Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Computer Communications 207 (2023) 36–45

Contents lists available at ScienceDirect

Computer Communications

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

A game theory-based COVID-19 close contact detecting method with edge computing collaboration

Yue Shen [a,b], Bowen Liu [a], Xiaoyu Xia [d], Lianyong Qi [e], Xiaolong Xu [f], Wanchun Dou [a,b,c,*]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China
[b] Guangdong Laboratory of Artificia Intelligence and Digita Economy(SZ), Shenzhen, China
[c] College of Big Data and Intelligent Engineering, Southwest Forestry University, Kunming, China
[d] School of Computing Technologies, RMIT University, Melbourne, Australia
[e] College of Computer and Software, China University of Petroleum (East China), China
[f] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China

ARTICLE INFO

Keywords:
Game theory
COVID-19
Edge computing
Decentralized method

ABSTRACT

People all throughout the world have suffered from the COVID-19 pandemic. People can be infected after brief contact, so how to assess the risk of infection for everyone effectively is a tricky challenge. In view of this challenge, the combination of wireless networks with edge computing provides new possibilities for solving the COVID-19 prevention problem. With this observation, this paper proposed a game theory-based COVID-19 close contact detecting method with edge computing collaboration, named GCDM. The GCDM method is an efficient method for detecting COVID-19 close contact infection with users' location information. With the help of edge computing's feature, the GCDM can deal with the detecting requirements of computing and storage and relieve the user privacy problem. Technically, as the game reaches equilibrium, the GCDM method can maximize close contact detection completion rate while minimizing the latency and cost of the evaluation process in a decentralized manner. The GCDM is described in detail and the performance of GCDM is analyzed theoretically. Extensive experiments were conducted and experimental results demonstrate the superior performance of GCDM over other three representative methods through comprehensive analysis.

1. Introduction

Novel coronavirus pneumonia is a new acute respiratory infectious disease [1]. The World Health Organization (WHO) named it as COVID-19 and declared that COVID-19 is a public health emergency of international concern (PHEIC) [2]. Since December 2019, the COVID-19 epidemic has continued to spread around the world. It not only poses severe tests to public medical and health systems [3] but also has a huge impact on economic and trade activities [4]. According to WHO, as of December 23, 2022, there had been over 650 million confirmed cases and over 6.6 million deaths worldwide, making COVID-19 one of the deadliest epidemics in human history [5]. Therefore, the use of scientific methods to detect the spread of the epidemic has important practical significance [6]. Using mathematical methods to extrapolate the spread and progression of COVID-19 can help researchers and policymakers understand how the virus spreads and provide important references for formulating countermeasures, playing an important role in the epidemic response.

Detecting traces of patients with COVID-19 is a prerequisite for avoiding the virus's rapid spread [7]. The patients' close contact should be recognized and quarantined as quickly as possible. However, because of the related community's great mobility and density, detecting close contacts of all confirmed cases during the incubation period is tough. Modern life style provides a new view to solve this problem. Nowadays, almost everyone has a mobile phone. Due to the limited coverage of base stations, mobile phones will constantly switch the connected base stations as the user's geographical location changes [8]. Therefore, the base station connection data in the mobile phone records the user's spatio-temporal data. Traditionally, such computing tasks have been performed only in the cloud center [9], based on large amounts of data about a user's location, resulting in extremely high transmission latency. In addition, uploading all users' temporal and spatial information together to the cloud center will also lead to the leakage of users' privacy data because the cloud faces the problem of single point of failure, and long transmission process will increase the risk of privacy leakage. Now, a new computing paradigm can solve these problems appropriately, which is called edge computing [10].

In recent years, edge computing has emerged as a promising computing paradigm for addressing problems by bringing resources to the

* Corresponding author.
E-mail addresses: yueshen@smail.nju.edu.cn (Y. Shen), liubw@smail.nju.edu.cn (B. Liu), xiaoyu.xia@rmit.edu.au (X. Xia), lianyongqi@gmail.com (L. Qi), njuxlxu@gmail.com (X. Xu), douwc@nju.edu.cn (W. Dou).

