquer the rapidly increasing communication demands when hundreds of millions of IIoT devices are connected at the same time. Considering the future 6G environment where smart network-in-box (NIB) nodes are everywhere (e.g., deployed in vehicles, buses, backpacks, etc.), we propose a crowdsourcing-based recruitment framework, leveraging the power of the crowd to provide extra communication resources and enhance the communication capabilities. We creatively treat NIB nodes as edge layer devices, and CrowdBox is devised using a Stackelberg game where the E-IIoT system is the leader, and the NIB nodes are the followers. CrowdBox can calculate the optimal reward to reach the unique Stackelberg equilibrium where the utility of E-IIoT can be maximized while none of the NIB nodes can improve its utility by deviating from its strategy. Finally, we evaluate the performance of CrowdBox with extensive simulations with various settings, and it shows that CrowdBox outperforms the compared algorithms in improving system utility and attracting more NIB nodes.

1. Introduction

The Industrial Internet of Things (IIoT), an important part of IoT [1], brings Internet access to all industrial assets including industrial devices and control systems [2]. As one of the basic pillars of digital manufacturing, IIoT changes the traditional manufacturing lines and systems significantly [3]. With the flexibility and scalability brought by wireless links [4], it paves the way for efficient and sustainable production in the era of Industry 4.0.

However, the emergence of IIoT applications results in a tenfold increase in the amount of data generated by industrial devices [5]. It would be a disaster for the network if that size of data which is produced at the edge of the network is transferred to the cloud directly. Not only the data volume but also traditional cloud computing is unable to deal with the various requirements such as data privacy and response time limitation. So, it will be more efficient to process the data at the edge of the network.

Edge computing (EC) is a new architecture [6–8] extending the cloud paradigm to the network edges, and it has attracted attention from both academia and industry recently since a large number of critical problems in IIoT (e.g., resource limitation [9], latency [10], and privacy [11]) could be solved with it easily. The combination of EC and IIoT, the so-called edge computing-enabled Industrial Internet of Things (E-IIoT), enables the design of high-performance and adaptive systems for IIoT [12]. Therefore, many edge computing-enabled new IIoT architectures and platforms have been proposed and devised to improve the service efficiency [13–15] from different aspects in the last few years.

With the rapid development of wireless communication technologies (e.g., the sixth-generation communication

technology) and new urban infrastructures (e.g., personal communication devices, edge servers, and smart vehicles), many new smart devices will emerge. Chen et al. [16] believe that 6G will merge computation and sensing with communications, so the functions of new smart devices will be more complex. In addition, the strong mobility of edge nodes has become the trend of development [17].

Network-in-box [18, 19] is a burgeoning technology to enhance network reliability, and its application significance is becoming more and more obvious with the development of edge computing and sixth-generation communication techniques. The design of NIB is focused on portability, so the NIB can move according to the requirement of the whole network. An NIB node can work alone as well as cooperate with other network elements. Some typical use cases of NIB including after-disaster scenario, connectivity provisioning in challenging contexts, and tactical networks are investigated in [20]. With the NIB node as the edge server, a more reliable and more flexible networking environment can be provided for the E-IIoT system.

1.1. Motivation. Crowdsourcing has been proven to be effective in task delivering applications including Amazon Mechanical Turks (AMT) [21–23]. The requirements for communication resources of devices can be regarded as a task, and NIB is responsible for fulfilling the task. In this paper, we propose to utilize the power of crowdsourced NIB nodes to improve communication capabilities at the edge layer. Although NIB brings great flexibility to the E-IIoT system, it is still an urgent problem to motivate NIB nodes to provide additional communication resources for devices. In order to better understand the problem, we build a Stackelberg game between the E-IIoT system and NIB nodes. By studying the Nash equilibrium, we can find the best response strategy of an NIB node, and the incentive problem could be solved at the same time.

1.2. Contributions. In this paper, we establish a crowdsourcing-based network-in-box recruitment platform. As far as we know, our work is the first to conceive a crowdsourcing-based recruitment platform in the E-IIoT environment. The main contributions of this paper are summarized as follows:

(i) We present an analytical model for the recruitment of NIB nodes where they can improve the system communication capabilities by crowdsourcing

(ii) For the recruitment problem, the utility function for the system and NIB nodes are formulated; then, the whole process is described as a Stackelberg game with two stages

(iii) The Nash equilibrium is defined, and CrowdBox is proposed to calculate the Nash equilibrium point. Finally, the best strategy of NIB nodes and E-IIot are both found

(iv) To prove the effectiveness and feasibility of CrowdBox, we both put forward the theoretical explanation and conduct extensive simulations. It turns out that CrowdBox achieves better performance than other benchmarks

1.3. Organization of the Paper. The rest of this paper is organized as follows. Section 2 discusses the related work. We introduce the system model of our network and the formulated problem in Section 3. Section 4 presents the definition of CrowdBox and details of the whole game process. Evaluation settings and numerical results are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

By leveraging the concept of crowdsourcing, many IIoT problems have been solved. The author in [24] emphasizes the impressive amounts of data by introducing crowdsourcing to IIoT which provides the opportunity to perform more advanced processes and applications. Also, a mobile crowdsensing and mobile crowdsourcing-based IIoT architecture is proposed in their work. In the field of the smart city which is an important application scenario of IIoT, Kong et al. [25] utilize crowdsourcing to collect and compute decentralized ubiquitous sensing data. The objective is to solve major urbanization problems in smart cities. As crowdsourcing is widely used for data collection and computing, privacy protection in this process has also become a direction of crowdsourcing research. A personalized privacy protection framework is proposed for mobile crowdsensing in [26]. Theoretical analysis and simulations show that the framework can make a balance between the quality of crowdsensing services and privacy. When the crowdsourcing IIoT data is stored in the cloud, Karati et al. [27] propose a new identity-based signcryption schema to meet the requirement of authenticity and confidentiality.

From the perspective of IIoT devices, resources from the edge servers are shared to enhance the ability of the whole network. So, our work is also somewhat similar to resource allocation in the E-IIoT system, and we also summarize it briefly. In [28], mobile edge computation offloading is considered on different multiple access technologies. Resource allocation is formulated as an optimization problem with mobile energy consumption as the constraint. Sun etal. [29] discuss the joint optimization problem of network economics and resource allocation in mobile edge computing. They propose two double auction schemes which both use dynamic pricing to establish connections of IIoT devices and edge servers. To improve the cognitive ability of edge intelligent IIoT, an ML-enabled framework is proposed in [30]. Via this framework, IIoT is able to make reasonable decisions at the network edge. Furthermore, deep reinforcement learning is invoked and shows good performance in an observable IIoT environment.

To realize the objective of optimization, game-based approaches have been applied in communication networks [31]. The Stackelberg game is a typical one to design interactive schemes in wireless networks. When the problem comes to resource allocation, Zhang etal. [32] formulate a Stackelberg game to analyze the problem and apply a matching

game to achieve satisfying performance. Yao et al. [33] invoke the Stackelberg game to model the interaction between the resource provider and miners. In [34], Jie et al. transfer the resource allocation problem to a double-stage Stackelberg game to maximize resource utilization. The Stackelberg game is also used in [35] to offload computation in IIoT, and the existence and uniqueness of equilibrium are analyzed. To sum up, the insight of crowdsourcing can help us formulate the problem as a recruitment problem. After that, the Stackelberg game can be utilized to motivate the NIB nodes to share their resources and make sure the IIot devices get the resources they need.

3. System Model and Problem Formulation

This section first introduces the system model of the edge computing-enabled Industrial Internet of Things (E-IIoT) system, then formulates the crowdsourcing-based network-inbox recruitment problem as a Stackelberg game.

3.1. System Model. Figure 1 is utilized to aid our description of the E-IIoT system. Similar to the other edge computing systems, E-IIoT mainly includes three layers—the cloud layer, the edge layer, and the device layer. To fulfill industrial missions from factories, E-IIoT devices have to upload their sensing data to edge/cloud nodes, and the following instructions will be directed to E-IIoT devices after the uploaded sensing data are processed by edge and cloud servers.

However, the existing communication resource at the edge layer may not be enough to satisfy the E-IIoT system when some edge nodes are broken down or sudden communication demands (e.g., factory adds many IIoT at the same time suddenly) are required. To overcome these difficulties, we propose to recruit crowdsourcing network-in-box nodes (e.g., box in buses, vehicles, packages, etc.) as temporary edge nodes to expand the network and offload sensing data from IIoT devices to servers and meet the communication needs finally.

Here, we treat the communication resource collection of the NIB nodes for E-IIoT as a task. To be specific, different from the general NIB deployment system where NIB nodes contribute their communication resource passively, the enterprise (i.e., resource consumer) could buy communication resources from participating NIB nodes with the total reward within a specific time period (e.g., one hour). Further, the NIB nodes could also select their participating levels (i.e., how long to contribute their communication resource) when using CrowdBox during daily life. The NIB nodes will select to participate only when the received reward could cover its cost in both data communication and data processing. Eventually, the NIB nodes could obtain the corresponding reward based on the participating levels of the total reward of the system and all NIB nodes.

In a word, crowdsourcing NIB nodes will have some costs when they provide communication resources to the system, and E-IIoT will offer some corresponding rewards to stimulate crowdsourcing E-IIoT nodes who perform the communication task and provide the communication resource. Actually, this is a gaming process, and we will formulate it with the Stackelberg gaming theory in the next subsection.

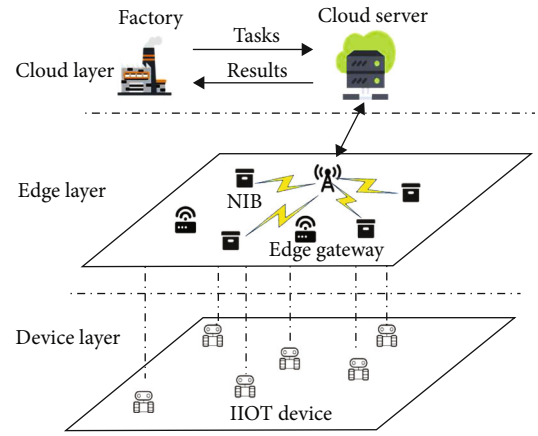


Figure 1: The E-IIoT architecture.

3.2. Problem Formulation. The E-IIoT system has one communication task (i.e., provide communication resources as more as possible) for all NIB nodes at the edge layer. To recruit NIB nodes fulfilling the communication task, the E-IIoT system announces a total reward R, where $R > 0$. According to the reward, each participant will choose their own participation level. To be specific, the participation level (strategy) of NIB node i is $t_i (\geq 0)$ which denotes the time duration it can provide the communication service. It is assumed that all NIB nodes are available when they choose to participate in the task, and they contribute equally at any time during the task period. The unit cost of NIB node i is denoted as c_i, and we can compute the utility of NIB node i as

$$u_i = \frac{t_i}{\sum_{j \in U} t_j} R - c_i t_i, \tag{1}$$

where $t_i / \sum_{j \in U} t_j R$ is the reward that NIB node i will get after participating in the task and $c_i t_i$ is its total cost for completing the task. It should be noticed that a rational NIB node will not cooperate with a negative utility, so NIB node i will do nothing (i.e., $t_i = 0$) when $t_i / \sum_{j \in U} t_j R < c_i t_i$. Thus, the equation should be rewritten as

$$u_i = \begin{cases} \dfrac{t_i}{\sum_{j \in U} t_j} R - c_i t_i, & \dfrac{R}{\sum_{j \in U} t_j} > c_i, \\ 0, & \dfrac{R}{\sum_{j \in U} t_j} \leq c_i. \end{cases} \tag{2}$$

According to equation (2), it is not hard to find that NIB nodes would like to choose the same plan (strategy) to maximize their utility when they have the same unit cost. Here, we define the unit cost set $S = \{s_1, s_2, \cdots, s_X\}$, and we have $c_i \in S$. All NIB nodes with the same unit cost will choose the same strategy if they want to maximize their utilities.

Therefore, the expected participation time duration for a specific unit cost s_x is denoted by $\hat{t}_x(x \in X)$, and we also have the expected participation time duration set $\hat{T} = \{\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_X\}$ correspondingly. Besides, the number of NIB nodes whose unit costs are the same is defined as $n_x(x \in X)$, and the set of the number of NIB nodes is $N = \{n_1, n_2, \cdots, n_X\}$. Based on the above definitions, the utility function of E-IIoT can be computed as follows:

$$u_0 = f\left(\hat{T}, N\right) - R, \tag{3}$$

where $f(\hat{T}, N)$ presents the valuation function of all NIB nodes' participation time of the E-IIoT system.

The goal of E-IIoT is to choose the optimal value of reward R to maximize equation (3). It is assumed that the utility function of E-IIoT is a strictly concave function in variables T for any fixed N and increasing monotonically for every t_i of NIB node i, and this is a general assumption in many other related papers [36]. Each NIB node $i \in U$ decides its strategy (i.e., t_i) to maximize equation (2) with a given reward value R.

In the next section, we will model the whole NIB node recruitment process as a *Stackelberg game* with two stages and depict how to determine the optimal reward value R^* to maximize the utility of the E-IIoT system. Besides, detailed explanations of employed notations in this paper are concluded in Table 1.

Table 1: Major notations employed in this paper.

Notation	Explanation
R	Reward of the E-IIoT system
R^*	Optimal reward value
R_{others}	Feasible reward values
U	Crowdsourcing NIB node set
U_i	NIB nodes except i
u_i	Utility function of NIB node i
u_0	Utility function of the E-IIoT system
n	Number of crowdsourcing NIB nodes
t	Strategy profile of all users
t_{-i}	Strategy profile of all users except NIB node i
t_i	Participation level (strategy) of NIB node i
c_i	Unit cost of NIB node i
S	Unit cost level set
X	Number of unit cost levels
s_x	xth unit cost level
$\hat{T}$	Expected participation time duration set
$\hat{t}_s$	Expected participation time duration for s_x
N	Set of the NIB node number at different unit cost levels
n_x	Number of NIB nodes at the same unit cost level

4. CrowdBox

The crowdsourcing-based network-in-box recruitment for edge computing-enabled Industrial Internet of Things process is modeled as a Stackelberg game (CrowdBox), and it has two stages. In the first stage, the E-IIoT system posts its reward R for the communication task and recruit NIB nodes to participate. Accordingly, the NIB node chooses a response strategy (i.e., service time duration it could provide) to maximize its own utility based on the given reward R in the second stage. Essentially, the E-IIoT system is the leader and the NIB nodes are the followers.

The Nash equilibrium is a stable status and very important for a game since no players can obtain extra profits by changing their strategy. For the NIB node, we first define the Nash equilibrium and its best response strategy. We prove that NIB nodes have a unique Nash equilibrium for any given reward R. Based on the utility function of the NIB node, we analyze the best response strategy of NIB nodes and show how to compute it. For the E-IIoT system, we also define its Nash equilibrium and the corresponding best response strategy and propose how to calculate the optimal reward R^* (i.e., the best response strategy for the E-IIoT system) to maximize the E-IIoT system's utility.

In sum, we show that the Stackelberg game we proposed has a unique Stackelberg equilibrium and how to calculate the optimal reward value to maximize E-IIoT's utility in this section.

4.1. NIB Node Nash Equilibrium. To study the NIB node strategy, we first define the Nash equilibrium of NIB node i as below.

Definition 2 (Nash equilibrium of NIB nodes). A strategy set $(t_1^{ne}, t_2^{ne}, \cdots, t_n^{ne})$ is a Nash equilibrium, when any NIB node i meets equation (18):

$$u(t_i^{ne}, t_{-i}^{ne}) \geq u(t_i, t_{-i}^{ne}). \tag{4}$$

We also give the definition of the best response strategy for NIB node i as the following.

Definition 3 (best response strategy of NIB nodes). Strategy t_i for NIB node i, denoted by B_i, is the best response strategy if it maximizes $u(t_i, t_{-i})$ over all $t_i \geq 0$.

Given a reward R from the E-IIoT system, NIB node i would like to choose the best response strategy to maximize its utility (i.e., equation (2)). An NIB node would like to play its best response strategy in Nash equilibrium.

To study the best response strategy of NIB node i, we compute the derivatives and second order of utility function u_i with respect to t_i as

$$\frac{\partial u_i}{\partial t_i} = \frac{R}{\sum_{j \in U} t_j} - \frac{t_i R}{\left(\sum_{j \in U} t_j\right)^2} - c_i, \tag{5}$$

$$\frac{\partial^2 u_i}{\partial t_i^2} = -\frac{2Rt_i}{\left(\sum_{j\in U} t_j\right)^3}. \tag{6}$$

It is very obvious that the second-order derivative $\partial^2 u_i/\partial t_i^2 \le 0$, so the utility function of NIB node i is a strictly concave function. By setting the first-order derivative $\partial u_i/\partial t_i = 0$, we have

$$\frac{R}{\sum_{j\in U} t_j} - \frac{t_i R}{\left(\sum_{j\in U} t_j\right)^2} - c_i = 0. \tag{7}$$

Sorting out equation (7), we can obtain

$$t_i = \sqrt{\frac{\left(\sum_{j\in U_{-i}} t_j\right)R}{c_i}} - \sum_{j\in U_{-i}} t_j. \tag{8}$$

Obviously, the best response strategy of node i can be summarized as

$$B_i = \begin{cases} \sqrt{\dfrac{\left(\sum_{j\in U_{-i}} t_j\right)R}{c_i}} - \sum_{j\in U_{-i}} t_j, & R \ge c_i \times \sum_{j\in U_{-i}} t_j, \\ 0, & R < c_i \times \sum_{j\in U_{-i}} t_j. \end{cases} \tag{9}$$

The above analysis shows that any NIB node $i(0 < i < n)$ has its best response strategy B_i for any given reward $R > 0$ and strategy profile t_{-i} of other NIB nodes. Next, we define the NIB node set $\tilde{U} = \{i \in U \mid t_i > 0\}$, and it is easy to get that $\sum_{j\in\tilde{U}} t_j = \sum_{j\in U} t_j$. Therefore, we can get

$$\frac{R}{\sum_{j\in\tilde{U}} t_j} - \frac{t_i R}{\left(\sum_{j\in\tilde{U}} t_j\right)^2} - c_i = 0. \tag{10}$$

Adding up equation (10) over all NIB nodes in $\tilde{U}$, we have

$$-R + \left|\tilde{U}\right| \times R - \left(\sum_{j\in\tilde{U}} k_j\right) \times \left(\sum_{j\in\tilde{U}} t_j\right) = 0. \tag{11}$$

Therefore, we obtain

$$\sum_{j\in\tilde{U}} t_j = \frac{\left(\left|\tilde{U}\right| - 1\right)R}{\sum_{j\in\tilde{U}} k_j}. \tag{12}$$

To calculate t_i for NIB node i, equation (10) can also be rewritten as

$$t_i = \sum_{j\in\tilde{U}} t_j - \frac{k_i}{R} \times \left(\sum_{j\in\tilde{U}} t_j\right)^2. \tag{13}$$

Substituting equation (12) into equation (13), we can calculate the best response strategy for NIB node i as

$$t_i = \frac{\left(\left|\tilde{U}\right| - 1\right)R}{\sum_{j\in\tilde{U}} k_j} - k_i \times R \times \left(\frac{\left(\left|\tilde{U}\right| - 1\right)}{\sum_{j\in\tilde{U}} k_j}\right)^2. \tag{14}$$

In addition, the unit cost set is denoted as $\tilde{C} = \{\tilde{c}_1, \tilde{c}_2, \cdots, \tilde{c}_l\}$, where $c_i \in \tilde{C}$ for any NIB node i. The distribution of $\tilde{C}$ usually could be learned from the historical data by the E-IIoT system, and it is also a realistic assumption adopted in the literature [37]. According to equation (14), we can compute the Nash equilibrium of NIB nodes.

So far, we have known how to calculate the Nash equilibrium for any NIB node i with any given reward value R in the E-IIoT system. In the next subsection, we will show how to choose the optimal reward value R^* to maximize the E-IIoT system's utility.

4.2. Maximizing Utility of E-IIoT. Obviously, the E-IIoT system, which is the leader of the Stackelberg game, can know the existing NIB node Nash equilibrium based on the above analysis. To be specific, E-IIoT can select the optimal reward value R to maximize its utility based on the following equations:

$$u_0 = \left(\frac{f\left(\hat{T}, N\right)}{R - 1}\right)R, \tag{15}$$

where

$$\begin{aligned} \hat{T} &= \{t_1^{ne}, t_2^{ne}, \cdots, t_X^{ne}\}, \\ N &= \{n_1, n_2, \cdots, n_X\}. \end{aligned} \tag{16}$$

For each $t_i^{ne} \in \hat{T}$, we have

$$\frac{t_i^{ne}}{R} = \frac{\left(\left|\tilde{U}\right| - 1\right)}{\sum_{j\in\tilde{U}} k_j} - k_i \times \left(\frac{\left(\left|\tilde{U}\right| - 1\right)}{\sum_{j\in\tilde{U}} k_j}\right)^2, \tag{17}$$

where u_0 is a strictly concave function, $\hat{T}$ could be obtained by Algorithm 1, and N could be calculated according to the previous statistics by analyzing the historical data.

Next, we define the Nash equilibrium of the E-IIoT system as below.

Definition 4 (Nash equilibrium of the E-IIoT system). The chosen reward value R is a Nash equilibrium of the E-IIoT system if it can make the E-IIoT system reach the maximum utility.

Besides, the best response strategy of the E-IIoT system is defined as below.

Definition 5 (best response strategy of the E-IIoT system). The optimal reward value R^* is the best response strategy

Input: E-IIoT Reward R, NIB node set U and $\tilde{U}$.
Output: Nash equilibrium strategy set for all NIB nodes $t^n e = \{t_1^{ne}, t_2^{ne}, \cdots, t_n^{ne}\}$
1: $t^{ne} \longleftarrow \Phi$;
2: **for** $i \longleftarrow 1$ to n **do**
3: $\quad t_{tmp} \longleftarrow (|\tilde{U}| - 1)R/\sum_{j\in\tilde{U}} k_j - k_i \times R \times ((|\tilde{U}| - 1)/\sum_{j\in\tilde{U}} k_i)^2$;
4: $\quad$ **if** $t_{tmp} > 0$ **then**
5: $\quad\quad t_i^{ne} \longleftarrow t_{tmp}$;
6: $\quad$ **else**
7: $\quad\quad t_i^{ne} \longleftarrow 0$;
8: $\quad$ **end if**
9: $\quad t^{ne} \longleftarrow t^{ne}.add(t_i^{ne})$;
10: **end for**
11: **return** t^{ne};

Algorithm 1: Computing the Nash equilibrium of all NIB nodes.

with the given NIB node Nash equilibrium strategy set $(t_1^{ne}, t_2^{ne}, \cdots, t_n^{ne})$, and R^* meets

$$u_0(R^*) \geq u_0(R_{\text{others}}). \tag{18}$$

In summary, it can be concluded that the Nash equilibrium exists in this Stackelberg game, and the optimal reward R^* can maximize the utility function u_0 in equation (15) over $R \in [0,+\infty)$. To calculate R^*, many efficient methods could be leveraged, such as Newton's method [38].

5. Performance Evaluation

We have studied CrowdBox theoretically, and this section evaluates the performances of CrowdBox in realistic problem settings. In the following, we mainly elaborate on the simulation settings, compared algorithms, performance metrics, and results separately.

5.1. Simulation Settings. We assume that the E-IIoT system has been widely deployed in the industry, and a large amount of IIoT devices require the communication resources to communicate with the edge and cloud servers. To satisfy the demands, a set of NIB nodes could be employed at the edge layer to improve the communication capacity of the system with a given reward value R.

The default values of simulation settings are defined as the following. In the E-IIoT system, the costs of NIB nodes are distributed uniformly over $S = \{1, 2, 3, \cdots, 10\}$, and the number of NIB nodes is equal to $n = 1000$. We set the $f(\widehat{T}, N) = \lambda \log(1 + \sum_{c_j\in S}(n_j \log(1 + t_j)))$, where $n_j \in N$ and $t_j \in \widehat{T}$, since $j \log(1 + t_j)$ represents E-IIoT's diminishing return on the work of an NIB node with unit cost c_j, and $\log(1 + \sum_{c_j\in S}(n_j \log(1 + t_j)))$ denotes E-IIoT's diminishing return on the number of participating NIB nodes. Unless otherwise specified, we use the default values to conduct the following simulations.

5.2. Compared Algorithms (Strategies). In order to evaluate the performances of CrowdBox, we adopted two compared strategies (benchmarks)—random and best effort strategy. Both two strategies are general game theory-based methods adopted in many related fields, and the basic ideas are as follows.

5.2.1. Best Effort Strategy. NIB node i is always trying its best to provide the communication resource (i.e., $t_i = t_{i(\max)}$, where $t_{i(\max)}$ could be computed by setting $u_i = 0$ for NIB node i) only if it can obtain the profit (i.e., the NIB node utility is larger than zero) from the E-IIoT system.

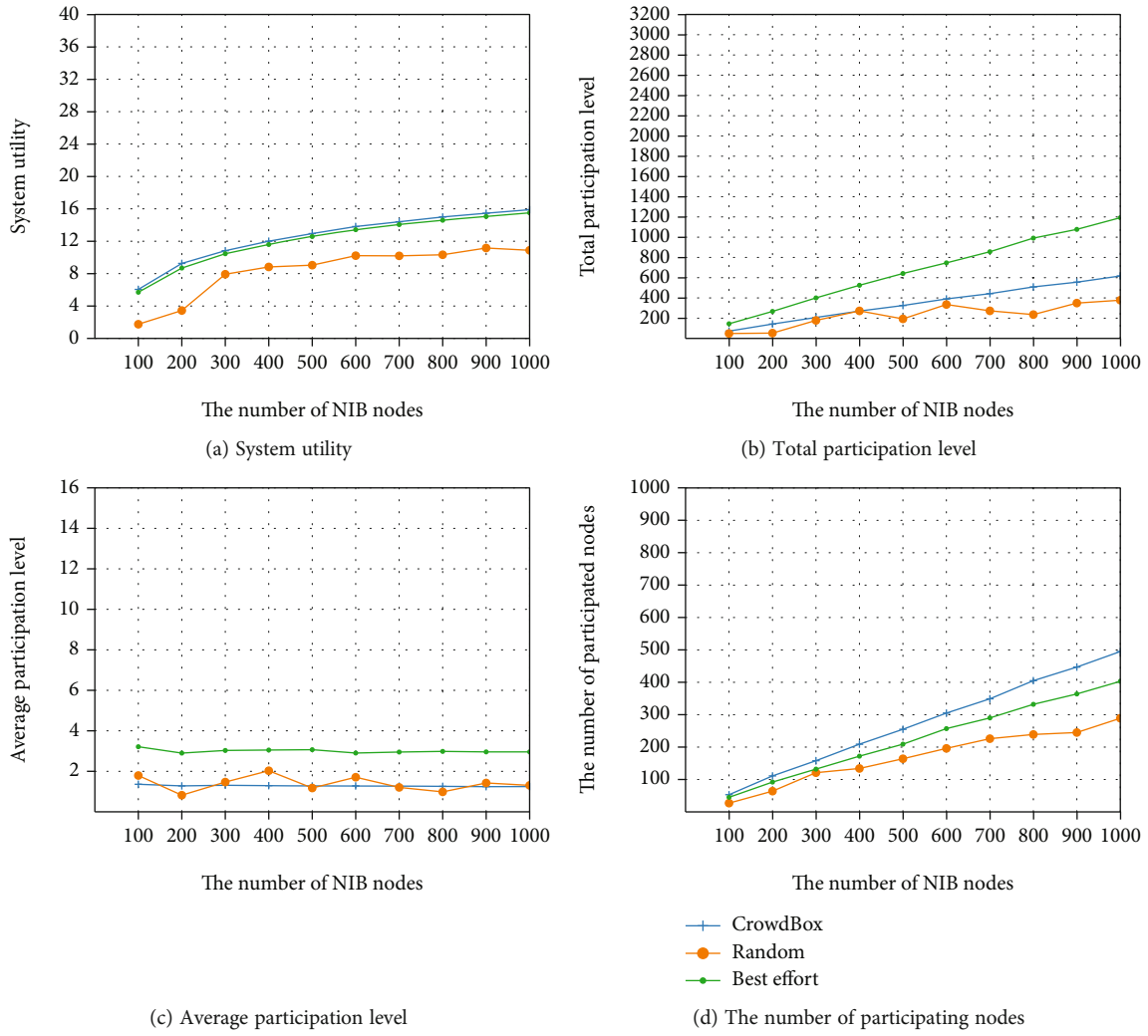
5.2.2. Random Strategy. The strategy of NIB node i is selected by itself based on its own preferences at random. For each NIB node i, we define a maximum cooperation time duration $t_{i(\max)}$, and NIB node i can choose the random value in $[0, t_{i(\max)}]$.

5.3. Performance Metrics and Results. Four major metrics are leveraged to measure the effectiveness of our proposed algorithm and benchmarks, including the system utility, total participation level, average participation level, and the number of participating NIB nodes during the whole Stackelberg game process.

The system utility u_o has been introduced in Section 3, and it is the main metric to measure the algorithm performance; the total participation level is defined as the sum of NIB node strategy t_i; the average participation level can be computed by the total participation level divided by the number of participating NIB nodes; the number of participating NIB nodes is the number of nodes whose $t_i > 0$ during the game process.

By changing the value of NIB nodes and the number of NIB nodes' maximum cost for the system, we show the performances of CrowdBox and the other two benchmarks in the following.

5.3.1. The Number of NIB Nodes. Here, we verify the effectiveness of CrowdBox with different numbers of NIB nodes. In particular, the number of NIB nodes is changed from 100 to 1000 with the increment of 100 nodes, and the results are depicted in Figure 2.



(a) System utility

(b) Total participation level

(c) Average participation level

(d) The number of participating nodes

Figure 2: Performances with different numbers of NIB nodes.

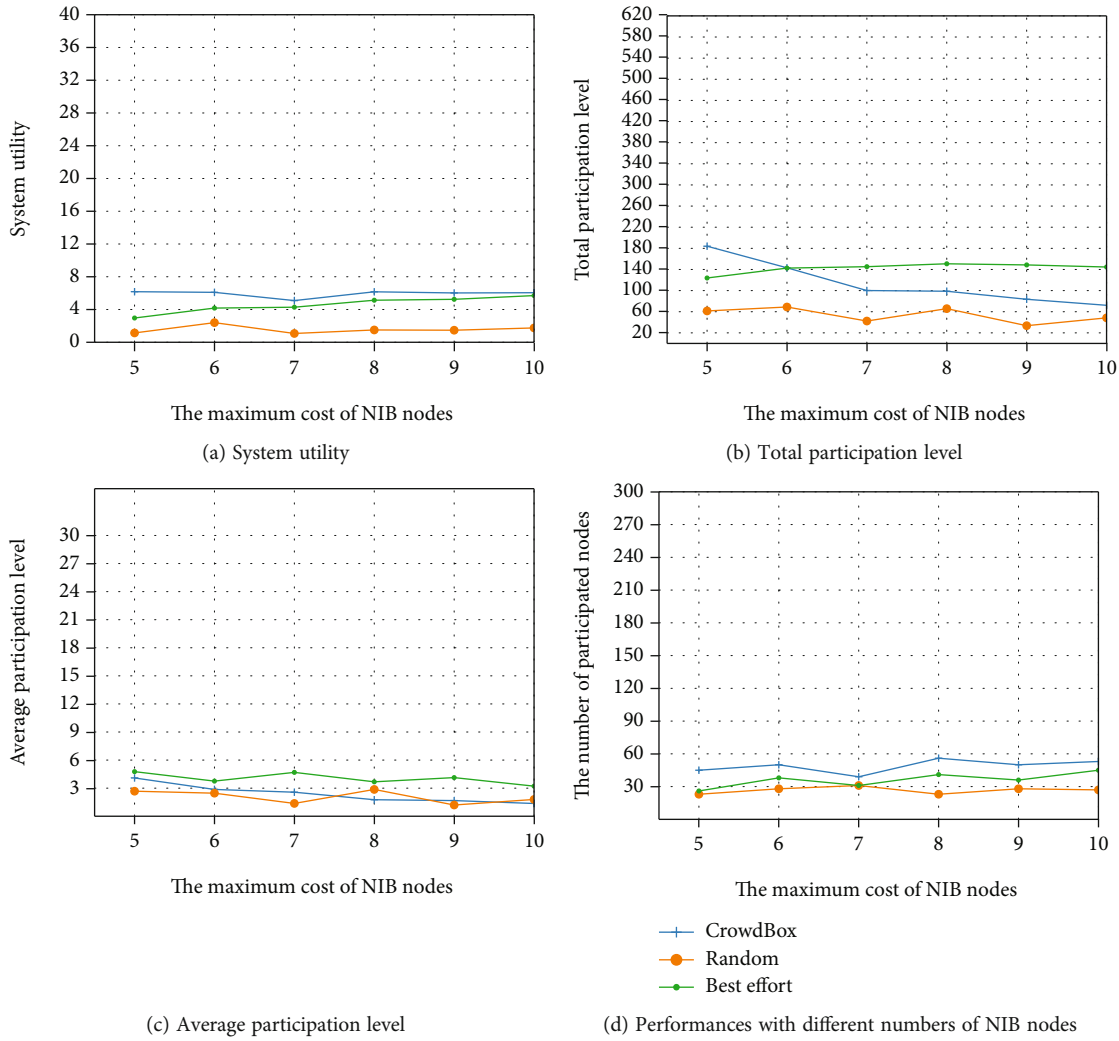
To be specific, Figure 2(a) shows that CrowdBox has the highest system utility contrasting with the other two algorithms, and this is our key goal to design CrowdBox which chooses the best response strategy to maximize the system utility. As it is expected, the total participation level of the best effort strategy is the highest as shown in Figure 2(b). The main reason here is that all NIB nodes try their best to do the task, but this is not a Nash equilibrium status in practice. Besides, CrowdBox is better than random strategy in the total participation level, and it is more obvious with the increase of the number of NIB nodes.

In terms of the average participation level depicted in Figure 2(c), both CrowdBox and best effort strategy reach a stable state. In contrast, the average participation level of random strategy varies as the number of NIB nodes increases, and this is mainly because the NIB nodes randomly choose how to cooperate (e.g., whether participate or not and how to participate). Similar to the result of system utility, the number of participating nodes for CrowdBox also achieves the highest with the number of NIB nodes increasing according to Figure 2(d). This is mainly because CrowdBox is a Nash equilibrium status and an NIB node will cooperate only if it can obtain profits from it. In addition, the random strategy is still the worst case among the three algorithms, and we notice it results from that it is not a Nash equilibrium status.

We can see that CrowdBox has the highest system utility and the number of participating nodes compared with the other two benchmarks. Although the participation level of the best effort strategy is higher than CrowdBox and the random strategy, it is difficult to reach since nodes are rational in practice. Thus, CrowdBox outperforms the other two strategies with the increase in the number of NIB nodes.

5.3.2. The Number of NIB Nodes' Maximum Cost. Then, we evaluate all three strategies with different numbers of NIB nodes' maximum cost. In particular, the maximum cost value varies from 5 to 10 with the increment of 1, and Figure 3 depicts the final results.

From the perspective of system utility (i.e., Figure 3(a)), CrowdBox still outperforms the other two strategies (i.e., random and best response strategy) since it is the Stackelberg Nash equilibrium status. Furthermore, the best effort strategy is better than the random strategy in system utility



(a) System utility

(b) Total participation level

(c) Average participation level

(d) Performances with different numbers of NIB nodes

Figure 3: Performances with different numbers of NIB nodes' maximum cost.

with different maximum costs, and this is mainly because those NIB nodes with best strategy effort are always doing their best to provide communication resources.

The participation levels (i.e., Figures 3(b) and 3(c)) of the three strategies show similar trends where the best strategy is the highest and the random strategy is the lowest in most cases. The number of participating NIB nodes depicted in Figure 3(d) concludes that CrowdBox also can stimulate more nodes to participate in providing extra communication resources under different maximum costs of NIB nodes, and this is also the main goal of this paper. Specifically, CrowdBox can reach a Stackelberg Nash equilibrium where all participating NIB nodes cannot obtain more extra profits if it derives from the current strategy.

Similar to the last simulation, we can get that although CrowdBox does not perform as well as the best effort strategy in the participation level (i.e., total and average), CrowdBox also has the highest system utility and number of participating nodes compared with the other two benchmarks with varying maximum costs of NIB nodes.

To summarize, CrowdBox can achieve the highest system utility and number of participating NIB nodes, the two key metrics of this paper, contrasting with the other two benchmarks under various settings based on the realistic simulation scenario. CrowdBox also performs better than random strategy in the other two metrics (i.e., total and average participation level), and the best effort strategy is not practiced since NIB nodes are usually rational (selfish).

6. Conclusion

The edge computing-enabled Industrial Internet of Things (E-IIoT) has aroused strong attentions recently since it creates a favorable communication environment to realize the concept of smart industry under the 6G scenario. In this paper, we propose CrowdBox, a crowdsourcing-based NIB node recruitment algorithm, leveraging the Stackelberg game theory, to improve the communication capacities with the power of crowds. CrowdBox shows the Nash equilibrium of the Stackelberg game and can choose the best response strategy for both E-IIoT and NIB nodes. Finally, extensive evaluations are conducted to verify the performance of CrowdBox under different realistic simulation scenarios,

and the result shows that CrowdBox outperforms the other two strategies.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the NSFC-Liaoning Province United Foundation under Grant U1908214, the National Natural Science Foundation of China under Grants 61872052 and 62172069, the China Computer Federation-Tencent Open Fund under Grant IAGR20210116, the Fundamental Research Funds for the Central Universities under Grants DUT21TD107 and DUT20RC(3)039, the LiaoNing Revitalization Talents Program under Grant XLYC2008017, the Liaoning Key Research and Development Program under Grant 2019JH210100030, the Xinghai Scholar Program in Dalian University of Technology, the Natural Science Foundation of Liaoning Province under Grant No. 2019-MS-055, and the Youth Science and Technology Star of Dalian under Grant 2018RQ45.

References

[1] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian, "Distributed Q-learning aided heterogeneous network association for energy-efficient IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2756–2764, 2020.

[2] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in Industrial Internet of Things and Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[3] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[4] L. Wan, Y. Sun, I. Lee, W. Zhao, and F. Xia, "Industrial pollution areas detection and location via satellite-based IIoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 1785–1794, 2021.

[5] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. P. C. Rodrigues, "SDN-enabled multi-attribute-based secure communication for smart grid in IIoT environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2629–2640, 2018.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] P. Wang, R. Yu, N. Gao, C. Lin, and Y. Liu, "Task-driven data offloading for fog-enabled urban IoT services," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7562–7574, 2021.

[8] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, "A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3664–3674, 2020.

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[10] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: towards minimizing delay in the Internet of Things," in *2017 IEEE International Conference on Edge Computing (EDGE)*, pp. 17–24, Honolulu, HI, USA, 2017.

[11] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the Internet of Things: security and privacy issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017.

[12] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in SDN-based industrial Internet of Things with edge computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[13] X. Hou, Z. Ren, K. Yang, C. Chen, H. Zhang, and Y. Xiao, "IIoT-MEC: a novel mobile edge computing framework for 5G-enabled IIoT," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Marrakesh, Morocco, 2019.

[14] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4702–4711, 2018.

[15] B. Yang, X. Cao, X. Li, Q. Zhang, and L. Qian, "Mobile-edge-computing-based hierarchical machine learning tasks distribution for IIoT," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2169–2180, 2020.

[16] S. Chen, Y. Liang, S. Sun, S. Kang, W. Cheng, and M. Peng, "Vision, requirements, and technology trend of 6G: how to tackle the challenges of system coverage, capacity, user data-rate and movement speed," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 218–228, 2020.

[17] X. Wei, S. Wang, A. Zhou et al., "MVR: an architecture for computation offloading in mobile edge computing," in *2017 IEEE International Conference on Edge Computing (EDGE)*, pp. 232–235, Honolulu, HI, USA, 2017.

[18] D. Balfanz, G. Durfee, R. E. Grinter, D. K. Smetters, and P. Stewart, "Network-in-a-box: how to set up a secure wireless network in under a minute," in *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA, 2004.

[19] E. Alshaer, W. Marrero, A. Elatawy, and K. Elbadawi, "Network configuration in a box: towards end-to-end verification of network reachability and security," in *2009 17th IEEE International Conference on Network Protocols*, pp. 123–132, Plainsboro, NJ, USA, 2009.

[20] M. Pozza, A. Rao, H. Flinck, and S. Tarkoma, "Network-in-a-box: a survey about on-demand flexible networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2407–2428, 2018.

[21] P. Wang, Z. Yu, C. Lin, L. Yang, Y. Hou, and Q. Zhang, "D2D-enabled reliable data collection for mobile crowd sensing," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 180–187, Hong Kong, 2020.

[22] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "Slade: a smart large-scale task decomposer in crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1588–1601, 2018.

[23] P. Wang, M. Varvello, C. Ni, R. Yu, and A. Kuzmanovic, "Web-lego: trading content strictness for faster webpages," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pp. 1–10, Vancouver, BC, Canada, 2021.

[24] V. Pilloni, "How data will transform industrial processes: crowdsensing, crowdsourcing and big data as pillars of Industry 4.0," *Future Internet*, vol. 10, no. 3, p. 24, 2018.

[25] X. Kong, F. Xia, J. Li, M. Hou, M. Li, and Y. Xiang, "A shared bus profiling scheme for smart cities based on heterogeneous mobile crowdsourced data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1436–1444, 2020.

[26] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.

[27] A. Karati, S. H. Islam, G. Biswas, M. Z. A. Bhuiyan, P. Vijayakumar, and M. Karuppiah, "Provably secure identity-based signcryption scheme for crowdsourced industrial Internet of Things environments," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2904–2914, 2017.

[28] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[29] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial Internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4692–4701, 2018.

[30] B. Chen, J. Wan, Y. Lan, M. Imran, D. Li, and N. Guizani, "Improving cognitive ability of edge intelligent IIoT through machine learning," *IEEE Network*, vol. 33, no. 5, pp. 61–67, 2019.

[31] C. Aduba and C. Won, "*N*-player statistical Nash game control: *m*-th cost cumulant optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2688–2694, 2016.

[32] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: a joint optimization approach combining Stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[33] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3602–3609, 2019.

[34] Y. Jie, C. Guo, K. R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based Industrial Internet of things environment," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3041–3052, 2020.

[35] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5444–5455, 2020.

[36] P. Loiseau, G. Schwartz, J. Musacchio, S. Amin, and S. S. Sastry, "Incentive mechanisms for Internet congestion management: fixedbudget rebate versus time-of-day pricing," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 647–661, 2014.

[37] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: crowdsourcing with smartphones," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1732–1744, 2015.

[38] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9488369, 14 pages
https://doi.org/10.1155/2021/9488369

Research Article

Short-Term IoT Data Forecast of Urban Public Bicycle Based on the DBSCAN-TCN Model for Social Governance

Dazhou Li,[1] **Chuan Lin**,[2] **Wei Gao**,[1] **Guangbao Yu**,[1] **Jian Gao**,[3] **and Wenbo Xia**[4]

[1]*College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110016, China*
[2]*Software College, Northeastern University, Shenyang 110819, China*
[3]*Shenyang University of Technology, Shenyang 110870, China*
[4]*Liaoning Technical University, Huludao 123032, China*

Correspondence should be addressed to Jian Gao; gaojian_sut@foxmail.com

Received 28 June 2021; Revised 9 August 2021; Accepted 27 October 2021; Published 30 November 2021

Academic Editor: José A. García-Naya

Internet of Things will play a vital role in the public transport systems to achieve the concepts of smart cities, urban brains, etc., by mining continuously generated data from sensors deployed in public transportation. In this sense, smart cities applied artificial intelligence techniques to offload data for social governance. Bicycle sharing is the last mile of urban transport. The number of the bike in the sharing stations, to be rented in future periods, is predicted to get the vehicles ready for deployment. It is an important tool for the implementation of smart cities using artificial intelligence technologies. We propose a DBSCAN-TCN model for predicting the number of rentals at shared bicycle stations. The proposed model first clusters all shared bicycle stations using the DBSCAN clustering algorithm. Based on the results of the clustering, the data on the number of shared bicycle rentals are fed into a TCN neural network. The TCN neural network structure is optimized. The effects of convolution kernel size and Dropout rate on the model performance are discussed. Finally, the proposed DBSCAN-TCN model is compared with the LSTM model, Kalman filtering model, and autoregressive moving average model. Through experimental validation, the proposed DBSCAN-TCN model outperforms the traditional three models in terms of two metrics, root mean squared logarithmic error, and error rate, in terms of prediction performance.

1. Introduction

Smart cities employ technology and data to increase efficiencies, economic development, sustainability, and life quality for citizens in urban areas. Inevitably, clean technologies promote smart city development including energy, transportation, and health [1–3]. In this sense, smart cities present themselves as a viable solution to aggregate public resources, human capital, social capital and information, and communication technologies, to promote sustainable development [4]. For instance, the information gathered by the advancements of the intelligent transportation systems are progressively intricate and are portrayed by heterogeneous devices, huge volume, mistakes in spatial and transient procedures, and continuous necessities of real-time processing [5, 6]. Various countries throughout the world have started their efforts in designing and implementing smart cities [7]. This has led to the concept of smart cities where Information Communication and Technology are merged with the existing traditional infrastructure of a city, which is playing a vital role in policy design, decision, implementation, and ultimate productive services [8, 9].

The Internet of Things (IoTs) and artificial intelligence (AI) are two cornerstone technologies enabling the smart city concept, which are fusing into an organic whole in recent years. Some particular joint points where IoTs meet AI are intelligent IoT devices, smart sensing boosted by AI, and IoT big data mining with AI [10, 11]. The new Internet of Things paradigm and architecture allows one to rethink the way smart city infrastructures are designed and managed [12]. In the densely populated IoT applications, the sensing range of the nodes might overlap frequently [13]. The world will be populated by billions of connected IoT devices that will be placed in our homes, cities, vehicles, and industries

[14]. The novel IoT devices collect cyber-physical data and provide information on the environment [15]. The IoTs enable a smart city to power and monitor multiple geographically distributed nodes to support a range of applications across various domains such as energy and resource management, intelligent transport systems, and E-health to name a few [16]. Systems get smarter with computing capabilities, especially in the form of IoT devices [17].

In a variety of smart cities, AI has been widely deployed, yielding numbers of revolutionary applications and services that are primarily driven by techniques for data offloading for urban IoT [18, 19]. The growing urbanization coupled with the high demands of daily commute for working professionals has increased the popularity of public transport systems [20, 21]. In urban public transportation systems, IoTs are deployed in major cities of both developed and developing countries [22]. The huge growth of IoT devices and different characteristics in the traffic patterns have brought attention to traffic methods to address various raised issues in IoT applications [23].

Density-Based Spatial Clustering with Noise Applications (DBSCAN) is an unsupervised ML clustering algorithm. Unsupervised means that it does not use prelabeled targets to cluster data points. Clustering is the attempt to group similar data points into manually determined groups or clusters. It is an alternative to popular clustering algorithms such as K-Means and hierarchical clustering. The core idea of density clustering is to use the number of points in the neighborhood of a data object as the density of data points, which are then connected by density and added to the clusters adjacent to them, completing the construction of clustered clusters. It allows clusters made up of irregularly shaped data objects to be found and noise points to be discovered.

From the perspective of graph theory, the node and node relationship of the graph is used to represent the information of the dataset, each data object corresponds to a node in the graph, and the connection between two points represents that the data objects are connected to each other. The data objects in the same connected branch are more similar, and the data objects in different connected branches are less similar. For nonnumerical objects, a concept similarity measure can be used based on the concept represented by the object. The more common or similar properties exist between concepts, the more similar the objects are to each other, at which point the cluster is a collection of objects with some common properties. The DBSCAN is more commonly used in text clustering and WEB data mining.

2. Related Work

At this stage, there has been a great deal of research in the field of bicycle sharing, with Bonilla-Alicea et al. [24]. O'Brienet al. proposed a bicycle sharing system to reduce environmental pollution [25]. Kadri et al. investigated the problem of bike-sharing scheduling in a static mode bike-sharing system [26]. Erdoğan et al. proposed an exact algorithm, calculated accurate algorithms, and tested the benchmark examples [27].

For the vehicle rebalancing problem, in addition to manual bike scheduling, operators and governments encouraged the use of two-way incentives [28]. Zhang et al. proposed a forecasting method that considers bike inventory forecasts and user arrivals in a unified way [29]. Faghih-Imani and Eluru proposed destination forecasting with a polynomial model [30]. Zhang et al. proposed two new regression-based inference models to predict potential travel destinations [31]. Froehlich et al. [32] and Kaltenbrunner et al. [33] proposed a spatio-temporal analysis of bike-sharing stations in Barcelona using a clustering approach.

Vogel et al. conducted a predictive analysis of bike-sharing usage using a time-series approach [34]. Yoon et al. proposed an improved ARMA-based predictive model for predicting the number of shared bikes within bike-sharing stations [35]. Noland et al. investigated how population size and employment density can affect bike-sharing usage [36]. Gebhart and Noland investigated the effect of weather factors that can affect bike-sharing use and the duration of bike-sharing rides [37]. Borgnat et al. viewed bike-sharing systems as dynamic networks and conducted a spatial analysis of the dynamic networks to derive the spatial distribution patterns of the networks [38]. Campbell et al. noted that temperature, precipitation, inclement weather, and air quality all affect bike-sharing through a survey of bike-sharing projects in Beijing [39]. Kaspi et al. propose a Bayesian estimation model and use it to estimate the number of unavailable bikes in a bike-sharing station [40]. The assumption that bicycle failure is a binary property is a limitation of the proposed model. Since some bicycles need to be maintained and most of the bicycles are still be rented, the proposed model is not valid under these conditions. The long-term transaction history of the needs of the bicycle is considered to solve the problem.

Traditional forecasting studies on bike-sharing rentals have generally only predicted the number of rentals at individual stations. The impact of other nearby station rentals on the rentals of the single station under study is not taken into account in the process of predicting the rentals of a single station. The disadvantage not only reduces the accuracy of the forecast but also makes it difficult to place shared bikes. It can lead to an excess of shared bikes remaining at some stations. Simultaneously, there is a lack of shared bikes at some of the stations. It is a very irrational management method. A lot of time, material, and human resources are wasted. It harms the development of shared bikes.

Based on the above reasons, according to the geographical location of the shared bicycle stations, we use the DBSCAN clustering algorithm to cluster the shared bicycle stations. The clustering results of the shared bicycle stations are fed into a temporal convolutional network (TCN) to achieve the demand forecasting for the rental volume of multiple shared bicycle stations. Finally, the DBSCAN-TCN model is evaluated using error rate (ER) and root mean squared logarithmic error (RMLSE) as evaluation metrics.

3. Analytical Model of DBSCAN-TCN

Both LSTM and RNN are very effective methods in the field of time series prediction. However, both of them have high requirements on the length of the input sequence during

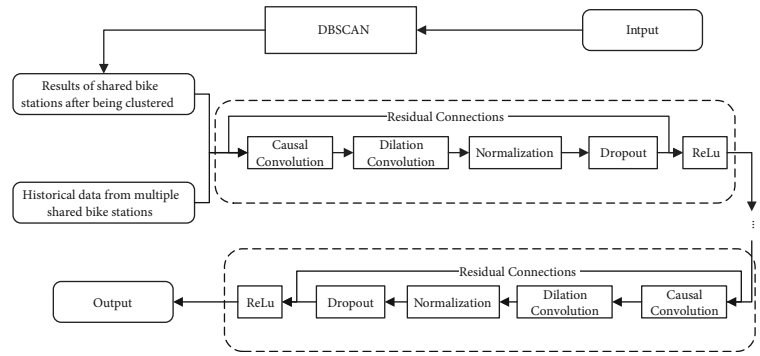


Figure 1: The architecture of the proposed DBSCAN-TCN model.

the computation. Also, the sequences they deal with are both one-dimensional. Long short-term memory (LSTM) and recurrent neural network (RNN) are not suitable when multiple time series are correlated and are considered as a whole. Figure 1 shows the architecture of the DBSCAN-TCN model proposed in the paper, which clusters that the bike-sharing stations in the geographic space are divided into different station clusters by the DBSCAN algorithm. Then, 48 hours of historical data from multiple bike-sharing stations belonging to the same station cluster are fed into the TCN. The DBSCAN-TCN model is shown in Figure 1.

3.1. DBSCAN Clustering of Public Bike-Sharing Stations. Compared to DBSCAN, K-Means is particularly susceptible to outliers. When the algorithm traverses the center of mass, outliers have a significant effect on how the center of mass moves before stability and convergence are achieved. In addition, K-Means suffers from the problem of clustering data exactly, when cluster sizes and densities vary. K-Means can only be applied to spherical clusters. If the data is not spherical, its accuracy suffers. Finally, K-Means requires that the number of clusters, which wishes to find be chosen first. On the other hand, DBSCAN does not require the number of clusters to be specified. It not only avoids outliers but determines that it works very well with clusters of any shape and size. DBSCAN has no center of mass. Clustering clusters are formed by joining neighboring points together.

In this paper, the DBSCAN clustering algorithm is chosen to cluster bike-sharing stations within a certain geographic space. The geographical information of the shared bicycle stations, i.e., latitude and longitude, is used as input to the DBSCAN clustering algorithm. According to the DBSCAN clustering rules, the clustering results of the shared bicycle stations within a certain geographic space are used as the output generated by the algorithm. To make better use of the DBSCAN algorithm, two important parameters of the DBSCAN clustering algorithm, *Eps* and *MmPtS*, need to be discussed. The parameters *Eps* and *MmPtS* are used to describe the closeness of the distribution of the neighborhood samples and the neighborhood distance threshold for a given sample, respectively.

Eps is the maximum radius of the community. If the distance of the data points from each other is less than or equal to the specified Eps, then they will be in the same class. It means that DBSCAN uses Eps to determine whether two points are similar and belong to the same class. Larger Eps will produce larger clusters, and smaller Eps will build smaller clusters. In general, the smaller value is chosen. Because only a very small percentage of the data points are within a distance of each other. However, if it is too small, the clusters will be split smaller. Step 2 of the DBSCAN algorithm was demonstrated to determine the best Eps.

Within the radius of a neighborhood, some MmPtS neighbors are considered to be a cluster. The initial points are contained in the MmPtS. A lower MmPtS helps the algorithm to build more clusters with more noise or outliers. A higher MmPtS will ensure more robust clusters. However, if the clusters are too large, the smaller clusters will be merged into the larger ones. In this paper, MmPtS is set to be greater than or equal to the dimensionality of the dataset. We multiply the number of dimensions of the features by two to determine their MmPtS values.

Assume that the dataset is $D = (x_1, x_2, \cdots, x_m)$. The density description of the DBSCAN clustering algorithm is defined as follows.

(1) *Eps Neighborhood*. For $x_j \in D$, the *Eps* neighborhood contains the subset of samples in the sample set D whose distance from is less than or equal to *Eps*, i.e., $N\epsilon(x_j) = \{x_i \in D \mid \text{distance}(x_i, x_j) \leq Eps\}$. The number of this subsample set is denoted as $|N\epsilon(x_j)|$

(2) *Core Objects*. For any sample $x_j \in D$, x_j is a core object, if its *Eps* neighborhood corresponds to $N\epsilon(x_j)$ containing at least *MinPts* samples, i.e., $|N\epsilon(x_j)| \geq MinPts$

(3) *Density Direct*. If x_i lies in the x_j neighborhood of *Eps* and is a core object, then it is said to be density direct from x_i to x_j. Note that the converse is not necessarily true. The density direct from x_j to x_i can not be confirmed unless and until x_i is also a core object

(4) *Density Reachable*. If $p_1, p_2, \cdots, p_T$ satisfies $p_1 = x_i$, $p_T = x_j$, and p_{t+1} directly reached p_t, it is said to be density reachable from x_i to x_j. Therefore, density reachability satisfies transmissibility. In this case, the transfer samples in the sequence $p_1, p_2, \cdots, p_T$ are

all core objects, as only the core objects can make the other samples' density reachable. It should be noted that density reachability also does not satisfy symmetry. It can be derived from the asymmetry of the density reach

(5) *Density Connected*. For x_i and x_j, if a sample of core objects x_k exists such that both x_i and x_j are accessible by density reachable from x_k, then x_i and x_j are said to be density connected. Density connectivity satisfies symmetry

The DBSCAN algorithm is calculated as shown below:

Input: sample set is $D=(x_1, x_2, \ldots, x_m)$, neighbourhood parameters (E, $MinPts$).

Output: Cluster division C.

Computational steps are as follows:

Step 1. The core set of objects $\Omega = \varnothing$, the cluster division $C = \varnothing$, the set of unvisited samples $\Gamma = D$, and the number of clusters $k = 0$ are initialized.

Step 2. For $j = 1, 2, \cdots, m$, all core objects are found according to the following steps. The subsample set $N\epsilon(x_j)$ of the E neighborhood of the sample is found by measuring the distance. If the number of samples in the subsample set satisfies $|N\epsilon(x_j)| \geq MinPts$, sample x_j is added to the sample set $\Omega = \Omega \cup \{x_j\}$ of the core objects.

Step 3. If the core set of objects is empty $\Omega = \varnothing$, the clustering algorithm ends. Otherwise, the algorithm proceeds to Step 4.

Step 4. A core object o is randomly selected in the core object set Ω. The current cluster core object queue $\Omega_{cur} = \{o\}$, the category ordinal number $k = k + 1$, and the current cluster sample set $C_k = \{o\}$ are initialized. The set of unvisited samples $\Gamma = \Gamma - \{o\}$ is updated.

Step 5. If the current cluster core object queue is empty and $\Omega_{cur} = \varnothing$, the current cluster C_k is generated. The cluster partition $C = \{C_1, C_2, \cdots, C_k\}$ and the set of core objects $\Omega = \Omega - C_k$ are updated. Otherwise, only the set of core objects $\Omega = \Omega - C_k$ is updated.

Step 6. A core object o' is removed from the current cluster core object queue Ω_{cur}. The set of all E neighborhood subsamples $N\epsilon(o')$ is found according to the neighborhood distance threshold E. Let $\Delta = N\epsilon(o') \cap \Gamma$. The current set of cluster samples $C_k = C_k \cup \Delta$, the set of unvisited samples $\Gamma = \Gamma - \Delta$, and $\Omega_{cur} = \Omega_{cur} \cup (\Delta \cap \Omega) - o'$ are updated. The algorithm moves to Step 5.

Output: The set of cluster divisions $C = \{C_1, C_2, ..., C_k\}$.

The data for Divvy Bike Share Chicago was processed according to the DBSCAN algorithm above [41]. The names of bike-share stations in the city of Chicago are typically long. To facilitate the recording of information about these stations, stations were described by the ID of each station. After DBSCAN clustering, seven cluster representatives of the clusters are given in Table 1. These seven clusters contain two bicycle stations, thirteen bicycle stations, four bicycle stations, twenty bicycle stations, eight bicycle stations, ten bicycle stations, and seven bicycle stations, respectively. The first cluster has only two cycle stations because of its remote location and low footfall. It is the cluster with the fewest bicycle stations of the site clusters. Cluster 4 has twenty bicycle stations, which is the cluster with the most bicycle stations. It is in the heart of the city of Chicago and has a high volume of foot traffic.

Table 1: The 7 typical station clusters and the contained stations after processing by the DBSCAN clustering.

Cluster number	The ID of the shared bike station
1	427, 413
2	47, 52, 35, 26, 81, 180, 197, 110, 211, 111, 43, 31, 177
3	260, 502, 290, 123
4	77, 288, 365, 60, 300, 138, 195, 47, 112, 225, 346, 93, 58, 127, 131, 61, 55, 107, 50, 394
5	13, 343, 451, 327, 308, 199, 296, 307
6	15, 19, 129, 254, 256, 312, 275, 19, 210, 26
7	130, 86, 113, 217, 259, 163, 308

Figure 2 gives the clustering diagram of some bicycle stations produced after clustering. In Figure 2, the horizontal coordinates represent the latitude of the bicycle stations, and the vertical coordinates represent the longitude of the bicycle stations. The different colors in Figure 2 represent clusters of different bicycle stations. Stations of the same color belong to one category. The middle part has more points of the same color. It indicates that the geospatial dataset used has a higher density of intermediate stations. The surrounding areas have fewer points of the same color. It indicates that the locations are more geographically isolated. There are fewer bicycle stations to cluster, so the nearby stations are clustered together to form a class. After getting the clustering information of the bike-sharing stations in the dataset based on the geographical location, the amount of bike-sharing rentals in the clustering information is used as input to TCN.

3.2. TCN-Based Prediction of Bike-Sharing Rentals. We predict the amount of bike-sharing rentals at each station. Historical data is used to predict future rental volumes. Classical algorithms, such as LSTM and RNN, are weak in capturing the past information of the data, resulting in inaccurate prediction results. A difficulty with traffic prediction models is that if the input sequence is too long, then the neural network loses previous traffic information. We use the dilation convolution method to solve this difficulty. By expanding the convolutional session, the perceptual field of view is increased to a larger size, for retaining more semantic information. In addition, we use causal convolution to solve the difficulty that the lengths of the input and output sequences are inconsistent.

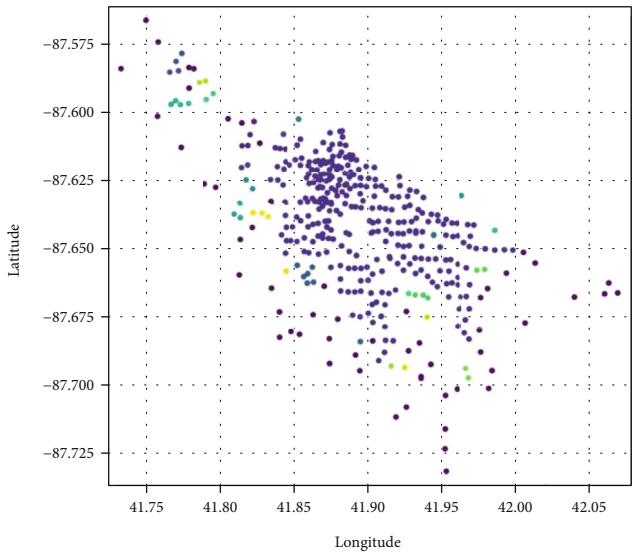


FIGURE 2: Clustering results of shared bike stations in the dataset according to the DBSCAN clustering algorithm.

In summary, the TCN used in the paper has 3 features. (1) The input sequence and the output sequence are kept consistent. In implementing full convolution, the input layer and the hidden layer are identical. In addition, 0 is used for padding to ensure that the subsequent layers are kept at the same length as the previous layers. (2) When predicting, the perceptual field of view is increased even more. To obtain a piece of longer history information, dilation convolution is introduced to construct a very deep neural network or a very large convolution kernel. (3) To prevent overfitting, the ReLu activation function and Dropout are employed. Therefore, the proposed structure consists of an input layer, causal convolution layer, dilation convolution layer, normalization layer, Dropout layer, ReLu layer, and output layer, as shown in Figure 3.

3.2.1. Input Layer of TCN. Based on the cluster information obtained by the DBSCAN algorithm clustering, the number of shared bicycle rentals belonging to all stations passing through a cluster is fed into the input layer of the TCN. The input layer is denoted **X**. Cluster 3 with 4 stations are used as an example. **X** is a two-dimensional matrix of size 4∗24. The number 4 represents the 4 stations. The number 24 represents a 24-hour day. The values in matrix **X** represent the number of rentals at each site in the dataset. The data in the input layer is shown in Equation (1).

$$X = \begin{bmatrix} x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} & \cdots & \cdots & x_{0,21} & x_{0,22} & x_{0,23} & x_{0,24} \\ x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & \cdots & \cdots & x_{1,21} & x_{1,22} & x_{1,23} & x_{1,24} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & \cdots & \cdots & x_{2,21} & x_{2,22} & x_{2,23} & x_{2,24} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & & & x_{3,21} & x_{3,22} & x_{3,23} & x_{3,24} \end{bmatrix}. \tag{1}$$

In Equation (1), $[x_{0,1}\, x_{0,2}\, x_{0,3}\, x_{0,4} \cdots \cdots x_{0,21}\, x_{0,22}\, x_{0,23}\, x_{0,24}]$ is a row of the matrix **X**. The column subscripts of the row elements represent the 24 hours of the day. The row elements represent the number of rentals at a particular shared bike station from 0 : 00 to 24 : 00. Using cluster 3 as an example, the first row in Equation (1) represents the number of rentals per hour from 0 : 00 to 24 : 00 for the bike-share station with ID 260.

In Equation (1), $\begin{bmatrix} x_{0,1} & x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}^T$ represents a column of the input matrix **X**. The 4 rows of this column correspond to the number of shared bicycle rentals from point 0 : 00 to 1 : 00 at each of the 4 stations. The first column in Equation (1) represents the number of rentals from 0 : 00 to 1 : 00 for the four stations in cluster 3, whose IDs are 260, 502, 290, and 123. The result of assigning matrix **X** to the Chicago Divvy bike-sharing dataset is shown in Equation (2).

$$X = \begin{bmatrix} 2 & 2 & 1 & 1 & \cdots & \cdots & 15 & 11 & 8 & 4 \\ 3 & 7 & 2 & 0 & \cdots & \cdots & 17 & 15 & 9 & 7 \\ 5 & 3 & 1 & 0 & \cdots & \cdots & 22 & 18 & 11 & 9 \\ 6 & 5 & 3 & 2 & \cdots & \cdots & 14 & 10 & 9 & 6 \end{bmatrix}. \tag{2}$$

3.2.2. Causal Convolution Layer of TCN. The first layer of TCN is a one-dimensional full convolution. The input matrix **X** is mapped through the full convolution network to the hidden layer. The dimensionality of the data does not change. Also to ensure the causality of the input and output, the TCN only performs convolutional operations on the input matrix **X** at the current moment as well as at previous moments.

Figure 4 shows the computational structure of the causal convolution. The input content of the causal convolution is shown in Equation (3). The computation process of the causal convolution is shown in Equations (4), (5), and (6).

$$X = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad \cdots \quad \cdots \quad x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]. \tag{3}$$

In Equation (3), x_1 is the 1st column in the input matrix, i.e.,$x_1 = \begin{bmatrix} x_{0,1} & x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}^T$.

$$j_1 \cdots j_t = F(x_1 \cdots x_t), \tag{4}$$

$$l_1 \cdots l_t = F(j_1 \cdots j_t), \tag{5}$$

$$y_1 \cdots y_t = F(l_1 \cdots l_t). \tag{6}$$

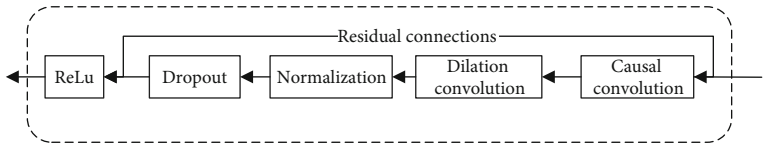


Figure 3: The basic structural unit of the proposed TCN.

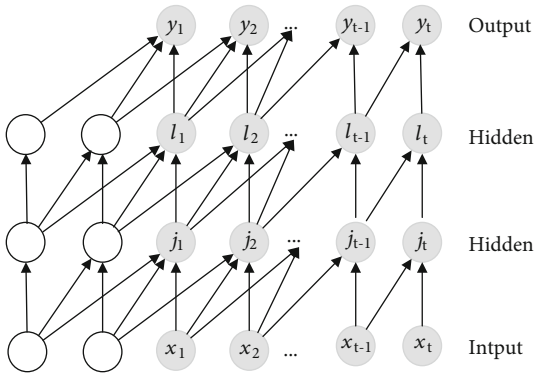


Figure 4: The structure of causal convolution layer of TCN.

As shown in Figure 4, j_t represents the result of the convolution when the input is x_t. l_t represents the result of the convolution when the input is j_t. y_t represents the result of the convolution when the input is l_t. They are all vectors. The dashed line represents the zero complement operation. The inputs and outputs are guaranteed to be the same size.

3.2.3. Dilation Convolution Layer of TCN. To keep the data input and output consistent, very deep network structures or very large convolutional kernels are required, when the amount of data is large. It places a significant burden on the computation of the model. Therefore, we introduce dilation convolution on top of causal convolution. The most important feature of the dilation convolution is that the perceptual field is expanded by controlling the dilation factor. In 1-dimensional dilation convolution, the dilation factor is equivalent to making the convolution kernel larger. The dilation factor is usually set to an exponential form of 2, e.g., 1, 2, 4, 8,…,2^i. When the dilation factor is equal to 1, the dilation convolution becomes a normal convolution operation. When the dilation factor becomes larger, the range of the field of perception also becomes larger. The output of the convolution will be linked to a long history of inputs.

When dealing with traffic data, the structure of the dilation convolution layer is shown in Figure 5. The first layer is the causal convolution layer. The output of the causal convolution layer is used as the input to the dilation convolution layer. As the number of layers of the dilation convolution increases, the length of the state and historical input data of the hidden layer increases exponentially. At the same time, the hidden layers will rapidly become smaller. The input data is represented as more high-dimensional.

Equation (7) is the formula for dilation convolution. Equation (8) represents the relationship between the number of layers of the network and the size of the convolution kernel. The convolution kernel size, the dilation factor, and the number of layers of the network are denoted as k, d, and i, respectively. An element of the input and the output of the causal convolution is denoted as s and y, respectively. N is positive integers.

$$z_1 \cdots z_t = \sum_{i=0}^{k-1} W(i) * y_{s-d\cdot i}, \tag{7}$$

$$d = 2^{i-1}(i = 1, 2, \cdots N). \tag{8}$$

As shown in Figure 5, $z_1 \cdots z_t$ is the output of the expanded convolution when the input is $y_1 \cdots y_t$, which are vectors.

3.2.4. Normalization Layer of TCN. In the TCN model, as the training continues to iterate and the network deepens, the distribution of the activation input values before the nonlinear transformation is done will gradually shift or update. The overall distribution will gradually move towards the upper and lower limits of the range of values of the activation function. If the model uses a Sigmoid activation function, after the model has been trained, the activation value will be updated to around 1. Because a feature is overweighted. Subsequently, no matter how much the value is increased and iterated over, the activation value will remain around 1. Too large a change will not occur. The above phenomenon can lead to an activation function that is insensitive to the features, affecting the effectiveness of the TCN.

