Journal of Network and Computer Applications 135 (2019) 11–19

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Coupling resource management based on fog computing in smart city systems

Tian Wang [a,b], Yuzhu Liang [a], Weijia Jia [b], Muhammad Arif [c], Anfeng Liu [d], Mande Xie [e,*]

[a] College of Computer Science and Technology, Huaqiao University, Xiamen, China
[b] State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China
[c] Department of Computer Science and Technology, Guangzhou University, Guangzhou, China
[d] School of Information Science and Engineering, Central South University, Hunan, China
[e] School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, China

ARTICLE INFO

Keywords:
Smart city
Sensor-cloud
Cloud computing
Fog computing

ABSTRACT

The smart city is set to be the next important stage in human history. It consists of numerous Cyber-physical systems (CPS) that can be endowed with different levels of intelligence. However, the increasing use of the CPS and their application to key infrastructure components means that failures can result in disruption, damage and even loss of life. Avoiding these consequences is not a trivial problem and some researchers propose that cloud computing can provide the effective and sustainable services for smart city. With the integration of CPS and cloud computing, sensor-cloud system becomes popular in many fields where the physical nodes can be shared with multiple users, which can achieve high performance computing. However, when a physical sensor node receives multiple services commands simultaneously, there will be some service collisions, namely, coupling resource management problem. This coupling resource management problem leads to the failures of services. In order to solve the coupling resource management problem, we propose a fog computing model and extend the Hungarian algorithm to manage the coupling resource which can get smaller delay to realize effective and sustainable services. The fog computing layer acts as a buffer and controller between CPS layer and cloud layer which can handle malicious attacks to build highly sustainable systems. Experimental results and theoretical analysis show that our method can reduce the coupling computing and increase the resource utilization to make systems more effectively.

1. Introduction

The smart city is one of the demanding aspects that uses digital technologies to enhance performance and wellbeing, to reduce costs and resource consumption, and to engage more effectively and actively with its citizens. Smart city focuses on intelligent computing infrastructure and the smart city and CPS be closely bound up.

CPS are reliable and evolvable networked time-sensitive computational systems integrated with physical processes, and are being widely used in many critical areas, such as manufacturing, traffic control, military surveillance (Liu et al., 2018), to target the locations (Qi et al., 2018a), environmental (Wang et al., 2018a) and medical monitoring (Qi et al., 2017a). CPS can be further developed to achieve the goals of intelligent, flexible and adaptive machines. Additionally, by integrating CPS with production, logistics and services in the current industrial practices, it will turn factory today into an industrial 4.0 factory. However, the shortcomings are limited the fields of CPS, such as small memory, lack of energy, weak communication capacity, and computing power (Wang et al., 2018b) (Liu et al., 2019) (Xu et al., 2018).

In order to overcome them, CPS are combined with cloud computing, so-called sensor-cloud system (Luo et al., 2018) (Zhang et al., 2018; Chen et al., 2018a). Cloud computing provides superior computing power and near-unlimited storage capacity (Qi et al., 2018b). What's more, moving the data to the cloud not only reduces the costs but also provides the ability to the user's customize their systems. One of the characteristics of this system is the virtual sensor layer has appeared, which can perform well in providing service for multiple users. For

* Corresponding author.
E-mail address: xiemd@zjgsu.edu.cn (M. Xie).

https://doi.org/10.1016/j.jnca.2019.02.021
Received 30 September 2018; Received in revised form 21 January 2019; Accepted 25 February 2019
Available online 1 March 2019

improving the resource utilization, one physical sensor can be shared by several users (Wang et al., 2017).

However, compared with the fast developing data processing speed, the bandwidth of the network has already come to a halt (Qi et al., 2017b). Therefore, a new paradigm, fog computing is emerging technology. Fog computing (Edge computing) refers to enabling the technologies that allow computation to be performed at the edge of the network, downstream data on behalf of the cloud services and upstream data on behalf of the IoTs (Internet of Things) services (Al Omar et al., 2019; Qi et al., 2017c). The fog computing can provide highly reliable services which are more efficient than cloud computing for the CPS.

Even though, if a sensor receives commands from multiple users at the same time in the system, the conflict happens, which is the coupling resource management problem. For one thing, the sensor may have only one command channel and cannot receive multiple commands at the same time. For another, some commands may contradict with others. For example, for a mobile sensor, one user may schedule it to move to one direction for data collection while another user may schedule it to the opposite direction.

The coupling resource management problem has a negative impact on the user's usage of sensor-cloud systems because of system conflicts. In order to solve the coupling resource management problem, we proposed a method based on Hungarian algorithm in paper (Liang et al., 2017). This paper is an extension of the paper (Liang et al., 2017) and the main extensions of this paper are shown below. First of all, we revised the definition of the problem and gave the formal description of the problem. Secondly, we rewrote the algorithm and designed the buffer queue in the fog computing layer which cache most scheduled resources. Finally, we rewrote the experimental environment settings and added more realistic parameter settings. As for the experimental results, we added the experimental results of the algorithm costs.

In summary, our contributions are three-fold:

(1) We find a coupling resource management problem in the smart city system which can cause low-sustainability and system conflicts. As far as we know, this problem is a new problem by the integration of cloud computing and CPS and a formal description of the new problem is given in this paper.
(2) In order to solve the coupling resource management problem, we extend the Hungarian algorithm with buffer queue (EHGB) to obtain the minimum cost schedule and maximum resource utilization. In addition, we design a buffer queue in the fog computing layer which will return the result to the cloud layer directly. When there are some malicious nodes and user's requests in the system, fog layer can cache the conflicting node's data which will avoid system conflicts to realize high-sustainability.
(3) We conduct the extensive simulations to compare the proposed scheduling algorithm with existing solutions. The performance of the proposed method is validated by extensive experimental results.

The rest of this paper is organized as follows. Section 2, the related work is reviewed. The new problem of coupling resource management problem is defined in Section 3. The design of algorithm is described in Section 4. Section 5 is about the analysis and comparison of experimental results. We provide the conclusion in Section 6.

2. Related work

In this section, we briefly review the recent advances related to CPS and fog computing, and analysis some solving coupling resource management problem methods.

2.1. CPS and fog computing

As mentioned above, CPS has made great achievements recently. In managing big data and leveraging the interconnectivity of machine, CPS can be further developed to achieve the goals of intelligent, flexible and adaptive machines. Additionally, by integrating CPS with production, logistics and services in the current industrial practices, it will turn factory today into an industrial 4.0 factory with great economic potential (Lee et al., 2015). In paper (Shu et al., 2016), a comprehensive literature reviews the recent contributions, which focused on improving the Internet of Medical Things through the use of formal methodologies provided by the cyber-physical systems community. Shu Zhaogang et al. describe the practical application of the democratization of medical devices for both patients and health-care providers. In recent years, cloud computing has been widely used in CPS. So far, it has achieved rich research results. Cloud computing provides computing, storage, services and applications over the CPS, such as mobile cloud computing. Mobile cloud computing is defined as an integration of cloud computing technology with mobile devices to allow the mobile devices resource-full in computational power, memory, storage, energy, and context awareness (Yang et al., 2018; Huang et al., 2019; Chen et al., 2018b). In the infrastructure based mobile cloud, the hardware infrastructure remains static while also providing services to the mobile users. However, there are several applications use the cloud resources, but the usage is limited to storing and applying specific services such as Apples Siri (voice based personal assistant) and iCloud storage service (Stergiou et al., 2018).