https://doi.org/10.1016/j.comcom.2023.04.029
Received 31 December 2022; Received in revised form 9 March 2023; Accepted 22 April 2023
Available online 8 May 2023

network's edge, which can reduce service latency and avoid the uploading of sensitive data [11]. It has been widely applied in many fields such as task offloading and data caching [12,13]. In an area powered by edge computing, edge servers are located beside the access points or base stations [14]. In the edge computing environment, vendors can hire resources according to the same pay-as-you-go business pattern as in cloud computing [15]. By sending users' spatio-temporal data to the edge servers, the possibility of a user becoming a close contact with a COVID-19 patient can be detected. Furthermore, thanks to the feature of the edge computing paradigm, users' privacy spatio-temporal data are sent to different edge servers in close proximity. The short transmission process and distribution of edge servers' location alleviate the problem of privacy leakage to a great extent [16]. However, edge servers' computing and storage capabilities are indeed limited compared with cloud computing [17], which may not meet the computation or time-efficient requirements of all detecting activities. This poses a significant difficulty to the edge computing collaborative COVID-19 close contact detection, which must be addressed properly.

Users need to complete their detection events before the specified deadline and pay for the resources that they consume [18]. In the edge computing diagram, there may be many offloading decisions that fit the time limitation, thus each selfish user will choose the option which is the most favorable to itself. From a system standpoint, real-time resources are constrained, which may not suit the needs of all users. Failing to detecting in time may increase the risk of getting infected and speed up the spread of COVID-19. Therefore, while optimizing latency and cost per user, it should be ensured that as many users as possible can finish the detection event in time. Thus, improving the COVID-19 close contact detection completion rate while decreasing the overall system latency and cost are two main purposes of the COVID-19 close contact detecting problem under the edge computing environment.

What is more, even though distributed edge servers show advantage over central cloud in terms of privacy protection, the advantage will disappear if we design a centralized mechanism relying on a central organization to collection all users' information and decide the offloading decisions, which faces similar problems to traditional cloud computing paradigm as discussed above. Therefore, considering the distribution of the users and protection of privacy data, the COVID-19 close contact detecting problem should be solved in a decentralized way without relying on any central organization. Each user makes decisions in a distributed way and will not send any specific spatio-temporal data until finally choosing the best edge server, which improves time efficiency and mitigates privacy problem at the same time.

In this paper, an effective Game Theory-based Covid-19 Detecting Method, called GCDM is proposed. GCDM aims to improve the COVID-19 close contact detection efficiency and increase the utilization of the edge resources. The GCDM allows selfish users to compete for the edge resource sufficiently by leveraging exact potential game theory. It could maximize close contact detection completion rate while minimizing the latency and cost in a decentralized way. Three principal contributions of this paper are:

- Considering selfishness and rationality of each user, the COVID-19 close contact detection problem is formulated systematically with the edge computing collaboration. Through building a potential function, it is proved that the game is an exact potential game and can converge to a Nash Equilibrium within a finite number of iterations.
- To reach a Nash Equilibrium solution, a game theory-based COVID-19 close contact detection method, named GCDM, is proposed. GCDM consists of four steps, including establishing an exact potential game model, evaluating players, finding better decisions and deciding actions.
- To demonstrate the performance of the method, GCDM is evaluated against three baseline methods through extensive experiments and comparison analysis.

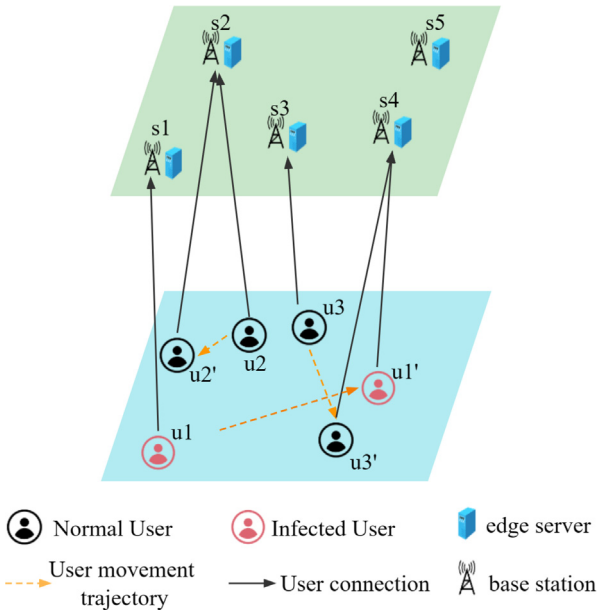

Fig. 1. A scenario of COVID-19 close contact detecting with edge computing collaboration.

