IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2019, accepted January 24, 2019, date of publication January 31, 2019, date of current version February 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895899

A Cost-Efficient Greedy Code Dissemination Scheme Through Vehicle to Sensing Devices (V2SD) Communication in Smart City

HAOJUN TENG[1], WEI LIU[2], TIAN WANG[3], ANFENG LIU[1], XUXUN LIU[4], (Member, IEEE), AND SHAOBO ZHANG[5]

[1]School of Information Science and Engineering, Central South University, Changsha 410083, China
[2]School of Informatics, Hunan University of Chinese Medicine, Changsha 410208, China
[3]Department of Computer Science and Technology, Huaqiao University, Xiamen 361021, China
[4]College of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China
[5]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

Corresponding author: Anfeng Liu (afengliu@mail.csu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China, under Grant 61772554.

ABSTRACT Recently, the vehicle-to-everything (V2X) paradigm is attracting more attention from both academia and industry. In the smart city, there are a huge number of roadside smart devices (RSDs) undertaking various sensing and monitoring tasks, and they collect information among V2X devices for various applications. With the software-defined technology applying into RSDs, one of the challenging issues is how to update the software of RSDs in a fast and low-cost way. We argue that recruiting a large number of vehicles to disseminate the update code for such RSDs through vehicle-to-sensing devices communications technology is an effective method. A cost-efficient greedy code mules selection scheme (CGCSS) is proposed to disseminate code to a huge number of RSDs in the smart city. In CGCSS, a task is defined as the process of transmitting the update code from the mobile code station (MCS) to RSD. So, the goal of CGCSS is to recruit an appropriate number of vehicles to finish the tasks with low cost and high coverage. A measure function is proposed to take the historical trajectories and cost into account. Therefore, the vehicles with high task completion possibility and low price will be selected as code mules (CMs). Then, a high-performance MCS deploy scheme (HMDS) is proposed to select the optimized MCS positions according to the movement and frequency of the CMs to optimize the performance of the system. Finally, extensive experiments using the real trajectory dataset have been done. The results show the better performance of CGCSS than the basic greedy code mule select scheme and the random code mules select scheme in terms of completion rate, average price, and cost-performance ratio, and the results confirm the validity of the proposed HMDS as well.

INDEX TERMS Vehicle to everything, code dissemination, vehicles as code mules, recruit vehicles, low cost.

I. INTRODUCTION

With the development of microprocessor technology, component integration is getting higher and higher, making the functions of microprocessor components rich and powerful [1], [2]. This has led to a greater development of microprocessor-based sensing devices. These sensing devices with rich features are deployed in wide kinds of IoT applications. They are used to sense and acquire different kinds of data to achieve the goal of ubiquitous perception [3]–[5]. A vehicle is a mobile sensing and communication platform carrying a lot of equipment. It contains a variety of sensing devices that can sense itself and the environment, such as sensing devices for speed, vibration, temperature, humidity, and acceleration [2], [6], [7]. At the same time, the vehicle is also equipped with various kinds of communication equipment. Vehicle to everything (V2X) paradigm is attracting more attention from both academia and industry [8], [9]. In V2X, while connecting all the devices (motor-vehicle, non-motor-vehicle, pedestrian, etc.) on the road and

The associate editor coordinating the review of this manuscript and approving it for publication was Zhi Liu.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

roadside, the real-time information (speed, accelerate, route, etc.) among V2X devices can be collected and shared for data exchanging, automatic piloting and intelligent traffic control [10], [11]. On the other hand, more and smarter sensing devices are deployed in a wider range of applications to monitor various objects, events, and perceptions to achieve the goal of so-called Earth-aware [12]–[16]. And roadside smart devices (RSDs) refer to a large number of sensing devices with short-range wireless communication on the side of the road [7]–[10], [17]. The work of these sensing devices is monitoring events and sensing environmental features on the side of the road [18]–[21]. On the other hand, the main task of RSDs is to communicate with mobile vehicles to form a mobile vehicle network [2], [6]–[10], [17].

With the emerging of the smart city, more and more objects and events need to be monitored, and the scope of monitoring is becoming wider and wider [22]–[25]. Important key locations and monitoring objects have already been implemented the deployment of fixed sensing networks, such as wireless local area sensing networks [26]–[28]. However, there are some scenarios that it is not possible to deploy a fixed monitoring network [2], [6], [8], [17]. On the one hand, because the distribution of objects that need to be monitored is wide and the density of that is sparse [6], [7]. Therefore, it is hard to deploy a fixed network due to high costs [2], [29], [30]. On the other hand, it is not feasible to deploy a fixed network for technical reasons. These technical reasons include: those objects that need to be monitored change their locations from time to time, or sensor devices cannot be deployed due to physical constraints [31], [32]. But mobile vehicular network can play a more important role in the construction of the smart city [2], [6]–[10]. In order to realize the goal of the smart city, it is necessary to deploy a large number of sensing devices in the city to realize the intelligent perception and monitoring of the city. Although some monitoring networks have been implemented in the city, such as surveillance camera equipment deployed on the road, the monitoring devices currently deployed are very limited in terms of the monitoring range [23]. This camera equipment is often only deployed at important intersections or important sections, but there is no monitoring equipment on more extensive sections. More importantly, current monitoring equipment is often based on image monitoring, but physical characteristics and biological characteristics of different monitoring objects have been hardly achieved [33]–[35].

Now in the smart city, smart sensing devices are introduced continuously. For example, the newly-appeared smart garbage can, which is equipped with sensing devices that can sense the remaining capacity of the garbage can. When the remaining capacity of the garbage can is insufficient, it will send a message to sanitation worker to empty the garbage can [36]. And there are some smart street lights, which is equipped with sensing devices that sense the status of the street lights. If the control center can obtain the data of the sensing devices, it can monitor the status of the street lights and response to the change of the status. The more sensing devices deployed, the more intelligent perception can be achieved [36]–[38]. For example, in order to ensure the high quality of plants, the soil is monitored by deploying sensors and watered or fertilized based on the results of the monitoring [16]. And more ready-to-deploy and location-changing sensing devices are emerging. How this kind of sensing devices exchange data with low cost becomes an important issue [39], [40]. In this regard, roadside smart devices (RSDs) can play an important role. On the one hand, in the smart city, the distribution of roads is very dense. Therefore, the RSDs are widely distributed in the city, and RSDs can be used as access points for opportunistic communication. On the other hand, a large number of vehicles in the city can be interconnected and exchanged data [2], [6]–[10]. At the same time, RSDs are also a kind of sensing devices, which can effectively monitor and sense the surrounding environment [17]. Since the RSDs are located on the side of the road, it can act as a relay node to connect a larger number of sensing devices deployed near the road to the mobile vehicle network for data exchange [2], [6], [8], [17].

Another important factor in promoting the development of the above-mentioned ubiquitous perception applications is the widespread use of soft-define technology in sensing devices [41]. Soft-define devices are the devices that redesign hardware and software for soft-define technology, which allows the function of devices to be flexibly configured via software [7], [13], [31]. Soft-define devices technology provides a broader space for the development of sensing devices. As mentioned earlier, a large number of sensing devices are deployed in the smart city. Due to the rapid development of current applications, many applications need to software update in every a few months. Before the use of soft-define devices, these large number of fixed-function sensing devices can only be discarded and redeployed with new features of the sensing devices. Due to the high cost of network deployment, in such cases, it will cause a lot of waste. New applications will not start before the new sensing devices are redeployed. Therefore, it will also waste a long time. Moreover, repeated deployment of sensing devices can also cause environmental pollution. The emergence of soft-define devices brings new opportunities to solve these problems. Soft-define sensing devices can get new features by getting a new code. The new application can be started immediately as long as the soft-define device completes the adjustment of the function, which saves deployment time and cost [7], [13], [17], [31]. This is of great significance in promoting the development of the Internet of Things (IoTs).

However, transmitting the code to the deployed sensor devices with a low-cost and fast way is a very challenge issue [7], [13], [17], [31]. In the above application scenarios, there are some unique cases that did not appear in before. (1) Although some sensing devices may have the ability to connect to the Internet, more sensing devices do not have this ability due to the limit of cost [42]. Therefore, it is challenging to find an appropriate way to disseminate the code to sensing devices. (2) The low cost and high-efficiency requirements

make the code dissemination scheme design more challenging. The number of RSDs (or sensing devices) deployed in the city is very large and the deployment area is very wide. In addition, RSDs are deployed for different applications, so the requirements for update time, delay requirements are different. The effective code dissemination mechanism is a challenging issue [7], [13], [17].

Due to the wide distribution of sensing devices, deploying a dedicated wired network can lead to excessive costs. And communication method using cellular network will limit the application because each sensing device needs a SIM card and the cost of data fee is much higher than the cost of the devices [7], [36]. At the same time, the deployment location of many sensing devices changes with the changes of the application requirements, so its high dynamics leads to the lack of effective data exchange channels. For example, it is often necessary to deploy a large number of sensing devices for short-term applications such as road construction and greening projects to obtain monitoring data [17], [36]. These huge number of sensing devices are facing huge challenges. Some researchers have proposed some solutions and measures. For how to collect a large amount of data generated by these sensing devices, Bonola *et al.* [36] discussed the possibility of collecting sensing data from sensing devices through mobile vehicles. They experimented by using the actual trajectory data of the vehicle in the city, and the results showed: Due to the huge number of vehicles in the city, the trajectory of their movement can cover most areas of the city with a high probability. When the mobile vehicle passes near these sensing devices (ie, within the communication range) [43], these sensing devices can transmit the perceived data to the vehicle. And the mobile vehicle can transmit the collected data to the data processing center when passing through the data processing center [36]. This method of data exchange is called opportunistic communication. Because the mobile vehicle just dropped by to exchange the data on the way to their destination. Therefore, this method is low-cost and scalable, can be used in a wide range. It can be said that using vehicles as data mules is a low-cost, effective way [36].

Obviously, code dissemination is an inverse process of data collection. It is a natural choice to use vehicles as data mules for code dissemination [7], [13], [17]. However, we still have the following issues of further study by using the mobile vehicle to carry out code dissemination in the form of opportunistic communication.

(1) In the previous scheme that used vehicles as code mules for code dissemination, it was assumed that vehicles voluntarily act as mules for code dissemination. Thus, the main research objectives of these vehicles as code mules based code dissemination schemes have been the feasibility of vehicles as code mules, the time required for code dissemination, and how to ensure maximum coverage of code dissemination. But in practice, it is not easy to assume that vehicles can voluntarily act as mules. Therefore, the scheme proposed in this paper does not assume that vehicles are voluntarily acting as mules, but need to pay a certain amount of money to effectively complete the code dissemination. Obviously, the scheme proposed in this paper is more in line with the actual situation.

(2) In the previous code dissemination scheme, since vehicles do not need to pay, each vehicle can be used for code dissemination. In practice, vehicles need to consume certain resources, such as communication bandwidth, electrical resources, etc. for code dissemination. Therefore, if the rewards are not given to the vehicles, most of the vehicles will not participate in the code dissemination, which makes the scheme perform poorly in practice. From a practical point of view, the code dissemination party should pay a certain reward to the vehicles to make them willing to act as mules. On the other hand, the code dissemination party should save the cost as much as possible and maximize the coverage of code dissemination. Obviously, there is actually a paid relationship between the code dissemination party and the vehicles. These have not been considered in previous studies. Although these early research results technically indicate that this is a feasible scheme. In practice, it is necessary to consider the actual benefits of code dissemination in order to achieve good results.

Based on the above analysis, we argue that recruiting a large number of vehicles to disseminate the update code to RSDs through vehicle to sensing devices (V2SD) communications technology is an effective method even considering the interests of all parties. In this paper, a Cost-efficient Greedy Code Mules Selection Scheme (CGCSS) is proposed to disseminate code to a large number of RSDs in the smart city. The main innovations of this paper are as follows:

(1) The CGCSS takes a mechanism based on recruiting vehicles to dissemination code which is more in line with the actual situation of vehicle to sensing devices (V2SD) communications. In CGCSS, a task is defined as the process of transmitting the update code from mobile code station (MCS) to RSD. And the goal of this paper is to complete n tasks with the lowest possible cost and as less time as possible to recruit optimized vehicles. In order to achieve the goal of the scheme, the CGCSS innovatively proposes to predict the future trajectory of vehicles based on the historical trajectory similarity of vehicles. And the frequency is used to indicate the probability of completing the specified task. The above indicators are used to select those vehicles with low price, and high task completion possibility to complete the code dissemination. Therefore, the scheme can complete the code dissemination task faster and with low-cost.

(2) A High-performance MCS Deploy Scheme (HMDS) is proposed to select the optimized mobile code station (MCS) positions according to the movement law and movement frequency of the CMs to optimize the performance of the system. In HMDS, first, analyze the historical trajectory of the code mules (CMs) selected by the Cloud Service Platform (CSP); then, the area which the selected CMs pass through is counted as a candidate area for deploying the MCS; next, the frequency of CMs passing through each candidate region is counted; finally, the areas with top frequency are selected

to be the location of MCS for optimizing the efficiency of the code dissemination.

(3) Finally, we experimented with our proposed CGCSS using real trajectory data of taxis in Rome city. Our extensive simulation experiments demonstrate that the effectiveness of the CGCSS. The results show the better performance of CGCSS than BGCSS and RCSS in terms of completion rate, average price and cost-performance ratio. And the results confirm the validity of the proposed HMDS as well.

The rest of the paper is organized as follows. We review related work in Section II. In Section III, we describe the system model and formulate the problem of our data collection scheme. Sections IV present the details of Cost-efficient Greedy Code Mules Selection Scheme (CGCSS). We evaluate the proposed CGCSS via simulations in Section V. We conclude the paper in Section VI.