Fog computing is a distributed computing paradigm that acts as a middle layer between cloud data centers and IoT equipment/sensors. Fog computing is a promising technique for CPS compared to cloud computing. It extends cloud computing to the fog of the network (Wang et al., 2018c), also enabling the new applications and services. It offers compute, net-working and storage facilities so as to Cloud-based services can be extended closer to the IoT devices/sensors (Qi et al., 2016). The concept of fog computing was first introduced by Cisco in 2012 to meet the challenges of IoT applications in traditional Cloud computing. A large number of IoT devices/sensors are highly distributed at the edge of the network along with real-time and latency-sensitive service requirements. Since Cloud data centers are geographically centralized, they are often unable to meet the storage and processing needs of billions of geo-distributed IoT devices/sensors (Pavel and Zdenek, 2017). As a result, congested network, high latency in service delivery and poor Quality of Service (QoS) are experienced (Xie et al., 2016; Bhuiyan et al., 2017). Typically, a fog computing environment consists of traditional networking components such as routers, switches, set top boxes, proxy servers, Base Stations (BS), etc (Arif et al., 2018). It can be placed closer to IoT devices/sensors. These components are provided with a variety of computing, storage, networking and other functions that can support the execution of service applications.

2.2. The methods to solve coupling resource management problem

There are some related solutions based on fog computing. In IoT system, there exist a tighter coupling between events and actions (Steiner and Poledna, 2016). All raw events are collected from sensors readers and processed solely on the Cloud, resulting in unnecessary data redundancy and network overhead. By processing this data locally based on fog computing, costs can be lowered and the overall responsiveness of this system can be improved (Giang et al., 2015). In paper (de Brito et al., 2018), a current line of action that leads to this direction is the development of IoT systems; considered as the coupling of physical processes and the digital world, its influence in the next industrial revolution is essential. In this work, the authors discuss its implementation, taking the fog computing paradigm into consideration and extending a standard-compliant machine-to-machine communication architecture to support container-based orchestration mechanisms. In paper (Magurawalage et al., 2015), the authors introduce a new vision for providing computing services for connected devices. It is based on the key concept that future computing resources will be coupled with

communication resources, for enhancing user experience of the connected users, and also for optimizing resources in the providers' infrastructures. Besides, such coupling is achieved by cooperative resource allocation algorithms, by integrating computing and communication services and by integrating hardware in networks. In paper (Zeng et al., 2016), the authors propose that there is no strict coupling on the computation and storage servers in the beginning. Then, the authors re-couple the storage and computation servers based on fog computing to minimize the overall task completion time with the consideration of transmission time.

In operating system, in order to solve the coupling resource management problem, some scheduling strategies are adopted to avoid the system deadlock. The most common way is first come first served (FCFS) method (Siahaan, 2016). The second common algorithm is the time slice rotation scheduling algorithm which uses the deprivation strategy. Time slice rotation scheduling algorithm also known as RR (Round-robin) scheduling algorithm (Sangwan et al., 2017). Each process is allocated a time period, called its time slice, the time allowed for the process to execute. The principle of the algorithm is as follows: let the ready process use the time slice for FCFS to rotate the scheduling mode of the CPU. All the ready processes in the system are arranged into a queue according to the FCFS principle, and the CPU is assigned to the team first process each time, so that it performs a time slice, and the length of the time interval is from several ms (millisecond) to hundreds ms.

In the database system, there are mainly the buffer management strategies to avoid coupling resource management problems. It is assumed that there are some memory buffers to store the required data, which provide the available memory buffer for the query on the database. As for the buffer queue, it can be modified when the system has some new read and write requests (Deng et al., 2018). The purpose of the buffer management is not only managing user's requests but also saving and minimizing the delay and reducing the unnecessary requirements as far as possible. In buffer management, the system uses scheduling policies in similar operating systems to replace the cache data, such as the least recently used (LRU), first in first out (FIFO), clock algorithm. For avoiding coupling resource management problem, the concurrency control method in the system can also solve this problem very well, such as based on the timestamp.

In other system about coupling resource management problem, researchers mostly study energy-efficient platform for different computing (Kim et al., 2018; Teng et al., 2019). However, there are few studies on coupling resource management problems in sensor-cloud systems. It is essential to design an algorithm to deal with these type of problem. In order to solve the problem of multiple distribution, the Kuhn-Munkres (*k-m*) algorithm is a classical method to deal with task assignment, and the authors improve *k-m* algorithm to propose solutions for *m-m* assignment problems (Zhu et al., 2016). In the work (Amponsah et al., 2016), Amponsah S K et al. propose a heuristic method based on Hungarian algorithm (Hamuda et al., 2018), this method can reduce computing time. In the paper (Krishnaveni and Prakash, 2019), Rodriguez M A et al. present the infrastructure as a service (IaaS) cloud scientific workflow resources allocation and scheduling strategy. The authors put forward optimization algorithm based on the Meta heuristic, using particle swarm optimization. Bhoi U, Ramanuj P N puts forward the improved Max-min task scheduling algorithm. The improved algorithm of the Max-min distributes tasks to resources to generate the maximum execution time of (maximum) minimum completion time (the slowest resources) (Bhoi and Ramanuj, 2013). In the work (Vizueteluciano et al., 2015), VizueteLuciano E et al. develop new assignment algorithms by using a wide range of aggregation operators in the Hungarian algorithm. The new process of using an ordered weighted average mean distance (OWAD) operator and the induction of OWAD (IOWAD) operator is introduced. The main advantage of this method is that a parameterized aggregation operator can be provided between minimum and maximum. Numerical results show that the proposed technique can provide approximate optimal performance with polynomial complexity.

The above study is the allocation of resources in some improvements of classic algorithms, like joining parameters in classical algorithm and combining with heuristic algorithm for processing. However, in the case of coupling resource management problem, the proposed algorithm cannot effectively meet the optimal matching of the users and the resources. Firstly, we do not have enough information to predict the realtime demand in the sensor network, and the combination of heuristic algorithms cannot solve this problem well. Secondly, if we call a longest or the shortest task firstly, it can lead to deadlock of system. In addition, the longest or the shortest resources are not the most urgent resources. Finally, the matching as mentioned above is about 1 to 1 while the matching of 1 to n is allowed in the coupling resource management problem.