The rest of this paper is organized as follows. Preliminary knowledge about exact potential game and edge computing environment is discussed in Section 2. The formulation of COVID-19 close contact detecting problem with edge computing collaboration is demonstrated in Section 3. Section 4 gives the details of the game theory-based COVID-19 close contact detection method. The experiments are presented and discussed in Section 5. Related work is reviewed in Section 6 and Section 7 gives a conclusion of the paper.

2. Preliminary knowledge

To facilitate the discussion of GCDM mechanism for COVID-19 close contact detection services, this section gives a brief introduction of the edge computing collaboration mode and the exact potential game theory.

2.1. Edge collaboration mode

There are X users denoted as $U = \{u_1, u_2, \ldots, u_X\}$. Each user has a COVID-19 close contact detection event to be processed, defined as $event_p = (s_p, z_p, lim_p, \gamma_{p,t}, \gamma_{p,c}, \theta_p)$, where s_p is the size of input data and z_p is the number of CPU cycles required to complete the detection event. $event_p$ should be completed before deadline lim_p, or it would fail. $\gamma_{p,t}$ and $\gamma_{p,c}$ are the weights of latency and cost when making offloading decisions, which satisfy $0 \le \gamma_{p,t}, \gamma_{p,c} \le 1$ and $\gamma_{p,t} + \gamma_{p,c} = 1$. These two parameters can represent each user's individual preference for event latency and event cost. Besides, we use $\theta_p \in \{1, \ldots, Y\}$ to represent the offloading decision. $\theta_p = q$ represents that $event_p$ is offloaded to the qth edge server to be executed.

There are Y edge servers denoted as $S = \{s_1, s_2, \ldots, s_Y\}$, each of which can be defined as $s_q = (g_q, h_q, u_q)$, where g_q, h_q and u_q represent communication capacity, computing capacity and unit computing price, respectively. Due to limited resources of edge servers and great demands from users' detection events, we assume that each edge server equally allocates communication and computing resources to all events offloaded to it. The more events choosing the same edge server, the less resources each can get.

A typical scenario is shown in Fig. 1. To ensure the normal operation of users' mobile phones, the phones need to frequently connect to nearby base stations via wireless channels, which generates BS connection data stored in phones and can reflect users' spatio-temporal

information. When it comes to COVID-19 close contact detection, the BS connection data can serve as input. Through judging whether one user's spatio-temporal trajectory intersected with those of infected people, we can evaluate the risk of injection and complete the close contact detection event. Besides, to efficiently complete the detection event, each user can select an appropriate edge server with low latency and cost, and then offload the detection event to it for execution and get evaluation result of infection risk.

2.2. Exact potential game

Game theory is a multidisciplinary concept designed to describe the interaction of players in a multi-decision setting with rational behaviors [19]. Cooperative and non-cooperative games are two different types of games. The following are descriptions of a non-cooperative game and Nash Equilibrium.

A non-cooperative game consists of three elements. It can be formulated as a triple $G = (\mathcal{N}, \{\mathcal{A}_p\}_{p\in\mathcal{N}}, \{\mathcal{C}_p\}_{p\in\mathcal{N}})$, where $\mathcal{N}$ is the set of players, $\mathcal{A}_p$ represents the decision space of player p, and $\mathcal{C}_p$ is the payoff function that player p aims to minimize [20].

For the Nash Equilibrium, given $\theta_{-p} = \{\theta_1, \ldots, \theta_{p-1}, \theta_{p+1}, \ldots, \theta_X\}$ as strategies of all other players except player p, a strategy profile $\theta^* = \{\theta_1^*, \ldots, \theta_X^*\}$ is a Nash Equilibrium of game G if no player can further reduce its payoff function by changing its strategy unilaterally [21]. When a game reaches a Nash Equilibrium, all players can obtain the most advantageous decision for themselves and will not actively deviate from the decision.

$$\mathcal{C}_p(\theta_p^*, \theta_{-p}^*) \le \mathcal{C}_p(\theta_p, \theta_{-p}^*), \forall p \in \mathcal{N}, \forall \theta_p \in \mathcal{A}_p \tag{1}$$

As a subset of non-cooperative games, exact potential games possess good convergence features. According to the finite improvement property, any exact potential game can reach a pure-strategy Nash Equilibrium after a finite number of iterations (FIP). A game can be called an exact potential game if we can find a potential function $\phi(\theta_p, \theta_{-p})$ that satisfies the following equation for $\forall p \in \mathcal{N}$, $\forall \theta_p, \theta_p' \in \mathcal{A}_p$.