II. RELATED WORK

A. SMART CITY AND V2X

With the introduction of the concept of the smart city, there are more and more Internet of Things applications especially the wireless sensor networks applications, and there are many application scenarios in road traffic, such as intelligent traffic signal control, traffic congestion warning, traffic accident detection, driving assistance, etc. Nowadays, with the introduction of V2X (Vehicle-to-Everything) technology, the possibility of realizing a smart city can be accelerated. Aissaoui *et al.* [44] proposed a Real-Time Traffic Monitoring System based on V2X Communications, the authors used the cluster-based V2X traffic data collection mechanism to collect data more reliable and less overhead. More than that, V2X Communication can help to achieve autonomous driving. Hobert *et al.* [45] analyzed the current V2X communication standards from ETSI for missing features, introduced a core functional design of cooperative autonomous driving and addressed the required evolution of communication standards in order to support a selected number of autonomous driving use cases.

One of the mainstreams is LTE-based V2X communication, the LTE cellular network especially the 5G technology coming in the future can meet the strict requirements of transmission speed and communication delay when the vehicle driving in the high speed. But now the communication via LTE cellular network is based on the base stations. And frequent base station switch may reduce the stability of network links while the vehicle is driving. Actually, the LTE based V2X communication only supports the V2I communication and lack of the communication of V2V. Chen *et al.* [46] proposed LTE-V, a TD-LTE-Based V2X solution. They introduced an LTE-V-direct mode into traditional LTE architecture. The LTE-V-direct is a new decentralized architecture which can provide short-range direct communication with low latency and high reliability. But the security of data transmission is the top requirement. Because data leakage may impact the life safety of the driver. Ahmed and Lee [47] proposed privacy-preserving security for LTE-based V2X service using asymmetric encryption. Their scheme is scalable while fulfilling basic wireless message security requirements.

In V2X based Vehicular Ad hoc Networks (VANETs), data dissemination is an important technology. The topology of VANETs changes very quickly due to the high-speed mobility of vehicles. How to disseminate data more reliable is a challenging issue. Oliveira *et al.* [48] propose Adaptive Data Dissemination Protocol, which providing reliability to data dissemination. Their protocol can dynamically adjust the beacon periodicity and reduce the number of beacons by using different mechanisms. This protocol is efficient in a sparse scenario but not perform well in high density scenario. In fact, most protocols have the problem of lack of scalability in high-density scenarios. To solve this problem, paper [49] provides a scalable broadcast approach for data dissemination in multi-hop VAENT that relies on traffic regime estimation but without extra communication overhead.

In general, V2X communication is used to serve for driving safer and easier. But V2X communication is also used to help the data collection or data delivery of the IoT applications in the smart city. In this paper, we propose a code dissemination architecture by using the V2X communication, which can make the software update of sensor device possible in the sparse deployment condition.

B. CODE DISSEMINATION

For code dissemination, it is very common in the traditional wireless sensor network in order to do some update (also called wireless reprogramming) for wireless sensor networks such as changing topology, updating parameters, etc. In traditional wireless sensor networks, the deployment scope is relatively small. How to perform Code Dissemination more quickly and reliably is the key issue. There are some papers propose the optimization methods for code dissemination, such as [50] and [51]. An SYNAPSE++ system for over the air reprogramming of WSNs are presented in [50], by using the Fountain Code, they can realize better performance comparing to the Deluge. And Dong *et al.* [51] proposed an Efficient Code Dissemination protocol, they can reduce the code dissemination time by changing the packet size according to the link quality, using an accurate sender selection algorithm to mitigate transmission collisions and employing a simple impact-based backoff timer to coordinating senders.

But the code dissemination methods mentioned above are all limited in a small scope. The code dissemination task can be finished within a limited area within multi-hop transmission. But our code dissemination scheme is a method to address the new issue of how to disseminate the code to discrete, wide scope and even heterogeneous wireless sensor networks. The key issue of Code dissemination of sensors in a wide area is how to increase the coverage and keep the cost is acceptable.

The sensor devices widely deployed in the smart city can seem as a sparse sensor network. The sensor node cannot

transmit the data via multi-hop transmission, because they are too far to communicate with each other. But the data generated by sensor devices can be collected by data mules, which is first proposed in [52]. The authors presented an architecture to collect sensor data in sparse sensor networks by using data mules (mobile entities). The architecture can solve the problem of data collection of nodes with a short communication range but deployed sparsely. But using data mules special to collect data in sparse sensor network will cost a lot especially in a large region.

Fortunately, there is a large number of vehicles running on the street all around the city. And with the communication technology developing, most of these vehicles are equipped with communication device supporting multi-communication methods, like radio, LTE, RFID, Bluetooth, etc. These vehicles can be used as data mules to collect data, from the hardware perspective. Because of the motivation mentioned above, the paper [36] has done research on the oblivious data mules that are uncontrolled for data collection or delivery. The authors use taxis as oblivious data mules to perform an experiment using the real mobile trace dataset in Rome. And their experiments show the feasibility. In this paper, our proposed low-cost code dissemination model is based on [36] and furthermore take the cost into account which makes the model more in line with the real world. The novelty of this paper, in addition to the more realized model, we proposed the Cost-efficient Greedy Code Mules Selection Scheme to select vehicles for code dissemination. By using this scheme, the code dissemination tasks can be completed in a cost-efficient way, which can achieve a higher completion rate with lower cost. Moreover, we proposed High-performance MCS Deploy Scheme to deploy MCSs can furthermore improve the completion rate. With the joint performance optimization, our scheme achieved a better performance than other schemes.

III. SYSTEM MODELS AND PROBLEM STATEMENTS

A. SYSTEM MODEL

Let us consider a scenario, there are many smart devices deployed on the roadside in smart cities. For example, smart trash can on the roadside can send reminders to relevant departments when the garbage in the trash can reach a certain height; the smart billboards on the roadside can display various latest advertisements. These roadside smart devices can update their software by accepting specific update codes to achieve functional adjustments. In order to solve the problem of how to disseminate the update code to the roadside smart device, we propose a low-cost update code dissemination model for the roadside smart device. In this model, there are four main parts, Cloud Service Platform (CSP), Roadside Smart Devices (RSD), Code Mules (CM) that are the vehicles used to carry update code to roadside smart devices, and Mobile Code Station (MCS) that used to transmit update code to the CM safely.

In this model, the CSP is a key to coordinate other parts. The target users of the cloud service platform are the departments that manage various RSDs deployed in the smart city. When a user needs to update the software of RSD that it manages, the user can initiate an update request to the CSP, and the CSP will accept the user's update request and the corresponding update code. After that, the CSP will notify the CMs the update task via the cellular network or the other wireless network. Each CM will report its price to the cloud service platform for executing each update task according to its own situation. Then, the platform will select the CMs based on factors such as prices, trajectories to complete the task of updating the smart device. After that, the CSP will push the corresponding update code to the MCS. The MCS will stay in a certain position during the execution of the task, transmit the update code to the CM, and the CM will transmit the update code to the RSD.

Assume that there are m RSDs need to be updated in the task requested by the user. The update task set can be expressed as $S = \{s_1, s_2, \cdots, s_m\}$, where s_i denotes the task of updating the i-th RSD ($1 \leq i \leq m$). Assume that there are n vehicles can serve as CMs, we call the set of these vehicles as candidate CM set, they can be denoted as $V = \{v_1, v_2, \cdots, v_n\}$. In the low-cost update code dissemination model, the driving routes of these CMs are determined by drivers. The CM does not intentionally change the driving route due to the assignment of tasks, so the cost of completing the task is independent of the route. But the cost of completing a task is related to factors such as computation resources, storage resources, communication resources, and power consumption consumed to complete task. For the same CM, the cost of completing each task can be considered the same. That is cost (s_1) = cost (s_2) = $\cdots$ = cost (s_m). CM v_i ($v_i \in V$) will report the expected price to the CSP according to its cost and expected profit, after receiving the task notification from CSP. The price of CM reported can be denoted as b_i, the set of prices of all the CMs can be denoted as $B = \{b_1, b_2, \cdots, b_n\}$. In general, only a CM can't complete all tasks, so the CSP need to select CMs to execute tasks S after all CM reported the prices.

Below, we make a systematic and comprehensive textual explanation of the running process of the entire model system and succinctly demonstrates the main running process of the low-cost update code dissemination model through Figure 1.

(1) The user sends a code request S (including the information of the smart device to be updated, the update code and the maximum task execution time D) to the cloud service platform.

(2) The platform accepts the user's task request and notifies the profile of the task to all CMs in the candidate CM set V.

(3) Each CM v_i in the candidate CM set V returns its price b_i to the CSP according to the profile of the task.

(4) The CSP selects CMs from the candidate CM set V to execute the update task. The selected CM set can be denoted as $U = \{u_1, u_2, \cdots, u_l\}$ ($U \subseteq V$). After the selection is

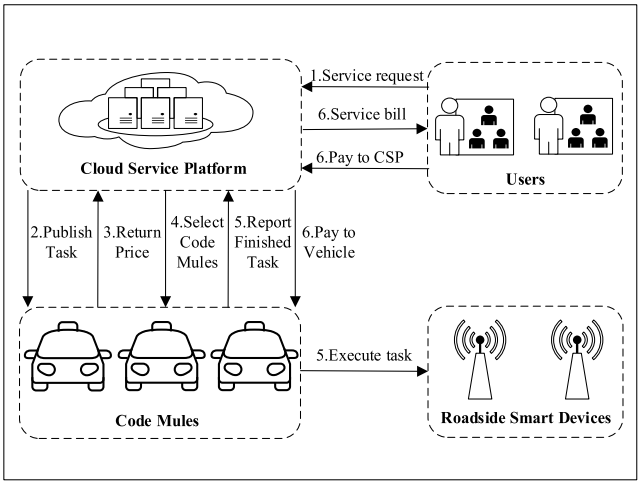


FIGURE 1. The running process of low-cost code dissemination model.

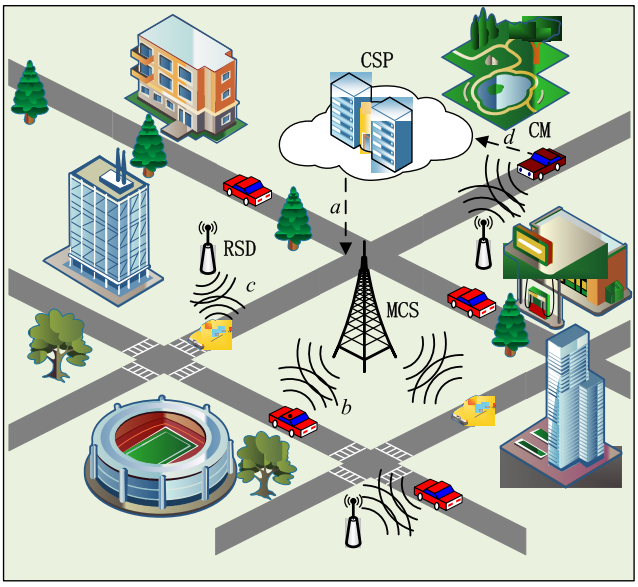


FIGURE 2. The execution process of code dissemination task.

complete, the CSP will notify all selected CMs in U to start executing task.

(5) All CMs in the selected CM set U start to execute tasks after receiving the start notification. The specific process of the CM to execute the task is shown in Figure 2. In order to ensure the security of the code dissemination, the update code is obtained by the MCS in a dedicated manner from the CSP, which is the process a in Figure 2. The MCS is deployed at a certain location during the execution of the code dissemination task and is responsible for transmitting the update code to CMs. When the CM passes by the MCS, CM can obtain the update code from MCS by wireless communication, that is, the process b in Figure 2. When the CM passes by the RSD, CM transmits the update code to the RSD to complete the code dissemination task, which is the process c in Figure 2. The CM need to report to the CSP once a code disseminate task is completed, that is, the process d in Figure 2.

(6) After reaching the deadline of the task, the platform will notify the selected CM to stop executing the task and enter the settlement state. The payoff p_i of CM u_i is the number of completed task q_i multiply by the price of completing a task, which can be expressed by equation (1):

$$p_i = q_i \times b_i \tag{1}$$

The total cost that the CSP needs to pay to CMs can be calculated by equation (2):

$$P_{CM} = \sum_{i=0}^{l} p_i = \sum_{i=0}^{l} (q_i \times b_i) \tag{2}$$

The fees that the CSP needs to charge the user can be calculated by equation (3):

$$P_{CSP} = P_{CM} + P_{profit} \tag{3}$$

where P_{profit} is the profit earned by the CSP.

B. PROBLEM STATEMENTS

In a low-cost code dissemination model, users always want to get a higher quality of service with less cost. And the CSP also hopes to reduce the service cost to get higher profits while ensuring the quality of service.

The quality of service can be measured by the completion rate of the task, we define the completion rate of the task as: The ratio of the number of tasks completed to the total number of tasks within the time required by the user. It can be expressed by formula (4):

$$C = N_{finish}/N_{total} \tag{4}$$

where N_{finish} denotes the number of completed tasks. N_{total} denotes the total number of tasks that the user requested to the CSP.

The service cost for CSP mainly comes from the payment of CM, so how to reduce the cost of CSP payment to CM is the key to reduce the service cost of CSP. The total cost paid by the CSP to the CM can be calculated using equation (2).

In a word, the problem that needs to be addressed in this low-cost code dissemination model is how to achieve the following goals as much as possible:

(1) Maximize the completion rate of the task to ensure the quality of service.

(2) Minimize the cost of paying to CM to reduce service costs, increase profits and reduce charges to users.