In this paper, normalization is applied after each dilation of the causal convolution. The distribution of the input data is pulled back to the standard normal distribution. The inputs of each layer of the neural network are kept in the same distribution. It improves the training speed and convergence rate and avoids the gradient converging to 0. The normalization formula is shown in Equations (9), (10), (11), (12), and (13).

The input is denoted as $P = \{p_1, p_2, \cdots, p_n\}$. The linear activation value s_{p_i} for a given sample p_i is calculated as shown in Equation (9), and the weight matrix and bias parameters are set as W_p and b, respectively.

$$s_{p_i} = W_l p_i + b. \tag{9}$$

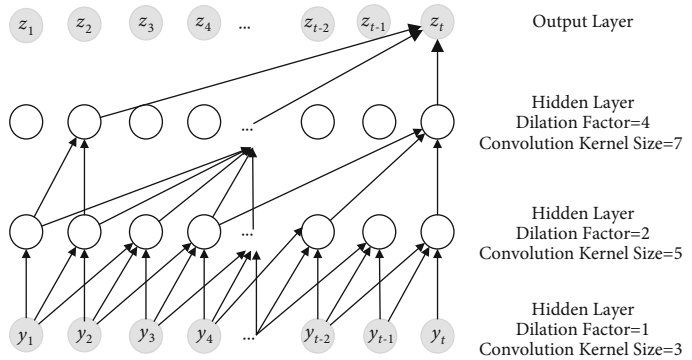


Figure 5: The structure of the dilation convolution layer of TCN.

According to Equations (10) and (11), the mean μ_p and variance ${\sigma_p}^2$ of the input data are calculated.

$$\mu_p = \frac{1}{u}\sum_{i=1}^{u} s_{p_i}, \tag{10}$$

$${\sigma_p}^2 = \frac{1}{u}\sum_{i=1}^{u}\left(s_{p_i} - \mu_p\right). \tag{11}$$

The activation value s_{pi} of each neuron within the hidden layer is transformed by a normal distribution. A new activation value $\hat{s}_{p_i}$ is obtained. In Equation (12), ε is a constant and is used to ensure the stability of the variance.

$$\hat{s}_{p_i} = \frac{s_{p_i} - \mu_p}{\sqrt{{\sigma_p}^2 + \varepsilon}}. \tag{12}$$

With Equations (9)–(12), the training speed of the model is boosted. The stability of the model is enhanced. The tuning process is optimized. However, this normal distribution transformation is equivalent to a linear function, which leads to a decrease in the expressiveness of the network. Therefore, to ensure that nonlinearity is obtained, two conditioning parameters γ and β are set in this paper, as shown in Equation (13). According to these two parameters, the activation values after the normal distribution transformation are then inverse transformed, for counteracting the side effects of the normal distribution transformation.

$$\hat{p}_i = \gamma\hat{s}_{p_i} + \beta. \tag{13}$$

3.2.5. Dropout Layer of TCN. Because the TCN uses causal convolution and dilation convolution, the number of network layers and intermediate layer results is increased. It not only results in an overall larger network structure and increased computational effort but also increases the memory used for training the network. To solve the above problem, the Dropout technique is used in this paper. As Equations (14) and (15) show, half of $z_1 \cdots z_t$ is assigned to 0. Dropout is randomly assigned to 0.

$$z_1 \cdots z_t = F(x_1 \cdots x_t), \tag{14}$$

$$z_2, z_4 \cdots z_{t-2}, z_t = 0. \tag{15}$$

3.2.6. ReLu Layer of TCN. ReLu is a commonly used activation function in neural networks. ReLu is adopted for the following reasons. (1) Traditional functions such as Sigmoid are used, which are complex to derive and computationally intensive when back-propagating the parameters to find the error gradient. When the ReLu activation function is used, the computational effort and computational time are saved. (2) In deeper network structures, gradient disappearance and gradient explosion can easily occur when back-propagating the parameters of the Sigmoid function, resulting in deeper network structures that are difficult to train. (3) The ReLu function being employed leads to some neurons having an output of 0. The network structure becomes sparse. Interdependencies between parameters are reduced. The problem of overfitting is alleviated. (4) The ReLu function is more easily optimized. Because the ReLu function is segmented linearly, it is easier to derive. Traditional Sigmoid functions tend to discard information in the propagation process.

After Dropout, as the ReLu function is used, any values less than 0 in $z_1, z_3 \cdots z_{t-3}, z_{t-1}$ become 0, and any values greater than 0 are the values themselves. Next, $z_1' \cdots z_t'$ is used as input to obtain $k_1 \cdots k_t$, and $k_1 \cdots k_t$ is treated in the same way as $z_1 \cdots z_t$ to obtain $m_1 \cdots m_t$. The above calculation process is shown in Equations (16), (17), (18), (19), and (20), where $x_1 \cdots x_t$ is the input, $z_1' \cdots z_t'$ and $k_1' \cdots k_t'$ are $z_1 \cdots z_t$ and $k_1 \cdots k_t$ after the ReLu activation function, and $m_1 \cdots m_t$ is the output.

$$z_1' \cdots z_t' = \mathrm{Relu}(z_1 \cdots z_t), \tag{16}$$

$$k_1 \cdots k_t = F\left(z_1' \cdots z_t'\right), \tag{17}$$

$$k_2, k_4 \cdots k_{t-2}, k_t = 0, \tag{18}$$

$$k_1' \cdots k_t' = \mathrm{Relu}(k_1 \cdots k_t), \tag{19}$$

$$m_1 \cdots m_t = F\left(k_1' \cdots k_t'\right). \tag{20}$$

3.2.7. Output Layer of TCN. After the ReLu activation function, the result of the ReLu function, m_t, is output by the output layer. The operation of the ReLu activation function layer and the final output layer is shown in Equation (21). The model finally obtains the prediction result s_t

$$m_1 \cdots m_t = s_1 \cdots s_t, \tag{21}$$

where $m_1 \cdots m_t$ is the result of the ReLu calculation and $s_1 \cdots s_t$ is the result of the output layer.

4. Numerical Evaluation and Discussion

4.1. Experimental Data and Preprocessing. The Chicago Divvy bike-sharing dataset is used in this paper. Divvy is part of the official bike-sharing system of the Chicago Department of Transportation (CDOT) and operated by Motivate, a leading bike-sharing manufacturer. There are currently more than 6,000 bikes available at over 580 stations. The bikes are available at fixed stations, i.e., docked bikes. The process of using the bicycle consists of obtaining a permit, unlocking at a station, riding, and returning at any station. The data for each year includes both station and trip data. Each station contains five fields, which are name, station name; latitude, station latitude; longitude, station longitude; dpcapacity, number of total docks at each station; and online date, date the station went live in the system. Each trip contains twelve fields, which are trip_id, ID attached to each trip taken; starttime, day and time trip started, in CST; stoptime, day and time trip ended, in CST; bikeid, ID attached to each bike; tripduration, time of trip in seconds; from_station_name, name of station where trip originated; to_station_name, name of station where trip terminated; from_station_id, ID of station where trip originated; to_station_id, ID of station where trip terminated; usertype, "Customer" is a rider who purchased a 24-Hour Pass, "Subscriber" is a rider who purchased an Annual Membership; gender, gender of rider; and birthyear, birth year of rider.

First, the geographic location information in this dataset was clustered, and then, bike-share rentals were predicted for different stations in the same clustered cluster. The data in the dataset between 1 January 2016 and 31 December 2016 were selected. The 24 clusters in the clustering result each contained sites ranging from 2 to 20. Clusters with 7-17 sites generally predominated. Cluster 7, which contains 7 sites, was selected as an example. Cluster 7 contains sites with IDs of 130, 86, 113, 217, 259, 163, and 308 as shown in Table 2. The number of bicycles rented from the seven stations with IDs 130, 86, 113, 217, 259, 163, and 308 during 10 : 00-11 : 00 on December 1, 2016, was 8, 1, 15, 1, 6, 17, and 8, respectively. The time interval was set to 1 hour, and the number of bicycles rented from cluster 7 at each period was calculated in this way.

Bicycle rental volume data, from January 1 to December 20, 2016, was selected for training the DBSCAN-TCN model. When the model was trained, information on the number of shared bicycle rentals at each station was fed into the DBSCAN-TCN model for prediction. The predicted results are compared with the true values. The error rate and root mean squared logarithmic error of the predicted and true values are calculated to evaluate the performance of the proposed prediction model.

Table 2: Number of bikes rented out in cluster 7 on 1 Dec. 2016.

Time	Station ID						
	130	86	113	217	259	163	308
6 : 00-7 : 00	1	4	0	0	1	0	1
7 : 00-8 : 00	5	9	0	3	1	1	2
8 : 00-9 : 00	3	13	3	3	1	2	3
9 : 00-10 : 00	12	28	4	3	2	2	1
10: 00-11: 00	8	1	15	1	6	17	8
11: 00-12: 00	13	6	5	1	11	7	19
12: 00-13: 00	6	8	10	13	16	7	2
13 : 00-14 : 00	9	11	13	7	2	4	3
14 : 00-15 : 00	10	7	12	13	16	6	8

Table 3: Parameter setting for the DBSCAN-TCN model.

Parameter	Setting
Activation functions	ReLu
Number of time steps	time_step = 24
Number of hidden cells per layer	24
Input layer dimension	input_size = 2
Output layer latitude	output_size = 1
Number of cycles	max_epoch = 100
Number of layers of the temporal convolutional network	4 layers
Convolutional kernel size	3
Dropout	0.5
Number of samples per batch	bitch_size = 24
Learning rate	learning_rate = 1

4.2. Implementation of DBSCAN-TCN Prediction Model with Pytorch. Pytorch has undergone rapid development in recent years and is used in a wide range of applications [42] Pytorch has many advantages. The Pytorch library is relatively easy to understand and works seamlessly with NumPy and SciPy. The tensor-based GPU acceleration is very powerful. During the calculation of network gradients, the network structure can be designed dynamically, without the need to build static graphs from scratch. Based on the simple and flexible design, the GPU can be used to accelerate the training of the network. Therefore, the proposed DBSCAN-TCN model is implemented using the Pytorch deep learning framework.

We use Pytorch to build the DBSCAN-TCN model and use the model to predict the amount of bike-sharing rentals. The implementation process involves setting up a set of

Table 4: Specific implementation steps of the DBSCAN-TCN model.

Step	Content
Step 1	The bike-sharing data is collated into a suitable input format. The data is fed into the model according to chronological order.
Step 2	The structure and parameters of the model are determined. For example, the dilation factor, the number of network layers, the size of the convolutional kernel, the number of layers in the hidden layers, and the number of neurons per layer.
Step 3	The learning rate, the appropriate optimization technique (Dropout), and the appropriate activation function (ReLu) are all selected.
Step 4	The training dataset is used to train the optimization model. The parameters of the prediction model are trained.
Step 5	The validation dataset is used to verify the model predictions. If the prediction is good, the model goes to Step 6; otherwise, it returns to Step 2.
Step 6	The model with the best prediction and the test dataset are used to predict the number of shared bicycle rentals.

Table 5: Performance of the proposed model on cluster 4 when choosing different Dropout rates.

Dropout	RMLSE	ER
0.4	0.268	0.253
0.5	0.241	0.231
0.6	0.312	0.301

TCN network structural units. The size of the convolutional kernel is set to 3. The number of layers of the causal and dilation convolutional networks is set to 4. The number of nodes in the hidden unit is set to 24. The input and output tensor dimensions of each TCN unit are set to 2 and 1, respectively. The activation function is set to ReLu. The number of cycles was set to 100. The number of time steps and the number of training samples per batch were both set to 24. As shown in Table 3, the learning rate was set to 1, and the Dropout was chosen to be 0.5.

Table 4 gives the six steps for training and using the DBSCAN-TCN model for predicting the amount of bike-sharing rentals.

4.3. Experimental Evaluation Criteria. The evaluation metrics used in this paper are RMLSE (root mean squared logarithmic error) and ER (error rate), which are shown in Equations (22) and (23), XCi,t are the true values of the number of bike-share rentals in a cluster Ci over time t, and $X^{\wedge}_{C_{i,t}}$ are the predicted values generated by the model. The predicted length of time is denoted as m, $m = 240$, and the time horizon of a day is denoted as T, $T = 24$. The number of clusters is denoted as i, e.g., C_1 denotes the first cluster.

$$\mathrm{RMLSE} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\log\left(X^{\wedge}_{C_{i,t}}+1\right)-\log\left(X_{C_{i,t}}+1\right)\right)^{2}}, \tag{22}$$

$$\mathrm{ER} = \frac{1}{T}\sum_{1}^{T}\frac{\sum_{i=1}^{m}\left|X^{\wedge}_{C_{i,t}}-X_{C_{i,t}}\right|}{\sum_{i=1}^{m}X_{C_{i,t}}}. \tag{23}$$

4.4. Effect of Hyperparameters on Experimental Results. The function of Dropout is to prevent all the feature extractors from acting together and causing some features to be scaled up or down all the time. Dropout not only prevents these problems from occurring during training but also increases the ability of the model to generalize. For the selection of Dropout rate, cluster 4 was chosen in this paper to validate as a representative. The reason is that cluster 4 contains the most number of bicycle stations, with 20 stations. During the training of the model, cluster 4 would be very computationally intensive. Dropout is therefore most evident for cluster 4. Table 5 shows that with a Dropout rate equal to 0.5, the trained model works best for cluster 4 and has the lowest error.

4.5. Experimental Results and Analysis. When people travel, they do not always use a shared bike to reach their destination directly. Usually, people change to other means of transport to reach their destination. For work and school, it is more common. Therefore, the use of shared bikes is inseparable from the period. During work and school hours, the number of shared bicycle rentals increases substantially. However, at other times of the day, bicycle-sharing rentals gradually return to normal levels. It is just the pattern of bicycle rental during normal working hours. When it is a day off, people are usually at home and resting. Most people are off work, although overtime can occur on weekends. Figure 6 shows the average number of bike-sharing rentals across the city at different times of the day in December 2016.

As can be seen in Figure 6, the curve changes are broadly the same throughout the weekdays in terms of bike-share rentals. Each inflection point is also roughly the same. On weekends, the number of shared bicycle rentals is much lower than on weekdays. It means that people are resting and not going out. The shared bikes are not being used. The curves for Saturday and Sunday are also broadly similar and have roughly the same inflection points.

Figure 6 shows that the number of shared bicycle rentals is lower in the early hours of a day. After 7 AM, the number of shared bicycle rentals gradually increases. By the end of the day, the magnitude of the change in rentals becomes even greater. After work hours, the number of bicycle-sharing rentals steadily drops back to normal levels. At the end of the work, the number of shared bicycle rentals rises again. By the end of the day, there will be very little use of shared bikes, because people are resting.

As can be seen in Figure 7, the difference between the predicted and true value curves for the station with ID 47

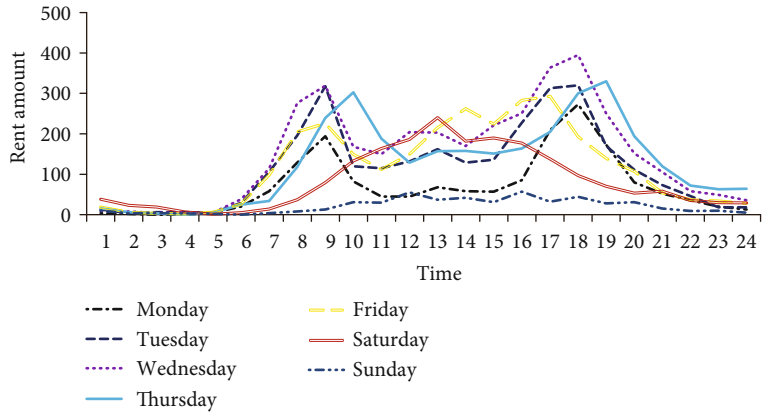


Figure 6: The rent amount of shared bikes varies over time, during different days of a week.

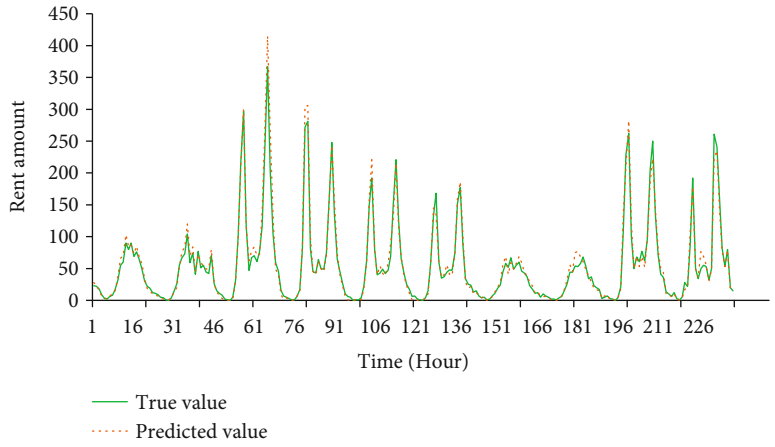


Figure 7: Predicted and true values of shared bicycle rentals at the site with ID 47 in cluster 2.

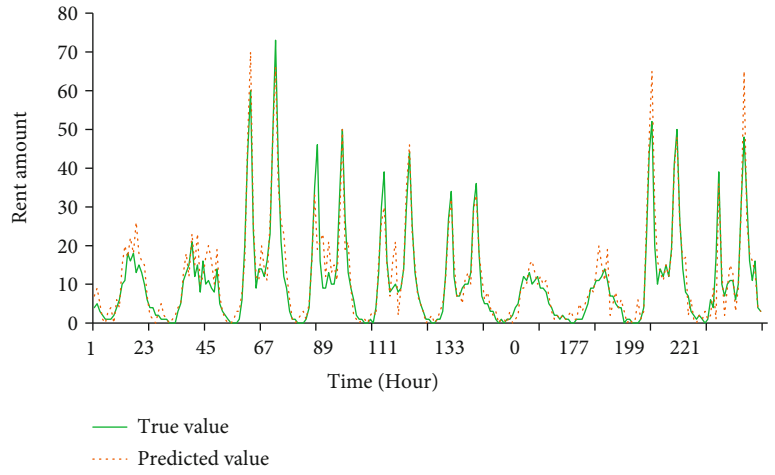


Figure 8: Predicted and true values of shared bicycle rentals at the site with ID 130 in cluster 7.

in cluster 2 is very small. The trends of the predicted and true values are also very similar. The range of variation between the predicted and true values is also very similar. It indicates that the proposed DBSCAN-TCN model has a good prediction effect. The horizontal and vertical coordinates in Figure 7 represent time and rental volume, respectively. The dashed line represents the predicted value, and the solid line represents the true value.

As shown in Figure 8, the prediction results for the site with ID 130 in cluster 7 are worse compared to the prediction for the site with ID 47 in cluster 2. In Figure 8, the difference between the predicted curve and the curve of the true

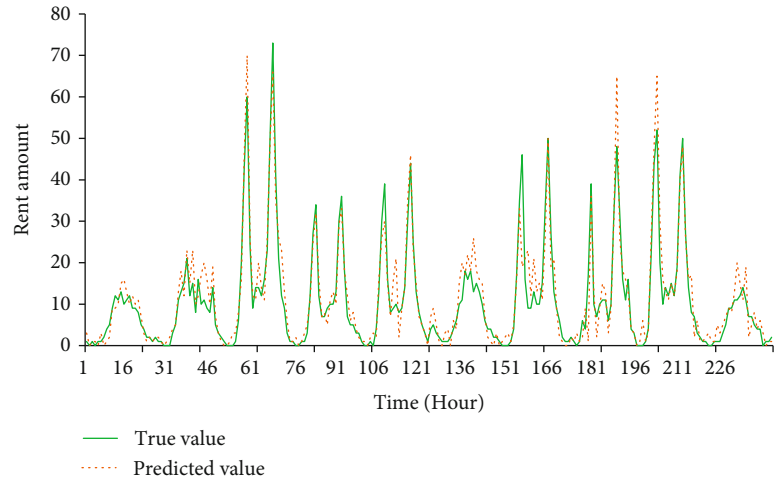


Figure 9: Predicted and true values of shared bicycle rentals at the site with ID 13 in cluster 5.

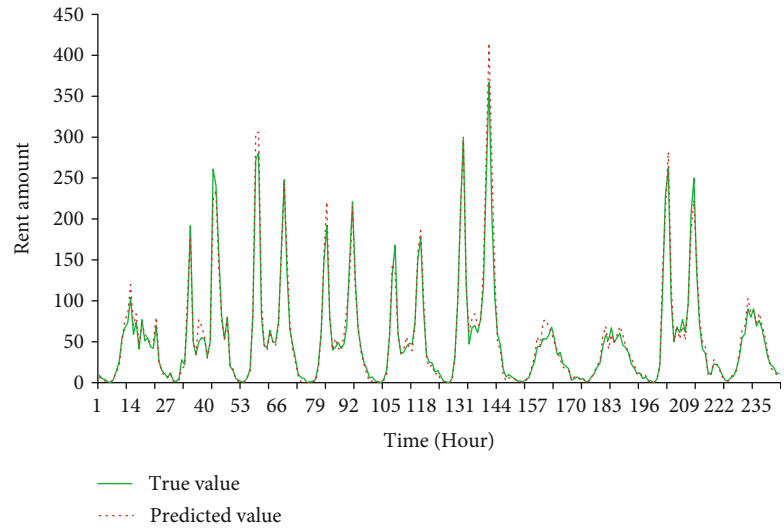


Figure 10: Predicted and true values of shared bicycle rentals at the site with ID 15 in cluster 6.

Table 6: Effect of different convolution kernel sizes on performance.

Convolutional kernel size	3		4		5	
	RMLSE	ER	RMLSE	ER	RMLSE	ER
Cluster 1	0.278	0.268	0.268	0.253	0.288	0.278
Cluster 2	0.242	0.255	0.232	0.215	0.252	0.265
Cluster 3	0.232	0.241	0.212	0.211	0.248	0.256
Cluster 4	0.249	0.238	0.241	0.231	0.254	0.265
Cluster 5	0.266	0.267	0.256	0.241	0.272	0.271
Cluster 6	0.247	0.268	0.234	0.256	0.251	0.273
Cluster 7	0.226	0.235	0.206	0.229	0.235	0.248

value is large. The reason for this occurrence is due to the small number of sites within cluster 7. It results in an inadequate capture of the information in the data.

Figures 9 and 10 show the predicted and true values for the number of shared bike stations rented for ID 13 in cluster 5 and ID 15 in cluster 6. The horizontal coordinates represent time, and the vertical coordinates represent the number of shared bicycle rentals at the stations in the cluster with the corresponding IDs.

The size of the convolution kernel has an impact on the performance of the model. The average performance evaluation of the model for each cluster is given in Table 6, for convolutional kernel sizes of 3, 4, and 5, respectively. It can be seen from Table 6 that as the convolutional kernel size increases, both the RMLSE and the ER increase. The average RMLSE and ER are both the largest for the clusters containing a small number of stations as they increase. It indicates that the effect of a small number of stations on the prediction of shared bicycle rentals is significant. The average RMLSE and ER of shared bicycle rentals for clusters with a convolution kernel size of 3 are the lowest. The small RMLSE and ER indicate that the model is predicting well.

In summary, taking the seven clusters in the dataset as an example, we use a Kalman filter model [43], an ARMA

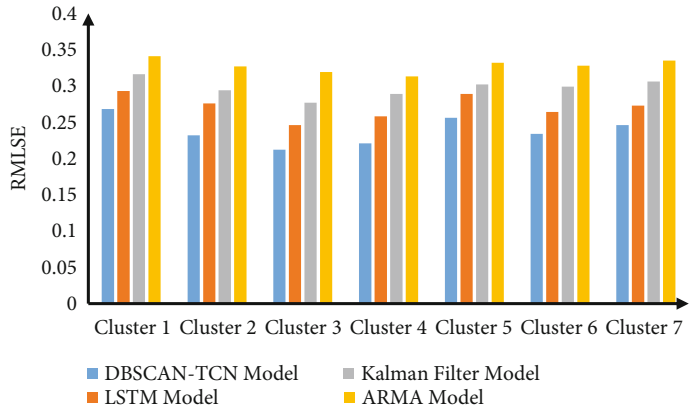


Figure 11: Comparison of the RMLSE of four forecasting models.

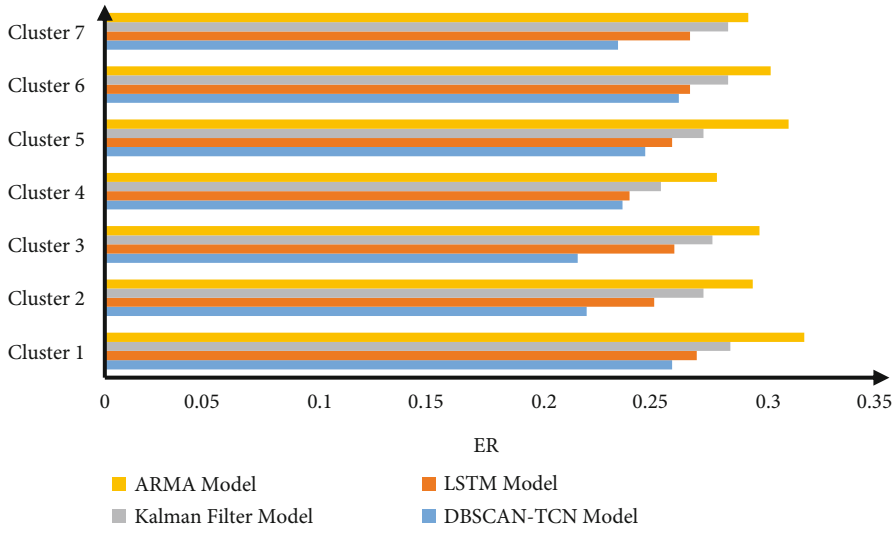


Figure 12: Comparison of the ER of four forecasting models.

model [44], and an LSTM model [45] to predict the average number of shared bicycle rentals for all stations within each cluster. Both ER and RMLSE were accounted for. The comparison results compared with the proposed DBSCAN-TCN model are shown in Figures 11 and 12.

Between Figures 11 and 12, it can be seen that the RMLSE and ER values of the DBSCAN-TCN model are lower than the Kalman filter model, the ARMA model, and the LSTM model. It indicates that the DBSCAN-TCN model is better than the Kalman filter model, the ARMA model, and the LSTM model in predicting the number of shared bicycle rentals, and the RMLSE and ER of the LSTM model are lower than the Kalman filter model and the ARMA model. The ARMA model has the worst prediction results. Therefore, it can be found that the proposed DBSCAN-TCN model in this paper has the best prediction effect.

5. Conclusions

In this paper, a DBSCAN-TCN model is proposed to forecast the amount of bike-sharing rentals. The proposed model uses the DBSCAN clustering algorithm to cluster the geographical data of bike-sharing stations. Based on the DBSCAN clustering results of the shared bicycle sites, the shared bicycle rentals of different sites belonging to the same cluster are transformed into the input data of the TCN. Using the trained DBSCAN-TCN model, the bike-sharing rentals are predicted for each station within each cluster. The effect of convolution kernel size on the model was also verified through the experiments. The experimental results show that the model predicts best when the convolution kernel size is 3. The performance of the model is evaluated by calculating and comparing the average RMLSE and ER of the sites in seven different clusters. The proposed DBSCAN-TCN model is compared with the Kalman filter model, the ARMA model, and the LSTM prediction model. The proposed model outperforms the classical three forecasting models in terms of both RMLSE and ER.

The main research of the paper is the prediction of the rental volume of the bike-sharing system. By analyzing the Chicago Divvy bike-sharing dataset, the DBSCAN-TCN model is proposed for the prediction of rental volumes. Certain research results were achieved. However, since a bicycle sharing system is a complex system, various factors need to

be considered. The study of bicycle-sharing rental volume still needs further improvement. In subsequent studies, other factors such as weather, temperature, and social holidays will be considered. A hybrid model based on the urban public bicycle prediction model is a good alternative. In the future, the model will have three main components to be considered in the research process. The first part will be a two-factor clustering of stations based on their geographical location and a historical transition model. The second part will predict the overall city bicycle hire by taking into account the time of day, weather, temperature, and wind speed factors. The third part seeks to find an inference model based on a coefficient of variation function. The model will be used to find the proportion of overall rental volume in each cluster and thus to predict the amount of bicycle rental in each cluster.

Data Availability

Previously reported Chicago Divvy bike-sharing dataset data were used to support this study and are available at https://www.divvybikes.com. These prior studies and datasets are cited at relevant places within the text as references [29].

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The project is supported by the Science and Technology Funds from Liaoning Education Department (No. LJ2020033 and No. LJKZ0449) and the National Social Science Foundation (No. 19CKS050).

References

[1] C. S. Lai, Y. Jia, Z. Dong et al., "A review of technical standards for smart cities," *Clean Technologies*, vol. 2, no. 3, pp. 290–310, 2020.

[2] P. Wang, C. Lin, M. S. Obaidat, Z. Yu, and Q. Zhang, "Contact tracing incentive for COVID-19 and other pandemic diseases from a crowdsourcing perspective," *IEEE Internet of Things Journal*, vol. 8, no. 21, 2021.

[3] C. Lin, G. Han, J. Du, T. Xu, and Z. Lv, "Spatiotemporal congestion-aware path planning toward intelligent transportation systems in software-defined smart city IoT," *IEEE Internet of Things Journal*, vol. 7, 2020.

[4] J. C. F. de Guimarães, E. A. Severo, L. A. Felix Júnior, W. P. L. B. da Costa, and F. T. Salmoria, "Governance and quality of life in smart cities: towards sustainable development goals," *Journal of Cleaner Production*, vol. 253, p. 119926, 2020.

[5] L. Zhao, H. Li, N. Lin, M. Lin, C. Fan, and J. Shi, "Intelligent content caching strategy in autonomous driving toward 6G," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2021.

[6] L. Zhao, W. Zhao, A. Hawbani et al., "Novel online sequential learning-based adaptive routing for edge software-defined vehicular networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, pp. 2991–3004, 2021.

[7] C. K. Toh, J. A. Sanguesa, J. C. Cano, and F. J. Martinez, "Advances in smart roads for future smart cities," *Proceedings of the Royal Society A*, vol. 476, no. 2233, 2020.

[8] M. A. Ahad, S. Paiva, G. Tripathi, and N. Feroz, "Enabling technologies and sustainable smart cities," *Sustainable Cities and Society*, vol. 61, 2020.

[9] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of artificial intelligence and machine learning in smart cities," *Computer Communications*, vol. 154, pp. 313–323, 2020.

[10] D. Zou and D. Gong, "Differential evolution based on migrating variables for the combined heat and power dynamic economic dispatch," *Energy*, vol. 238, 2022.

[11] R. Yu, D. Ye, Z. Wang et al., "CFFNN: cross feature fusion neural network for collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, 2021.

[12] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Sii-Mobility: an IoT/IoE architecture to enhance smart city mobility and transportation services," *Sensors*, vol. 19, no. 1, 2019.

[13] M. A. Jan, M. Zakarya, M. Khan et al., "An AI-enabled lightweight data fusion and load optimization approach for Internet of Things," *Future Generation Computer Systems*, vol. 122, pp. 40–51, 2021.

[14] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for AI-enabled IoT devices: a review," *Sensors*, vol. 20, no. 9, 2020.

[15] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi, "IoT traffic: modeling and measurement experiments," *IoT*, vol. 2, no. 1, pp. 140–162, 2021.

[16] S. U. Jamil, M. A. Khan, and S. U. Rehman, "Intelligent task off-loading and resource allocation for 6G smart city environment," *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 441–444, Sydney, NSW, Australia, 2020.

[17] B. Charyyev and M. H. Gunes, "IoT traffic flow identification using locality sensitive hashes," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6, Dublin, Ireland, 2020.

[18] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and fog computing enabled AI for IoT-an overview," *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019, pp. 51–56, Hsinchu, Taiwan, 2019.

[19] D. Zou, S. Li, X. Kong, H. Ouyang, and Z. Li, "Solving the dynamic economic dispatch by a memory-based global differential evolution and a repair technique of constraint handling," *Energy*, vol. 147, pp. 59–80, 2018.

[20] A. Ladha, P. Bhattacharya, N. Chaubey, and U. Bodkhe, "*IIGPTS*: IoT-based framework for intelligent green public transportation system," *Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)*, , pp. 183–195, Springer, 2020.

[21] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, "A distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, 2020.

[22] V. Puri, S. S. Jagdev, J. G. Tromp, and C. Van Le, "Smart Bicycle: IoT-Based Transportation Service," in *Intelligent Computing in Engineering*, pp. 1037–1043, Springer, 2020.

[23] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in IoT networks: a survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020.

[24] R. J. Bonilla-Alicea, B. C. Watson, Z. Shen, L. Tamayo, and C. Telenko, "Life cycle assessment to quantify the impact of technology improvements in bike-sharing systems," *Journal of Industrial Ecology*, vol. 24, no. 1, pp. 138–148, 2020.

[25] O. O'Brien, J. Cheshire, and M. Batty, "Mining bicycle sharing data for generating insights into sustainable transport systems," *Journal of Transport Geography*, vol. 34, pp. 262–273, 2014.

[26] A. A. Kadri, I. Kacem, and K. Labadi, "A branch-anDBSCAN-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems," *Computers & Industrial Engineering*, vol. 95, pp. 41–52, 2016.

[27] G. Erdoğan, M. Battarra, and R. Wolfler Calvo, "An exact algorithm for the static rebalancing problem arising in bicycle sharing systems," *European Journal of Operational Research*, vol. 245, no. 3, pp. 667–679, 2015.

[28] L. Li and M. Shan, "Bidirectional incentive model for bicycle redistribution of a bicycle sharing system during rush hour," *Sustainability*, vol. 8, no. 12, p. 1299, 2016.

[29] D. Zhang, C. Yu, J. Desai, H. Y. Lau, and S. Srivathsan, "A time-space network flow approach to dynamic repositioning in bicycle sharing systems," *Transportation Research Part B:Methodological*, vol. 103, pp. 188–207, 2017.

[30] A. Faghih-Imani and N. Eluru, "Analysing bicycle-sharing system user destination choice preferences: Chicago's Divvy system," *Journal of Transport Geography*, vol. 44, pp. 53–64, 2015.

[31] J. Zhang, X. Pan, M. Li, and P. S. Yu, "Bicycle-sharing system analysis and trip prediction," in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, pp. 174–179, Porto, Portugal, 2016.

[32] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *IJCAI 2009, Proceedings of the, international joint conference on artificial intelligence*, pp. 1420–1426, Pasadena, California, USA, 2009.

[33] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, "Urban cycles and mobility patterns: exploring and predicting trends in a bicycle-based public transport system," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 455–466, 2010.

[34] P. Vogel, T. Greiser, and D. C. Mattfeld, "Understanding bike-sharing systems using data mining: exploring activity patterns," *Procedia - Social and Behavioral Sciences*, vol. 20, no. 6, pp. 514–523, 2011.

[35] J. W. Yoon, F. Pinelli, and F. Calabrese, "Cityride: a predictive bike sharing journey advisor," in *2012 IEEE 13th International Conference on Mobile Data Management*, pp. 306–311, Bengaluru, India, 2012.

[36] R. B. Noland, M. J. Smart, and Z. Guo, "Bikeshare trip generation in New York City," *Transportation Research Part A*, vol. 94, pp. 164–181, 2016.

[37] K. Gebhart and R. B. Noland, "The impact of weather conditions on bikeshare trips in Washington, DC," *Transportation*, vol. 41, no. 6, pp. 1205–1225, 2014.

[38] P. Borgnat, C. Robardet, P. Abry, P. Flandrin, J. B. Rouquier, and N. Tremblay, "A dynamical network view of Lyon's Vélo'v shared bicycle system," *Dynamics On and Of Complex Networks*, vol. 2, pp. 267–284, 2013.

[39] A. A. Campbell, C. R. Cherry, M. S. Ryerson, and X. Yang, "Factors influencing the choice of shared bicycles and shared electric bikes in Beijing," *Transportation Research Part C*, vol. 67, no. 6, pp. 399–414, 2016.

[40] M. Kaspi, T. Raviv, and M. Tzur, "Detection of unusable bicycles in bike-sharing systems," *Omega*, vol. 65, no. 12, pp. 10–16, 2016.

[41] G. Zhang, H. Yang, S. Li, Y. Wen, and F. Liu, "What is the best catchment area of bike share station? A study based on Divvy system in Chicago, USA," *2019 5th International Conference on Transportation Information and Safety (ICTIS)*, 2019, https://ieeexplore.ieee.org/document/8883774.

[42] N. Ketkar, "Introduction to Pytorch," in *Deep Learning with Python*, pp. 195–208, Springer, 2017.

[43] S. Mandal, P. Chandramohan, P. H. Yen, C. D. Jan, Y. P. Lee, and H. F. Lee, "Discussion and closure: application of Kalman filter to short-term tide level prediction," *Journal of Waterway, Port, Coastal, and Ocean Engineering*, vol. 124, no. 4, pp. 213-214, 1998.

[44] I. Rojas, O. Valenzuela, F. Rojas et al., "Soft-computing techniques and ARMA model for time series prediction," *Neurocomputing*, vol. 71, no. 4-6, pp. 519–537, 2008.

[45] B. S. Pramod and P. M. Mallikarjuna Shastry, "Stock price prediction using LSTM," *Test Engineering and Management*, vol. 83, pp. 5246–5251, 2021.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6457099, 13 pages
https://doi.org/10.1155/2021/6457099

Research Article

Deep Reinforcement Learning for Collaborative Computation Offloading on Internet of Vehicles

Yureng Li,[1] **Shouzhi Xu**,[1] **and Dawei Li**[2]

[1]*College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China*
[2]*Department of Computer Science, Montclair State University, Montclair, NJ, USA*

Correspondence should be addressed to Shouzhi Xu; xsz@ctgu.edu.cn

Received 16 August 2021; Accepted 25 October 2021; Published 30 November 2021

Academic Editor: Pengfei Wang

With the increase of Internet of vehicles (IoVs) traffic, the contradiction between a large number of computing tasks and limited computing resources has become increasingly prominent. Although many existing studies have been proposed to solve this problem, their main consideration is to achieve different optimization goals in the case of edge offloading in static scenarios. Since realistic scenarios are complicated and generally time-varying, these studies in static scenes are imperfect. In this paper, we consider a collaborative computation offloading in a time-varying edge-cloud network, and we formulate an optimization problem with considering both delay constraints and resource constraints, aiming to minimize the long-term system cost. Since the set of feasible solutions to the problem is nonconvex, and the complexity of the problem is very large, we propose a Q-learning-based approach to solve the optimization problem. In addition, due to the dimensional catastrophes, we further propose a DQN-based approach to solve the optimization problem. Finally, by comparing our two proposed algorithms with typical algorithms, the simulation results show that our proposed approaches achieve better performance. Under the same conditions, by comparing our two proposed algorithms with typical algorithms, the simulation results show that our proposed Q-learning-based method reduces the system cost by about 49% and 42% compared to typical algorithms. And in the same case, compared with the classical two schemes, our proposed DQN-based scheme reduces the system cost by 62% and 57%.

1. Introduction

With the rapid development of the Internet of Vehicles (IoVs), vehicles have generated an increasing number of intensive computation tasks, such as online interactive applications, route planning, and traffic flow prediction. However, since these applications require a certain amount of computing resources and have Quality of Service (QoS) requirements, basic equipment in IoVs cannot meet the needs of vehicles suffering from hardware and other limitations [1].

To relieve the pressure of tight resources in IoV, Mobile Cloud Computing (MCC) is often used as a promising solution in such cases. In MCC networks, cloud servers (CS) have large volume of computation and communication resources to provide offloading services to multiple users at the same time [2]. However, since CS is usually deployed far away from vehicles, MCC can lead to significant transmission delays and system costs [3].

Mobile edge computing (MEC) is believed to serve as a reliable paradigm due to its ability to improve the QoS of vehicles in a way that reduces latency and energy costs [4]. To address latency-sensitive tasks while satisfying their demand for resources, MEC sinks computing services to the network edge, while network resources are also fully utilized [5, 6]. In IoVs, MEC servers are deployed on roadside units (RSUs), and vehicles within their coverage area are able to receive offloading services. The computing resources in MEC are still limited, which may result in some of the computation tasks failing when the amount of tasks in the network is too large [7]. Most of the previous studies have considered the optimal allocation of computing resources and model selection separately. Moreover, in terms of

computation offloading, some studies offload computing tasks from vehicles to edge servers or CS without considering the optimization of resources or the combination of both. Therefore, it is necessary to put forward a combined solution when solving practical problems.

Moreover, with the development of machine learning, reinforcement learning (RL) is considered as an effective method for finding optimal computation offloading strategies in time-varying scenarios. Compared with other optimization methods, agents in RL can find an optimal policy by observing the current environment to select actions as well as obtaining future rewards by constantly interacting with the environment [8]. Therefore, it is of great interest to design an efficient RL-based computation offloading and resource allocation scheme.

In this paper, we propose an efficient offloading scheme in the edge-cloud network by jointly optimizing offloading decision and resource allocation and ultimately achieving the optimization goal of minimizing system costs. In the concerned scenario, CS makes offloading decisions for each vehicle based on the current system state. Distinct from existing works, we consider both cooperation between CS and RSUs, as well as cooperation between RSUs. We formulate the optimization problem as a mixed-integer nonlinear programming (MINLP) problem, and to solve the optimization problem, we first propose a Q-learning-based approach. However, since Q-learning method may lead to dimensional catastrophe when the state and action space become too large [9], we further propose a DQN-based approach to compensate for this drawback. The main contributions of this article are as follows:

(i) We construct an edge-cloud network for time-varying IoVs, in which CS together with RSUs can process computing tasks for vehicles cooperatively

(ii) We study the cooperative offloading problem in proposed model and formulate the optimization problem as a MINLP problem, aiming to minimize the system cost

(iii) After defining the state space, action space, and reward function, we approximate the optimisation process as a Markov decision process (MDP). Based on the MDP, we propose Q-learning and DQN-based methods to solve the optimisation problem, respectively

(iv) Numerical results demonstrate that our proposed schemes significantly reduce system cost compared with other typical algorithms

The remainder of this paper is organized as follows. Section 2 introduces the related work of this article. Section 3 describes the system model in detail, including the computational model and the communication model. In Section 4, we describe the optimization problem and formulate the optimization problem as an MINLP problem. In Section 5, a Q-based method and a DQN-based method are proposed for the optimization problem, respectively. Simulation results and analysis are given in Section 6. Finally, Section 7 gives a summary of this paper.

2. Related Work

In recent years, research on computational offloading in MEC and MCC scenarios has become increasingly popular due to the need for practical scenarios. Specifically, in [10], Mao et al. perform a joint optimization for power and computational offloading in a MEC scenario using NOMA in order to achieve the optimization goal of minimizing system energy consumption. In [11], Ning et al. consider a MEC heuristic offloading scheme based on partial offloading with the optimization goal of reducing the system delay. In [12], Kuang et al. use a hierarchical approach to obtain suboptimal solutions to optimize the offloading pattern and power allocation in the MEC scenario. In [13], authors offload tasks to nearby vehicles as well as edge devices, and in this way solve the problem of computational offloading and probabilistic caching. In [14], Bi et al. consider the problem of service delivery and deployment in a single-user MEC system with the optimization goal of minimizing the overall system latency.

Considering the different characteristics of MEC and MCC, some studies have also considered the option of combining both. In [15, 16], the authors studied the system architecture of an edge-cloud system. In [17], Lin et al. proposed a directional charging scheme and improved energy transfer model in the MEC system. In [18], Wang et al. consider the server allocation problem for edge computing system deployment where each edge cloud is modeled as an M/M/c queue. In [19, 20], authors study the computation, communication, and the storage resources problems in both IoV and MEC networks. Wang et al. in [21] consider using D2D technology in MEC system to collect larger and better quality sensing data.

For the problems in MEC and MCC scenarios, some extant studies have used RL or DRL as solutions. In [22], Wang et al. transform the edge caching problem as a Markov decision process and propose a distributed cache replacement strategy based on Q-learning to address the optimization problem. In [23], Su et al. propose a Q-learning-based spectrum access scheme to optimize spectrum and maximize the transmission rate. In [24], Dinh et al. propose a Q-learning-based scheme to solve the optimization problem in a multiuser multiedge-node computation offloading scenario. He et al. use a dueling DQN approach in [25] to solve joint optimization problem in connected vehicle networks, considering not only network gains but also caching and computation gain in the proposed framework. In [26], Wang et al. investigate the best strategy for resource allocation in ICWNs by maximizing spectrum efficiency and system capacity across the network and propose a DQN-based task offloading scheme for MEC networks in urban cities. In [27], Zhou et al. use a DDQN-based approach to solve the energy minimization problem and simultaneously efficiently approximate the value function.

Unlike these existing studies, the content of this paper mainly considers the problem of MEC and MCC

collaboration in an IoV environment. Through the collaboration of edge-side and cloud-side servers, the paper is aimed at minimizing the energy consumption of the system. In order to solve the optimization problem of offloading decision and resource allocation, we propose two algorithms based on Q-learning method and DQN method, respectively.

3. System Model

In this section, we first present an edge-cloud network including mobile vehicles, RSUs equipped with MEC servers, and cloud servers (CS). Next, we give precise definitions of the model components.

3.1. Model Architecture. The edge-cloud network we consider is shown in Figure 1. The set of vehicles is denoted by $N = \{1, 2, \cdots, n\}$, and the set of MEC servers deployed at roadside units (RSUs) is denoted by $M = \{1, 2, \cdots, m\}$. In particular, we set that in this system time is divided into time slots of $t \in \{0, 1, 2, \cdots, T\}$, where T is the finite time horizon. And the computing task on a vehicle i $(i \in N)$ in time slot t is defined as $\Lambda_i(t) = \{S_i(t), C_i(t), D_i^{\max}(t)\}$, where $C_i(t)$ represents the total number of CPU cycles required to process the task, $S_i(t)$ represents the size of the computing task, and $D_i^{\max}(t)$ denotes the maximum delay tolerant of the task. Typically, MEC servers are deployed at the edge of the network, consisting of cellular networks to provide services to vehicles. The CS, on the other hand, is deployed away from vehicles and provides computing services through the core network. In order to guarantee the reliability of data transmission and provide offloading service for vehicles, RSUs and CS are connected via core networks. In general, the amount of computing resources and bandwidth is much higher on CS than on MEC servers, but the offload service is more costly on CS. The descriptions of the main symbols in this paper can be found in Table 1.

In the relevant scenario, there are multiple vehicles driving on the road within the coverage of CS and RSUs. Here, we denote $L_{i,j}(t)$ as the link between vehicle i and RSU j, where $L_{i,j}(t) = 1$ means that in time slot t vehicle i is in the coverage area of RSU j and vehicle i is associated with RSU j. In time slot t, vehicle i needs task offloading service. After obtaining its motion information and task information, the vehicle's offloading request is sent to its associated RSU. If its associated RSU does not have enough resource and cannot fulfill its requirement, its task information will be further sent to other RSUs in this area for cooperative offloading. If all the RSUs in this area fail to meet the task's demand for resources, the computing task will be offloaded to CS. We define the offloading decision of vehicle i as the integer variable $x_{i,j}(t) \in \{0, 1\}$, where $x_{i,j}(t) = 1$ means that the computing task of vehicle i is offloaded to RSUs in time slot t, and $x_{i,j}(t) = 0$ means that the computing task is offloaded to CS. Based on the current resource status of the RSUs, the offloading decision of the vehicle and the resource allocation are made dynamically by the control center in the CS. The task offloading process described above is shown in Figure 2.

3.2. Communication Model. Each vehicle can only be connected to one RSU at a time within the RSU's coverage range. We assign the bandwidth of $b_{i,j}$ to the link between vehicle i and RSU j. As shown in eq. (1), we calculate the data transmission rate according to Shannon's formula

$$R_{i,j}(t) = b_{i,j}(t) \log_2\left(1 + \frac{P_{i,j}^{tr} h_{i,j}}{\bar{\omega}_0}\right), \tag{1}$$

where $h_{i,j}$ denotes the channel gain, $P_{i,j}^{tr}$ denotes the transmission power, and $\bar{\omega}_0$ denotes the power level of white noise.

3.3. Computation Model

3.3.1. Computing Model for CS. For $x_{i,j}(t) = 0$, vehicle i decides to offload the computation task to CS. Although cloud servers have a huge volume of computing and communication resources, the amount of resources available in each current time slot is still limited considering the maintenance cost of the resources and other factors. The offloading process in this case is divided into the following parts: (i) task transmission between vehicle and its associated RSU, (ii) the process of uploading tasks to CS, and (iii) task processing on CS. According to eq. (1), we can obtain the data transmission rate between vehicle and its associated RSU as $R_{i,j}$. The data transmission rate between CS and RSUs can be obtained as $R_{i,o}$. To sum up, we can obtain the transmission times for the first two processes as

$$T_{i,j}^{tr} = \frac{S_i(t)}{R_{i,j}(t)}, \tag{2}$$

$$T_{i,o}^{tr} = \frac{S_i(t)}{R_{i,o}(t)}, \tag{3}$$

where $S_i(t)$ denotes the task size of the vehicle. We define $f_{i,o}(t)$ as the assigned computational capacity from CS to vehicle. Thus, we can obtain the computation time of this process as

$$T_{i,o}^{cp} = \frac{C_i(t)}{f_{i,o}(t)}, \tag{4}$$

where $C_i(t)$ denotes the required CPU cycles of task. Then, we define the total execution time to offload tasks to CS as $T_{i,j}$. Thus, we have

$$T_{i,j} = T_{i,j}^{tr} + T_{i,o}^{tr} + T_{i,o}^{cp}. \tag{5}$$

Based on the above discussion, the system cost when tasks are offloaded to CS is formulated as

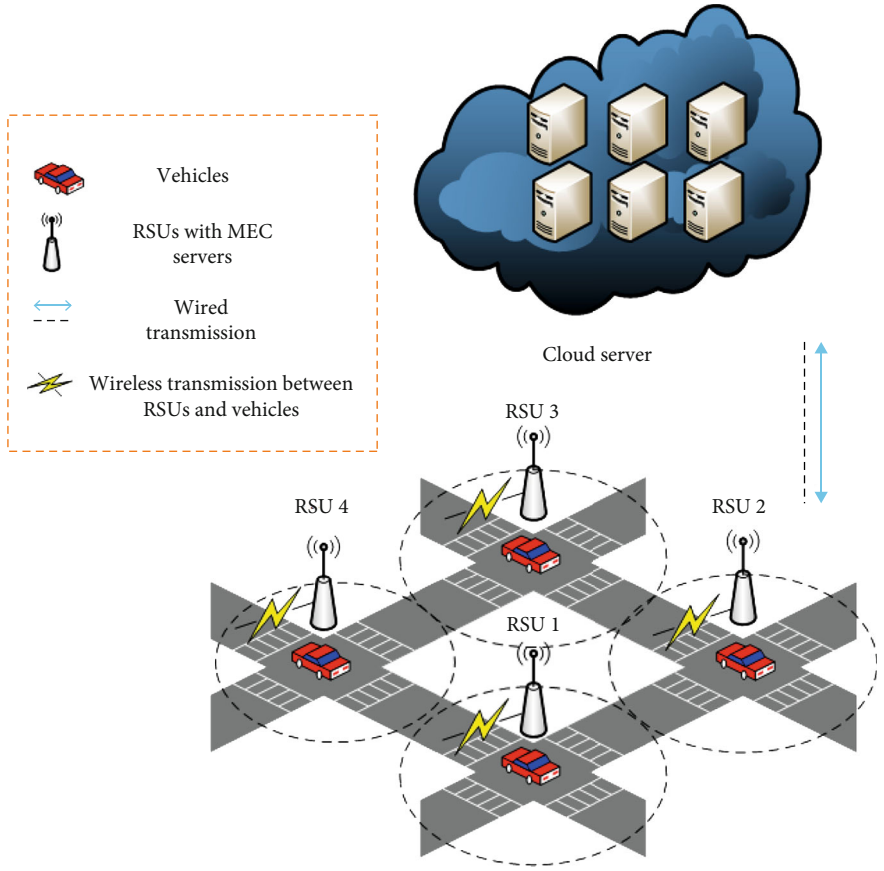


Figure 1: Network description.

Table 1

Notation	Definition
M	A set of RSUs
N	A set of vehicles
$P_{i,j}^{tr}$	The transmission power between vehicle and RSUs
$P_{i,o}^{tr}$	The transmission power between RSUs and CS
P_{CS}	The execution power of CS
P_{RSU}	The execution power of RSUs
$x_{i,j}(t)$	The offloading decision of vehicle i
$C_n(t)$	The CPU cycles of computing task for vehicle i
$S_n(t)$	The task size of computing task for vehicle i
$F_j(t)$	The total computational capability of RSU j
$B_j(t)$	The total radio capability of RSU j
$f_{i,j}(t)$	The computational resources of RSU j allocated to vehicle i
$b_{i,j}(t)$	The radio resources of RSU j allocated to vehicle i

$$E_1 = \sum_{i \in N} \left[\left(1 - x_{i,j}(t)\right) \left(P_{i,j}^{tr} T_{i,j}^{tr} + P_{i,o}^{tr} T_{i,o}^{tr} + P_{CS} T_{i,o}^{cp} \right], \quad (6)$$

where $P_{i,j}^{tr}$ denotes the transmission power between vehicle and RSUs, $P_{i,o}^{tr}$ denotes the transmission power between RSUs and CS, and P_{CS} denotes the execution power of CS.

3.3.2. Computation Model for Associated Offloading. For $x_{i,j}(t) = 1$ and $L_{i,j}(t) = 1$, the vehicle's associated RSU has

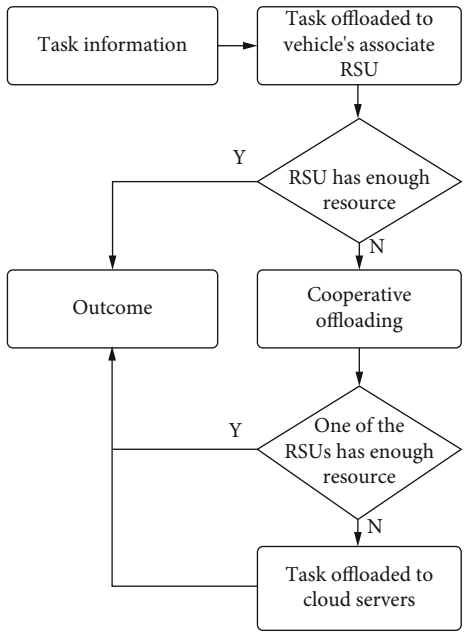


Figure 2: Offload process.

enough resource to process its computing task, then vehicle i decides to offload the computation task to its associated RSU. Thus, the task processing in this case can be divided into two parts: (i) the transmission process from vehicle to its associated RSU and (ii) task processing on the associated RSU. Similar to the above, we have

$$T_{i,j}^{tr} = \frac{S_i(t)}{R_{i,j}(t)}, \tag{7}$$

$$T_{i,j}^{cp} = \frac{C_i(t)}{f_{i,j}(t)}, \tag{8}$$

where $f_{i,j}(t)$ denotes the assigned computational capacity. The total execution time can be obtained as

$$T_{i,j} = T_{i,j}^{tr} + T_{i,j}^{cp}, \tag{9}$$

Based on the above discussion, the system cost for associated offloading is formulated as

$$E_2 = \sum_{i\in N}\sum_{j\in M}\left[x_{i,j}(t)L_{i,j}(t)\left(P_{\mathrm{RSU}}T_{i,j}^{cp} + P_{i,j}^{tr}T_{i,j}^{tr}\right)\right], \tag{10}$$

where P_{RSU} denotes the execution power of RSUs.

3.4. Computation Model for Cooperative RSUs. For $x_{i,j}(t) = 1$ and $L_{i,j}(t) = 0$, since the associated RSU cannot meet its requirements in terms of resources, vehicle i decides to offload the computation task to cooperative RSU j. The task processing at this point consists of three processes: (i) the transmission process from vehicle to its associated RSU, (ii) the transmission process of the task between RSUs, and (iii) task processing on the target RSU j. We define the computational capacity assigned from target RSU to vehicle as $f_{i,j}(t)$, thus, we have

$$T_{i,j}^{tr} = \frac{S_i(t)}{R_{i,j}(t)}, \tag{11}$$

$$T_{r,r}^{tr} = \frac{S_i(t)}{R_{r,r}(t)}, \tag{12}$$

$$T_{i,j}^{cp} = \frac{C_i(t)}{f_{i,j}(t)}, \tag{13}$$

where $R_{r,r}$ denotes the transmission rate between RSUs.

The total task execution time in this case is expressed as

$$T_{i,j} = T_{i,j}^{tr} + T_{r,r}^{tr} + T_{i,j}^{cp}. \tag{14}$$

Combining the above discussion, the system cost when tasks are further offloaded to cooperative RSUs can be formulated as

$$E_3 = \sum_{i\in N}\sum_{j\in M}\left[x_{i,j}(t)\left(1 - L_{i,j}(t)\right)\left(P_{RSU}T_{i,j}^{cp} + P_{i,j}^{tr}T_{i,j}^{tr} + P_{r,r}^{tr}T_{r,r}^{tr}\right)\right], \tag{15}$$

where $P_{r,r}^{tr}$ denotes the transmission power between RSUs.

4. Problem Formulation

In this section, we first calculate the total system cost based on the previous section. Then, we formulate an optimization problem, aiming to minimize the long-term system cost.

Based on the previous section, we define total system cost as follows

$$U(t) = E_1 + E_2 + E_3. \tag{16}$$

By jointly optimizing computational offloading and resource allocation in the proposed system, we formulate the optimization problem with minimizing long-term system cost as the optimization objective, which can be indicated as follows:

$$2(\mathscr{P}_1): \min_{x(t),f(t),b(t)} \bar{U} = \lim_{T\longrightarrow\infty}\frac{1}{T}\sum_{t=0}^{T-1}U(t), \tag{17}$$

$$s.t. x_{i,j}(t) \in \{0,1\}, \tag{18}$$

$$T_{i,j} \le D_i^{\max}, \tag{19}$$

$$\sum_{n\in N} f_{i,j}(t) \le F_j(t), \tag{20}$$

$$\sum_{n\in N} b_{i,j}(t) \le B_j(t). \tag{21}$$

The meanings of the constraints are explained as follows:

(i) Constraint (18) indicates that the decision variable is a Boolean value

(ii) Constraint (19) guarantees that tasks need to be completed within the maximum time delay

(iii) Constraint (20) guarantees that the computation resources allocated by each RSU do not exceed its current available computation resources

(iv) Constraint (21) guarantees that the bandwidth allocated by each RSU does not exceed its current available bandwidth

According to the previous discussion, $x_{i,j}(t)$ represents the Boolean variable for the offloading decision. Meanwhile, $f_{i,j}(t)$ and $b_{i,j}(t)$ represent the computational resources and the bandwidth allocated for the task, respectively. Also, there are nonlinear conditions in the optimization problem. Therefore, optimization problem P_1 is a typical mixed integer nonlinear programming (MINLP) problem [28], which is an NP problem and cannot be solved in polynomial time.

5. Problem Transformation and Solution

In this section, we describe the optimization problem as a Markov decision process (MDP). Next, to solve the optimization problem based on Q-learning method and DQN method, we define state space, action space, and reward function of this problem.

5.1. State, Action, and Reward Definitions

(i) *State Space*. The state space indicates the current state of the environment in the system. In the concerned scenario, the system state of the available resources at current time slot is determined by $B_j(t) - \sum_{n\in N} b_{i,j}(t)$, and $F_j(t) - \sum_{n\in N} f_{i,j}(t)$, which, respectively, represent the available bandwidth and the available computation resources. Moreover, in order to compare among the states to determine if the system has reached the optimal state, we need to obtain the system cost $U(t)$ in each time slot. Hence, the state vector can be obtained as $z_t = (U(t), F_j(t) - \sum_{n\in N} f_{i,j}(t), B_j(t) - \sum_{n\in N} b_{i,j}(t))$

(ii) *Action Space*. In our concerned scenario, agents need to perform multiple actions including developing offloading decisions and deciding how much resource to allocate at each time slot. Therefore, the action vector consists of the offloading decision vector, the computation resource allocation vector, and the bandwidth resource allocation vector. Hence, the action vector in current time slot can be obtained as $d_t = (x_{i,j}(t), f_{i,j}(t), b_{i,j}(t))$

(iii) *Reward Function*. The optimization objective in this paper is to minimize the system cost, which is the opposite of the meaning of the system reward value. Therefore, we define the reward that agents can obtain at state z_t when performing action d_t as $r(z_t, d_t) = -U(z_t, d_t)$, where $U(z_t, d_t) = U(t)$ is the system cost of the current state

5.2. Markov Decision Process.

In this step, we transform the optimization problem into a MDP problem where agents perform adaptive learning and decision making through iterative interactions with the unknown environment. The specific steps are as follows: first, an agent observes the current system state z_t. This intelligence performs action d_t based on the current policy π for each time slot. As a mapping from the current system state to action, policy π can be obtained as $\pi : \mathscr{Z} \longrightarrow p(\mathscr{D} = d \mid \mathscr{Z})$, where $\mathscr{D}$ denotes the set of actions and $\mathscr{Z}$ denotes the set of states. The probability of an agent moving to the next state z_{t+1} is $p(z_{t+1} \mid z_t, d_t)$, and the reward can be obtained as $r_t = r(z_t, d_t)$.

To summarize what was discussed above, a state value function $V^{\pi}(z_t)$ can be defined to indicate the long-term effect of the current state. Hence, the state value function $V^{\pi}(z_t)$ under the policy $\pi(z_t)$ can be expressed as

$$V^{\pi}(z_t) = \mathbb{E}_{\pi}[R_t \mid z_0 = z_t] = \mathbb{E}_{\pi}\left[\left(\sum_{t=0}^{\infty} \varphi^t r_t\right) | z_0 = z_t\right], \tag{22}$$

where z_0 denotes the initial state, and φ denotes the discounting factor indicating the importance of future rewards.

Finally, combined with the optimization objective, the agents need to obtain the optimal strategy in the current state to maximize the cumulative reward. Therefore, the optimization problem can be translated into an optimal state value function as

$$V^{\pi^*}(z_t) = \max_{\pi}\left[r(z_t, d_t) + \varphi \sum_{z_{t+1}} p(z_{t+1} \mid z_t, d_t) V^{\pi^*}(z_{t+1})\right]. \quad \text{s.t.constraintsin}(19) - (22) \tag{23}$$

Thus, the optimal action for state z_t can be obtained as

$$d_t^* = \underset{d_t}{\operatorname{argmax}} V^{\pi}(z_t, d_t). \tag{24}$$

5.3. Q-Learning-Based Solution.

As an efficient value-based model-free iterative learning algorithm, the Q-learning approach enables agents to continuously approximate the optimal Q-value by learning the optimal action in the corresponding environment at each time slot. Specifically, agents of Q-learning method need to obtain the results of the state-value function for each policy and update the two-dimensional Q-table with the corresponding Q-values. Thus, agents can get the optimal strategy for each state based on the magnitude of the Q-values.

Specifically for the content of this paper, we use a Q-learning method to solve the optimization problem. The optimal action values can be defined as $Q(z, d)$, and the optimal state value function can be obtained as

$$V^{\pi^*}(z_t) = \max_{d_t} Q^{\pi}(z_t, d_t). \tag{25}$$

Input: state space $\mathscr{Z}$, action space $\mathscr{D}$, learning rate ε, discount factor φ
Output: the Q-values for every state-action pair
Initialize $Q(z, d)$ arbitrarily for $\forall z \in \mathscr{Z}$, $d \in \mathscr{D}$
for each episode **do**
  for each step of episode **do**
    In the current state z_t choose an action with a random probability ϕ
    If $\phi < \varepsilon$ **then**
      randomly select an action d_t
    else
      select $d_t = \arg\max_d Q(z_t, d_t)$
    end if
    Execute action d_t, observe the reward r_t and the next state z_{t+1}
    Update $Q(z_t, d_t)$ according to eq.(29)
    Update state $z_t \longleftarrow z_{t+1}$
  end for
end for

Algorithm 1: Q-learning-based joint computation offloading and resource allocation algorithm.

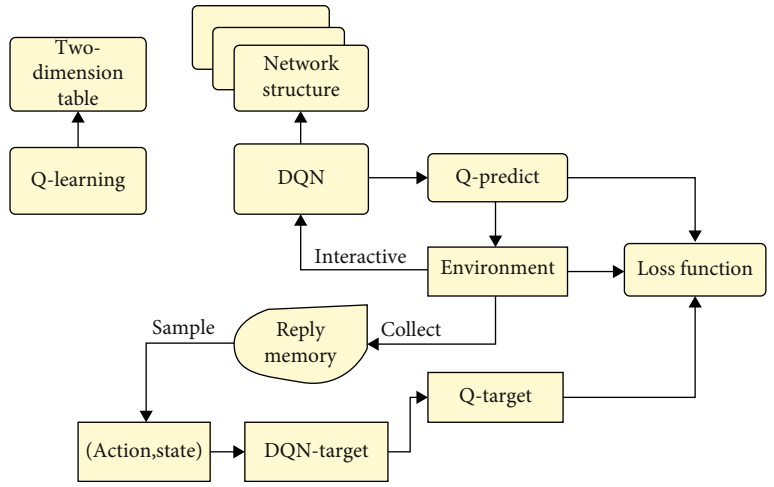


Figure 3: The network structure of DQN.

Therefore, the cumulative reward after performing action d_t can be obtained as

$$Q^{\pi}(z_t, d_t) = r(z_t, d_t) + \varphi \sum_{z_{t+1}} p(z_{t+1} \mid z_t, d_t) V^{\pi^*}(z_{t+1}). \quad (26)$$

Summarizing the two formulas above, the expected reward can be obtained as

$$Q^{\pi}(z_t, d_t) = r(z_t, d_t) + \varphi \sum_{z_{t+1}} p(z_{t+1} \mid z_t, d_t) \max_{d_{t+1}} Q^{\pi}(z_{t+1}, d_{t+1}). \quad (27)$$

The iterative formula for the optimal Q-value can be obtained by updating the state-action function as

$$Q(z_t, d_t) = Q(z_t, d_t) + \varepsilon \left[r(z_t, d_t) + \varphi \max_{d_{t+1}} Q(z_{t+1}, d_{t+1}) - Q(z_t, d_t) \right], \quad (28)$$

where $\varepsilon \in (0, 1)$ denotes the learning rate parameter.

Combining the above discussion, our proposed algorithm is shown in Algorithm 1. In order to make a trade-off between the exploration and the exploitation, we use ε -greedy strategy to choose actions [29].

5.4. DQN-Based Solution. In the time-varying scenarios, we consider the number of vehicles and the size of the tasks are stochastic in nature. This leaves the possibility of a huge action-state space, where although the above Q-learning-based solution can obtain the best policy by updating the Q-table, it may lead to a dimensional disaster in real scenarios. If we stick to the above Q-learning-based solution,

Initialize replay memory set D
Initialize action-value function with random weights θ
Initialize target action-value function with weights $\theta^{-} = \theta$
for episode =1, M
  Initialize sequence $z_1 = \{r_1\}$ and preprocessed sequence $\tau_1 = \tau(z_1)$
  for t =1,2,...,T **do**
    With probability ε select a random action d_t
    Otherwise select $d_t = \underset{d}{\text{argmax}} Q(z_t, d_t; \theta)$
    Execute action d_t, observe the reward r_t and the next state z_{t+1}
    Set $z_{t+1} = z_t, d_t, r_{t+1}$ and preprocess $\tau_{t+1} = \tau(z_{t+1})$
    Store experience $(\tau_t, d_t, r_t, \tau_{t+1})$ in D
    Sample random minibarch of experience $(\tau_i, d_i, r_i, \tau_{i+1})$ from D
    Set $y_i = r_i$ if episode terminates at step $i + 1$
    Otherwise $y_i = r_{t+1} + \varphi \max_d Q(\tau_{i+1}, d_{i+1}; \theta_i)$
    Perform a gradient descent step on $(y_t - Q(\tau_i, d_i; \theta))^2$ with respect to the network parameters θ
    Every C step reset
  end for
end for

Algorithm 2: DQN-based joint computation offloading and resource allocation algorithm.

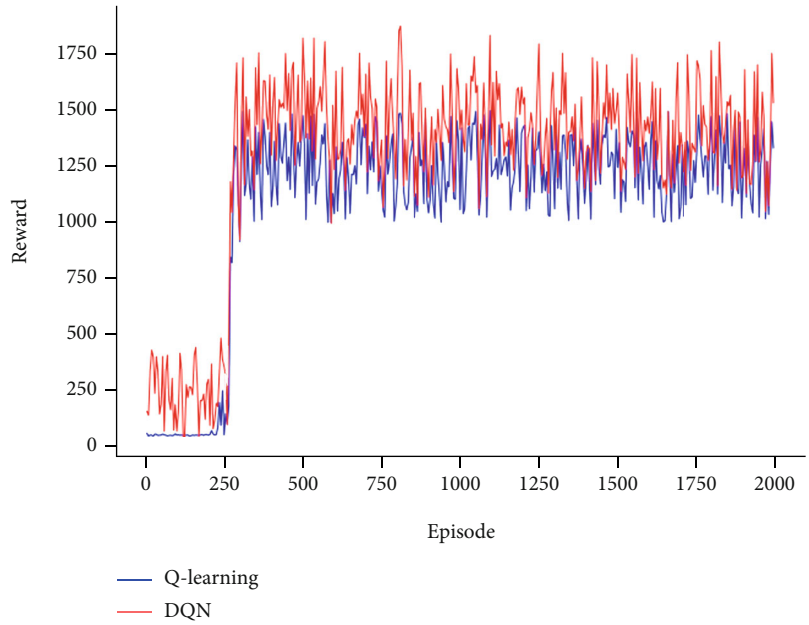


Figure 4: Convergence performance.

finding the corresponding Q-value in a huge Q table can be costly in time and memory.