$$\phi(\theta_p', \theta_{-p}) - \phi(\theta_p, \theta_{-p}) = \mathcal{C}_p(\theta_p', \theta_{-p}) - \mathcal{C}_p(\theta_p, \theta_{-p}) \tag{2}$$

Besides, PoA (price of anarchy) is a game theory concept that is frequently used to assess the effectiveness of a Nash Equilibrium solution. The efficiency ratio of the worst case Nash Equilibrium strategy $\hat{\theta}$ over the centralized optimum strategy θ^* is used to quantify PoA.

$$PoA = \frac{\sum_{p=1}^{N} \mathcal{C}_p(\hat{\theta})}{\sum_{p=1}^{N} \mathcal{C}_p(\theta^*)} \tag{3}$$

3. Formulation of the edge computing collaborative COVID-19 close contact detecting model

We begin this part by defining event latency, event cost, event latency-cost, and event completion rate. We define the edge collaborative event offloading problem as a multi-objective optimization problem on this basis.

Definition 1 (EL_p)**.** For $event_p$, event latency (EL_p) refers to the time to complete the detection event, which consists of data transmission latency and execution latency. EL_p can be calculated as follows, where g and h represent the communication and computation resources acquired by the event, respectively.

$$\begin{aligned} EL_p &= EL_p^{trans} + EL_p^{exec} \\ &= \frac{s_p}{g} + \frac{z_p}{h} \end{aligned} \tag{4}$$

Definition 2 (EC_p)**.** For $event_p$, event cost (EC_p) refers to the cost paid for computing resources when executing the event, which can be calculated as the product of unit price and number of CPU cycles.

$$EC_p = z_p u \tag{5}$$

Definition 3 (ELC_p)**.** For $event_p$, event latency-cost (ELC_p) refers to the weighted sum of event latency and event cost. When making offloading decisions, each user attempts to minimize its own event latency and event cost on the premise of meeting the deadline restriction.

$$ELC_p = \gamma_{p,t} EL_p + \gamma_{p,c} EC_p \tag{6}$$

Definition 4 (ECR)**.** Given the set of all X detection events in the system, event completion rate (ECR) is defined as the proportion of the number of events completed before its deadline to the number of all events X. $I()$ is a symbolic function. If $EL_p \le lim_p$, then $I(EL_p \le lim_p) = 1$, otherwise $I(EL_p \le lim_p) = 0$.

$$ECR = \frac{\sum_{p=1}^{X} I(EL_p \le lim_p)}{X} \tag{7}$$

To calculate event latency-cost of each user, we take the pth user u_p as an example. If u_p decides to offload $event_p$ to the qth edge server s_q, we should firstly count the number of events choosing the same edge server:

$$num_q = \sum_{x=1}^{X} I(\theta_x = q) \tag{8}$$

The qth edge server distributes its communication and computing resources to num_q events equally, so resources allocated to $event_p$ can be calculated:

$$g_{p,q} = \frac{g_q}{num_q}, h_{p,q} = \frac{h_q}{num_q} \tag{9}$$

Therefore, event latency-cost of u_p equals to:

$$\begin{aligned} ELC_p = ELC_{p,q} &= \gamma_{p,t}\left(\frac{s_p}{g_{p,q}} + \frac{z_p}{h_{p,q}}\right) + \gamma_{p,c} z_p u_q \\ &= \gamma_{p,t}\left(\frac{s_p}{g_q} + \frac{z_p}{h_q}\right) \sum_{x=1}^{X} I(\theta_x = q) + \gamma_{p,c} z_p u_q \end{aligned} \tag{10}$$

From the rational perspective of each individual user, u_p intends to make the event offloading decision which is the most beneficial to itself. "Most beneficial" means minimizing its event latency-cost (ELC_p) on the premise of meeting the deadline constraint.

From the overall perspective of the whole system, due to the limited real-time resources, not all events can be completed before deadline. While optimizing latency-cost of each user, it is necessary to maximize the event completion rate (ECR) of the system through restricting the rationality of each user, so that more users' COVID-19 close contact detection events can be completed successfully in time.