These can be summarized as formula (5):

$$\begin{cases} \max(C) = \max\left(N_{finish}/N_{total}\right) \\ \min(P_{vehicle}) = \min\left(\sum_{i=0}^{l}(q_i \times b_i)\right) \end{cases} \tag{5}$$

For the sake of easy understanding, we summarize the main notations used in this paper in Table 1.

IV. SCHEME DESIGN

A. MOTIVATION

(1) In smart cities, there are a large number of vehicles running in urban streets. These vehicles (especially taxis,

TABLE 1. Main notations.

Notation	Description
$\boldsymbol{V}$	Candidate Code Mules Set
$\boldsymbol{S}$	Code Dissemination Task Set requested by the user
$\boldsymbol{U}$	Code Mules Set selected by CSP to execute tasks
$\boldsymbol{k_j^i}$	The frequency of CM v_i passed sub-task s_j
$\boldsymbol{K_i^r}$	The set of the frequency of CM v_i more than threshold k_t
$\boldsymbol{S_i^r}$	The set of subtask that CM v_i can finish with high possibility
$\boldsymbol{s_i}$	i-th Code Dissemination task
$\boldsymbol{b_i}$	The price of CM v_i to finish a task

buses) are mostly equipped with wireless communication devices. With the ability of wireless communication, they are well-suited for Code Mules to participate in the dissemination of update code, with little or no hardware changes. By executing the Code Dissemination task assigned by CSP, Code Mules can earn revenue, and the cost is only a small amount of power consumption, communication and computing resources. Due to the different vehicle conditions and the difference in expected profit, this will cause the price reported by Code Mules to CSP to be different. Code Mules exist in a large number of smart cities, so the CSP have the space to choose. It can select CMs with a lower price through a strategy, and at the same time guarantee the completion rate.

(2) For vehicles running in the city, there was a certain similarity between the trajectories in adjacent time periods. In order to explore the similarity, we used the real taxi trajectory dataset in Rome to calculate the vehicle trajectory similarity. This data set contains a real trajectory of approximately 316 taxis for a period of 30 days. We have done some processing on each taxi trajectory, and the detail will be described in the experimental part. The trajectories of taxis on the period from 2014/2/1 to 2014/2/7 were regarded as Historical Trajectories 1 (HT1) and from 2014/2/8 to 2014/2/14 was regarded as Historical Trajectories 2 (HT2). We use equation (6) and equation (7) to calculate the similarity between HT1 and HT2:

$$\lambda_i^1 = N_i^{common}/N_i^1 \tag{6}$$

$$\lambda_i^2 = N_i^{common}/N_i^2 \tag{7}$$

where N_i^{common} is the number of the areas where the vehicle v_i passed both in HT1 and HT2. N_i^1 is the number of areas where the vehicle v_i passed in HT1. N_i^2 is the number of areas where the vehicle v_i passed in HT2.

Equation (6) has a difference from equation (7). Equation (6) can be used to calculate the proportion of the same part of the trajectory to the HT1. We call the value got by equation (6) similarity λ_1. And the proportion of the same part of the trajectory to the HT2 can be calculated by equation (7). We call it similarity λ_2. By comparing similarity λ_1 with similarity λ_2, we can understand the similarity between the trajectories in adjacent time periods more comprehensively and accurately.

The similarity between HT1 and HT2 had been calculated and the results had shown in Table 2, Figure 3, 4, 5. Table 2 lists some statistics for similarity λ_1 and similarity λ_2. As can be seen from Table 2, the similarities of 296 taxis are calculated in total actually. Because the historical trajectories or future trajectories of some taxis do not exist at the same

TABLE 2. Statistics of similarity.

Statistics	$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$
Number of taxis	296	296
Minimum	2%	13%
Maximum	68%	71%
Average	43%	42%
Median	45%	42%

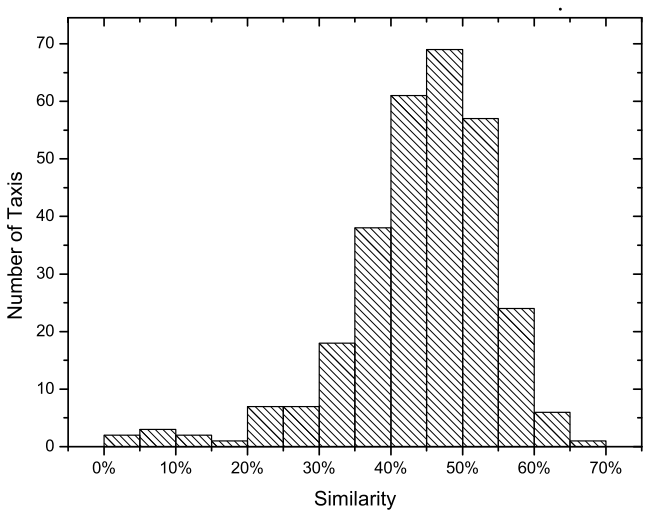


FIGURE 3. Histogram of similarity λ_1.

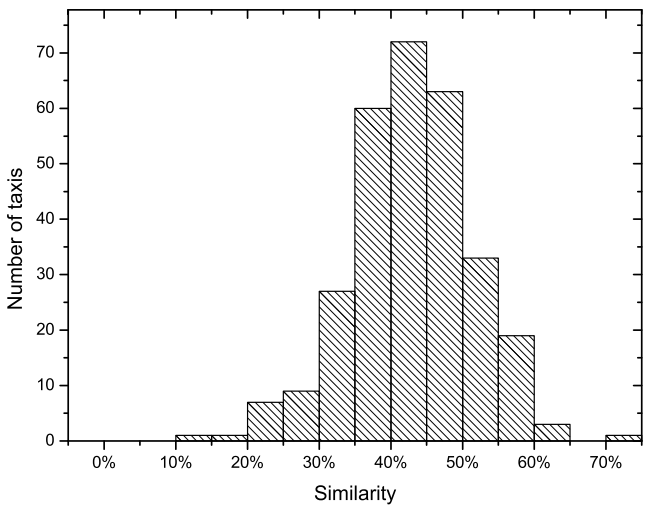


FIGURE 4. Histogram of similarity λ_2.

time, so the similarity of these taxis cannot be calculated. Comparing the statistics of similarity λ_1 and Similarity λ_2, we can see that except the minimum value, the difference between λ_1 and λ_2 in the maximum value and the mean value is small. Figure 3 and Figure 4 showed the histograms of similarity λ_1 and similarity λ_2, respectively. Comparing Figure 3 with Figure 4, it can be seen that the distribution of λ_1 and λ_2 are similar, and taxis with both similarity λ_1 and λ_2 between 20% and 60% are the majority. In Figure 3, the proportion of taxis with similarity greater than 20% is as high as 97.3%, and the proportion of taxis with similarity greater than 30% is 92.6%. The proportion of taxis with similarity greater than 40% is 73.6%. In Figure 4, the proportion of taxis with similarity greater than 20% is as high as 99.3%, and the proportion of taxis with similarity greater than 30% is 93.9%. The proportion of taxis with similarity greater than 40% is 64.5%.

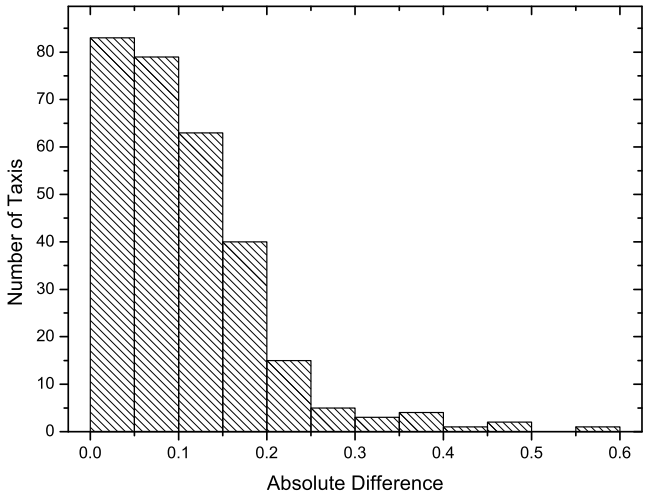


FIGURE 5. Histogram of absolute difference between λ_1 and λ_2.

And Figure 5 showed the absolute difference between λ_1 and λ_2 (the absolute value of the difference between λ_1 and λ_2) of all taxis in HT1 and HT2. It can be seen that most absolute differences of taxis were less than 0.3, accounting for 96% of the total. And the number of taxis with absolute differences less than 0.1 accounted for 56% of the total.

It can be seen from the above statistical results that the historical trajectories of most taxis have similarities with future trajectories, and the trajectory similarity of more than half of the taxis is greater than 42%. Therefore, it is reasonable to use the analysis of the historical trajectories as a reference for selecting CMs to execute the task.

B. COST-EFFICIENT GREEDY CODE MULES SELECTION SCHEME

In proposed low-cost update code dissemination model, after accepting the user's task request, how the CSP select CM to execute tasks is the key to accomplish the goals mentioned above. First of all, in order to ensure the completion of the task, it is necessary to select the vehicles that can pass the greatest number of positions of the tasks in S. And it is more feasible to make a selection based on the historical trajectory of the candidate CM. For the research problem of this paper, we proposed two reference values that can be derived from the CM historical trajectory: frequency and comprehensive trajectory similarity.

(1) **Frequency:** In this paper, the frequency of CM v_i passed the task s_j refers to the number of times CM v_i passed the task s_j position in the historical trajectory, it can be denoted as k_j^i. The k_j^i can be calculated by the historical trajectory of CM v_i. First, we divide the records of historical trajectory into T time periods of equal length. And we use z_t^j to indicate whether the CM passes the position of the task s_j in the t-th time period. If the CM passed through the position of s_j in the t-th time period, then $z_t^j = 1$, otherwise $z_t^j = 0$. Then k_j^i can be calculated by formula (8):

$$k_j^i = \sum_{t=1}^{T} z_t^j \tag{8}$$

where

$$z_t^j = \begin{cases} 1, & CM\ passed\ s_j \\ 0, & CM\ didn't\ passs_j \end{cases}$$

The K_i are used to indicate the frequency set for all task locations in S that CM v_i passed through, $K_i = \{k_1^i, k_2^i, \cdots, k_m^i\}$. Obviously, the larger k_j^i, the higher the probability that CM v_i will pass the position of task s_j in the future. If $k_j^i = 0$, it means that CM v_i has never passed the position of task s_j in the historical trajectory. Therefore, it can be inferred that CM v_i is unlikely to pass the position of task s_j in the future. We approximately regard that CM can accomplish this task as long as the CM can pass the location of the task. We set a threshold k_t, and regard the task s_j can be completed with high possibility if k_j^i is great than k_t. And we filter K_i by removing the k_j^i less than k_t. We use K_i^r to indicate K_i after filtered and use S_i^r to represent the set of tasks s_j corresponding to the element k_j^i in K_i. Therefore S_i^r represents the set of tasks that CM v_i is likely to complete in the future. The set S_i^r has a reference significance for selecting CM based on the assumption that there are a similarity between historical trajectory and future trajectory of CM. Obviously, the CSP can select a CM set $U = \{u_1, \cdots, u_i, \cdots, u_l\}$ according to S_i^r by using a method, so that the CM in U can complete all tasks with high possibility. That is $S_1^r \cup \cdots \cup S_i^r \cup \cdots \cup S_l^r = S$.

(2) **Comprehensive trajectory similarity:** In this paper, the comprehensive trajectory similarity of CM v_i is also the similarity between the trajectories in adjacent time periods. But it considered comprehensively the similarity λ_1, λ_2 and the difference of them, so it is called the comprehensive trajectory similarity. The comprehensive trajectory similarity can be calculated by equation (9). In the equation (9), the larger the λ_1 and λ_2, the larger the λ. And the larger the difference between λ_1 and λ_2, the smaller the λ. The larger λ_1 and λ_2 and less difference between λ_1 and λ_2 mean the more similarity and stability of the HT1 and HT2. This will get higher value of comprehensive trajectory similarity. This is in

line with the real-world situation. Comparing to the similarity λ_1 and similarity λ_2, the comprehensive trajectory similarity λ can express the similarity more reasonable.

$$\lambda_i = (1 - \left|\lambda_i^1 - \lambda_i^2\right|) \cdot \frac{\lambda_i^1 + \lambda_i^2}{2} \tag{9}$$

According to the analysis above, the selection of CM can be converted approximately into a problem of a set coverage. To solve this problem, a simple and feasible method is to use a greedy algorithm to find a CM set that can complete all the tasks in S. However, the future trajectory of CM is not fully similar to the historical trajectory, so by using the basic greedy algorithm the completion rate may not reach to maximum, and the total cost cannot reduce to the minimum.

In order to overcome the shortcomings of the basic greedy algorithm, we propose a measure function $\varpi(K_i^r, b_i, \lambda_i)$:

$$\varpi_i = \lambda_i \cdot (\theta \cdot (1 - b_i) + (1 - \theta) \cdot \frac{\sum_{k_j^i \in K_i^r} k_j^i}{\left|K_i^r\right|}) \tag{10}$$

where θ is the adjust factor, $\left|K_i^r\right|$ is the number of elements of the set K_i^r.

This measure function can comprehensively take into account the frequency, the comprehensive trajectory similarity and the price of CM v_i. It can be known by formula (10) that the value of ϖ_i increases with the increase of λ_i, increases with the decrease of b_i, and increases with the increase of k_j^i. That means, the CM v_i with a higher value of ϖ_i has a higher the comprehensive similarity, a lower price and a higher frequency. Therefore, there will be better performance in theory by using the measure function base on the greedy thought.