To avoid this drawback of Q-learning method, we further use a DQN-based approach to solve the optimization problem. Compared to Q-learning, DQN is essentially an improvement method. As a value function approximation, in order to solve the problem of large state space, also known as dimensional disaster, DQN uses the architecture of deep neural network (DNN) to replace Q-table. As a nonlinear approximator of the optimization problem, the DNN in DQN can capture the complex interaction between states and actions [30]. After taking the states as the input to the DQN network, we can get the Q-value of the actions as the output. By doing so, we can estimate the Q-value as $Q(z_t, d_t) \approx Q(z_t, d_t; \theta)$, where θ are the weights of the DQN. Therefore, as in eq. (25), the optimal action in this method can be obtained as

$$d_t^* = \underset{d_t}{\text{argmax}}\ V^{\pi}(z_t, d_t; \theta). \tag{29}$$

In actual engineering application, DQN mainly needs to solve two obvious problems: low sample utilization rate and unstable value obtained by training. In order to deal with these two problems, DQN uses the following two key technologies

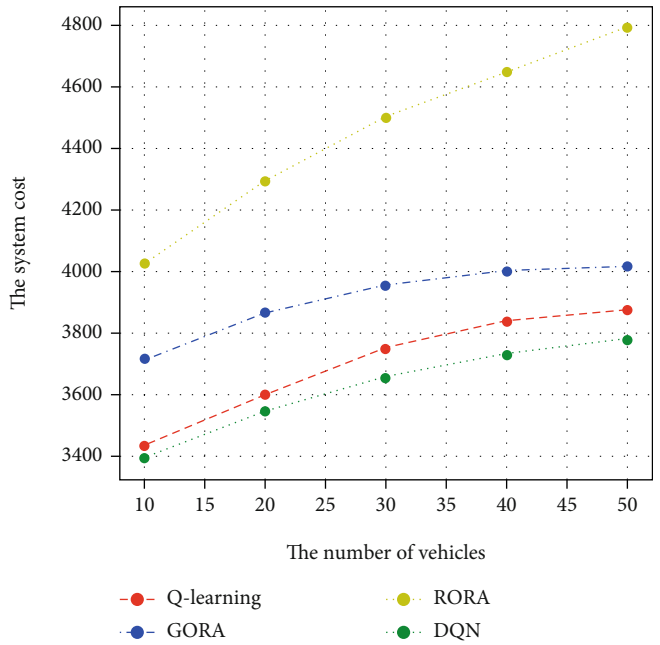


Figure 5: Effects of different number of vehicles on system cost.

(i) *Experience Replay*. An experience pool is constructed to remove data correlations which is a dataset consisting of the recent experiences of the intelligences

(ii) *Freezing Q-Target Networks*. The parameters in the goal are fixed for a time period (or for a fixed number of steps) to stabilize the learning goal

Next, we describe the specific execution steps of DQN. Figure 3 shows the network structure of DQN and the difference between DQN and Q-learning. The network first outputs a prediction Q-value $Q(z_t, d_t)$, then selects the next action based on this Q-value and passes it into the environment for interaction, then obtains a new state value and continues to feed it into the training. At the same time, the results of each interaction with the environment are stored in a fixed-length experience pool. A target Q-network with the same structure and parameters is copied from the Q-network at certain steps to stabilize the output target, and the target Q-network samples the data from the experience pool to output a stable target value y_t. And y_t can be obtained as

$$y_t \equiv r_{t+1} + \varphi \max_d Q(z_{t+1}, d_{t+1}; \theta_t). \tag{30}$$

And the update of the value function of Q-value can be obtained as

$$Q(z_t, d_t) \longleftarrow Q(z_t, d_t) + \varepsilon[y_t - Q(z_t, d_t)]. \tag{31}$$

DQN approximates the value function using a deep convolutional neural network. The value function here corresponds to a set of parameters, which in a neural network are the weights of each layer of the network, denoted by θ. At this point, updating the value function is actually

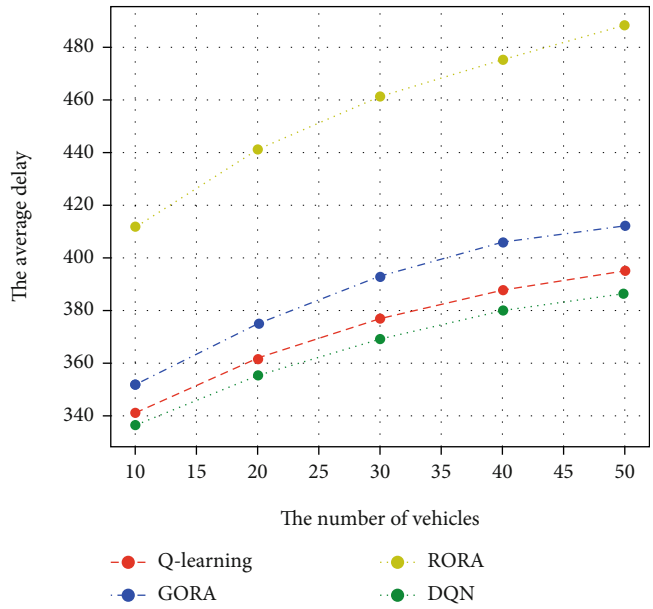


Figure 6: Effects of different number of vehicles on average delay.

updating the parameter θ. When the neural network is determined, θ indicates the value function. And the update method of θ is the gradient descent, which can be expressed as

$$\theta_{t+1} = \theta_t + \varepsilon[y_t - Q(z_t, d_t; \theta_t)]\nabla_{\theta_t} Q(z_t, d_t; \theta_t). \tag{32}$$

In each training iteration, we train the DQN network by minimizing the loss function. In the previous Q-learning based method, we updated the Q-table by iterating through it using the rewards and the current Q-table at each step. Then, we can use this calculated Q-value as

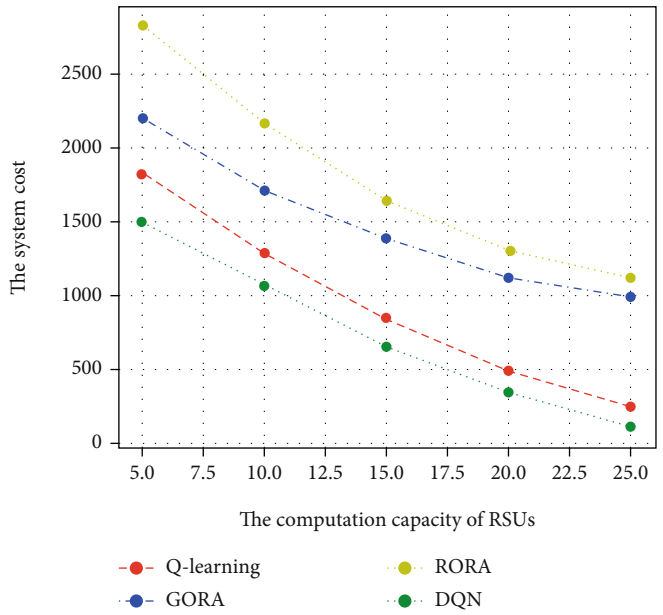


Figure 7: Effects of different computation capacity of RSUs on system cost.

the label to design the loss function, and we use the mean squared difference between the approximate and true values to represent the loss function, which can be obtained as

$$Loss(t) = \mathbb{E}_{\pi}\left[\left(Q(z_t, d_t) - Q(z_t, d_t; \theta_t)\right)^2\right]. \quad (33)$$

To summarize the above about DQN, the specific algorithm steps are shown in Algorithm 2. Same as the Q-learning-based method, we use ε-greedy strategy in the selection of actions.

6. Performance Evaluation

In this section, we evaluate numerical results of the proposed joint computation offloading and resource allocation algorithm in a dynamic edge-cloud network and compare it with other typical schemes.

6.1. Simulation Settings. In the simulation experiments, we consider a dynamic scenario in which there are several vehicles driving in this area and several RSUs distributed on the roadside. Similar to the experimental in [31], for each vehicle, the required CPU cycles of the computing tasks are randomly selected in the range of [0.4, 0.6, 0.7, 0.8, 0.3, 0.2, 0.8, 0.9, 0.4, 0.5, 0.2, 0.3, 0.8, 0.9, 0.4] Gcycles.

6.2. Simulation Results. In the next simulation experiments, in general, our proposed schemes are compared with the random offloading and resource allocation (RORA) scheme and greedy offloading and resource allocation (GORA) scheme [32].

Figure 4 reveals the convergence performance of our two proposed algorithms. In terms of general trends, both curves tend to increase in reward value and then converge after a period of time. However, the number of training episodes

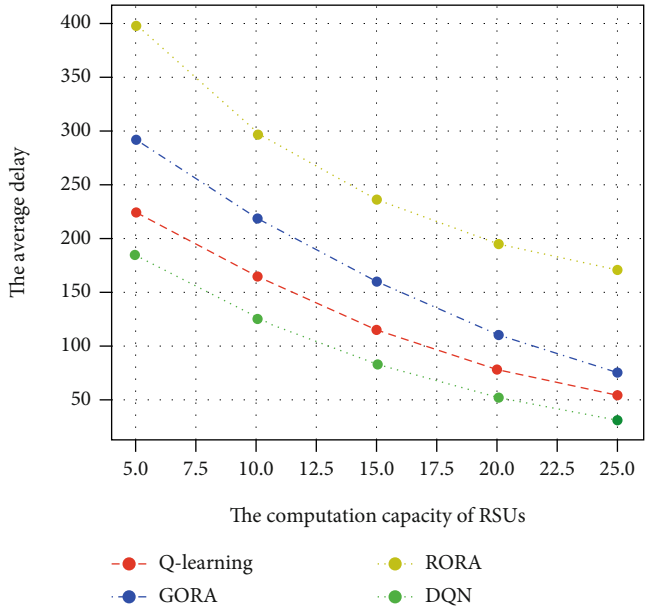


Figure 8: Effects of different computation capacity of RSUs on average delay.

to reach convergence and the final convergence to the reward value differ due to the difference between these two methods. What can be seen from the figure is that the Q-learning method reaches convergence after about 280 training episodes, while the curve of the DQN method reaches convergence after about 250 training episodes. In addition, the DQN-based approach allows for higher reward values at the time of final convergence.

Figures 5 and 6 show how the total system cost and average delay is affected by changes in the number of vehicles, respectively. We intercepted the curve with independent variables changing in the range of 10-50. According to the

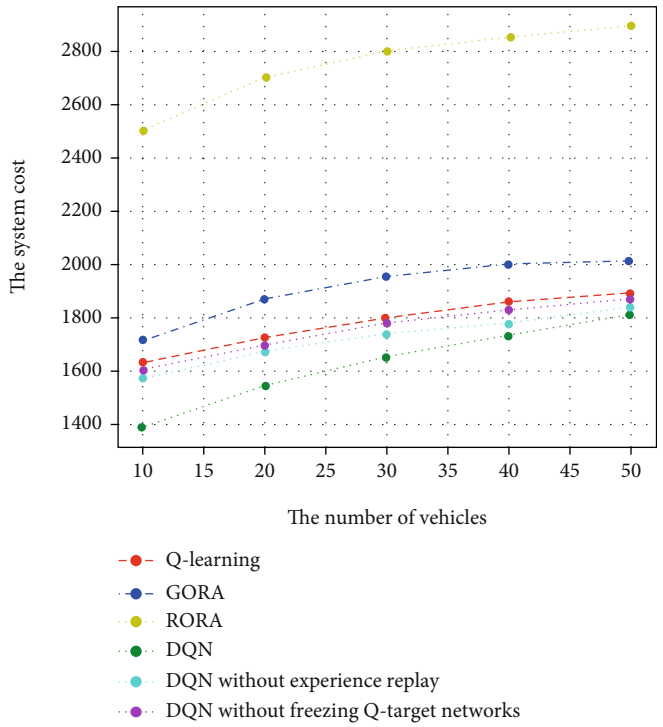


Figure 9: The impact of these two key technologies in evaluation.

optimization objective equation, the number of vehicles has direct effect on the system cost and average delay, so the general trend of all curves is that the system cost and average delay are positively related to the number of vehicles. As joint optimization schemes of computation offloading, the system cost of our proposed Q-learning based algorithm and DQN-based method keeps increasing, and they always work better compared to the other two schemes in the figure. This is because our proposed schemes consider the offloading decision making of RSUs and CS cooperatively, by which the utilization efficiency of network resources can be increased and system cost has been greatly reduced. For example, when the number of vehicles equal to 30, our proposed Q-learning-based method reduces the cost of the system by about 16% and 5% compared to the classical solutions RORA and GORA, respectively. And in the same case, compared with the classical two schemes, our proposed DQN-based scheme reduces the system cost by 18% and 7.5%. When we take the average delay as the dependent variable, the performance improvement of our proposed Q-learning-based method relative to RORA and GORA can reach 18% and 4%. And likewise, the performance improvement of our proposed DQN-based method relative to RORA and GORA can reach 19.9% and 9%.

Figures 7 and 8 show how the total system cost and the average delay are affected by changes in the computation capacity of RSUs, respectively. Compared to the two graphs above, the curves in Figures 7 and 8 show more dramatic changes. In general, the system cost and average latency are reduced with the increase of the computation capacity of RSUs. However, the performance varies due to the different schemes. Obviously, our two proposed solutions have some performance advantages. Specifically, for computational resources, when the computation capacity of RSUs is equal to 30 (in GHz), when compared to RORA and GORA, the performance improvement of our proposed Q-learning-based method can reach 49% and 42%. And the performance improvement of our proposed DQN-based method relative to RORA and GORA can reach 62% and 57%.

In addition, in order to reflect the impact of two key techniques, experience replay and freezing Q-target networks, on DQN method, we tried to add two experiments without one key technology under the same experimental conditions, and the effect is shown in Figure 9. Since the impact of the above two techniques on DQN is mainly in eliminating data correlation and speeding up convergence, the effect of the scheme after removing these two techniques, respectively, is similar to Q-learning method. From the evaluations, these two key technologies have a relatively obvious performance additive effect on the DQN method. Without these two key techniques, DQN method's performance is close to Q-learning method.

In summary, we have compared our proposed scheme with RORA and GORA. From the experimental results, it is clear that the proposed scheme minimizes the system cost and has some advantage over the other two schemes. For the proposed two schemes, compared with the Q-learning-based method, the DQN-based method has a better performance due to the advantages of using deep neural networks.

7. Conclusion

In this paper, we propose a joint computation offloading and resource allocation scheme for the edge-cloud network. The

optimization problem aims to minimize the system cost, including the computation cost and the radio cost. Meanwhile, we also consider the capacity constraints and the latency constraints. Then, we transform the original optimization problem into a Mixed Integer Nonlinear Programming problem. Eventually, a Q-learning-based method for computation offloading and resource allocation is developed to enable tractable analysis. To avoid the dimensionality catastrophe due to the two-dimensional table structure of Q-learning, we further propose a DQN-based algorithm to solve the optimization problem. Through a series of comparative experiments, it is clear that the proposed schemes have good performances in system cost minimization.

Data Availability

The processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant no. U1703261). The corresponding author is Shouzhi Xu.

References

[1] J. Zhao, S. Ni, L. Yang, Z. Zhang, Y. Gong, and X. You, "Multiband cooperation for 5G HetNets: a promising network paradigm," *IEEE Vehicular Technology Magazine*, vol. 14, no. 4, pp. 85–93, 2019.

[2] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4942–4956, 2019.

[3] P. Wang, R. Yu, N. Gao, C. Lin, and Y. Liu, "Task-driven data offloading for fog-enabled urban IoT services," *IEEE Internet of Things*, vol. 8, no. 9, pp. 7562–7574, 2021.

[4] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8769–8780, 2018.

[5] J. Du, L. Zhao, J. Feng, X. Chu, and F. R. Yu, "Economical revenue maximization in cache enhanced mobile edge computing," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, 2018.

[6] Z. Ning, X. Wang, and J. Huang, "Mobile edge computing-enabled 5G vehicular networks: toward the integration of communication and computing," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 54–61, 2019.

[7] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: a novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 711–722, 2020.

[8] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[9] C. Xu, W. Zhao, L. Li, Q. Chen, D. Kuang, and J. Zhou, "A Nash Q-learning based motion decision algorithm with considering interaction to traffic participants," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12621–12634, 2020.

[10] S. Mao, S. Leng, and Y. Zhang, "Joint communication and computation resource optimization for NOMA-assisted mobile edge computing," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, Shanghai, China, 2019.

[11] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[12] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[13] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: a deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10190–10203, 2018.

[14] S. Bi, L. Huang, and Y. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4947–4963, 2020.

[15] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2018.

[16] N. Cheng, W. Xu, W. Shi et al., "Air-ground integrated mobile edge networks: architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[17] C. Lin, Z. Yang, H. Dai, L. Cui, L. Wang, and G. Wu, "Minimizing charging delay for directional charging in Wireless Rechargeable Sensor Networks," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1819–1827, 2019.

[18] E. Wang, D. Li, B. Dong, H. Zhou, and M. Zhu, "Flat and hierarchical system deployment for edge computing systems," *Future Generation Computer Systems*, vol. 105, no. 2020, pp. 308–317, 2020.

[19] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, "Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[20] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Enhancing QoE-aware wireless edge caching with software-defined wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 10, pp. 6912–6925, 2017.

[21] P. Wang, Z. Yu, C. Lin, L. Yang, Y. Hou, and Q. Zhang, "D2D-enabled reliable data collection for mobile crowd sensing," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 180–187, Hong Kong, 2020.

[22] W. Wang, R. Lan, J. Gu, A. Huang, H. Shan, and Z. Zhang, "Edge caching at base stations with device-to-device offloading," *IEEE Access*, vol. 5, pp. 6399–6410, 2017.

[23] Z. Su, M. Dai, Q. Xu, R. Li, and S. Fu, "Q-learning-based spectrum access for content delivery in mobile networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 35–47, 2020.

[24] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.

[25] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2018.

[26] D. Wang, H. Qin, B. Song, X. Du, and M. Guizani, "Resource allocation in information-centric wireless networking with D2D-enabled MEC: a deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 114935–114944, 2019.

[27] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile edge computing," *IEEE Internet of Things Journal*, 2021.

[28] A. Heidari, Z. Y. Dong, D. Zhang, P. Siano, and J. Aghaei, "Mixed-integer nonlinear programming formulation for distribution networks reliability optimization," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 1952–1961, 2018.

[29] Q. Chen, X. He, and W. Meng, "Air-ground cooperative access control algorithm based on Q-learning," in *International Conference on Computing, Networking and Communications*, pp. 461–465, Big Island, HI, USA, 2020.

[30] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2445–2460, 2021.

[31] Z. Ning, K. Zhang, X. Wang et al., "Joint computing and caching in 5G-envisioned internet of vehicles: a deep reinforcement learning-based traffic control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5201–5212, 2021.

[32] H. Zhou, X. Chen, S. He, C. Zhu, and V. C. M. Leung, "Freshness-aware seed selection for offloading cellular traffic through opportunistic mobile networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2658–2669, 2020.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 3877285, 15 pages
https://doi.org/10.1155/2021/3877285

Research Article

TPD: Temporal and Positional Computation Offloading with Dynamic and Dependent Tasks

Mingzhi Wang,[1] Tao Wu,[1] Xiaochen Fan,[2] Penghao Sun,[3] Yuben Qu,[4] and Panlong Yang[5]

[1]*National University of Defense Technology, Hefei 230031, China*
[2]*Centre for Assessment and Demonstration Research, Academy of Military Science, Beijing 100091, China*
[3]*Academy of Military Science, Beijing 100091, China*
[4]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China*
[5]*University of Science and Technology of China, Hefei 230031, China*

Correspondence should be addressed to Tao Wu; terence.taowu@gmail.com and Xiaochen Fan; fanxiaochen33@gmail.com

Received 7 August 2021; Revised 10 September 2021; Accepted 11 October 2021; Published 10 November 2021

Academic Editor: Pengfei Wang

With the rapid development of wireless communication technologies and the proliferation of the urban Internet of Things (IoT), the paradigm of mobile computing has been shifting from centralized clouds to edge networks. As an enabling paradigm for computation-intensive and latency-sensitive computation tasks, mobile edge computing (MEC) can provide in-proximity computing services for resource-constrained IoT devices. Nevertheless, it remains challenging to optimize computation offloading from IoT devices to heterogeneous edge servers, considering complex intertask dependency, limited bandwidth, and dynamic networks. In this paper, we address the above challenges in MEC with TPD, that is, temporal and positional computation offloading with dynamic-dependent tasks. In particular, we investigate channel interference and intertask dependency by considering the position and moment of computation offloading simultaneously. We define a novel criterion for assessing the criticality of each task, and we identify the critical path based on a directed acyclic graph of all tasks. Furthermore, we propose an online algorithm for finding the optimal computation offloading strategy with intertask dependency and adjusting the strategy in real-time when facing dynamic tasks. Extensive simulation results show that our algorithm reduces significantly the time to complete all tasks by 30–60% in different scenarios and takes less time to adjust the offloading strategy in dynamic MEC systems.

1. Introduction

With the rapid development of wireless communication technologies and the pervasive use of the urban Internet of Things (IoT), IoT devices have become deeply integrated into daily life [1]. At the same time, the rapid increase in the number of IoT devices has brought new challenges in processing the massive amount of IoT data [2]. With constrained computing resources and limited battery capacity, IoT devices are facing rapid energy drains as a result of processing computation-intensive tasks [3, 4]. While some initial efforts have been made to solve such problems by leveraging centralized cloud computing, the long communication distance and unstable network connection can cause unacceptable delays to end users [5, 6]. To tackle this issue, mobile edge computing (MEC) has emerged as a promising computing paradigm for providing highly responsive computing services with low latency [7–10]. By offloading computation tasks to edge servers, IoT devices incur fewer costs than they would if processing these tasks locally or remotely at cloud data centers [11].

Nevertheless, applications on IoT devices are highly diverse, and there can be intertask dependency in many computation tasks offloaded to MEC [12]. For instance, Figure 1(a) shows an access control system for MEC. When a user wants to enter, they must swipe their card and pass

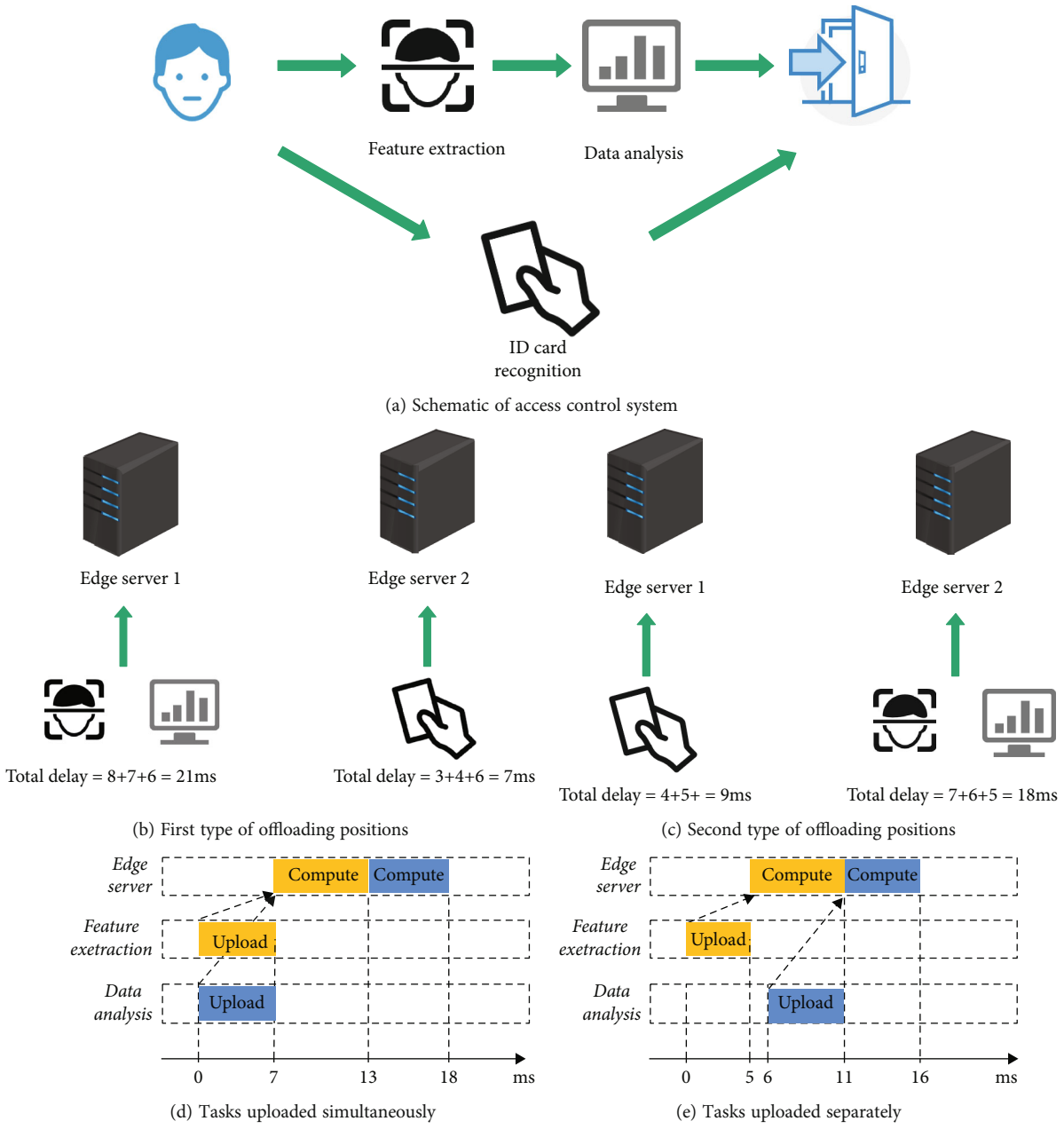


Figure 1: A simple example of an access control system.

facial recognition. The card reader generates the *ID card recognition* task, whereupon the camera takes user pictures and then generates the *feature extraction* task. When the latter is completed, the camera then generates the *data analysis* task to analyze the results of the *feature extraction* task. Intuitively, the door opens only after the *data analysis* and *ID card recognition* tasks have been processed successfully. The arrows in Figure 1(a) show the dependencies among the three computation tasks. Therefore, we must find an optimal offloading strategy for all three tasks. Suppose that there are two edge servers, i.e., edge server 1 and edge server 2, and that edge server 2 has greater computing capability and larger channel bandwidth than those of edge server 1. Correspondingly, the three tasks executed on the edge servers will generate different uploading delays and computation delays, as shown in Table 1. Note that there will be channel interference when tasks are uploaded simultaneously, which will cause longer delays than expected. Specifically, we carefully consider the following two issues of computation offloading with intertask dependency in MEC.

Offloading position: the first step is to find the optimal offloading positions for the three tasks. Here, we have produced two different types of offloading positions for tasks. With the first type (Figure 1(b)), the *ID card recognition* task is offloaded to edge server 2 and its total delay is $3 + 4 = 7$ ms. We assume that the *feature extraction* and *data analysis* tasks are uploaded simultaneously, so that the total delay is $8 + 7 + 6 = 21$ ms (*data analysis* must wait for *feature extraction* to complete). In this case, the door opens after 21 ms. As can be seen, both the *feature extraction* and *data analysis*

Table 1: Delays of three tasks on different servers.

	Feature extraction	Data analysis	ID card recognition
Uploading on S_1 (separated)	6 ms	6 ms	4 ms
Uploading on S_1 (synchronous)	8 ms	8 ms	5 ms
Computing on S_1	7 ms	6 ms	5 ms
Uploading on S_2 (separated)	5 ms	5 ms	3 ms
Uploading on S_2 (synchronous)	7 ms	7 ms	4 ms
Computing on S_2	6 ms	5 ms	4 ms

tasks determine the delay of door opening. However, if we offload both the *feature extraction* and *data analysis* tasks to edge server 2 (Figure 1(c)), then the total delay is reduced to 18 ms. Therefore, it is essential to consider task dependencies when seeking desirable offloading positions.

Offloading moment: if the *feature extraction* and *data analysis* tasks are offloaded simultaneously to edge server 2 at 0 ms (Figure 1(d)), then there is channel interference and the uploading delay of the two tasks becomes 7 ms. Consequently, the door opens after 18 ms. However, if we set the offloading moment of the *data analysis* task as 6 ms (Figure 1(e)), then channel interference is avoided and the overall uploading delay is reduced to 5 ms. Finally, the door opening time is advanced to 16 ms. By introducing this example, we show that the offloading moment also has considerable impact on task processing in mobile computation offloading.

Existing solutions for mobile computation offloading are focused mainly on reducing task response delays, and generally, they offload all computation tasks to MEC and then make optimized adjustments to the edge servers. However, as the above observation cases show, the potential intertask dependency and channel contention will cause significant processing delays in MEC systems with thousands or even millions of IoT devices. Herein, we study the problem of TPD, that is, temporal and positional computation offloading with dynamic-dependent tasks in MEC. In particular, we investigate computation offloading in MEC considering both intertask dependencies and offloading moments. We also address the dynamics of mobile edge networks, where new tasks are involved continuously in computation offloading. Formally, we aim to solve the following three critical challenges.

(i) How to find the optimal offloading positions of tasks under dependency restrictions: with more tasks, the dependencies among them become very complicated. Therefore, a challenge is how to find the optimal offloading position for each task under such complicated dependency restrictions

(ii) How to calculate the offloading moments of tasks to avoid channel interference: the execution and upload delays of tasks are affected by various factors such as channel quality and the computation capabilities of the edge servers. Therefore, it is difficult to estimate these delays, thereby posing the challenge of how to calculate the offloading moments of tasks to reduce channel interference

(iii) How to reduce the time overhead of updating the offloading strategy in real time: in a dynamic MEC system, IoT devices generate new tasks frequently, and the offloading strategy must also be updated frequently. However, doing so incurs a time overhead, so another challenge is how to update the offloading strategy in real time

To overcome the above challenges, we propose an online offloading algorithm known as the critical task first (CTF) algorithm. Combined with the idea of the critical path, the CTF algorithm marks the criticality of tasks based on their dependencies and decides the offloading positions of tasks according to their criticality. When IoT devices generate new tasks in the MEC system, the CTF algorithm can update the offloading strategy in real time with minimal time overhead. The main contributions in this paper are summarized as follows:

(i) We not only decide the offloading positions and execution order of tasks but also calculate the offloading moment of each task to reduce channel interference and uploading delay

(ii) Considering the dependencies among tasks and combining the critical-path idea, we propose an efficient online offloading algorithm, i.e., the CTF algorithm. This uses a backward criticality notation (BCN) method to mark the criticality of tasks, and it can update the offloading strategy with less time overhead in the dynamic MEC system

(iii) We conduct simulations to assess the effectiveness of the CTF algorithm in different scenarios. Compared with previous studies, the CTF algorithm is effective at reducing the time to complete all tasks and updating the offloading strategy with less time overhead

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the system model and problem formulation. Section 4 introduces the main design of the offloading strategy. Section 5 presents the evaluation results. Section 6 concludes the paper.

2. Related Work

As one of the core technologies in MEC, computation offloading has attracted considerable research attention in recent years [13, 14]. According to the optimization goal, computation offloading can be divided into three categories:

(i) minimizing the delay [15–17], (ii) reducing the energy consumption [18, 19], or (iii) balancing the delay and energy consumption [20, 21]. Herein, we consider mainly computation offloading to minimize task delays.

Previous studies of minimizing task delays can be classified into two broad categories: centralized and distributed. The centralized method [22, 23] sets the MEC controller to collect global information with which it makes an optimal offloading decision for each task. Cooperation among multiple mobile devices is presented in [22], wherein the authors construct a potential game to realize task scheduling aimed at minimizing the overhead of each mobile device. In [23], Chen and Hao propose an efficient task-offloading policy in software-defined ultradense networks; they formulate the task-offloading problem as an NP-hard nonlinear mixed-integer programming problem and transform this optimization problem into a task-placement subproblem and a resource-allocation subproblem. Another direction is distributed resource allocation for multiuser MEC systems [24–26]. In [24], by targeting a multiuser and multiserver scenario involving a small-cell network integrated with MEC, the authors formulate the problem as a distributed overhead-minimization problem to minimize the overhead for users.

Dependency-aware computation offloading has also been studied widely. In [27], the authors propose a dependency-aware offloading scheme in MEC with edge–cloud cooperation under task-dependency constraints. In [28], the authors propose a model-free approach based on reinforcement learning, i.e., a Q-learning approach that adaptively learns to optimize the offloading decision and energy consumption jointly by interacting with the network environment. In [29], the authors study the impact of interuser task dependency on task offloading and resource allocation in a two-user MEC network, and they propose an efficient algorithm to optimize such decisions.

In [30, 31], the authors use game theory to solve the problem of optimal offloading. In [31], the authors also consider the dependency among subtasks and the contention among multiple users, and they propose the distributed earliest finish-time offloading (DEFO) algorithm based on noncooperative game theory to reduce the overall completion time of IoT applications. Compared with our CTF algorithm, the DEFO algorithm only decides the offloading positions and execution order of tasks without considering the offloading moments, thereby causing serious channel interference. Also, the DEFO algorithm is designed for static MEC systems, but actual MEC systems change dynamically. By contrast, the proposed CTF algorithm can perform temporal and positional computation offloading according to task dependencies and update the offloading strategy with less time overhead in dynamic MEC systems.

3. System Model

This section introduces the system model. We begin by describing the network model and then present the communication and computation models in detail. For clarity of presentation, Table 2 summarizes the notations used in the following formulation.

Table 2: Notations used herein.

Notation	Definition
$\mathcal{A}$	Set of IoT devices
$\mathcal{S}$	Set of edge servers
$T_{i,j}$	Task j of A_i
$m_{i,j}$	Computation input data size of $T_{i,j}$
$c_{i,j}$	No. of CPU cycles required for $T_{i,j}$
$a_{i,j}$	Offloading position of $T_{i,j}$
$P_{i,j}$	Predecessor task set of $T_{i,j}$
$S_{i,j}$	Successor task set of $T_{i,j}$
$ST_{i,j}$	Offloading moment of $T_{i,j}$
$FT_{i,j}$	Finishing moment of $T_{i,j}$
$r_{i,k}^t$	Data upload rate from A_i to S_k at time t
B_k	Bandwidth of channel connected to S_k
ω	Background noise
q_i	Transmission power of IoT device A_i
$g_{i,k}$	Channel gain
f_i^l	Computation capability of IoT device A_i
f_k^e	Computation capability of edge server S_k
$t_{i,j}$	Total delay of $T_{i,j}$
Φ	Offloading strategy for all tasks
Ψ	Moment at which all tasks are completed

3.1. Network Model. As shown in Figure 2, we consider a typical multiuser and multiserver MEC system with n IoT devices, l nearby edge servers, and wireless access points such as macro base stations and small-cell base stations with relatively powerful and heterogeneous computation abilities. The IoT devices can offload tasks to any edge server through the base stations and the core network. The sets of IoT devices and edge servers are denoted as $\mathcal{A} = \{A_1, A_2, \cdots, A_i, \cdots, A_n\}$ and $\mathcal{S} = \{S_1, S_2, \cdots, S_k, \cdots, S_l\}$, respectively. We suppose that each IoT device generates a set of computationally intensive tasks to be executed during the work. Let $\{T_{i,1}, T_{i,2}, \cdots, T_{i,j}, \cdots, T_{i,m}\}$ denote the set of tasks from IoT device A_i, where $T_{i,j}$ represents task j. For efficient computation offloading, we exploit the potential dependency correlation among the tasks [32]. As in the example in Figure 1(a), executing the *data analysis* task requires completion of the *feature extraction* task as a prerequisite. Therefore, we define the *feature extraction* task as the predecessor task of the *data analysis* task, and the *data analysis* task is its successor task.

Task $T_{i,j}$ can then be described by the five-tuple $(m_{i,j}, c_{i,j}, a_{i,j}, P_{i,j}, S_{i,j})$, where the terms are as follows: (i) $m_{i,j}$ is the size of the input data for task execution (e.g., the program codes and input parameters); (ii) $c_{i,j}$ is the total number of CPU cycles required to accomplish task $T_{i,j}$; (iii) because each task can be either executed locally or offloaded to a nearby edge server, we use $a_{i,j} \in \{0 \cup S\}$ to denote the

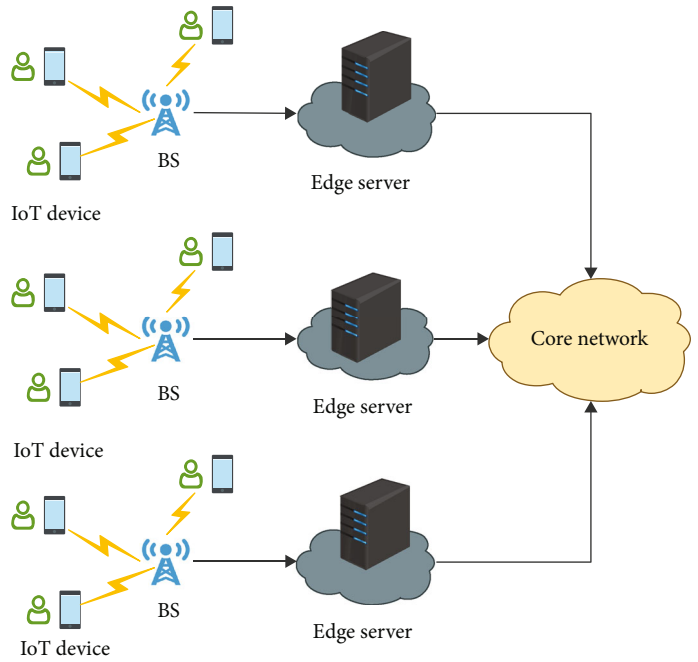


Figure 2: Typical multiuser and multiserver mobile edge computing (MEC) system.

offloading position of task $T_{i,j}$, where $a_{i,j} = 0$ means that task $T_{i,j}$ is executed locally on IoT device A_i and $a_{i,j} = k$ means that $T_{i,j}$ is offloaded to edge server S_k for execution; (iv) $P_{i,j}$ is the predecessor task set of $T_{i,j}$; and (v) $S_{i,j}$ is the successor task set. A task can start only after all its predecessors have been completed. $ST_{i,j}$ is the offloading moment of $T_{i,j}$, and $FT_{i,j}$ is the finishing moment.

3.2. Communication and Computation Models. In this subsection, we introduce how to model the task delay, which can be divided into communication and computation parts. If tasks are executed by the IoT devices themselves, then only a computing delay is incurred; otherwise, the IoT devices select an edge server from the edge server set S for computation offloading, which incurs uploading, waiting, and computing delays. Next, we introduce the communication model followed by the computation model.

Communication model: the communication channel quality changes constantly during the process of computation offloading. In [30, 31], the authors give a formula for calculating the task upload rate. Based on this, we introduce how to calculate the aforementioned communication delays in detail according to the time. $p_{i,k}^t$ indicates whether device A_i is uploading tasks to edge server S_k at time t. If there is a task $T_{i,j}$ such that $a_{i,j} = k$ and its uploading interval contains time t, then $p_{i,k}^t = 1$, otherwise $p_{i,k}^t = 0$. $r_{i,k}^t$ denotes the data upload rate from device A_i to edge server S_k at time t and is given as

$$r_{i,k}^t = B_k \log_2\left(1 + \frac{q_i g_{i,k}}{\varpi + \sum_{i' \in [1,..,n], i' \neq i} q_{i'} g_{i',k} p_{i',k}^t}\right), \quad (1)$$

where B_k is the bandwidth of the communication channel connected to edge server S_k, q_i is the transmission power of A_i, $g_{i,j}$ is the channel gain for the link between A_i and S_k, ϖ is the background noise, and $\sum_{i' \in [1,..,n], i' \neq i} q_{i'} g_{i',k} p_{i',k}^t$ is the wireless interference suffered from other IoT devices when A_i uploads tasks to S_k.

Computation model: a computation task can be either handled locally or offloaded to an edge server for computing, which leads to two different delay computing methods.

(1) *Local computing*: the task is handled on the IoT device itself, and there is only a computing delay in the process. f_i^l is the computing capability of IoT device A_i, which represents the number of CPU cycles that A_i can calculate per second. Given the number of CPU cycles for task $T_{i,j}$, the delay $t_{i,j}^l$ of $T_{i,j}$ with local computing can be represented as

$$t_{i,j}^l = \frac{c_{i,j}}{f_i^l} \quad (2)$$

(2) *Edge computing*: tasks are offloaded to edge servers for computing, and the process involves the following steps: the task is uploaded to the edge server; then, it is queued in the server for execution, and finally, the edge server computes the task to obtain the results. Therefore, these three steps produce a three-part delay (upload delay, queuing delay, and computing delay). The upload delay $t_{i,j}^{k,U}$ of $T_{i,j}$ from

A_i to S_k is determined by the data upload rate $r_{i,k}^t$ and the data volume $m_{i,j}$. Therefore, $t_{i,j}^{k,U}$ is calculated as

$$t_{i,j}^{k,U} = \frac{m_{i,j}}{r_{i,k}^t} \tag{3}$$

After arriving at the edge server, $T_{i,j}$ may need to queue in the server for computing, and $t_{i,j}^{k,Q}$ denotes the queuing delay of $T_{i,j}$ on S_k. We use $Q_{i,j}$ to denote the task set that is queued in front of $T_{i,j}$ on S_k. Therefore, $t_{i,j}^{k,Q}$ is calculated as

$$t_{i,j}^{k,Q} = \sum_{T_{i',j'} \in Q_{i,j}:} \boxtimes \frac{c_{i',j'}}{f_k^e}. \tag{4}$$

f_k^e is the computing capability of edge server S_k, and we calculate the computing delay $t_{i,j}^{k,C}$ as

$$t_{i,j}^{k,C} = \frac{c_{i,j}}{f_k^e}. \tag{5}$$

In *edge computing*, the delay $t_{i,j}^k$ of $T_{i,j}$ comprises the uploading, queuing, and computing delays as

$$t_{i,j}^k = t_{i,j}^{k,U} + t_{i,j}^{k,Q} + t_{i,j}^{k,C}. \tag{6}$$

$t_{i,j}$ is the total delay of $T_{i,j}$ and is denoted as

$$t_{i,j} = \begin{cases} t_{i,j}^l & \text{if } a_{i,j} = 0, \\ t_{i,j}^k & \text{if } a_{i,j} = k. \end{cases} \tag{7}$$

If $T_{i,j}$ chooses *local computing*, then $t_{i,j} = t_{i,j}^l$. If $T_{i,j}$ chooses edge server S_k, then $t_{i,j} = t_{i,j}^k$. Therefore, given $ST_{i,j}$ and $a_{i,j}$ of the task, we calculate the finishing moment as

$$FT_{i,j} = ST_{i,j} + t_{i,j}. \tag{8}$$

3.3. Problem Formulation. Herein, given the dependencies among tasks, the dynamics of the MEC system, and the multiuser wireless interference, we jointly optimize the task offloading position and moment.

(1) Optimizing offloading positions: tasks are either executed locally at the IoT devices or offloaded to different edge servers, thereby causing different delays. Optimizing the offloading positions involves finding the optimal offloading position for each task that minimizes the time to complete all tasks

(2) Optimizing offloading moments: offloading tasks at different moments produces channel interference among IoT devices. Optimizing the offloading moments involves determining the optimal offloading moments to avoid channel interference and reduce the time of uploading tasks

Herein, our goal is to find both (i) the offloading position $a_{i,j}$ at which task $T_{i,j}$ is executed and (ii) the offloading moment $ST_{i,j}$ at which $T_{i,j}$ starts to be offloaded. We use Φ $[(a_{1,1}, ST_{1,1}), \cdots, (a_{i,j}, ST_{i,j}), \cdots, (a_{n,m}, ST_{n,m})]$ to represent the offloading strategy. Our goal is then to find the optimal offloading strategy that minimizes the total time to complete all tasks. Let Ψ denote the moment at which all tasks are completed, which is equivalent to the finishing moment of the final completed task, i.e.,

$$\Psi = \text{Max}\left(ST_{i,j} + t_{i,j}\right), \quad i \in [1,.,n], j \in [1,.,m]. \tag{9}$$

Therefore, we formulate the problem of TPD (temporal and positional computation offloading with dynamic and dependent tasks) mathematically as follows:

$$\text{Opt.} \quad \min\Psi \tag{10}$$

$$\begin{aligned} &\text{s.t.} \quad \left[ST_{i,j}, FT_{i,j}\right] \cap \left[ST_{i',j'}, FT_{i',j'}\right] = \varnothing, \\ &\forall a_{i,j} = a_{i',j'}, i, i' \in [1, n], \quad j, j' \in [1, m], \end{aligned} \tag{11}$$

$$ST_{i,j} \geq \max\left\{ST_{i',j'} + t_{i',j'}\right\}, \quad T_{i',j'} \in P_{i,j}. \tag{12}$$

We seek the optimal offloading strategy Φ with the minimized Ψ (Equation (10)). However, there are two restrictions: (i) an edge server can only calculate one task at a time; therefore, the time intervals of two tasks that select the same server cannot overlap (Equation (11)); (ii) the limitation of dependencies means that a successor task can only start after all its predecessor tasks are completed, and its offloading moment must be later than the finishing moments of all its predecessors (Equation (12)).

3.4. Complexity Analysis of the Problem. Here, we analyze the complexity of the above optimization problem by comparing it with the flexible job-shop scheduling problem (FJSP), which is the problem of scheduling workpiece processing on machines. In the FJSP, a group of workpieces must be processed on a group of machines. Each workpiece involves multiple dependency-aware jobs, a machine can only process one job at a time, and jobs have different execution times on different machines. As a classic NP-hard problem [33, 34], the goal of the FJSP is to determine the execution machines and order of the jobs to reduce the time to complete all jobs.

After abstraction, our optimization problem is similar to the FJSP. The applications running on the IoT devices can be regarded as workpieces, the tasks can be regarded as jobs, and the edge servers can be regarded as machines in the FJSP. Also, the constraints and optimization objectives of the two problems are the same. Therefore, our optimization problem of computation offloading can be regarded as a type of FJSP. However, our optimization problem is more complicated than the classic FJSP because we not only determine

the offloading positions and order of the tasks but must also calculate the precise offloading moments, and we consider the communication time and channel interference. Therefore, the optimal offloading problem proposed by us is also an NP-hard problem.

4. Task Offloading Strategy

In this section, we introduce the detailed task-offloading strategy for our problem. We begin by constructing a directed acyclic graph (DAG) according to the dependency correlation of tasks and proposing a BCN method to mark the criticality of all tasks. We then devise the online CTF offloading algorithm to minimize the time to complete all tasks.

4.1. Marking of Task Criticality. The dependency correlation among tasks means that we can first construct a DAG, as the example shown in Figure 3, to present their topological structure. The nodes in the DAG represent tasks generated by IoT devices, and the directed edges denote the specific dependency relations between tasks. A node is weighted by the number of CPU cycles required by the corresponding task, and the length of a path in the DAG is the sum of the weights of the nodes on the path. The critical path is the longest path in the DAG, and the completion times of tasks on the critical path have a direct effect on the time Ψ.

Intuitively, we can find the critical path and offload tasks on it to edge servers with sufficient computing capability to reduce Ψ. However, searching for the longest path in the DAG using traditional methods such as the Floyd or Dijkstra algorithm would generally incur a large computation time overhead. Also, the critical path will change frequently with the emergence of new tasks. Therefore, we can transform the problem of finding the critical path into marking the criticality of each task. A task with a greater impact on Ψ has higher criticality, and we use $\gamma_{i,j}$ to denote the criticality of task $T_{i,j}$. Next, we introduce the method for calculating the criticality of tasks.

Tasks without successors are called *exit tasks*, such as T_3, T_4, and T_6 in Figure 3, and the path from a task to an *exit tasks* is called an influence path. As shown in Figure 3, T_2 has two influence paths such as $(T_2 \longrightarrow T_4)$ and $(T_2 \longrightarrow T_5 \longrightarrow T_6)$. The length of an influence path of T_2 is the sum of CPU cycles required by tasks on that path and indicates the amount of calculation affected by T_2. We reason that a task that affects a greater amount of calculation is more critical, so we define the criticality of a task as the length of its longest influence path.

On this basis, we propose a BCN method for calculating the length of a task's longest influence path and marking the task's criticality with less time overhead. The criticality of an *exit task* is the number of CPU cycles that it requires. The criticality of a nonexit task depends on its successor tasks and is calculated as

$$\gamma_{i,j} = c_{i,j} + \max \gamma_{i',j'}, \quad T_{i',j'} \in P_{i,j}. \tag{13}$$

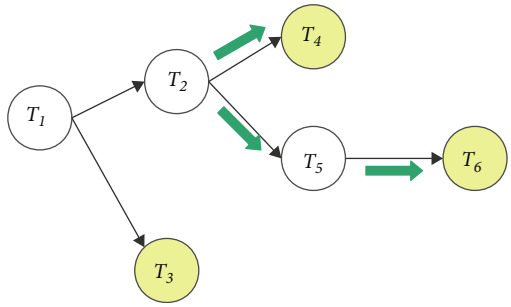


Figure 3: Example of topological diagram of tasks.

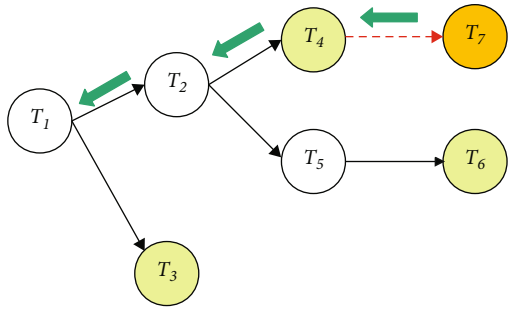


Figure 4: A new task is added to the topological diagram of tasks.

In this way, we can mark the criticality of all tasks. When IoT devices generate new tasks (e.g., T_7), thereby changing the task topology, the BCN method can update the criticality of affected tasks with a small overhead. As an example in Figure 4, T_7 is generated and added to the task topology, thereby affecting its predecessor tasks. Therefore, the BCN method recalculates the criticality of T_4 by means of Equation (13). In this way, the criticality of the other affected tasks (T_1, T_2), as indicated by the arrows in Figure 4, is updated.

4.2. CTF Algorithm. Based on the criticality of tasks, we propose the CTF algorithm. In a dynamic MEC system, the CTF algorithm will monitor the tasks' criticality in real time and find the optimal offloading position and moment for each task to minimize Ψ. The pseudocode of the CTF algorithm is shown in Algorithm 1 and has four stages.

(1) State division

First, the CTF algorithm collects information about all the tasks and divides them (*line 1*) according to the following three states.

(1) *Waiting state*: a task whose predecessor tasks have not been completed

(2) *Ready state*: a task whose predecessor tasks have been completed but have not been offloaded

(3) *Computing state*: a task that has been offloaded and is now computing

Correspondingly, the CTF algorithm generates three types of task queues: the waiting queue, the ready queue,

Input: *Information about tasks, edge servers, and IoT devices*
Output: Offloading strategy with minimized Ψ
1 Put tasks into three types of queues according to task status. Use the BCN method to mark the criticality of tasks. Sort ready tasks from high to low according to criticality. all tasks $T_{i,j}$ in ready queue **do**
2 **for** all S_k in S**do**
3 Calculate $ST_{i,j}^k$ and $FT_{i,j}^k$ on S_k
4 **end**
5 Choose the edge server with the smallest $FT_{i,j}^k$ Calculate $ST_{i,j}^l$ and $FT_{i,j}^l$ on local IoT device $FT_{i,j}^l < FT_{i,j}^k$**then**
6 $FT_{i,j} = FT_{i,j}^l$, $ST_{i,j} = ST_{i,j}^l$, $a_{i,j} = 0$
7 **end**
8 **else**
9 $FT_{i,j} = FT_{i,j}^k$, $ST_{i,j} = ST_{i,j}^k$, $a_{i,j} = k$
10 **end**
11 **end**
12 Go back to step 2 when a new task or new ready task is generated.

Algorithm 1: Critical task first (CTF) algorithm.

and the computing queue. As shown in Figure 5, tasks generated by IoT devices are first put into the waiting queue. After all its predecessors have been completed, a task is moved from the waiting queue to the ready queue. When its offloading moment comes, a ready task is offloaded for computing and then put into the computing queue. Finally, the computation results of tasks are obtained. After the tasks have been divided into the three states, the CTF algorithm uses the BCN method to mark their criticality (*line 2*).

(2) Sorting of ready tasks

Because waiting tasks are not yet ready for computing and computing tasks have been offloaded, we only need to determine when and where the tasks in the ready queue should be offloaded. First, the CTF algorithm sorts the tasks according to their criticality (*line 3*). Based on the principle of critical tasks first, the CTF algorithm decides the offloading positions and moments for critical tasks first and then for other tasks. If two tasks choose the same edge server, the one with the higher criticality is offloaded first, and the other tasks must wait.

(3) Deciding on task offloading

In the process of computation offloading, the CTF algorithm calculates the offloading and finishing moments of the task on each edge server or locally and chooses the position with minimized finishing moment as the final result. Next, using $T_{i,j}$ as an example, we introduce how the CTF algorithm calculates the offloading and finishing moments on edge servers (taking S_k as an example) for ready tasks at time t^{now} (*lines 5–8*). The detailed calculation process can be divided into the following three cases.

Case 1. S_k is idle, and no more-critical tasks have chosen it. In this case, task $T_{i,j}$ can be offloaded directly; the offloading moment on S_k is $ST_{i,j}^k = t^{\text{now}}$, and $T_{i,j}$ is calculated after it arrives on the edge server, as in Figure 6. Therefore, the finishing moment of $T_{i,j}$ on S_k is $FT_{i,j}^k = t^{\text{now}} + t_{i,j}^{k,U} + t_{i,j}^{k,C}$.

Case 2. A task is being executed on S_k but no more-critical tasks have chosen it. As shown in Figure 7, we suppose that $T_{a,b}$ is the task being executed on S_k. We calculate the offloading moment of $T_{i,j}$ as $ST_{i,j}^k = FT_{a,b}^k - t_{i,j}^{k,U}$. Therefore, $T_{i,j}$ can be calculated as soon as it arrives at S_k, and its finishing time is $FT_{i,j}^k = FT_{a,b}^k + t_{i,j}^{k,C}$.

Case 3. A task is being executed on S_k but some more-critical tasks have also chosen it. Although task $T_{a,b}$ is being executed, multiple more-critical tasks have selected S_k, so $T_{i,j}$ must wait for those tasks to complete before it can be executed on the edge server (Figure 8). We use $\varphi_{i,j}$ to represent this more-critical task set. The finishing moment of task $T_{i,j}$ is calculated as

$$FT_{i,j}^k = FT_{a,b}^k + \sum_{T_{i',j'} \in \varphi_{i,j}} t_{i',j'}^{k,C} + t_{i,j}^{k,C}, \tag{14}$$

and the offloading moment of $T_{i,j}$ is $ST_{i,j}^k = FT_{i,j}^k - t_{i,j}^{k,C} - t_{i,j}^{k,U}$.

As shown above, the CTF algorithm controls the offloading moments of tasks so that they (i) do not need to wait on a server, thereby avoiding waiting delay and (ii) are uploaded separately, thereby avoiding channel interference from other IoT devices. Therefore, the upload rate of task $T_{i,j}$ becomes

$$r_{i,k}^t = B_k \log_2\left(1 + \frac{q_i g_{i,k}}{\omega}\right). \tag{15}$$

Finishing moment on local IoT device: On the local IoT device, task $T_{i,j}$ can be calculated directly (*line 9*), and the offloading moment is then $ST_{i,j}^l = t^{\text{now}}$ and $FT_{i,j}^l = ST_{i,j}^l + t_{i,j}^l$.

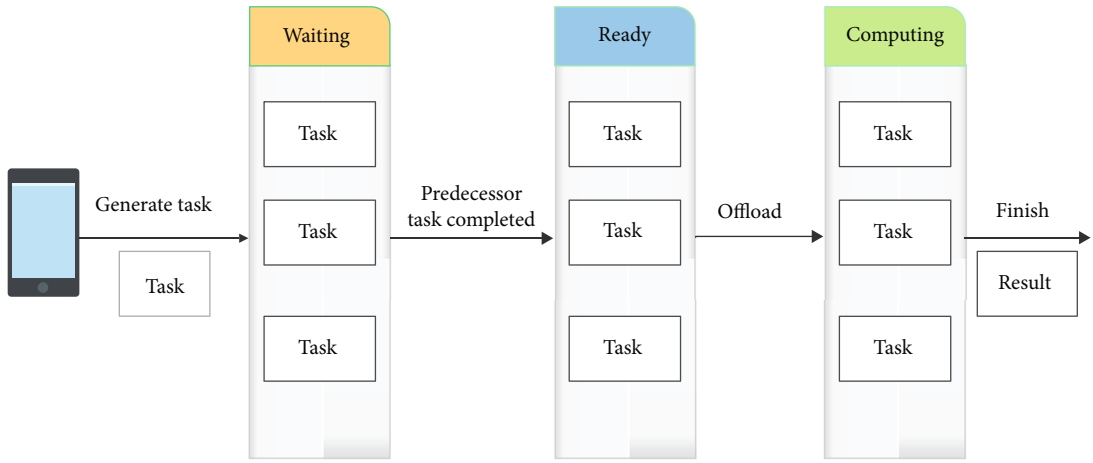


Figure 5: Transformation of the three task states.

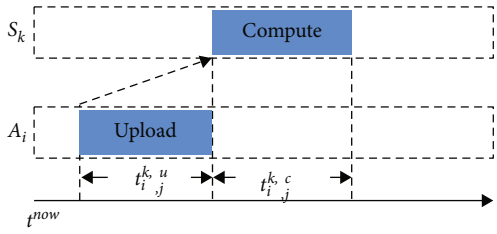


Figure 6: First case of the calculation.

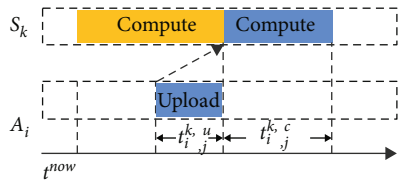


Figure 7: Second case of the calculation.

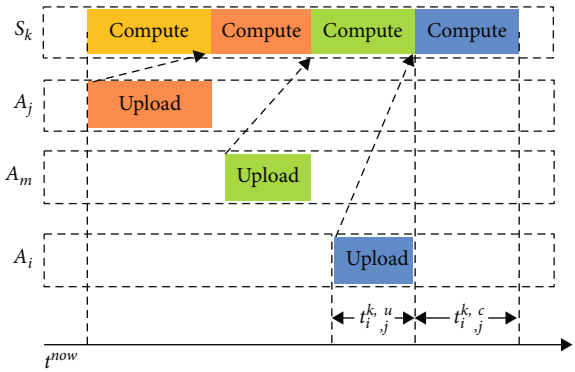


Figure 8: Third case of the calculation.

After calculating the finishing moments of $T_{i,j}$ on all edge servers and locally, the CTF algorithm chooses the offloading position with minimized $FT_{i,j}$, and the offloading moment is calculated according to the offloading position (*lines 10–15*). Therefore, we can obtain the offloading strategy for ready tasks at time t^{now}.

Table 3: Experimental parameters.

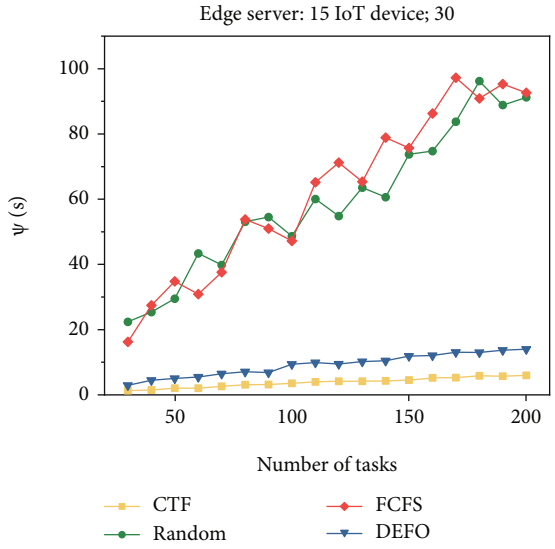
Parameter	Value
Base-station bandwidth	20 MHz ± 20%
Computing capability of edge servers	10 GHz ± 20%
Computing capability of IoT devices	1 GHz ± 20%
Transmission power of IoT devices	100 mW ± 20%
Distance from IoT device to each server	20–50 m
Pass loss factor	4
Task data volume	1000 kB ± 50%
No. of CPU cycles required for task	1000–5000 megacycles
Background noise	−100 dBm

(4) Updating of offloading strategy

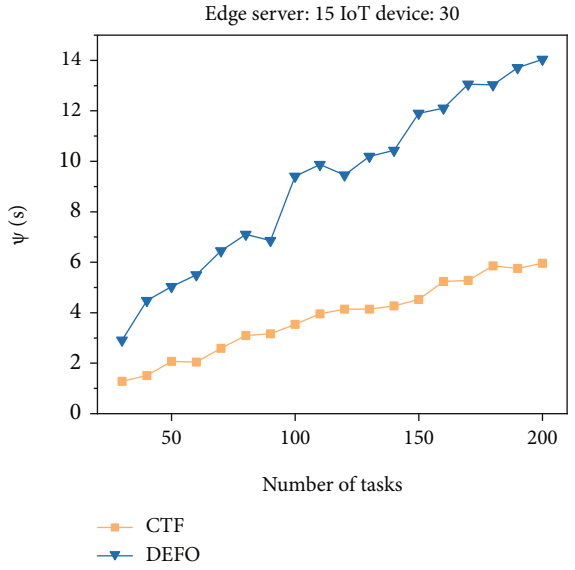
In a dynamic MEC system, the status of tasks changes over time, thereby decreasing the effectiveness of the offloading strategy. In the following two situations, the offloading strategy for ready tasks at time t^{now} may become suboptimal and require recalculation: (i) IoT devices generate new waiting tasks, thereby changing the task topology and thus the task criticality; (ii) new ready tasks are added to the ready queue but the offloading strategy at time t^{now} does not account for them.

As an online algorithm, the CTF algorithm updates the offloading strategy for ready tasks (*line 17*) in time to ensure the best effect. After a new waiting task is generated, the CTF algorithm first uses the BCN method to update the criticality of the affected tasks and then recalculates the best offloading position and moment for each ready task. If only new ready tasks are generated, then the CTF algorithm only needs to recalculate the offloading strategy for ready tasks.

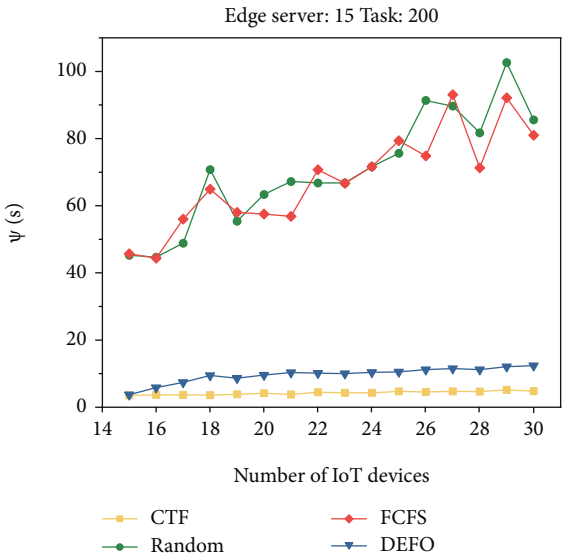
4.3. Complexity Analysis of the CTF Algorithm. Here, we take calculating the finishing moments of tasks as the basic calculation for analyzing the complexity of the CTF algorithm. Suppose that the MEC system has k edge servers and that the IoT devices generate n tasks in total, with Q denoting the number of ready tasks. When deciding the offloading



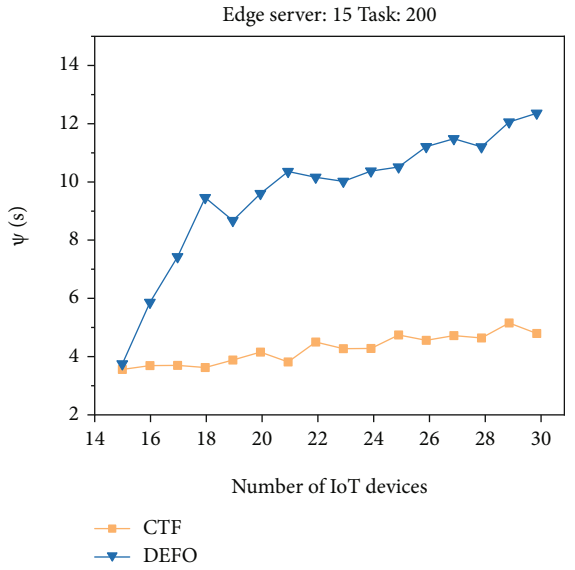
(a) Ψ for different numbers of tasks (4 algorithms)



(b) Ψ for different numbers of tasks (2 algorithms)

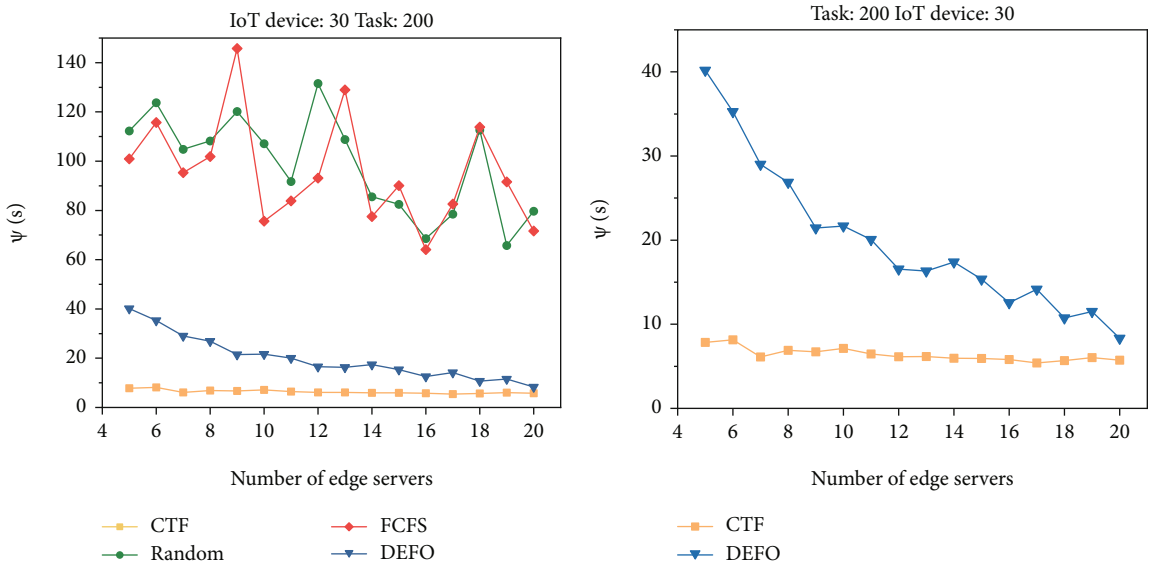


(c) Ψ for different numbers of IoT devices (4 algorithms)



(d) Ψ for different numbers of IoT devices (2 algorithms)

Figure 9: Continued.



(e) Ψ for different numbers of edge servers (4 algorithms)

(f) Ψ for different numbers of edge servers (2 algorithms)

Figure 9: Effects of different algorithms with a static MEC system.

positions of ready tasks, the CTF algorithm calculates the finishing moment of the ready task on each edge server and locally, so it does $k+1$ basic calculations. Therefore, finding the offloading strategy for all ready tasks requires $Q \times (k+1)$ basic calculations.

In the worst case, tasks are generated one at a time and no ready tasks are executed. Therefore, every time a new task is generated by an IoT device and put into the ready queue, the CTF algorithm updates the offloading strategy of all ready tasks. The number of tasks in the ready queue increases gradually from 1 to n, thereby requiring $(1+2+,\cdots,+n)(k+1) = ((n^2+n)/2)(k+1)$ basic calculations in total.

As analyzed above, in a dynamic MEC system containing k edge servers, the CTF algorithm offloads n tasks and requires $((n^2+n)/2)(k+1)$ basic calculations at most. In actual situations, tasks are rarely generated one at a time, and the number of tasks in the ready queue decreases with offloading. Therefore, the actual number of calculations is much less than $((n^2+n)/2)(k+1)$. Every time an IoT device generates a new task, the CTF algorithm requires only $Q \times (k+1)$ basic calculations. However, in some other methods (e.g., noncooperative game methods, genetic algorithms), once a new task is generated, the offloading strategy for all tasks must be recalculated, and the process of iterative convergence is very complicated, thereby generating a large time overhead.