To solve these problems, this paper proposes an extended Hungarian algorithm with buffer queue. The core idea of the method is to use the Hungarian algorithm to complete the initial matching, and determine whether the matched resources can continue to be scheduled. Then it is added to the initial matching to achieve the near-optimal match. In addition, we design a buffer queue based on the fog computing mode. The algorithm with the support of buffer queue can further improve the utilization of the resources to reduce the coupling computing and realize high-effective and sustainable system.

3. Coupling resource management problem

We first provide a formal definition of the coupling in Section 3.1, and then we give a specific example of coupling resource management problem in Section 3.2, which makes it easy for readers to understand the problem.

3.1. Problem definition

Definition 1: Assume that there are M (M_1, M_2, …, M_m) users and N (N_1, N_2, …, N_n) sensors (resources). One user can use the resources provided by the several sensors but one sensor can only be used by one user at the same time. When some users almost simultaneously request for the same physical node in the sensor cloud system, the cloud has a large delay in the underlying network. It is likely to feedback the "idle" results, resulting in conflicts, namely, coupling resource management problem. The average time taken by the system to complete all services is T. Therefore, the main objective is to find a matching for the users and sensors while maximizing the resource utilization to minimum the T.

3.2. A specific example

Here is an example showing a simple coupling resource management problem in sensor-cloud system.

As shown in Fig. 1, there are five different resources and seven users, assuming that the user calls one resource will take three time units. The relationship among the users and resources are as follows: User 1 calls resource B and C; User 2 calls resource A and E; User 3 calls resource B and D; User 4 calls resource A and E; User 5 calls resource D; User 6 calls resource C; User 7 calls resource D and E.

The results of different scheduling algorithms are shown in Table 1. The FIFO needs 4 times to complete the scheduling, accumulating 12 time units; HG takes 4 times to complete the scheduling, accumulating 12 time units; and the more effective algorithm (The more effective algorithm refers to the near-optimal schedule which proposed in this paper.) needs 3 times to complete the service, overall 9 time units.

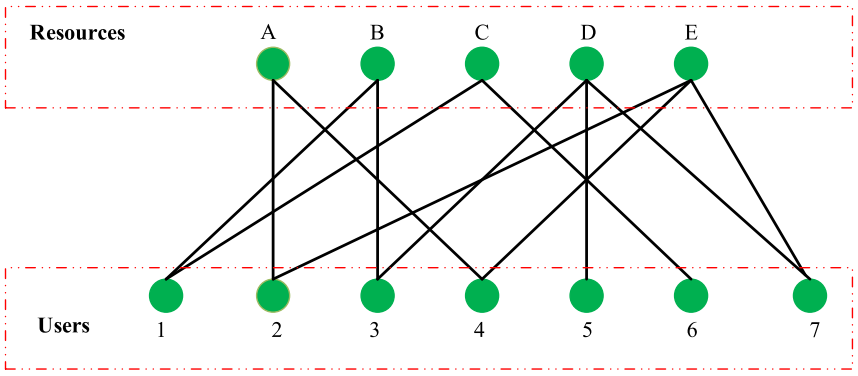

Fig. 1. An example of coupling resource management problem.

Table 1
Comparisons for different scheduling methods.

Timestamp	FIFO (First Input First Output)	HG (Hungarian algorithm)	More effective algorithm
1	1-B 1-C 2-A 2-E	4-A 3-B 1-C 5-D 2-E	4-A 3-B 1-C 5-D 2-E
2	3-B 3-D 4-A 4-E	2-A 1-B 6-C 3-D 5-E	2-A 1-B 6-C 3-D 5-E
3	5-D 6-C	7-D	7-D 7-E
4	7-D 7-E	7-E	completed

4. Algorithms design and analysis

The comprehensive design of Hungarian algorithm is introduced in Section 4.1, the detailed steps of algorithms are in Section 4.2, and analyses are discussed in Section 4.3.

4.1. Algorithms design

The Hungarian algorithm mainly solves the problem of assignment or maximum matching. The framework of the Hungarian algorithm is as follows: Assume M is a maximum match for bipartite graph G. First of all, M be the empty set. Secondly, a larger matching M may be found instead of M though augmenting path. Thirdly, second process will be repeated until the augmenting path cannot be found. The problem supposes n individuals and n tasks, each person can only complete a task and each task can only be completed by a person. The problem is that the user and the resource are not a 1-to-1 correspondence, but a m-to-n correspondence, and even in a match the user can still call multiple resources. Thus, we add a loop to the Hungarian algorithm. When the number of corresponding relationships are equal to the sum of all the matching numbers, the loop is finished. If there are unused resources after the initial match, the user continue to call the resources to maximize the resource utilization.

At the same time, we put the algorithm on the fog computing layer for processing. A buffer queue is designed in the fog computing layer which returns the result directly. The number of resources in the buffer queue accounts for 1/4 of the total resources. These resources are determined by the size of the number of requests which are received in the

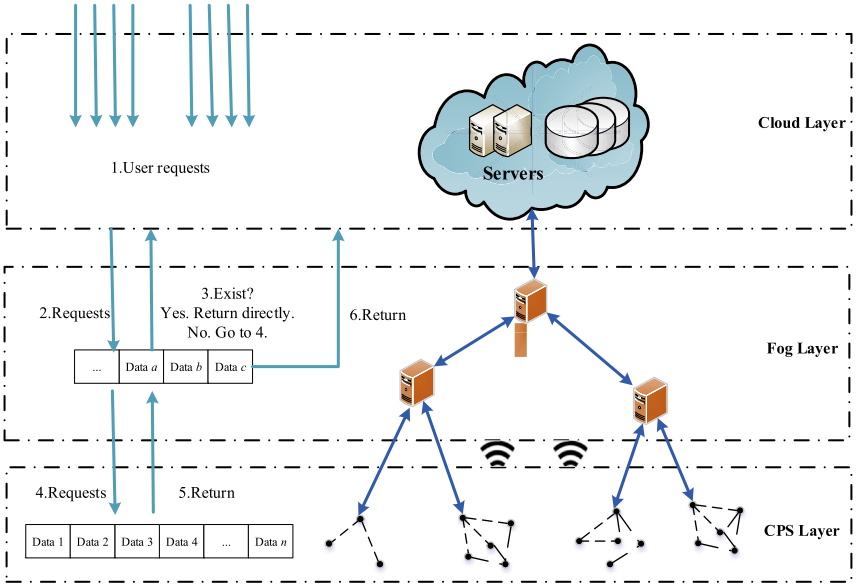

Fig. 2. An illustration of the three-tier architecture in fog computing framework.

cloud. On the one hand, it can process order or data in the fog computing layer to further reduce the scheduling time; on the other hand, fog node has better understanding of the underlying sensor node, and the security privacy of the resources will be better. The three-tier architecture of the fog computing is shown in Fig. 2.