Note that, the values of b_i and k_j^i might not in the same range, it is necessary to do normalization on b_i and k_j^i before calculating ϖ_i using the equation (10). The equation (11) was used to convert the values of b_i and k_j^i to the range between 0 and 1.

$$x_{normalization} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{11}$$

In the following, we formally propose the Cost-efficient Greedy Code Mules Selection Scheme (CGCSS). Its execution can be divided into the following steps:

(1) Calculate the frequency k_j^i by traversing the historical trajectory of the candidate CM set V. Filter the K_i according to the threshold k_j^i. Get the filtered frequency set $\mathbb{K}_r = \{K_1^r, \cdots, K_i^r, \cdots, K_n^r\}$ and corresponding task set $\mathbb{S}_r = S_1^r, \cdots, S_i^r, \cdots, S_n^r$.

(2) Calculate the comprehensive similarity λ_i of CM v_i in V using equations (6), (7) and (9) according to the historical trajectory of CM.

(3) Traverse the candidate CM set V, calculate the value of ϖ_i according to formula (10), and select the CM with the largest ϖ_i join the selected CM set U.

(4) Update V, S, $\mathbb{K}_r, \mathbb{S}_r$. Remove selected CM v_{max} from V. Remove task set S_{max}^r corresponding to the v_{max} from S. Traverse $\mathbb{K}_r$, remove the frequency of the task in S_{max}^r. Traverse $\mathbb{S}_r$, remove the tasks contained in S_{max}^r.

(5) Go back to step (3) until the task set S is empty.

The details of the CGCSS is shown in Algorithm 1.

Execute tasks by CMs selected by using CGCSS, in theory, can achieve higher completion rate at a lower cost. In section V, we will use a real trajectory dataset to perform a complete experiment to test the validity of CGCSS.

Algorithm 1 Cost-Efficient Greedy Code Mules Selection Scheme (CGCSS)

Input: Historical trajectory, S, V, B
Output: U

$\mathbb{K}_r = \emptyset, \mathbb{S}_r = \emptyset, U = \emptyset$
For each CM v_i in $V(1 \le i \le n)$:
  $K_i = \emptyset, K_i^r = \emptyset, S_i^r = \emptyset$
  For each task s_j in S:
    For $t = 1$ **to** T:
      $k_j^i = k_j^i + z_t^i$
    End for
    If $k_j^i > k_t$:
      $K_i^r = K_i^r \cup \{k_j^i\}$
      $S_i^r = S_i^r \cup \{s_i\}$
    End If
  End for
  $\mathbb{K}_r = \mathbb{K}_r \cup \{K_i^r\}$
  $\mathbb{S}_r = \mathbb{S}_r \cup S_i^r$
End for
For each CM v_i in $V(1 \le i \le n)$:
  calculate λ_i according to the equation (6)(7)(9)
End For
While $S \neq \emptyset$:
  For each CM v_i in V:
    calculate ϖ_i according to the equation (10)
  End For
  $\varpi_{max} = \max\{\varpi_1, \cdots, \varpi_i, \cdots, \varpi_{|V|}\}$
  $U = U \cup \{v_{max}\}$
  $V = V - v_{max}$
  $S = S - S_{max}^r$
  Update $\mathbb{K}_r, \mathbb{S}_r$.
End while
Return U

C. HIGH-PERFORMANCE MCS DEPLOY SCHEME

In the low-cost code dissemination model, after CM selection, MCSs need to be deployed in some positions to transmit the update code to CMs during the execution of tasks. This means that to complete a task, the CM not only must be able to reach the location of the task, but also need to obtain an Update Code from MCS before this. Therefore, in order to complete a task, CM need to obtain the update code from MCS at first and transmit the update code when passing by RSD. However, if CM does not get the update code from the MCS when it passes by the RSD, CM cannot transmit update code to the

RSD, that means, CM cannot complete the task. Therefore, in order to ensure the completion rate of Code Dissemination, it is necessary to choose the appropriate location to deploy MCS. Obviously, it is better to select the area where the CM frequently passed through to deploy MCS, so that more CMs can obtain the update code from the MCS and increase the possibility of task completion.

We have proposed the High-performance MCS Deploy Scheme (HMDS). In this scheme, we analyze the historical trajectory of the selected CM at first. Then, get the areas that the selected CM passed through as candidate areas for deploying the MCS. Next, count the frequency of CM passing through each candidate area. Finally, sort the candidate areas according to the frequency that CM passed through. If h MCS need to be deployed, the top h areas can be selected from the sort candidate areas to deploy MCS. According to the discussion in the motivation section, there is a similarity between HT1 and HT2. Therefore, CM will pass through the areas selected by HMDS frequently in the future.

The details of the deploy scheme are shown in Algorithm 2. Where the input value HT is the Historical Trajectories of CM selected by CSP. And h is the number of areas that needed to deploy MCS.

Algorithm 2 High-Performance MCS Deploy Scheme

Input: HT, h
Output: R

$A = \emptyset, PT = \emptyset, R = \emptyset$
//count all the areas that CMs in U passed through
For each HT_i **in** HT:
    count the areas $a_{i1}, \cdots, a_{in}$ that u_i passed in HT_i
    $A = A \cup a_{i1}, \cdots, a_{in}$
End For
//count the times of CMs passed through in each area of A
For each a_i **in** A:
    $t_i =$ count the times of CMs passed through a_i
    $PT = PT \cup t_i$
End For
$A_s =$ Sort A according to PT in descending order
// select top h area to return
For $i = 1$ **to** h:
    $R = R \cup a_i$ $(a_i \in A_s)$
End for
Return R

V. EXPERIMENT AND PERFORMANCE ANALYSIS

In this section, we used the dataset of mobility traces of taxi cabs in Rome, Italy [53] to perform our experiments. This dataset contains GPS coordinates of approximately 316 taxis collected over 29 days from 2014/2/1 to 2014/3/1. The size of the dataset is about 1.49GB, include more than 21 million (21,817,851) records. In the following part, the experiment will be explained in detail.

A. THE PREPARATION OF EXPERIMENT

The dataset needed to be preprocessed at first in order to meet the experimental requirements. The preprocess followed the steps below:

(1) Split the dataset. In this dataset, each record was stored in the format (id, date, time, GPS coordinates), and all CM records were stored in the same file in chronological order. In order to perform the experiment easily at a specific time in the dataset, it was necessary to divide the experimental data into multiple files by date. In addition, in order to conveniently select the specified CM, the data set needed to be further segmented according to the id of the CM.

(2) Filter the dataset. Through the analysis of the dataset, we found that there were some illegal records in the dataset, which may be due to the weak signal of GPS in some locations. The latitude and longitude coordinates of these records were not in the range of Rome's latitude and longitude, these records could not reflect the real situation of driving of CM so that they need to be filtered out. In the data set, we selected records with a longitude range between 41.7933E and 41.9924E and a latitude range between 12.3716N and 12.6818N. In addition, we obtained the CM speed through two consecutive records of the same CM. After analyzing, it was found that there were some speeds in the data set that were far beyond the normal value. In order to ensure the accuracy of the data, we filtered out the second record in two consecutive records of abnormal speed. Assume that the CM can complete the task with a top speed limit of 90km/h. So, we only kept records with speeds less than 90km/h. Figure 6 showed the filtered dataset of Rome track points for 2014/2/2.

(3) Grid the dataset. The communications between CM and RSD or MCS were wireless. Therefore, the data transmission between them can carry out within the large communication distance. Therefore, we could divide the experimental area according to the communication range. To be convenient, we divided the experimental area into a lot of square grids of the same size. As shown in Figure 7, under the condition of communication distance $\sqrt{2}\,d$, as long as CM and MCS or RSD were located in the same square grid cell with side length d, it can be regarded as a successful data transmission. We set $d = 70m$, the experimental area can be divided into 93267 grids, where 241 in longitude and 387 in latitude. To be convenience, we numbered the longitude from 0 to 240, and the latitude was numbered from 0 to 386. The longitude lon_i and the latitude lat_i of the i-th record in the dataset could be converted into the corresponding abscissa X_i and ordinate Y_i in the grid system by formula (12) and formula (13), respectively.

$$X_i = \lceil (lon_i - lon_{min})/e_{lon} \rceil \tag{12}$$

$$Y_i = \lceil (lat_i - lat_{min})/e_{lat} \rceil \tag{13}$$

where lon_{min} and lat_{min} respectively represent the minimum longitude and minimum latitude of our selected range.

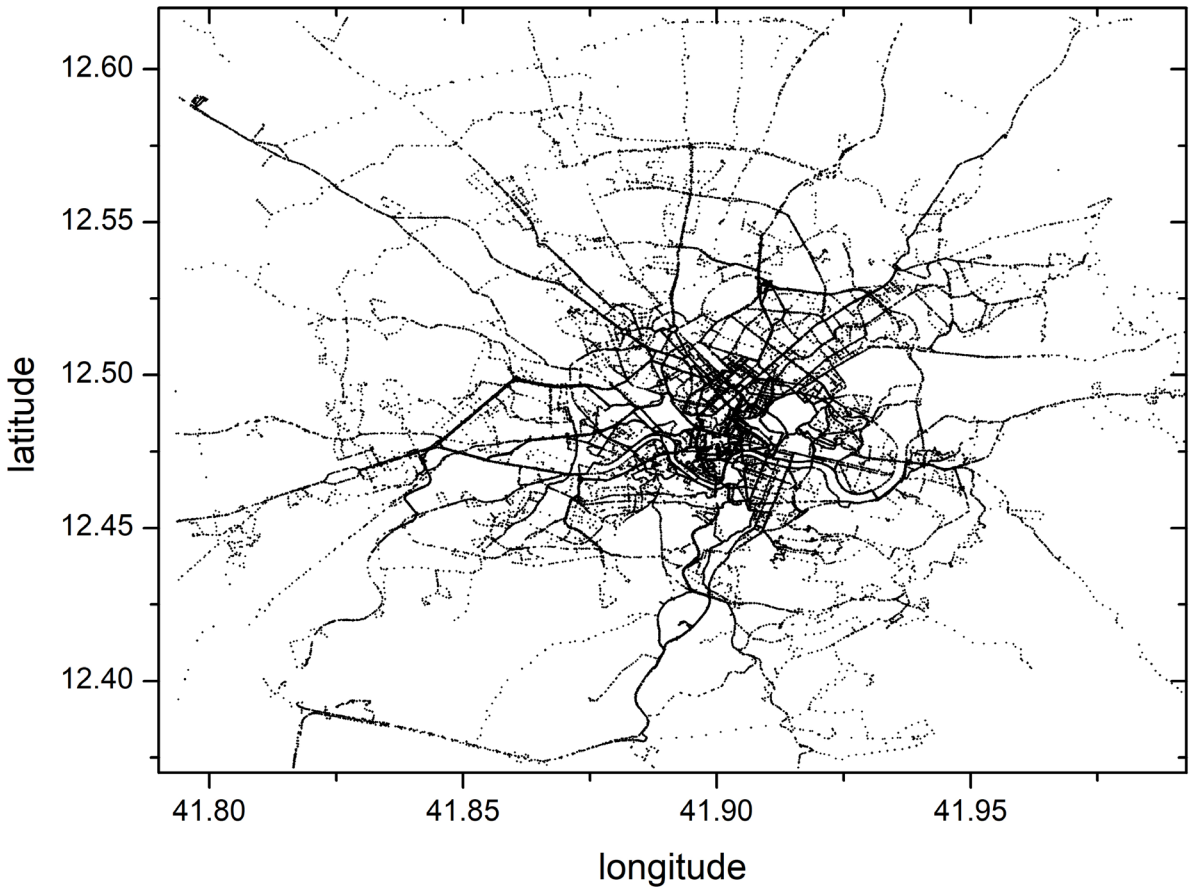


FIGURE 6. The trace points of taxis in the dataset of Rome on 2014/2/2 after filtered.

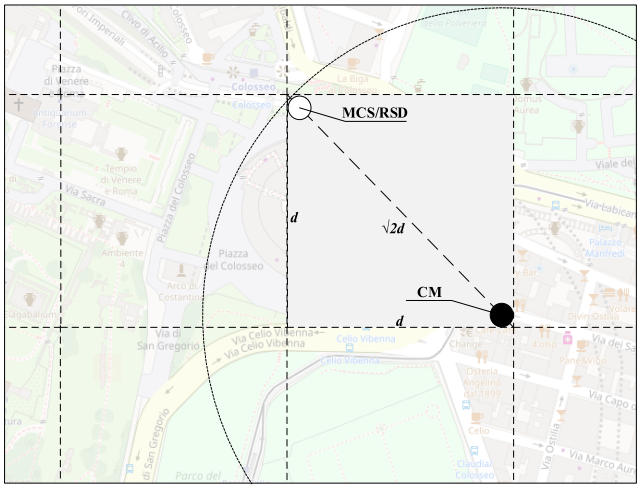


FIGURE 7. Gridding the dataset.

e_{lon} and e_{lat} represent the longitude difference and latitude difference of 70m distance respectively.

(4) Compress the dataset. After gridding the dataset, the trajectory of the CM was converted from the geographic coordinate system to the new grid systems. The granularity of the new grid system was larger than before. Therefore, there were many consecutive records with the same coordinates in the dataset after conversion. In our experiments, these records were redundant and can be compressed. Leaving only the first one of the redundant records was enough. The speed of the experiment can be greatly accelerated after compression.

The dataset that meet the experimental requirements could be obtained after preprocessing mentioned above. Figure 8 showed the trace points of the dataset in 2014/2/2 after all the preprocessing steps

After preprocessing, we selected data from 2014/2/1 to 2014/2/14 in the dataset of Rome as historical reference data, using data from 2014/2/15 to 2014/2/21 as experimental test data. In the experiment, the locations of the RSDs in the update task requested by the user was selected randomly from the 6461 areas where taxis passed through every day from 2014/2/1 to 2014/2/14. And we assigned a price to each CM in the candidate CM set in a random manner, satisfying a normal distribution with an expectation of 10 and a variance of 1. Figure 9 showed the distribution statistics of CM prices in an experiment simulation. For the task execution time requested by the user, we set it to 7 days.