According to above two perspectives from each individual user and the whole system, the optimization problem of COVID-19 close contact detection event offloading has two objectives: one is to minimize ELC of each user, and the other is to maximize ECR of the whole system. Therefore, the optimization problem is formulated as follows:

$$\begin{aligned} &\min_{\theta_p} \quad ELC_p, \forall p \in \{1, \ldots, X\} \\ &\max \quad ECR \end{aligned} \tag{11}$$

4. A game theory-based COVID-19 close contact detecting method with edge computing collaboration

This section covers the COVID-19 close contact detection method, which is based on game theory. In the beginning, we construct a flow chart to depict the overall strategy. Then the method's four specific phases and related algorithms are introduced in turn.

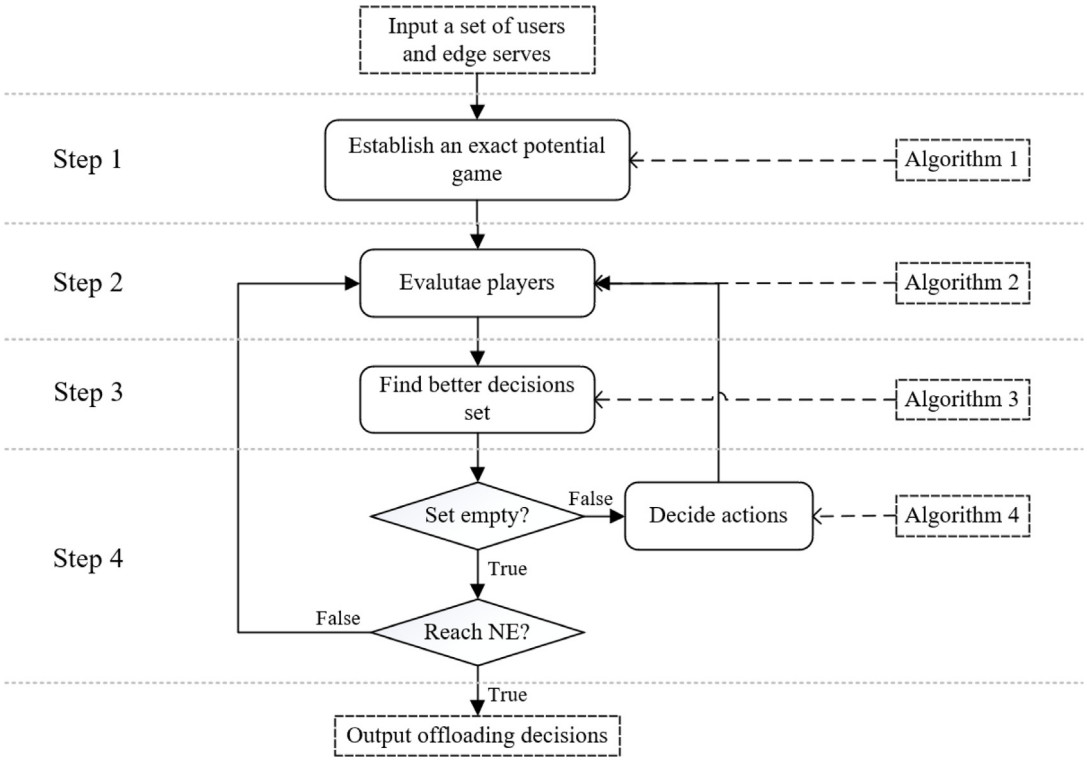

Fig. 2. Overview of game theory-based COVID-19 close contact detecting method.

4.1. Overview of the GCDM method

In order to achieve the first goal of the optimization problem, each user sensibly strives to reduce its own ELC. Game theory naturally has the benefit of taking into account each player's rational demands. The scope of game theory is very broad, and there exist a great number of different game theory models such as Stackelberg game, Stochastic game, Minority game and so on. Through the problem formulation and analysis, we can conclude that our problem belongs to the scope of non-cooperative game. Besides, among different non-cooperative game models, exact potential games shows exhibit convergence and finite improvement property, which means that the model can reach a pure-strategy Nash Equilibrium after a finite number of iterations. These good properties are beneficial for quickly getting a good solution of the problem. Therefore, the problem is initially represented as an exact potential game.

In order to achieve the second goal of the optimization problem, which is to maximize the system's ECR, the mutual influence of various users' actions is critical. To enhance ECR without breaking the convergence of the exact potential game, we must limit each user's rationality to some extent.