In fog computing framework, the fog computing layer is the intermediate tier between the upper and lower layers of cloud and CPS respectively: Firstly, the fog computing layer controls the underlying node because the fog computing layers information on the underlying node is more direct and timely than the cloud. In addition, the fog computing layer can merge the multiple duplicate instructions to reduce the number of calls for physical node. Secondly, transmission of information should be minimized, especially the control information. The information transmitted in the sensor-cloud system is divided into data information and control information. In order to reduce the coupling degree, the control information should be reduced between the sensor and the cloud. The fog computing layer can effectively reduce the control message by making initial processing of the data.

At the same time, we put the algorithm on the fog computing layer for processing. A buffer queue is designed in the fog computing layer which returns the result directly. The number of resources in the buffer queue accounts for 1/4 of the total resources. The reason why we choose 1/4 is that storage capacity of fog computing is limited and it shows best results in experiment.

Algorithm 1 Caching queue algorithm
Input: cost matrix, location of fog node;
Output: cost matrix after caching queue;
1: every physic nodes has same initial energy E;
2: map = cost matrix;
3: vector < *integer* > cacheQueue.size() = $X/4 + 1$
4: **for** each line (session) in map **do**
5: find maximum time for each session(*maxTime*), get the hop to the fog node of each session(T_{hop}), get each node's remained energy($E_{residual}$)
6: $S = \frac{maxTime[i]}{\sum_1^n maxTime[i]} + \frac{T_{hop}[i]}{\sum_1^n T_{hop}[i]} + \frac{E_{residual}[i]}{\sum_1^n E_{residual}[i]}$;
7: sort(S[]);
8: **for** each element in cacheQueue **do**
9: **if** S[] > queue item.time **then**
10: cacheQueue.insert(this line);
11: cacheQueue.popout(last);
12: **end if**
13: **end for**
14: **end for**
15: **for** every element in cacheQueue **do**
16: put element in queue;
17: **end for**

4.2. The detailed steps of algorithms

The detailed steps of algorithms are illustrated in Fig. 3. The input is a cost matrix, where the cost represents the service time need of current user associated with the physical node. In the first phase, we judge whether the cost matrix is *O* matrix. If it is *O* matrix, there is no conflict; if it is not *O* matrix, then it goes to the second phase. The second phase is to use the buffer queue to preprocess the cost matrix to reduce its size. The third phase is to use the extended Hungarian algorithm to achieve the maximum matching and complete the initial matching. The fourth phase is to determine whether there is an unused resource to be scheduled, and add it to the initial match to achieve near-optimal matching. Repeat the above process and solve the coupling resource management problem in the shortest time. The details description of all phases are as follows.

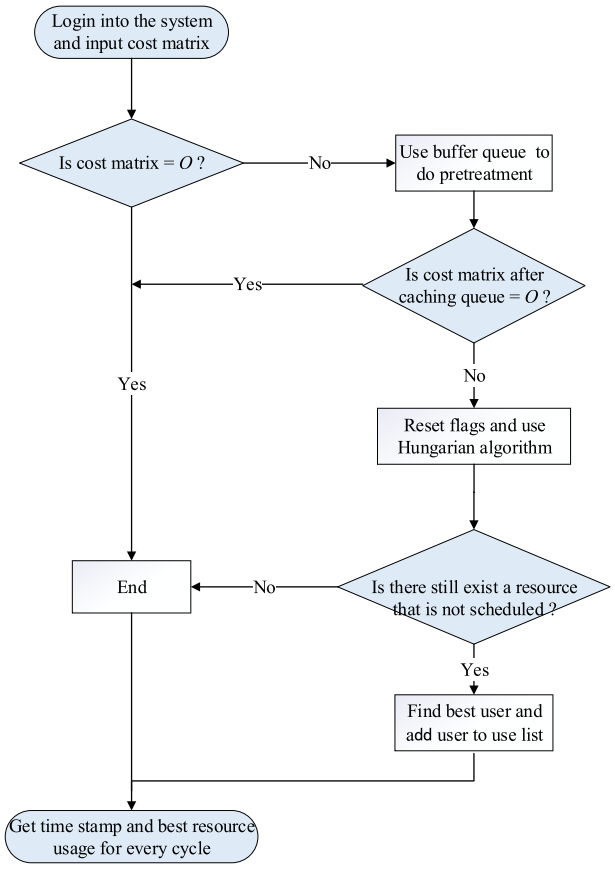

Fig. 3. An illustration of the detailed steps of algorithms.

Algorithm 1 shows the algorithm designed for caching queue. The input of the algorithm is current scheduling matrix and the location of fog nodes, and the output is the matrix after caching queue reprocessing. Limited by the size of actual caching queue, in our simulation experiment, we set the caching capacity on the fog layer to 1/4 of the total amount of data generated by the underlying sensor network on a particular moment. In the algorithm, the cost of energy are taken into account. We assume all of the sensors have same energy in the beginning. In the caching process, physics nodes has have priority over other nodes for caching data, when they have more requests, less transmission hops and more rest of its energy.

Algorithm 2 gives the detailed steps to extend the Hungarian algorithm. Firstly, the input and output of the method are the cost matrix after caching queue and the matching results. After that, using the HG to find the maximum matching and matching results. Finally, we added the detection of scheduling completion status. For the unmatched resources, we check whether it can be further scheduled, which can achieve a better match. Also, we will strip any finished sessions.

4.3. Theoretical analysis

Theorem 1. *Suppose M is a match of the bipartite graph. M being necessary and sufficient conditions for the maximum matching is that the augmenting path is not existing.*

Proof. Necessity: *M* is the maximum match → there's an augmenting path.

Assume that *M* is the largest match and there is an augmenting path. According to the definition of the maximum match we can know that

the matching number is the biggest. If there is an augmenting path in the graph, there must be a new match increasing 1 by the original match. The original match is the biggest match. There is no augmenting path for maximum match.

Algorithm 2 Extension of Hungarian algorithm
Input: cost matrix after caching queue
Output: the matching results

```
while sessions not finished do //determine whether the cost matrix is a 0 matrix
  time stamp ++;
  for every session in map do
    bool finished = true;
    if finished then
      running = false; //not all sessions are running, need relocate resources
      strip finished session, move out of matrix;
    else
      move to next session;
    end if
  end for
  if running then
    bool consumed = false;
    every flag try to use resource;
    if any resource < 0 then
      consumed = true; //resource attempted use failed, flag current resource consumed and roll back needed
    end if
    if not consumed then
      usage this cycle and move to next cycle, roll back resource usage;
    else//roll back
    end if
  end if
  reset flag;
  use Hungarian Algorithm (Jonker and Volgenant, 1986);
  check the unused resources and add them to use list;
end while
```

Sufficiency: The augmenting path ence $\rightarrow M$ is the maximum match.

Suppose there is a greater match M' and $|M'| = k$, finding the augmenting path P of M'. P contains $2K + 1$ edge, where k belongs to M', $k+1$ does not belong to M'. Modify M' for M'& P (M' and P symmetry difference operation) that $|M'| + 1$, there is augmenting path. □

Theorem 2. *For the bipartite graph G, we define a match M. There is a necessary and sufficient condition for M to be saturated in X. For any subset A of X, the adjacent point sets of A is T(A) and $|T(A)| > = |A|$ has always been established.*

Proof. Necessity: according to the definition of saturation, each vertex of X is paired with $T(A)$ at different vertices in M, so the $T(A)$ >= |A| is established, and the necessity is established.