5. Experiments and Evaluation

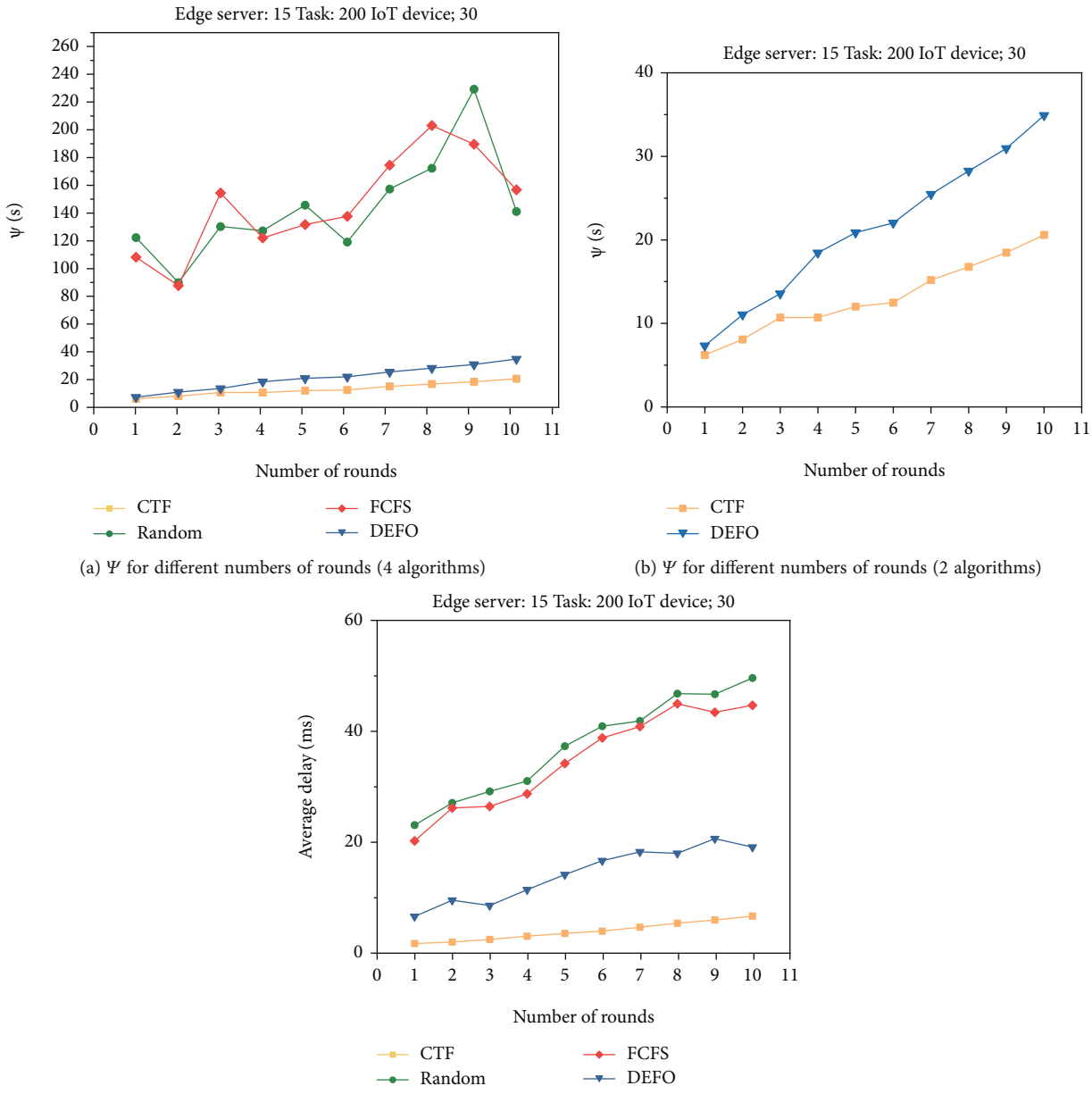
In this section, we report simulation experiments conducted to assess how the CTF algorithm performs with static and dynamic MEC systems. We also compare the CTF algorithm with the DEFO algorithm [31] and two other classic algorithms, i.e., the random algorithm and the first-come first-served (FCFS) algorithm. The random algorithm chooses offloading positions and moments randomly for tasks, whereas in the FCFS algorithm, tasks that are generated first are sent to the nearest edge server and executed first. The design of the simulation experiments and analysis of the results are shown below.

5.1. Experimental Settings. The simulation experiments involved a multiuser and multiserver MEC system with the parameter settings given in Table 3. The channel bandwidth of each base station was 20 MHz ± 20%, and the computing capability of each edge server was 10 GHz ± 20%. The computing capability of the IoT devices was 1 GHz ± 20%, and the transmission power of each IoT device was 100 mW ± 20%. Considering the mobility of IoT devices, the distance l between an IoT device and an edge servers was 20–50 m and changed continuously. The channel gain $g_{i,k}$ from A_i to S_k is $g_{i,k} = l^{\alpha}$, where α is the pass loss factor, and we set $\alpha = 4$. For tasks generated by IoT devices, the task data volume was 1000 kB ± 50%, and the number of CPU cycles required by a task was 1000–5000 megacycles. The background noise was −100 dBm, and the dependencies among tasks were generated randomly.

5.2. Effects of Different Algorithms with Static MEC System. Here, we begin by assessing the performances of the four algorithms with a static MEC system without generating new tasks. To explore how different factors (numbers of tasks, IoT devices, and edge servers) influence the algorithms, we conducted simulation experiments in the following three scenarios.

(1) MEC system with different numbers of tasks

In this experiment, we studied how different task numbers affect Ψ. To prevent the influence of other factors, we set 15 edge servers and 20 IoT devices. The number of tasks in the MEC system was 30 initially and then was increased



(a) Ψ for different numbers of rounds (4 algorithms)

(b) Ψ for different numbers of rounds (2 algorithms)

(c) Average task delay for different numbers of rounds (4 algorithms)

Figure 10: Effects of different algorithms with a dynamic MEC system.

gradually to 200. The Ψ of the offloading strategies generated by the four algorithms varied with the task number, and the results are shown in Figures 9(a) and 9(b).

Figure 9(a) shows that as the number of tasks increases, the computation resources become insufficient, and the Ψ values of the four algorithms increase constantly. In particular, the Ψ growth rates of the FCFS and random algorithms are very high, and their Ψ values are much higher than those of the CTF and DEFO algorithms. For clarity, we plot the results of the CTF and DEFO algorithms separately in Figure 9(b). The CTF algorithm performs better when facing a different number of tasks, and its Ψ values are only approximately half of those for the DEFO algorithm, which shows that the CTF algorithm adapts well to the MEC system with more tasks.

(2) MEC system with different numbers of IoT devices

IoT devices are the sources of tasks in the MEC system, so it is necessary to study how their number influences Ψ. In this experiment, we set 15 edge servers. Because task number is related to the number of IoT devices, we set the former as six times the latter. Starting with a MEC system with only 15 IoT devices, we gradually increased that number to 30, and the results of the experiment are shown in Figures 9(c) and 9(d).

As their number increases, IoT devices compete more fiercely for computing resources, thereby increasing the task execution delay. The Ψ values of the four algorithms continue to increase as the number of IoT devices is increased. However, the experimental results show that the Ψ values

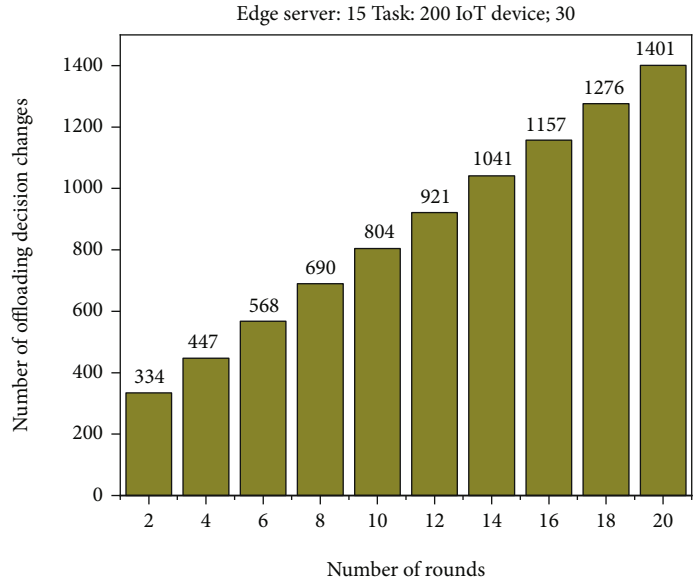


Figure 11: Numbers of decision changes in different rounds.

Table 4: Time overhead and Ψ of DEFO and CTF algorithms.

Round	1	2	3	4	5	6	7	8	9	10
Time overhead of CTF (s)	0.81	1.30	1.16	1.19	1.23	1.30	1.43	1.44	1.27	1.38
Time overhead of DEFO (s)	23.18	30.85	36.42	42.07	60.22	52.93	70.68	105.41	106.22	132.66
Ψ of CTF (s)	6.22	8.07	10.71	10.79	12.01	12.50	15.18	16.76	18.47	20.60
Ψ of DEFO (s)	7.32	11.05	13.56	18.45	20.86 s	22.03	25.45	28.24	30.93	34.91

for the CTF algorithm are 30% of those for the DEFO algorithm when the number of IoT devices exceeds 18 and remain low for the MEC system with different numbers of IoT devices.

(3) MEC system with different numbers of edge servers

As an important computing resource in a MEC system, the number of edge servers has a significant effect on the time to complete all tasks. Therefore, we explore how the number of edge servers influences Ψ. In this experiment, there were 30 IoT devices and 200 tasks. The number of edge servers was 5 initially and was increased gradually to 20, and the results of the experiment are shown in Figures 9(e) and 9(f).

The results show that Ψ can be reduced significantly by having more edge servers. In particular, the Ψ values for the random, FCFS, and DEFO algorithms decrease significantly. By contrast, the CTF algorithm performs better and is more stable in a MEC system with different numbers of servers.

From the three simulation experiments above, we conclude that the CTF algorithm can reduce Ψ significantly in different static MEC systems.

5.3. Effects of Different Algorithms with Dynamic MEC System. Here, we report simulation experiments conducted to assess the performances of the four algorithms with a dynamic MEC system. To build such a system, we added new tasks continuously to simulate the process of IoT devices generating tasks while the algorithms are running. First, we constructed an initial MEC system with 15 edge servers, 30 IoT devices, and 200 tasks. During the execution of the algorithms, we added 20 new tasks to the MEC system in each round. As the number of rounds increased, the MEC system became more dynamic, and the results of the experiment with different numbers of rounds are shown in Figure 10.

The enhanced dynamics of the MEC system decrease the effectiveness of the offloading strategy. Figure 10(a) shows that as the number of rounds increases, the Ψ values of each algorithm increase significantly, but those for the CTF algorithm are lower, as is their growth rate. In particular, the Ψ values for the CTF algorithm are only 60% of those for the DEFO algorithm (Figure 10(b)). In addition to Ψ, the average task delay is also affected. Figure 10(c) shows that as the number of rounds increases, so does the average task delay. However, the average task delay with the CTF algorithm is significantly lower than that with the other three algorithms, which means that the CTF algorithm adapts better to the dynamic MEC system.

In a dynamic MEC system, the CTF algorithm adjusts the offloading positions and moments of tasks continuously to maintain the most effective offloading strategy. Figure 11

shows the numbers of changes in offloading positions and moments of tasks made by the CTF algorithm in different rounds. As the number of rounds increases, the MEC system becomes more dynamic, and the CTF algorithm adjusts the offloading strategy frequently to ensure the best results.

An online algorithm incurs a short time overhead to update the offloading strategy in a dynamic MEC system. Here, we study the time overhead of the DEFO and CTF algorithms with the dynamic MEC system. The DEFO and CTF algorithms were both programmed in Python on a Windows 10 platform and tested in an Intel 2.60 GHz, 16 GB memory environment. We measured the time overhead of running the two algorithms in different rounds, as given in Table 4. Clearly, the time overhead of the CTF algorithm under different rounds is short and stable. With 10 rounds, the time overhead of the DEFO algorithm is 100 times that of the CTF algorithm. Compared with Ψ, the time overhead incurred by the CTF algorithm is acceptable, whereas that for the DEFO algorithm is too long to deal with the dynamic MEC system. Therefore, the CTF algorithm can update the offloading strategy in the dynamic MEC system with less time overhead and deliver better results.

6. Conclusion

Herein, we studied the problem of temporal and positional computation offloading in a dynamic MEC system with dependent tasks. Unlike previous studies, we also considered the impact of dependencies on the offloading positions of tasks and calculated the offloading moments of tasks accurately to reduce channel interference. To cope with dynamic tasks, we proposed the CTF algorithm, which updates the offloading strategy in real time to ensure the most effective computation offloading. Simulation experiments were conducted, and the results showed that the CTF algorithm reduces significantly the time to complete all tasks and incurs less time overhead.

Data Availability

The simulated data used to support the findings of this study are included within the article, In the fifth chapter of our manuscript, "Experiments and Evaluation," we introduce in detail how to generate the simulation data needed for the experiment. Any authors can reproduce the experiment according to our data generation method, so we do not give additional experimental data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported partially by the National Natural Science Foundation of China under Grant Nos. 62002377, 62072424, 61772546, 61625205, 61632010, 61751211, 61772488, and 61520106007; Key Research Program of Frontier Sciences, Chinese Academy of Sciences (CAS), No. QYZDY-SSW-JSC002; NSFC with No. NSF ECCS-1247944 and NSF CNS 1526638; and in part by the National Key Research and Development Plan Nos. 2017YFB0801702 and 2018YFB1004704.

References

[1] X. Fan, C. Xiang, C. Chen et al., "BuildSenSys: reusing building sensing data for traffic prediction with cross-domain learning," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2154–2171, 2021.

[2] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757–1771, 2016.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] I. Lee and K. Lee, "The Internet of Things (IoT): applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[5] P. Mell and T. Grance, *The Nist Definition of Cloud Computing*, National Institute of Standards and Technology, NIST Special Publication 800-145, 2011.

[6] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: A berkeley view of cloud computing," *Science*, vol. 53, no. 4, pp. 50–58, 2010.

[7] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, 1611 pages, 2013.

[8] S. Deng, Z. Xiang, P. Zhao et al., "Dynamical resource allocation in edge for trustable Internet-of-Things systems: a reinforcement learning method," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6103–6113, 2020.

[9] W. Lu, Z. Ren, J. Xu, and S. Chen, "Edge blockchain assisted lightweight privacy-preserving data aggregation for smart grid," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1246–1259, 2021.

[10] Q. Ye, H. Bie, K. C. Li et al., "EdgeLoc: a robust and real-time localization system towards heterogeneous iot devices," *IEEE Internet of Things Journal*, 2021.

[11] T. Wu, X. Fan, Y. Qu, and P. Yang, "Mobiedge: mobile service provisioning for edge clouds with timevarying service demands," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 1–6, Beijing, China, 2021.

[12] M. Wang, T. Ma, T. Wu, C. Chang, F. Yang, and H. Wang, "Dependency-aware dynamic task scheduling in mobile-edge computing," in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pp. 785–790, Tokyo, Japan, December 2020.

[13] W. Lu, S. Zhang, J. Xu, D. Yang, and L. Xu, "Truthful multiresource transaction mechanism for P2P task offloading based on edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6122–6135, 2021.

[14] C. Zhang, H. Tan, H. Huang et al., "Online dispatching and scheduling of jobs with heterogeneous utilities in edge computing," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol*

Design for Mobile Networks and Mobile Computing, pp. 101–110, New York, NY, USA, October 2020.

[15] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 29–43, 2020.

[16] Y. Nan, W. Li, W. Bao et al., "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE Access*, vol. 5, pp. 23947–23957, 2017.

[17] X. Fan, P. Yang, and Q. Li, "Fairness counts: Simple task allocation scheme for balanced crowdsourcing networks," in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 258–263, Shenzhen, China, December 2015.

[18] X. Liu and N. Ansari, *Green/energy-efficient design for D2D*, American Cancer Society, 2019.

[19] H. Liu, L. Cao, T. Pei, Q. Deng, and J. Zhu, "A fast algorithm for energy-saving offloading With Reliability and latency requirements in multi-access edge computing," *IEEE Access*, vol. 8, pp. 151–161, 2020.

[20] X. Lan, L. Cai, and Q. Chen, "Execution latency and energy consumption tradeoff in mobile-edge computing systems," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 123–128, Changchun, China, August 2019.

[21] X. Fan, X. He, D. Puthal et al., "CTOM: collaborative task offloading mechanism for mobile cloudlet networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[22] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access*, vol. 5, pp. 5609–5622, 2017.

[23] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[24] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 271–278, New York, NY, USA, November 2015.

[25] S. Deng, Z. Xiang, J. Taheri et al., "Optimal application deployment in resource constrained distributed edges," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1907–1923, 2021.

[26] W. Xu, X. Fan, T. Wu, Y. Xi, P. Yang, and C. Tian, "Interest users cumulatively in your ads: a near optimal study for Wi-Fi advertisement scheduling," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, Vancouver, BC, Canada, May 2021.

[27] L. Chen, J. Wu, J. Zhang, H. N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE Transactions on Cloud Computing*, p. 1, 2020.

[28] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, "Dependency-Aware computation offloading in mobile edge computing: a reinforcement learning approach," *IEEE Access*, vol. 7, pp. 134742–134753, 2019.

[29] J. Yan, S. Bi, Y. J. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235–250, 2020.

[30] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2762–2773, 2018.

[31] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multiuser offloading for edge computing networks: a dependency-aware and latency-optimal approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2020.

[32] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4242–4251, 2019.

[33] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[34] A. Soukhal, A. Oulamara, and P. Martineau, "Complexity of flow shop scheduling problems with transportation constraints," *European Journal of Operational Research*, vol. 161, no. 1, pp. 32–41, 2005.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 7821241, 12 pages
https://doi.org/10.1155/2021/7821241

Research Article

Recognition for Human Gestures Based on Convolutional Neural Network Using the Off-the-Shelf Wi-Fi Routers

Haixia Yang,[1] Zhaohui Ji,[2] Jun Sun,[3] Fanan Xing,[2] Yixian Shen,[2] Wei Zhuang,[2] and Weigong Zhang[1]

[1]*School of Instrument Science and Engineering, Southeast University, Nanjing 211189, China*
[2]*School of Computer, Nanjing University of Information Science and Technology, Nanjing 210044, China*
[3]*College of Computer and Information, Hohai University, Nanjing 210098, China*

Correspondence should be addressed to Weigong Zhang; zhangwg@seu.edu.cn

Received 21 August 2021; Revised 4 October 2021; Accepted 9 October 2021; Published 3 November 2021

Academic Editor: Pengfei Wang

Human gestures have been considered as one of the important human-computer interaction modes. With the fast development of wireless technology in urban Internet of Things (IoT) environment, Wi-Fi can not only provide the function of high-speed network communication but also has great development potential in the field of environmental perception. This paper proposes a gesture recognition system based on the channel state information (CSI) within the physical layer of Wi-Fi transmission. To solve the problems of noise interference and phase offset in the CSI, we adopt a model based on CSI quotient. Then, the amplitude and phase curves of CSI are smoothed using Savitzky-Golay filter, and the one-dimensional convolutional neural network (1D-CNN) is used to extract the gesture features. Then, the support vector machine (SVM) classifier is adopted to recognize the gestures. The experimental results have shown that our system can achieve a recognition rate of about 90% for three common gestures, including pushing forward, left stroke, and waving. Meanwhile, the effects of different human orientation and model parameters on the recognition results are analyzed as well.

1. Introduction

Gesture is a common visual body activity that contains a rich content of indications. Unlike physical device-based control methods, gesture control does not require specific devices for smart home applications such as keyboards, mice, and remote controller. Nowadays, human gesture is becoming a promising indication for human-computer interaction due to the contactless and device-free characteristics for many Internet of Things applications [1].

In traditional gesture recognition systems, a variety of sensing devices such as high-resolution cameras and various sensors have been proved to be able to recognize gesture activities [2]. But these sensing devices have great limitations in installation difficulty, privacy security, and installation cost [3]. In contrast, wireless signal gradually becomes a very attractive carrier for gesture recognition. Because with the advent of the urban Internet of Things, Wi-Fi devices will become ubiquitous, and Wi-Fi signals also spread throughout the home environment [4]. Wireless signals do not need to be received in the range of sight and can easily penetrate walls, so Wi-Fi has the advantages of easy deployment and low cost in activity recognition [5].

Therefore, Wi-Fi is easy and inexpensive to deploy for motion recognition. At the same time, the gesture recognition based on Wi-Fi signal can also realize the gesture recognition in a more private way without strong user perception, and the security of privacy is also guaranteed.

In the nineteenth century, the famous physicist Fresnel proposed the Fresnel zone model, which is widely used in the field of optical wave dynamics to explain the interference and diffraction of light. With the development of wireless sensing, researchers found that the propagation of radio frequency signal can also be explained by the Fresnel zone model. Different gestures will crossdifferent Fresnel zone at different angles and speeds, so the interference is also

different, and the final CSI sequence is also different. This provides a theoretical basis for identifying different gestures through the CSI sequence of Wi-Fi devices.

2. Related Works

The gesture recognition based on the wireless signal of commercial Wi-Fi devices can be roughly divided into two categories, one is the early RSS-based system, and the second is the more popular CSI-based system at present.

In 2015, Abdelnasser et al. proposed a gesture recognition system WiGest [6]. The system identified five gestures by analyzing received signal strength (RSS) changes. The recognition rate reaches 87.5% when using a single wireless access point and more than 96% when there are multiple wireless access points. However, RSS can only get a value of overall signal change in measurement, so the RSS-based perceptual recognition system can only identify some coarse-grained human activities. In 2010, Halperin et al. from the University of Washington, in collaboration with Intel, released a tool called CSITOOL based on the Intel 5300 network card [7]. This tool provides a method to extract CSI values from commercial Wi-Fi devices on Linux systems. Since then, passive sensing technology based on CSI of commercial Wi-Fi devices soon became a research hotspot in wireless sensing. In 2018, Yu et al. from Nanjing University proposed a QGesture gesture recognition system [8]. This system separates gesture movements from the daily environment by modeling the amplitude and phase changes in CSI with the direction and distance of gesture movements. The system finally achieves more than 90% recognition rate in the presence of multiple interfering daily activities. Jiang et al. at the State University of New York at Buffalo, USA, proposed an EI framework for device-free activity recognition based on deep learning techniques [9]. The core module of the EI framework is an adversarial network, which consists of three main components: feature extractor, human activity recognizer, and environment discriminator. The EI framework removes the environmental information contained in the activity data in the process of integrated learning, and the recognition rate reached more than 90% [10]. Zheng et al. of Tsinghua University proposed a gesture recognition system called WIDAR3.0 in 2019 [11]. This system adopts an environment-independent signal feature—velocity spectrum BVP in human body coordinate system. It applies time-frequency analysis to CSI sequence to estimate Doppler frequency shift profile, restores BVP sequence, and realizes crossdomain gesture recognition. Its intradomain recognition rate for pushing, waving, and other activities reaches 92.7%, and the crossdomain recognition rate reaches between 82.6% and 92.4%.

It is not difficult to see from the above; using channel state information for behavior recognition has become the mainstream scheme in the field of wireless perception. Moreover, the behaviors that can be identified through CSI gradually shift from coarse-grained daily activities such as falls and walking to fine-grained actions such as gestures and gaits. In this paper, we proposed a gesture recognition system based on the CSI of Wi-Fi devices. The main content of our paper is structured as follows: in Section 2, we outlined the basic content of the system design. In Section 3, we introduced the Fresnel zone model, discussed the influence of gesture activity on CSI signal, and finally explained the characteristics and principles of CSI quotient model. In Section 4, the principles and steps of the four basic modules, namely, system data acquisition and extraction, CSI data preprocessing, gesture feature extraction, and gesture classification and recognition, were detailed. In Section 5, we designed the experimental scheme and made a comprehensive analysis and comparison of the experimental results. In Section 6, we summarized the system design process.

3. The Analysis of Wi-Fi Signals of Gesture Activities

3.1. Channel State Information of Wi-Fi. CSI estimates the channel information by representing the channel attribute of the communication link, which can reflect the influence of the environment on the signal transmitted from the transmitter equipment to the receiver [12]. CSI can be extracted from commercial Wi-Fi devices based on 802.11.n protocol. Wi-Fi equipment adopts multiple-input multiple-output (MIMO) technology. A communication system consists of multiple transmitting antennas and receiving antennas. Therefore, CSI data collected from commercial Wi-Fi devices include multiple subcarriers on different receiving and transmitting antenna channels. Each pair of amplitude and phase of CSI signal describes the state information of subcarriers.

In a narrowband smooth fading channel, the orthogonal frequency division multiplexing system in the frequency domain is modeled as

$$Y = HX + N. \tag{1}$$

In the formula, Y and X represent the received signal vector and the transmitted signal vector, respectively, and H and N represent the channel matrix and the random Gaussian white noise vector, respectively. The channel state information of the subcarrier can be estimated from the above equation as

$$\widehat{H} = \frac{y}{x}. \tag{2}$$

It describes the channel gain between the transmitting antenna of a commercial Wi-Fi device and the receiving antenna of a network card, where the CSI of a single subcarrier is mathematically represented as

$$h = |h| e^{j \bullet \sin(\angle h)}, \tag{3}$$

where $|h|$ is the amplitude and $\angle h$ is the phase of each subcarrier. CSI-based sensing systems usually use the amplitude and phase values of CSI to achieve the sensing of action behavior.

3.2. The Influence of Gesture Activities on CSI. In this paper, we design a preexperiment to verify the interference of hand gestures to Wi-Fi signal CSI sequences. In the case of no change in the position of the receiving end and the sending end, the experimenters made two different gestures of push and clap. The Savitzky-Golay filter is used to smooth the CSI amplitude, and the final CSI amplitude sequence is shown in Figures 1 and 2, respectively. Through comparative analysis, we can draw the following conclusions. Gesture movements will have an impact on the CSI sequence waveform, and different gesture movements have different impacts on the CSI sequence waveform. These conclusions provide a theoretical basis for the recognition of different gestures through CSI sequence.

3.3. CSI Quotient Model. The CSI amplitude extracted in the actual typical indoor environment has noise due to hardware defects and environmental interference, which greatly limits the clarity of CSI data. However, in the recognition of fine-grained gestures, there is a high requirement for the clarity of CSI sequence. The amplitude of CSI can be filtered to reduce some ambient noise so as to restore a relatively regular CSI amplitude sequence. However, there is a random offset noise in the CSI phase sequence that is difficult to compensate, which completely limits the use of CSI phase information in monitoring the direction and displacement of hand movement.

In 2019, Zeng et al. proposed the CSI quotient model to push the range limit of Wi-Fi-based respiration sensing [13]. This model abandons the CSI value collected by the original single antenna and takes the quotient of the CSI values received by the two adjacent antennas at the receiver as a new metric. Mathematically, the magnitude of the CSI quotient is the quotient of the original CSI magnitude, while the phase of the CSI quotient is the phase difference of the original CSI data.

$$\mathrm{CSI}_{\mathrm{quotient}} = \frac{\mathrm{CSI}_{\mathrm{antenna1}}}{\mathrm{CSI}_{\mathrm{antenna2}}}. \tag{4}$$

The CSI quotient model is based on the following two conclusions. (1) For commercial wireless NETWORK CARDs, the same set of crystal clocks and RF processing circuits are shared between different antennas on the NETWORK CARD, and their amplitude noise is equally proportional. At the same time, the time-varying phase offsets of different antennas are the same. (2) When the observation target moves for a small distance, the difference of the reflected path length between two adjacent antennas can be considered as a constant.

In order to further compare the difference between original CSI and CSI quotient, we conducted a comparative experiment. During the experiment, the volunteer made a pushing forward gesture within 1 m of the receiving and transmitting equipment of the receiver and then extracted the CSI values and CSI quotient amplitude sequences collected on the original two receiving antennas for comparison. The amplitude sequence of CSI value is shown in Figures 3 and 4, and the amplitude sequence of CSI quotient is shown in Figure 5. It can be seen that the amplitude waveform of CSI quotient is obviously better than that of the other two original CSI values.

From the above analysis and comparison experiments, it is easy to find that the CSI quotient has more excellent characteristics than the original CSI value. CSI quotient can also provide better performance addition to the system when used for gesture recognition. Therefore, the CSI quotient model is used for gesture recognition in this paper.

4. Design of Gesture Recognition System

4.1. Software Environment. The system is built on personal laptop. This study uses Python to parse, preprocess, extract features, and classify CSI gesture data and uses Pycharm as an integrated development environment for system development. Anaconda3 is selected as the interface. Anaconda integrates a large number of Python tool libraries, which is one of the common development environments for developers. In terms of data preprocessing, scipy.signal library is used, which contains common filter functions. In the selection of neural network framework, the system uses Keras to train the neural network model. Keras is a commonly used highly modular deep learning framework. The Scikit-Learn library in Python is used for machine learning, which integrates various machine learning algorithms, and is also very friendly in data preprocessing and classification results visualization.

4.2. Raw Data Acquisition and CSI Extraction. In order to further verify the stability of the gesture recognition system, the system also uses raw CSI gesture data from some public datasets on the basis of self-collected CSI gesture data. The experimental data source in this system design includes the gesture recognition data set adopted by the Zheng et al. team of Tsinghua University in WiDar3.0 in 2019, which is the original CSI gesture package in multiple scenarios [14]. By studying the original CSI data of gestures in different scenarios, it provides great help to study the mechanism behind gestures. At the same time, the WiAr data set released by Guo of Dalian University of Technology is also used in the system [15]. The data set includes 30 groups of reference cases of 16 different activities of 10 volunteers, including gestures such as waving, sliding, and throwing. These data provide a large number of reliable experimental data reference for the study of gesture data preprocessing and gesture feature extraction in this paper.

In the process of actual experiments, phase and amplitude information were used as the comparison for several times. These experimental results showed that the phase information of CSI quotient was identified with high recognition rate and stability in the real environments. Therefore, we decided to use CSI quotient phase information as the original data for identification of human gestures.

4.3. Data Preprocessing

4.3.1. Filtering and Denoising. The CSI data parsed from the original CSI packet contains a large amount of noise, which seriously interferes with the clarity of the CSI sequence waveform. Although CSI quotient model is adopted in this

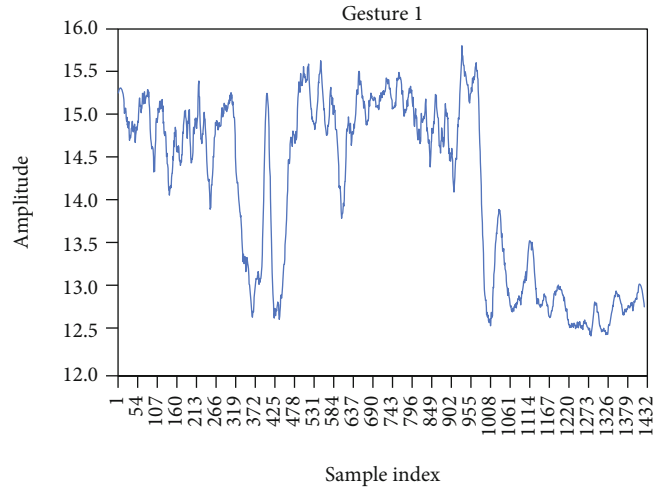


Figure 1: Amplitude sequence of gesture 1.

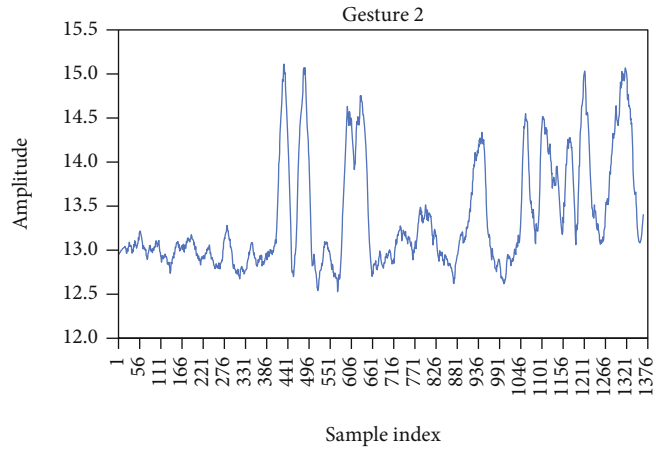


Figure 2: Amplitude sequence of gesture 2.

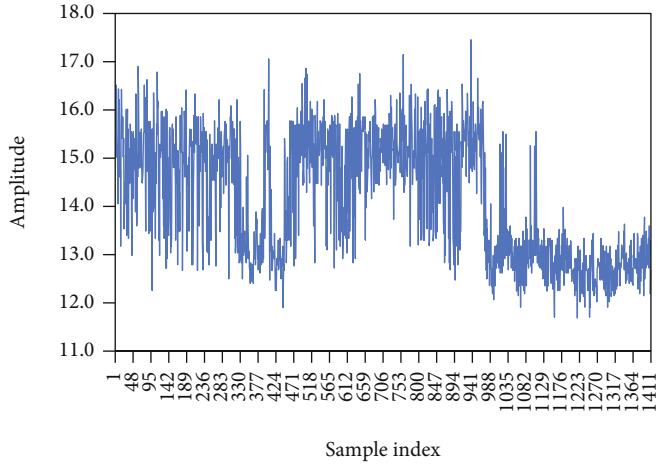


Figure 3: The first receiving antenna amplitude sequence.

system and a large number of noises have been filtered out, the amplitude waveform is not smooth enough. Therefore, the Savitzky-Golay filter is used to further smooth the waveform for subsequent feature extraction and classification.

Savitzky-Golay filtering is a filtering method based on local area polynomial least square fitting for time series signals. One of the characteristics of the filter is that it can ensure the basic stability of the signal width while filtering

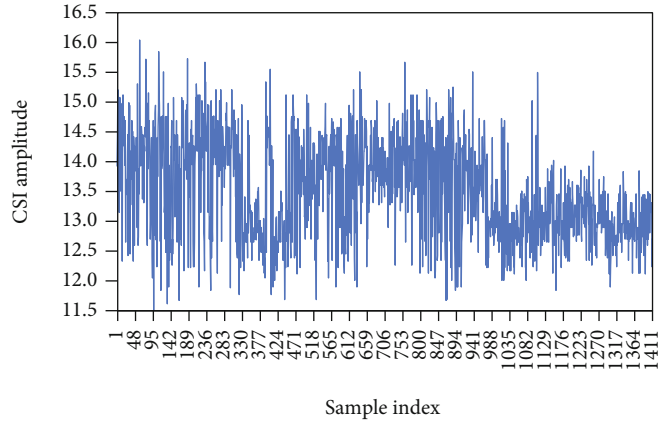


Figure 4: The second receiving antenna amplitude sequence.

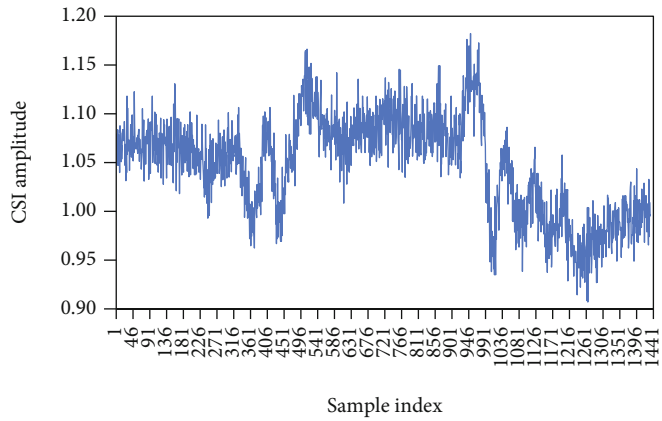


Figure 5: Amplitude sequence of CSI quotient.

out the high-frequency noise in the original signal. Savitzky-Golay method is used to smooth the CSI data, which can improve the smoothness of the CSI gesture data sequence and reduce the interference of environmental noise, so as to facilitate the subsequent gesture feature extraction and classification. When the typical peak of the signal is narrow, the filter has good filtering effect.

The method requires first setting a sliding window of size n and a fitting order k, where $n = 2m + 1$, to perform a left-to-right filtering of the curve before filtering with this window size. Suppose the original signal is $x[r]$, where $r = 0$, 1, -1, 2, -2… Firstly, the filtering center $x[0]$ is selected, and 2 $m + 1$ points of each m point around the center are selected as the primary filtering object. The polynomial fitting of $k - 1$ order is used for n data points in the selected window, as shown in the following formula.

$$y = \sum_{k=0}^{N} a_k n^k. \tag{5}$$

The fitting equation is expressed as a matrix, as the following.

$$Y = X \bullet A + \varepsilon. \tag{6}$$

The least square solution $\widehat{A}$ of A is

$$\widehat{A} = \left(X^T \bullet X\right)^{-1} \bullet X^T \bullet Y. \tag{7}$$

The final predicted value obtained $\widehat{Y}$ is

$$\widehat{Y} = X \bullet \widehat{A} = X \bullet \left(X^T \bullet X\right)^{-1} \bullet X^T \bullet Y. \tag{8}$$

The window slides from left to right on the original signal, and the data points are fitted sequentially until all data points are fitted to the end. The fitted CSI data are finally used for subsequent gesture feature extraction and classification recognition.

A comparison between the original CSI data amplitude sequence and the CSI amplitude sequence filtered by Savitzky-Golay filter is shown in Figures 6 and 7. Through image comparison, it can be obviously observed that most of the noise in the original CSI sequence is removed by two filtering, and most of the change information of the original CSI amplitude curve is retained. After filtering, the CSI amplitude sequence fluctuation caused by gesture can be clearly identified.

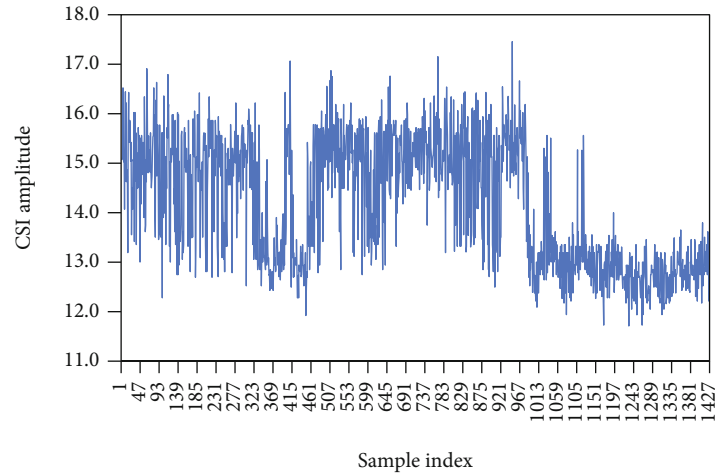


Figure 6: CSI amplitude sequence before denoising.

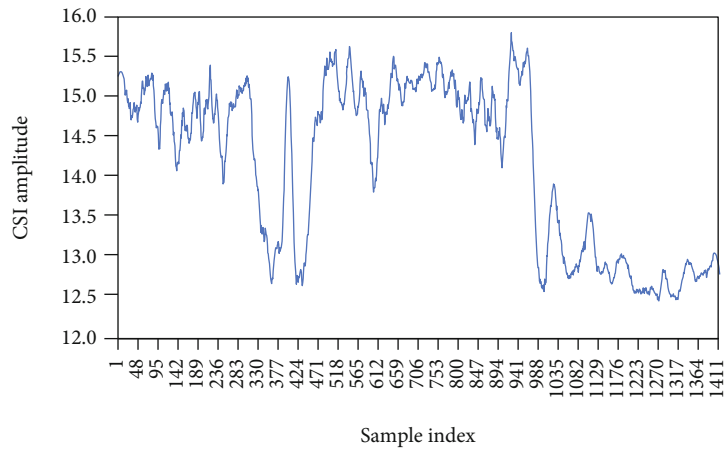


Figure 7: CSI amplitude sequence after denoising.

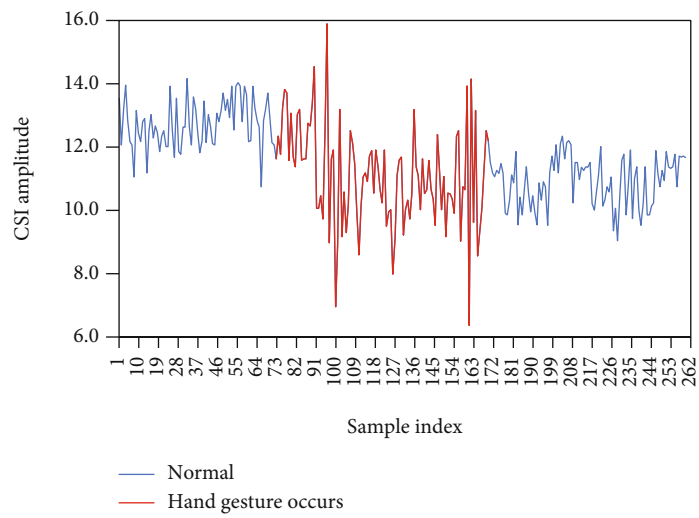


Figure 8: Gesture occurrence interval.

4.3.2. Segmentation of CSI Stream Data. As shown in Figure 8, the starting and ending points of gestures can be determined by detecting the fluctuation of this mutation. In this system, the sliding window variance comparison method is used to detect the occurrence of gesture activity, and the specific implementation process is as follows. Firstly,

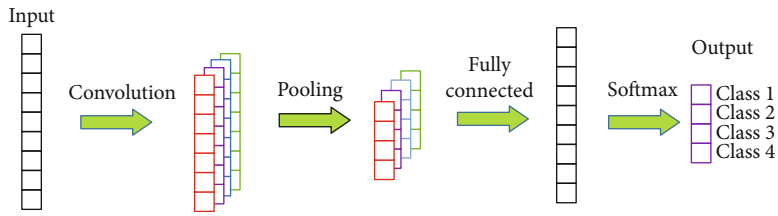


Figure 9: Structure of a one-dimensional CNN.

the window size is set as 10 CSI packets, and the sliding distance is set as 5 CSI packets. By calculating the variance of CSI amplitude in each window, a variance sequence can be obtained. By comparing the variance sequence, the position where the amplitude of CSI fluctuates greatly due to the influence of gesture can be detected. It can be used to roughly identify the beginning and end of each gesture from the data sequence. It is stipulated that when the variance of the latter window is three times that of the former window, gesture activity can be considered to occur between the two windows. The end of the former window is recorded as the occurrence point of the activity. Similarly, the end point of the activity can be found from the back forward.

4.4. Gesture Feature Extraction

4.4.1. 1D-CNN-Based Gesture Feature Extraction. Convolutional neural network (CNN) is a deep learning structure. It consists of multiple convolutional layers and pooling layers combined with each other and is capable of feature extraction and classification recognition of images, sounds, and texts [16].

Since CNNs are widely used in the field of image recognition, researchers usually prefer to convert CSI data into the form of images and use two-dimensional CNNs for image recognition to achieve the purpose of classification [17]. However, CSI values are a set of one-dimensional time series data that can be classified and recognized by a one-dimensional CNN. The one-dimensional CNN has a simpler structure and fewer parameter settings, so it can save computer resources and time, and is conducive to rapid recognition of gestures. The basic structure of the 1D-CNN is shown in Figure 9.

In the convolution layer, the neural network convolutions the segments of the input signal to generate the corresponding one-dimensional feature map. Various convolution kernels extract different features from the input signal, and each convolution kernel detects the specific features of all positions on the input feature map to achieve the weight distribution on the same input feature map. After the convolution layer, the number of feature maps and the dimension of data features increases greatly, which makes the next work difficult. Therefore, the pooling layer is used here to process the features, reduce the number of parameters, and extract the main features. Through multilayer convolution and pooling, the classification results are finally output through the dense layer, and the classification activation function is generally set as softmax.

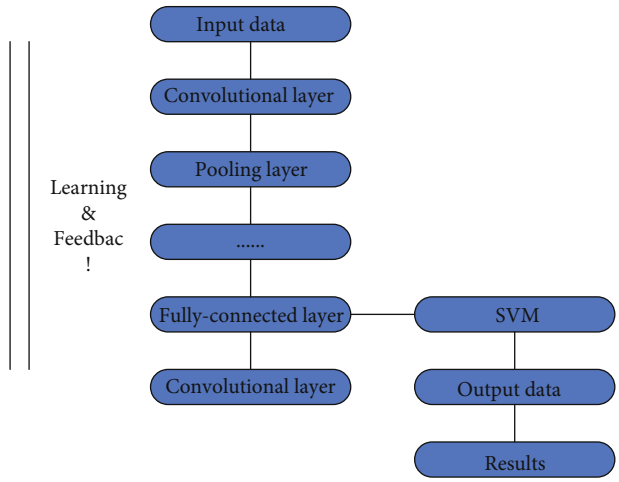


Figure 10: CNN-SVM model diagram.

4.4.2. 1DCNN-SVM Model. Based on the RCNN model proposed by Lei [18], this paper does not use the softmax function to output the results at the end of CNN, but only uses CNN as a feature extractor and inputs the extracted feature vectors into the SVM classifier for training and classification. Girshick and others from Harbin Institute of Technology applied the model to Chinese text recognition [19], and Liu et al. applied the CNN-SVM model to classification of liquids [20], both of which achieved good results. Based on the above research, this paper uses neural network as a feature extractor and SVM as a classifier. The CNN-SVM model is shown in Figure 10.

This paper is always faced with the problems of small amount of data and difficulty to selecting gesture features. 1DCNN-SVM model is undoubtedly very suitable to solve the above two problems. SVM classifier can solve the model overfitting problem caused by the small amount of data to some extent. CNN is a powerful feature extractor, and 1DCNN can automatically and effectively extract the features of CSI gestures [21]. A comparative discussion of the recognition rate of the 1DCNN-SVM model and the CNN model alone will be presented in chapter 4 below.

Through the above analysis, this paper finally adopts the hierarchical structure as shown in Table 1. Regardless of the additional layers, such as batch normalization and dropout layers, the network mainly consisted of five convolutional layers, three pooling layers, one flat layer, and two fully connected layers.

In this paper, the output of full connection layer of one-dimensional convolutional neural network is put into SVM

Table 1: Hierarchical structure of proposed 1D-CNN.

Number	Layer name	Function	Parameter setting
1	Convolutional layer	Extracting local features	Number of convolutional kernels: 16 Size of convolutional kernel: 3
2	Pooling layer	Spatially dimension reduction	Number of pooling kernels: 16 Size of pooling kernel: 3
3	Convolutional layer	Extracting local features	Number of convolutional kernels: 32 Size of convolutional kernel: 3
4	Pooling layer	Spatially dimension reduction	Number of pooling kernels: 16 Size of pooling kernel: 3
5~7	Convolutional layer	Extracting local features	Number of convolutional kernels: 64 Size of convolutional kernel: 3
8	Pooling layer	Spatially dimension reduction	Number of pooling kernels: 16 Size of pooling kernel: 3
9	Flat layer	Realizing the transition from convolution layer to full connection layer	
10	Fully connected layer		Output size: 64
11	Fully connected layer	Classification and recognition using SVM classifier	Output size: 3

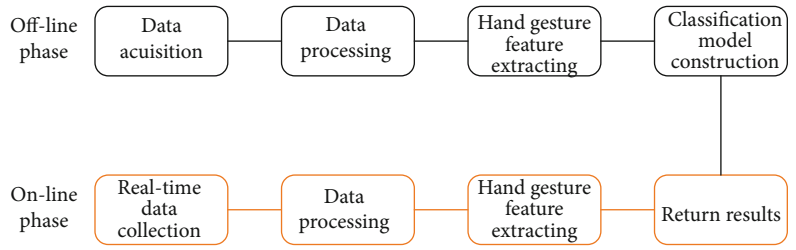


Figure 11: Real-time processing flow chart.

classifier as feature vector to classify and recognize. Therefore, the number of neurons in the full connection layer in the model is the characteristic number of SVM training. By comparing the system recognition accuracy and training time under different feature number, the most suitable value of feature number is 64.

4.5. Design of the Real-Time Recognition System. A set of real-time acquisition system is designed for the data collected by the self-collected Atheros network card. The real-time system is divided into the offline stage and the online stage. In the off-line stage, the basic process of data acquisition and extraction, data preprocessing, and feature extraction process classification and recognition are completed, and the classification model is trained to detect and classify the collected data. The flow chart is shown in Figure 11.

To meet the requirements of data volume and real-time, an unconventional gesture acquisition method is used in this project for the actual data acquisition process. In the process of gesture acquisition, the single gesture activity is no longer stored separately, but the same gesture activity is repeated many times in a fixed time. The same gesture activity is collected in the same CSI original packet. The gestures are sliced, and the models are built using these fragments. In the process of detection, the gestures collected on the scene are sliced with the same size. Then, each fragment is classified, and the classification results are determined to return the final recognition results.

5. Experimental Result and Analysis

5.1. Platform and Data Extraction. In this design, gesture data samples are collected in two indoor environments. The first indoor environment is the home environment outside the school, and 20 gesture samples of hand waving, push forward, and stroke are collected. The second indoor acquisition environment is the school dormitory. Here, 20 gesture samples of hand waving, push forward, and left stroke in four different directions of the human body are collected for comparative experiments.

Two TP-Link TL-WDR4310 Wi-Fi routers were used as signal transceiver devices for data acquisition. The routers were equipped with Atheros AR9344 network cards, and a computer was used as the console to control the two routers for interactive communication for the acquisition of the original CSI data. The two routers are located at the same height, 1.5 m apart, and the collector is located on the vertical line of the two routers, 0.5 m apart from the transceiver equipment. Human face toward two routers for gesture data collection.

In this paper, the data collected by two kinds of network cards are tested simultaneously. The CSI data of Atheros network card are collected by ourselves, and the data collected by Intel5300AGN network card are from the WiAr data set published by Guo, Dalian University of Technology [22–24]. The data of real-time system is collected by Atheros network card.

5.2. Analysis of Experimental Results

5.2.1. Classification Results Based on CSI Data. In order to improve the efficiency of data acquisition and data volume, the data collected by the Atheros network card is classified by window sliding slice. The window size is set to 2 seconds; sliding step size is set to 0.2 seconds. Figure 12 shows the change curves of accuracy and loss during the training process, and it can be seen that the model accuracy and loss tend to smooth out after 10 iterations. Finally, the recognition rate of three different hand gestures, pushing forward, waving, and left stroke, can reach up to 92% or more. After 50 times of random division of the data set, the comprehensive accuracy of the recognition results is between 85% and 92%.

The recognition rate for each gesture is shown in Figure 13, where the recognition rate is 90.1% for the left stroke, 88.6% for the waving motion, and 97.6% for the forward pushing motion. Among the three different gestures, the recognition rate of forward pushing activity is the highest, and the recognition rate of left stroke and waving activity is relatively lower. Left strokes and hand wavings are both right-to-left in physical space, crossing the Fresnel zone similarly and having similar effects on CSI. Therefore, it is easy to confuse the recognition, and the recognition rate is lower than that of the former.

5.2.2. Classification Results. Due to the limited type and quantity of gesture data collected by the Atheros network card, it is difficult to fully detect the stability of the system. Therefore, the system readapts the CSI data of Intel 5300 network card. The gesture data in WiAr dataset are classified as a supplement to the correctness of the detection system [25–27]. Figure 14 shows the curves of accuracy and loss during the training process, and it can be seen that the model accuracy as well as the loss tends to be smooth after 10 iterations. The final combined accuracy for the recognition of the four different gestures can reach up to 94% or more. After 50 random divisions of the data set, the combined accuracy is between 85% and 94%.

5.3. Experimental Analysis

5.3.1. The Influence of Different Human Orientations on the Results. In order to explore the influence of different human orientations on the accuracy of gesture recognition, a comparative experiment is set up in this paper. Three different gesture data are collected in four different directions of the human body. The final recognition results for the three gestures are shown in Figure 15. Direction 1 is the direction facing the transmitter and the receiver connection, direction 2 facing the receiver, direction 3 back to the transmitter and the receiver connection, and direction 4 back to the receiver. From the results, we can see that the recognition rate is lower when on the direction that is facing the transmitter and the receiver connection, and the recognition results of the other three directions are basically similar [28–31].

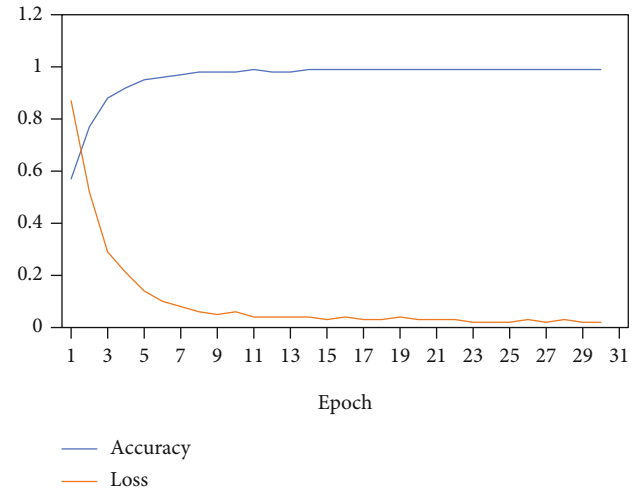


Figure 12: Model training results based on CSI data of the Atheros Network Interface Card.

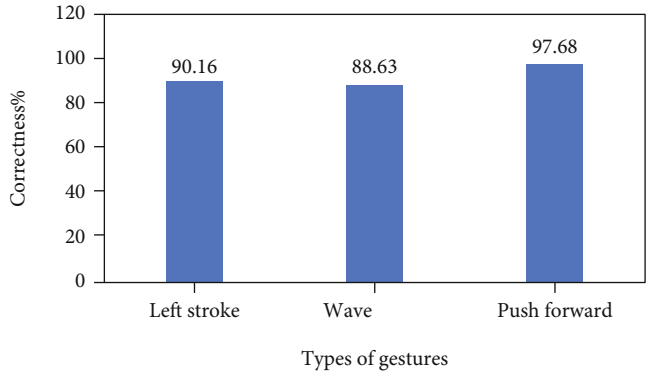


Figure 13: The results of recognition rate of different gestures.

5.3.2. Comparison of Experimental Results of 1DCNN-SVM Model. At the same time, the experiment also compared the performance of separate CNN and CNN-SVM under the same gesture data set [32]. After 50 comparative experiments, the accuracy of CNN and CNN-SVM is shown in Figure 16. It can be seen in 50 comparative experiments that CNN-SVM model is better than the softmax function-activated CNN model in 43 times, accounting for 86%. Among them, the average recognition rate of CNN model using softmax activation classification is 89.8%, and the average recognition rate of CNN-SVM can reach 91.4%. It can be seen that CNN-SVM is suitable for CSI gesture data [33], which can slightly improve the recognition rate of gesture recognition system [34].

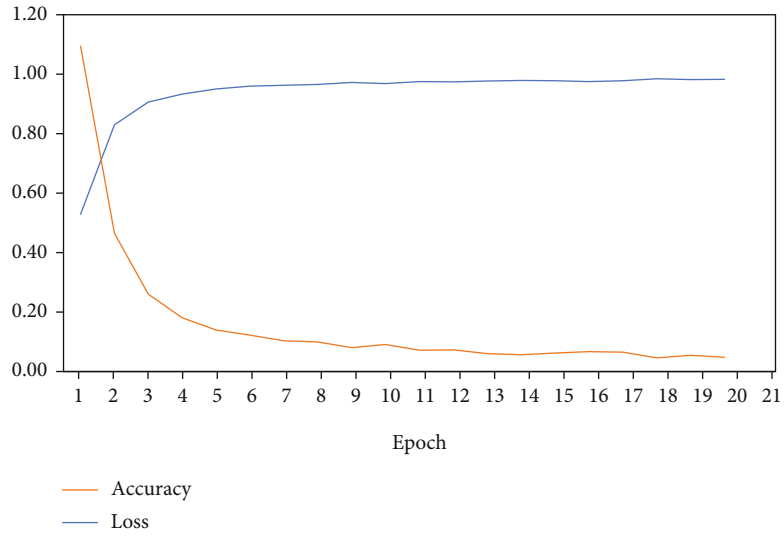


Figure 14: Model training results based on CSI data of Intel 5300 Network Interface Card.

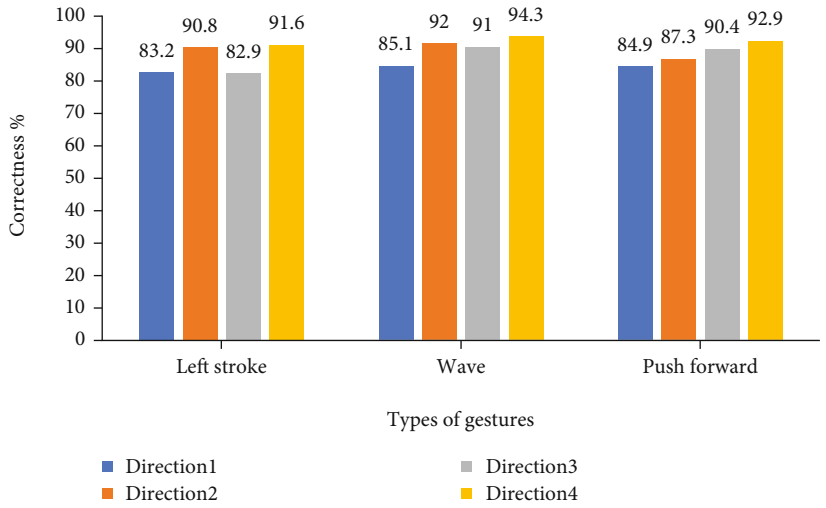


Figure 15: Recognition results under different orientations.

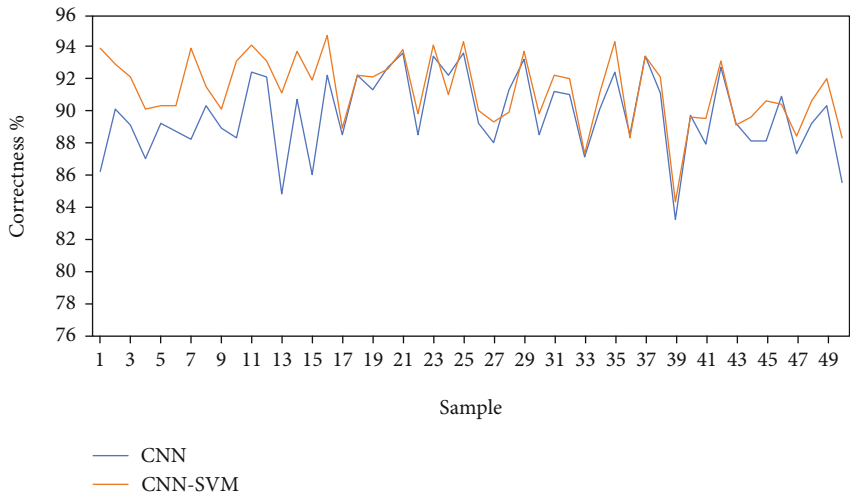


Figure 16: Comparison of the recognition rate.

6. Conclusion

Gesture recognition technology based on the CSI of Wi-Fi device is an emerging gesture recognition scheme. Compared with the traditional gesture recognition technologies based on video images and sensors, it has unique advantages of high privacy and convenient deployment. As a result, the scheme has great potential for use in smart homes as well as smart healthcare. In this paper, a gesture recognition system based on CSI is designed based on Wi-Fi equipment. The system adapts the CSI data collected on two different Network Interface Cards and also compares the correctness of system recognition when the human body makes gestures in different orientations. The Savitzky-Golay filter is used to reduce the noise and smooth the curve of CSI sequence, and the sliding window is used to separate the process of static and motion from CSI, so as to extract the segment of gesture activity. Then, we build appropriate 1D-CNN model to achieve gesture feature extraction. Furthermore, we use the SVM classifier to recognize the gestures and compare the effect of different kernel functions on the gesture recognition results, then select the appropriate kernel function.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

Acknowledgments

This research was funded by the National Natural Science Foundation of China, grant number 61802196; Jiangsu Provincial Government Scholarship for Studying Abroad; the Priority Academic Program Development of Jiangsu; the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD); and NUIST Students' Platform for Innovation and Entrepreneurship Training Program, grant number 202010300080Y.

References

[1] W. Zhuang, Y. Shen, L. Li, C. Gao, and D. Dai, "Develop an adaptive real-time indoor intrusion detection system based on empirical analysis of OFDM subcarriers," *Sensors*, vol. 21, no. 7, p. 2287, 2021.

[2] J. Su, Z. Sheng, A. Liu, Z. Fu, and Y. Chen, "A time and energy saving based frame adjustment strategy (TES-FAS) tag identification algorithm for UHF RFID systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 2974–2986, 2020.

[3] J. Su, Z. Sheng, A. X. Liu, Y. Han, and Y. Chen, "Capture-aware identification of mobile RFID tags with unreliable channels," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 1–14, 2020.

[4] J. Su, Z. Sheng, V. C. M. Leung, and Y. Chen, "Energy efficient tag identification algorithms for RFID: survey, motivation and new design," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 118–124, 2019.

[5] H. Yang, Y. Shen, W. Zhuang, C. Gao, D. Dai, and W. Zhang, "A smart wearable ring device for sensing hand tremor of Parkinson's patients," *Computer Modeling in Engineering and Sciences*, vol. 126, no. 3, pp. 1217–1238, 2021.

[6] H. Abdelnasser, K. A. Harras, and M. Youssef, "WiGest demo: an ubiquitous WiFi-based gesture recognition system," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1472–1480, Hong Kong, Hong Kong, 2015.

[7] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, p. 53, 2011.

[8] N. Yu, W. Wang, A. X. Liu, and L. Kong, "QGesture," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–23, 2018.

[9] W. Jiang, C. Miao, F. Ma et al., "Towards environment independent device free human activity recognition," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking Mobi Com*, pp. 289–304, New York: ACM, 2018.

[10] Y. Zheng, Y. Zhang, K. Qian et al., "Zero-effort cross-domain gesture recognition with Wi-Fi," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 313–325, New York: ACM, 2019.

[11] Y. Zhang, Y. Zheng, G. Zhang, K. Qian, C. Qian, and Z. Yang, "GaitID: robust Wi-Fi based gait recognition," *Wireless Algorithms, Systems, and Applications: 15th International Conference, WASA 2020, Qingdao, China, September 13–15, 2020, Proceedings, Part I*, , pp. 730–742, Springer Nature, 2020.

[12] W. Xi, H. Dong, K. Zhao, Y. Yan, and D. Chen, "Device-free human activity recognition using CSI," in *In Proceedings of the 1st Workshop on Context Sensing and Activity Recognition CSAR*, pp. 31–36, new York: ACM, 2015.

[13] Y. Zeng, D. Wu, J. Xiong, E. Yi, R. Gao, and D. Zhang, "FarSense," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, pp. 1–26, 2019.

[14] H. Ge, Z. Yan, W. Yu, and L. Sun, "An attention mechanism based convolutional LSTM network for video action recognition," *Multimedia Tools and Applications*, vol. 78, no. 14, pp. 20533–20556, 2019.

[15] L. Guo, L. Wang, J. Liu, and W. Zhou, "A survey on motion detection using Wi-Fi signals," in *12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pp. 202–206, Hefei, China, 2016.

[16] W. Zhuang, Y. Shen, L. Li, C. Gao, and D. Dai, "A brain-computer interface system for smart home control based on single trial motor imagery EEG," *International Journal of Sensor Networks*, vol. 34, no. 4, p. 214, 2020.

[17] L. Guo, L. Wang, J. Liu, W. Zhou, and B. Lu, "HuAc: human activity recognition using crowdsourced Wi-Fi signals and skeleton data," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 6163475, 2018.

[18] L. P. Lei, "Curve smoothing denoising based on Savitzky-Golay algorithm," *Computer and Information Technology*, vol. 22, no. 5, pp. 30-31, 2014.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic

segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Columbus, OH, USA, 2014.

[20] L. Lu, P. Yang, S. Weiwei, and J. Ma, "Similar handwritten Chinese character recognition based on CNN-SVM," in *In Proceedings of the International Conference on Graphics and Signal Processing*, pp. 16–20, new York: ACM, 2017.

[21] L. Hu, *Research on Liquid Classification Detection of WI-FI Channel State Information Based on CNN-SVM*, Hunan University, Changsha, 2018.

[22] W. He, *Research on WI-FI-Based Gesture Recognition Technology*, Shenzhen University, Shenzhen, 2015.

[23] T. Zhang, *Research on Hand Gesture Recognition Based on WI-FI Channel State Information*, University of Electronic Science and Technology, Chengdu, 2019.

[24] Y. Lu, S. Lv, X. Wang, and X. M. Zhou, "A review of research on human behavior sensing technology based on Wi-Fi signal," *Journal of Computer Science*, vol. 2, pp. 1–21, 2019.

[25] L. Jiahui, W. Yujie, and L. Yi, "LSTM-based gesture recognition method for CSI," *Computer Science*, vol. 46, Supplement 2, pp. 283–288, 2019.

[26] X. Pan, *Research on the Key Technology of Dynamic Gesture Recognition Based on 1D-CNN in Wi-Fi Environment*, Beijing University of Posts and Telecommunications, Beijing, 2019.

[27] W. Zhuang, Y. Shen, J. Zhang, C. Gao, Y. Chen, and D. Dai, "A contactless sensing system for indoor fall recognition based on channel state information," *International Journal of Sensor Networks*, vol. 34, no. 3, p. 188, 2020.

[28] G. Yang, *Research on Wi-Fi-Based Hand Action Recognition Algorithm*, Northwestern University, Xi'an, 2019.

[29] K. Niu, Z. Fusang, D. Wu, and D. Q. Zhang, "Exploring the stability of Wi-Fi-aware system with Fresnel zone model," *Computer Science and Exploration*, vol. 15, no. 1, pp. 60–72, 2021.

[30] K. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, 2017.

[31] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: indoor localization via channel response," *ACM Computing Surveys*, vol. 46, no. 2, p. 25, 2013.

[32] F. Youbing, Y. Lu, and Z. Weibo, "Design and implementation of face recognition system based on CNN and SVM," *Computer and Digital Engineering*, vol. 49, no. 2, pp. 378–420, 2021.

[33] Y. Di, *Research on Location-Unknown Gesture Recognition Method Based on Wi-Fi Signal*, Henan University, Kaifeng, 2020.

[34] X. Li, D. Q. Zhang, Q. Lv et al., "IndoTrack," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–22, 2017.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9959304, 21 pages
https://doi.org/10.1155/2021/9959304

Research Article

Cooperative Edge Computing Task Offloading Strategy for Urban Internet of Things

Bo Wang[1,2,3] **and Mingchu Li**[1,3]

[1]*School of Software Technology, Dalian University of Technology, Dalian 116620, China*
[2]*School of Applied Technology, University of Science and Technology Liaoning, Anshan 114051, China*
[3]*Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116620, China*

Correspondence should be addressed to Bo Wang; wangboustl@126.com

Received 30 March 2021; Revised 30 August 2021; Accepted 4 September 2021; Published 7 October 2021

Academic Editor: Ning Wang

With the continuous progress of edge computing technology and the development of the Internet of Things technology, scenarios such as smart transportation, smart home, and smart medical care enable people to enjoy the smart era's convenience. Simultaneously, with the addition of many smart devices, a large number of tasks are submitted to the edge server, making the edge server unable to meet the needs of completing tasks submitted by the smart device. Besides, if the task is submitted to the remote cloud data center, it increases the user's additional delay and cost. Therefore, it is necessary to improve the task offloading strategy and resource allocation scheme to solve these problems. This paper first proposes a new task offloading mechanism and then proposes a two-stage Stackelberg game model to solve each participant's interaction problem in the task offloading mechanism and ensure the maximization of their respective interests. Finally, a theoretical analysis proves the equilibrium of the two-stage Stackelberg game. Experiments are used to prove the effectiveness of the proposed mechanism. Comparative experimental results show that the proposed model can achieve better results regarding delay and energy consumption.

1. Introduction

The urban Internet of things (IoT) plays a vital role in our daily life. It realizes smart city, urban brain, and other applications by processing the data generated by the intelligent devices deployed in the city. In the urban IoT scenario, the traditional processing method is cloud computing, because it can provide rich computing or storage resources to process much urban IoT data. However, when the task is submitted from an urban IoT device to a cloud computing data center, a large amount of data transmission seriously affects the processing performance, resulting in network congestion or high latency. Therefore, edge computing technology is used to offload tasks to edge servers or IoT devices. It is closer to end-users, to improve the system's performance in terms of service delay, QoS, and resource utilization. However, with the proliferation of smart devices in recent years, massive amounts of data are transmitted to edge servers or IoT devices. These bring a heavy burden to the communication bandwidth because the resources of the edge server or the IoT devices are limited. It inevitably leads to the phenomenon that multiple tasks compete to use the limited resources when providing services. A large number of resource contentions in edge servers undoubtedly lead to unbearable waiting delays, energy consumption, and degradation of end-user service quality. There are also many other edge servers or IoT devices in the urban IoT that may be idle or unused, which leads to the waste of resources. In addition, end-users obtain related services through payment. To obtain lower latency and higher satisfaction, users are willing to pay more, which inevitably causes users to spend more. Therefore, how can we offload the overloaded data to suitable processing facilities and provide high-quality services? How can we achieve low latency and high quality of user experience (QoE) for users with less expenditure and energy consumption? These have become an important issue.

In response to the above challenges, our goal is to design a method to utilize the unused or idle resources in the urban IoT. The task is offloaded to other servers or IoT devices to improve the efficiency of edge computing. In this paper, we call this pattern as "*cooperative edge computing* (CEC)." There are many types of research on collaborative edge computing. Ren et al. [1] propose to offload tasks to remote cloud computing centers for collaborative execution; it can use the system resources more effectively. Chen et al. [2] propose to offload tasks to multiple edge servers for collaborative execution. This method improves the performance of edge service providers. In [3], Device-to-Device (D2D) is utilized to offload tasks to other smart devices for collaborative execution. Various device resources are shared with other users through high-quality cellular connections. It can provide more and better services to attract more users. Jie et al. [4] propose an optimal resource allocation scheme for the IoT environment based on fog computing. To maximize resource utilization, the authors model the resource allocation problem as a two-stage Stackelberg game and propose three algorithms to achieve the Nash equilibrium and Stackelberg equilibrium. Although these methods all adopt the collaborative approach, they are fundamentally different from the model proposed in this paper. End-users perform the collaboration methods above by offloading tasks to remote cloud servers or other edge servers or IoT devices. The model proposed in this paper uses the edge scheduler to offload the overloaded tasks on the edge server to other idle edge servers or IoT devices for execution. The edge scheduler is also responsible for executing CEC task offloading strategies between other edge servers and IoT devices.

In the scenario with collaborative service providers, users hope to obtain more resources within their budget. Local edge service providers hope to obtain more revenue by providing services to more users. Collaborative service providers can get rewards by providing resources to local edge service providers. Users, local edge service providers, and collaborative service providers all pursue their interests. It is necessary to establish a feasible incentive mechanism to maximize interests for all participants and promote cooperation. Game theory is an effective tool to solve this problem.

This paper proposes a cooperative task offloading and resource allocation mechanism between edge servers and other edge servers or IoT devices. The proposed mechanism uses a two-stage Stackelberg game [5, 6] to solve matching tasks and resources in multiple rounds. Users give corresponding payments when the service quality meets their requirements. To increase the utilization rate of resources to save energy consumption and reduce costs, edge service providers match tasks with available resources in specific strategies according to users. According to particular strategies, the collaborative service provider matches the tasks submitted by the local edge service provider with the available resources. The two-stage game is based on the opponent's possible strategy, choosing the strategy to ensure that interests are maximized under its strategies to achieve the game equilibrium. The experimental results show that under the premise of ensuring high user satisfaction and not exceeding the user total budget, our proposed mechanism can obtain a near-optimal offloading strategy and is superior to traditional solutions in time delay and energy consumption.