B. THE COMPARISON EXPERIMENT OF CGCSS

To test the effectiveness of our proposed CGCSS, we used the Basic Greedy Code Mule Select Scheme (BGCSS) and

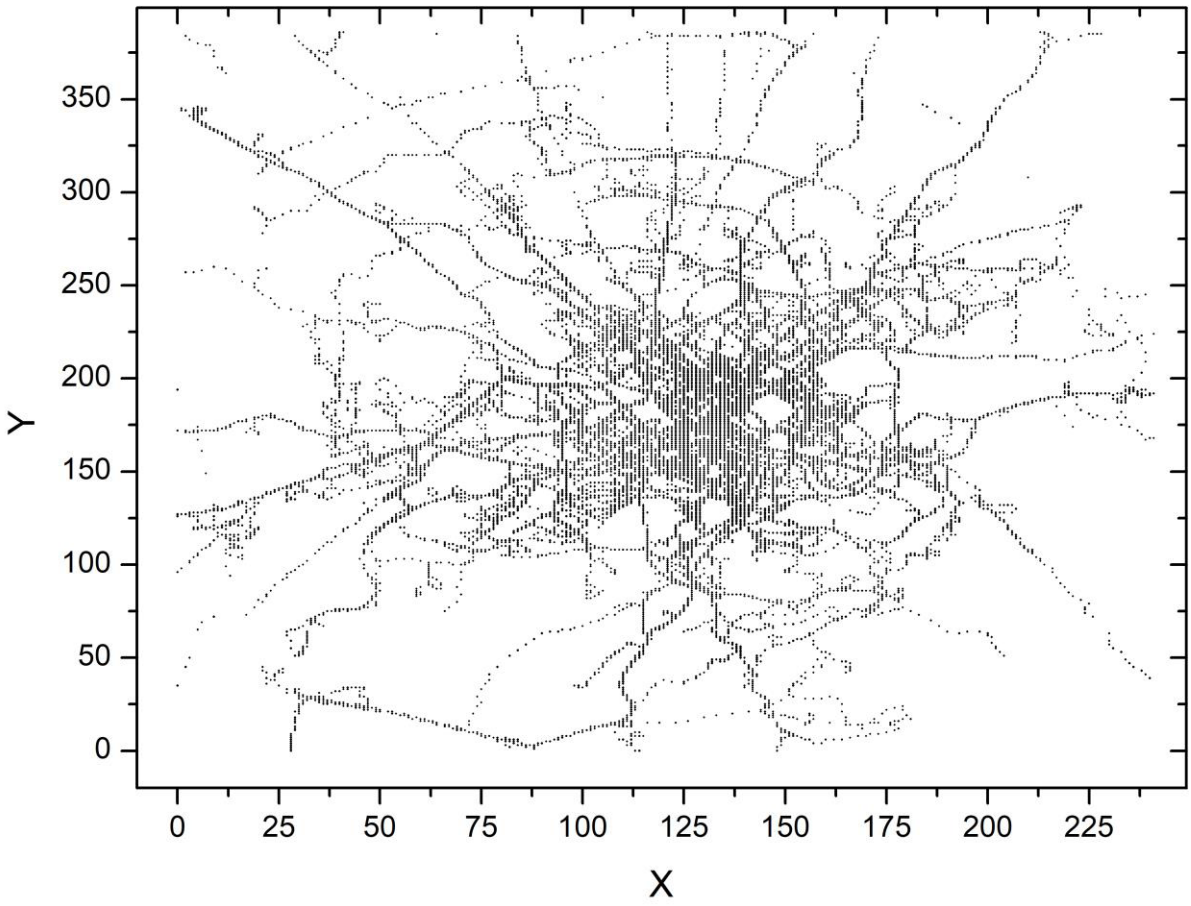


FIGURE 8. The trace points of taxis in the dataset of Rome on 2014/2/2 after preprocessing.

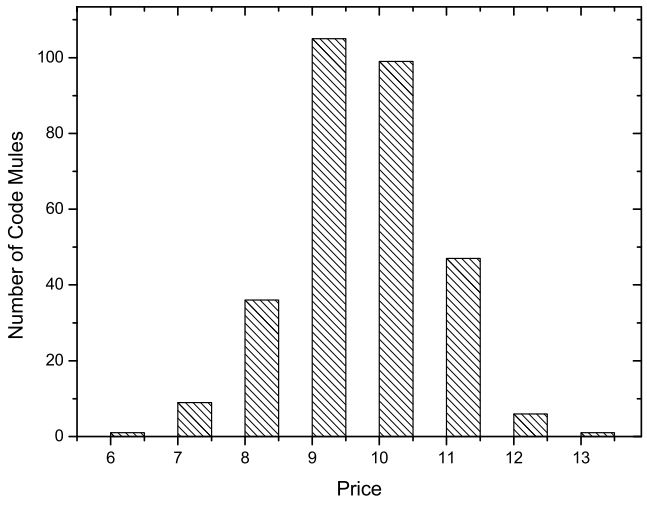


FIGURE 9. The price distribution of 304 CMs.

Random Code Mules Select Scheme (RCSS) for comparison. The BGCSS selects CM according to $\mathbb{S}_r$, and always selects the CM with the largest number of tasks that may complete. And the selection method of the RCSS was similar to the CGCSS, but the value of ϖ_i was randomly generated. For our CGCSS, we set $\theta = 0.5$ and $k_t = 0$. For the value of θ, setting $\theta = 0.5$ can balance the weight of price and frequency, and the influence of θ would be discussed in part C. For the value of k_t, in this dataset, if $k_t \geq 1$, the number of candidate vehicles was not enough for finishing selection. And k_t need to meet the requirement that $k_t \geq 0$. So, we set $k_t = 0$. After CM selection, we randomly select areas from the 6461 candidate areas to deploy the MCS.

We conducted 6 groups of comparative experiments with the total number of tasks requested by the user m was 100, 200, 300, 400, 500, 600 respectively. Each group of comparison experiments was carried out 5 times to reduce the contingency of the experiment, a total of 30 comparison experiments. Table 3 listed the organization order of 30 comparative experiments. In the 5 comparison experiments in a group, the total number of tasks was the same, but the locations of RSDs in task and the prices of each CM changed in each experiment.

(1) At first, we compared the completion rate of CGCSS RCSS and BGCSS. Figure 10 and Figure 11 showed the comparison of completion rate under the number of tasks m was 100 and 200, respectively. As can be seen from Figure 10,

TABLE 3. Organization of scheme comparison experiments.

Group	Experiment Number	Condition
1	1-5	$m = 100$
2	6-10	$m = 200$
3	11-15	$m = 300$
4	16-20	$m = 400$
5	21-25	$m = 500$
6	26-30	$m = 600$

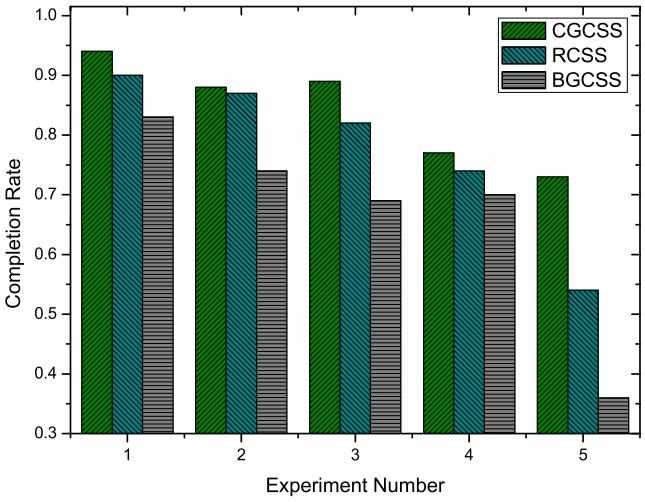


FIGURE 10. Comparison of Completion rate ($m = 100$).

CGCSS achieved better completion rates in all five experiments than BGCSS and RCSS. The maximum completion rate of CGCSS reached 94%. Compared RCSS, CGCSS has a slight improvement, but has an improvement of 35.2% in experiment number 5.

As can be seen from Figure 11, in the case of 200 tasks, the CGCSS was better than the BGCSS in the completion rate of the five experiments as well. CGCSS was lower than RCSS in two experiments but higher in three experiments. In this group of experiments, CGCSS and RCSS have similar performance in most case, but RCSS was higher than CGCSS to 9% in experiment number 10.

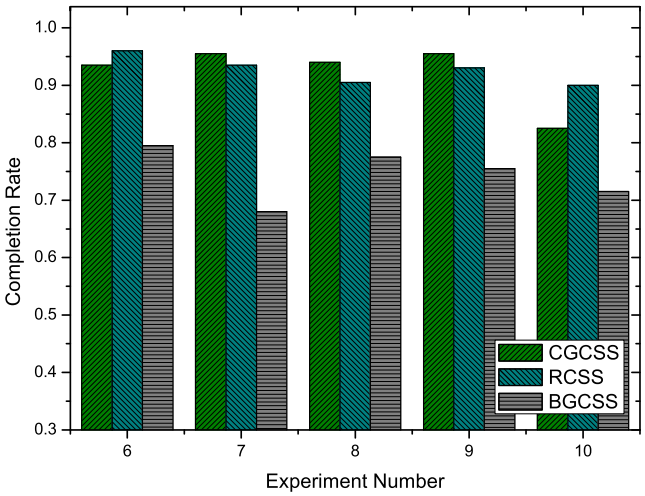


FIGURE 11. Comparison of Completion rate ($m = 200$).

All 30 experimental results on completion rate were shown in Figure 12. As can be seen that, the completion rate of CGCSS was slightly higher than RCSS in most experiments. And CGCSS was more stable than RCSS on the change of experiment results. Both CGCSS and RCSS were better than BGCSS in completion rate in all experiments. Compared with BGCSS, CGCSS had a maximum increase of more than 102%. And CGCSS had a maximum increase of 35.2% compared to RCSS.

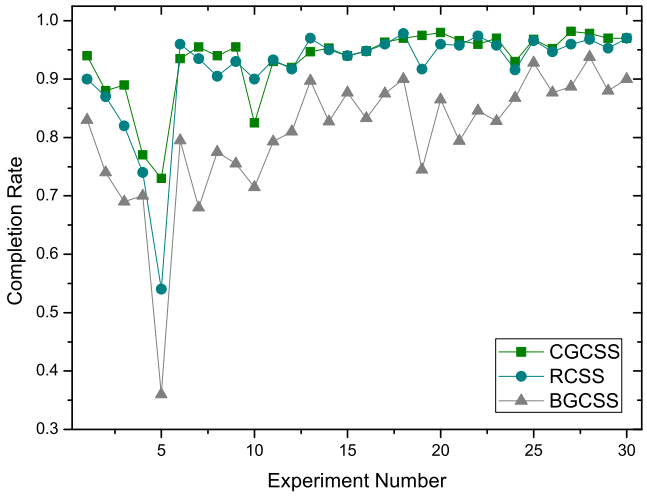


FIGURE 12. Comparison of Completion rate.

(2) Next, we compared the service cost of CGCSS, RCSS and BGCSS. It can be seen that the number of completed tasks of these schemes was different. The more tasks completed, the more service cost, in the condition that the difference of prices between CMs was small. In this case, the service cost of CGCSS was more than that of BGCSS even if CGCSS selected the CMs with a lower price. Therefore, it was fairer to compare the average price of completing a single task. The average price of completing a single task b_{avg} can be calculated by equation (14).

$$b_{avg} = P_{total}/N_{finish} \tag{14}$$

where P_{total} is the total service cost, N_{finish} is the total number of completed tasks.

Figure 13 and Figure 14 showed the average price under $m = 100$ and $m = 200$ respectively. Note that, the average prices here were the average value of prices after normalization, so the range of value was different with that in Figure 9. As can be seen that, the average prices of CGCSS were much lower than RCSS and BGCSS under $m = 100$. In this group of experiments, compared to RCSS, the average price of CGCSS was max reduced by up to 64% and reduced by to 41.6% at least. Compared to BGCSS, the average price of

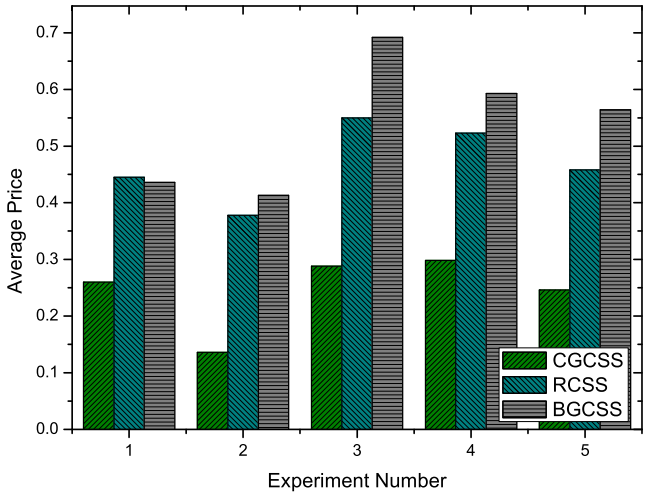


FIGURE 13. Comparison of average price ($m = 100$).