Sufficiency: Assuming there exists a bipartite graph $G = < V_1, V_2, E >$ and there exists $a_1, a_2 \in V_1, x \in V_2, (a_1, x), (a_2, x) \in E$. As shown in Fig. 4, any of (a_i, x) is deleted, which will destroy the condition. So, $A_1, A_2 \in V_1, a_i \in A_i$ and a_1 join along x, $|T(A_i)| = |(A_i)|$. So $|T(A_1) \cap T(A_2)| \geq |T(A_1 - \{a_1\}) \cap T(A_2 - \{a_2\})| + 1 \geq |T(A_1 \cap A_2)| + 1 \geq |A_1 \cap A_2| + 1$. $|T(A_1 \cup A_2)| = |T(A_1) \cup T(A_2)| \geq |T(A_1))| + |T(A_2)| - |T(A_1) \cap T(A_2)| \leq |A_1| + |A_2| - |A_1 \cap A_2 + 1| = |A_1 \cup A_2| - 1$. This is in contradiction with the known conditions. Accordingly, the theorem is proved. □

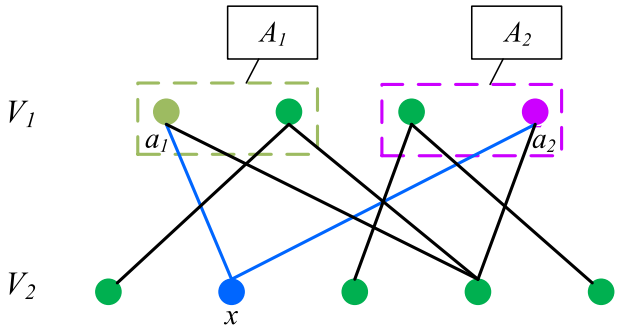

Fig. 4. An illustration of Theorem 2.

Theorem 3. *Assuming that the user's resources usage time must be t time intervals, if there are M users (M_1, M_2, ···, M_m), N resources (N_1, N_2, ···, N_n), M users using the number of resources are P (P_1, P_2, ···, P_m), then FIFO method takes X($X = M \times t$) time intervals. Using the extended Hungarian algorithm, the near-optimal time used is Y($Y = \sum(P_i)$) and $Y < X$.*

Proof. For the scheduling situation, each time can only meet the needs of a user. The larger the number of users, the longer the scheduling time, and the total time required is the number of users × use time which is $M \times t$. Every time the resources can be used, the resource utilization will be largest, and natural time will be the shortest. The time it takes for the user to call all the resources ÷ the number of resources which is $\sum(P_i)/N \times t$. We know that $M \times N$ means that each user uses all the resources, obviously the value must be bigger than $\sum(P_i)$, that is, $\sum(P_i)/N \times t < M \times t$. The original proposition $Y < X$ proved. □

Theorem 4. *The time complexity of the proposed algorithm is O ($n \times (n + m)$)*

Proof. We make the bipartite graph $G = < V, U, E >$ store in the adjacency matrix. The two types of vertices are $|U|$ and $|V|$, where $n = |U| + |U|$. The number of correspondence is m where $m = |E|$. The basic statement in the algorithm is the number of matches increases. This basic statement is contained in a loop and a recursive lookup function. So the basic statement is executed for $n \times (n + m)$ times, the time complexity of the algorithm is O ($n \times (n + m)$). The time complexity of the algorithm is polynomial time. □

5. Experiment

In this section, extensive experiments are conducted to validate the performance of our proposed solutions by using Visual Studio and MATLAB R2016a. In Section 5.1, the distribution of the initial network is shown in Table 2. Comparison and analysis of four different methods of performance results are shown in Section 5.2, which are FIFO, HG, EHG (Extended Hungarian algorithm) and EHGB (Extended Hungarian algorithm with buffer queue), respectively.

5.1. Experimental environment settings

This section describes the basic settings of the experimental environment. The experiment constructs five experimental scenarios. In Scene 1, the number of users is 7, and the number of resources is 5. In Scene 2, the number of users is 20, and the number of resources is 15. In Scene

Table 2
Experiment parameters.

Parameter	scene 1	scene 2	scene 3	scene 4	scene 5
Users	7	20	40	80	160
Resources	5	15	30	60	120

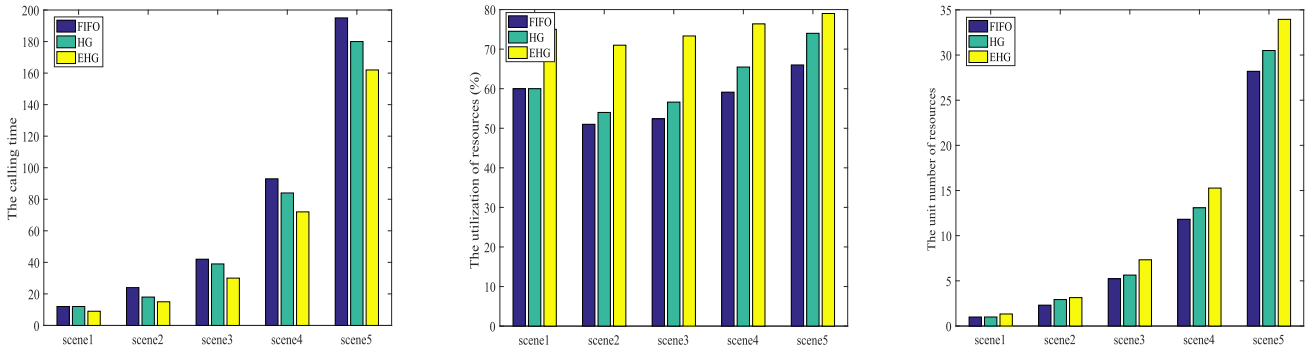

Fig. 5. Performance of three different methds (a) Comparison of total scheduling times in different scenes. (b) Comparison of resource utilization in different scenes. (c) Comparison of unit number of resources in different scenes.