The main contributions of this paper are as below.

(1) We establish delay, user satisfaction, cost, and energy consumption models and define formal task offloading and resource management problems in cooperative edge computing

(2) We propose a two-stage Stackelberg game model to solve the participant interactive problem in the task offloading mechanism and ensure the maximization of their respective interests

(3) We conduct a theoretical analysis of the game equilibrium of the two-stage Stackelberg game. Experiments are used to prove the effectiveness of the proposed mechanism

The remainder of this paper is organized as follows. In Section 2, we briefly review related works. In Section 3, we describe the system model and problem formulation. In Section 4, we propose a two-stage Stackelberg game model and analyze the model's game equilibrium. Section 5 shows the simulation results, and the conclusion is presented in Section 6.

2. Related Work

In recent years, researchers have made many studies in task offloading strategies and resource allocation for edge computing. Edge computing not only enriches cloud computing but also brings many challenges. Research on CEC has attracted wide attention.

Many studies deal with end-user offloading tasks collaboratively with multiple edge servers or base stations. Chen et al. [2] propose to offload tasks to multiple edge servers for collaborative execution. This method improves the performance of edge service providers. Fan and Ansari [7] design an application workload allocation scheme for the IoT based on edge computing. By determining the target cloudlet for each request of different IoT users, the amount of computing resources is allocated to each IoT user; IoT application requests' response time is minimized. Niu et al. [8] propose a workload allocation mechanism for the power IoT based on edge computing to minimize service delays. A workload optimization allocation model is established. Based on the optimization of computing resources in a single edge node, the optimal workload allocation between multiple edge nodes based on the delay is further realized. Hao et al. [9] propose a Smart-Edge-CoCaCo algorithm. To minimize the total delay and confirm the computational offloading decision, Smart-Edge-CoCaCo utilized the wireless communication model, the collaborative filter cache model in the edge cloud, and the joint optimization of the computational offloading model. Parwez and Rawat [10] propose a resource allocation method in an adaptive virtual wireless network with mobile edge computing. According to the demand of users, the authors study how to allocate mobile

virtual network operator resources to maximize revenue with satisfied service quality. Cooperative resource allocation is proposed to maximize the utilization ratio of mobile virtual network operators with desired QoS of user network speed. Zhang et al. [11] propose a collaborative task offloading scheme that considers the reuse of calculation results and minimizes energy consumption.

Many studies consider multiple influencing factors and handle tasks offloaded by end-users in a cloud-edge collaborative manner. Ren et al. [1] study the collaboration between cloud computing and edge computing, where tasks can be partially processed on edge nodes and cloud servers. The authors further transformed the communication and computing resource allocation problem into an equivalent convex optimization problem and obtained a closed-form resource allocation strategy. Jie et al. [4] propose an optimal resource allocation scheme for the IoT environment based on fog computing. To maximize resource utilization, the authors model the resource allocation problem as a two-stage Stackelberg game and propose three algorithms to achieve the Nash equilibrium and Stackelberg equilibrium. Guo et al. [12] propose a delay-based workload distribution scheme. It realizes the optimal workload distribution among local edge servers, neighboring edge servers, and remote clouds to achieve the minimum energy consumption of the IoT-edge-cloud system and a delay guarantee for arriving jobs. For the problem of cloud-mobile edge computing collaborative computing offloading, Guo and Liu [13] propose an approximate collaborative computing offloading scheme and a game theory cooperative computing offloading scheme as approximate solutions. Deng et al. [14] introduce an intermediate fog layer between mobile users and the cloud. An optimal workload distribution method between fog and cloud is proposed to solve the trade-off between power consumption and transmission delay in the fog-cloud computing system. The service delay is decreased with minimum power consumption. To minimize the average response time, mobile users offload their application workloads to geographically dispersed cloudlets. He et al. [15] provide a collaborative computing offloading example based on energy consumption, computing power, variable transmission power, and remaining battery power. To handle delay-sensitive tasks effectively, they designed an iterative search algorithm for a collaborative computing offloading scheme to minimize task offloading overhead. To minimize energy consumption under the conditions of ensuring service completion time, Liu et al. [16] propose an energy-saving collaborative task computing offloading algorithm based on semideterministic relaxation and random mapping methods. Wu et al. [17] propose an edge-cloud collaborative multitask computing offloading model. The latency and energy costs are considered. By converting the model solution to a search solution in limited strategy space, it is solved by a nonlinear exponential inertia weighted particle swarm optimization algorithm. By dynamically adjusting the inertia weight, the premature convergence defect of the standard particle swarm algorithm can be compensated. The optimal local solution can be effectively avoided. Li et al. [18] propose a computing offloading mechanism based on the Stackelberg game to analyze the interaction between multiple edge clouds and numerous industrial IoT devices. The payment cost is considered. The author also formulates the revenue function by considering the social interaction information from potential industrial IoT devices.

Some scholars use cloud-edge-end collaboration to handle tasks offloaded from end-users. Hossain et al. [19] propose a collaboration model. It can dynamically offload computing tasks' execution to the SBS-MEC server and mobile devices or remote cloud. In [20], one task can offload subtasks. The subtask can be offloaded according to the characteristics of the edge server (such as transmission distance and central processing unit capacity). The authors propose a low-complexity adaptive offloading scheme based on the Hungarian algorithm using a multi-subtask multiserver model for new applications that require real-time information work.

Some scholars use D2D collaboration to implement collaborative edge computing. Ciobanu et al. [21] propose a computational offloading solution that can improve user QoE, reduce application and service developers' cost, and reduce battery consumption. In [3], the D2D is utilized to offload tasks to other smart devices for collaborative execution. Various device resources are shared with other users through high-quality cellular connections. It can provide more and better services to attract more users.

Some scholars implement collaborative edge computing in an edge-to-end cooperative manner. He et al. [22] propose an effective method to find the best solution to optimize the system allocation time, transmit power and CPU on each device, and maximize the amount of data that two users can process in a given time frame. Kim et al. [23] propose a new concept of IoT-assisted edge computing, which provides edge services by integrating idle resources in IoT devices and offloading tasks to nearby IoT devices.

Although these methods all adopt the collaborative approach, they are fundamentally different from the model proposed in this paper. End-users perform the collaboration methods above by offloading tasks to remote cloud servers or other edge servers or IoT devices. The model proposed in this paper uses the edge scheduler to offload the overloaded tasks on the edge server to other idle edge servers or IoT devices for execution. The edge scheduler is also responsible for executing CEC task offloading strategies between other edge servers and IoT devices. In addition, due to the user time tolerance and the user total budget constraints, relevant research does not consider the impact of user costs and user satisfaction. Therefore, the main research content of this paper is to integrate idle trusted devices within a short distance from the end-user, assist the local edge service provider to complete the end-user offloading tasks, and jointly consider the impact of the end-user cost and QoE factors on the offloading strategy from the user view.

3. System Model and Problem Formulation

As shown in Figure 1, the entire system consists of end-users, edge service providers, edge scheduler, and

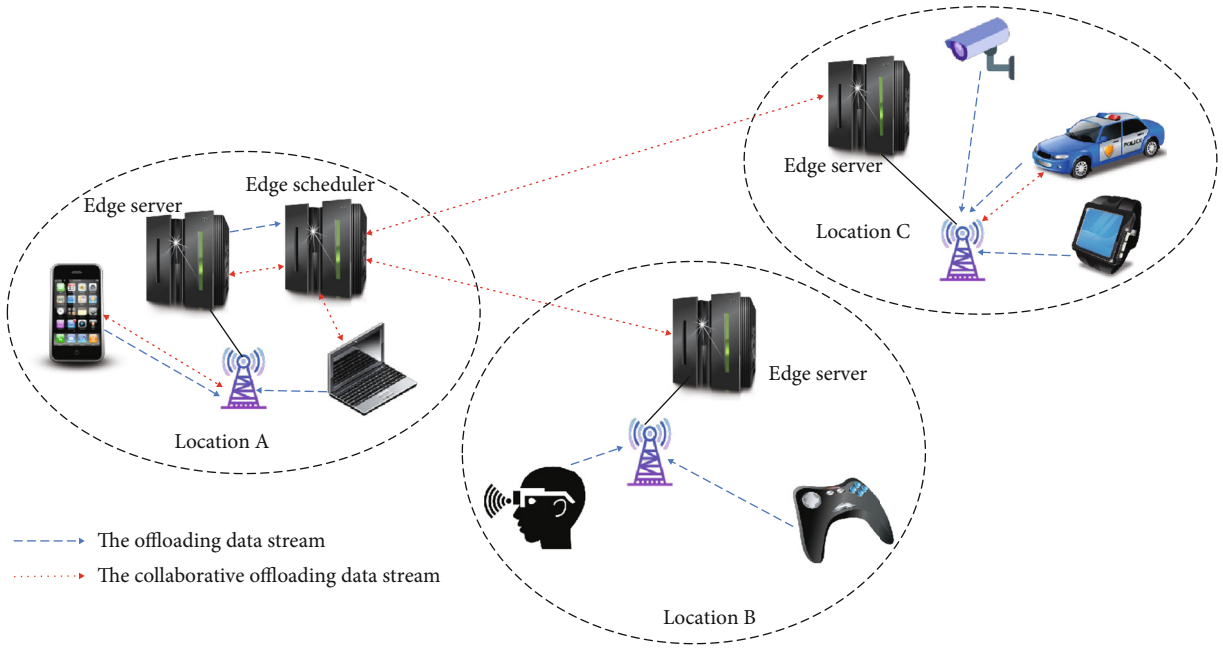


Figure 1: Multiedge resource cooperative offloading mechanism system structure diagram.

collaborative service providers. Smart devices include mobile smart terminals, IoT devices, notebooks, smart medical devices, and the Internet of Vehicles. Here, $EU = \{eu_1, eu_2, \cdots, eu_N\}$ represents the set of all devices in the smart device, where N represents the number of devices. The edge service provider is composed of base stations with edge servers distributed across different locations. The edge service provider can handle tasks submitted by related smart devices in their area and tasks migrated by other edge devices if resources are available. The set $ESP = \{esp_1, esp_2, \cdots, esp_M\}$ represents the collection of all edge servers in the edge service provider, where M represents the number of edge servers in the edge service provider. The set $R = \{R_1, R_2, R_3, \cdots, R_N\}$ is the set of resources required by all smart device users. The collaborative service provider includes other edge servers in the edge service provider and other smart devices with idle resources. The collaborative service provider can help the local edge device complete the smart device tasks as needed. For example, for the edge server at location A in Figure 1, when the resources of the edge server cannot meet user requirements, the task can be offloaded to the edge server at location B or C where the resources are available for execution. We call the edge server at location A local edge server and the edge server at location B or C collaborative edge servers. Besides, at position A in Figure 1, the smartphone's task can also be offloaded to the laptop for execution while waiting in the waiting queue of the local edge device. We call the laptop at this time as a collaborative smart device.

In the urban IoT scenario, the edge scheduler runs the task offloading decision mechanism to allocate task offloading strategies. Tasks are reasonably scheduled to different edge servers or edge devices by the edge scheduler, which can realize the collaborative processing of tasks between edge servers and edge devices. In this paper, cloud data centers are generally far from edge servers and devices. The offloading task to remote cloud data centers takes a long time and consumes more energy. Therefore, we do not consider the implementation of task offloading to remote cloud data centers. The mathematical symbols are summarized in Table 1.

3.1. Delay Model. As shown in Figure 1, the end-user offloads the task to the service device of the edge service provider. The end-user offloads the task to the base station wirelessly, and then, the base station transmits the task to the edge server through the optical fiber. This part of the delay can be expressed by

$$T_o = \sum_{i=1}^{\Gamma} \left(\frac{B_i^o}{C_i^o} + \frac{B_i^o}{F_i^o} \right). \tag{1}$$

Here, B_i^o represents the size of the offloading task i, C_i^o represents the transmission rate of the uploading in the wireless method, and F_i^o represents the transmission rate of the uploading using the fiber method. Γ represents the number of offloading tasks.

When a large number of tasks are waiting on the local edge server, the edge scheduler offloads tasks to other collaborative edge servers and other collaborative smart devices to reduce the waiting time and increase the completion rate of tasks.

When the local edge service provider has no resources to use, some tasks can be offloaded to the server of the cooperative edge service provider for execution. In this case, the delay of this part includes task transmission time $T_{i,ne}^o$, task execution time T_i^{ne}, and return time after task completion $T_{i,ne}^d$. Since the parallel mode is adopted, the delay is

Table 1: Symbol summary.

Symbol	Definition
T	Total delay
E	Total energy consumption
C^{co}	The cost of end-user
p_i^c	The price of CPU for the edge server
p_i^s	The price of storage for the edge server
p_i^m	The price of memory for the edge server
C^{esp}	The cost of edge service provider
p_i^{ne}	The price of other collaborative edge servers executing various types of resources
p_i^{ns}	The price of other collaborative smart devices executing various types of resources
Q_i	The satisfaction score of user i
bid_i	The bidding price of the smart device
p_i	The edge service provider's resource price

calculated according to the longest time required to complete the offloading task:

$$T^{ne} = \sum_{i=1}^{\kappa}\left(T_{i,ne}^{o} + T_i^{ne} + T_{i,ne}^{d}\right) = \sum_{i=1}^{\kappa}\left(\frac{B_{i,ne}^{o}}{C_o^{ne}} + \frac{B_{i,ne}^{o}}{\text{Cap}_{ne}} + \frac{B_{i,ne}^{d}}{C_d^{ne}}\right). \tag{2}$$

Here, κ represents the number of tasks that are offloaded to the server of the collaboration service provider for execution. $B_{i,ne}^{o}$ represents the size of the task i that is offloaded to the neighboring collaborative edge server, and C_o^{ne} represents the local edge server's transmission rate to the collaborative edge server. Cap^{ne} represents other collaborative edge devices' computing power. $B_{i,ne}^{d}$ represents the size of the backhaul data of the offloading task i from the neighboring collaborative edge server, and C_d^{ne} represents the backhaul data transmission rate from the collaborative edge server to the local edge server.

When the local edge service provider has no resources to use, some tasks can be offloaded to the edge device of the collaborative edge service provider for execution. In this case, this part of the delay includes task transmission time $T_{i,ns}^{o}$, task execution time T_i^{ns}, and return time after the task is completed $T_{i,ns}^{d}$. Since it is carried out in parallel, the delay is calculated according to the longest time required to complete the offloading task:

$$T^{ns} = \sum_{i=1}^{l}\left(T_{i,ns}^{o} + T_i^{ns} + T_{i,ns}^{d}\right) = \sum_{i=1}^{l}\left(\frac{B_{i,ns}^{o}}{C_o^{ns}} + \frac{B_{i,ns}^{o}}{\text{Cap}^{ns}} + \frac{B_{i,ns}^{d}}{C_d^{ns}}\right). \tag{3}$$

Here, l represents the number of tasks performed offloaded to the smart device of the collaboration service provider. $B_{i,ns}^{o}$ represents the offloaded task i's size to the neighboring collaborative smart device, and C_o^{ns} represents the local edge server's transmission rate to the collaborative smart device. Cap^{ns} represents the computing capabilities of other collaborative smart devices. $B_{i,ns}^{d}$ represents the size of the back-transfer data of the offloaded task i from the neighboring collaborative smart device, and C_d^{ns} represents the backhaul data transmission rate from the collaborative smart device to the local edge server.

When the tasks are completed, the edge service provider returns the result to the end-user. The delay of this part can be expressed by

$$T^{d} = \sum_{i=1}^{\Gamma}\left(\frac{B_i^d}{F_i^d} + \frac{B_i^d}{C_i^d}\right). \tag{4}$$

Here, B_i^d represents the size of the backhaul data of the offloading task i, F_i^d represents the transmission rate of the backhaul using the fiber method, and C_i^d represents the transmission rate of the backhaul in the wireless method.

In summary, the total time delay to complete the task can be expressed by

$$T = T^{o} + T^{ne} + T^{ns} + T^{d}. \tag{5}$$

3.2. Energy Consumption Model. According to the Shannon formula, the transmission rate of uploading data and the transmission rate of backhauling data are shown in equations (6) and (7):

$$C_i^o = W_o \times \log_2^{\left(1+P_i^o \times G/P_n\right)}. \tag{6}$$

Here, W_o represents the bandwidth of the wireless transmission channel from the smart device to the local edge server, P_i^o represents the smart device's transmit power during the process of wirelessly uploading data, G represents the gain of wireless transmission channel from the smart device to the local edge server, and P_n represents the Gaussian

white noise power. $P_i^o \times G/P_n$ is the signal-to-noise ratio:

$$C_i^d = W_d \times \log_2^{\left(1+P_i^d \times G/P_n\right)}. \tag{7}$$

Here, W_d represents the bandwidth of the wireless backhauling data transmission channel from the local edge server to the smart device and P_i^d represents the transmit power of the local edge server in the process of backhauling data in the wireless link mode. According to the research in [24], $G = 103.8 + 20.9 \times \log_{10}^{(\text{dist})}$, where dist represents the distance between the smart device and the local edge server.

According to equations (6) and (7), we then obtain the energy consumption from the smart device to the local edge server and the energy consumption of the backhauling process after the task is completed, as below:

$$E^o = \sum_{i=1}^{\Gamma}\left(P_i^o \times \frac{B_i^o}{C_i^o} + P_f^o \times \frac{B_i^o}{F_i^o}\right), \tag{8}$$

$$E^d = \sum_{i=1}^{\Gamma}\left(P_i^d \times \frac{B_i^d}{C_i^d} + P_f^d \times \frac{B_i^d}{F_i^d}\right). \tag{9}$$

Here, P_f^o represents the upload power of optical fiber transmission and P_f^d represents the backhaul power of optical fiber transmission.

When the edge service provider has no resources to use, some tasks can be offloaded to the servers of other collaborative edge service providers or smart devices for execution. This part of the energy consumption includes the energy consumption of tasks transferred to the collaboration server and transfer to the collaboration energy consumption of edge devices. In this case, the energy consumption can be expressed by equations (10) and (11), respectively:

$$E_{ne}^o = \sum_{i=1}^{\kappa}\left(P_{ne}^o \times \frac{B_{i,ne}^o}{C_o^{ne}}\right), \tag{10}$$

$$E_{ns}^o = \sum_{i=1}^{l}\left(P_{ns}^o \times \frac{B_{i,ns}^o}{C_o^{ns}}\right). \tag{11}$$

Here, P_{ne}^o represents the power offloaded from the local edge device to the cooperative edge server and P_{ns}^o represents the power offloaded from the local edge device to the collaborative smart device.

When executed on the server of the collaborative edge service provider, the energy consumption of this part includes the energy consumption of task execution and the energy consumption of the return after the task is completed. In this case, the energy consumption can be expressed by

$$E^{ne} = \sum_{i=1}^{\kappa}\left(P_{ne}^{im} \times \frac{B_{i,ne}^o}{\text{Cap}^{ne}} + P_{ne}^d \times \frac{B_{i,ne}^d}{C_d^{ne}}\right). \tag{12}$$

Here, P_{ne}^{im} represents the power performed by the cooperative edge server and P_{ne}^d represents the backhaul power after the completion of the cooperative edge device. $B_{i,ne}^o$ represents the size of task i offloaded to the adjacent collaborative edge server, and Cap^{ne} represents the computing power of the collaborative edge server. $B_{i,ne}^d$ represents the size of the returned data after task i is offloaded to the adjacent collaborative edge server for execution, and C_d^{ne} represents the backhaul data transmission rate from the collaborative edge server to the local edge server.

When executed on smart devices of collaborative edge service providers, this part of the energy consumption includes task execution energy consumption and backhaul energy consumption after the task is completed. In this case, the energy consumption can be expressed by

$$E^{ns} = \sum_{i=1}^{l}\left(P_{ns}^{im} \times \frac{B_{i,ns}^o}{\text{Cap}^{ns}} + P_{ns}^d \times \frac{B_{i,ns}^d}{C_d^{ns}}\right). \tag{13}$$

Here, P_{ns}^{im} represents the power executed by the collaborative smart device and P_{ns}^d represents the backhaul power after the completion of the collaborative smart device. $B_{i,ns}^o$ represents the size of the task i offloaded to the adjacent collaborative smart device, and Cap^{ns} represents the computing power of collaborative smart devices. $B_{i,ns}^d$ represents the size of the returned data after task i is offloaded to the neighboring collaborative smart device for execution, and C_d^{ns} represents the transmission rate of backhaul data from the collaborative smart device to the local edge server.

In summary, the total energy consumption to complete the task can be expressed by

$$E = E^o + E_{ne}^d + E^{ne} + E_{ns}^d + E^{ns} + E^d. \tag{14}$$

3.3. Cost Model. Each edge service provider completes various tasks of different types submitted from end-users by providing relevant resources. Each edge service provider incurs expenses when performing tasks. The cost function of the edge service provider can be defined as below:

$$C^{\text{esp}} = \sum_{i=1}^{\Gamma}\left(p_i^c \times B_i^o \times \Omega_i\right) + \sum_{i=1}^{\Gamma}\left(p_i^s \times B_i^o \times \xi_i\right) + \sum_{i=1}^{\Gamma}\left(p_i^m \times B_i^o \times \zeta_i\right), \tag{15}$$

where Γ represents the number of offloaded tasks; Ω_i, ξ_i, and ζ_i are coefficients; p_i^c represents the price of CPU for the edge server; p_i^s represents the price of storage for the edge server; and p_i^m represents the price of memory for the edge server.

When a large number of tasks are waiting to be executed in the edge server, the waiting tasks can be offloaded to other collaborative edge servers and another collaborative edge device. In this case, the cost function of the cooperative service provider can be defined as shown in

$$C^{co} = \sum_{i=1}^{\kappa}\left(p_i^{ne} \times B_{i,ne}^o \times \Upsilon_i\right) + \sum_{i=1}^{l}\left(p_i^{ns} \times B_{i,ns}^o \times \Psi_i\right). \tag{16}$$

Here, κ represents the number of tasks offloaded to other collaborative edge servers and l represents the number of tasks offloaded to other collaborative edge devices. $\Upsilon_i = 1$ represents that task i is executed on the collaborative edge server; otherwise, $\Upsilon_i = 0$, and $\Psi_i = 1$ represents that task i is executed on the collaborative smart device; otherwise, $\Psi_i = 0$. p_i^{ne} represents the price of other collaborative edge servers executing various types of resources, and p_i^{ns} represents the price of other collaborative edge devices executing various types of resources.

3.4. User Satisfaction Model. In this paper, we introduce the user satisfaction model to evaluate the satisfaction degree of users. Based on the literature [25, 26], the satisfaction function can be defined as below:

$$Q_i = \beta \times \log_{\alpha}^{(\psi_i(1-p_i/\mathrm{bid}_i))}. \tag{17}$$

Here, α and β represent the correlation coefficient. ψ_i represents the priority of the task i submitted by the user. The larger the value of Q_i, the more satisfied the user i is. On the contrary, the more dissatisfied the user i is.

3.5. Problem Formulation. In urban IoT, many smart devices submit tasks to various edge service providers and realize their demand with payment. Due to the distance among each edge service provider device, the energy consumption of smart devices, and the quality of network communication, many tasks cannot reach the system simultaneously. We assume that each task can use multiple different resources, such as CPU, memory, and storage. Edge service providers have a large number of edge servers that can handle various types of tasks. When the edge server resources are not available, the task can be offloaded to other collaborative edge devices and collaborative smart devices collaboratively to complete tasks more quickly to ensure user satisfaction. CEC resource allocation is performed by the edge scheduler.

In the above description, it can be seen that the collaborative service provider has idle resources and obtains corresponding benefits by selling idle resources. In this paper, the utility of the collaborative service provider is expressed as

$$U^{co} = \theta_c(R^{\mathrm{esp}} - C^{co}) - \theta_e E^{co} - \theta_t T^{co}. \tag{18}$$

Here, R^{esp} represents that the cooperative service provider obtains revenue from the edge service provider, as shown in equation (19). The coefficients θ_e, θ_t, and θ_c, respectively, represent the weight coefficients of energy consumption, delay, and cost in the utility function of the cooperative service provider, with values ranging from 0 to 1:

$$R^{\mathrm{esp}} = \sum_{i=1}^{\Gamma} \left(\mathrm{pay}_i^{\mathrm{esp}} \times B_i^o\right). \tag{19}$$

C^{co} represents the cost of the collaborative service provider to perform the offloading task of the edge service provider, as shown in equation (16).

E^{co} represents the energy consumption of the collaborative service provider's offloading task of the edge service provider. Energy consumption includes the energy consumption of the execution and the return result, as shown in

$$E^{co} = \sum_{i=1}^{\Gamma} (E_i^{ns} + E_i^{ne}). \tag{20}$$

T^{co} represents the execution delay of the collaborative service provider, as shown in

$$T^{co} = \sum_{i=1}^{\Gamma} (T_i^{ns} + T_i^{ne}). \tag{21}$$

Therefore, the goal of collaborative service providers is to maximize their utility, and the utility optimization problem of collaborative edge service providers can be defined as shown in

$$\max_{j \in M} U_j^{co} \tag{22}$$

$$\text{s.t.} \begin{cases} \mathrm{C1}: \mathrm{pay}_i^{\mathrm{esp}} \geq p_i^{ns}, \\ \mathrm{C2}: \mathrm{pay}_i^{\mathrm{esp}} \geq p_i^{ne}, \\ \mathrm{C3}: \mathrm{Cap}_{ns}^{\min} \leq \mathrm{Cap}_i^{ns} \leq \mathrm{Cap}_{ns}^{\max}, \\ \mathrm{C4}: \mathrm{Cap}_{ne}^{\min} \leq \mathrm{Cap}_i^{ne} \leq \mathrm{Cap}_{ne}^{\max}, \\ \mathrm{C5}: 0 \leq \sum_{j=1}^{M} \sum_{i=1}^{\Gamma} R_{i,j} \leq R^{\max}, \\ \mathrm{C6}: \sum_{j=1}^{M} \sum_{i=1}^{\Gamma} E_{i,j}^{ns} \leq E_{\mathrm{battery}}. \end{cases} \tag{23}$$

Here, Γ represents the number of offloading tasks and M represents the number of edge service providers. T_{tolerate} represents the user's maximum tolerable waiting time, and E_{Battery} represents the maximum available battery of the collaborative edge service provider. $R^{\max}$ represents the number of maximum available resources. $\mathrm{Cap}_{ns}^{\min}$ and $\mathrm{Cap}_{ns}^{\max}$ represent the minimum and maximum processing capacity of the smart device of the collaborative service provider, respectively; $\mathrm{Cap}_{ne}^{\min}$ and $\mathrm{Cap}_{ne}^{\max}$ represent the minimum and maximum processing capacity of the server of the collaborative service provider, respectively. C1 and C2 represent that the fee paid by the edge service provider must be greater than or equal to the cost of the collaborative service provider to ensure that the collaborative service provider can provide services normally. $\mathrm{pay}_i^{\mathrm{esp}}$ represents the fees paid by the local edge service provider. C3 and C4 ensure that the offloaded tasks do not exceed the processing capacity of the collaborative service provider. C5 indicates that the resources allocated by the collaboration service provider are within the permitted range of available resources. C6 ensures that the energy consumption does not exceed the allowable range of battery power when performing tasks

on the smart devices allocated to the collaboration service provider.

The edge service provider purchases the resources of the collaborative service provider and still provides paid services for one or more end-users under the condition of limited resources. This paper expresses the utility function of the edge service provider as shown in

$$U^{\text{esp}} = \upsilon_c(R^{\text{eu}} - C^{\text{esp}}) - \upsilon_e E^{\text{esp}} - \upsilon_t T^{\text{esp}}. \tag{24}$$

Here, R^{eu} represents the service fee paid by the end-user to complete the task, as shown in equation (25). υ_e, υ_t, and υ_c represent the weight of energy consumption, delay, and cost of the edge service provider:

$$R^{\text{eu}} = \sum_{i=1}^{\Gamma} (\text{bid}_i \times B_i^o). \tag{25}$$

C^{esp} represents the cost of the edge service provider to perform the offloading task, as shown in equation (15).

E^{esp} represents the energy consumption generated by the edge service provider offloading the task to the cooperative service provider, as shown in

$$E^{\text{esp}} = \sum_{i=1}^{\Gamma} \left(E_{i,ns}^t + E_{i,ne}^t\right). \tag{26}$$

T^{esp} represents the transmission delay of the edge service provider, as shown in

$$T^{\text{esp}} = \sum_{i=1}^{\Gamma} \left(T_{i,ns}^o + T_i^{ns} + T_{i,ns}^d + T_{i,ne}^o + T_i^{ne} + T_{i,ne}^d\right). \tag{27}$$

Therefore, the goal of the edge service provider is to maximize its utility, and the utility optimization problem of the edge service provider can be defined as shown in

$$\max_{j \in N} \quad U_j^{\text{esp}} \tag{28}$$

$$\text{s.t.} \quad \text{C1} : R^{\text{eu}} \geq R^{\text{esp}}. \tag{29}$$

Here, N represents the number of end-users, and C1 represents that the revenue must be greater than or equal to the system expenditure to ensure the normal operation of the system.

End-users submit tasks that need to be processed to edge service providers. Because they need to pay relevant fees, they can obtain resources to perform the tasks. This paper defines the utility function of the end-user as shown in

$$U^{\text{eu}} = Q - \rho_c \times R^{\text{eu}} - \rho_e E^{\text{eu}} - \rho_t T^{\text{eu}}. \tag{30}$$

Here, ρ_e, ρ_t, and ρ_c represent the weights of energy consumption, delay, and cost of the end-user, respectively.

E^{eu} represents the energy consumption when the end-user offloads tasks to the edge service provider, as shown in

$$E^{\text{eu}} = \sum_{i=1}^{\Gamma} E_i^o. \tag{31}$$

T^{eu} represents the time delay for the end-user to complete the task, as shown in

$$T^{\text{eu}} = \sum_{i=1}^{\Gamma} T_i. \tag{32}$$

Therefore, the goal of each end-user is to minimize the payment, energy consumption, and time delay and maximize user satisfaction. The utility optimization problem of the end-user can be defined as shown in

$$\max_{j \in N} \quad U_j^{\text{eu}} \tag{33}$$

$$\text{s.t.} \begin{cases} \text{C1} : R^{\text{eu}} \leq \text{Budget}, \\ \text{C2} : \text{bid} \geq p, \\ \text{C3} : \sum_{i=1}^{\Gamma} B_i^o \leq B^{\text{total}}, \\ \text{C4} : B_i^o \geq 0, \\ \text{C5} : E^{\text{eu}} \leq E_{\text{battery}}, \\ \text{C6} : T^{\text{eu}} < T_{\text{tolerate}}. \end{cases} \tag{34}$$

Here, Γ represents the number of offloading tasks and N represents the number of end-users. T_{tolerate} represents the maximum tolerable waiting time of the user. E_{Battery} represents the maximum available battery power of the end-user, and B^{total} represents the upper limit of the executable task size. C1 ensures that the total cost of the task does not exceed the total budget. C2 guarantees that the edge service provider only provides the corresponding service when the user's payment is greater than or equal to the minimal price of the edge service provider. If the cost value paid by the user is less than the minimum price of the edge service provider, the edge service provider refuses to provide the service. C3 and C4 ensure that the offloading tasks can be executed. C5 represents that the energy consumption of transmission must be within the allowable range of battery power. C6 ensures that the delay does not exceed the maximum tolerable waiting time of the user.

Through the above analysis, the task offloaded by the end-user can be regarded as multiple objects, and the resources of the collaborative service provider can be regarded as boxes, and then, the problem can be transformed into the multiobjective maximum packing problem. It can be proved that such problems are NP-hard problems. This type of problem is more difficult to solve. Next, the game theory is used to find an approximate solution to the problem, and the validity of the solution is proved. This

paper uses the entropy weight method [27] to calculate the value of each parameter.

4. Proposed Mechanism

Different from the traditional edge computing task offloading mechanism, this paper takes other collaborative edge devices and other smart devices into consideration in the design of the task offloading mechanism. In this paper, the proposed task offloading mechanism has three types of participants who affect each other: smart devices and edge service providers and collaborative service providers. Their respective behavior strategies determine their ultimate benefits. For example, when a collaborative service provider provides services to an edge service provider, if the fee paid by the edge service provider to the collaborative service provider is too low, the resources of collaborative service providers for providing collaborative services to the edge service provider are correspondingly reduced. In this case, the delay for edge service providers to provide services to smart devices will increase, affecting the satisfaction and benefits of customers.

Conversely, when the edge service provider pays too much to the collaborative service provider without increasing user fees, this leads to a significant reduction to the benefits of the edge service provider. When the edge service provider is overpriced for providing services to smart devices, the smart devices will submit fewer tasks or not to the edge service provider. It will reduce the revenue of the edge service providers. On the contrary, when the price of the edge service provider is too low, a large number of tasks will be submitted to the edge service provider, which will increase the expenses of the edge service provider or reduce the actual income. It will increase the waiting time of smart devices and affect the quality of service. We can see interest relationships among smart devices, edge service providers, and collaborative service providers from the above analysis. Therefore, we propose a task offloading mechanism based on a two-stage Stackelberg game model. By calculating the equilibrium solution of the Stackelberg game, we can obtain the optimal resource allocation scheme. This paper assumes that the cooperative service providers are secure and reliable.

4.1. Game Model. In this paper, we propose a two-stage Stackelberg game model. The specific description of this model is given as below.

Stage 1: the collaborative service provider submits the number of its remaining resources and the corresponding price to the edge scheduler. The edge scheduler, as the leader of the game, submits the relevant price strategy to the collaborative service provider according to the actual condition. As a follower of the game, the cooperative service provider decides its resource allocation strategy according to the leader's strategy. Algorithm 1 is used to solve the resource allocation problem of the edge scheduler.

Stage 2: we regard the interaction between the edge service provider and each smart device as a repeated Stackelberg game with multiple participants. On the one hand, as long-term participants, edge service providers obtain profits by processing various tasks of smart devices. On the other hand, as short-term participants, smart devices complete the tasks by paying fees to edge service providers. Algorithm 2 is used to solve the problem of the task offloading strategy of users. In each round of the Stackelberg game, the edge service provider, as the leader, first chooses its bidding strategy. Then, the smart devices, as followers, can decide their bidding strategy based on the edge service provider devices and their prediction for the next round of the game. The edge scheduler is responsible for executing the bidding strategy of end-users and edge service providers and allocating resources according to the final price. When there are multiple edge service devices in the vicinity of a smart device, each smart device will choose single or multiple edge service providers to submit tasks for execution based on the price of each edge service provider. The bidding strategy of the edge service provider is shown in Algorithm 3. The processing diagram of this game model is shown in Figure 2.

4.2. Game Equilibrium Analysis. According to the proposed game model, end-users, edge service providers, and collaborative service providers can maximize their respective benefits. In this section, we will analyze the equilibrium of the two-stage Stackelberg game model.

Theorem 1. *The two-stage Stackelberg game proposed in this paper has a Nash equilibrium among end-users, edge service providers, and collaborative service providers.*

The detailed certification process is given in the appendix.

5. Numerical Results and Discussion

5.1. Experimental Parameter Setting. In order to simulate the collaborative edge computing scenario of the urban IoT, we set up an experimental environment. The specific configuration is as follows: Within the range of 1×1 (km^2), there are one cloud server, one edge server, two collaborative edge servers, and twenty collaborative smart devices. Collaborative smart devices are all smartphones. It can be extended to more edge devices and more edge servers, with similar results.

The task size varies from 0 to 9 G. The bandwidth between the edge and the cloud is 1 Gbps. The bandwidth between the edge and the collaborative edge is 54 Mbps. The bandwidth between the edge and the smart device is 40 Mbps. The CPU processing capacity of the cloud data center is 10 G cycles/s, and the CPU processing capacity of the edge server is 6 G cycles/s. For the edge server, its idle power is 135 W. Its peak power is 495 W. The number of cores is 22. For the cloud computing server, the idle power is 150 W. The peak power is 750 W, and the number of cores is 64 [28]. The prices of different resources of the edge service provider are as below: 3 for CPU resources, 0.1 for storage resources, and 0.05 for memory resources. The prices of different resources for cloud resource providers are 24 for CPU resources, 0.82 for storage resources, and 0.67 for

Input:
B_i, location information for collaborative service provider, the number of the collaboration service provider, the set $\mathscr{C}'$ is a subset of the cooperative offloading service provider strategy set $\mathscr{C}$
Output:
The optimal strategy set $\mathcal{S}$ of the edge scheduler
1: A weighted directed graph G can be generated according to the location of the collaborative service provider
2: While true do
3: Calculate the delay, energy consumption, and cost of offloading tasks
4: Using dynamic programming to solve the equilibrium solution $\mathcal{S}$ of the subgame on the cooperative offloading service provider strategy set $\mathscr{C}'$
5: Solve the optimal corresponding strategy $\mathbb{C}$ of the cooperative offloading service provider to the subgame equilibrium solution $\mathcal{S}$
6: If $\mathbb{C} \in \mathscr{C}'$ then
7: Return $\mathcal{S}$
8: Else
9: $\mathscr{C}' \leftarrow \mathscr{C}' \cup \{\mathbb{C}\}$
10: End If

Algorithm 1: Edge scheduler resource allocation algorithm.

Input:
Number of end-users N, Resource requirements of each end-user R_i, Each end-user pays the resource price bid_i, The size of the end-user offloading task B_i^o, End-user's budget $Budget$
Output:
End-user task offloading strategy
1: For $i = 1$; $i <= N$ do
2: if $bid_i \times B_i^o <= Budget$ then
3: if the resources of local edge service provider j meet the needs of end-user i then
4: The task of end-user i is offloaded to the local edge service provider j for execution;
5: beak;
6: endif
7: else if the resources of local edge service provider j do not meet the needs of end user i then
8: Calculate the size of the remaining unfinished tasks of the end-user i on the local edge service provider j;
9: Calculate the remaining budget of the end-user i;
10: endif
11: endif
12: endfor

Algorithm 2: End-user task offloading algorithm.

memory resources. The power of task upload is 1.3 W, and the power of task return is 1.2 W.

The available resources of the system are key indicators that will affect the performance of the system. In the experiment, we use the percentage of unoccupied resources to indicate the availability of resources. The percentage of unoccupied resources is set to 10%, 30%, 50%, 70%, and 90%, respectively. When the percentage of unoccupied is 10%, the system is extremely busy. The number of devices that can be serviced by the collaboration service provider is minimal. The system is idle when the percentage of unoccupied is 90%. The collaboration service provider can provide more services.

We compare three offloading modes with our model. The details are as follows:

(i) *Edge Execution*. Task offloaded to edge server for execution

(ii) *Edge and Cloud Collaborative Execution*. Task offloaded to edge and cloud for collaborative execution

(iii) *Cloud Execution*. Task offloaded to cloud server for execution

(iv) *Our Model*. Task offloading to other collaborative edge servers on the adjacent edges and (or) other smart devices for execution

5.2. Experimental Results and Analysis

5.2.1. Experiment 1: The Performance of the Proposed Mechanism. When the task size submitted by users varies from 1 to 9 G and the fee paid by the local edge service provider to the collaborative service provider varies from 5 to 30, Figure 3 shows the optimal utility determined by the local edge service provider under the Stackelberg equilibrium condition. We can observe that with the increase of

Input:
Number of edge service providers M,Number of tasks submitted by end-users Γ,end-user bids $\{bid_1, bid_2,\cdots,bid_m\}$, Bids of edge service providers $\{p_1, p_2,\cdots,p_N\}$, Resource requirements of each end-user R, the available resources RES, Small changes in the price of local edge service providers Δp_k
Output:
The optimal bidding strategy set S of the local edge service provider
1: For $i = 0$; $i <= M$ do
2: For $j = 0$; $j <= \Gamma$ do
3: if $bid_j < p_i$ and $R > RES$ then
4: Denial of service;
5: else
6: Submit the task to the waiting queue;
7: endif
8: endfor
9: endfor
10: Sort the tasks in the waiting queue in descending order of the end user's bid
11: For $i = 1$; $i <= M$ do
12: For $k = 1$; $\mathbf{k} <= L$ do
13: Calculate the energy consumption, cost, and time delay of the task
14: if $U_i^{esp}(p_k+\Delta p_k) \le U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) \le U_i^{esp}(p_k)$ then
15: $S \longleftarrow S \cup \{p_k\}$
16: else if $U_i^{esp}(p_k+\Delta p_k) > U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) \le U_i^{esp}(p_k)$ then
17: $S \longleftarrow S \cup \{p_k+\Delta p_k\}$
18: else if $U_i^{esp}(p_k+\Delta p_k) \le U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) > U_i^{esp}(p_k)$ then
19: $S \longleftarrow S \cup \{p_k-\Delta p_k\}$
20: else if $U_i^{esp}(p_k+\Delta p_k) > U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) > U_i^{esp}(p_k)$ then
21: $S \longleftarrow S \cup \max\{p_k-\Delta p_k, p_k+\Delta p_k\}$
22: endif
23: endfor
24: endfor
25: Return S

Algorithm 3: Bidding strategies of the edge service provider.

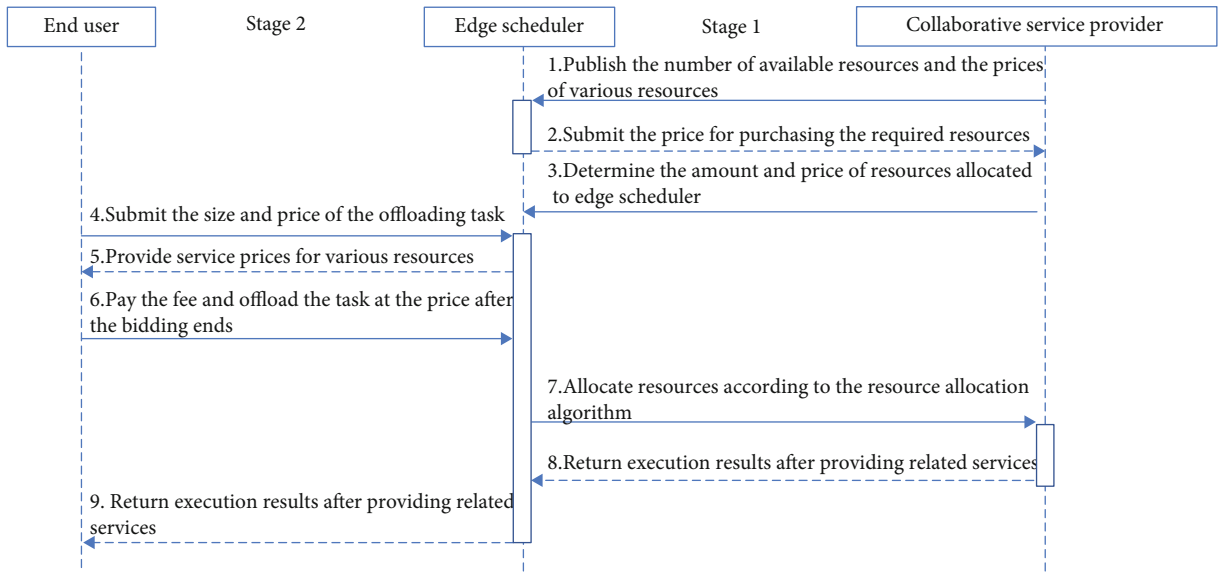


Figure 2: The sequence diagram of the game model.

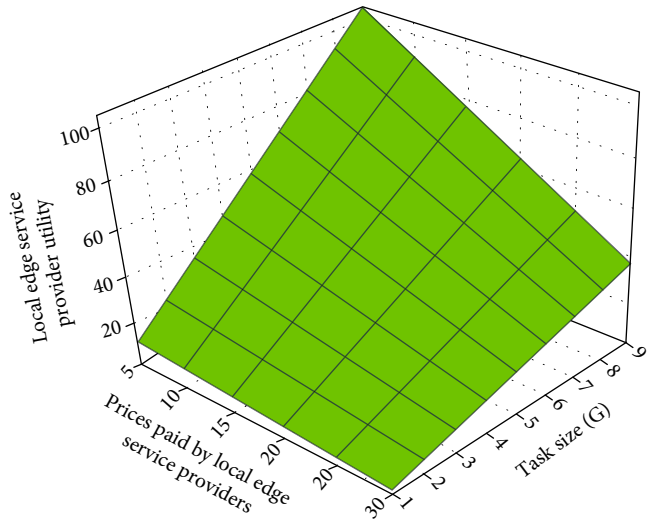


Figure 3: The utility of local edge service providers under different task sizes.

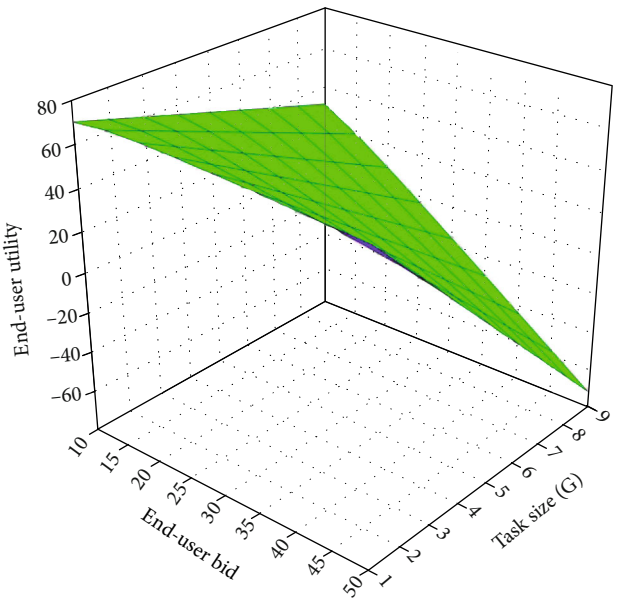


Figure 4: End-user utility under different task sizes.

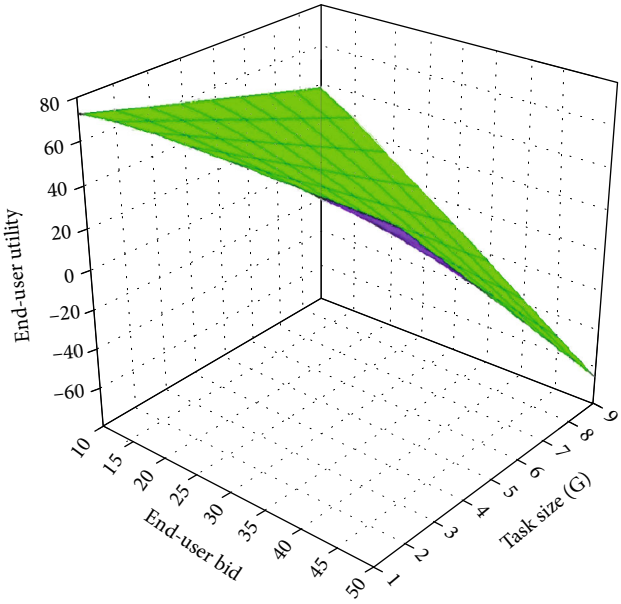
the number of tasks, the utility value of the local edge service increases correspondingly. To motivate the collaborative service provider to complete the tasks of the local edge provider, the utility value of the local edge service provider is reduced when the payment to the collaborative service provider is increased. The user's bid is fixed at 50 in this experiment.

When the task size varies from 1 to 9 G and the fee paid by the user varies from 10 to 50, Figure 4 shows the optimal utility determined by the end-user under the Stackelberg equilibrium condition. We can observe that with the increase of the number of tasks submitted by users, the time and energy consumption of the task execution increase correspondingly. To ensure user satisfaction, the utility value of end-users decreases. With the increase of the fee paid by users, users' expenses increase. It will directly lead to the decrease of utility value.

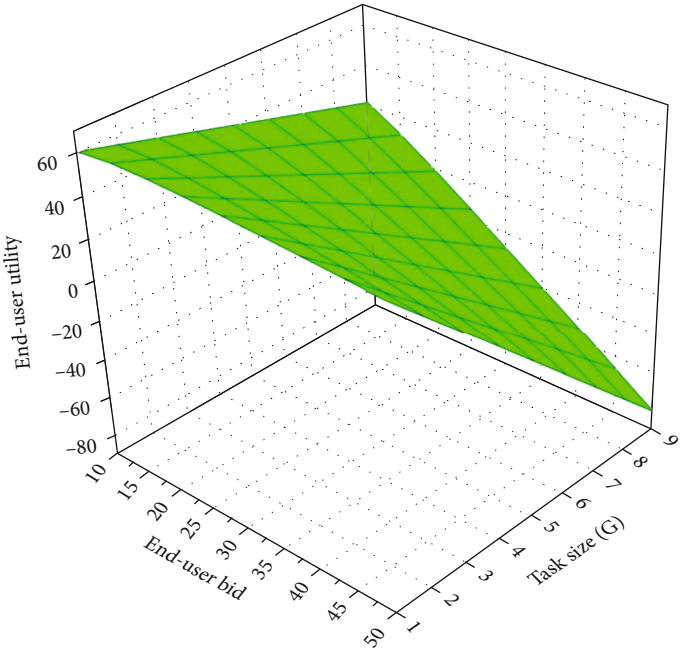
When the task size varies from 1 to 9 G and the fee paid by the user varies from 10 to 50, Figure 5 shows the end-user's optimal utility under the Stackelberg equilibrium condition. The priority of the task ranges from 0.1 to 1. The higher the priority value of the task submitted by the end-user, the more priority the task is executed. Due to space limitations, only $\psi = 0.5$ and $\psi = 0.75$ are shown, and other cases are similar to them. Figure 5(a) shows the result with $\psi = 0.75$, and Figure 5(b) shows the result with $\psi = 0.5$. It can be observed that with the increase of the number of tasks, the task priority is higher, and the user's utility value is greater. High-priority tasks are executed first. Moreover, high-priority tasks can get far more resources than low-priority tasks.

Figure 6 shows the optimal utility determined by the collaborative service provider under Stackelberg equilibrium when the collaborative service provider's task size varies from 1 to 9 G and the payment of the local edge service provider varies from 5 to 30. It can be observed that with the increase of the number of tasks processed by the collaborative service provider, tasks will be assigned to different collaborative service providers to execute in parallel; the corresponding execution time and energy consumption will be less than that of the sequential execution. Correspondingly, the utility of collaborative service providers will become greater and greater. When the number of tasks performed does not change, with the increase of payment of the local edge service provider, the revenue of the collaborative service provider and the utility value increase correspondingly.

When the proportion of unoccupied resources in the system varies from 10% to 90%, and the payment of the local edge service provider to the collaborative service provider varies from 5 to 30, Figure 7 shows the optimal utility of the local edge service provider under the Stackelberg



(a) The utility of the end-user when $\psi = 0.75$



(b) The utility of the end-user when $\psi = 0.5$

Figure 5: The utility of end-users under different priorities and different task sizes.

equilibrium condition. The task size is fixed at 9 G in this experiment. We can observe that with the increase of the unoccupied resources of the system, the system has more resources to process tasks submitted by end-users. Meanwhile, the utility value of the local edge service increases. To encourage collaborative service providers to complete the tasks of local edge providers, the utility value of local edge service providers decreases with the increase of fees paid to collaborative service providers. It can be explained as follows: with the increase of the fee provided by the local edge service providers, the profit value of local edge service providers decreases. We can observe that when the resource utilization rate is below 50%, the utility of the local edge service provider does not change much. When the system resource utilization rate is above 50%, the utility of the local edge service provider is greatly affected.

Figure 8 shows the end-user's optimal utility under the Stackelberg equilibrium condition when the proportion of unoccupied resources in the system varies from 10% to 90%, and the fee of the end-user varies from 10 to 50. It can be observed that with the increase of unoccupied resources of the system, the system can use more resources to handle the tasks submitted by end-users. Moreover, the execution time and energy consumption of users are

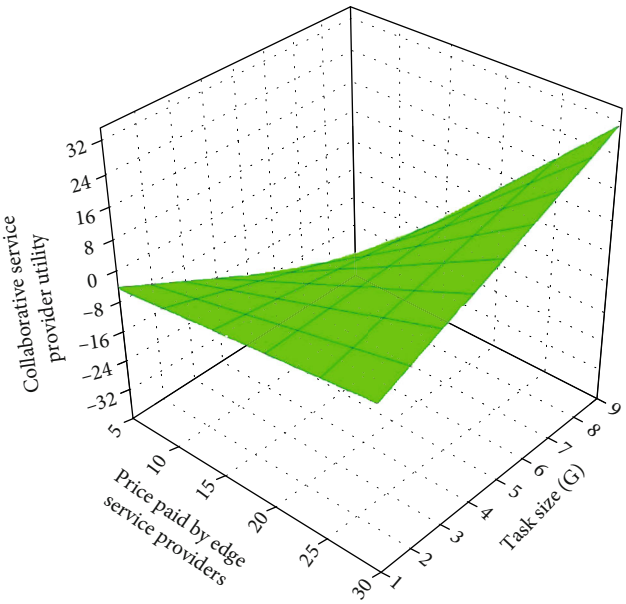


Figure 6: The utility of collaborative service provider in different task sizes.

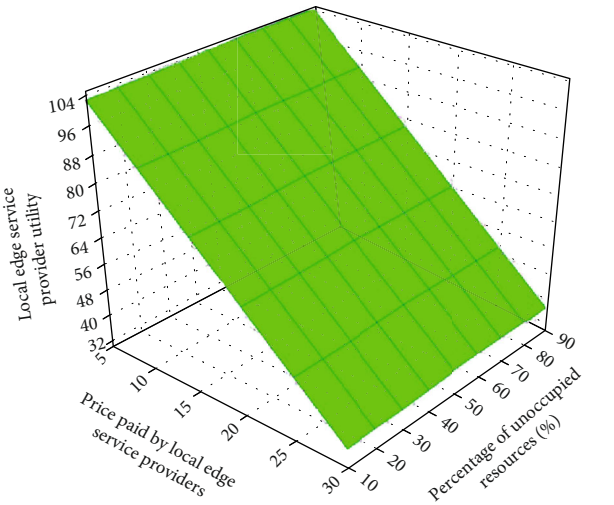


Figure 7: The utility of local edge service providers under different unoccupied resources.

reduced. Therefore, the utility value of end-users is reduced to ensure user satisfaction. To encourage the local edge service provider to complete the task, the end-user increases their payment under the condition of constant system utilization. When the utility value is less than 20, with the increase of the fee, the end-user's change is more significant. It indicates that the user expense has a greater impact on the utility value in this case. When the utility value is greater than 20 and less than 50, the change of the end-user's utility value is small. It indicates that the impact of the user expense on the utility value is small in this case.

Figure 9 shows the optimal utility of the user under the Stackelberg equilibrium condition when the proportion of unoccupied resources in the system varies from 10% to

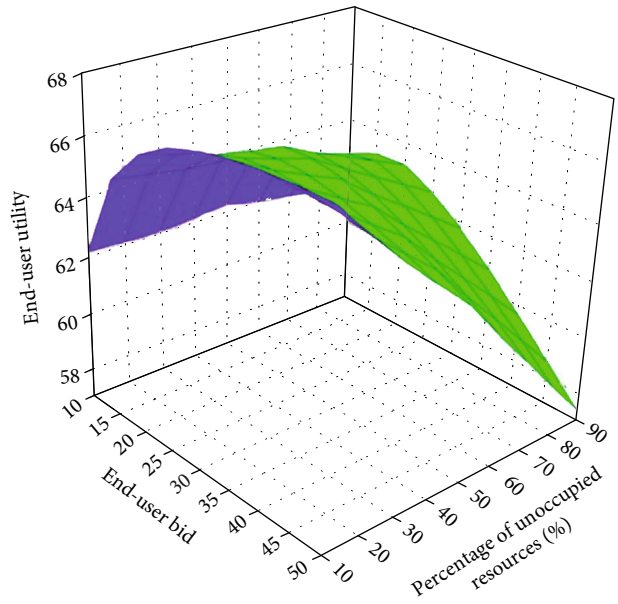
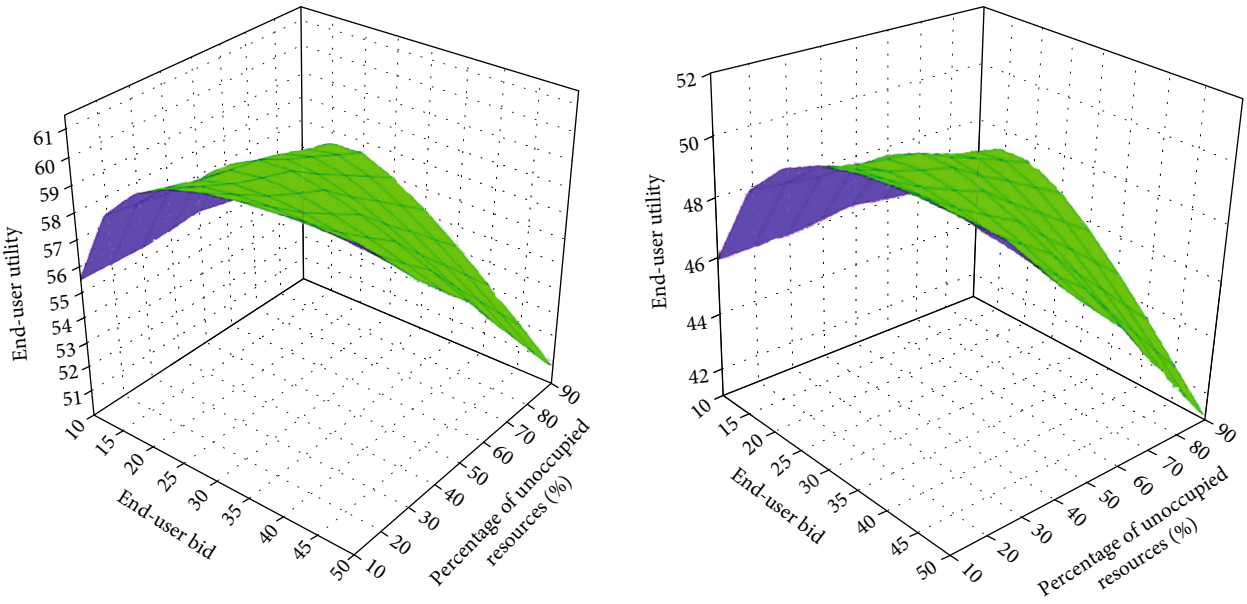


Figure 8: End-user utility under different resource utilization.

90%, and the user expense varies from 10 to 50. The priority of the task ranges from 0.1 to 1. The higher the priority value of the task submitted by the end-user, the more priority the task is executed. Due to space limitations, only $\psi = 0.5$ and $\psi = 0.75$ are shown, and other cases are similar to them. Figure 9(a) shows the result with $\psi = 0.75$. Figure 9(b) shows the result with $\psi = 0.5$. It can be observed that with the increase of the unoccupied resources of the system, the system can use more resources to process tasks. Meanwhile, the execution time and energy consumption of users are reduced. Therefore, the utility value of the user is reduced. Besides, with the increase of the priority of the task, the utility value of the user increases. The reason is that with the increase of the task priority, the waiting delay of the task decreases. More resources can be obtained for high-priority tasks.

Figure 10 shows the optimal utility determined by the collaborative service provider under Stackelberg equilibrium conditions when the proportion of unoccupied resources in the system varies from 10% to 90% and the payment of the local edge service provider varies from 5 to 30. It can be observed that with the increase of the unused resources of the system, the collaborative service provider has more resources to process the tasks submitted by the end-user. Therefore, the utility value of the collaborative service provider increases. It can be observed that when the remaining available resources are in the range of 10% to 30%, the improvement of the utility for the collaboration service provider is slower. The reason is that available resources are limited and the demand for a large number of tasks cannot be met in this case. With the increase of the number of available resources in the range of 30% to 70%, it can be found that the speed of processing tasks increases due to the increase of available resources. The execution time of the



(a) The utility of the end-user when $\psi = 0.75$

(b) The utility of the end-user when $\psi = 0.5$

Figure 9: End-user utility under different priorities and different resource utilization rates.

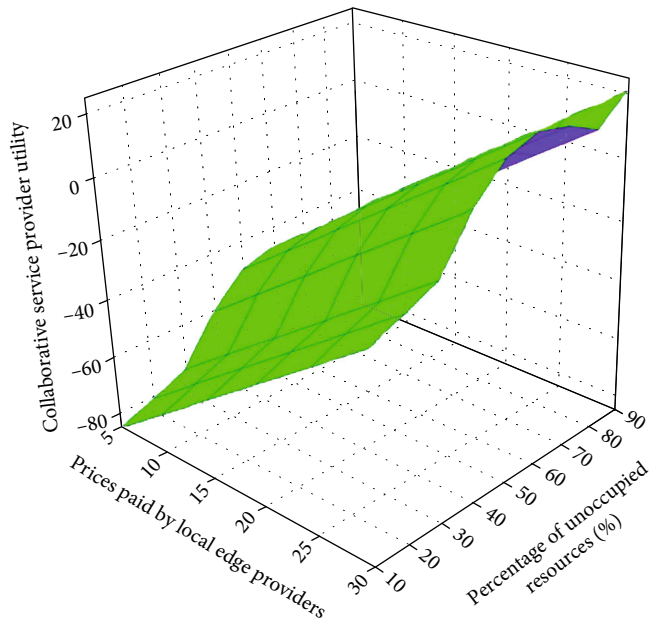


Figure 10: The utility of collaborative service providers in the case of different unoccupied resources.

task decreases. The revenue of the collaboration service provider increases significantly. When a large number of devices are in idle condition, the task execution time decreases slowly. The growth of the utility value for the collaborative service provider slows down.

5.2.2. Experiment 2: The Performance Comparison of Task Offloading Mechanisms with Different Task Sizes. We assign the task size for task offloading mechanisms from 0 to 9 G to evaluate how it affects the energy consumption and delay in the system. Figures 11 and 12, respectively, show the comparison results of the energy consumption and delay of computing offloading in different collaborative service modes under different task scales. The unused rate of resources is fixed at 90% in this experiment.

Figure 11 shows the comparison results of the energy consumption under different task sizes. It can be seen from Figure 11 that with the increase of the task number, energy consumption increases. By comparing the model proposed in this paper with cloud execution, edge execution, and

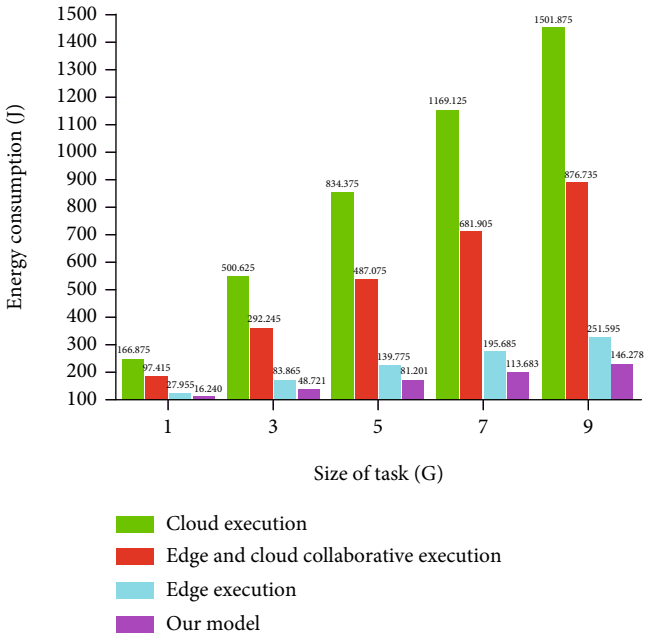


Figure 11: Comparison of energy consumption for different task sizes.

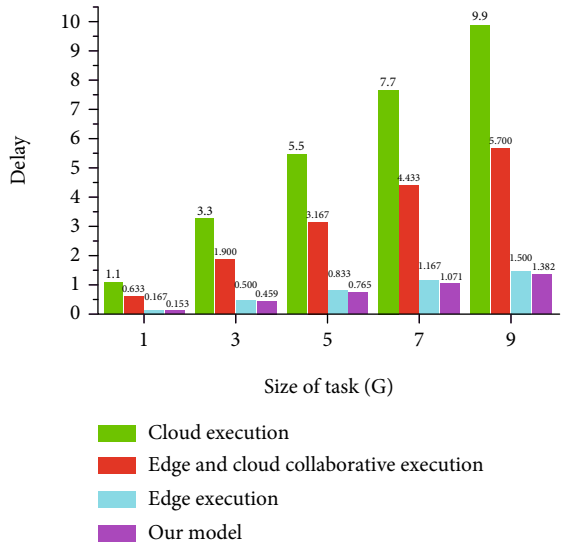


Figure 12: Comparison of delay for different task sizes.

edge-cloud collaborative execution, we observe that the energy consumption of the model proposed in this paper is less than that of the other models. When the task size is small (less than 1 G), the difference in energy consumption between the proposed model and the reference model is small. However, when the task size is large (larger than 9), the difference in energy consumption between the proposed model and the reference model is huge. All offloading to the cloud is 1501.875 J, while the energy consumption of the collaboration method proposed in this paper is 146.278 J. In terms of energy consumption, it can be seen that the model proposed in this paper has good results. As the number of tasks increases, the performance improved significantly.

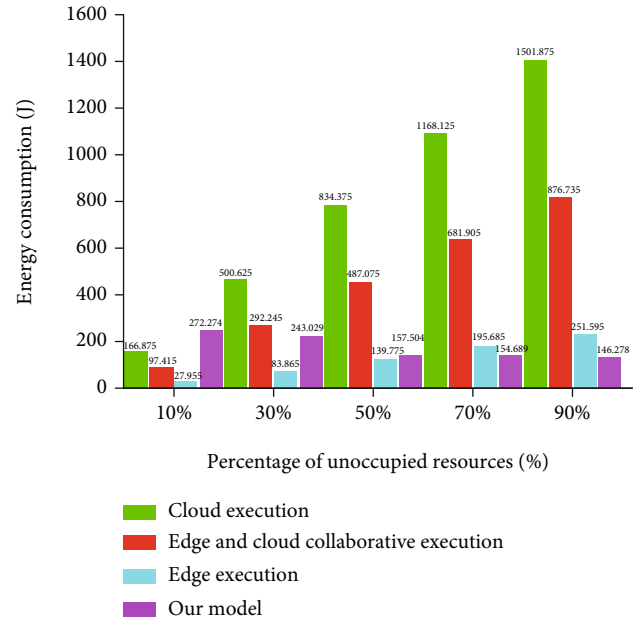


Figure 13: Comparison of energy consumption for different resource occupancies.

Figure 12 shows the comparison results of the delay under different task sizes. It can be seen from Figure 12 that with the increase of the number of tasks, the delay for tasks to be executed also increases. Compared to cloud execution, edge execution, and edge-cloud collaborative execution methods, the proposed model has better delay performance. When the task load is small (1 G), the difference of the delay for different methods is not significant, the longest value is 1.1 s, and the lowest is 0.153 s. However, when the task volume is large (9 G), the delay for all offloading tasks is 9.9 s, while the delay caused by the collaboration method proposed in this paper is 1.382 s. In terms of delay, it can be seen that the model proposed in this paper has good results. As the number of tasks increases, the model proposed in this paper also has better performance.

5.2.3. Experiment 3: The Performance Comparison of Task Offloading Mechanisms with Different Resource Usage. We assign the resource usage for task offloading mechanisms from 10% to 90% to evaluate how it affects the energy consumption and delay in the system. Figures 13 and 14, respectively, show the comparison results of the energy consumption and delay of computing offloading in different collaborative service modes under different resource usage. The task size is fixed at 9 G in this experiment.

Figure 13 shows the comparison results of the energy consumption under different resource usage. It can be seen from Figure 13 that with the increase of the percentage of unoccupied resources for collaboration service providers, more tasks will be executed. The energy consumption will increase correspondingly. By comparing the model proposed in this paper with cloud execution, edge execution, and edge

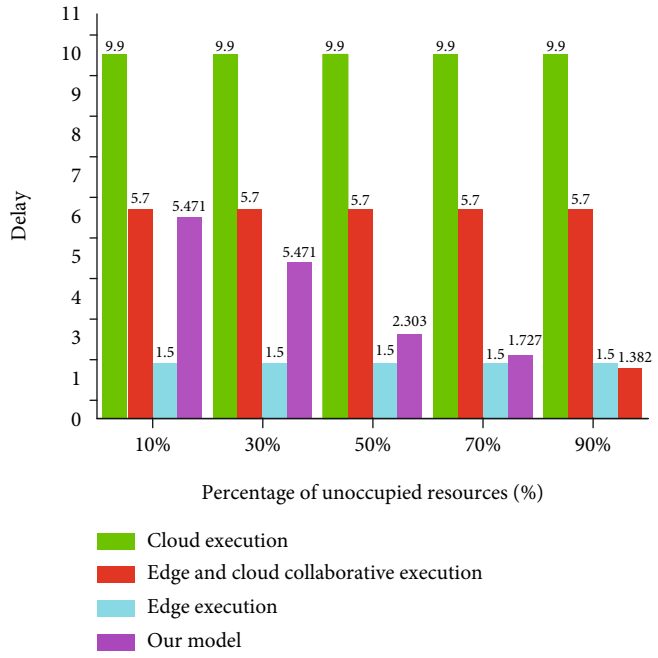
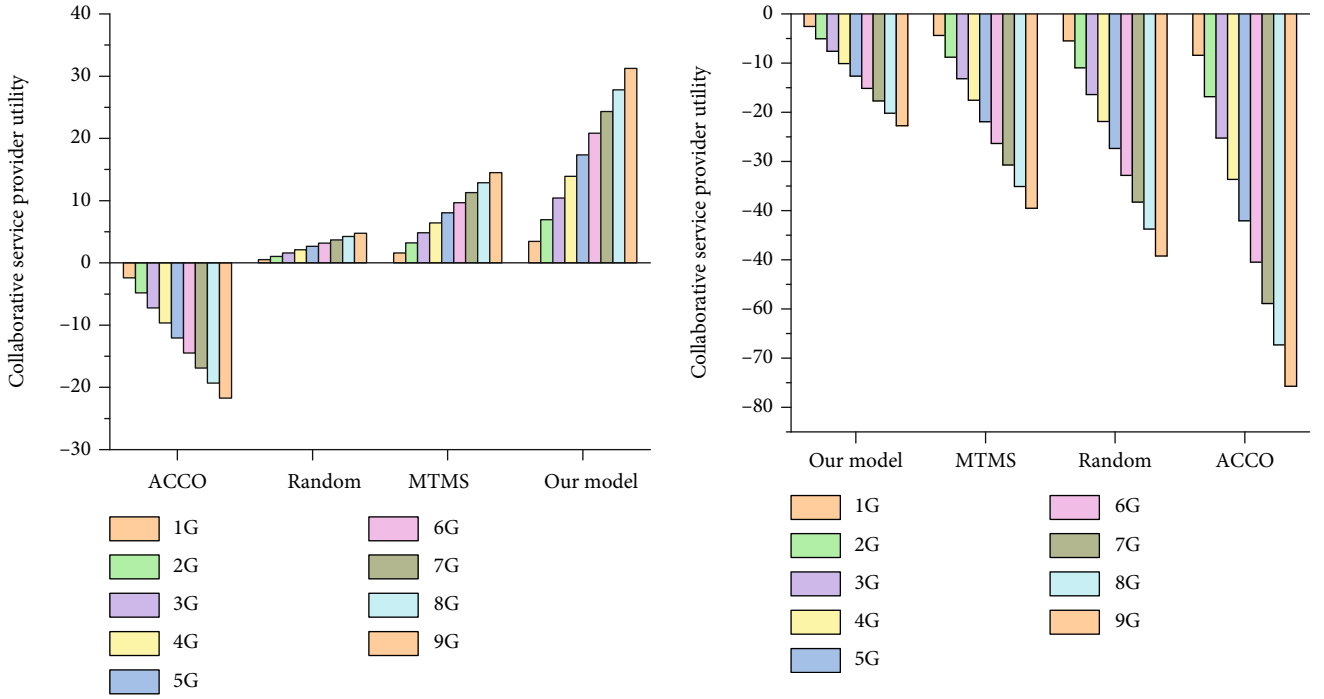


Figure 14: Comparison of delay for different resource occupancies of collaborative service providers.