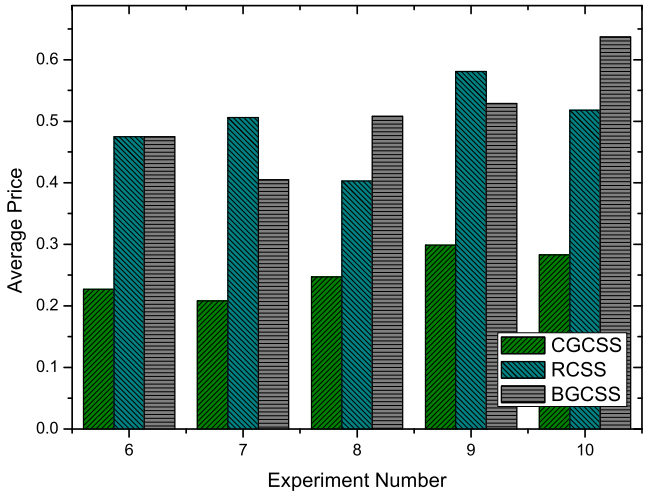


FIGURE 14. Comparison of average price ($m = 200$).

CGCSS had a max decrease by up to 67%, a min decrease by to 40%.

As can be seen from Figure 14, the average prices of CGCSS were also much lower than RCSS and BGCSS under m = 200. In this group of experiments, the lowest average price of CGCSS was 0.208, and the highest average price was 0.299. The lowest average price of RCSS was 0.403, and the highest average price was 0.581. And the lowest average price of BGCSS was 0.405, and the highest average price was 0.637. Compared with RCSS, the maximum decrease of CGCSS was 58.9%, the minimum decrease was 38.7%. And Compared with BGCSS, the maximum decrease of CGCSS was 55.6%, the minimum decrease was 43.5%.

Figure 15 showed a comparison of the average price for all 30 experiments. It can be seen that, in all the comparison experiments, the average prices of CGCSS were always lower than RCSS and BGCSS. And the change of average prices was less than RCSS and BGCSS.

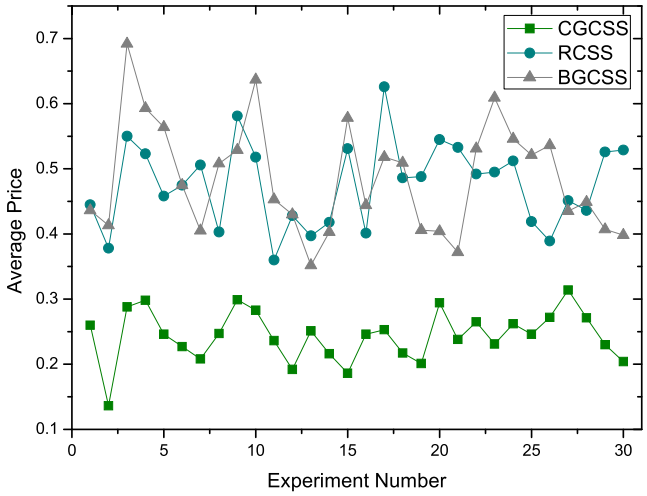


FIGURE 15. Comparison of average price.

(3) And we compared the cost-performance ratio. In this paper, the cost-performance ratio was the ratio of coverage to average price, it can be calculated by equation (15).

$$cp = {}^{C}/_{b_{avg}} \tag{15}$$

where C is the completion rate and b_{avg} is the average price.

A high value of cost-performance ratio meant a high task completion rate and a low cost. Therefore, the higher the cp value, the better. The cost-performance ratio could reflect the performance of the scheme.

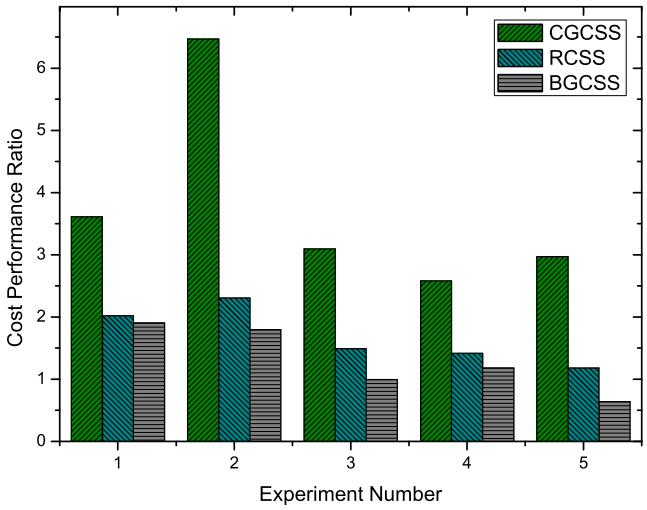


FIGURE 16. Comparison of Cost Performance ratio ($m = 100$).

Figure 16 and Figure 17 showed the comparison of cost-performance ratio under $m = 100$ and $m = 200$ respectively. As can be seen from Figure 16, in the five experiments, the cost-performance ratios of CGCSS were higher than RCSS and BGCSS. Compared with RCSS, the maximum improvement of CGCSS was up to 180.7%, and the minimum improvement was 78.7%. And Compared with

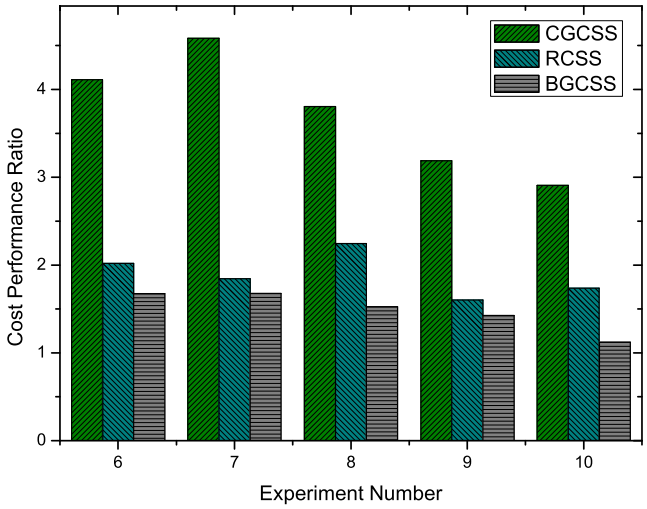


FIGURE 17. Comparison of Cost Performance ratio ($m = 200$).

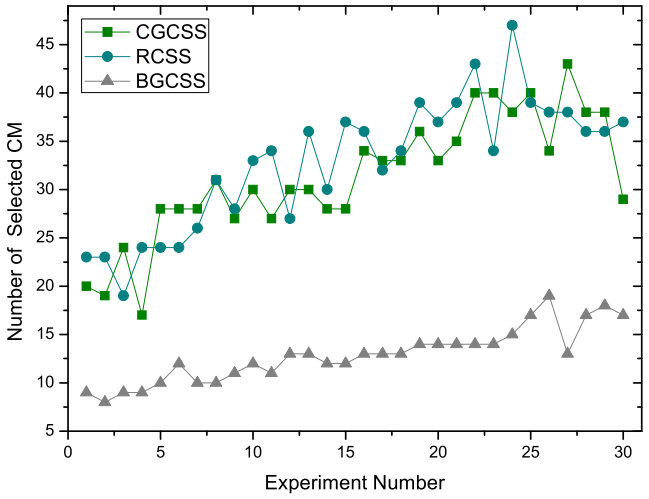


FIGURE 19. The number of selected CM.

BGCSS, the maximum improvement of CGCSS was up to 365.5%, and the minimum improvement was 89.5%. And Figure 17 showed the comparison of cost-performance ratio under $m = 200$. It can be seen that, the cost-performance ratios of CGCSS were higher than RCSS and BGCSS as well. Compared with RCSS, CGCSS max increased by 148.3% and the minimum increased by 67.5%. Compared with BGCSS, CGCSS max increased by 173.3% and the minimum increased by 123.6%.

Figure 18 showed the comparison of the cost-performance ratio in all 30 experiments. We could see from Figure 18, the cost-performance ratios of CGCSS were all higher than RCSS and BGCSS in 30 experiments. And the cost-performance ratio difference between CGCSS and RCSS and BGCSS was much larger than that between RCSS and BGCSS.

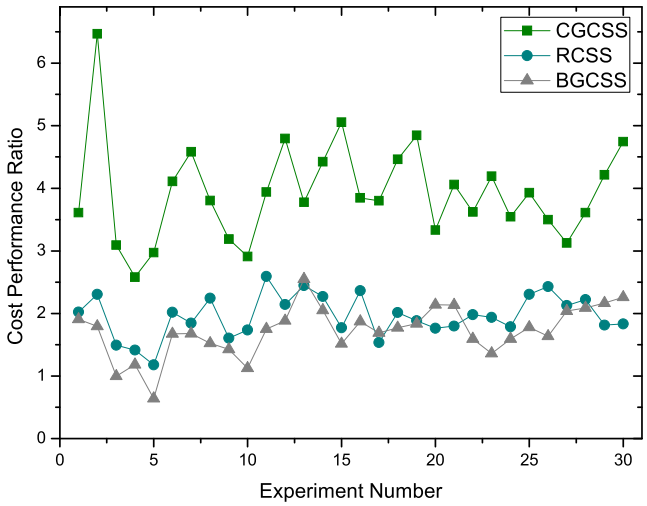


FIGURE 18. Comparison of Cost Performance ratio.

(4) At last, we analyzed the reason why CGCSS has a better performance than RCSS and BGCSS. As can be seen from the above comparison, CGCSS was better than RCSS and BGCSS in terms of completion rate, the average price of completed tasks, and cost-performance ratio in the case of different total number of tasks. In theory, this was because BGCSS always tend to select a CM that may complete the most tasks. This selection strategy cannot be affected by price and similarity, etc. In RCSS, selecting the CMs randomly did not consider these factors as well. But CGCSS used the measure function to select CM, referred by frequency, trajectory similarity, and price factor comprehensively. With the measure function, CGCSS tend to select the CM with high trajectory similarity, low price and high frequency. This allows CGCSS to increase the completion rate and reduce the task price with higher probability.

And we also analyzed the reason from the experiment result. Figure 19 shows a comparison of the number of CMs selected by CGCSS, RCSS and BGCSS in 30 experiments. As can be seen from Figure 19, the number of CMs that CGCSS and RCSS selected to execute tasks was much larger than that of BGCSS. Obviously, it was more likely to complete more tasks with more CMs to execute tasks. As can be seen in Figure 19 and Figure 12, CGCSS selected less CMs than RCSS, but achieve higher completion rate in the most experiments. Because the similarity and frequency were considered in CGCSS.

C. THE COMPARISON EXPERIMENT OF θ VALUE

It was clear that CGCSS has better performance than RCSS and BGCSS through the above analysis. However, when using CGCSS to select CM, different θ values of measure function, might lead to different results for completion rate and service cost. In the following, we used the CGCSS of the measure function with θ was 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 respectively, and carried out 6 groups experiments under the conditions of total task number m was 100, 200,

TABLE 4. Organization of θ comparison experiments.

Group	Number	Condition
1	1-5	m=100
2	6-10	m=200
3	11-15	m=300
4	16-20	m=400
5	21-25	m=500
6	26-30	m=600

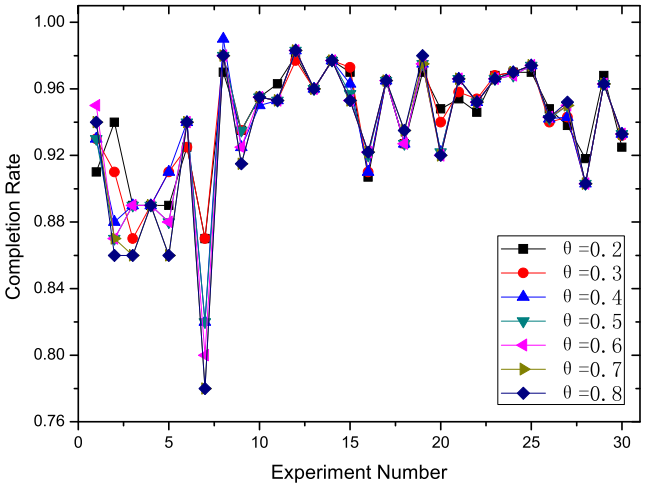


FIGURE 20. Completion rate with different values of θ.

300, 400, 500, 600. To avoid contingency, each group performed 5 experiments, a total of 30 comparison experiments. Table 4 listed the organization of all 30 experiments.

Figure 20 showed the completion rate with different θ values in 30 comparison experiments. It can be seen that the difference was small that between the completion rates with different θ values. And the trends of completion rate with different θ values were quite similar. But there was still a difference between them by comparing the average completion rate of 30 experiments. As shown in Figure 21, the higher the θ value, the smaller the completion rate under CGCSS. Obviously, this was consistent with the character of the measure function. It can be seen from the equation (9) that the higher the θ value, the higher the value of $\theta \cdot (1 - b_i)$ and the lower the value of $(1-\theta) \cdot \frac{\sum_{k_j^i \in K_i^r} k_j^i}{|K_i^r|}$, which made the measure function more affected by the prices but less affected by frequencies. And the frequency was related to the completion rate. So that could explain the regulation between θ and completion rate.

Figure 22 showed the comparison of average price with different θ. The difference between average prices with different θ values still was not very large. But the average price with a lower value of θ was higher than that with a higher value of θ in most experiments.

Figure 23 showed the averages of average price of different θ values in 30 experiments. It showed that the higher the θ

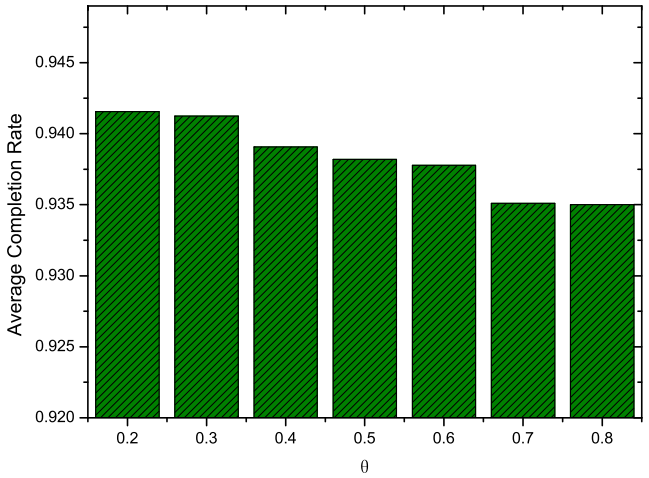


FIGURE 21. Average completion rate with different θ.