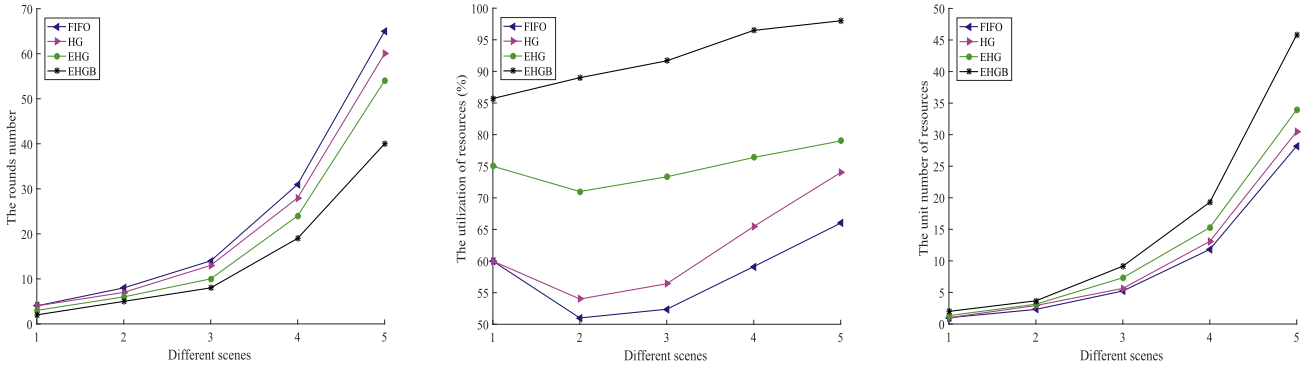

Fig. 6. Performance of three different methds (a) Comparison of rounds number in different scenes. (b) Comparison of resource utilization in different scenes. (c) Comparison of unit number of resources in different scenes.

3, the number of users is 40, and the number of resources is 30. In Scene 4, the number of users is 80, and the number of resources is 60. In Scene 5, the number of users is 160, and the number of resources is 120. The sum of resources invoked by the users are all random values. As for the parameters, some simulation results are unavailable, because they are far from the real world. The parameters used for experiments are shown as typical values. Considering the complexity of the experimental scenario, this section describes the experimental environment no longer adding a legend. The legend can refer to the example in Section 3.2.

5.2. Analysis of experimental results

In order to compare the advantages of our proposed method, we compare it with FIFO, HG and EHG methods respectively. Among them, HG algorithm is the classic Hungarian algorithm, which is the most commonly used method solving the maximum matching of bipartite graphs. FIFO is the method of user scheduling, which follows the principle of "first come, first serve". EHGB is an extension of EHG which introduces buffer queue in fog computing layer to reduce coupling computing and realize high-effective system. Following calculations: The number of rounds can get through the scheduling results by the program. The total time = the number of rounds × user calls each resources time. Resources utilization = total number of resources calls ÷ (the number of round × resources number); the unit number of resources = total number of resources calls ÷ total time. The user calls each resources to take 3 time intervals.

Fig. 5(a) shows the calling time of the three scenarios under these three algorithms. In the Scene 1, the EHG algorithm obtains the call time for 9 time intervals, while the HG and the FIFO are 12 time intervals. Further analysis shows that the EHG time is shortened by 33.3%, compared with the FIFO and the HG. In the Scene 2, the EHG algorithm obtains the call time for 18 time intervals, while the HG and the FIFO are 24 and 21 time intervals. We can know that, during comparing with the FIFO and the HG, the time is reduced by 25%, 14.2%. In the Scene 3, the EHG algorithm obtains the call time for 30 time intervals, while the HG and the FIFO are 39 and 42 time intervals. We can know that, during comparing with the FIFO and the HG, the time was reduced by 40%, 30%. In the Scene 4, the EHG algorithm obtains the call time for 72 time intervals, while the HG and the FIFO are 84 and 93 time intervals. Therefore, compared with the FIFO, the EHG time is shortened by 29.2%; compared with the HG, its time is shortened by 16.7%. In the Scene 5, the EHG algorithm obtains the call time for 162 time intervals, while the HG and the FIFO are 180 and 195 time intervals. So, compared with the FIFO, the EHG time is shortened by 22.4%; compared with the HG, its time is shortened by 11.1%.

Fig. 5(b) shows the resource utilization of the three scenarios under these three algorithms. In the Scene 1, the EHG algorithm obtains the resource utilization for 75%, while the HG and the FIFO are 60%. Further analysis shows that the resource utilization of EHG is raised by 25%, compared with the FIFO and the HG. In the Scene 2, the EHG algorithm obtains the resource utilization for 71%, while the HG and the FIFO are 54.4% and 51.2%. We can know that, during comparing

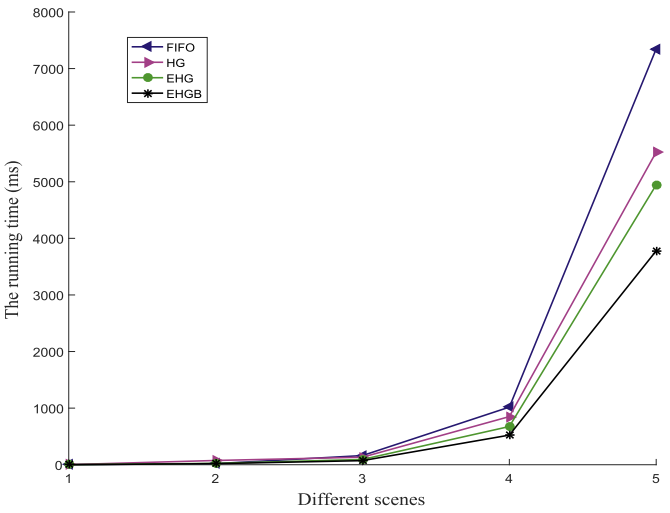

Fig. 7. Comparison of the running time in different scenes.

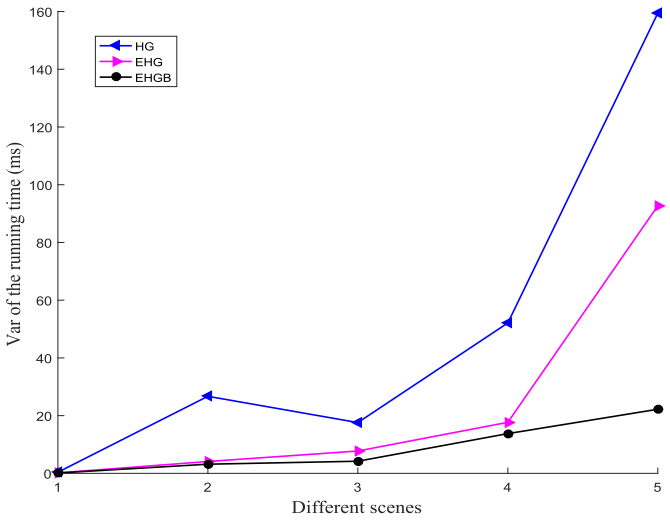

Fig. 8. Comparison of running time variance in different scenes.

with the FIFO and the HG, the resource utilization was enhanced by 38.7%, 30.5%. In the Scene 3, the EHG algorithm obtains the resource utilization for 73.3%, while the HG and the FIFO are 56.4% and 52.4%. We can know that, during comparing with the FIFO and the HG, the resource utilization was enhanced by 39.9%, 29.9%. In the Scene 4, the EHG algorithm obtains the resource utilization for 76.4%, while the HG and the FIFO are 65.5% and 59.1%. So, compared with the FIFO, the EHG time is raised by 29.3%; compared with the HG, its time is raised by 16.7%. In the Scene 5, the EHG algorithm obtains the resource utilization for 79.4%, while the HG and the FIFO are 75.3% and 66.7%. So, compared with the FIFO, the EHG time is raised by 19.1%; compared with the HG, its time is raised by 5.2%.