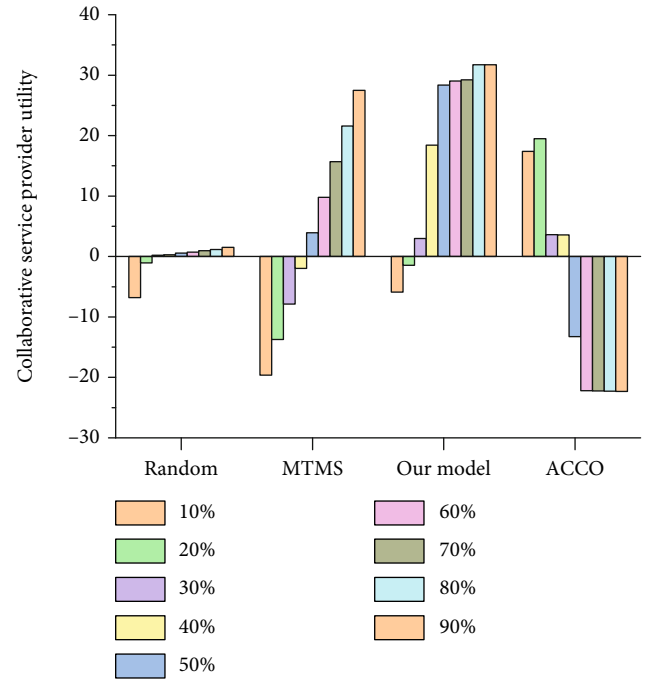


(a) The utility of the collaborative service provider utility when $pay^{esp} = 30$ (b) The utility of the collaborative service provider utility when $pay^{esp} = 10$

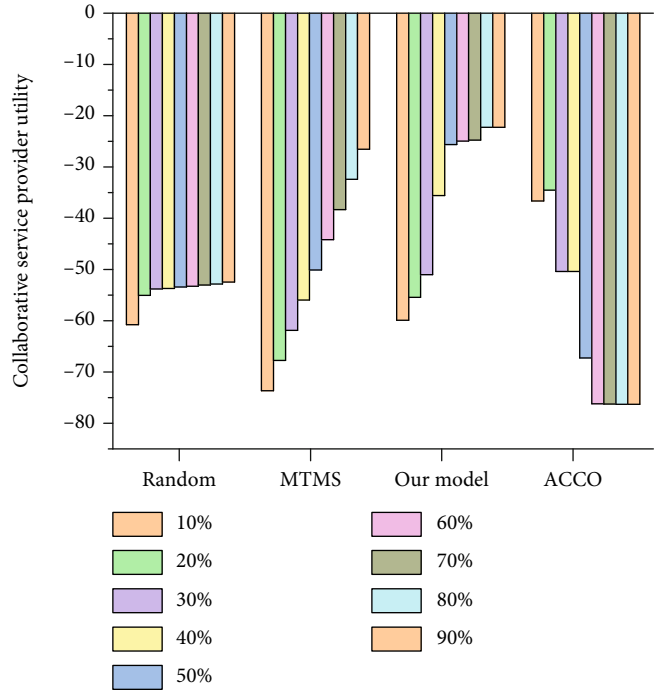
Figure 15: Comparison of collaborative service provider utility for different task sizes.

and collaborative cloud execution, it is found that the energy consumption of the model proposed in this paper is less than that of the other models when the unoccupied percentage of resources of the collaborative service provider is more than 50%. The energy consumption of the model proposed in this paper is higher than edge execution when the resource utilization rate is less than 50%. This result shows that when the collaborative service provider does not have free resources to complete all offloading tasks, many tasks are offloaded to the collaborative service providers and generate more energy than that executed at the edge.

Figure 14 shows that with the increase of the number of unoccupied resources for the collaboration service provider, more free resources are obtained, and more tasks are



(a) The utility of the collaborative service provider utility when $pay^{esp} = 30$ and task size = 9



(b) The utility of the collaborative service provider utility when $pay^{esp} = 10$ and task size = 9

Figure 16: Comparison of collaborative service provider utility for different resource occupancies.

executed. The latency of the model proposed in this paper is less than that of the other models. Compared to cloud execution, edge execution, and edge-cloud collaborative execution, when the percentage of unoccupied resources of the collaborative service provider is less than 70%, the latency of the proposed model is higher than that of the edge execution. It means that the collaborative service provider does not have free resources to complete all the offloading tasks. The offloading of many tasks to the collaborative service provider will not be processed. Therefore, there will be a larger delay compared to the execution at the edge.

5.2.4. Experiment 4: Comparison of the Proposed Mechanism, ACCO, MTMS, and Random. Experiment 1 shows that the proposed mechanism is affected by the change of its variables. Experiments 2 and 3 merely show that the

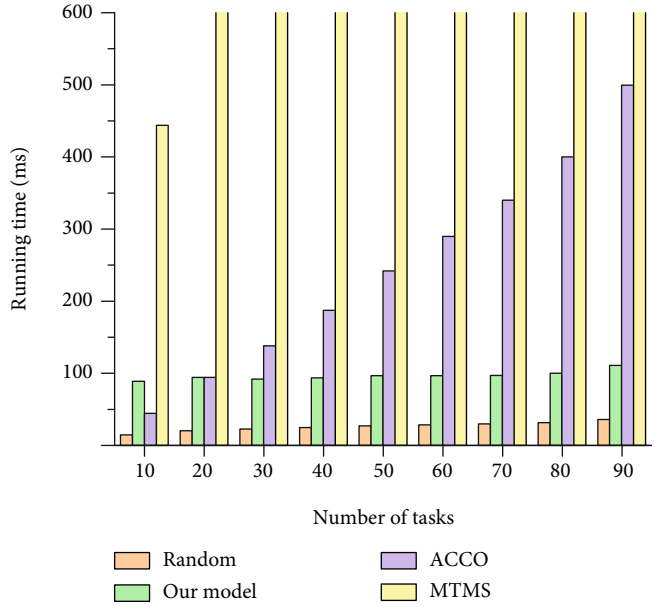


Figure 17: Comparison of the algorithm running time for different numbers of tasks.

performance comparisons of task offloading mechanisms are affected by the task size and resource usage. To prove the good performance of the proposed mechanism, we make a comparison of it with ACCO presented in [13], MTMS presented in [20], and the random allocation mechanism.

Figure 15(a) shows the utility of collaborative service providers for different task sizes when the edge service provider pays 30. It can be seen from the experimental results that the performance of the proposed scheme is better than the other three mechanisms because the edge scheduler in this mechanism obtains resources and services from edge servers and edge devices. The difference is that MTMS first divides the task into subtasks and then allocates the subtasks from the edge server where other resources are idle. When other edge servers are far away or resources are limited, there will be transmission or waiting-related delay, cost, and energy consumption. ACCO offloads the task to the remote cloud. As the size of the task increases, the user's energy consumption and time delay will increase, which will inevitably lead to a decrease in the utility value while the payment fee remains unchanged. Figure 15(b) shows the utility of collaborative service providers for different task sizes when the edge service provider pays 10. It can be seen from the experimental results that as the size of the task increases, when the edge service provider's pay is low, the utility function value of each mechanism has declined, and the decline rate of the scheme proposed in this paper is lower than that of the other three mechanisms. This shows that the delay and energy consumption of the proposed scheme in this paper are smaller than other schemes. However, the utility value of each mechanism is negative, indicating that the payment of edge service providers cannot meet the requirements of encouraging cooperative service providers to provide services.

Figure 16(a) shows the utility function value for different available resource percentages of collaborative service providers when the edge service provider pays 30. The experimental results show that with the increase of the scale of available resources, when the pay of edge service providers is high, only the utility value of the ACCO mechanism decreases, and other utility function values increase. This shows that when the available resources are only 10% to 30%, a large number of tasks are waiting to be executed, resulting in increased delay and energy consumption. However, the ACCO mechanism offloads tasks to the remote cloud. Because the processing capacity of the cloud data center is higher than that of other devices, the processing time and energy consumption are lower than other mechanisms. Therefore, the ACCO mechanism will be superior to other mechanisms. However, when the idle resources are higher than 30%, the energy consumption and cost of execution in the cloud are higher than other mechanisms due to factors such as distance, energy consumption, and cost. Therefore, other utility value increases, and the utility value of the ACCO mechanism decreases. Figure 16(b) shows the utility function value for different available resource percentages of collaborative service providers when the edge service provider pays 10. It can be seen from the experimental results that as the scale of available resources increases when the edge service provider's pay is low, the changing trend of the utility function value of each mechanism is similar to the result in Figure 16(a). However, the utility value of each mechanism is negative, indicating that the payment of edge service providers cannot meet the requirements of encouraging cooperative service providers to provide services.

5.2.5. Experiment 5: Running Time Comparison. Figure 17 shows the execution time of each mechanism under the different number of tasks. It can be seen from the figure that as the number of tasks increases, the execution time of each mechanism increases. Random runs the shortest, and MTMS runs the longest. This is because available resources

are randomly selected in the random mechanism, and the task will be offloaded as long as the execution conditions are met, while other methods need to consider the influencing factors such as energy consumption, delay, and cost and other influencing factors to make a comprehensive decision, so the execution time of other mechanisms is higher than random. However, it can be seen from previous experiments that although the execution time of the model proposed in this paper is not as good as the random mechanism, it is better than the random mechanism in other performances.

6. Conclusion

In order to reduce the time and energy consumption of task processing, tasks on edge servers with limited resources are offloaded to collaborative edge servers and edge devices for execution. Based on the collaboration task offloading mechanism, this paper proposed a two-stage Stackelberg game model to solve the interactive problem of the participants in the task offloading mechanism. The new proposal ensures the maximization of interests for all participants. Experiments and simulations verify the effectiveness of our method.

Appendix

Lemma 1. *The set of cooperative service providers' strategies can maximize their profits, and the optimal strategy is unique.*

Proof. According to equations (22) and (23), the first-order and second-order partial derivatives of the size of the offloading tasks from the local edge service provider to the collaborative service provider can be written as equations (35) and (36). Because the first derivative is greater than zero, the second derivative is equal to zero. Therefore, the function is incremental, there will be a maximum point, and there will only be one. The conclusion is established:

$$\frac{\partial U_i^{co}}{\partial B_i^o} = \theta_c\left(\text{pay}_i^{\text{esp}} - \sum_{i=1}^{l}\left(p_i^{ns} \times \Psi_i\right)\right) - \theta_e \sum_{i=1}^{l}\left(\frac{p_{ns}^{im}}{\text{Cap}^{ns}}\right) - \theta_t \sum_{i=1}^{l}\left(\frac{1}{C_o^{ns}} + \frac{1}{\text{Cap}^{ns}}\right), \tag{35}$$

$$\frac{\partial^2 U_i^{co}}{\partial B_i^{o2}} = 0. \tag{36}$$

□ □

Lemma 2. *The set of end-user's strategies can maximize their profits, and the optimal strategy is unique.*

Proof. According to equations (33) and (34), the first-order and second-order partial derivatives of the bid of the end-user are shown in

$$\frac{\partial U_i^{\text{eu}}}{\partial \text{bid}_i} = \sum_{i=1}^{\Gamma}\left(\frac{\beta \times p_i}{\text{bid}_i(\text{bid}_i - p_i) \times \ln \alpha} - \rho_c \times B_i^o\right), \tag{37}$$

$$\frac{\partial^2 U_i^{eu}}{\partial \text{bid}_i^2} = \sum_{i=1}^{\Gamma}\left(\frac{-(2 \times \text{bid}_i - p_i) \times \beta \times p_i}{\ln \alpha \times \left(\text{bid}_i^2 - p_i \times \text{bid}_i\right)^2}\right). \tag{38}$$

The second derivative is less than 0. Therefore, U_i^{eu} is a convex function. Since the function is increasing and convex, there is only one maximum point in the function. Therefore, the conclusion is established.□ □

Lemma 3. *The edge service provider's strategy set can maximize the benefits, and the optimal strategy is unique.*

Proof. In stage 2, the income of edge service providers can be expressed by

$$\frac{\partial U_i^{\text{esp}}}{\partial p_i} < 0, \tag{39}$$

$$\frac{\partial^2 U_i^{\text{esp}}}{\partial {p_i}^2} = 0. \tag{40}$$

Since the second derivative is equal to zero, the first derivative is less than zero. Therefore, the function U_i^{esp} is decreasing. The function has a maximum point, and there is only one. The conclusion is established.

In stage 1, the income of edge service providers can be expressed by

$$U_i^{\text{esp}} = -\upsilon_c \sum_{i=1}^{\Gamma}\left(\text{pay}_i^{\text{esp}} \times B_i^o\right) - \upsilon_e\left(\sum_{i=1}^{\Gamma}\left(P_f^o \times \frac{B_{i,ne}^o}{F_i^o} \times \varphi_{ne} + P_i^o \times \frac{B_{i,ns}^o}{C_o^{ns}} \times \varphi_{ns}\right)\right) - \upsilon_t\left(\sum_{i=1}^{\Gamma}\left(\left(\frac{1}{F_i^o} + \frac{1}{\text{Cap}^{ne}}\right) \times B_{i,ne}^o \times \varphi_{ne} + \left(\frac{1}{C_o^{ns}} + \frac{1}{\text{Cap}^{ns}}\right) \times B_{i,ns}^o \times \varphi_{ns}\right)\right). \tag{41}$$

For the first derivative and second partial derivative of the function for the payment of the edge service provider, the solutions shown in equations (42) and (43) can be obtained:

$$\frac{\partial U_i^{\text{esp}}}{\partial \text{pay}_i^{\text{esp}}} < 0, \tag{42}$$

$$\frac{\partial^2 U_i^{\text{esp}}}{\partial \text{pay}_i^{\text{esp}2}} = 0. \tag{43}$$

Since the second derivative is equal to zero, the first derivative is less than zero. Therefore, the function U_i^{esp} is decreasing. The function has a maximum point, and there is only one. The conclusion can be established.□ □

In summary, the edge service provider's strategy set can maximize the benefits, and the optimal strategy is unique. According to Lemmas 1–3, Theorem 1 is proved.

Data Availability

The simulation parameter data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grant 61572095 and Grant 61877007.

References

[1] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[2] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020.

[3] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: an energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[4] Y. Jie, C. Guo, K.-K. R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial internet of things environment," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3041–3052, 2020.

[5] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, SIAM, 1998.

[6] H. V. Stackelberg, *Marktform Und Gleichgewicht*, Springer, University of California, 1934.

[7] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[8] X. Niu, S. Shao, C. Xin et al., "Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things," *IEEE Access*, vol. 7, pp. 83771–83784, 2019.

[9] Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin, "Smart-Edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT," *IEEE Network*, vol. 33, no. 2, pp. 58–64, 2019.

[10] M. S. Parwez and D. B. Rawat, "Resource allocation in adaptive virtualized wireless networks with mobile edge computing," in *2018 IEEE International Conference on Communications (ICC 2018)*, Chengdu, China, 2018.

[11] Z. Zhang, J. Wu, L. Chen, G. Jiang, and S.-K. Lam, "Collaborative task offloading with computation result reusing for mobile edge computing," *The Computer Journal*, vol. 62, no. 10, pp. 1450–1462, 2019.

[12] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78685–78697, 2019.

[13] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[14] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2018.

[15] C. He, R. Wang, and Z. Tan, "Energy-aware collaborative computation offloading over mobile edge computation empowered fiber-wireless access networks," *IEEE Access*, vol. 8, pp. 24662–24674, 2020.

[16] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," *Sensors (Basel)*, vol. 19, no. 5, article 1105, 2019.

[17] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 959–962, Tianjin, China, 2019.

[18] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5444–5455, 2020.

[19] M. D. Hossain, L. N. Huynh, T. Sultana et al., "Collaborative task offloading for overloaded mobile edge computing in small-cell networks," in *2020 International Conference on Information Networking (ICOIN)*, pp. 717–722, Barcelona, Spain, 2020.

[20] J. Wang, W. Wu, Z. Liao, A. K. Sangaiah, and R. Simon Sherratt, "An energy-efficient off-loading scheme for low latency in collaborative edge computing," *IEEE Access*, vol. 7, pp. 149182–149190, 2019.

[21] R.-I. Ciobanu, C. Dobre, M. Balanescu, and G. Suciu, "Data and task offloading in collaborative mobile fog-based networks," *IEEE Access*, vol. 7, pp. 104405–104422, 2019.

[22] B. He, S. Bi, H. Xing, and X. Lin, "Collaborative computation offloading in wireless powered mobile-edge computing systems," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, Waikoloa, HI, USA, 2019.

[23] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative task scheduling for IoT-assisted edge computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020.

[24] M. Ding, D. Lopez-Perez, H. Claussen, and M. A. Kaafar, "On the fundamental characteristics of ultra-dense small cell networks," *IEEE Network*, vol. 32, no. 3, pp. 92–100, 2018.

[25] I. Rec, "G. 107-the E model, a computational model for use in transmission planning," *International Telecommunication Union*, vol. 8, no. 20, 2003.

[26] P. Li, Y. Wang, W. Zhang, and Y. Huang, "QoE-oriented two-stage resource allocation in femtocell networks," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pp. 1–5, Vancouver, BC, Canada, 2014.

[27] M. Zeleny, *Multiple Criteria Decision Making Kyoto 1975*, Springer, Berlin Heidelberg, New York, NY, USA, 2012.

[28] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *the 1st ACM symposium on Cloud computing*, pp. 39–50, Indianapolis, Indiana, USA, 2010.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 2050489, 18 pages
https://doi.org/10.1155/2021/2050489

Research Article

Tree Index Nearest Neighbor Search of Moving Objects along a Road Network

Wei Jiang,[1] **Fangliang Wei**,[1] **Guanyu Li**,[1] **Mei Bai**,[1] **Yongqiang Ren**,[1] **and Jingmin An**[1,2]

[1]*College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*
[2]*Faculty of Computer and Software, Dalian Neusoft University of Information, Dalian 116026, China*

Correspondence should be addressed to Guanyu Li; rabitlee@163.com

Received 23 June 2021; Revised 22 August 2021; Accepted 31 August 2021; Published 29 September 2021

Academic Editor: Pengfei Wang

With the widespread application of location-based service (LBS) technology in the urban Internet of Things, urban transportation has become a research hotspot. One key issue of urban transportation is the nearest neighbor search of moving objects along a road network. The fast-updating operations of moving objects along a road network suppress the query response time of urban services. Thus, a tree-indexed searching method is proposed to quickly find the answers to user-defined queries on frequently updating road networks. First, a novel index structure, called the double tree-hash index, is designed to reorganize the corresponding relationships of moving objects and road networks. Second, an index-enhanced search algorithm is proposed to quickly find the k-nearest neighbors of moving objects along the road network. Finally, an experiment shows that compared with state-of-the-art algorithms, our algorithm shows a significant improvement in search efficiency on frequently updating road networks.

1. Introduction

The rapid development of mobile communication and spatial positioning technology has extensively promoted the rise of location-based services (LBS). The LBS promotes the vigorous development of user-centric applications on urban transportation, and there has been widespread research in user recommendation systems and road network searching. One key issue of urban transportation is the nearest neighbor search of moving objects along a road network. However, the fast-updating operations of moving objects along a road network hinder the query response time of urban services. Currently, it is still difficult to quickly find the answers to user-defined queries on a frequently updating road network. Thus, an index-enhanced search algorithm is proposed to accelerate the response time to user-defined queries.

The response problem faced by user-defined queries on a road network is generalized as a k-nearest neighbor search problem (k-NN searching). Research on the k-NN problem in road networks has societal and commercial value [1]. For example, when someone uses the Meituan errand business, the system will send the order information to multiple salespersons nearest the user. Furthermore, k-NN technology promotes the improvement of mobile travel software, such as Didi and Uber, which frequently dispatches vehicles nearest to requesting users.

Figure 1 shows a simple example of a k-NN problem, which is a snapshot at time t. Assuming that $k = 4$, and there exists a user-defined query point q at time t, then, 4-NN searching can be conducted with different calculation strategies. Considering only the pairwise distance of a user and a moving object, then four moving objects (m_{12}, m_{13}, m_{11}, and m_8) are the closest to the query point q. However, taking into account the road intersection nodes and the moving objects' driving directions, the four moving objects (m_{13}, m_{11}, m_{12}, and m_{14}) respond to user q because the moving objects m_{12} and m_{14} can turn around on the road intersection nodes v_1 and v_3, respectively.

Existing research primarily focuses on the k-NN problem on road networks using Euclidean distance strategies

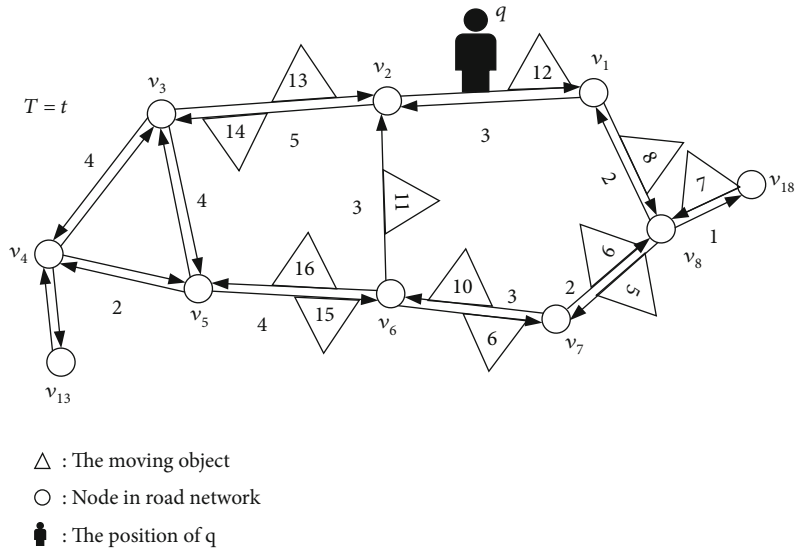


Figure 1: Segment of an urban road network.

for fixed objects [2–9] and Euclidean distance strategies for road-constrained moving objects [10–17].

The Euclidean distance strategies for fixed objects ignore the complexity of the road network, such as the nearest gas station on a highway, and prefer to calculate the shortest distance in a straight line in space, without considering the driving direction of vehicles and road intersection nodes. Road-constrained distance strategies more suitably address the complexity of transportation conditions and moving objects.

A preliminary work [17] provided an outstanding contribution to road-constrained distance strategies of moving objects. The work proposed a tree-based k-NN search algorithm. The method first constructs the balanced search-tree index, divides the whole road network into multiple partial road networks by hierarchical iteration, and then bounds the vertices to calculate the k-nearest objects. However, the strong search-tree index of the road network and moving objects results in a high cost to update the active records on the search-tree index.

A state-of-the-art work is the tree-decomposed k -NN search algorithm [18], which contains a tree-based decomposition strategy to establish the one-to-one relationship between each vertex and tree node. This method can quickly calculate the shortest distance between any two nodes; however, it needs to recalculate for updated inputs of k, resulting in increased time consumption thanks to redundant calculations.

Thus, we propose an index-enhanced search algorithm for the k-NN query problem of moving objects in a road network. The contributions of this paper are summarized as follows:

(1) A double tree-hash (DTH) index is designed to reorganize the weak relationships between the moving objects and the road network. The DTH index builds a weak bridge between tree and hash structures. The tree structure encapsulates the nodes, edges, and quantized relationships of intersubgraphs. The hash index records the moving objects. Through the weak bridge, the DTH index can realize the continuous updating-dynamic of moving objects in the road network

(2) An index-enhanced k-NN search algorithm is proposed to quickly find the k-nearest neighbors of moving objects on the road network. We call this the DMOK algorithm. Firstly, we establish an updating model to maintain the moving object updates while quickly and effectively expanding the positive nodes. Secondly, conditional distance rules are designed to reduce the number of expanded negative nodes. Finally, the k answers are responses to the user-defined requests and obtained using the DTH index

(3) Experiments with real data sets and artificial synthetic data sets verify the proposed algorithm's effectiveness

The structure of this paper is as follows. Section 2 reviews related work. Section 3 introduces the related definitions of the k-NN search problem for moving objects on a road network. A tree-hash index is presented in Section 4. Section 5 presents the index-enhanced k-NN search algorithm. Section 6 gives the experimental results and analysis. Conclusions are provided in Section 7.

2. Related Work

This section mainly summarizes the related research on the nearest neighbor search, including the related works on k-NN search problems for fixed objects (Section 2.1) and moving objects (Section 2.2) on a road network.

2.1. Related Works about k-NN Search Problem of Fixed Objects. The nearest neighbor search problem is one of the critical query types based on location services and was first proposed by Knuth in 1973 [19]. After ongoing research by many scholars, the theoretical knowledge has been enriched. The application field of the nearest neighbor

search has also been broadened, such that the nearest neighbor search owns a pivotal position in location-based services. The earliest studies on k-NN search problems mainly focused on fixed objects, and two classic k-NN search algorithms are IER [20] and INE [20].

Incremental Network Expansion (INE) adopted the greedy idea of Dijkstra's algorithm and expanded the answer by constantly asking the node closest to node q. The advantage of the INE algorithm is that the concept is clear, and the implementation is straightforward. However, its disadvantages are also apparent. For one, it needs to visit all the vertices close to vertex q. However, the number of these vertices may be much larger than k, and many of the visited nodes may be far away from query point q; thus, there is no need to visit them all.

In addition, the time and space complexities of the algorithm are high within large-scale data. For example, when using the priority queues to store the distance from other vertices to q, the algorithm's time complexity can reach $O(|W|\log|V|)$, where W is the number of edges and V is the number of vertices in a graph. Therefore, the INE algorithm has a good time and space complexity within small data sizes; it can easily make a preliminary judgment based on the input data size and then decide whether to choose the INE algorithm.

Kolahdouzan and Shahabi [21] proposed an improved algorithm, the Voronoi Network Nearest Neighbor (VN^3) algorithm, which answers k-NN queries by dividing the spatial network into multiple smaller Voronoi polygons based on the object-driven precalculating network distances. However, the performance of VN^3 depends on the density and distribution of the specific data set. As the data set becomes denser, the computational cost of calculating the distance each time becomes higher. Therefore, VN^3 is only suitable for sparse data sets.

The programming complexity of the Incremental Euclidean Restriction (IER) algorithm is higher than that of INE, and it needs to use data structures for preprocessing, such as R-tree. The advantage of the IER algorithm is that it avoids redundant nodes, especially nodes far away from query point q that should rarely be visited. The disadvantage of the IER algorithm is that it still needs to check the extreme cases of all nodes, and its worst time and space complexity is no less than that of the INR algorithm.

2.2. Related Works about the k-NN Search Problem of Moving Objects on a Road Network. Zhang et al. [22] proposed a keyword clustering query method (AKNN), which used user-specified query keywords to obtain text similarity. They designed a dual index structure (DG) generalized as a tree. The index structure saves the road network information and keeps the detailed knowledge of the network to establish an adjacency table. In literature [23], Huang et al. introduced the S-GRID algorithm to divide the spatial network into disjointed subnets and precalculated the shortest path of each pair within boundary nodes. In order to find k-nearest neighbors, network expansion is performed first within the subnet, and then, precalculated information is used to perform external expansion between boundary nodes. Samet et al. [10] proposed the DisBrw method, which precalculates the shortest path between all vertex pairs, and used a quadtree-based encoding for storing distance data. By storing the Euclidean distance as the shortest path distance boundary, they could find the k-nearest neighbors. Lee et al. [24] proposed a system framework called "ROAD" for spatial object searching on a road network. This method aimed at different object types effectively handled various location-related spatial queries, avoided the disadvantages that each node in the INE algorithm must be expanded, and directly skipped subnets with no obvious nodes of interest. Zhong et al. [25] proposed a balanced search-tree index (G-tree). G-tree uses a recursive segmentation method to divide the road network into corresponding subroad networks until the subnetwork is sufficiently small. This method can achieve an efficient search. Each node in G-tree is a subnet, and the leaf nodes in G-tree are equivalent to the nodes in a road network. If a leaf node and other external leaf nodes have directly connected edges, the leaf is the boundary of its upper network. Thus, G-tree calculates the shortest path of any two-leaf node boundaries on each divided network. In this paper, an assembly-based method is designed to calculate the minimum distance from the G-tree node to the query location and to reduce the complexity of the k-NN search problem.

Most studies on the underlying index structure employed the structures of R-tree [26] or other improved R-tree indices, such as $R*$-tree [27], FNR-tree [28], MON-tree [29], STCode-tree [30], and DBR$*$-tree [31]. Other works have used the general R-tree expansion, which converted two-dimensional space and one-dimensional time data into three-dimensional space data processing; examples include the 3DR-tree [32], RT-tree [33], and STR-tree [34]. Luo et al. [35] proposed a TOAIN algorithm. It used the SCOB index to optimize searching or update processing time and conducted workload analysis to configure SCOB indices to maximize throughput. Tianyang et al. [36] proposed a new algorithm for direction-aware KNN (DAKNN) searching of moving objects on a road network. This method used R-tree and a grid as the index structure, and the authors also proposed a novel local network extension method to quickly determine the direction of moving objects, reduce the communication costs, and simplify the calculation of the moving direction between moving objects and query points.

Two main issues should be considered: (1) road network environment representation and (2) moving object processing. Existing works mainly used data graphs to simulate the existing road network in terms of a road network environment [37]. $G = (V, E, W)$, where V is a node set and represents the connection between road segments, E is an edge set and illustrates the road segment in each road segment, and W means some additional information (road section cost, waiting time, etc.).

3. Problem Description

This paper focuses on quickly performing the k-NN search of moving objects on a road network environment and using the division strategy to improve the tree-mapping index

Table 1: Description of the symbols used in this paper.

Symbol	Meaning
K	Returns the number of moving objects
$\mathrm{Path}(v_1, v_2)$	Path from node v_1 to v_2
$\mathrm{Dist}(v_1, v_2)$	Distance from node v_1 to v_2
$\mathrm{SPath}(v_1, v_2)$	The shortest path from node v_1 to v_2
$\mathrm{SPDist}(v_1, v_2)$	The shortest path distance from node v_1 to v_2
M_N	Number of objects moved between two nodes
V_S	The starting node of the edge
V_E	End node of the edge
$\mathrm{AdSub}(v_i)$	Adjacency subgraph of v_i
$\mathrm{AcNode}(G_i)$	The active node of G_i
m.offset	The offset of the moving object from N_E
q.offset	Query the offset of point q from N_S
Cset	Candidate set of moving object results
R	k-NN query result set
$\mathrm{ShortChain}(v_i, v_j)$	The shortest chain from node v_i to v_j

mechanism for answering queries. Below, we introduce the definitions of the road network and associated concepts for the k-NN search on a road network environment. Table 1 lists and defines the main symbols used in this paper.

Definition 1 (road network). Given a directed weighted graph $G(V, E, W)$, where V is the road nodes and $V = \{v_1, v_2, \cdots, v_i, \cdots, v_n\}$. Each node v_i represents the intersection of roads on a road network, E is the set of directed edges on a road network, and $e_{1,2} = (v_1, v_2)$ means the edge from vertex v_1 to v_2. W represents the weights of the edge on the road network, in the paper, which indicates the distance of the edge $\mathrm{Dist}(v_1, v_2)$, formed as $w(v_1, v_2) = \mathrm{Dist}(v_1, v_2)$.

This paper studies the k-NN query problem of moving objects on a road network. Before introducing the definition of the k-NN query, some definitions of road networks and moving objects are given.

As shown in Figure 2, $\mathrm{Path}(v_6, v_8)$ represents the path from node v_6 to v_8 (for example, v_6-v_2-v_1-v_8 and v_6-v_7-v_8), and the Euclidean distance corresponding to the edge is $\mathrm{Dist}(v_6, v_8) = \mathrm{Dist}(v_6, v_2) + \mathrm{Dist}(v_2, v_1) + \mathrm{Dist}(v_1, v_8) = 8$ (or $\mathrm{Dist}(v_6, v_8) = \mathrm{Dist}(v_6, v_7) + \mathrm{Dist}(v_7, v_8) = 5$, etc.). On the road network, $\mathrm{SPath}(v_6, v_8)$ represents the shortest path from node v_6 to v_8. The shortest path of $\mathrm{SPath}(v_6, v_8)$ is v_6-v_7-v_8, which corresponds to the shortest distance $\mathrm{SPDist}(v_6, v_8) = 5$.

Definition 2 (moving object). On the road network, each moving object is expressed as $m = <(v_i, v_j), m.\mathrm{offset}, t>$; that is, at time t, the moving object m is on edge $<v_i, v_j>$, and the position offset from the vertex of v_j is m.offset.

As shown in Figure 2, the offset distance from m_{13} to the end node v_2 on the edge $e_{3,2}$ is $m_{13}.\mathrm{offset} = 2$, and $q.\mathrm{offset} = 1$ refers to the offset distance from the query object q to the start node v_2 on the edge $e_{1,2}$ which is 1.

Definition 3 (active node). A moving object m moves from node v_s to node v_e, and then, we call v_e a moving node of m. The active node of the subgraph G_i is denoted as $\mathrm{AcNode}(G_i)$.

For example, the moving object m_{13} moves from node v_3 to node v_2 on edge $< v_3, v_2 >$ in Figure 2. Therefore, v_2 is called an active node of m_{13}.

Definition 4 (moving object list). On a road network, a moving object is constantly moving, and its location information is regularly updated. The moving object list stores the positions of moving objects on the road network at a certain timestamp.

In Table 2, the moving object m_{11} exists on edge $E(v_6, v_2)$, and the distance from the terminal node v_2 is 1; meanwhile, the moving object m_{12} exists on edge $E(v_3, v_2)$, and the distance from the terminal node v_3 is 2. The number of moving objects with terminal node v_2 is 2. The moving object m_{14} exists on edge $E(v_2, v_3)$, and the distance from the terminal node v_3 is 2. Therefore, the number of moving objects with terminal node v_2 is 1.

4. Tree-Hash Index

This section introduces the double tree-hash (DTH) index proposed in this article, as well as the moving object update model (MOU-Model) based on the design of moving objects. We then introduce the processing method of moving object query on a road network, including the rapid solution of the road network distance and the fast processing of a query. Finally, a detailed introduction of the algorithm is given.

4.1. Division of Graphs. Due to the complexity of the road network, the calculation cost of the direct k-NN query is too high for our algorithm, and thus, we first divide graphs.

Definition 5 (k division of graph). Given a graph $G = (V, E, W)$, k division refers to dividing V into k subsets; that is, $V = V_1 \cup V_2 \cup \cdots \cup V_k$, and for any $i, j \in [1, k], i \neq j$, there is $V_i \cap V_j = \varphi$, $|V_i| = n/k$.

In the division of graphs, it is required that the number of edges E, where the associated vertices belong to different subsets, is minimized.

A tree hierarchy is established based on the graph division. The original graph is G, and graph G is divided iteratively as follows. Graph G is divided into $f = 2$ subgraphs G_1, G_2, G_3, and G_4; these subgraphs are regarded as subgraphs of G. The subgraph set of G is denoted by $C(G)$. G is the parent graph, and G_i^p represents the parent-child graph of subgraph G_i. Figure 3 shows the division process. For example, G_0 is divided into two subsubgraphs G_1 and

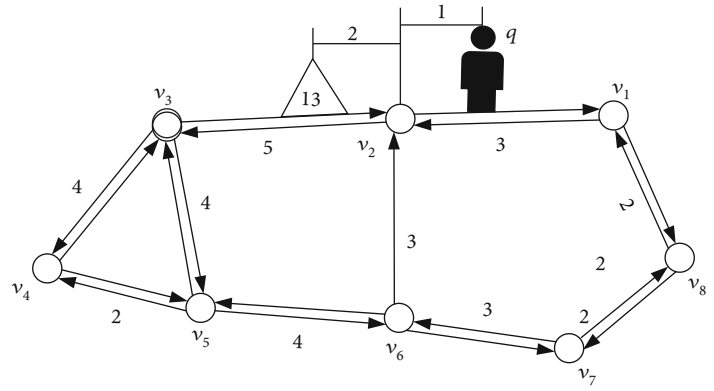


Figure 2: Moving objects under the road network with a query.

Table 2: Moving objects.

V_{end}	M	V_{start}	Offset	M_N
V_2	m_{11}, m_{13}	V_6, V_3	1,2	2
V_3	m_{14}	V_2	2	1

G_2, and then, $C(G_0) = (G_1, G_2)$ and $G_1^p = G_0$, as shown in Figure 3. Figure 4 is a subgraph of the road network.

Definition 6 (border). Given any one subgraph G_i, if node v_i ($v_i \in G_i$) is a boundary node, it needs to satisfy the following condition: there is a node v_j on the subgraph G_j ($i \neq j$), $v_i \in G_j$, such that $\langle v_i, v_j \rangle \in E$ is established. Thus, $B(G_i)$ is the set of boundary nodes of G_i.

4.2. DTH Index Construction. The double tree-hash (DTH) index is based on the characteristics of the road network structure and the characteristics of moving objects. By constructing a tree index structure, the shortest path of any two nodes in the road network can be quickly determined using the static and complex road network information; moreover, by constructing a distance hash table index structure, fast batch updates can be made for the frequently updated moving object information.

The divided subgraph information and the subgraph border nodes construct the road network tree structure, as shown in Figure 5.

Definition 7 (DisSet). The node distance set stores the global shortest distance between all nodes in each leaf subgraph and stores the global shortest distance between all border nodes in the nonleaf subgraphs with common ancestors. This is denoted as $\text{DisSet}(G_i) = \{S_i(v_i)\}$, $S_i(v_i) = \{v_j, \text{SPDist}(v_j, v_i), v_i, v_j \in G_i, i \neq j\}$. For the leaf subgraph, $S_i(v_i)$ stores the global shortest distance from other nodes in the subgraph to v_i; for nonleaf subgraphs, $S_i(v_i)$ stores the global shortest distance from other border nodes to border node v_i.

For example, for the leaf subgraph G_3 (Table 3), the global shortest distances between all nodes in the subgraph stored in the node distance set are stored in ascending order of $\text{DisSet}(G_3) = \{S_3(v_1), S_3(v_2), S_3(v_3), S_3(v_4), S_3(v_5), S_3(v_6), S_3(v_7), S_3(v_8)\}$, where $S_3(v_1) = \{(v_8, 2), (v_2, 3), (v_7, 4), (v_6, 4), (v_3, 8), (v_5, 10), (v_4, 12)\}$.

Tables 3 and 4 present the node distance sets of the leaf subgraph G_3 and the nonleaf subgraph G_2, respectively. The bold numbers in the tables are the distances between the border nodes and the other relevant border nodes.

Lemma 8 (see [17]). *Given a subgraph $G_i = (V_i, E_i, W_i)$, and vertices $v_i \in V_i$, $v_j \notin V_j$, the shortest distance from v_i to v_j must include a border node β of G_i. Then,*

$$SPDist(v_i, v_j) = SPDist(v_i, \beta) + SPDist(v_i, \beta). \quad (1)$$

Proof. Proof can be found in our technical report [38]. □

Corollary 9. *For the given two subgraphs G_i and G_j, the corresponding two nodes are $v_i \in G_i$, $v_j \in G_j$, and the two border nodes $\beta_i \in G_i$ and $\beta_j \in G_j$. Similarly,*

$$SPDist(v_j, v_i)SPDist(v_i, v_j) = SPDist(v_i, \beta_i) + SPDist(\beta_i, \beta_j) + SPDist(\beta_j, v_j). \quad (2)$$

Corollary 10. *For a given number of subgraphs G_i, G_j, and G_k, the corresponding nodes are $v_i \in G_i$, $v_j \in G_j$, and $v_k \in G_k$. G_k is the subgraph connecting G_i and G_j, and each subgraph corresponds to the border node $\beta_i \in G_i, \beta_j \in G_j$, $\beta_k \in G_k$. Similarly,*

$$SPDist(v_i, v_j) = SPDist(v_i, \beta_i) + SPDist(\beta_i, \beta_k) + SPDist(\beta_k, \beta_j) + SPDist(\beta_j, v_j). \quad (3)$$

Proof. According to Lemma 8 and Corollary 10, it can be proved. □

An example is shown in Figure 3, $v_{14} \in G_5, v_{31} \in G_6$, border nodes $v_{11} \in G_5, v_{10} \in G_6$, from Corollary 10, we

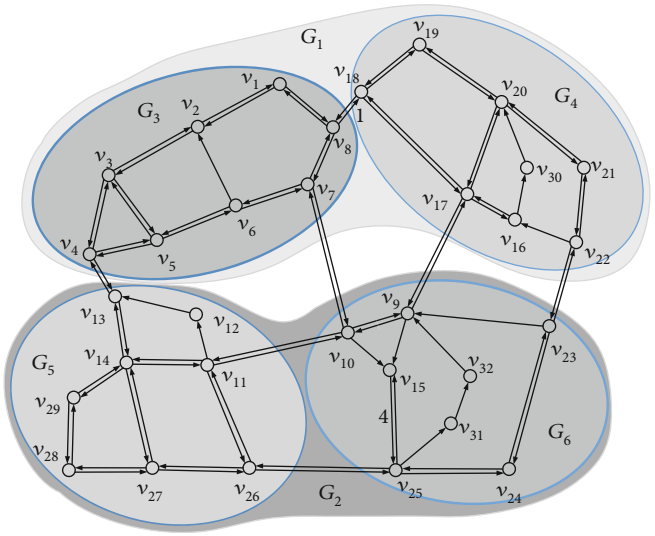


Figure 3: Subgraph division of road network.

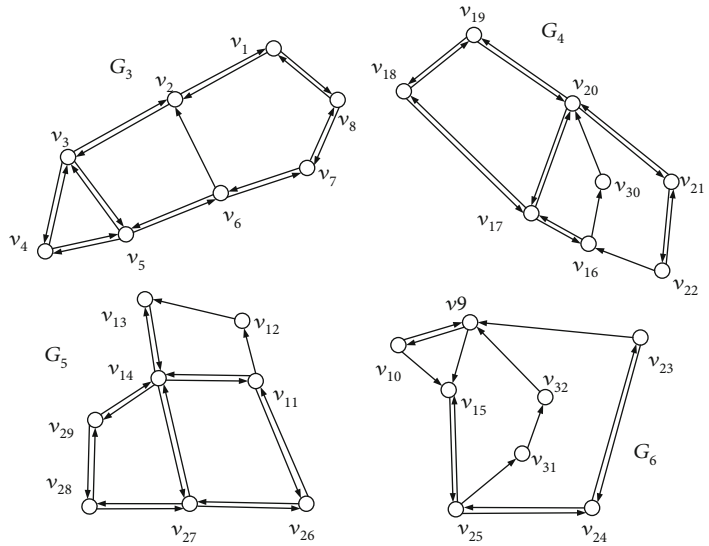


Figure 4: Subgraph of road network division.

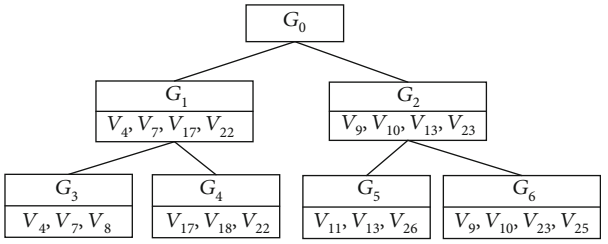


Figure 5: Tree structure diagram of the road network.

can get $\mathrm{SPDist}(v_{14}, v_{31}) = \mathrm{SPDist}(v_{14}, v_{11}) + \mathrm{SPDist}(v_{11}, v_{10}) + \mathrm{SPDist}(v_{10}, v_{31})$.

Definition 11 (activity-heap). Given a heap H, $v_i \in H$, if H satisfies $H = \{v_i \mid v_i \in B(G) \text{ or } v_i \in \mathrm{AcNode}(G)\}$, then H is called the active heap. This is denoted as H_{Ac}. If $H = \{v_i \mid v_i \notin B(G), v_i \notin \mathrm{AcNode}(G)\}$, then H is called an inactive heap and is denoted as H_{InAc}.

Corollary 12. *Assuming graph* $G = (V, E, W)$, $v_i \in G$, *then* $H_{Ac} \cup H_{InAc} = G(V)$.

Proof. According to Definition 11, it can be proved. Because $H_{\mathrm{Ac}} = \{v_i \mid v_i \in B(G) \text{ or } v_i \in \mathrm{AcNode}(G)\}$, $H_{\mathrm{InAc}} = \{v_i \mid v_i \notin B(G), v_i \notin \mathrm{AcNode}(G)\}$, $H_{\mathrm{Ac}} \cup H_{\mathrm{InAc}} = \{v_i \mid v_i \in B(G) \text{ or } v_i \in \mathrm{AcNode}(G)\} \cup \{v_i \mid v_i \notin B(G), v_i \notin \mathrm{AcNode}(G)\} = G(V)$. $H_{\mathrm{Ac}} \cup H_{\mathrm{InAc}} = G(V)$. □

4.3. DTH Index Process of Moving Objects. On the road network, the location information of moving objects (such as vehicles) is updated constantly. The V-tree index structure proposed in [17] needs to rerecord the states of the active vertices and update the recent global activities. In a busy road section, the position of moving vehicles changes

Table 3: S_3 node distance collection.

S_3	V_1	S_3	V_2		S_3	V_7	S_3	V_8
$\boldsymbol{V_8}$	**2**	V_1	3		$\boldsymbol{V_8}$	**2**	V_1	2
V_2	3	V_6	3		V_6	3	$\boldsymbol{V_7}$	**2**
$\boldsymbol{V_7}$	**4**	V_3	5	.....	V_1	4	V_2	5
V_6	6	$\boldsymbol{V_8}$	**5**		V_2	7	V_6	5
V_3	8	$\boldsymbol{V_7}$	**6**		V_5	7	V_5	9
V_5	10	V_5	7		$\boldsymbol{V_4}$	**9**	V_3	10
$\boldsymbol{V_4}$	**12**	$\boldsymbol{V_4}$	**9**		V_3	11	$\boldsymbol{V_4}$	**11**

Table 4: S_2 node distance collection.

S_2	V_{11}	S_2	V_{13}		S_2	V_{23}	S_2	V_{25}
$\boldsymbol{V_{10}}$	**5**	V_{11}	5		V_{25}	13	$\boldsymbol{V_9}$	**6**
$\boldsymbol{V_9}$	**6**	$\boldsymbol{V_{10}}$	**10**		$\boldsymbol{V_9}$	**19**	$\boldsymbol{V_{10}}$	**6**
$\boldsymbol{V_{13}}$	**6**	$\boldsymbol{V_9}$	**11**	.....	$\boldsymbol{V_{10}}$	**19**	V_{26}	10
V_{26}	7	V_{26}	12		V_{26}	23	$\boldsymbol{V_{11}}$	**11**
$\boldsymbol{V_{23}}$	**13**	$\boldsymbol{V_{23}}$	**18**		V_{11}	24	$\boldsymbol{V_{23}}$	**12**
V_{25}	16	V_{25}	18		$\boldsymbol{V_{13}}$	**30**	V_{13}	17

frequently in this environment. This update cost is very high, which is not conducive to query processing when moving objects change significantly. Thus, we propose a method based on updating moving objects. The road network structure diagram can be seen in Figure 6.

For example, in the bustling section of the city center, the number of moving vehicles is large, and the range of position changes is relatively large. The position of the moving object at $T = t$ and $T = t + 1$, and details of the moving objects are shown in Figure 7.

The existing k-NN algorithm only returns the k-nearest moving objects, regardless of their moving direction. After dividing the graph, according to the location relationship between the moving object and the query, it can be divided into the following situations.

As shown in Figure 8(a), the moving object is on the left side of the query, which is gradually approaching the query point along the road direction.

When $\mathrm{Dist}(v_2, v_1) - m_{12}.\mathrm{offset} \geq q.\mathrm{offset}$, the distance between the moving object and the query is

$$\mathrm{Dist}(m_{12}, q) = 2\mathrm{Dist}(v_2, v_1) - m_{12}.\mathrm{offset} - q.\mathrm{offset}. \quad (4)$$

Correspondingly, as shown in Figure 8(b), the moving object is on the right side of the query request point, that is, away along the road near the query point.

When $\mathrm{Dist}(v_2, v_1) - m_{12}.\mathrm{offset} < q.\mathrm{offset}$, it is necessary to recalculate the information of the nearest k moving objects. The distance between the moving object and the query is

$$\mathrm{Dist}(q, m_{12}) = \mathrm{Dist}(v_1, v_2) + m_{12}.\mathrm{offset} + q.\mathrm{offset}. \quad (5)$$

Table 5 lists the moving objects at time $T = t$. The gray entries denote nodes that are both a border node in the subgraph and an active node of the moving object.

For example, $G_4(2)$ means that at time t, there are two moving objects (m_{23} and m_{24}) in subgraph G_4, and they are moving from node v_9 to node v_{17}. v_{17} is both the border node of subgraph G_4 and the active node of the moving objects m_{23} and m_{24}.

Due to frequent updates of moving objects on the road network, we can establish the hash table as shown in Tables 6(a) and 6(b), which use V_E as the hash key and use $(V_S, M, \mathrm{Offset}, M_N)$ as the value; the mapping function is the $V_E \longrightarrow (V_S, M, \mathrm{Offset}, M_N)$.

For example, when key $= V_2$, value $= \{(V_6, m_{11}, 1, 2), (V_3, m_{13}, 2, 2)\}$, it means the moving object m_{11} is from V_6 to V_2, and $m_{11}.\mathrm{offset} = 1$, there are two moving objects on edge $<v_6, v_2>$.

5. Index-Enhanced k-NN Search Algorithm

This section mainly describes the proposed k-NN algorithm.

We present the calculation of the distance interval between two nodes and then the k-NN algorithm as a whole.

5.1. Node Expansion Strategy. Due to the complexity of the road network environment, calculating the global shortest distance between nodes requires an enormous time cost. This paper uses an inverted index structure to manage the distance calculated on the road network. Specifically, we sort the distance between two nodes from small to large and insert this information into the heap. When the distance between two nodes is an interval, we sort from small to large according to the upper bound of the interval. During subsequent calculations, we always process the first element in the heap H.

Given two nodes v_i and v_j, $v_i \in G_i, v_j \in G_j$, we quickly determine the global shortest distance between the two nodes.

Definition 13 (ShortChain). Given a subgraph $G_i = (V_i, E_i, W_i)$, for two nodes $v_i \in G_i$ and $v_j \in G_j$, the shortest chain of v_i, v_j refers to the DTH index. The path taken by the smallest common ancestor is denoted as $\mathrm{ShortChain}(v_i, v_j) = \{G_i \longrightarrow G_{i+1} \longrightarrow \cdots \longrightarrow G_j\}$, where G_i is the leaf subgraph to which v_i belongs and G_j is the leaf subgraph to which v_j belongs. The middle subgraph is the path of the divided subgraph (except for the common smallest ancestor) that G_i to G_j traverses.

For example, given two nodes v_{31} and v_2, the shortest chain from v_{31} to v_2 is $\mathrm{ShortChain}(v_{31}, v_2) = \{G_6 \longrightarrow G_2 \longrightarrow G_1 \longrightarrow G_3\}$, because $v_{31} \in G_6, v_2 \in G_3$, and $G_6 \neq G_3$, ${G_6}^p = G_2, {G_3}^p = G_1$, and $G_2 \neq G_1, {G_2}^p = G_0, {G_1}^p = G_0$, the node v_{31} and v_2 have the smallest common ancestor G_0. The shortest chain is shown in Figure 9.

Given two different nodes v_i and v_j, we have $v_i \in G_i, v_j \in G_j$, and $G_i \neq G_j$.

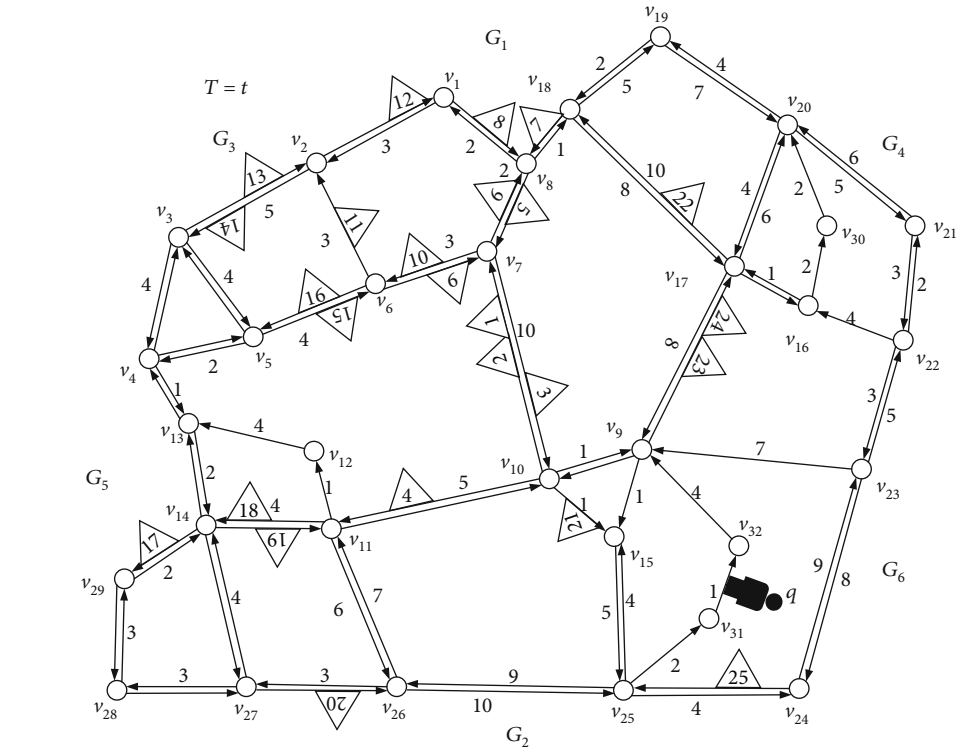
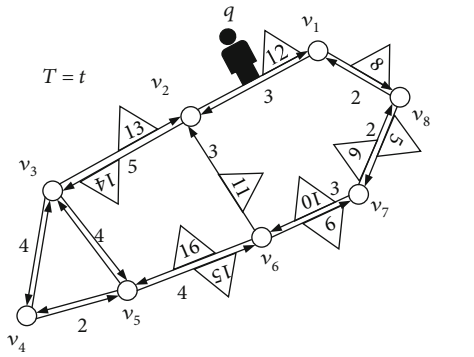
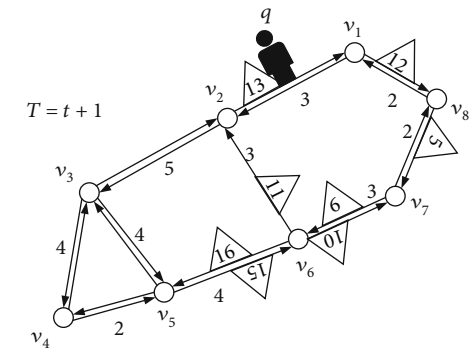


Figure 6: Road network structure diagram.

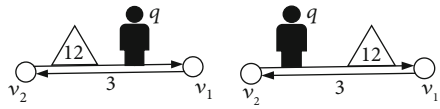


Moving object list ($T = t$)				
V_E	M	V_S	$Offset$	M_N
V_1	m_{12}	V_2	1	1
V_2	m_{11}, m_{13}	V_6, V_3	1,2	2
V_3	m_{14}	V_2	2	1
V_5	m_{16}	V_6	1	1
V_6	m_{10}, m_{15}	V_7, V_5	1, 1	2
V_7	m_5, m_9, m_1, m_2	V_8, V_6, V_{10}, V_{10}	1, 1.5, 2,4	4
V_8	m_7, m_8, m_6	$V1_8, V_1, V_7$	0.5, 0.5, 1	3
…	…	…	…	…



Moving object list ($T = t + 1$)				
V_E	M	V_S	$Offset$	M_N
V_1	m_{13}	V_2	2	1
V_2	m_{11}	V_6	1	1
V_4	m_{16}, m_{14}	V_6, V_5	1,1	2
V_6	m_6, m_{15}	V_7, V_5	2,1	2
V_7	m_5, m_{10}	V_8, V_6	1,2	2
V_8	m_{12}	V_1	1	1
…	…	…	…	…

Figure 7: Position of the moving objects at $T = t$ and $T = t + 1$.



(a) (b)

Figure 8: (a) M is on the left side of q; (b) M is on the right side of q.

Table 5: Moving objects at time $T = t$.

Moving object list ($T = t$)					
G_i	V_E	M	V_S	Offset	M_N
$G_3(14)$	V_1	m_{12}	V_2	1	1
	V_2	m_{11}, m_{13}	V_6, V_3	1, 2	2
	V_3	m_{14}	V_2	2	1
	V_5	m_{16}	V_6	1	1
	V_6	m_{10}, m_{15}	V_7, V_5	1, 1	2
	V_7	m_5, m_9, m_1, m_2	V_8, V_6, V_{10}, V_{10}	1, 1.5, 2, 4	4
	V_8	m_7, m_8, m_6	V_{18}, V_1, V_7	0.5, 0.5, 1	3
$G_4(3)$	V_{17}	m_{24}, m_{23}	V_9, V_9	2, 3	2
	V_{18}	m_{22}	V_{17}	2, 3	2
$G_5(5)$	V_{26}	m_{20}	V_{27}	1	1
	V_{29}	m_{17}	V_{14}	1	1
	V_{11}	m_4, m_{19}	V_{10}, V_{14}	2, 1.5	2
	V_{14}	m_{18}	V_{11}	2	1
$G_6(3)$	V_{10}	m_3	V_7	4	1
	V_{15}	m_{21}	V_{10}	0.3	1
	V_{25}	m_{25}	V_{24}	3	1

(1) We determine the leaf subgraph G_j where v_j is located and add the distances from all border nodes in G_j to v_j into the heap H, in ascending order

(2) We then process the first element v_{j-1} of the heap H and determine the next ShortChain subgraph as G_{j-1} according to the shortest chain ShortChain(v_j, v_i) sequence from v_j to v_i. Next, we calculate the respective border nodes of G_{j-1} to v_{j-1} and add them to the heap H in ascending order

(3) We end the expansion when the node expansion stop condition is met

When the node is expanded, the expansion is observed in heap H. When the following conditions are met for heap H, the node expansion ends:

(1) When the nodes in heap H are expanded in sequence

(2) When the nodes in the heap H have not been expanded, but the value of the first element of the heap H from the query point is greater than the set threshold or the maximum value in the candidate set

Table 6

(a) The hash table of moving objects ($T = t$)

Hash key (AcNode(V_i))	Value ($T = t$)			
V_E	M	V_S	Offset	M_N
V_1	m_{12}	V_2	1	1
V_2	m_{11}, m_{13}	V_6, V_3	1, 2	2
V_3	m_{14}	V_2	2	1
V_5	m_{16}	V_6	1	1
V_6	m_{10}, m_{15}	V_7, V_5	1, 1	2
V_7	m_5, m_9, m_1, m_2	V_8, V_6, V_{10}, V_{10}	1, 1.5, 2, 4	4
V_8	m_7, m_8, m_6	V_{18}, V_1, V_7	0.5, 0.5, 1	3
…	…	…	…	…

(b) The hash table of moving objects ($T = t + 1$)

Hash key (AcNode(V_i))	Value ($T = t + 1$)			
V_E	M	V_S	Offset	M_N
V_1	m_{13}	V_2	2	1
V_2	m_{11}	V_6	1	1
V_4	m_{16}, m_{14}	V_6, V_5	1, 1	2
V_6	m_6, m_{15}	V_7, V_5	2, 1	2
V_7	m_5, m_{10}	V_8, V_6	1, 2	2
V_8	m_{12}	V_1	1	1
…	…	…	…	…

Given two nodes, the calculation of the shortest chain is as shown in Algorithm 1.

5.2. DTH-Enhanced Nodewise Distance Calculation. This section mainly describes the algorithm for calculating the shortest distance between any two nodes in the road network environment based on the DTH index structure. This DTH-based algorithm to calculate the distance between two nodes (called DCDP) combines the advantages of the DTH index and can complete a search quickly. At the same time, we use the node expansion strategy to reduce unnecessary node expansion and network overhead.

Definition 15 (conditional distance). Given three different nodes v_i, v_j, and v_k, $v_i \in G_i, B(G_i) = \{v_k\}, B(G_j) = \{v_j\}$, and $G_i \neq G_j$. The distance from v_i through G_i's border node v_k to v_j in the subgraph G_j is called the distance from node v_i to v_j under the condition of passing node v_k. It is denoted as Dist$((v_i, v_j) \mid v_k)$.

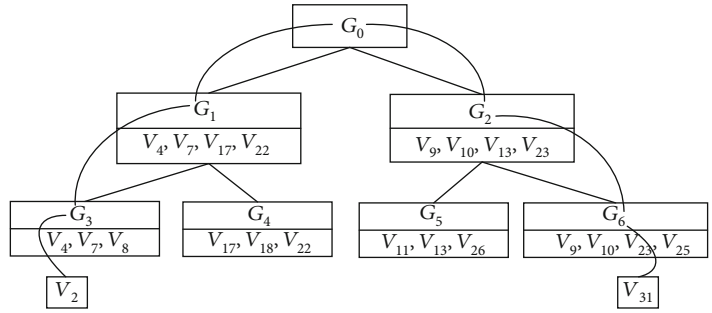


Figure 9: The shortest chain graph of nodes v_{31} to v_2.

Input: $v_i \in G_i, v_j \in G_j$
Output: **ShortChain**(v_i, v_j)
1 Initialize **ShortChain**$(v_i, v_j) = \{\varnothing\}$;
2 Make sure the subgraphs G_i, G_j where v_i, v_j are located;
3 while $G_i \neq G_j$ do
4 Add G_i, G_j to **ShortChain**(v_i, v_j), get **ShortChain**$(v_i, v_j) = \{G_i \longrightarrow G_j\}$;
5 Determine the parent node $G_i^p G_j^p$ of G_i, G_j;
6 $G_i = G_i^p$; $G_j = G_j^p$;
7 Output **ShortChain**(v_i, v_j)

Algorithm 1: The algorithm for calculating the shortest chain.

For example, for $v_{31} \in G_6$, we have $B(G_6) = \{v_{25}\}$ and $B(G_5) = \{v_{11}\}$. The distance from node v_{11} to node v_{31} through border node v_{25} is expressed as $\text{Dist}((v_{11}, v_{31}) \mid v_{25})$.

Lemma 16. *Given two nodes v_i and v_j, the shortest path must pass through the border nodes of the divided subgraphs corresponding to ShortChain(v_i, v_j), where multiple divided subgraphs can fit the same border node.*

In the above example, $\text{ShortChain}(v_{31}, v_2) = \{G_6 \longrightarrow G_2 \longrightarrow G_1 \longrightarrow G_3\}$. Then, the shortest path from v_{31} to v_2 is $v_{31}(G_6) \longrightarrow v_9(G_6, G_2) \longrightarrow v_7(G_1, G_3) \longrightarrow v_2(G_3)$.

We can directly obtain the distance between two nodes in the same subgraph through the node distance set, and thus, there is no need to calculate the shortest chain. For nodes in two different subgraphs, the process of calculating the shortest chain is as in Algorithm 1.

The process of calculating the global shortest distance between any two nodes is given below. According to the node distance set, the global shortest distance can be quickly obtained for two nodes in the same leaf subgraph (v_s, v_e):

(a) Use Algorithm 1 to calculate the shortest chain $\text{ShortChain}(v_e, v_s)$ between v_e and v_s; calculate the distance from the subgraph where v_s is located to the border node of the next chain subgraph according to the shortest chain sequence, and store it in heap H (whose elements are sorted from small to large)

(b) Take the first element V_{cur} of the heap H and expand according to the node expansion strategy

(c) Iterate successively until the next chain subgraph of the first element V_i of the first heap H is the same as the subgraph where v_e is located. Calculate $\text{Dist}(v_s, v_e)$, and temporarily store its value in $\text{Dis t}'(v_s, V_i)$

(d) Calculate other elements in heap H in turn. When $\text{Dist}(v_s, V_{cur}) < \text{Dist}'(v_s, V_i)$, $\text{SPDist}(v_s, v_e) = \text{Dist}(v_s, V_{cur})$; when $\text{Dist}(v_s, V_{cur}) \geq \text{Dist}'(v_s, V_i)$, terminate the algorithm

In Section 5.1, we introduce the algorithm used to calculate the shortest chain. Based on the DTH index and Algorithm 1, we propose the DCDP algorithm to calculate the shortest distance between any two nodes quickly. The $\text{SPDist}(v_{31}, v_2)$ solution process is shown in Figure 10. The specific description is given in Algorithm 2.

For example, the process of solving $\text{SPDist}(v_{31}, v_2)$ is as follows:

(1) According to Algorithm 1, the shortest chain from v_2 to v_{31} is $\text{ShortChain}(v_2, v_{31}) = \{G_3 \longrightarrow G_1 \longrightarrow G_2 \longrightarrow G_6\}$

(2) According to the order of the shortest chain $\text{ShortChain}(v_2, v_{31})$, expand using the node expansion strategy

Processing node	Border point of the next chain graph	Expansion of this node	Heap HS
$v_2 \in G_3$	$G_3(v_4, v_7, v_8)$	$SPDist(v_4, v_2) = 9$ $SPDist(v_7, v_2) = 6$ $SPDist(v_8, v_2) = 5$	$\boxed{v_8(5)}$ $v_7(6)$ $v_4(9)$
$v_8(5) \in G_3$	$G_1(v_4, v_7, v_{17}, v_{22})$	$Dist((v_4, v_2)\|v_8) = 16$ $Dist((v_7, v_2)\|v_8) = 7$ $Dist((v_{17}, v_2)\|v_8) = 16$ $Dist((v_{22}, v_2)\|v_8) = 18$	$\boxed{v_7(6)}$ $v_4(9)$ $v_{17}(16)$ $v_{22}(18)$
.....	.....	.....	.....
$v_{10}(16) \in G_6$	v_{31}	$Dist((v_{10}, v_2\|v_{31}) = 24$	$\boxed{v_{17}(16)}$ $v_9(17)$ $v_{22}(18)$ $v_{23}(28)$ $v_{25}(18)$ $v_{31} \to v_2(24)$
$v_{17}(16) \in G_1$	$G_2(v_9, v_{23})$	$Dist((v_9, v_2\|v_{17}) = 24$ $Dist((v_{23}, v_2\|v_{17}) = 26$	$\boxed{v_9(17)}$ $v_{22}(18)$ $v_{23}(26)$ $v_{25}(28)$ $v_{31} \to v_2(24)$
$v_9(17) \in G_6$	v_{31}	$Dist((v_{31}, v_2\|v_9) = 22$	$\boxed{v_{23}(23)}$ $v_9(24)$ $v_{25}(28)$ $v_{31} \to v_2(22)$
$v_{23}(23)$	$Dist(v_{23}, v_2) = 23 > Dist(v_{31}, v_2) = 22$		$v_{31} \to v_2(22)$

Figure 10: $\text{SPDist}(v_{31}, v_2)$ solution process.

Input: Given two nodes v_s, v_e, $\textbf{Dist}(v_s, v_e) = \infty$
Output: $\textbf{SPDist}(v_s, v_e)$
1 If v_s, v_e belong to the same subgraph, directly output $\textbf{SPDist}(v_s, v_e)$ according to the node distance set;
2 Determine the shortest chain $\textbf{ShortChain}(v_s, v_e)$;
3 $V' = \varnothing$; $V_{cur} = \varnothing$; Let $\textbf{Dist}(v_s, V') = \infty$;
4 Add v_e to heap H;
5 Make sure the subgraphs G_i, G_j where v_i, v_j are located;
6 **while** *H is not empty* **do**
7 Take V_{cur}, the first element of H;
8 **if** $\textbf{Dist}(v_s, V_{cur}) < \textbf{Dist}(v_s, V')$ **then**
9 break;
10 Get G_i's next chain graph G_j according to **ShortChain**;
11 **for** *every border point* V_i of G_j **do**
12 if $\textbf{Dist}(v_s, V_{cur}) < \textbf{Dist}(v_s, V_i)$ then
13 $\textbf{Dist}'(v_s, V_i) < \textbf{Dist}(v_s, V_{cur}) + Dist(v_s, V_i)$
14 **if** $\textbf{Dist}'(v_s, V_i) < \textbf{Dist}(v_s, V_i)$ **then**
15 $\textbf{Dist}(v_s, V_i) = \textbf{Dist}'(v_s, V_i)$
16 Add $V_i(\textbf{Dist}(v_s, V_i))$ to the heap H in descending order of value;
17 $V' = V_{cur}$;
18 $\textbf{SPDist}(v_s, v_e) = \textbf{Dist}(v_s, V_{Cur})$

Algorithm 2: DCDP algorithm().