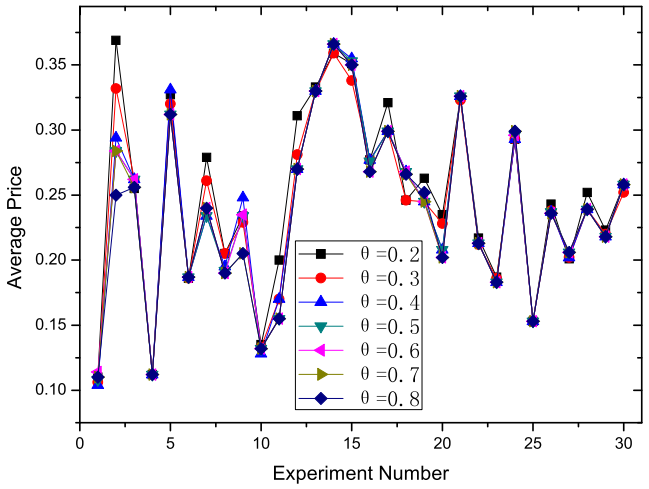


FIGURE 22. Average price with different θ.

values, the lower the average of average price. According to the equation (9), the higher the θ value, the higher the value of $\theta \cdot (1 - b_i)$ and measure function was more affected by the price of CM. That meant the CM with a lower price was more likely to be selected. Therefore, the average price of completing a single task was lower.

Figure 24 showed the comparison of cost-performance ratio with different θ values. The cost-performance ratios with different θ values were very similar but there were still differences in some experiments like experiment number 1,2, 7,8, etc. In these experiments, the rough rule was that the higher the θ value, the higher the cost-performance ratio. Figure 25 showed the average of cost-performance ratios with different θ values. As can be seen from Figure 25, the higher the θ value, the higher the average of cost-performance, except $\theta = 0.6$. This was because, as θ increases, the decrease in the average completion rate was smaller than the decrease in the average of average price in most case.

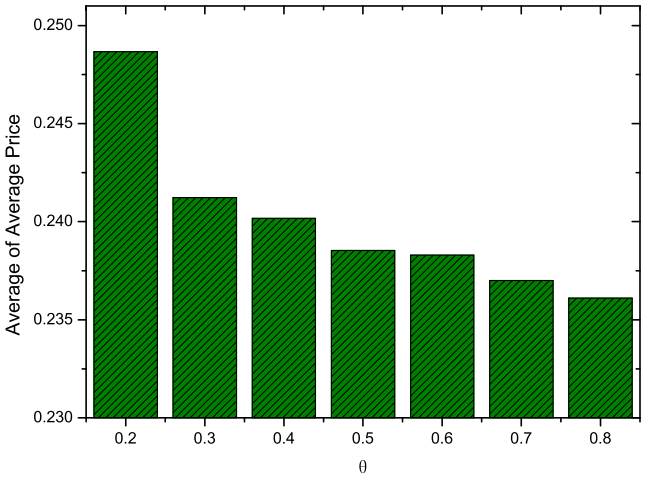


FIGURE 23. Average of average price with different θ.

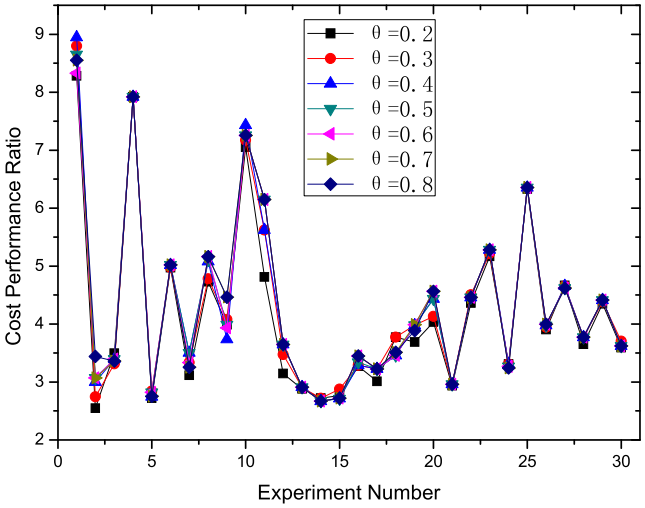


FIGURE 24. Cost-performance ratio with different θ.

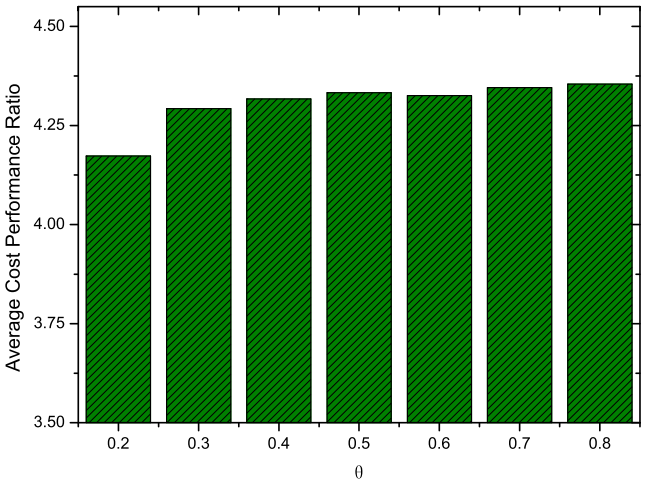


FIGURE 25. Average cost-performance ratio with different θ.

Figure 26 showed the number of CM selected by CSP under CGCSS with different θ values. As can be seen from Figure 26, the difference of the number of selected CM

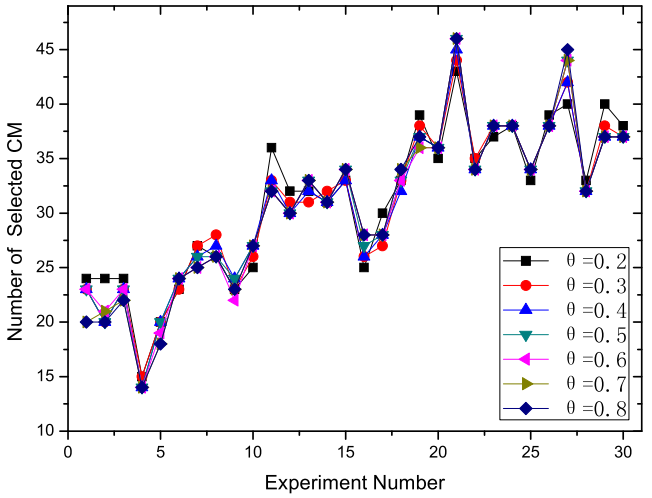


FIGURE 26. Number of selected CM with different θ.

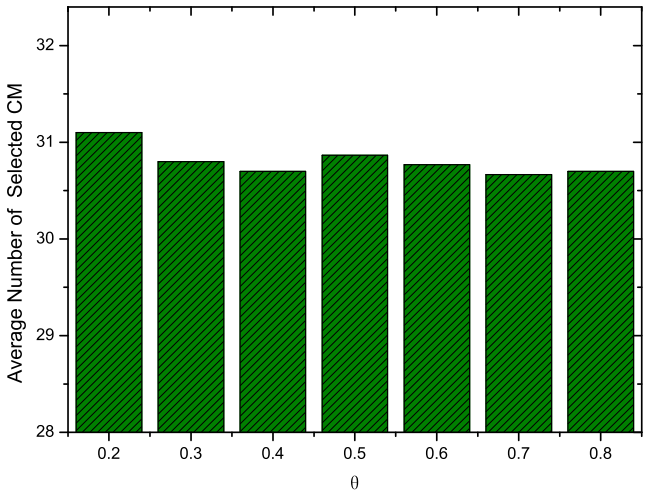


FIGURE 27. Average number of selected CM with different θ.

with different θ was very small in the most experiments. A rough rule was that the more total tasks the user had requested, the more CMs that CSP selected under CGCSS. Figure 27 showed the average number of selected CM with different θ. The difference between the average number of selected CM was really small, the difference between maximum and minimum was less than 0.3. But the difference of average completion rate was relatively large, this phenomenon showed the effectiveness of frequency.

Through the above comparison experiment, it can be seen that under the premise of this data set and the price setting, the relatively high θ value could reduce the average price of the completed task and improve the cost performance, but the completion rate was reduced. Therefore, in the case of stricter requirements for completion and relatively loose price requirements, a lower value of θ was preferred. And when the price requirement was stricter and the requirement of completion rate was more relaxing, it was more suitable to adopt a higher θ value.

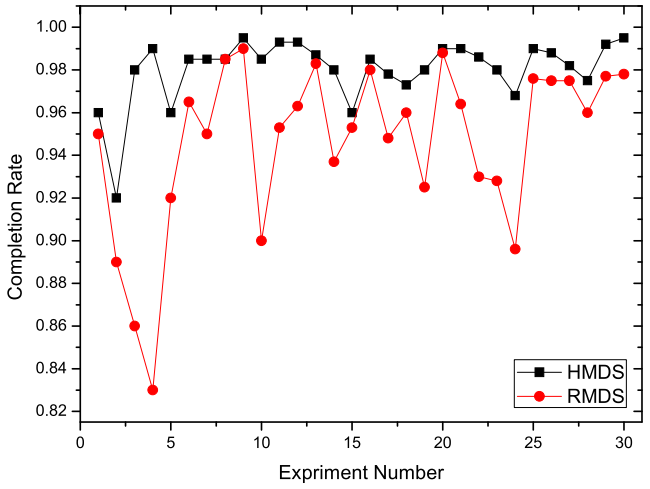


FIGURE 28. Comparison of HMDS and RMDS under CGCSS.

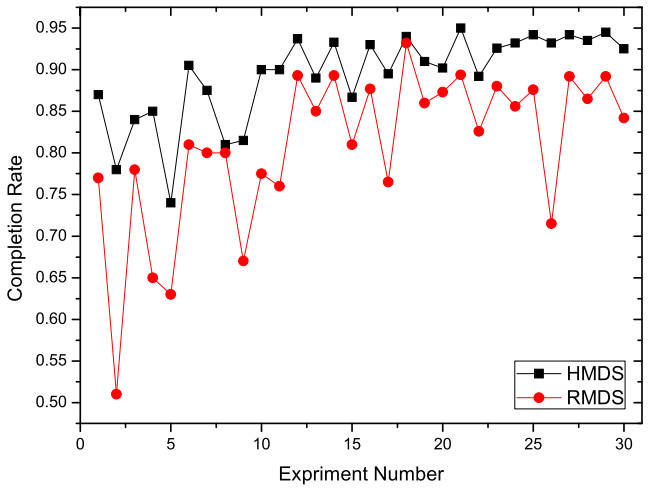


FIGURE 29. Comparison of HMDS and RMDS under BGCSS.

D. THE COMPARISON EXPERIMENT OF HMDS

In this section, we tested the effect of High-performance MCS Deploy Scheme (HMDS) and used the Random MCS Deploy Scheme (RMDS) for comparison. First, we used CGCSS to select CM and use HMDS and RMDS to select 5 MCS deployment areas respectively. We conducted 6 groups of comparison experiments under the condition that the total task number was 100, 200, 300, 400, 500, 600. We conducted 5 experiments in every group, and a total of 30 comparison experiments.

Figure 28 showed a comparison of task completion rates for deploying MCS using HMDS and RMDS, respectively. We could see from Figure 28, the completion rate of HMDS was always higher than RMDS. In all 30 experiments, the minimum task completion rate of HMDS was 92%, the minimum task completion rate of RMDS was 83%. And the change of completion rate of HMDS was smaller than that of RMDS. Obviously, HMDS showed better performance than RMDS when used CGCSS to select CMs.

Next, we compared the completion rate of HMDS and RMDS when used BGCSS to select CMs. Figure 29 showed that the task completion rate of HMDS was still higher than RMDS. The minimum completion rate for HMDS was 74%, and the minimum completion rate for RMDS was 51%. HMDS significantly improved completion rate in experiment number 2, 4, 9, etc. Obviously, HMDS showed better performance than RMDS when used BGCSS to select CMs as well.

Through the comparison experiments above, we verified that the MCS deployment area had an impact on the completion rate, and we also verified that the proposed HMDS was effective to improve the completion rate. Therefore, the better performance could be achieved by using CGCSS to select CMs and using HMDS to deploy MCSs.

VI. CONCLUSION

In this paper, we proposed the model that use vehicles to help the IoT applications in the smart city to disseminate the update code to the Roadside Smart Devices by V2X communication. And we proposed a Cost-efficient Greedy Code Mules Selection Scheme (CGCSS) to select the vehicles as the code mules to execute the code dissemination task. In this scheme, we using a measure function to compute a weight as the reference of the selection. And the cost and historical trajectory of a vehicle can be comprehensively considered into the measure function. The extensive comparison experiments using the real trajectories dataset in Rome city show the effectiveness of our proposed CGCSS. By selecting the vehicles as code mules using our CGCSS, the code dissemination tasks can be executed with a low cost and high completion rate. And we further analyzed the impact of the adjust factor θ, and given the advices to choose a proper value of θ under different conditions. Moreover, a High-Performance MCS Deploy Scheme is proposed to find better locations to deploy the mobile code stations for improving the performance of the system.