Fig. 5(c) shows the unit number of resources of these three scenarios under the three algorithms. In the Scene 1, the EHG algorithm obtains the unit number of resources for 1.33, while the HG and the FIFO are 1. Further analysis shows that the unit number of resources of EHG is raised by 33.3%, compared with the FIFO and the HG. In the Scene 2, the EHG algorithm obtains the unit number of resources for 3.14, while the HG and the FIFO are 2.93 and 2.31. We can know that, during comparing with the FIFO and the HG, the unit number of resources was enhanced by 35.9%, 7.2%. In the Scene 3, the EHG algorithm obtains the unit number of resources for 7.33, while the HG and the FIFO are 5.64 and 5.24. We can know that, during comparing with the FIFO and the HG, the unit number of resources was enhanced by 39.9%, 29.9%. In the Scene 4, the EHG algorithm obtains the unit number of resources for 15.27, while the HG and the FIFO are 13.09 and 11.82. So, compared with the FIFO, the unit number of resources of the EHG is raised by 29.3%; compared with the HG, it is raised by 16.7%. In the Scene 5, the EHG algorithm obtains the unit number of resources for 33.95, while the HG and the FIFO are 30.5 and 28.2. So, compared with the FIFO, the unit number of resources of the EHG is raised by 11.3%; compared with the HG, it is raised by 20.4%.

The total rounds number results are shown in Fig. 6(a), the resource utilization results are shown in Fig. 6(b), and the unit number of resources results are shown in Fig. 6(c). It is seen that EHGB spends the least amount of round and gets the largest resource utilization and unit number of resources. The reason for this is that EHG with the aid of buffer queue can process some data in fog computing layer.

As for the cost of the proposed algorithm, we have done comprehensive experiments based on the five scenes and compared with other related methods. The results are shown in Fig. 7. The delay of EHGB are 0.92 ms, 18.89 ms, 71.92 ms, 523.02 ms and 3778.92 ms in different scenes. In a word, when the scale of the scene is small, the delay is only a few milliseconds. When the scene is gradually complicated, the delay increases. The shorter delay means higher resource utilization and smaller scheduling time. What's more, Fig. 8 is the comparison of delay variance. It can be seen that our method performs more stably.

These results show that the EHGB takes the shortest calling time, the least calling round, and the largest resource utilization. The reason is that our method can combine the advantages of the classic EHG and FIFO, which can maximize the resource utilization. In addition, with the support of buffer queue, fog computing layer can provide pretreatment which can further shorten calling time and improve the resource utilization to reduce coupling computing and realize high-effective system.

6. Conclusion

The smart city system combines CPS and cloud computing and has gradually become a research hotspot. Aiming at overcoming coupling resource management problem, a mechanism based on the fog computing is designed, in which the mechanism ensures the near-optimal resource management. The fog computing layer is the intermediate tier between the upper and lower layers of cloud and CPS respectively. Besides, the EHGB introduces buffer queue in fog computing layer to reduce coupling computing. When there are some malicious nodes and user's requests in the system, fog layer can cache the conflicting node's data which will avoid system conflicts. Experimental results and theoretical analysis show this method can solve the coupling resource management problem efficiently to realize sustainable smart city system.

Acknowledgment

This work is supported by grants from the General Projects of Social Sciences in Fujian Province under grant No. FJ2018B038 and National Natural Science Foundation of China (NSFC) under grant No. 61872154, No. 61772148 and No. 61672441 and Natural Science Foundation of Fujian Province of China (No. 2018J01092) and the Fujian Provincial Outstanding Youth Scientific Research Personnel Training Program and National China 973 Project No. 2015CB352401 and NSFC Key Project No. 61532013 and No. 61872239 and FDCT/0007/2018/A1 and DCT-MoST Joint-project No. (025/2015/AMJ) and University of Macau grant Nos: MYRG2018-00237-RTO, CPG2019-00004-FST and SRG2018-00111-FST of SAR Macau, China and Subsidized Project for Postgraduates' Innovative Fund in Scientific Research of Huaqiao University (No. 17013083005).

References

Al Omar, A., Bhuiyan, M.Z.A., Basu, A., Kiyomoto, S., Rahman, M.S., 2019. Privacy-friendly platform for healthcare data in cloud based on blockchain environment. Future Gener. Comput. Syst., https://doi.org/10.1016/j.future.2018.12.044.

Amponsah, S.K., Otoo, D., Salhi, S., Quayson, E., 2016. Proposed heuristic method for solving assignment problems. Am. J. Oper. Res. 6 (6), 436–441.

Arif, M., Wang, G., Balas, V.E., 2018. Secure vanets: trusted communication scheme between vehicles and infrastructure based on fog computing. Stud. Inf. Contr. 27 (2), 235–246.

Bhoi, U., Ramanuj, P.N., 2013. Enhanced max-min task scheduling algorithm in cloud computing. Int. J. Appl. Innov.. Eng. Manag. 2 (4), 259–264.

Bhuiyan, M.Z.A., Wang, G., Wu, J., Cao, J., Liu, X., Wang, T., 2017. Dependable structural health monitoring using wireless sensor networks. IEEE Trans. Dependable Secure Comput. 14 (4), 363–376.

Chen, Y., Tang, S., Pei, S., Wang, C., Du, J., Xiong, N., 2018. Dheat: a density heat-based algorithm for clustering with effective radius. IEEE Trans. System, Man., Cybernetics: Syst. 48 (4), 649–660.

Chen, Y., Tang, S., Zhou, L., Wang, C., Du, J., Wang, T., Pei, S., 2018. Decentralized clustering by finding loose and distributed density cores. Inf. Sci. 433, 510–526.

de Brito, M.S., Hoque, S., Steinke, R., Willner, A., Magedanz, T., 2018. Application of the fog computing paradigm to smart factories and cyber-physical systems. Trans. Emerg. Telecommun. Technol. 29 (4), https://doi.org/10.1002/ett.3184.

Deng, Z., He, T., Ding, W., Cao, Z., 2018. A multimodel fusion engine for filtering webpages. IEEE Access 6, 66062–66071.

Giang, N.K., Blackstock, M., Lea, R., Leung, V.C., 2015. Developing iot applications in the fog: a distributed dataflow approach. In: 2015 5th International Conference on the IoT. IEEE, pp. 155–162.

Hamuda, E., Mc Ginley, B., Glavin, M., Jones, E., 2018. Improved image processing-based crop detection using kalman filtering and the Hungarian algorithm. Comput. Electron. Agric. 148, 37–44.

Huang, M., Liu, A., Xiong, N.N., Wang, T., Vasilakos, A.V., 2019. A low-latency communication scheme for mobile wireless sensor control systems. IEEE Trans. Syst. Man Cybernetics-Syst. 49 (2), 317–332.