(3) When the first element of heap H is v_9, calculate $\text{Dist}(v_{31}, v_2) = 22$

(4) When the expansion node is the first element v_{23} of heap H, $\text{Dist}(v_{23}, v_2) = 23 > \text{Dist}(v_{31}, v_2) = 22$, stop node expansion and terminate the algorithm

(5) Output $\text{SPDist}(v_{31}, v_2) = 22$

For any nodes v_i, v_j, v_k and subgraph G_j, G_j, the node distance set $\text{DisSet}(D_j)$, where $v_i \notin G_j, v_j \in G_j, v_k \in G_j$, and $\text{SPDist}(v_i, v_j)$ can be directly calculated by Algorithm 2. Through the node distance set $\text{DisSet}(D_j)$, we can obtain

Table 7: Distances to the border nodes between G_5 and G_6.

$B(G_5) \setminus B(G_6)$	$v_9(8)$	$v_{10}(8)$	$v_{23}(14)$	$v_{25}(2)$
v_{11}	6	5	24	11
v_{13}	12	11	30	17
v_{26}	13	12	23	10

the distance $\mathrm{SPDist}(v_j, v_k)$ between any two nodes in the subgraph. Then,

$$\mathrm{SPDist}(v_i, v_k) = \mathrm{SPDist}(v_i, v_j) + \mathrm{SPDist}(v_j, v_k). \quad (6)$$

5.3. Conditional Rules. To reduce the calculations required to obtain the distance between two nodes and perform node expansion, we propose a reduction strategy. First, through the expansion of the node, a distance interval is approximated, and the upper and lower bounds of the distance between the two nodes are determined. Then, in the expansion calculation, according to the upper and lower bounds determined above, we reduce the amount of calculation.

Definition 18 (distance interval). Given two nodes v_i and v_j ($v_i \in G_i, v_j \in G_j$) in different subgraphs, the maximum distance between node v_i and node v_j in subgraph G_j is called the upper bound of $\mathrm{Dist}(v_i, v_j)$ and is denoted as $\mathrm{upDist}(v_i, v_j)$. The minimum value of this distance is called the lower bound of $\mathrm{Dist}(v_i, v_j)$ and is represented as $\mathrm{lowDist}(v_i, v_m)$. The distance interval from node v_i to node v_j is $\mathrm{Dist}(v_i, v_j) = [\mathrm{lowDist}(v_i, v_m), \mathrm{upDist}(v_i, v_j)]$.

Lemma 19 (conditional distance interval). *Given nodes v_i and v_j, and $v_i \in G_i, v_j \in G_j$, and $v_k \in B(G_j)$, through node v_k, the conditional distance interval from node v_i to node v_j is as follows:*

$$\begin{aligned} lowDist\left((v_i, v_j) \mid v_k\right)_d &= \min\left\{Dist\left(v_i, B\left(G_j\right)\right)\right\} \\ &\quad + Dist\left(v_k, v_j\right), upDist\left((v_i, v_j) \mid v_k\right) \\ &= Dist\left(v_i, v_k\right) + Dist\left(v_k, v_j\right), \end{aligned} \quad (7)$$

and thus, $Dist((v_i, v_j) \mid v_k) = [lowDist((v_i, v_j) \mid v_k), upDist((v_i, v_j) \mid v_k)]$.

For example, to solve the distance interval between the node in G_5 through the border node v_{25} to v_{31}, that is, $\mathrm{Dist}((B(G_5), v_{31}) \mid v_{25})$, the distances to the border nodes between G_5 and G_6 are showed in Table 7.

$v_{31} \in G_6, v_{25} \in G_6$, according to Lemma 8, the distance from node to node v_{31} in subgraph G_5 must pass through the border node of G_5. $B(G_5) = \{v_{11}, v_{13}, v_{26}\}, B(G_6) = \{v_9, v_{10}, v_{23}, v_{25}\}, \mathrm{SPDist}(v_{25}, v_{31}) = 2$. The distances of the G_5 border node $B(G_5) = \{v_{11}, v_{13}, v_{26}\}$ to node v_{25} are $\mathrm{SPDist}(v_{11}, v_{25}) = 11, \mathrm{SPDist}(v_{13}, v_{25}) = 17$, and $\mathrm{SPDist}(v_{26}, v_{25}) = 10$. $\mathrm{SPDist}(v_{11}, v_9) = 6, \mathrm{SPDist}(v_{11}, v_{10}) = 5, \mathrm{SPDist}(v_{11}, v_{23}) = 24$, and $\mathrm{SPDist}(v_{11}, v_{25}) = 11$. Therefore, the distance of border nodes v_{11}, v_{13}, and v_{26} to v_{31} through v_{25} is

$$\begin{aligned} \mathrm{Dist}((v_{11}, v_{31}) \mid v_{25})_d &= \min\{\mathrm{Dist}(v_i, v_k)\} + \mathrm{Dist}(v_k, v_j) \\ &= \mathrm{SPDist}(v_{11}, v_{10}) + \mathrm{SPDist}(v_{25}, v_{31}) = 7, \\ \mathrm{Dist}((v_{11}, v_{31}) \mid v_{25})_u &= \mathrm{Dist}(v_i, v_k) + \mathrm{Dist}(v_k, v_j) \\ &= \mathrm{SPDist}(v_{11}, v_{25}) + \mathrm{SPDist}(v_{25}, v_{31}) = 13. \end{aligned} \quad (8)$$

$\mathrm{Dist}((v_{11}, v_{31}) \mid v_{25}) = [7, 13]$, expressed. The lower bound of the distance of v_{11} through node v_{25} to v_{31} is 7, and the upper bound is 13. Similarly, $\mathrm{Dist}((v_{13}, v_{31}) \mid v_{25}) = [13, 19]$, and $\mathrm{Dist}((v_{26}, v_{31}) \mid v_{25}) = (12)$.

Definition 20 (minimum value of subgraph). Given node v_i, $v_i \in G_i$, and subgraph G_j, the minimum distance between node v_i and $B(G_j)$ is called the minimum value of the subgraph and is denoted as $\mathrm{minDist}(v_i, G_j)$.

Corollary 21. *Given the subgraphs G_i and G_j, where $i \neq j$, then $minDist(G_i, G_j) = \min\{Dist(B(G_i), B(G_j)\}$.*

Proof. We know from Corollary 9 that the distance from node v_i outside the subgraph to the subgraph G_j must pass through the border node $B(G_j)$ of the subgraph G_j. Similarly, the distance between the node of the subgraph G_j and the subgraph G_i must pass through the border node $B(G_i)$ of the subgraph G_i. Therefore, $\mathrm{minDist}(G_i, G_j) = \min\{\mathrm{Dist}(B(G_i), B(G_j)\}$, which can be obtained from the node distance set. The proof is complete. □

For example, from Table 7, it can be seen that $\mathrm{minDist}(G_5, G_6) = \mathrm{Dist}(v_5, v_{10}\} = 5$; that is, the shortest distance from subgraph G_5 to subgraph G_6 is 5.

For distance interval values with the same starting node and ending node, the upper bound takes a smaller value, and the lower bound takes a more considerable value.

Theorem 22. *Given that the nodes $v_i \in G_i, v_j \in G_j$, and $v_k \in B(G_k)$, and $G_i^p = G_k^p$, and $Dist(v_k, v_j) = K$, for the first determined value node v_k of the heap H, if $lowDist((v_i, v_j) \mid v_k) = upDist((v_i, v_j) \mid v_k)$, that is, $Dist((v_i, v_j) \mid v_k)$ is a certain value, then $Dist((v_i, v_j) \mid v_k)$ is the shortest distance through node v_k. Moreover, node v_i does not need to be expanded in subsequent node expansion.*

Proof. Suppose that there are nodes v_i', v_j, and $v_k, v_i' \in G_i$, $v_j \in G_j$, and $v_k \in B(G_k)$. $\mathrm{Dist}((v_i', v_j) \mid v_k)$ is the smallest value because $\mathrm{lowDist}((v_i', v_j) \mid v_k) = \min\{\mathrm{Dist}(v_i', B(G_j))\} + \mathrm{Dist}(v_k, v_j)$. Only $\mathrm{Dist}(v_i', B(G_j))$ takes the minimum value, so $\mathrm{Dist}(v_i', v_k)$ takes the minimum value, because $\mathrm{Dist}((v_i, v_j) \mid v_k)$ takes the minimum value. Similarly, $\mathrm{Dist}(v_i, v_k)$

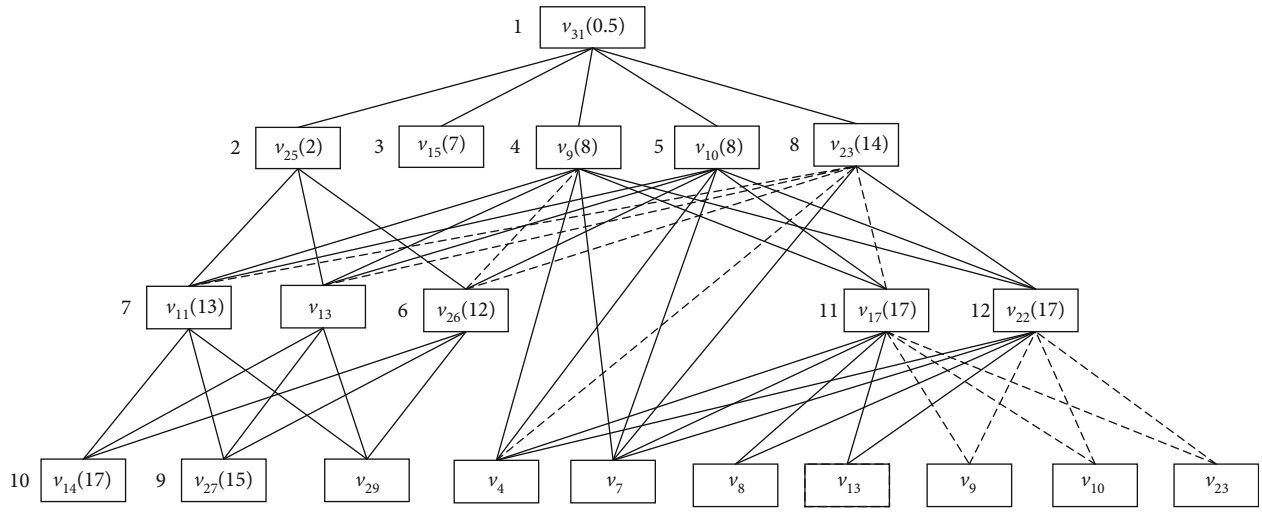


Figure 11: The expansion process of query $(q, 5)$.

takes the minimum value; that is, v_i makes $\mathrm{Dist}(v_i, B(G_j))$ take the minimum value. Since v_i and $v_i{}'$ are both in the same subgraph G_i, the uniqueness of the node shows that the assumption that there is node $v_i{}'$ in the G_i is wrong; that is, the value of $\mathrm{Dist}((v_i, v_j) \mid v_k)$ passes through the shortest distance of node v_k. Furthermore, to reduce the redundant calculations in node expansion, the minimum node does not need to be expanded repeatedly. □

Figure 11 shows the query process for $\mathrm{Query}(q, 5)$. The number, which lies on the left of the box, means the query processing steps. The solid lines represent the q expansion process using the distance interval method, and the dotted lines represent the calculation amount using the expansion method compared to the standard calculation method, which is not using the expansion method. There are 40 solid lines and 12 dotted lines; the calculation efficiency of the expansion method is increased by $12/(40 + 12) = 23\%$.

5.4. Tree-Based k-NN Search Algorithm

Definition 23 (adjacent subgraph). Take two subgraphs G_i and G_j, and $G_i \neq G_j$, $B(G_i) = \{v_i\}$, and $B(G_j) = \{v_j\}$. G_i and G_j have a common ancestor, i.e., $G_i{}^p = G_j{}^p$. We call G_i and G_j adjacent subgraphs. The adjacent subgraph of border node v_i is G_j and is denoted by $\mathrm{AdSub}(v_i) = \{G_j\}$.

For example, in Figure 6, v_{23} is the border node of G_6, and $v_{23} \in G_6$, $G_6{}^p = G_2$, and $G_5{}^p = G_2$. Therefore, G_5 and G_6 are adjacent subgraphs, and the adjacent subgraph of v_{23} is G_5. Furthermore, because $v_{23} \in G_2$, and $G_2{}^p = G_1{}^p = G_0$, G_1 is the v_{23} adjacency subgraph; that is, the adjacency subgraph of v_{23} is G_1. This is then denoted by $\mathrm{AdSub}(v_{23}) = \{G_5, G_1\}$.

Definition 24 (road network k-NN query). Given a directed weighted graph $G(V, E, W)$, M is the moving object set, and $M = \{m_1, m_2, m_3, \cdots, m_k\}$. The result of the query set is R. In the road network environment, the k-NN query at time t is expressed as $k\text{-NN} = (q, k, t)$, where k is a natural number and q is the query point. These variables are expressed as follows:

$$\begin{aligned} &|R| = k, \\ &R \subset M, \\ &\forall x \in R, \forall y \in M - R, \mathrm{SPDist}(q, x) \leq \mathrm{SPDist}(q, y). \end{aligned} \tag{9}$$

When giving a query point q, we query the k-nearest moving objects to q. First, we determine the subgraph G_i in which q is located. The subgraph contains the number of moving objects M_N and the starting node v_i of the edge. According to the node distance set $\mathrm{DisSet}(D_i)$, we then find the distance between v_i and the end node v_e where each moving object is located. Next, we sort the items in heap H from small to large and process the first element of the heap.

For example, when given the query point q, $k = 3$ means that q is in the subgraph G_6, and G_6 contains 5 moving objects. The starting node of its edge is v_{31}, and we need to find the 3 moving objects closest to v_{31}.

Here, we introduce the calculation for the k-NN for moving objects on the road network. In the existing k-NN algorithm for moving objects on the road network, there are redundant calculations. For this reason, we propose the DCDP-based moving object k-NN (DMOK) algorithm.

In the road network environment, for a given query point $q(v_s, q.\mathrm{offset})$, the k moving objects closest to the query point q are found, namely, $\mathrm{Query}(q, k)$. Figure 12 shows the processing of $\mathrm{Query}(q, 4)$. The specific calculation process is as follows:

(1) Determine the subgraph G_i of the starting node v_s of q

(2) Calculate the distance between the border node and the active node to v_s in G_i, and add the distance to H_{Ac}. Calculate the distance from the inactive node in G_i to v_s, and add the distance to H_{InAc}

Expansion of this node

Processing node	The border point and the vertex of the moving object	Expansion of this node	H_{Ac}	H_{InAc}	R
$v_{31} \in G_6$		$SPDist(v_9, v_{31}) = 8$ $SPDist(v_{10}, v_{31}) = 8$ $SPDist(v_{15}, v_{31}) = 7$ $SPDist(v_{25}, v_{31}) = 2$ $SPDist(v_{23}, v_{31}) = 14$ $SPDist(v_{24}, v_{31}) = 6$ $SPDist(v_{32}, v_{31}) = 12$	$\boxed{v_{25}(2) \in G_6}$ $v_{15}(7) \in G_6$ $v_9(8) \in G_2$ $v_{10}(8) \in G_2$ $v_{23}(14) \in G_2$	$v_{24}(6) \in G_6$ $v_{32}(12) \in G_2$	
$v_{25}(2) \in G_6$	$G_5(v_{11}, v_{13}, v_{26})$ $V_6(v_{25})$	$Dist((v_{11}, v_{31})\|v_{25}) = [7,13]$ $Dist((v_{13}, v_{31})\|v_{25}) = [13,19]$ $Dist((v_{26}, v_{31})\|v_{25}) = 12$	$\boxed{v_{15} \to v_{31}(7)}$ $v_{11} \to v_{31}[7, 13]$ …… $v_{23} \to v_{31}(14)$	$v_{24}(6) \in G_6$ $v_{32}(12) \in G_2$	$m_{25}(5)$
$v_{15}(7) \in V_6$	$V_6(v_{15})$		$\boxed{v_9 \to v_{31}(8)}$ $v_{10} \to v_{31}(8)$ …… $v_{13} \to v_{31}[18, 19]$	$v_{24}(6) \in G_6$ $v_{32}(12) \in G_2$	$m_{25}(5)$, $m_{21}(7.3)$
……	……	……	……	……	……
$v_{26}(12) \in G_5$	$V_5(v_{14}, v_{29})$ $N_5(v_{12}, v_{27}, v_{28})$	$Dist((v_{14}, v_{31})\|v_{26}) = [14,19]$ $Dist((v_{29}, v_{31})\|v_{26}) = [16,21]$ …… $Dist((v_{28}, v_{31})\|v_{26}) = 18$	$\boxed{v_{11} \to v_{31}(13)}$ $v_4 \to v_{31}[14, 20]$ …… $v_{13} \to v_{31}(19)$	$v_{24} \to v_{31}(6)$ $v_{32} \to v_{31}(12)$ …… $v_{28} \to v_{31}(18)$	$m_{25}(5)$, $m_{21}(7.3)$, $m_3(12)$, $m_{20}(13)$
$v_{11}(13) \in G_5$	$V_5(v_{11}, v_{14}, v_{29})$ $N_5(v_{12})$	$Dist((v_{14}, v_{31})\|v_{11}) = [15,17]$ $Dist((v_{29}, v_{31})\|v_{11}) = [17,19]$ $Dist((v_{12}, v_{31})\|v_{11}) = [17,23]$	$\boxed{v_{23} \to v_{31}(14)}$ $v_4 \to v_{31}[14, 20]$ …… $v_{13} \to v_{31}(19)$	$v_{24} \to v_{31}(6)$ $v_{32} \to v_{31}(12)$ …… $v_{28} \to v_{31}(18)$	$m_{25}(5)$, $m_{21}(7.3)$, $m_3(12)$, $m_{20}(13)$, $m_{19}(14.5)$
$v_{23}(14) \in G_2$	$SPDist(v_{14}, v_{31}) = 14 \geq \text{Dth} = SPDist\ (m_{20}, v_{31}) = 13$				$m_{25}(5)$, $m_{21}(7.3)$, $m_3(12)$, $m_{20}(13)$

Figure 12: Processing of Query $(q, 4)$.

(3) Expand according to the expansion strategy. According to the first element V_{cur} of heap H_{Ac}, if V_{cur} is the border node, calculate the distance between the border node of the adjacent subgraph and v_s, and add the node to H_{Ac} (whose elements are sorted from smallest to largest). If V_{cur} is the active node, calculate the distance between the moving object m and v_s, and add m to the moving object candidate set Cset

(4) According to the order of heap H_{Ac}, the first element V_{cur} of the heap is processed. When the number of moving object candidate set elements Cset.size $\geq k$, take the distance between the kth m and v_s in the moving object candidate set as the moving object distance threshold Dth, update Dth. When Dist $(V_{cur}, v_s) \geq$ Dth, the processing of heap H_{Ac} ends

(5) Output the result set R of moving objects

According to the node expansion strategy, and combined with the updating model, the set of moving objects that meet the query criteria can be quickly determined. Algorithm 3 shows the specific process.

For example, the process of solving query$(q, 4)$ according to Algorithm 3 is as follows:

(1) Query point is $q(V_S, q.\text{offset})$, $V_s = v_{31}$, and the offset from the starting node v_{31} is $q.\text{offset} = 0.5$

(2) $q \in G_6$, and according to the moving object table, the number of moving objects in the subgraph G_6 is 3 because the number of the moving objects $M_N = 3 < k = 4$; it needs to further expand to adjacent subgraphs. Determine the distance from the end node V_e of the moving object in the subgraph G_6 to the start node v_{31} of the query point q, calculate the distance from the border node of G_6 to v_{31}, and add nodes to H_{Ac}. In the same way, add the node from the inactive node in the subgraph G_6 to the starting node v_{31} of the query point q and add them to the heap H_{InAc}

(3) Expand according to the node expansion strategy. When the distance between the same two nodes is an interval, the lower bound of the distance interval takes a larger value, and the upper bound takes a smaller value. At the same time, the nodes in the

Input: $q(v_s, q.offset)$
Output: moving object result set R

Determine the starting node v_s of q and its subgraph;
$V_{cur} = \varnothing$; $Dth = 0$; $C\,set = \varnothing$;
Calculate the distance between v_s and its border nodes and active nodes in the subgraph, and add these distances to the H_{Ac} i order from small to large, calculate the distance between v_s and the inactive nodes in the subgraph, and add these distances to the heap of H_{InAc} in order from small to large;
while H_{Ac} *is not empty* **do**
  Take the first element V_{cur} of H_{Ac};
  if V_{cur} *is the border point* **then**
    if V_{cur} and v_s *are not in the same subgraph* **then**
      Calculate the distance between the active node of the subgraph where V_{cur} is located and v_s, and add it to the heap H_{Ac}
    if $Cset.size < k$ **then**
      Calculate the distance between the border node of the V_{cur} adjacency subgraph and v_s, and add it to the heap H_{Ac}
    else
      According to the expansion strategy, the nodes that meet the conditions are added to the H_{Ac}
  else if V_{cur} is the active node **then**
    Obtain the set M of moving objects corresponding to V_{cur} through the Moving Object List and sort them in descending order of the distance from V_{cur};
    for *Take each moving object* $m \in M$ *and process if* **do**
      Calculate **Dist**(m, v_s);
      if $Cset, size < k$ **then**
        Continue to Line 10;
      else
        if *Dth value is empty* **then**
          Determine the initial threshold Dth;
        else
          if Dist$(m, v_s) < Dth$ **then**
            Use m and **Dist**(m, v_s) to update $Cset$ and threshold Dth respectively;
          else
            The algorithm ends;
Output $R = Cset$

Algorithm 3: DMOK algorithm().

heap H_{Ac} follow the upper bound of the distance interval and are sorted in ascending order

(4) When the candidate set of moving objects Cset.size = 4, use the maximum value in Cset at this time as the distance threshold Dth = 13

(5) When the first element V_{cur} of heap H_{Ac}, the distance from v_{11} to v_{31}, $\mathrm{Dist}(v_{11}, v_{31}) = 13$ is less than or equal to Dth = 13, continue to expand the node for the first element V_{cur} of the heap H_{Ac}, and update the distance threshold Dth. When the distance from the first element in H_{Ac} to v_{31} $\mathrm{Dist}(v_{11}, v_{31}) \geq$ Dth, the node expansion ends

(6) Output the result set of moving objects $R = \{m_{25}, m_{21}, m_{22}, m_4\}$, and terminate the algorithm

6. Results and Discussion

The experimental environment is configured as a PC with Intel Core i7-6700 CPU @ 3.40 GHz, with 16 GB memory, 2 T hard disk, and Windows 10 operating system. The algorithm is written in *C++* language.

The data set used in this experiment is a real-world road network from an urban road network [39]. In order to verify the effectiveness of the algorithm proposed in this paper, we selected four data sets of them for experimental comparison. The detailed information is shown in Table 8.

For V-tree and $G*$-tree ($G*$-tree means the algorithm in the G-tree paper [25]), we set $f = 4$ and $\tau = 32$. We evaluated the k-NN search efficiency of DMOK and used TEN-Index, V-tree, and G^*-tree as baselines. We vary k as 10, 20, 50, and 100. We present the search time of the four

Table 8: Road network data statistics.

Data set	$\|V\|$	$\|E\|$
Pune (PN)	28649	36925
Baghdad (BD)	60108	88876
Tehran (TR)	110580	147339
Bangkok (BK)	154352	187798

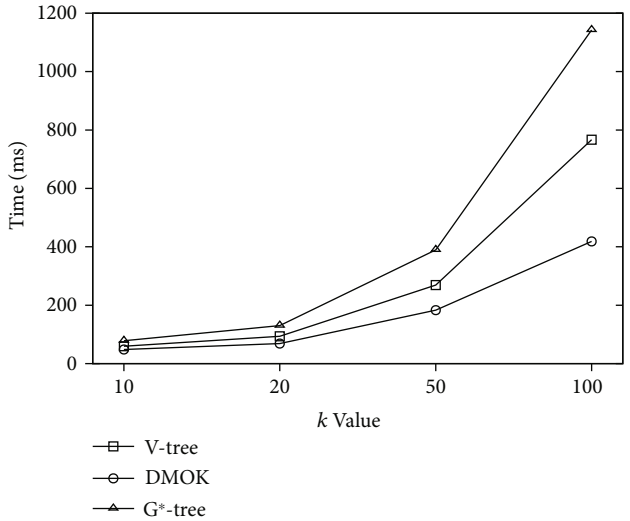


Figure 13: Comparison of various algorithms under different k values.

algorithms on the BK data set. The abscissa is the variation range of k.

As can be seen from Figure 13, for different values of k, the search time of our algorithms is the least compared with the other two algorithms; this is due to the node processing strategy proposed in this algorithm. Because the TEN-Index can only query k-NN within the preset k value of the index, the query cannot be processed when the k value of k-NN is greater than the preset k value; the query is not compared to it in this set of experiments.

In Figures 14 and 15, we compared the DTH index construction time and space consumption with V-tree, TEN-Index, and G^*-tree on the four data sets. Object density θ = candidate nodes/total number of vertices. In the experiment, $k = 10$, and $\theta = 0.3$. We randomly generate 1000 vertices as query points and take the average total query time as the final query time. From the histogram, we can clearly see that building the DTH index proposed in the paper takes less time than other indices. Conversely, the building index times of V-tree and $G*$-tree are high; the main reason for this is that they set up an index of structural information such as nodes, edges, active nodes, and moving objects. Meanwhile, our algorithm constructs indices separately for road network information and moving objects. When moving objects are updated, we only need to update locally, which saves a lot of time. TEN-Index built the index the fastest because it uses H2H-Index for the shortest distance calculation of the road networks, unlike the other algorithms that use Dijkstra. The disadvantage of TEN-Index is that it depends on the

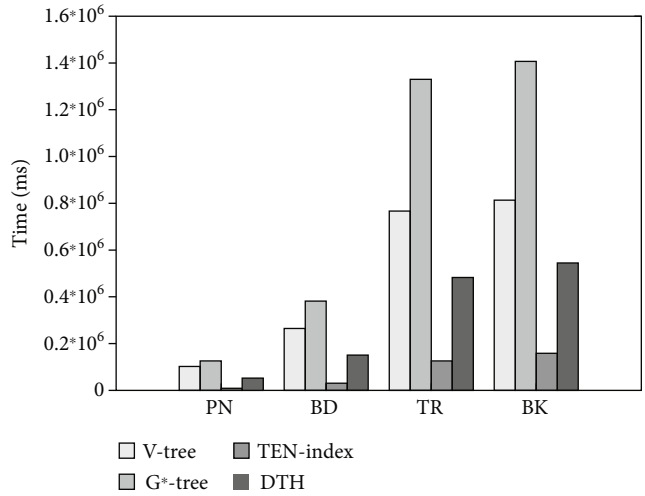


Figure 14: Comparison of index construction time.

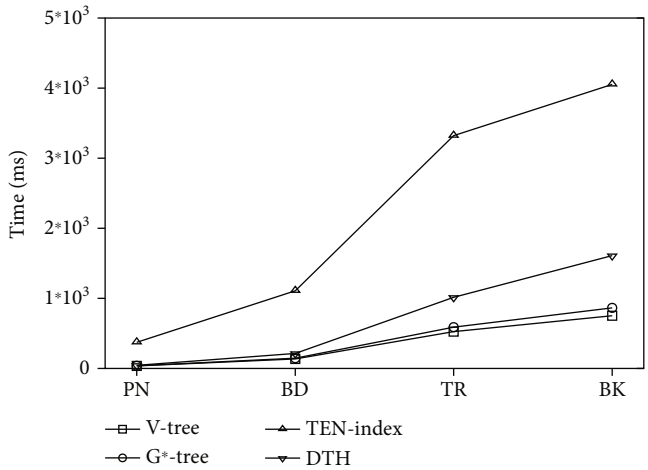


Figure 15: Index construction space.

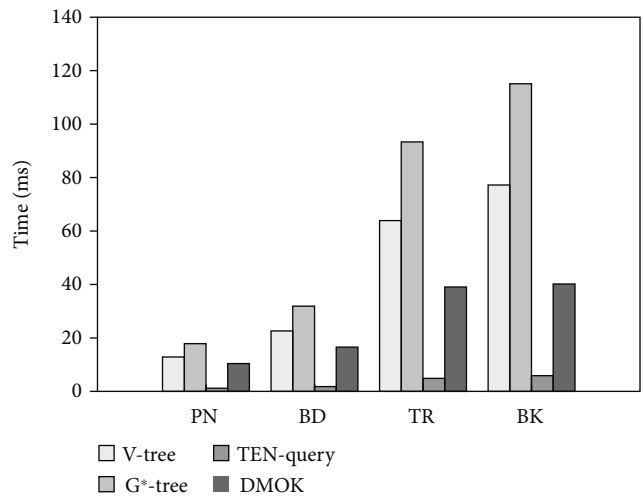


Figure 16: Comparison of search time.

value of k; that is, the value of k needs to be specified when building the index, and only queries within the value of k can be processed. Therefore, when the value of k is fixed, TEN-Index is relatively fast. In addition, its constructed index consumes much space and needs to be stored during tree decomposition. Considering a large amount of information, the index structure proposed in this article has certain advantages.

It can be seen from Figure 16 that the TEN-Index algorithm has weak adaptability to the value k, and the method proposed in this paper has advantages over the other algorithms in terms of search time. The reason for this is that in the initial stage of index construction, although a certain amount of space consumption is sacrificed, in the latter search, the proposed algorithm dramatically improves the search time due to its node strategy and avoidance of redundant calculations.

7. Conclusions

This paper studies a fast tree-indexed k-NN search algorithm of moving objects on a road network. Firstly, a novel index structure, called the double tree-hash index, is designed to reorganize the relationships between moving objects and the road network. The DTH index builds a weak bridge between tree and hash structures. The tree structure encapsulates the nodes, edges, and quantized relationships of intersubgraphs. The hash index records the moving objects. Through the weak bridge, the DTH index can realize activity-oriented updates of moving objects in the road network. Secondly, an index-enhanced k-NN search algorithm is proposed to quickly find the k-nearest neighbors of moving objects on the road network. This algorithm is called the DMOK algorithm. An updating model is established to quickly update the moving object and effectively expand the positive nodes. Through the conditional distance rules, the number of expanded negative nodes can be reduced. Then, multiple answers are provided to queries based on the DTH index. Finally, experiments with real data sets and artificial synthetic data sets verify the algorithm's effectiveness.

Data Availability

The data used to support the findings of this study have been deposited in the Road Networks, Urban (2016): Urban Road Network Data. figshare. Dataset. https://doi.org/10.6084/m9.figshare.2061897.v1.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61602076, Grant 61702072, Grant 62002039, and Grant 61976032; in part by the China Postdoctoral Science Foundation funded projects under Grant 2017M611211, Grant 2017M621122, and Grant 2019M661077; in part by the Natural Science Foundation of Liaoning Province under Grant 20180540003; and in part by the CERNET Innovation Project under Grant NGII20190902.

References

[1] N. Saeed, H. Nam, T. Y. al-Naffouri, and M. S. Alouini, "A state-of-the-art survey on multidimensional scaling-based localization techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3565–3583, 2019.

[2] J. Q. Xu, R. H. Güting, Y. Zheng, and O. Ouri Wolfson, "Moving Objects with Transportation Modes: A Survey," *Journal of Computer Science and Technology*, vol. 34, no. 4, pp. 709–726, 2019.

[3] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Transactions on Database Systems*, vol. 24, no. 2, pp. 265–318, 1999.

[4] M. Sharifzadeh and C. Shahabi, "Vor-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1231–1242, 2010.

[5] S. Ji and C. Li, "Location-based instant search," in *Scientific and Statistical Database Management. SSDBM 2011*, J. Bayard Cushing, J. French, and S. Bowers, Eds., vol. 6809 of Lecture Notes in Computer Science, pp. 17–36, Springer, Berlin, Heidelberg, 2011.

[6] K. C. K. Lee, W. C. Lee, and B. Zheng, "Fast object search on road networks," in *Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology - EDBT '09*, pp. 1018–1029, New York, NY, USA, 2009.

[7] J. Sankaranarayanan, H. Alborzi, and H. Samet, "Efficient query processing on spatial networks," in *Proceedings of the 2005 international workshop on Geographic information systems - GIS '05*, pp. 200–209, New York, NY, USA, 2005.

[8] S. Rogers, P. Langley, and C. Wilson, "Mining GPS data to augment road models," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*, pp. 104–113, San Diego, CA, USA, 1999.

[9] S. Berchtold, B. Ertl, D. A. Keim et al., "Fast nearest neighbor search in high-dimensional space," in *Proceedings 14th International Conference on Data Engineering*, pp. 209–218, Orlando, FL, USA, 1998.

[10] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, pp. 43–54, New York, NY, USA, 2008.

[11] H. J. Cho and R. Jin, "Efficient processing of moving k-range nearest neighbor queries in directed and dynamic spatial networks," *Mobile Information Systems*, vol. 2016, Article ID 2406142, 17 pages, 2016.

[12] H. J. Cho, R. Jin, and T. S. Chung, "A collaborative approach to moving-nearest neighbor queries in directed and dynamic road networks," *Pervasive and Mobile Computing*, vol. 17, Part A, pp. 139–156, 2015.

[13] G. Li, P. Fan, Y. Li, and J. Du, "An efficient technique for continuous k-nearest neighbor query processing on moving

objects in a road network," in *2010 10th IEEE International Conference on Computer and Information Technology*, pp. 627–634, Bradford, UK, 2010.

[14] Z. Yu, Y. Liu, X. Yu, and K. Q. Pu, "Scalable distributed processing of k nearest neighbor queries over moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1383–1396, 2015.

[15] L. Heendaliya, M. Wisely, D. Lin, S. S. Sarvestani, and A. Hurson, "Indexing and querying techniques for moving objects in both euclidean space and road network[M]," in *Advances in Computers*, vol. 102, pp. 111–170, Springer, 2016.

[16] D. He, S. Wang, X. Zhou, and R. Cheng, "An efficient framework for correctness-aware kNN queries on road networks," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1298–1309, Macao, China, 2019.

[17] B. Shen, Y. Zhao, G. Li et al., "V-tree: efficient knn search on moving objects with road-network constraints," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, San Diego, CA, USA, 2017.

[18] D. Ouyang, D. Wen, L. Qin, L. Chang, Y. Zhang, and X. Lin, "Progressive top-K nearest neighbors search in large road networks," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 1781–1795, New York, NY, USA, 2020.

[19] M. B. Wells and E. Donald, "Book review: the art of computer programming, volume 1. Fundamental algorithms and volume 2. Seminumerical algorithms," *Bulletin of the American Mathematical Society*, vol. 79, no. 3, pp. 501–510, 1973.

[20] A. Mahmood and W. G. Aref, "Query processing techniques for big spatial-keyword data," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1777–1782, New York, NY, USA, 2017.

[21] M. Kolahdouzan and C. Shahabi, "Voronoi-based K nearest search in spatial network databases," in *Proceedings of the 30th International Conference on Very Large DataBases*, pp. 840–851, Canada, Toronto, 2004.

[22] P. Zhang, H. Lin, Y. Gao, and D. Lu, "Aggregate keyword nearest neighbor queries on road networks," *GeoInformatica*, vol. 22, no. 2, pp. 237–268, 2018.

[23] X. Huang, C. S. Jensen, H. Lu, and S. Šaltenis, "S-GRID: a versatile approach to efficient query processing in spatial networks," in *Advances in Spatial and Temporal Databases. SSTD 2007*, D. Papadias, D. Zhang, and G. Kollios, Eds., vol. 4605 of Lecture Notes in Computer Science, pp. 93–111, Springer, Berlin, Heidelberg, 2007.

[24] K. C. K. Lee, W. C. Lee, B. Zheng, and Y. Tian, "ROAD: a new spatial object search framework for road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 3, pp. 547–560, 2012.

[25] R. Zhong, G. Li, K. L. Tan, L. Zhou, and Z. Gong, "G-tree: an efficient and scalable index for spatial search on road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2175–2189, 2015.

[26] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data - SIGMOD '95*, pp. 71–79, New York, NY, USA, 1995.

[27] Y. Liu, G. Liu, and Z. He, "Spatial index technology for multiscale and large scale spatial data," in *2010 18th International Conference on Geoinformatics*, pp. 1–4, Beijing, China, 2010.

[28] E. Frentzos, "Indexing objects moving on fixed networks," in *Advances in Spatial and Temporal Databases. SSTD 2003*, T. Hadzilacos, Y. Manolopoulos, J. Roddick, and Y. Theodoridis, Eds., vol. 2750 of Lecture Notes in Computer Science, pp. 289–305, Springer, Berlin, Heidelberg, 2003.

[29] V. T. de Almeida and R. H. Güting, "Indexing the trajectories of moving objects in networks," *GeoInformatica*, vol. 9, no. 1, pp. 33–60, 2005.

[30] N. Du, J. Zhan, M. Zhao, D. Xiao, and Y. Xie, "Spatio-temporal data index model of moving objects on fixed networks using hbase," in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, pp. 247–251, Ghaziabad, India, 2015.

[31] L. Liu, W. Li, Y. Guo, and J. Le, "Supporting high updates disk-based index in road network," in *2008 IEEE International Conference on e-Business Engineering*, pp. 517–522, Xi'an, China, 2008.

[32] Y. Theodoridis, M. Vazirgiannis, and T. Sellis, "Spatio-temporal indexing for large multimedia applications," in *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pp. 441–448, Hiroshima, Japan, 1996.

[33] I. Kamel and C. Faloutsos, *Hilbert R-Tree: An Improved R-Tree Using Fractals*, Institute for Systems Research Technical Reports, DRUM, 1993.

[34] S. T. Leutenegger, M. A. Lopez, and J. Edgington, "STR: a simple and efficient algorithm for R-tree packing," in *Proceedings 13th International Conference on Data Engineering*, pp. 497–506, Birmingham, UK, 1997.

[35] S. Luo, B. Kao, G. Li, J. Hu, R. Cheng, and Y. Zheng, "TOAIN: a throughput optimizing adaptive index for answering dynamic k-nn queries on road networks," *Proceedings of the VLDB Endowment*, vol. 11, no. 5, pp. 594–606, 2018.

[36] D. Tianyang, Y. Lulu, C. Qiang, C. Bin, and F. Jing, "Direction-aware KNN queries for moving objects in a road network," *World Wide Web*, vol. 22, no. 4, pp. 1765–1797, 2019.

[37] R. H. Güting, V. T. De Almeida, and Z. Ding, "Modeling and querying moving objects in networks," *The VLDB Journal*, vol. 15, no. 2, pp. 165–190, 2006.

[38] "V-tree: efficient knn search on moving objects with road-network constraintsTsinghua Technical Reporthttp://dbgroup.cs.tsinghua.edu.cn/ligl/v-tree.pdf.

[39] https://figshare.com/articles/dataset/Urban_Road_Network_Data/2061897/1.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8402138, 13 pages
https://doi.org/10.1155/2021/8402138

Research Article

Lightweight Privacy-Preserving Data Sharing Scheme for Internet of Medical Things

Zhuo Zhao,[1] Chingfang Hsu,[1] Lein Harn,[2] Qing Yang,[1] and Lulu Ke[1]

[1]*Computer School, Central China Normal University, Wuhan 430079, China*
[2]*Department of Computer Science Electrical Engineering, University of Missouri-Kansas City, Kansas City, 64110 MO, USA*

Correspondence should be addressed to Chingfang Hsu; cherryjingfang@gmail.com

Received 16 June 2021; Accepted 30 August 2021; Published 13 September 2021

Academic Editor: Pengfei Wang

Internet of Medical Things (IoMT) is a kind of Internet of Things (IoT) that includes patients and medical sensors. Patients can share real-time medical data collected in IoMT with medical professionals. This enables medical professionals to provide patients with efficient medical services. Due to the high efficiency of cloud computing, patients prefer to share gathering medical information using cloud servers. However, sharing medical data on the cloud server will cause security issues, because these data involve the privacy of patients. Although recently many researchers have designed data sharing schemes in medical domain for security purpose, most of them cannot guarantee the anonymity of patients and provide access control for shared health data, and further, they are not lightweight enough for IoMT. Due to these security and efficiency issues, a novel lightweight privacy-preserving data sharing scheme is constructed in this paper for IoMT. This scheme can achieve the anonymity of patients and access control of shared medical data. At the same time, it satisfies all described security features. In addition, this scheme can achieve lightweight computations by using elliptic curve cryptography (ECC), XOR operations, and hash function. Furthermore, performance evaluation demonstrates that the proposed scheme takes less computation cost through comparison with similar solutions. Therefore, it is fairly an attractive solution for efficient and secure data sharing in IoMT.

1. Introduction

Internet of Things (IoT) is a system, which connects different sorts of sensors and computing devices using network to gather and share medical data. IoT lets devices become smarter, processing becomes intelligent, and communication becomes informative [1]. IoT has bred kinds of new technology solutions used in many disparate domains due to its convenience. Certainly, IoT has also penetrated into the healthcare system and has brought great changes. Internet of Medical Things (IoMT) is substantially IoT devices applied to medical industry [2]. The application of IoMT brings lots of conveniences to patients and medical professionals. For example, in IoMT, medical professionals can receive the data and information they need and provide telemedicine for patients anywhere [3].

IoMT provides continuous health monitoring. It relies on different sorts of physiological sensors that are placed on the patients without reducing the user's comfort to collect live health data and information, such as oxygen saturation rate, heart rate, pulse, temperature, blood pressure, and respiration [4–8]. Due to the sensibility of personal health data and information and the limited resources of sensors, it is crucial that security and lightweight computation are included as a fundamental element in IoMT [9]. Cloud computing is a kind of outsourcing platform that has large storage memory and computing resources. Due to its advantages, it can be combined with IoMT to eliminate the issues of storing large data. With the help of cloud computing servers, patients can efficiently store, manage, and share great amount of medical information. By storing data in the cloud, it can be providing easy access for users and improve storage utilization of the health information system [10]. However, the information of the patients (such as the identity of the patients, electronic medical records, and personal condition related to health) is highly private and vulnerable. Data breaches are harmful to patients as

the sensitive information will reveal patients' identity privacy and data security. Hence, the security of health data is the major concern for sharing schemes. Besides, the completeness of shared patients' health data is extremely important [11]. For example, if an adversary tampers patient's conditions related to health, it will mislead medical professionals into making faulty analyses and affect the patient's health. Therefore, integrity verification can prevent tampering by malicious attackers. Moreover, the scheme must provide authentication for users to verify users' legitimacy. This is due to the fact that unauthorized users may tamper with medical records; falsified data will lead to misdiagnosis by medical professionals [12]. Meanwhile, the physiological sensors, used in medical systems, have limited storage memory and power and low computation speed and bandwidth. Accordingly, this motivates us to design a low-cost and lightweight data sharing scheme applied to the IoMT, which consumes less power and meets higher security requirements.

Many researchers have devoted to designing effective data sharing schemes in cloud computing over the past few years. However, some [13–15] are not suitable to be deployed in IoMT system because of the use of bilinear pairings which lack efficiency. These heavy calculations with the high resource constraints are not lightweight enough. Analysis in [16] demonstrates that a bilinear pairing operation has very high computation cost. On the contrary, the computation complexity of elliptic curve cryptography (ECC) is several times smaller than that of pairing operation. This is because in the ECC algorithm, the arithmetic requirements are low, the key size is small, and the operand length is shorter. As a result, based on the previous discussion, ECC is regarded as a better encryption technology for resource-constrained devices.

Hence, for the purpose of ensuring the anonymity of patients, preserving shared data privacy, and improving the computation efficiency of physiological sensors in IoMT, this paper constructs a lightweight privacy-preserving data sharing scheme applied to the IoMT using ECC. In this scheme, after collecting the health data, patients with physiological sensors must encrypt collected health data to prevent personal privacy from leaking. Then, the patient generates a fake identity to protect his identity and achieve anonymity. With the help of a cloud server, health data can be shared with authorized users after uploading by patients. Furthermore, to realize the authorized access, patients should designate the identity set of users. Before accessing the health data, users must authenticate to the cloud server. Users are eligible to access encrypted health data only if their identities and access time are valid. Finally, the main contributions of this paper are summarized below.

(1) A lightweight privacy-preserving data sharing scheme for IoMT using ECC is proposed, which anonymizes the identity of patients and designs authorized access to shared health data

(2) The proposed scheme realizes lightweight computations by ECC, hash, and XOR operations, which does not require heavy computations such as bilinear pairings

(3) The proposed protocol can resist possible attacks and achieve all desired security features, including replay attack, eavesdropping attack, correctness, freshness of encryption key, authentication, anonymity of patient, integrity certification, and forward secrecy of encryption key

(4) Compared with the similar solutions, the proposed scheme satisfies all desired security features and achieves more lightweight computations on patients

The remaining of this paper is adjusted as follows. Previous studies are conducted in Section 2. The basic knowledge of mathematical preliminaries is introduced in Section 3. Then, Section 4 illustrates the model of the proposed scheme including the network model, types of attack, security properties, design goals, and syntax of the proposed scheme. This data sharing scheme including three phases, system initialization, data encryption and upload, and data sharing, is given in Section 5. The security verification of this scheme is provided in Section 6. The performance evaluation and the comparisons with similar schemes in terms of computation cost and security are presented in Section 7. Finally, we culminate conclusions of this paper in Section 8.

2. Related Work

Cloud computing has emerged as a convenient platform of sharing data that enables multiple users from different domains to obtain their needed information simultaneously. It is highly necessary to authenticate users who want to access the health data. However, it worth noting that existing solutions may suffer from a series of issues such as data owner privacy, completeness of the data, data access control, and computation cost in encryption/decryption. These issues have been of widespread concerns.

In 2010, Itani et al. [17] presented a lightweight protocol such that mobile clients can verify the completeness of storage information in mobile cloud computing. In 2013, Wang et al. [18] constructed a cloud storage system that can realize privacy protection, where users can use third-party auditor to verify the completeness of outsourced data. Later, in 2014, Wang et al. [19] presented a novel data integrity verification mechanism using ring signature that is able to ensure identity privacy. Yang et al. [20] designed a data sharing solution in cloud. This solution provided integrity verification while guarantying users' identity privacy. In order to achieve sensitive data concealing in data integrity certificating, Shen et al. [21] presented an efficient data sharing protocol in 2019.

Due to the limited storage of small devices, the large data needs to be outsourced. Outsourced data may contain private information, so ensuring data security has become a challenge. Some works focused on designing valid schemes for this issue. For example, Wang et al. [22] provided a processing mechanism to achieve a flexible user access control. However, this solution takes no account of the energy consumption due to data owner needs to share the pairwise keys with users, which consumes plenty of storage memory. Later, a novel certificateless proxy reencryption (CL-PRE)

scheme was presented by Xu et al. [23], which is used to share information in cloud server securely. This paper showed that the certificateless scheme can cut down the cost of computation and communication for data owners. Nevertheless, this scheme can consume a large amount of computation because of the use of bilinear pairing operation. Khan et al. [24] designed a proxy reencryption scheme for reducing the energy consumption and memory consumption, in which the computational complexity of bilinear pairing still remains. A cloud computing technology-based electronic health record system supporting data privacy preserving was presented in [25]. Ramesh et al. [26] proposed a secure model using e-stream cipher ChaCha20. This model provides integrity verification of sensitive data and guarantees the authenticity of the data. Wang et al. [27] constructed a system framework based on cloud for the electronic medical field. They had utilized identity-based encryption and proxy reencryption in this study for security purpose. This study also provided users authorized by the data owner with the right to access health information. He et al. [28] designed an encryption technology for wireless body area networks to check the completeness of the stored medical data that provides better performance.

A scheme for sharing personal health data and access control was designed by Jiang et al. [29]. This scheme is applied to mobile healthcare social networks, and it adopts attribute-based encryption as the main encryption method. Ding et al. [13] presented a health storage system to resolve data integrity verification, which provides convenience for the patient and physician safety communications. Sowjanya et al. [30] introduced an end-to-end authentication protocol. The protocol reduces the overall complexity due to the use of elliptic curve cryptography (ECC). Zhang et al. [31] presented a practical scheme for cloud-assisted electronic health information systems using identity-based encryption to enable the sensitive data sharing efficiently.

Most of the available schemes are not secure enough. In addition, some of the schemes use complex operations such as bilinear pairing, which make the calculation cost more and are not lightweight enough for IoMT. What is more, the anonymity of patients is often ignored by some schemes. As a result, to guarantee the anonymity of patients and provide access control for shared health data, we design a lightweight privacy-preserving data sharing scheme for IoMT that is based on ECC, hash, and XOR operations.

3. Preliminaries

The work of elliptic curve cryptosystem (ECC) was firstly put forth by Koblitz [32] and Miller [33] individually. ECC is a public key encryption technique. Elliptic curve is a kind of cubic curve over finite fields, which is based on the algebraic structure. ECC with the benefit of lightweight and high security has aroused widespread concerns in modern cryptography. 160-bit ECC key and 1024-bit RSA key can provide equivalent security, which leads to the fact that the encryption key generated by ECC is smaller and more efficient. An elliptic curve E is simply described by the equation $y^2 (\mathrm{mod}\, p) = x^3 + ax + b (\mathrm{mod}\, p)$, where p is a large prime number. In addition, $(4a^3 + 27b^2) \neq 0 (\mathrm{mod}\, p)$ needs to be satisfied in order to exclude singular elliptic curves. Z_p indicates a prime finite field and $a, b, x, y \in Z_p$. Then, we omit $(\mathrm{mod}\, p)$ for the sake of simplicity. The three operations of ECC over G_E are defined below.

(1) Point addition: given two random points, P and Q, on the elliptic curve E, the point R on E represents the addition of these two points. The formula is as follows: $P + Q + R = 0$. Here, R refers to the third point where the line connecting P and Q intersects the elliptic curve. And the point $-R$ is the reflection of point R on the x-axis

(2) Point doubling: it refers to the addition of a point on E with itself. The point Q represents the addition of a point P on the same curve E. The formula is as follows: $2P + Q = 0$. Here, the point $-Q$ is the reflection of point Q (point of intersection of tangent line at P with E) on the x-axis

(3) Scalar point multiplication: it means a point that repeatedly performs point doubling and point addition operations. Let $n \in Z_q^*$ be a positive integer and then $n \bullet P$ is given by $P + P + \cdots + P$ (n times)

There are two hard problems in the elliptic curve domain, which are widely used in designing encryption schemes because there is no probabilistic polynomial time algorithm that can effectively run on computer. The following computational hard problems over ECC [34] have been widely utilized for secure schemes.

Elliptic Curve Discrete Logarithm Problem (*ECDLP*): let $k \in Z_q^*$ be a positive integer, and let $P, Q \in G_1$ be two elliptic curve random points. The ECDLP is to determine k given P and Q, where $P = k \bullet Q$. It is obvious that knowing k and Q is easy to calculate P, but conversely, it is not feasible to calculate k by knowing P and Q, if the prime number q is large.

Elliptic Curve Computational Diffie-Hellman Problem (*ECCDHP*): the ECCDHP is stated as it is difficult for any random instance (B, $c \bullet B$, $d \bullet B$) to compute the value $c \bullet d \bullet B$, where B is the base point of the elliptic curve and $c, d \in Z_q^*$ are two positive integers.

4. Model of the Proposed Scheme

We first design a network model suitable for IoMT and a security model for the data sharing scheme in this section. And then, the types of attack and security properties and illustration of the design goals and the syntax of the proposed scheme are provided.

4.1. Network Model. A network model for IoMT is presented. It consists four types of entities, i.e., a trusted authority (TA), patients, cloud severs (CS), and users. Their relationship in the network model is shown in Figure 1.

(1) Trusted authority (TA): TA acts as a public and private secret generation system and is a fully trusted

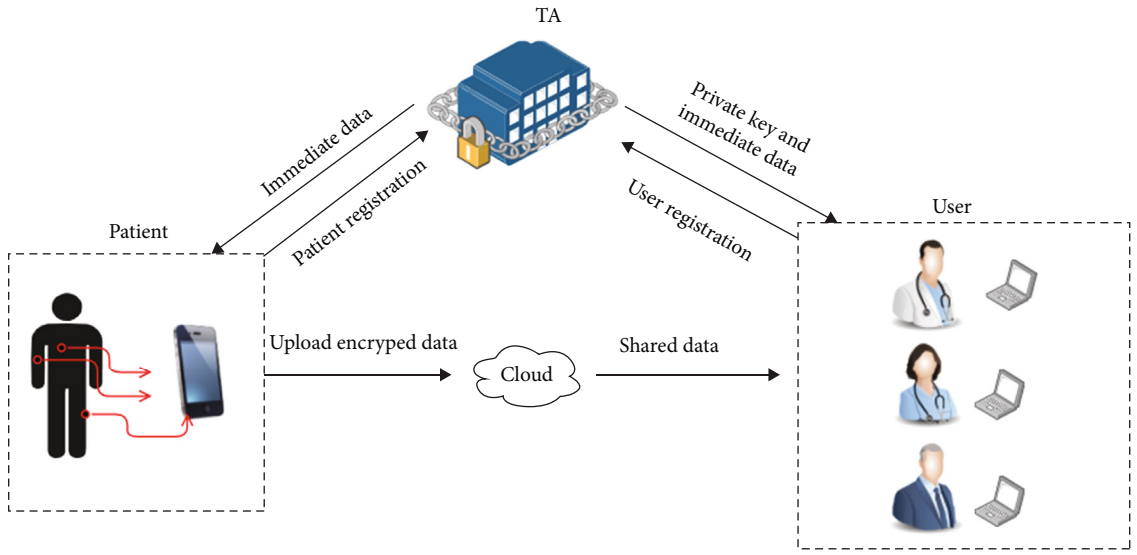


Figure 1: Proposed architecture for IoMT.

authority. In this scheme, system initialization is performed by TA. Patients and users must register with TA before receiving system services. In addition, TA could communicate with different entities via a secure channel. The fact that a secure channel exists does not mean that the data can be shared through the secure channel, due to shared data can be in a large amount

(2) Patient: it refers to data owners with physiological sensors. Patients gather personal health data through these physiological sensors. Patients must register with TA before accepting the service of system. And then, they can upload data to cloud server for storing and sharing health data with authenticated legitimate users due to their own limited memory. Since all shared data is uploaded to cloud server through a public channel, patients should encrypt the gathering information and hide identity to preserve personal privacy and health information security. Besides, his real identity is only known by TA and authorized users

(3) Cloud server (CS): CS is responsible for storing the encrypted information of patients and authenticating users who want to access data because it has a large storage memory and strong computing power. Besides, CS is considered as semitrusted. In other words, if the stored data is lost, it may fake the missing data to hide it from users for economic reasons

(4) User: this entity appertains to medical professional, who can communicate with CS to obtain patients' health information for medical analysis and diagnosis. Before accessing the health data, legitimate users should register with TA. In this scheme, it is important to note that only identified and authorized users can obtain the required health information from CS and decrypt the patients' encrypted data

Now, we will give the description of our proposed scheme. There are three main phases in the proposed sharing scheme, namely, (1) system initialization phase, (2) data encryption and upload phase, and (3) data sharing phase. The subphases of these phases are detailed below.

(1) Setup: trusted authority (TA) executes this phase for defining the system public parameters, choosing a unique nonce $S_{TA} \in Z_q^*$ as its own private key, and computing the public key PK_{TA}, separately

(2) User registration: this phase is processed by the TA. After TA receives the identity Uid sent by user, it generates the warrant of the user warr and private key sk_{Uid}. Further, TA sends (warr, sk_{Uid}) to the user via secret channel

(3) Patient registration: it is performed by the patient and the TA. Firstly, it is run by the patient for generating the temporary identity Ptid and choosing user identity set S and then sends them to TA. Secondly, it is run by the TA for checking the patient's Ptid and computes the intermediate result a_n for data encryption and then sends a_n to the patient via secret channel and S to CS

(4) Encryption: this phase is performed by patients and it encrypts sensitive data M to M'

(5) Upload: it is performed by the patient, by sending the ciphertext M' and related parameters to the CS

(6) User request: it is executed at the user side, by sending request to the CS

(7) Verify integrity: this phase is performed by the user and the CS, for verifying the integrity of the ciphertext M'

(8) Decryption: it is performed by the user, and the cipher text M is decrypted by taking input the ciphertext M' and related parameters

4.2. Security Model. To analyze the security of the proposed data sharing scheme more accurately, we briefly introduce the two types of attacks. Then, we define the required security features and design goals. The detailed security analysis about these security requirements will be described in Section 6.

We consider the following two types of attack.

(1) Replay attack: this attack may repeat the message or delay the message. This can be done by adversary who intercept the message of an old conversation and retransmit it

(2) Eavesdropping attack: it refers to the attacker passively monitoring the communication between users to obtain the transmitted data when the network communications are unsecure

For secure data sharing, the proposed scheme must meet the following security properties.

(1) Correctness: the proposed scheme allows legitimate users to correctly detect whether the information stored in CS is complete. Besides, only authorized users can obtain encrypted data within a valid time and restore the data correctly

(2) Freshness of encryption key: the encryption key generated by the patient in the data encryption and upload phase is only used once. Freshness of encryption key ensures that attackers cannot reuse one encryption key to recover other encrypted sensitive data

(3) Authentication: the purpose of authenticating user is to ensure that, for a given user U, any user N other than U, executing the agreement and impersonating U, CS or TA will not accept the identity of U. The proposed scheme should be required to guarantee that only authorized users designated by the patient himself could access the encrypted health data through CS. And unauthorized users cannot obtain the shared health information. What is more, the authorized users could only access the data for a limited time. The authentication process can prevent user impersonation attack in which attackers act like a legitimate user

(4) Anonymity of patient: since the patient's identity will reveal privacy-sensitive information, it is essential to keep the user's identity confidential. Anonymity means hiding the patient's identity to prevent others from knowing it. In this scheme, the anonymity of patient is ensured if any attackers cannot obtain the real identity Pid of any patient

(5) Integrity certification: the messages transmitted on the public channel can be certificated by the receiver. Besides, any incomplete shared data will be detected by users before decrypting the data. This feature is very important to verify that health data has not been tampered with during transmission and storage process

(6) Forward secrecy of encryption key: the forward secrecy could ensure that past users cannot access the sensitive data uploaded in the future

Furthermore, it is important to propose a solution for security and privacy in IoMT, which should reduce the computational cost and consume few resources. Hence, the security design goals of our data sharing scheme for IoMT should meet the following points.

Privacy preserving: data privacy includes the privacy of the patient's identity and the privacy of shared medical data. The medical data contains electronic medical records and personal condition related to health. If the health information is leaked or accessed by unauthorized adversaries, there is no doubt that it will have a great impact on patients. Hence, it is necessary to guarantee that shared health data is kept confidential from CS and any unauthorized users. Then, this article needs to provide access control for shared data. All users who want to access data need to verify their identity. Any unauthorized users that are not defined by the patient and CS cannot access the encrypted health data. In addition, the proposed scheme needs to anonymize the identity of patients to protect the identity information from being leaked. Consequently, the proposed scheme should provide the anonymity of patient and data access control to ensure the privacy of patient identity and the security of personal health information.

Lightweight operations: the physiological sensors deployed on patients are resource-constrained devices; therefore, the proposed scheme needs to reduce the amount of calculation of patients to improve efficiency of data sharing. To address this issue, we aim to design a lightweight data sharing scheme using ECC. This is because ECC can implement higher security with a small key. Besides, it can also insulate privacy with lower computational complexity as compared to bilinear pairing. Accordingly, this scheme realizes lightweight computations by ECC, hash, and XOR operations.

Effectiveness: in the proposed scheme, it is important to ensure that patients can efficiently share health data with users. Firstly, patients should securely upload health data to CS for sharing with authorized users. Secondly, authorized users should be able to decrypt the required health data for effective medical analysis.

5. Proposed Scheme

For the purpose of privacy protection, we design a secure data sharing scheme for IoMT. This scheme contains the following three phases: (1) system initialization phase, (2) data encryption and upload phase, and (3) data sharing phase. In addition, Table 1 provides the main notations used throughout this paper.

Table 1: Notation table.

No.	Notation	Explanation
1	p	A large prime number
2	E	An elliptic curve of prime order p
3	G_E	An additive elliptic curve group of order q
4	B	Base point of G_E
5	q	Order of G_E
6	O	Point at infinity
7	Z_q	A set with q elements
8	Z_q^*	$Z_q^* = Z_q - \{0\}$
9	h	One-way hash function, $h : \{0,1\} \times G \longrightarrow Z_q^*$
10	S_{TA}	Secret key of trusted authority (TA)
11	$\mathrm{PK}_{\mathrm{TA}}$	Public key of TA
12	warr	Warrant of user
13	Uid, Pid	Identity of user and patient
14	$\mathrm{sk}_{\mathrm{Uid}}$, $\mathrm{sk}_{\mathrm{CS}}$	Secret key of the user and cloud server (CS)
15	Ptid	Temporary identity of patient
16	M	Health data
17	M'	Encrypted data
18	$X \| Y$	Concatenate operation
19	$\oplus$	Bitwise XOR operation
20	$A \longrightarrow B$	Entity A sends the message towards entity B through a public channel

5.1. System Initialization Phase. Firstly, TA generates public parameters and its own secret key. Then, any user in the scheme who wants to access health data should first register with TA. Next, he can obtain his secret key and warrant generated by TA. Like users, patients also need to register with TA before receiving system services. During registration, the patient transfers his temporary identity instead of his real identity via open channel. Hence, the patient's identity information is protected. In addition, the patient needs to define a user identity set. This phase is described in detail below and its process is described in Figure 2.

(1) *Setup*: firstly, TA selects a hash function $h : \{0,1\} \times G \longrightarrow Z_q^*$. Then, TA selects its secret key $S_{\mathrm{TA}} \in Z_q^*$ and CS's secret key $\mathrm{sk}_{\mathrm{CS}} \in Z_q^*$ and calculates its public key according to $\mathrm{PK}_{\mathrm{TA}} = S_{\mathrm{TA}} \bullet B$. TA keeps secret key S_{TA} secretly and publishes public system parameters $\{E, B, h, \mathrm{PK}_{\mathrm{TA}}, G_E\}$. Besides, TA sends $\mathrm{sk}_{\mathrm{CS}}$ to CS via a secure channel

(2) *User registration*: after receiving identity $\mathrm{Uid}_j \in \{0,1\}^*$ from user, TA selects random $r \in Z_q^*$ and computes the private key $\mathrm{sk}_{\mathrm{Uid}} = \mathrm{Uid}_j \cdot r$ for him. Then, TA chooses random $a_1, a_2 \in Z_q^*$ and computes $b_1 = a_1 \cdot B$, $b_2 = a_2 \cdot B$. Then, the warrant of the user is $\mathrm{warr} = a_1 + a_2 \cdot h(\mathrm{Uid}_j \| t_1)$, where t_1 means that authorized users can effectively access shared health information within this time. Next, TA transfers $\mathrm{sk}_{\mathrm{Uid}}$ and warr towards user through a secure channel. Finally, TA computes $E_1 = \mathrm{sk}_{\mathrm{CS}} \bullet h(\mathrm{Uid}_j \| b_1 \| b_2 \| t_1)$ and sends $\{\mathrm{Uid}_j, b_1, b_2, t_1, E_1\}$ to CS. After receiving $\{\mathrm{Uid}_j, b_1, b_2, t_1, E_1\}$, CS computes $E_1' = \mathrm{sk}_{\mathrm{CS}} \bullet h(\mathrm{Uid}_j' \| b_1' \| b_2' \| t_1')$ and checks whether the equation $E_1' = E_1$ holds. If not established, CS terminates this session. On the contrary, CS keeps $\{\mathrm{Uid}_j, b_1, b_2, t_1\}$ locally for the later computation

(3) *Patient registration*: patient Pid first chooses $k \in Z_q^*$ and computes $P_1 = k \cdot B$, $P_2 = k \cdot \mathrm{PK}_{\mathrm{TA}}$, $y_n = h(P_2) \bigoplus \mathrm{Pid}$. Next, the patient defines a set, $S = \langle \mathrm{Uid}_j \rangle_{j=1}^{t}$, which represents a collection of the identities of users who can access his health information. If the identity of user meets $\mathrm{Uid}_j \subseteq S$ and the access time is valid, he can access shared data M. Then, the patient generates a timestamp t_2 and computes his temporary identity $\mathrm{Ptid} = h(\mathrm{Pid} \| P_2 \| S \| t_2)$. After receiving register information $\langle S, P_1, \mathrm{Ptid}, y_n, t_2 \rangle$ from the patient, TA checks the validity of the predicate $(t^* - t_2 (?/<) \Delta t)$, where t^* is the message receiving time and the maximum transmission delay is described by Δt, and aborts if the predicate is not justified. Otherwise, TA calculates $P_2^* = P_1 \cdot S_{\mathrm{TA}}$, $\mathrm{Pid}^* = y_n \oplus h(P_2^*)$, $\mathrm{Ptid}^* = h(\mathrm{Pid}^* \| P_2^* \| S \| t_2)$. After that, TA checks whether the equation $\mathrm{Ptid}^* = \mathrm{Ptid}$ holds. If not, CS drops the

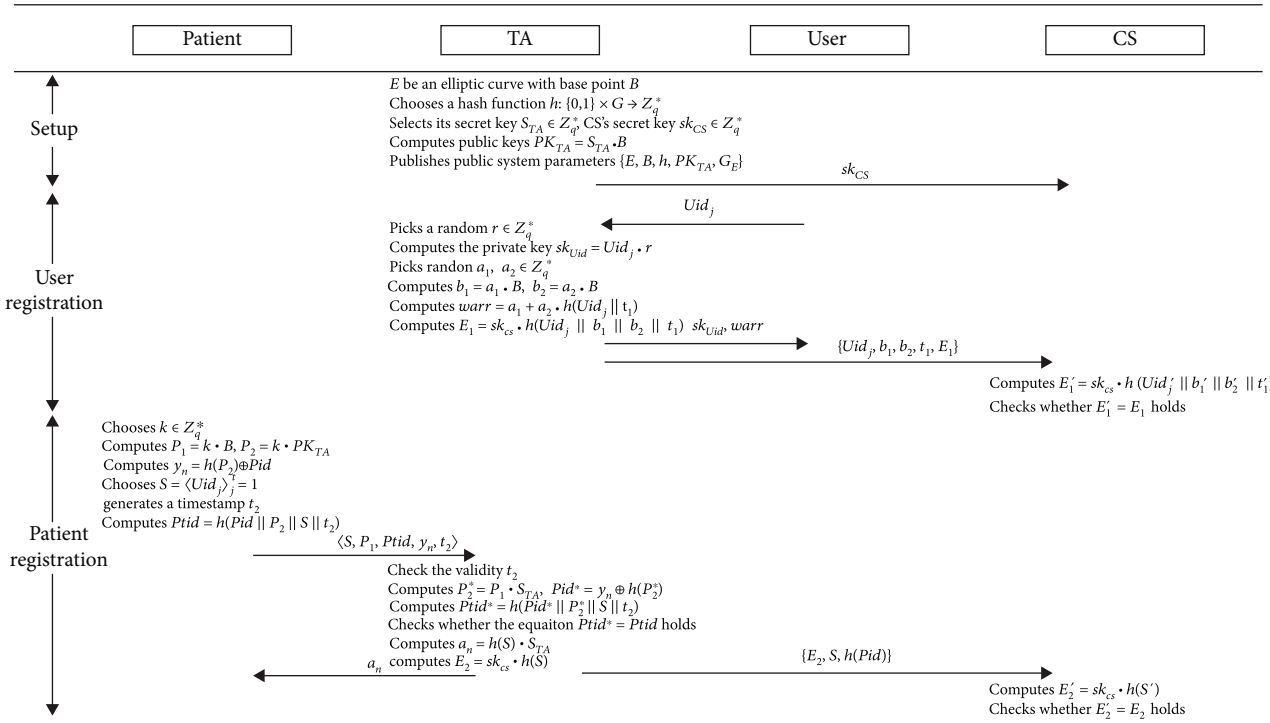


FIGURE 2: System initialization phase.

received message and terminates this session. Otherwise, TA computes $a_n = h(S) \cdot S_{\mathrm{TA}}$ and transfers a_n to the patient via a secure channel. Then, TA computes $E_2 = \mathrm{sk}_{\mathrm{CS}} \bullet h(S)$ and sends $\{E_2, S, h(\mathrm{Pid})\}$ to CS. After receiving $\{E_2, S, h(\mathrm{Pid})\}$, CS computes $E_2' = \mathrm{sk}_{\mathrm{CS}} \bullet h(S')$ and checks whether the equation $E_2' = E_2$ holds. If the equation does not hold, CS terminates the session. On the contrary, CS keeps S locally for the later verification

5.2. Data Encryption and Upload Phase. In this proposed scheme, we are given that the maximum length of shared health data is l. Patient should encrypt data $M \in \{0,1\}^l$ to M' to ensure the privacy of M and then upload M' to CS. This phase is described in detail below and its process is described in Figure 3.