Due to time constraints, there are still some aspects of this article that are not considered. The future work may include: extending the way to disseminate different type of update code; analyzing the top size limit of update code; improving the security of the code dissemination; introducing more kinds of V2X connection method, etc.

REFERENCES

[1] D. Zhang, Y. Liu, Z. Ding, Z. Zhou, A. Nallanathan, and T. Sato, "Performance analysis of non-regenerative massive-MIMO-NOMA relay systems for 5G," *IEEE Trans. Commun.*, vol. 65, no. 11, pp. 4777–4790, Nov. 2017.

[2] X. Liu *et al.*, "Construction of large-scale low-cost delivery infrastructure using vehicular networks," *IEEE Access*, vol. 6, no. 1, pp. 21482–21497, 2018.

[3] D. Zhang, Z. Zhou, C. Xu, Y. Zhang, J. Rodriguez, and T. Sato, "Capacity analysis of NOMA with mmWave massive MIMO systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 7, pp. 1606–1618, Jul. 2017.

[4] J. Tan *et al.*, "A low redundancy data collection scheme to maximize lifetime using matrix completion technique," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, p. 5, Jan. 2019, doi: 10.1186/s13638-018-1313-0.

[5] H. Teng *et al.*, "Adaptive transmission range based topology control scheme for fast and reliable data collection," *Wireless Commun. Mobile Comput.*, vol. 2018, Jul. 2018, Art. no. 4172049, doi: 10.1155/2018/4172049.

[6] T. Li, S. Tian, A. Liu, H. Liu, and T. Pei, "DDSV: Optimizing delay and delivery ratio for multimedia big data collection in mobile sensing vehicles," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3474–3486, Oct. 2018.

[7] X. Liu, W. Liu, Y. Liu, H. Song, A. Liu, and X. Liu, "A trust and priority based code updated approach to guarantee security for vehicles network," *IEEE Access.*, vol. 6, pp. 55780–55796, 2018.

[8] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018, doi: 10.1109/TII.2018.2816590.

[9] V. Vukadinovic, K. Bakowski, P. Marsch, I. D. Garcia, and H. Xu, "3GPP C-V2X and IEEE 802.11p for vehicle-to-vehicle communications in high-way platooning scenarios," *Ad Hoc Netw.*, vol. 74, pp. 17–29, May 2018.

[10] M. Sepulcre and J. Gozalvez, "Context-aware heterogeneous V2X communications for connected vehicles," *Comput. Netw.*, vol. 136, pp. 13–21, May 2018.

[11] A. H. Alavi, P. Jiao, W. G. Buttlar, and N. Lajnef, "Internet of Things-enabled smart cities: State-of-the-art and future trends," *Measurent*, vol. 129, pp. 589–606, Dec. 2018.

[12] H. Teng *et al.*, "A novel code data dissemination scheme for Internet of Things through mobile vehicle of smart cities," *Future Gener. Comput. Syst.*, vol. 94, pp. 351–367, May 2019.

[13] W. Qi *et al.*, "Minimizing delay and transmission times with long lifetime in code dissemination scheme for high loss ratio and low duty cycle wireless sensor networks," *Sensors*, vol. 18, no. 10, p. 3516, 2018, doi: 10.3390/s18103516.

[14] Y. Liu, A. Liu, X. Liu, and X. Huang, "A statistical approach to participant selection in location-based social networks for offline event marketing," *Inf. Sci.*, vol. 480, pp. 90–108, Apr. 2019.

[15] F. Xiao, X. Xie, Z. Li, Q. Deng, A. Liu, and L. Sun, "Wireless network optimization via physical layer information for smart cities," *IEEE Netw.*, vol. 32, no. 4, pp. 88–93, Jul./Aug. 2018.

[16] Y. Ren, W. Liu, Y. Liu, N. Xiong, A. Liu, and X. Liu, "An effective crowdsourcing data reporting scheme to compose cloud-based services in mobile robotic systems," *IEEE Access*, vol. 6, no. 1, pp. 54683–54700, 2018.

[17] J. Tan *et al.*, "An adaptive collection scheme-based matrix completion for data gathering in energy-harvesting wireless sensor networks," *IEEE Access*, vol. 7, no. 1, pp. 6703–6723, 2019. doi: 10.1109/ACCESS.2019.2890862.

[18] T. Li, Y. Liu, N. Xiong, A. Liu, Z. Cai, and H. Song, "Privacy-preserving protocol of sink node location in telemedicine networks," *IEEE Access*, vol. 6, no. 1, pp. 42886–42903, 2018.

[19] X. Li *et al.*, "Differentiated data aggregation routing scheme for energy conserving and delay sensitive wireless sensor networks," *Sensors*, vol. 18, no. 7, p. 2349, 2018, doi: 10.3390/s18072349.

[20] A. Liu and S. Zhao, "High performance target tracking scheme with low prediction precision requirement in WSNs," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 29, no. 4, pp. 270–289, 2018.

[21] X. Xiang, W. Liu, N. Xiong, H. Song, A. Liu, and T. Wang, "Duty cycle adaptive adjustment based device to device (D2D) communication scheme for WSNs," *IEEE Access*, vol. 6, pp. 76339–76373, 2018, doi: 10.1109/ACCESS.2018.2882918.

[22] J. Zhang *et al.*, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[23] Z. Ding *et al.*, "Orchestrating data as a services-based computing and communication model for information-centric Internet of Things," *IEEE Access*, vol. 6, pp. 38900–38920, 2018.

[24] T. Li, K. Ota, T. Wang, X. Li, Z. Cai, and A. Liu, "Optimizing the coverage via the UAVs with lower costs for information-centric Internet of Things," *IEEE Access*, to be published, doi: 10.1109/ACCESS.2019.2894172.

[25] C. Yang, Z. Shi, K. Han, J. Zhang, Y. Gu, and Z. Qin, "Optimization of particle CBMeMBer filters for hardware implementation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 9027–9031, Sep. 2018, doi: 10.1109/TVT.2018.2853120.

[26] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2870288.

[27] W. Hou *et al.*, "On-chip hardware accelerator for automated diagnosis through human–machine interactions in healthcare delivery," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 1, pp. 206–217, Jan. 2019, doi: 10.1109/TASE.2018.2832454.

[28] Y. Guo, X. Hu, B. Hu, J. Cheng, M. Zhou, and R. Y. K. Kwok, "Mobile cyber physical systems: Current challenges and future networking applications," *IEEE Access*, vol. 6, pp. 12360–12368, 2018.

[29] X. Ju *et al.*, "An energy conserving and transmission radius adaptive scheme to optimize performance of energy harvesting sensor networks," *Sensors*, vol. 18, no. 9, p. 2885, 2018, doi: 10.3390/s18092885.

[30] L. Guo, Z. Ning, W. Hou, B. Hu, and P. Guo, "Quick answer for big data in sharing economy: Innovative computer architecture design facilitating optimal service-demand matching," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1494–1506, Oct. 2018, doi: 10.1109/TASE.2018.2838340.

[31] W. Yang *et al.*, "Adding active slot joint larger broadcast radius for fast code dissemination in WSNs," *Sensors*, vol. 18, no. 11, p. 4055, 2018, doi: 10.3390/s18114055.

[32] Y. Ren, Y. Liu, N. Zhang, A. Liu, N. N. Xiong, and Z. Cai, "Minimum-cost mobile crowdsourcing with QoS guarantee using matrix completion technique," *Pervasive Mobile Comput.*, vol. 49, pp. 23–44, Sep. 2018.

[33] Y. Liu, M. Ma, X. Liu, N. Xiong, A. Liu, and Y. Zhu, "Design and analysis of probing route to defense sink-hole attacks for Internet of Things security," *IEEE Trans. Netw. Sci. Eng.*, doi: 10.1109/TNSE.2018.2881152.

[34] X. Liu, J. Yin, S. Zhang, B. Ding, S. Guo, and K. Wang, "Range-based localization for sparse 3D sensor networks," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2856267.

[35] S. Fang *et al.*, "Feature selection method based on class discriminative degree for intelligent medical diagnosis," *Comput., Mater., Continua*, vol. 55, no. 3, pp. 419–433, 2018.

[36] M. Bonola, L. Bracciale, P. Loreti, P. Amici, A. Rabuffi, and G. Bianchi, "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers," *Ad Hoc Netw.*, vol. 43, pp. 43–55, Jun. 2016.

[37] M. Huang, A. Liu, N. Xiong, T. Wang, and A. V. Vasilakos, "A low-latency communication scheme for mobile wireless sensor control systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 2, pp. 317–332, Feb. 2019.

[38] P. Le Nguyen, Y. Ji, Z. Liu, H. Vu, and K. V. Nguyen, "Distributed hole-bypassing protocol in WSNs with constant stretch and load balancing," *Comput. Netw.*, vol. 129, pp. 232–250, Dec. 2017.

[39] J. Xu, K. Ota, M. Dong, A. Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1546–1557, 2018.

[40] Z. Liu, T. Tsuda, H. Watanabe, S. Ryuo, and N. Iwasawa, "Data driven cyber-physical system for landslide detection," *Mobile Netw. Appl.*, pp. 1–12, Mar. 2018, doi: 10.1007/s11036-018-1031-1

[41] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2074–2081, Sep. 2018.

[42] X. Liu, Q. Yang, J. Luo, B. Ding, and S. Zhang, "An energy-aware offloading framework for edge-augmented mobile RFID systems," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2881295.

[43] S. Zhang, K. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Gener. Comput. Syst.*, vol. 86, pp. 881–892, Sep. 2018.

[44] R. Aissaoui, H. Menouar, A. Dhraief, F. Filali, A. Belghith, and A. Abu-Dayya, "Advanced real-time traffic monitoring system based on V2X communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 2713–2718.

[45] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of V2X communication in support of cooperative autonomous driving," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 64–70, Dec. 2015.

[46] S. Chen, J. Hu, Y. Shi, and L. Zhao, "LTE-V: A TD-LTE-based V2X solution for future vehicular network," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 997–1005, Dec. 2016.

[47] K. J. Ahmed and M. J. Lee. "Secure LTE-based V2X service," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3724–3732, Oct. 2017.

[48] R. Oliveira, C. Montez, A. Boukerche, and M. S. Wangham, "Reliable data dissemination protocol for VANET traffic safety applications," *Ad Hoc Netw.*, vol. 63, pp. 30–44, Aug. 2017.

[49] M. Chaqfeh and A. Lakas, "A novel approach for scalable multi-hop data dissemination in vehicular ad hoc networks," *Ad Hoc Netw.*, vol. 37, pp. 228–239, Feb. 2016.

[50] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, "SYNAPSE++: Code dissemination in wireless sensor networks using fountain codes," *IEEE Trans. Mobile Comput.*, vol. 9, no. 12, pp. 1749–1765, Dec. 2010.

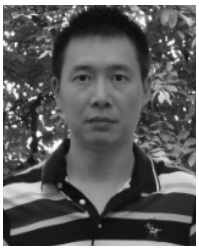
[51] W. Dong, Y. Liu, Z. Zhao, X. Liu, C. Chen, and J. Bu, "Link quality aware code dissemination in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1776–1786, Jul. 2014.
[52] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a three-tier architecture for sparse sensor networks," in *Proc. 1st IEEE Int. Workshop Sensor Netw. Protocols Appl. (SNPA)*, May 2003, pp. 30–41, doi: 10.1109/SNPA.2003.1203354.
[53] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. (Jul. 17, 2014). *CRAWDAD Dataset Roma/Taxi*. [Online]. Available: https://crawdad.org/roma/taxi/20140717, 10.15783/C7QC7M.

HAOJUN TENG is currently pursuing the master's degree with the School of Information Science and Engineering, Central South University, China. His research interests include big data, machine learning, and cloud computing.

WEI LIU received the Ph.D. degree in computer application technology from Central South University, China, in 2014. He is currently an Associate Professor and a Senior Engineer with the School of Informatics, Hunan University of Chinese Medicine, China. His research interests include software engineering, data mining, and medical informatics. He has published over 20 papers in the related fields.

TIAN WANG received the B.Sc. and M.Sc. degrees in computer science from Central South University, in 2004 and 2007, respectively, and the Ph.D. degree from the City University of Hong Kong, in 2011. He is currently an Associate Professor with the National Huaqiao University of China. His research interests include wireless sensor networks, social networks, and mobile computing.

ANFENG LIU received the M.Sc. and Ph.D. degrees in computer science from Central South University, China, in 2002 and 2005, respectively, where he is currently a Professor with the School of Information Science and Engineering. His major research interest includes wireless sensor networks. He is a member (E200012141M) of the China Computer Federation.



XUXUN LIU (M'14) received the Ph.D. degree in communication and information system from Wuhan University, Wuhan, China, in 2007. He is currently an Associate Professor with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. His research has been supported by the National Natural Science Foundation of China for three times. He has authored or co-authored over 30 scientific papers in international journals and conference proceedings. His current research interests include wireless sensor networks, wireless communications, computational intelligence, and mobile computing. He served as an Associate Editor for the IEEE Access, and as the Workshop Chair, as the Publication Chair, or as a TPC Member for a number of conferences. He has served as a Reviewer for over 30 journals, including ten IEEE journals and five Elsevier journals.

SHAOBO ZHANG received the B.Sc. and M.Sc. degrees in computer science from the Hunan University of Science and Technology, Xiangtan, China, in 2003 and 2009, respectively, and the Ph.D. degree in computer science from Central South University, Changsha, China, in 2017. He is currently a Lecturer with the School of Computer Science and Engineering, Hunan University of Science and Technology, China. His research interests include privacy and security issues in social networks and cloud computing.