Jonker, R., Volgenant, T., 1986. Improving the Hungarian assignment algorithm. Oper. Res. Lett. 5 (4), 171–175.

Kim, D., Hong, J.-E., Chung, L., 2018. Investigating relationships between functional coupling and the energy efficiency of embedded software. Software Qual. J. 26 (2), 491–519.

Krishnaveni, H., Prakash, V.S.J., 2019. Execution time based sufferage algorithm for static task scheduling in cloud. In: Advances in Big Data and Cloud Computing. Springer, pp. 61–70.

Lee, J., Bagheri, B., Kao, H.-A., 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufactur. Lett. 3, 18–23.

Liang, Y., Wang, T., Bhuiyan, M.Z.A., Liu, A., 2017. Research on coupling reliability problem in sensor-cloud system. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, pp. 468–478.

Liu, Y., Quan, W., Wang, T., Wang, Y., 2018. Delay-constrained utility maximization for video ads push in mobile opportunistic d2d networks. IEEE Int. Things J. 5 (5), 4088–4099.

Liu, Y., Liu, A., Liu, X., Huang, X., 2019. A statistical approach to participant selection in location-based social networks for offline event marketing. Inf. Sci. 480, 90–108.

Luo, E., Bhuiyan, M.Z.A., Wang, G., Rahman, M.A., Wu, J., Atiquzzaman, M., 2018. Privacyprotector: privacy-protected patient data collection in iot-based healthcare systems. IEEE Commun. Mag. 56 (2), 163–168.

Magurawalage, C.S., Yang, K., Wang, K., 2015. Aqua Computing: Coupling Computing and Communications. arXiv. doi:arXiv:1510.07250.

Pavel, M., Zdenek, B., 2017. Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Survey. Tutorials 19 (3), 1628–1656.

Qi, L., Dou, W., Chen, J., 2016. Weighted principal component analysis-based service selection method for multimedia services in cloud. Computing 98 (1–2), 195–214.

Qi, L., Zhang, X., Dou, W., Ni, Q., 2017. A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data. IEEE J. Sel. Area. Commun. 35 (11), 2616–2624.

Qi, L., Yu, J., Zhou, Z., 2017. An Invocation Cost Optimization Method for Web Services in Cloud Environment, Scientific Programming. , https://doi.org/10.1155/2017/4358536.

Qi, L., Dai, P., Yu, J., Zhou, Z., Xu, Y., 2017. time–location–frequency aware internet of things service selection based on historical records. Int. J. Distributed Sens. Netw. 13 (1), 1–13.

Qi, L., Meng, S., Zhang, X., Wang, R., Xu, X., Zhou, Z., Dou, W., 2018. An exception handling approach for privacy-preserving service recommendation failure in a cloud environment. Sensors 18 (7), 20–37.

Qi, L., Dou, W., Wang, W., Li, G., Yu, H., Wan, S., 2018. Dynamic mobile crowdsourcing selection for electricity load forecasting. IEEE Access 6, 46926–46937.

Sangwan, P., Sharma, M., Kumar, A., 2017. Improved round robin scheduling in cloud computing. Adv. Comput. Sci. Technol. 10 (4), 639–644.

Shu, Z., Wan, J., Zhang, D., Li, D., 2016. Cloud-integrated cyber-physical systems for complex industrial applications. Mobile Network. Appl. 21 (5), 865–878.

Siahaan, A.P.U., 2016. Comparison analysis of cpu scheduling: Fcfs, sjf and round robin. Int. J. Exp. Diabetes Res. 4 (3), 124–132.

Steiner, W., Poledna, S., 2016. Fog computing as enabler for the industrial internet of things. E I Elektrotechnik Inf. 133 (7), 310–314.

Stergiou, C., Psannis, K.E., Kim, B.-G., Gupta, B., 2018. Secure integration of iot and cloud computing. Future Gener. Comput. Syst. 78, 964–975.

Teng, H., Liu, Y., Liu, A., Xiong, N.N., Cai, Z., Wang, T., Liu, X., 2019. A novel code data dissemination scheme for internet of things through mobile vehicle of smart cities. Future Gener. Comput. Syst. 94, 351–367.

Vizueteluciano, E., Merig, J.M., Gillafuente, A.M., Boriareverter, S., 2015. Decision making in the assignment process by using the Hungarian algorithm with owa operators. Technol. Econ. Dev. Econ. 21 (5), 684–704.

Wang, T., Zeng, J., Lai, Y., Cai, Y., Tian, H., Chen, Y., Wang, B., 2017. Data collection from wsns to the cloud based on mobile fog elements. Future Gener. Comput. Syst., https://doi.org/10.1016/j.future.2017.07.031.

Wang, T., Zhou, J., Liu, A., Bhuiyan, M.Z.A., Wang, G., Jia, W., 2018. Fog-based computing and storage offloading for data synchronization in iot. IEEE Int. Things J., https://doi.org/10.1109/JIOT.2018.2875915.

Wang, T., Zhang, G., Bhuiyan, M.Z.A., Liu, A., Jia, W., Xie, M., 2018. A novel trust mechanism based on fog computing in sensor–cloud system. Future Gener. Comput. Syst., https://doi.org/10.1016/j.future.2018.05.049.

Wang, T., Zhang, G., Liu, A., Bhuiyan, M.Z.A., Jin, Q., 2018. A secure iot service architecture with an efficient balance dynamics based on cloud and edge computing. IEEE Int. Things J., https://doi.org/10.1109/JIOT.2018.2870288.

Xie, K., Wang, X., Wen, J., Cao, J., 2016. Cooperative routing with relay assignment in multiradio multihop wireless networks. IEEE/ACM Trans. Netw. 24 (2), 859–872.

Xu, X., Bhuiyan, M.Z.A., Qi, L., Zhang, X., Dou, W., 2018. Load aware management of cloudlets for a wireless area metropolitan network. In: 2018 IEEE Smart World Congress. IEEE, pp. 1283–1288.

Yang, L., Han, Z., Huang, Z., Ma, J., 2018. A remotely keyed file encryption scheme under mobile cloud computing. J. Netw. Comput. Appl. 106, 90–99.

Zeng, D., Gu, L., Guo, S., Cheng, Z., Yu, S., 2016. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans. Comput. 65 (12), 3702–3712.

Zhang, G., Wang, T., Wang, G., Liu, A., Jia, W., 2018. Detection of Hidden Data Attacks Combined Fog Computing and Trust Evaluation Method in Sensor-Cloud System, Concurrency and Computation: Practice and Experience. , https://doi.org/10.1002/cpe.5109.

Zhu, H., Liu, D., Zhang, S., Zhu, Y., Teng, L., Teng, S., 2016. Solving the many to many assignment problem by improving the kuhn-munkres algorithm with backtracking. Theor. Comput. Sci. 618, 30–41.