(1) *Encryption*: patient Pid needs to encrypt the gathering data M with a fresh encryption key K. Firstly, the patient randomly chooses random $x, y \in Z_q^*$, and computes $d_1 = a_n \bigoplus x \bigoplus \mathrm{Pid}$, $Y = y \cdot B$, $Z = x \cdot Y$, $\alpha = h(\mathrm{Pid} \| d_1 \| Y)$, $K = h(x \| a_n \| Z \| \mathrm{Pid})$. And then, the patient uses the formula $M' = K \oplus M$ to encrypt M and get ciphertext M'

(2) *Upload*: patient Pid generates a timestamp t_3 and computes $\beta = h(M' \| \mathrm{Pid} \| t_3)$. Then, the patient sends $\langle h(\mathrm{Pid}), Y, d_1, \alpha, \beta, M', t_3 \rangle$ to CS. On receiving this message, CS firstly examines the freshness of the timestamp t_3. If examination is successful, CS stores the information. On the contrary, CS drops this message and terminates this session

5.3. Data Sharing Phase. In order to obtain shared health data, user should verify his identity with TA and CS. He first generates timestamp and forwards related parameters towards CS through public channel. Then, CS will send encrypted data and intermediate parameters to the user if his warrant is valid and his visit time is within the valid time. Next, user verifies that the encrypted data is complete. If verification is successfully done, the user needs to verify himself with TA and obtain the intermediate parameter. If verified successfully, he can download and decrypt M'. This phase is described in detail below and its process is described in Figure 4.

(1) *User request*: user Uid_j first sends his request to CS when he wants to access the shared data M. Then, he generates a timestamp t_4 and transfers $\langle \mathrm{Uid}_j, h(\mathrm{Pid}), t_4, \mathrm{warr} \rangle$ to CS

(2) *Verify integrity*: firstly, CS checks whether Uid_j is in the corresponding set S. If not, CS drops user's requested message and terminates this session. Next, CS checks the validity of the timestamp t_4. Then, CS checks user's warrant with the equation $\mathrm{warr} \cdot B = b_1 + b_2 \cdot h(\mathrm{Uid}_j \| t_1)$. If they are equal, CS sends $\langle \beta, M', t_3 \rangle$ towards the user. After receiving $\langle \beta, M', t_3 \rangle$, the user examines that the data M is complete by computing the equation $\beta = h(M' \| \mathrm{Pid} \| t_3)$. If the

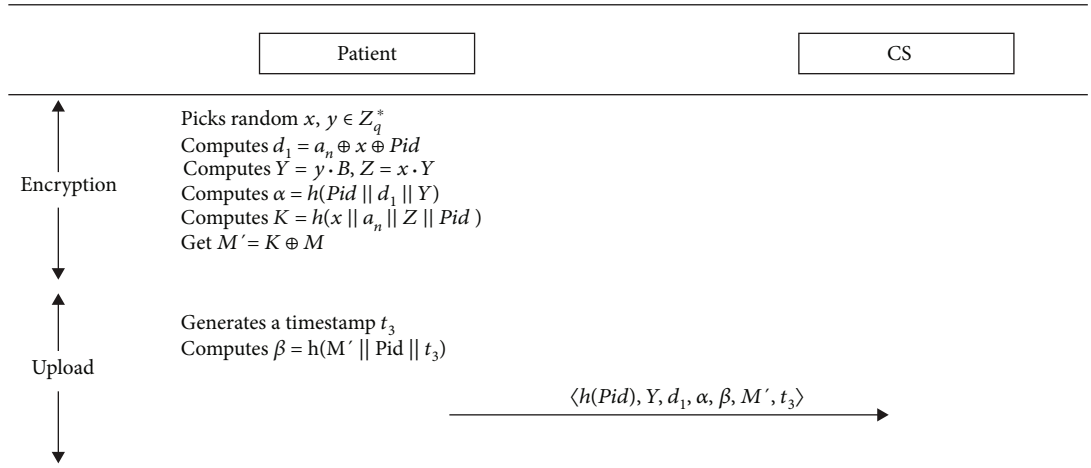


Figure 3: Data encryption and upload phase.

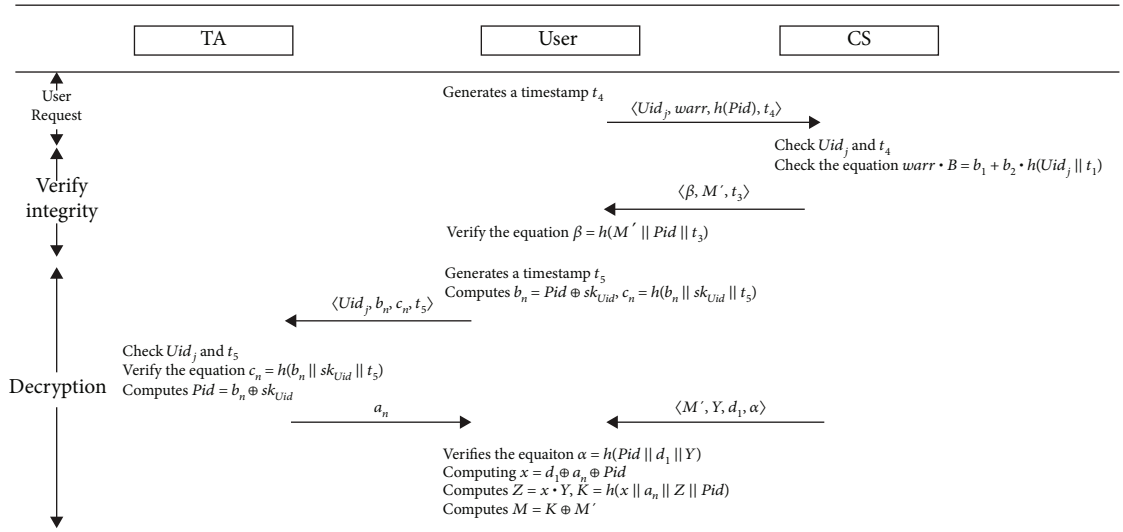


Figure 4: Data sharing phase.

equation is true, the user proceeds to the next step. Otherwise, the user terminates this session

(3) *Decryption*: the user first generates a timestamp t_5, computes $b_n = \text{Pid} \oplus \text{sk}_{\text{Uid}}$, $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$, and sends $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ to TA in order to obtain intermediate parameters for decrypting. After receiving this message, TA verifies the freshness of the timestamp t_5 and the legitimacy of identity Uid_j. If not, TA drops this message and terminates the session. Otherwise, TA verifies the equation $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$ and computes $\text{Pid} = b_n \oplus \text{sk}_{\text{Uid}}$ and then transfers the a_n of patient Pid to user via a secure channel for decrypting data. After getting $\langle M', Y, d_1, \alpha \rangle$ from CS, the user first verifies the equation $\alpha = h(\text{Pid} \| d_1 \| Y)$. If the equation holds, he retrieves symmetric key K by computing $x = d_1 \oplus a_n \oplus \text{Pid}$, $Z = x \cdot Y$, $K = h(x \| a_n \| Z \| \text{Pid})$. Finally, the user gets the plain text of encrypted data by computing $M = K \oplus M'$

6. Security Analysis

This section analyzes how the proposed scheme can effectively meet the security properties and two types of attack of the proposed scheme presented in Section 4.2.

6.1. Security Properties

(1) Correctness: in the *data sharing phase*, legitimate user verified by CS can correctly examine that the encrypted data is complete, which is stored in CS. After receiving $\langle \beta, M', t_3 \rangle$ from CS, the user first examines the completeness of data M' by computing $h(M' \| \text{Pid} \| t_3)$. The user compares the calculated result with the received value $\beta = h(M' \| \text{Pid} \| t_3)$. Other illegal users cannot fake this authentication response since the secret identity of patient, Pid, is unknown to them. In *data encryption and upload phase*, the correctness of this property is guaranteed

by the one-way nature of hash function. Besides, only authorized users can obtain encrypted data within a valid time and restore the data correctly. Legitimate user can obtain decryption key K by computing the following equations, $x = d_1 \oplus a_n \oplus \text{Pid}$, $Z = x \cdot Y$, and $K = h(x \| a_n \| Z \| \text{Pid})$. Finally, the user computes $M = K \oplus M'$ to recover the plaintext of shared data

(2) Freshness of encryption key: in the *data encryption*, the encryption key, $K = h(x \| a_n \| Z \| \text{Pid})$, is a hash output, where x is a random integer selected by the patient. This key is different in every encryption

(3) Authentication: since the data transmission is carried out on a public channel, it is important to authenticate users who want to access shared information. The authenticity of the user identity is confirmed by TA and CS. In the *data sharing phase*, the user first sends his request $\langle \text{Uid}_j, h(\text{Pid}), t_4, \text{warr} \rangle$ to CS. CS checks whether the Uid_j is in the corresponding set S. If not, TA drops user's requested message and terminates the session. Next, CS checks the validity of the timestamp t_4. Then, CS checks user warrant with the equation $\text{warr} \cdot B = b_1 + b_2 \cdot h(\text{Uid}_j \| t_1)$. If the equation does not hold, CS terminates the session. After the user passes the CS verification, he must also verify with the TA to obtain the intermediate parameter required for decryption. Hence, the user sends $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ to TA to get intermediate parameter of decryption, where $b_n = \text{Pid} \oplus \text{sk}_{\text{Uid}}$, $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$. On receiving $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$, TA verifies the validity of the timestamp t_5 and Uid_j; if verification is successful, then TA verifies the equation $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$ and computes $\text{Pid} = b_n \oplus \text{sk}_{\text{Uid}}$. Next, TA transfers a_n of patient Pid to user secretly for data decryption. Since the user's private key sk_{Uid} and Pid are secret and are not known by others, no adversary can pretend to be him to authenticate to the TA. Therefore, authentication of the user's identity before obtaining sensitive data ensures more secure communication

(4) Anonymity of patient: the patient transmits messages through a public channel. Because the user's identity Pid is hidden in Ptid or $h(\text{Pid})$, the proposed scheme can guarantee the anonymity of user, as identity of the patient Pid is masked as Ptid or $h(\text{Pid})$. In the *system initialization phase*, the patient transfers his temporary identity, $\text{Ptid} = h(\text{Pid} \| P_2 \| S \| t_2)$ to TA, where $P_2 = k \cdot \text{PK}_{\text{TA}} = P_1 \cdot S_{\text{TA}}$. Besides, in other phases, the patient's identity information is transmitted in the form of hash values, $h(\text{Pid})$. Hence, adversary cannot obtain the real identity Pid of any patient

(5) Integrity certification: to satisfy integrity service, all transmitted messages of the proposed scheme are attached with a verifiable value. In the *system initialization phase*, TA receives the message $\langle S, P_1, \text{Ptid}, y_n, t_2 \rangle$ and checks the integrity of Ptid and S by verifying the timestamp condition $t^* - t_2 < \Delta t$ and verifies $\text{Ptid}^* = \text{Ptid}$ by computing $P_2^* = P_1 \cdot S_{\text{TA}}$, $\text{Pid}^* = y_n \oplus h(P_2^*)$, and $\text{Ptid}^* = h(\text{Pid}^* \| P_2^* \| Sv \| t_2)$. CS receives $\{\text{Uid}_j, b_1, b_2, t_1, E_1\}$ or $\{E_2, S, h(\text{Pid})\}$ and checks the integrity of $\{\text{Uid}_j, b_1, b_2, t_1\}$ or S by computing $E_1' = \text{sk}_{\text{CS}} \bullet h(\text{Uid}_j' \| b_1' \| b_2' \| t_1')$ and checking whether the equation $E_1' = E_1$ holds or by computing $E_2' = \text{sk}_{\text{CS}} \bullet h(S')$ and checking whether the equation $E_2' = E_2$ holds. In the *data sharing phase*, after receiving $\langle \beta, M', t_3 \rangle$, the user verifies that the data M is complete by computing the equation $\beta = h(M' \| \text{Pid} \| t_3)$. During *decryption*, TA receives the message $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ and checks the integrity of b_n by verifying the timestamp condition $t^* - t_2 < \Delta t$ and verifies the equation $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$. User receives the message $\langle M', Y, d_1, \alpha \rangle$ and checks the integrity of d_1 and Y by verifying the equation $\alpha = h(\text{Pid} \| d_1 \| Y)$. As a result of using Pid and sk_{Uid} over the transmitted messages (which are not known by any adversary), any modification on the data by adversaries is detectable. The proposed scheme takes advantage of the one-way nature of the hash function to ensure that the attacker cannot tamper with the transmitted data

(6) Forward secrecy of encryption key: the disclosure of encryption key K does not influence the security of any past encrypted data. The freshness of the encryption key $K = h(x \| a_n \| Z \| \text{Pid})$ ensures that the proposed scheme meets this feature. The one-way nature of the hash function h prevents all secret parameters from being obtained by attackers. In addition, x, a_n, and Z are all dynamic change with the sessions, where $a_n = h(S) \cdot S_{\text{TA}}$, $Z = x \cdot Y$

6.2. Possible Attacks

Theorem 1 (replay attack). *The proposed scheme can resist the replay attack.*

Proof. The use of timestamp can protect the information transmitted in the proposed scheme from replay attack launched by the adversary. CS and TA can distinguish a replay attack by the examination of the freshness of the timestamp t_i as $t^* - t_i < \Delta t$, where t^* is the current time that the CS or TA gets the message and Δt is the maximum transmission delays. Besides, the use of timestamp t_i ensures the transmitted message cannot be tampered with by an adversary. For example, in the *system initialization phase*, there is an adversary $\mathcal{A}$ and he intercepted a message $\langle S, P_1, \text{Ptid}, y_n, t_2 \rangle$. $\mathcal{A}$ replays message $\langle S', P_1', \text{Ptid}', y_n', t_2' \rangle$. But process will terminate since on receiving $\langle S', P_1', \text{Ptid}', y_n', t_2' \rangle$, TA verifies the freshness of the timestamp

t_2' by computing $t^* - t_2'$ and found that the message $\langle S', P_1', \text{Ptid}', y_n', t_2' \rangle$ is not fresh, as shown in the following equation $t^* - t_2' > \Delta t$. In the *data encryption and upload phase*, $\mathscr{A}$ records message $\langle h(\text{Pid}), Y, d_1, \alpha, \beta, M', t_3 \rangle$. $\mathscr{A}$ initiates a session by transmitting message $\langle h(\text{Pid})', Y', d_1', \alpha', \beta', M'', t_3' \rangle$. But process will terminate since after obtaining the message, CS checks the freshness of the timestamp t_3'. And similarly, in the *data sharing phase*, $\mathscr{A}$ records message $\langle \text{Uid}_j, h(\text{Pid}), t_4, \text{warr} \rangle$ or $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$. $\mathscr{A}$ initiates a session by transmitting message $\langle \text{Uid}_j', h(\text{Pid})', t_4', \text{warr}' \rangle$ or $\langle \text{Uid}_j', b_n', c_n', t_5' \rangle$. But process will terminate since after obtaining the message, CS or TA can detect this message is illegal by verifying the freshness of the timestamp t_4' or the timestamp t_5'. Hence, the proposed scheme stands with the replay attack.□ □

Theorem 2 (eavesdropping attack). *From the intercepted communication parameters, an adversary cannot obtain any secret information.*

Proof. In the data sharing phase of the proposed scheme, an adversary $\mathscr{A}$ can capture the transmitted data by monitoring public channels. He collects the tuple $\langle M', Y, d_1, \alpha \rangle$ from CS to user and the tuple $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ from user to TA. It is noted that the encryption key $K = h(x \| a_n \| Z \| \text{Pid})$. $\mathscr{A}$ cannot reach x, a_n, Z, and Pid, depending on the intercepted messages. This is due to the parameter x, selected at random by patient, is unknown to $\mathscr{A}$. And since a_n is secretly transmitted by TA to user and patient, no one else knows the value of a_n. The parameter $d_1 = a_n \bigoplus x \bigoplus \text{Pid}$ guarantees that even if $\mathscr{A}$ obtains d_1, he cannot calculate x, Pid, and a_n. The hash function h guarantees that even if $\mathscr{A}$ obtains parameter $\alpha = h(\text{Pid} \| d_1 \| Y)$, he cannot guess the input parameter of h. Besides, $\mathscr{A}$ cannot calculate $Z = x \cdot Y$ because $\mathscr{A}$ does not know x. Finally, the parameter $b_n = \text{Pid} \oplus \text{sk}_{\text{Uid}}$ guarantees that even if $\mathscr{A}$ obtains b_n, he cannot calculate sk_{Uid} and Pid. Therefore, the proposed scheme can protect the encryption key K from being learned by the adversary $\mathscr{A}$, and $\mathscr{A}$ cannot obtain sensitive data from the ciphertext M'. In conclusion, the proposed scheme stands with eavesdropping attack.□ □

7. Performance Analysis

We concretely analyze the performance of the proposed scheme, including computational and communication overheads. Besides, there is a comparison regarding the execution time and security of the proposed scheme and other schemes in [6, 13, 30].

7.1. Computation Cost. The computation cost is analyzed by calculating the operations used in each phase of the scheme. It is noting that the proposed scheme uses t_h, t_{xor}, t_{ecm}, and t_{add} to denote the calculating time needed for the hash function, XOR operation, ECC scalar multiplication, and addition operation, respectively.

7.1.1. System Initialization Phase. In *setup*, TA selects its secret key $S_{TA} \in Z_q^*$ and CS's secret key $\text{sk}_{CS} \in Z_q^*$ and computes $\text{PK}_{TA} = S_{TA} \bullet B$, and the computation overhead is t_{ecm}. In *user registration*, TA first picks a random $r \in Z_q^*$ and computes the private key $\text{sk}_{\text{Uid}} = \text{Uid}_j \cdot r$ for user. Next, TA picks random $a_1, a_2 \in Z_q^*$ and computes $b_1 = a_1 \cdot B$, $b_2 = a_2 \cdot B$. The warrant of user represents as $\text{warr} = a_1 + a_2 \cdot h(\text{Uid}_j \| t_1)$. Then, TA computes $E_1 = \text{sk}_{CS} \bullet h(\text{Uid}_j \| b_1 \| b_2 \| t_1)$. After receiving $\{\text{Uid}_j, b_1, b_2, t_1, E_1\}$, CS computes $E_1' = \text{sk}_{CS} \bullet h(\text{Uid}_j' \| b_1' \| b_2' \| t_1')$.Hence, the calculation cost is $6t_{ecm} + 3t_h + t_{add}$. In *patient registration*, patient Pid first chooses $k \in Z_q^*$ and computes $P_1 = k \cdot B$, $P_2 = k \cdot \text{PK}_{TA}$, $y_n = h(P_2) \bigoplus \text{Pid}$. Next, the patient chooses $S = \langle \text{Uid}_j \rangle_{j=1}^{t}$, generates a timestamp t_2, and computes his temporary identity $\text{Ptid} = h(\text{Pid} \| P_2 \| S \| t_2)$. Then, TA computes $P_2^* = P_1 \cdot S_{TA}$, $\text{Pid}^* = y_n \oplus h(P_2^*)$, $\text{Ptid}^* = h(\text{Pid}^* \| P_2^* \| S \| t_2)$, $a_n = h(S) \cdot S_{TA}$, and $E_2 = \text{sk}_{CS} \bullet h(S)$. After receiving $\{E_2, S, h(\text{Pid})\}$, CS computes $E_2' = \text{sk}_{CS} \bullet h(S')$. Hence, the computation overhead of the algorithm is $6t_{ecm} + 6t_h + 2t_{xor}$.

7.1.2. Data Encryption and Upload Phase. In *encryption*, patient Pid picks random $x, y \in Z_q^*$ and computes $d_1 = a_n \bigoplus x \bigoplus \text{Pid}$, $Y = y \cdot B$, $Z = x \cdot Y$, $\alpha = h(\text{Pid} \| d_1 \| Y)$, and $K = h(x \| a_n \| Z \| \text{Pid})$. Then, the patient uses the formula $M' = K \oplus M$ to encrypt M. Hence, the calculation cost is $2t_{ecm} + 2t_h + 3t_{xor}$. In *upload*, the patient generates a timestamp t_2 and computes $\beta = h(M' \| \text{Pid} \| t_3)$ and the computation overhead is t_h.

7.1.3. Data Sharing Phase. In *user request*, the user generates timestamp t_4 and transfers $\langle \text{Uid}_j, h(\text{Pid}), t_4, \text{warr} \rangle$ to CS. Hence, the computation cost of the algorithm is 0. In *verify integrity*, CS examines user's warrant by computing the formula $\text{warr} \cdot B = b_1 + b_2 \cdot h(\text{Uid}_j \| t_1)$. Next, the user examines the completeness of data M by computing the formula $\beta = h(M' \| \text{Pid} \| t_3)$, so the computation cost of the algorithm is $t_{ecm} + 2t_h + t_{add}$. In *decryption*, the user generates a timestamp t_5, computes $b_n = \text{Pid} \oplus \text{sk}_{\text{Uid}}$, $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$, and sends $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ to TA. Then, TA verifies the equation $c_n = h(b_n \| \text{sk}_{\text{Uid}} \| t_5)$ and computes $\text{Pid} = b_n \oplus \text{sk}_{\text{Uid}}$. Finally, the user downloads $\langle M', Y, d_1, \alpha \rangle$ from CS and verifies the equation $\alpha = h(\text{Pid} \| d_1 \| Y)$, computing $x = d_1 \oplus a_n \oplus \text{Pid}$, $Z = x \cdot Y$, $K = h(x \| a_n \| Z \| \text{Pid})$, and $M = K \oplus M'$. Hence, the computation overhead of the algorithm is $t_{ecm} + 4t_h + 5t_{xor}$.

The calculation cost of the XOR operation is so small that it can be ignored. Table 2 illustrates the calculated cost of each stage in the proposed scheme.

7.2. Communication Cost. Table 3 lists the communication cost consumed by each transmission. The proposed scheme chooses SHA-1 as hash function, and the SHA-1 outputs a hash digest with length of 160 bits. In addition, we presume the length of elliptic curves $|q| = 160$ bits, the shared data $|M| = 320$ bits, the timestamp $|t_i| = 32$ bits, and the identity

Table 2: Computation cost of the proposed scheme.

Phase	Algorithm	Explanation
System initialization phase	Setup	t_{ecm}
	User registration	$6t_{ecm} + 3t_h + t_{add}$
	Patient registration	$6t_{ecm} + 6t_h + 2t_{xor} \approx 6t_{ecm} + 6t_h$
Data encryption and upload phase	Encryption	$2t_{ecm} + 2t_h + 3t_{xor} \approx 2t_{ecm} + 2t_h$
	Upload	t_h
Data sharing phase	User request	0
	Verify integrity	$t_{ecm} + 2t_h + t_{add}$
	Decryption	$t_{ecm} + 4t_h + 5t_{xor} \approx t_{ecm} + 4t_h$

Table 3: Communication cost of the proposed scheme.

Communication between entities	Communication cost
(User ⟶ TA)	416 bits
(User ⟶ CS)	384 bits
(Patient ⟶ TA)	$512 + 32t$
(Patient ⟶ CS)	1152 bits

$|\text{id}| = 32$ bits. In the transmission (user ⟶ TA), user sends Uid_j during the *system initialization phase* and $\langle \text{Uid}_j, b_n, c_n, t_5 \rangle$ during the *data sharing phase*. The size of these messages is $32 \times 2 + 160 \times 2 + 32 = 416$ bits. In the transmission (user ⟶ CS), user sends the tuple, $\langle \text{Uid}_j, h(\text{Pid}), t_4, \text{warr} \rangle$ of size 384 bits. In the transmission (patient ⟶ TA), the patient sends the tuple, $\langle S, P_1, \text{Ptid}, y_n, t_2 \rangle$ of size $512 + 32t$ bits, where t is the number of user identity to access his health data. In the transmission (patient ⟶ CS), the patient sends the tuple $\langle h(\text{Pid}), Y, d_1, \alpha, \beta, M', t_3 \rangle$ of size 1152 bits.

7.3. Comparisons with Related Schemes. In order to compare several schemes more intuitively, we construct Table 4 according to [7]. Table 4 illustrates the calculation cost of different operations. And we demonstrate the calculation overheads of the proposed scheme and other schemes in [6, 13, 30] according to Table 4. Table 5 summarizes the calculation overheads by patient in the proposed data sharing scheme and other recently proposed schemes. From the comparison in Table 5, the proposed scheme is extremely more lightweight than schemes in [6, 13, 30], because of the executing of ECC, hash, and XOR operations.

According to the data in Table 5, the proposed scheme reduced the computational cost from Ding et al. [13] which is $(1803.8t_h - 295t_h)/(1873.5t_h) = 83.6\%$. Computation cost reduction from Chen and Peng [6] is $(817.5t_h - 295t_h)/(817.5t_h) = 63.91\%$. Computation cost reduction from Sowjanya et al. [30] is $(584t_h - 295t_h)/(584t_h) = 49.49\%$.

The analysis of security features for the proposed scheme in comparison with the scheme of Ding et al. [13], Chen and Peng [6], and Sowjanya et al. [30] is in Table 6. From this table, the schemes in [6, 13] do not meet the anonymity of patients. Besides, Ding et al. [13] do not give the protection against replay attack. Chen and Peng [6] and Sowjanya et al. [30] may suffer from eavesdropping attack. It is clear from the result of the comparison that the proposed scheme is more secure than these similar schemes because it can resist

Table 4: Calculation overheads of different operations with t_h as the time unit.

Symbol	Description	Cost
t_h	SHA-1 hash function	t_h
t_{ecm}	ECC scalar multiplication	$72.5t_h$
t_{exp}	Modular exponentiation	$600t_h$
t_{sym}	Symmetric encryption	t_h
t_{mm}	Modular multiplication	$2.5t_h$
t_{ma}	Modular addition	$0.3t_h$

Table 5: Comparisons of the computation cost by patient.

Schemes	Computation cost by patient
Ding et al. [13]	$3t_{ecp} + t_h + t_{mm} + t_{ma} = 1803.8t_h$
Chen and Peng [6]	$3t_{ecm} + t_{exp} = 817.5t_h$
Sowjanya et al. [30]	$8t_{ecm} + 3t_h + t_{sym} = 584t_h$
Ours	$4t_{ecm} + 5t_h = 295t_h$

Table 6: Comparisons of security features.

Security features	Ding et al. [13]	Chen and Peng [6]	Sowjanya et al. [30]	Ours
F_1	No	Yes	Yes	Yes
F_2	Yes	No	No	Yes
F_3	Yes	Yes	Yes	Yes
F_4	No	No	Yes	Yes
F_5	Yes	Yes	Yes	Yes
F_6	Yes	Yes	Yes	Yes

F_1: resist replay attack; F_2: resist eavesdropping attack; F_3: provide authentication; F_4: provide anonymity of patient; F_5: provide integrity certification; F_6: provide forward security.

the above two kinds of attacks and can meet all desired security features.

In summary, compared with the three similar schemes, it is seen that the proposed scheme can perform less computations and meet more security features. Besides, our scheme provides the anonymity of patient's identity and the authentication of access to shared health data. Thus, the proposed scheme is more lightweight and secure for IoMT.

8. Conclusions

We propose a novel design of lightweight privacy-preserving data sharing scheme for IoMT. The presented scheme can not only provide anonymous feature for patient while achieving the data sharing between patients and users but also ensure that only authorized users designated by the patient himself could access the encrypted health data. Furthermore, this scheme realizes lightweight computations by ECC, hash, and XOR operations. Compared with similar solutions, the proposed scheme can satisfy all desired security features as well as achieve more lightweight computations on both patients and users. It is absolutely attractive for data sharing in IoMT.

Data Availability

The data used to support the findings of this study are included within the article.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Consent

Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work was partially supported by the National Nature Science Foundation of China (Grant Nos. 61772224, 62172181, and 62072133), the Fundamental Research Funds for the Central Universities (No. CCNU19TS019), the Research Planning Project of National Language Committee (No. YB135-40), and the key projects of Guangxi Natural Science Foundation (no. 2018GXNSFDA281040).

References

[1] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018.

[2] F. Al-Turjman, M. H. Nawaz, and U. D. Ulusar, "Intelligence in the Internet of Medical Things era: a systematic review of current and future trends," *Computer Communications*, vol. 150, pp. 644–660, 2020.

[3] M. M. Islam, A. Rahaman, and M. R. Islam, "Development of smart healthcare monitoring system in IoT environment," *SN Computer Science*, vol. 1, no. 3, pp. 1–11, 2020.

[4] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," *Wireless Networks*, vol. 17, no. 1, pp. 1–18, 2011.

[5] M. Shuai, B. Liu, N. Yu, L. Xiong, and C. Wang, "Efficient and privacy-preserving authentication scheme for wireless body area networks," *Journal of Information Security and Applications*, vol. 52, article 102499, 2020.

[6] R. Chen and D. Peng, "Analysis and improvement of a mutual authentication scheme for wireless body area networks," *Journal of Medical Systems*, vol. 43, no. 2, pp. 1–10, 2019.

[7] X. Li, M. H. Ibrahim, S. Kumari, A. K. Sangaiah, V. Gupta, and K. K. R. Choo, "Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks," *Computer Networks*, vol. 129, pp. 429–443, 2017.

[8] Z. Guan, Z. Lv, X. Du, L. Wu, and M. Guizani, "Achieving data utility-privacy tradeoff in Internet of Medical Things: a machine learning approach," *Future Generation Computer Systems*, vol. 98, pp. 60–68, 2019.

[9] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[10] J. Wu, L. Ping, X. Ge, W. Ya, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *2010 International Conference on Intelligent Computing and Cognitive Informatics*, pp. 380–383, Kuala Lumpur, Malaysia, 2010.

[11] Y. Ming and T. Zhang, "Efficient privacy-preserving access control scheme in electronic health records system," *Sensors*, vol. 18, no. 10, article 3520, 2018.

[12] D. Schröder and H. Schröder, "Verifiable data streaming," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 953–964, Raleigh North Carolina USA, 2012.

[13] R. Ding, H. Zhong, J. Ma, X. Liu, and J. Ning, "Lightweight privacy-preserving identity-based verifiable IoT-based health storage system," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8393–8405, 2019.

[14] X. Chen, W. Susilo, J. Li et al., "Efficient algorithms for secure outsourcing of bilinear pairings," *Theoretical Computer Science*, vol. 562, pp. 112–121, 2015.

[15] V. Odelu and A. K. Das, "Design of a new CP-ABE with constant-size secret keys for lightweight devices using elliptic curve cryptography," *Security and Communication Networks*, vol. 9, no. 17, 4059 pages, 2016.

[16] S. Ding, C. Li, and H. Li, "A novel efficient pairing-free CP-ABE based on elliptic curve cryptography for IoT," *IEEE Access*, vol. 6, pp. 27336–27345, 2018.

[17] W. Itani, A. Kayssi, and A. Chehab, "Energy-efficient incremental integrity for securing storage in mobile cloud computing," in *2010 International Conference on Energy Aware Computing*, pp. 1-2, Cairo, Egypt, 2010.

[18] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[19] Boyang Wang, Baochun Li, and Hui Li, "Oruta: privacy-preserving public auditing for shared data in the cloud," *IEEE*

Transactions on Cloud Computing, vol. 2, no. 1, pp. 43–56, 2014.

[20] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems and Software*, vol. 113, pp. 130–139, 2016.

[21] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2019.

[22] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud Computing Security*, pp. 55–66, Chicago Illinois USA, 2009.

[23] L. Xu, X. Wu, and X. Zhang, "CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM symposium on Information, Computer and Communications Security*, pp. 87-88, Seoul Korea, 2012.

[24] A. N. Khan, M. L. M. Kiah, S. A. Madani, M. Ali, A. U. R. Khan, and S. Shamshirband, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *Journal of Supercomputing*, vol. 68, no. 2, pp. 624–651, 2014.

[25] S. K. Nayak and S. Tripathy, "Privacy preserving provable data possession for cloud based electronic health record system," in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 860–867, Tianjin, China, 2016.

[26] D. Ramesh, R. Mishra, and D. R. Edla, "Secure data storage in cloud: an e-stream cipher-based secure and dynamic updation policy," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 873–883, 2017.

[27] X. A. Wang, J. Ma, F. Xhafa, M. Zhang, and X. Luo, "Cost-effective secure E-health cloud system using identity based cryptographic techniques," *Future Generation Computer Systems*, vol. 67, pp. 242–254, 2017.

[28] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal*, vol. 12, no. 1, pp. 64–73, 2018.

[29] S. Jiang, X. Zhu, and L. Wang, "EPPS: efficient and privacy-preserving personal health information sharing in mobile healthcare social networks," *Sensors*, vol. 15, no. 9, pp. 22419–22438, 2015.

[30] K. Sowjanya, M. Dasgupta, and S. Ray, "An elliptic curve cryptography based enhanced anonymous authentication protocol for wearable health monitoring systems," *International Journal of Information Security*, vol. 19, no. 1, pp. 129–146, 2020.

[31] X. Zhang, Y. Tang, S. Cao, C. Huang, and S. Zheng, "Enabling identity-based authorized encrypted diagnostic data sharing for cloud-assisted E-health information systems," *Journal of Information Security and Applications*, vol. 54, article 102568, 2020.

[32] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[33] V. S. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology — CRYPTO '85 Proceedings. CRYPTO 1985*, H. C. Williams, Ed., , pp. 417–426, Springer, Berlin, Heidelberg, 1985.

[34] S. Ray, G. P. Biswas, and M. Dasgupta, "Secure multi-purpose mobile-banking using elliptic curve cryptography," *Wireless Personal Communications*, vol. 90, no. 3, pp. 1331–1354, 2016.

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5798603, 8 pages
https://doi.org/10.1155/2021/5798603

Research Article

An Efficient Identification Algorithm to Identify Mobile RFID Tags

Yonglei Yao and Jian Su

School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China

Correspondence should be addressed to Jian Su; sj890718@gmail.com

Received 21 May 2021; Accepted 14 August 2021; Published 26 August 2021

Academic Editor: Pengfei Wang

Tag identification in a fast-moving environment is an emerging challenge for future RFID systems. However, existing literatures on the tag reading protocol design primarily apply to stationary scenarios, which fail to cope with mobile environments with unreliable channel condition. In this paper, we first review various types of prior reading protocols and then discuss a new direction of mobile tag reading by proposing a novel partitioning strategy. This analysis and experimental results show its superiority in achieving reading performance for the UHF RFID system under a mobile environment.

1. Introduction

With the advent of 5G mobile internet, the internet of everything will become a reality and be pervasively applied in our daily life [1]. It is reported that the number of global Internet of Things (IoT) [2, 3] connections will reach to 50 billion in 2020. As a key supporting technology for IoT, RFID has been boom developing in recent years. According to the analysis data in the IDTechEx market report [4], the total value of the global RFID market will be as high as 13.4 billion dollar in 2022, including readers, tags, and middleware for both passive and active RFID. IDTechEx also predicts that the sales of UHF RFID tags will increase from 15 billion dollar in 2019 to 41.2 billion dollars in 2024. The main applications are concentrated in areas including clothing label, air luggage, and inventory control. Since passive RFID can transmit wireless data with very little energy, this is especially important for some applications of IoT.

In recent years, the widespread use of passive RFID tags has promoted the development of various industries. For example, RFID tags can be widely used in the field of traffic transmission and easy access to existing departure control systems (DCS) and baggage reconciliation system (BRS). The introduction of RFID technology can enable airports to transport passenger's luggage more efficiently, quickly, and safely. Whether in the aviation industry or in other fields such as logistics warehousing and sales management [5], it is often necessary to accurately identify and sort the managed items through RFID technology in a mobile conveyor application scenario. The fundamental problem of these applications is how to accurately and effectively identify tags in a mobile scenario. Although the existing anticollision algorithms can solve the tag identification problem in a stationary scenario, there are still many new challenges in the identification of mobile tags. For example, the speed at which the object passes on the conveyor belt and the distance between the objects will affect the reading performance of the RFID system.

The conventional anticollision approaches consist of three categories: tree-based [6], Aloha-based, and their hybrid [7, 8]. Most of previous works focus on how to optimize the parameters when identifying static RFID tags under a stationary scenario [9]. However, in actual RFID applications, the status of the reader and tags will change over time, such as path loss, signal attenuation, and tags moving away from the reader's coverage, can lower the reading performance. Hence, the conventional anticollision solutions are difficult to adapt to a mobile environment. Focusing on tag moving scenario, some literatures [10, 11] have presented corresponding approaches to optimize the identification

performance of RFID network. In the tag moving scenario, these previous strategies [10, 11] consider some previously identified tags (they are called staying tags) stay in the reader vicinity in the ongoing frames, and some new arrival tags participate in the current reading process. Moreover, they use bit tracking technology (a technology that can accurately identify and locate the collided bit) to improve the reading performance. However, in the reverse link of UHF RFID system, the deviation in the backscatter link frequency (BLF) of various tags can be as high as plus or minus 22% [4], which causes the received data at the reader side is asynchronous. Therefore, these methods have not been adopted by mainstream UHF national standards, and their scope of implementation is limited. Based on the same tag moving scenario, literature [12] proposes a tree-based algorithm to reduce the identification time. Also, some works [13, 14] revealed the characteristics of the physical layer like signal attenuation and shadowing effects through experimental means, which will have a serious impact on the identification performance of RFID systems.

In the following sections, we provide an overview of various prior anticollision mechanisms and analyze the limitations of them in a realistic RFID environment. Furthermore, to cope with the fundamental limitations of the prior anticollision solutions, we present a novel partitioning-based anticollision algorithm (PBAA) to identify tags under a mobile scenario with the unreliable channel. Our proposed PBAA optimizes the mathematical expression of the probabilities of collision, empty, and success slots in the mobile RFID scenario, thereby effectively combining the tag number estimation and frame length setting.

The rest of this paper is structured as follows. Section 2 conducts a survey and taxonomy of the existing tag reading protocols. A new tag reading strategy suitable for a mobile environment is elaborated in Section 3. Section 4 evaluates our proposed algorithm and compares the experimental results. Finally, the conclusions are summarized in Section 5.

2. Review of Tag Reading Protocol in RFID

The RFID tag reading or anticollision problem is to identify a given set of tags within the reader's coverage as fast as possible. Most of the literatures and industrial applications define this problem as follows: Given a batch of tags, each tag is equipped with a unique identifier (ID) and is affixed on object to being identified. The user expects to quickly and efficiently recognize all tags with a handheld reader or a fixed reader. Since the channel between the reader and tags is shared, thus, the anticollision algorithm or tag reading protocol is required. In what follows, we will firstly review the existing tag reading protocols. Then, we will conduct a taxonomy of the existing solutions from multiple perspectives. Table 1 summarizes the features of various tag reading solutions.

2.1. Probabilistic Algorithms. Among Aloha-based algorithms, the dynamic framed slotted Aloha (DFSA) algorithm is the most typical representative. In DFSA algorithm, any tag attempts to reply a query submitted by the reader with its ID information at a randomly selected time slot of a frame. The working principle of DFSA is illustrated in Figure 1.

The performance of DFSA is mainly affected by two factors: one is the accuracy of cardinality estimation and the other is the setting of frame length. As the accuracy of the cardinality estimation increases, the complexity of the algorithm will become increasingly higher. In fact, it is proved that the DFSA algorithms with high mathematical and computational costs are energy inefficient [15]. The tag identification strategy based on subframe observation (TES-FAS) [15] is proposed to reduce the overall error. However, since its empirical correction in estimation is only based on a stationary RFID system, the accuracy may be not sufficient for a mobile RFID system. The literature [16] proposed a schedule-based tag identification (SAC) approach, which determines the maximum moving speed of a tag that satisfies a given identification rate. The mechanism of SAC is to divide the tag set into smaller subsets and assign the different identification priorities for various subsets.

2.2. Deterministic Algorithms. Unlike probabilistic solutions, the deterministic algorithm uses tag ID to separate the colliding tag set instead of the random number. Among deterministic algorithms, a query tree- (QT-) based algorithm is a typical representative. In QT-based algorithm, the reader queries tags with a binary string (called query prefix), and the tags whose IDs match the query prefix will respond. Once detecting a collided string from the tags, the reader will probe again after appending the previous query prefix by 0 and 1, respectively. This query-and-append loop continues until all tags are successfully identified. An identification example of a QT-based algorithm is show in Figure 2. In the example, there are four tags with IDs of (001, 011, 100, 110) waiting to be identified in the reader's coverage. The reader firstly probes the tags with the prefix 0 and then probes tags with the prefix 1. As can be found in Figure 2, the reader consumes 8 slots in total to identify the above four tags. In the tag identification process, there are four empty queries and six collision queries. Many past works focus on how to effectively reduce the number of collisions and identification time in stationary scenarios. However, they may fail in various mobile scenarios. For example, some new arriving tags participate in the current reading process, if their IDs are less than that of tags that have been identified, they will be a missing read.

Based on QT-based strategies, some algorithms [10, 11] have been presented to optimize the reading performance for a tag moving scenario. The literature [10] presented a dynamic anticollision solution named dynamic collision tree (DCT) protocol which updates query string and splits the colliding tag set based on the first collided bit. In DCT, when a new arriving tag participates in the current reading process, two new nodes are inserted in the traversal tree according to the tag's ID. However, it is very difficult to know the IDs of tags in advance. In [11], the authors present a scheme based on bit monitoring to determine the presence of identified tags which stay in the reader's coverage, and an M -ary collision bit-based query mechanism is introduced to

Table 1: The characteristics of various tag reading protocols.

Type	Algorithm	Slot efficiency	Time efficiency	Cardinality insensitive	Tag moving	Resist capture effect	Resist path loss	Implement to EPC C1 GeN2
DFSA	TES-FAS [15]	High	High	✗	✗	✗	✗	Very easy
	SAC [16]	High	High	✗	✓	✗	✗	Medium
	DP [17]	High	Medium	✗	✓	✗	✓	Medium
	PBAA	Very high	Very high	✗	✓	✓	✓	Easy
QT	TH [9]	Very high	Medium	✓	✗	✗	✗	Difficult
	DCT [10]	Very high	Medium	✓	✓	✗	✗	Difficult
	EBD [11]	Very high	High	✓	✓	✗	✗	Difficult
TS	RBA [12]	High	Low	✓	✓	✗	✗	Medium
	SSRB [18]	High	Medium	✓	✓	✗	✗	Medium
Hybrid	ABTSA [19]	Very high	Medium	✗	✗	✗	✗	Difficult

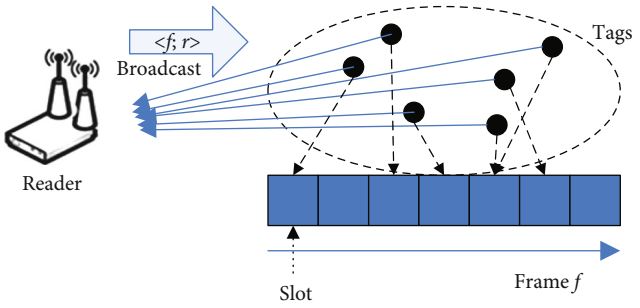


Figure 1: The working principle of DFSA algorithm.

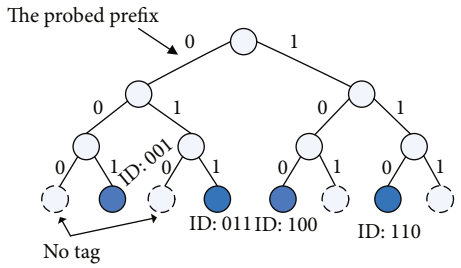


Figure 2: An illustrative example to identify four tags in the QT-based algorithm.

rapidly recognize the new arriving tags. Although the above solutions can improve the reading performance under the scenario that the tags are put on the moving conveyor belt, they are essentially dependent on bit tracking technology which is difficult to implement in the UHF RFID systems [8].

2.3. Hybrid Algorithm. There are many studies that attempt to integrate the characteristics of different algorithms to further improve the identification performance and thus develop various hybrid algorithms. The authors in [19] propose a hybrid algorithm named the tree slotted Aloha with adaptive binary splitting (ABTSA) which incorporates features of the Aloha-based and TS-based algorithms. ABTSA determines whether the current frame length is appropriate by monitoring the status of each slot. If it is appropriate, the TS-based algorithm is activated to resolve the collided slot; otherwise, the current frame length is readjusted. Although ABTSA can achieve a higher slot efficiency, it is time consuming due to identifying the collided tags in the TS manner.

In summary, the aforementioned anticollision algorithms can alleviate the tag collision to a certain extent; however, it suffers from serious challenges due to the complex realistic environment. All of the above solutions suppose that the communication link between the reader and tags is error free, without fully exploring many challenges in the actual situation, such as shadowing effect and blind spot. However, the instability of the practical communication links can seriously handicap the identification performance of RFID systems. Many works attempt to explore the impact of physical layer characteristics on reading performance through an experimental observation method. The authors in [13] verify the performance of RFID systems based on EPC C1 Gen2 standard using the testbed. Their research observations show that the physical and MAC layers should be considered in conjunction rather than separately as is done. The literature [14] presents the reliability of a method for estimating in a real measurement environment. However, in the work, only the capture effect is considered, and the path loss and other radio propagation issues are not fully explored. The authors in the literature [17] tested that mobile RFID tag reading is significantly affected by the influence conditions such as speed of moving tags and angle of antenna and further verified that physical and MAC layers should be considered at integrally. In the following, an efficient tag reading protocol is proposed to overcome the above challenges. The proposed solution is aimed at achieving both slot efficiency and time efficiency in a mobile environment.

3. A New Design of Efficient Reading Strategy under an Unreliable Channel

3.1. System Model. There are two communication links in the UHF RFID system, namely, downlink and uplink, respectively. Among these two links, the uplink represents the communication link from the tag end to the reader end, while the downlink represents the communication link from the reader end to the tag end. The international standard EPC C1 Gen2 of UHF RFID has strict specification on the timing of the communication links, which is shown in Figure 3. In the downlink, the reader continues to send a carrier wave to the tags, and for one thing, each tag absorbs energy from the RF wave to meet its own energy consumption. For another, each tag parses the command data included in the carrier wave sent by the reader in order to respond. In the uplink, the tag replies to the reader based on the parsed command data, and the response data is modulated onto the reflected carrier using FM0 or Miller coding. This modulation method is called backscatter modulation. The tag implements data transmission on the reverse link by backscatter modulation. In order to successfully read a tag in a UHF RFID system, two prerequisites need to be met. First, the receiving power $P_{(\text{tag},rx)}$ of the tag needs to be higher than the sensitivity P_{st} of the tag itself. Only in this way can the tag absorb enough energy to meet its own needs. Secondly, the receiving power $P_{(\text{reader},rx)}$ of the reader is higher than the sensitivity P_{rt} of the reader itself. Only in this way can the reader correctly decode the signals returned by the tags. The sensitivities of the tag or reader represent the minimal energy required to maintain their own work. As can be seen in Figure 1, there are three possible outcomes for a given time slot: no reply (empty slot), single reply (singleton slot), and multiple replies (collision slot). However, in the actual identification process, the communication link between the reader and tags is not always steady. For example, Figure 4(a) illustrates the scenario of identifying moving tags. In such a case, the reader's identification of tags will become extremely complicated, because of the existence of factors such as object moving, multipath, and capture effect. These factors make the originally assumptions about channel ideals no longer realistic. Figure 4(b) shows the translation relationship from a time slot in a stationary environment to a time slot in a more practical environment. Herein, P_{trR} represents the probability that a tag can acquire the reader's command correctly, and P_{Rrt} means the probability that the reader is able to parse the response from the tags. The $P_{c->s}$ and $P_{c->e}$ denotes the probability that original multiple replies in a steady channel model transfer into a singleton slot and an empty slot, respectively, which can be expressed as the function of P_{Rrt}. α denotes the occurrence probability of the capture effect. Under such conditions, the conventional tag reading solutions used in the stationary model will no longer be applicable. Therefore, it is critical to develop reading strategies based on a mobile model to tackle the tag collision problem.

3.2. Cardinality Estimation under the Unreliable Channel. In an actual RFID application paradigm, physical layer features such as multipath and capture effect, path loss, and attenuation all occur probabilistically. In order to characterize these features, we built a dynamic tag identification model and design the corresponding tag number estimation method based on this model. Assuming that there are n tags in the radiation range of an RFID reader, the reader uses an initial frame of length F to identify them. Accordingly, in our proposed probabilistic model, tags do not respond to the reader 100%, so in a frame, the number of tags that respond is $\hat{n} = n \cdot (P_{trR})^2$. Similarly, the probability P_m that m tags simultaneously select a time slot under our model can be calculated using binomial distribution. Combining the translation relationship described in Figure 4 and the existing analytical method, the reader can estimate the parameters of n, P_{trR}, P_{Rrt} and α. We can establish a scenario with only a single reader and single tag and then use an experimental testing method to obtain the values of these parameters. The specific method is as follows: we can place the tag at different distances d from the reader antenna. We first set the frame length F to 1, then let the reader attempt to read the tag k times repeatedly, and record the number of times m_1 that the tag is successfully identified. Then, we find a ratio as $r_1 = k_1/k$. Secondly, we increase the F value from 1 to a large number such as 256 and also let the reader independently recognize the same tag k rounds and obtain the number of times reading k_2 that the tag is successfully identified. Similarly, we can obtain a ratio as $r_2 = k_2/k$. In what follows, we can setup a scenario with only a single reader and multiple tags and set the frame length F to 1. We allow the reader to independently recognize a tag k rounds and obtain the number k_3 of times that the tag is successfully identified. Accordingly, we can obtain a ratio as $r_3 = k_3/k$. Hence, the environment parameters P_{trR}, P_{Rrt}, and α are expressed as

$$\alpha = \frac{r_3}{r_1} = \frac{k_3}{k_1}, \tag{1}$$

$$P_{trR} = \frac{r_2}{r_1} = \frac{k_2}{k_1}, \tag{2}$$

$$P_{Rrt} = \frac{k_1}{k_{21}} \sqrt{\frac{k_1}{k}}. \tag{3}$$

We can precalculate and save the values of these parameters for different distances. As a result, we can reduce the computational complexity in the tag number estimation phase.

3.3. The Partitioning-Based Identification Phase. According to the mentioned estimation method, the reader can get an approximate value of the number of tags n, P_{trR}, P_{Rrt}, and α. Instead of assigning a single full frame for the unread tags in conventional DFSA algorithms, the proposed PBAA solution divides unread tags into N subsets averagely according to the estimated n, P_{trR}, P_{Rrt}, and α. Then, the reader conducts the corresponding identification process on individual subsets. Each process is named a partitioning-based identification phase. The workflow of the partitioning-based

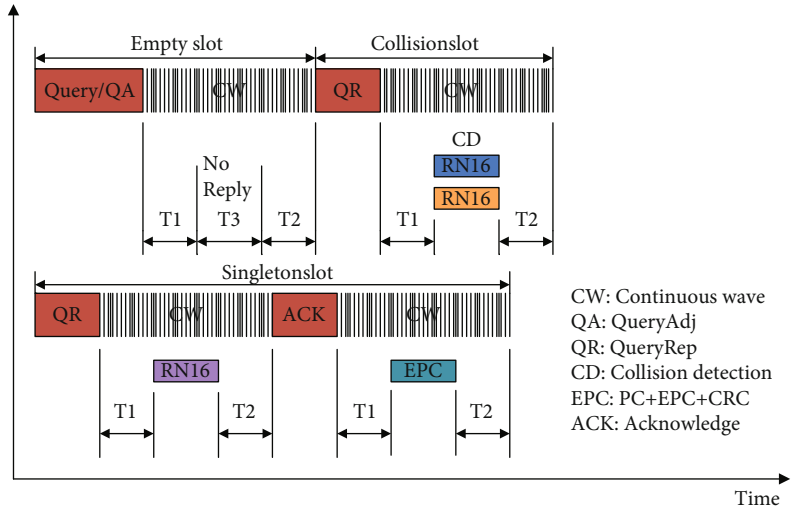


Figure 3: The link timing specified by EPC C1 Gen2.

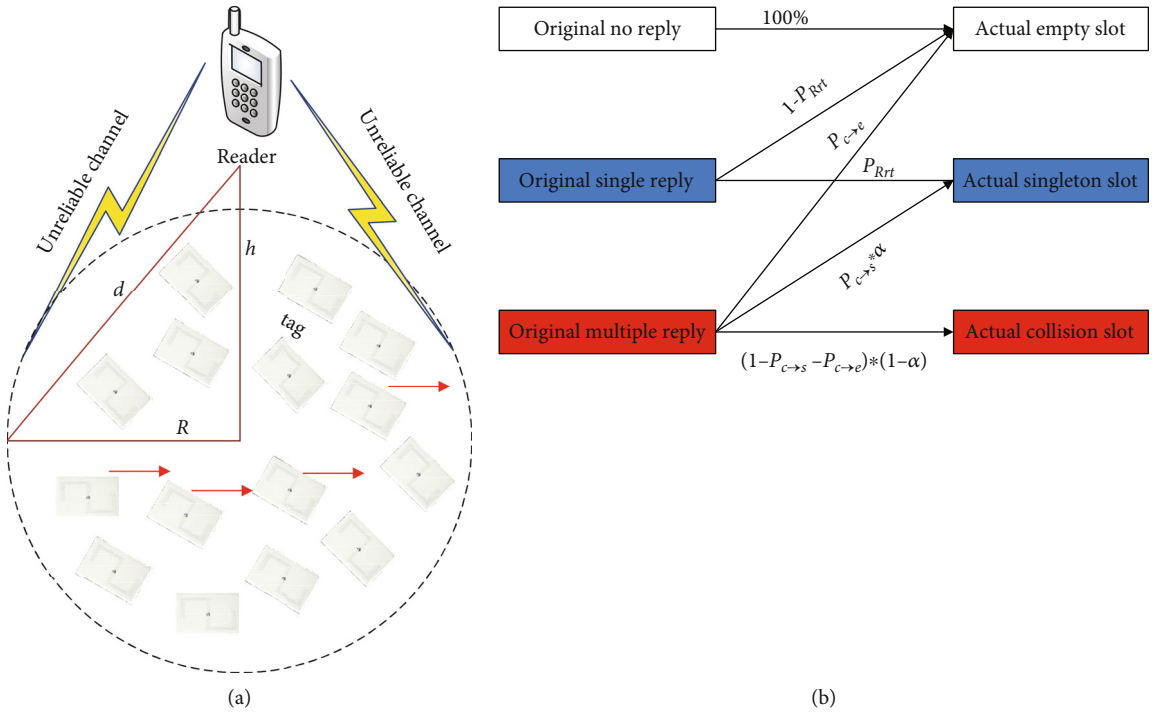


Figure 4: The tag reading under a mobile scenario.

identification phase is described as follows. The reader will fix the frame length to 2 to initiate an identification process for each subset, which is called the IIP process. The detailed working process of IIP is shown in Figure 5, where F means the initial frame size, N_{id} means the number of identified tags, and C means the number of collision slots. The reader triggers the reading process slot by slot and records its feedback results. If no response is monitored, the reader will increase the number of empty slots by one. If multiple responses are monitored, the reader will trigger the second slot and increase the number of collision slot by one. When a single response is monitored, the reader will directly read the tag and increases the number of singleton slot by one. The reader will iteratively execute the above identification process until all tags are completely identified. Since the above process does not need to estimate the number of unread tags, the problem of misreading will be alleviated due to the unreliable channel. Thus, with the introduction of IIP in the reading protocol, the identification efficiency of RFID in a mobile scenario will be significantly improved.

4. Evaluation Discussion

4.1. Simulation Setup. In this section, we evaluate the performance of PBAA and compare it with prior arts including TES-FAS [15], RBA [12], DP [17], and SAC [16] over

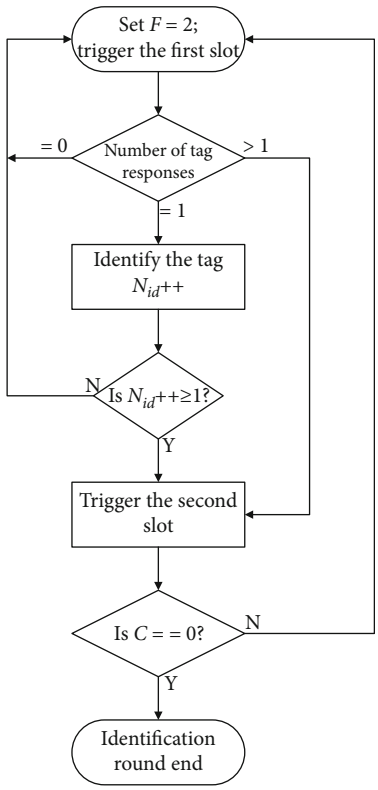


Figure 5: The detailed working principle of the IIP process.

extensive Monte Carlo simulations. The reader's coverage is a circle with a radius of 6 meters. In our simulations, we consider a scenario of reading moving tags on the conveyor, where the distance between the reader and the horizontal line is 2 meters. To ensure the robustness of the results, in the simulation, 1000 iterations are run to obtain each result [8, 15, 16, 20]. The parameters used in MATLAB simulation are listed in Table 2.

In order to better measure the performance of our proposed algorithm, we use the following two metrics: (1) slot efficiency, defined as the number of singleton slots over the total number of slots to identify all tags and (2) time efficiency, which is widely used in literatures and it can better reflect the performance of an algorithm in the time dimension.

We compared our proposed algorithm PBAA with other advanced algorithms in detail between the two performance evaluation metrics described above. In order to provide a more comprehensive and fair comparison, we choose some representatives in both TS-based and Aloha-based algorithms. Specifically, RBA [10] is the representative of TS-based algorithm. The representatives of Aloha-based algorithms are TES-FAS [15], SAC [16], and DP [17].

4.2. Results on Numerous Metrics. Figure 6(a) depicts the slot efficiency of comparative approaches under scenario A. The size of tag cardinality is between 100 and 1000 in steps of 50. The frame length is initialized as 64. Observed from the results, the proposed PBAA can always maintain a more stable and higher performance. The reason why the proposed

Table 2: The link parameter setting between the reader and tag communication.

Experimental scenario	A	B
R- > T modulation	PR-ASK	DSB-ASK
R- > T coding	PIE	PIE
Tari (us)	25	6.25
PW (us)	12.5	3.13
RTcal (us)	62.5	15.63
TRcal (us)	85.33	20
DR	21.33	8
T- > R modulation	Miller-4	FM0
TRExt	1	1
BLF (kHz)	250	400
Data rate (kbps)	62.5	400
Tag moving speed	1 m/s	2 m/s

anticollision algorithm PBAA can achieve such improvement is twofold. For one thing, the proposed estimation strategy can obtain the accurate estimation results of the number of tags to be identified n_{est}, the probability that the tag can correctly receive the reader's command P_{trR}, and the occurrence probability of capture effect α. For the others, the proposed IIP process can alleviate the negative impacts on reading performance caused by path loss and capture effects. The slot efficiency of RBA is higher than that of other Aloha-based algorithms because it allows all of unread tags to respond to the reader at the following slots. It somewhat reduces the impact of channel uncertainly on slot efficiency. On the contrary, Aloha-based algorithms are more susceptible to factors such as path loss and capture effect. Thus, their slot efficiency is lower than that of tree-based algorithm. Figure 6(b) plots the slot efficiency under scenario B. As can be observed, some algorithms show discrepant performance. For example, the average slot efficiency of TES-FAS is close to DP and SAC when tag moving speed is 1 meter per second. However, its slot efficiency is lowest when the moving speed is increased to 2 meters per second. The reason is that the mechanism of TES-FAS including cardinality estimation and frame length setting is optimized based on ideal channel conditions. As the channel conditions deteriorate, the TES-FAS is increasingly unable to adjust an appropriate frame length to fit the unread tags, resulting in sharply performance degradation. Since the frame length setting and cardinality estimation are based on variable channel conditions, the PBAA can always maintain the better slot efficiency compared to reference solutions.

To compare the identification performance of various solutions from a time perspective, Figure 7 illustrates the time efficiency of all methods under different scenarios. Where scenario A is in low-rate mode (date rate ≤ 80 kbps), scenario B is in high-rate mode (date rate ≥320 kbps). It can be seen from Figure 4(a) that the performance ranking of algorithms from the highest to the lowest is PBAA, SAC, DP, TES-FAS, and RBA. Such ranking is different with the ranking observed in Figure 6(a). For example, RBA is

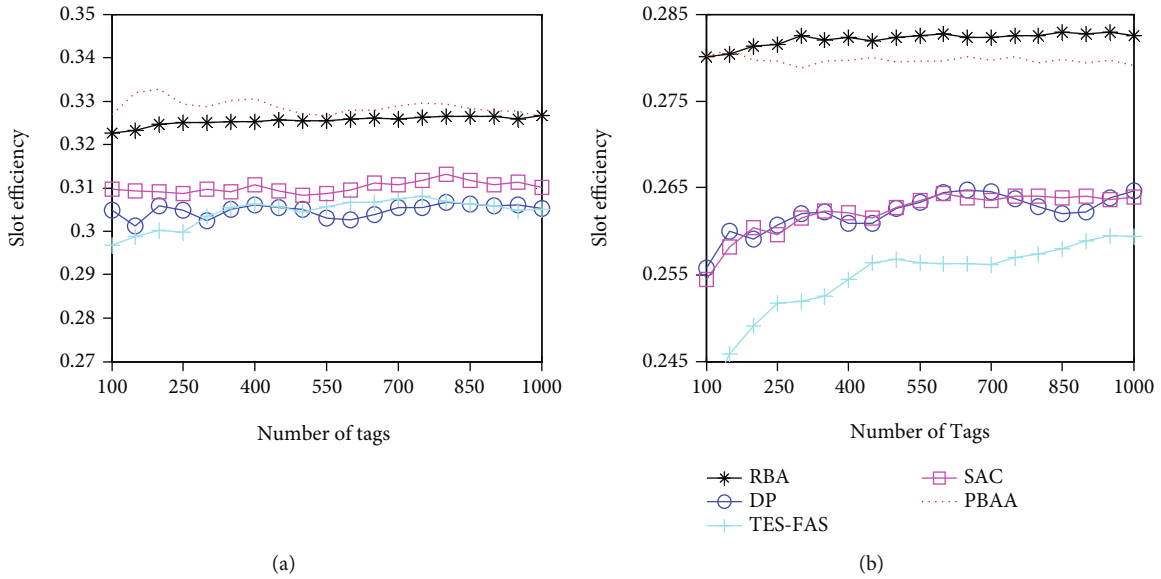


Figure 6: Simulation results: comparison of various methods in slot efficiency.

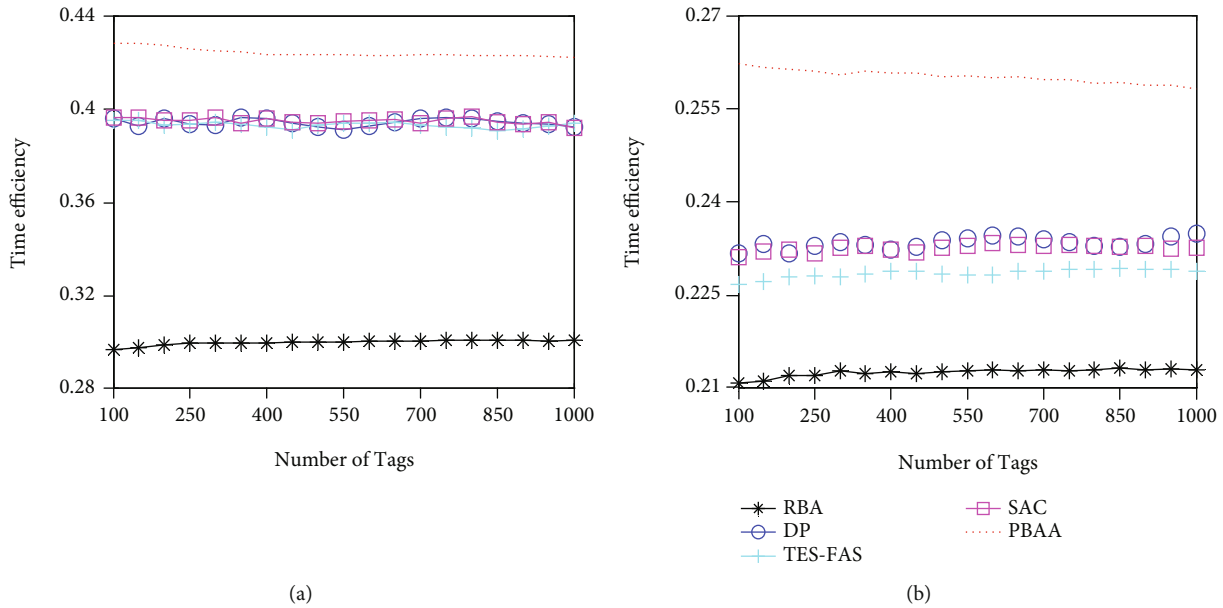


Figure 7: Simulation results: comparison of various methods in time efficiency.

superior to SAC, DP, and TES-FAS in terms of slot efficiency. However, its time efficiency is lowest. The reason is analyzed in Section 2. The time efficiency metric takes into account both the slot type and the slot duration. Thus, our proposed PBAA performs exceptionally well in this evaluation metric. The reasons are summarized as follows. When the reader uses the PBAA algorithm to identify the tags, the number of empty slots consumed is much greater than collision slots, and the occupied time duration of each empty slot is much shorter than collision slot. Explicitly, the time efficiency should increase as the data rate increases. However, the opposite is true. Observed from Figure 7(b), the performance of all algorithms deteriorates under scenario B where the moving speed is 2 meters per second. The reason is as follows. As the moving speed increases, the tag quantity that can be read in a frame drops significantly due to the deteriorative channel, causing many time slots to be wasted and thus prolonging the total identification time. Therefore, the time efficiency will be degraded. For example, the TES-FAS provides the constrained formula between the probabilities that a slot is collided or empty based on the consumption that the communication link is ideal. However, such constraints are not suitable for the nonideal case. As the P_{trR} decreases, the performance deterioration will get worse. We can see from the Figure 7(b) that the time efficiency of TES-FAS is lowest. Unlike conventional DSFA algorithms,

our proposed PBAA can estimate the P_{trR} and α, thereby eliminating their negative impact on mobile RFID tag identification performance. Both benefiting from the designed estimation strategy and partitioning-based identification phase, our proposed PBAA can achieve the highest performance in terms of slot efficiency and time efficiency under mobile RFID tag identification scenarios.

5. Conclusion

In this paper, we make the following key contributions. First, we analyze and discuss the drawbacks of various tag reading protocols especially on the performance of the MAC layer under mobile scenarios with the unreliable channel. Different from conventional DFSA approaches in a stationary scenario, the PBAA can adaptively obtain an accurate estimation value of the environment parameters and hence optimize the reading performance. In our view, our proposed PBAA solution makes RFID capable of adapting to mobile scenarios with the unreliable channel and meeting the IoT application requirements of time efficiency and high slot efficiency. The new results on the reading performance obtained in this paper can provide a valuable reference and guideline for the practical RFID system under the mobile environment.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Natural Science Foundation of China (Nos. 61802196 and 61972207), the Natural Science Foundation of Jiangsu Province (No. BK20180791), and the Natural Science Foundation of Jiangsu Higher Education Institution of China (No. 17KJB510036). This work is also supported in part by Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology.

References

[1] A. al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] S. M. Köppen, A. K. Bashir, and Y. Jin, "Advanced ICT and IoT technologies for the fourth industrial revolution," *Intelligent automation & soft computing*, vol. 26, no. 1, pp. 83–85, 2020.

[3] C. T. Poomagal, G. A. Sathish Kumar, and D. Mehta, "Multi level key exchange and encryption protocol for Internet of Things (IoT)," *Computer Systems Science and Engineering*, vol. 35, no. 1, pp. 51–63, 2020.

[4] R. Das, *RFID Forecasts, Player and Opportunities 2018-2028*, IDTechEx, 2018.

[5] X. Liu, K. Li, A. A. Liu et al., "Multi-category RFID estimation," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 264–277, 2017.

[6] G. Dong, W. Zhang, S. Xuan, F. Qin, H. Tan, and J. Wang, "An improved binary search anti-collision protocol for RFID tag identification," *Computers, Material s & Continua*, vol. 65, no. 2, pp. 1855–1868, 2020.

[7] X. Wang, M. Zhang, and Z. Lu, "A frame breaking based hybrid algorithm for UHF RFID anti-collision," *Computers, Material s & Continua*, vol. 59, no. 3, pp. 873–883, 2019.

[8] J. Su, Z. Sheng, L. Xie, G. Li, and A. X. Liu, "Fast splitting-based tag identification algorithm for anti-collision in UHF RFID system," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2527–2538, 2019.

[9] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for RFID identification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 796–809, 2015.

[10] X. Jia, M. Bolic, Y. Feng, and Y. Gu, "An efficient dynamic anti-collision protocol for mobile RFID tags identification," *IEEE Communications Letters*, vol. 23, no. 4, pp. 620–623, 2019.

[11] L. Zhang, W. Xiang, and X. Tang, "An efficient bit-detecting protocol for continuous tag recognition in mobile RFID systems," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 503–516, 2018.

[12] J. S. Li and Y.-M. Huo, "An efficient time-bound collision prevention scheme for RFID re-entering tags," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1054–1064, 2013.

[13] M. Buettner and D. Wetherall, "An empirical study of UHF RFID performance," in *Proceedings of the 14th ACM international conference on Mobile computing and networking - MobiCom '08*, pp. 223–234, New York, NY, USA, September 2008.

[14] P. Solic, J. Maras, J. Radic, and Z. Blazevic, "Comparing theoretical and experimental results in Gen2 RFID throughput," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 349–357, 2017.

[15] J. Su, Z. Sheng, A. X. Liu, Z. Fu, and Y. Chen, "A time and energy saving-based frame adjustment strategy (TES-FAS) tag identification algorithm for UHF RFID systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 2974–2986, 2020.

[16] W. Zhu, J. Cao, H. Chan, X. Liu, and V. Raychoudhury, "Mobile RFID with a high identification rate," *IEEE Transactions on Computers*, vol. 63, no. 7, pp. 1778–1792, 2014.

[17] L. Xie, B. Sheng, C. C. Tan, H. Han, Q. Li, and D. Chen, "Efficient tag identification in mobile RFID systems," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, San Diego, CA, USA, March 2010.

[18] R. Jayadi, Y.-C. Lai, and C.-C. Lin, "Efficient time-oriented anti-collision protocol for RFID tag identification," *Computer Communications*, vol. 112, pp. 141–153, 2017.

[19] H. Wu, Y. Zeng, J. Feng, and Y. Gu, "Binary tree slotted Aloha for passive RFID tag anticollision," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 19–31, 2013.

[20] S. Kaur and V. K. Joshi, "Hybrid soft computing technique based trust evaluation protocol for wireless sensor networks," *Intelligent Automation and Soft Computing*, vol. 26, no. 2, pp. 217–226, 2020.