Fig. 2 shows the overview of game theory-based COVID-19 Close Contact detecting method, which consists of four steps:

- Step 1: Model the COVID-19 close contact detecting event offloading problem as an exact potential game.
- Step 2: Evaluate each user of the exact potential game model through estimating event urgency.
- Step 3: Find the set of offloading decisions better than current decisions for each user.
- Step 4: Decide actions at the end of each iteration for each user.

Apparently, the method can be realized in a centralized manner by relying on a central entity to evaluate event urgency and assign an order for each user by sorting them according to event urgency. However, a centralized approach frequently encounters issues such as single point of failure, privacy problems, and so on. Therefore, we implement the method shown in Fig. 2 in a decentralized way, where priorities among different users are evaluated by broadcasting messages instead of relying on a centralized entity. Overall process of the method is described as follows.

Algorithm 1 is firstly run to model the COVID-19 close contact detection event offloading problem as an exact potential game. Then the game is run in an iterative manner until it reaches a Nash Equilibrium and outputs final offloading decisions of all users. During each iteration, each player simultaneously tries to optimize its offloading decision by running algorithm 2, 3, 4 in sequence. Take the pth player u_p as an example, the player firstly evaluates its event urgency based on the global decisions of the last iteration with algorithm 2. Then, it adopts algorithm 3 to finds better decisions opt_p satisfying three conditions, and makes use of algorithm 4 to decide its action for current iteration.

4.2. Specification of the GCDM method

(1) Establish an Exact Potential Game Model

As the beginning step of GCDM, a game model based on each user's unique logic is created. Algorithm 1 shows the details.

According to the discussion of non-cooperative games, the game could be described as $G = (\mathcal{N}, \{\mathcal{A}_p\}_{p\in\mathcal{N}}, \{\mathcal{C}_p\}_{p\in\mathcal{N}})$. The set of X users participate in the game as players. For player u_p, decision space $\mathcal{A}_p$ is $\{1, \ldots, Y\}$, and ELC_p serves as the payoff function $\mathcal{C}_p$ which needs to be minimized. We will prove that this game model is an exact potential game when we analyze properties of the method in the next subsection.

(2) Evaluate Players of the Exact Potential Game Model

When running the exact potential game model, each rational player is exclusively concerned with its own interest. However, the system's limited real-time edge resources may not be able to meet the needs of all participants. Only taking into account the individual interests of the players may result in a low ECR and the inability of many users' COVID-19 close contact detection events to be completed on time, leading to the spread of the pandemic. Thus, as the second part of the method, players are evaluated through estimating the degree of event

Algorithm 1 Establish an exact potential game model

Input: attributes of users and edge servers
Output: an exact potential game model
1: Set players $\mathcal{N} = \{u_1, ..., u_X\}$.
2: **for** u_p in $\mathcal{N}$ **do**
3: Set decision space $\mathcal{A}_p = \{1, ..., Y\}$.
4: Set $\mathcal{C}_p = ELC_p$.
5: Establish a game model $G = (\mathcal{N}, \{\mathcal{A}_p\}_{p\in\mathcal{N}}, \{\mathcal{C}_p\}_{p\in\mathcal{N}})$.
6: **return** G

urgency. Higher-ranked players can choose real-time edge resources and change offloading decisions earlier. Algorithm 2 shows the detailed procedures.

Algorithm 2 Evaluate players of the exact potential game model

Input: G, attributes of u_p and edge servers
Output: event urgency of u_p
1: **initialize**: event urgency $urgency_p$ of user u_p.
2: Set total latency $total_p = 0$.
3: **for** q in $\{1, ..., Y\}$ **do**
4: Calculate latency $EL_{p,q}$ if u_p offloads the detection event to s_q.
5: $total_p = total_p + EL_{p,q}$.
6: Calculate average latency by $avg_p = total_p/Y$.
7: Evaluate event urgency by $urgency_p = lim_p/avg_p$.
8: **return** $urgency_p$

To evaluate urgency of users' detection event, we firstly try to calculate the sum and average of event latency under all possible offloading decisions by setting θ_p to every element in decision set $\{1, \dots, Y\}$.

$$total_p = \sum_{\theta_p\in\{1,...,Y\}} EL_{p,\theta_p} \tag{12}$$

$$avg_p = \frac{total_p}{Y} \tag{13}$$

Then, we define u_p's event urgency $urgency_p$ as the ratio of the deadline to the average event latency under all possible offloading decisions. A smaller $urgency_p$ indicates that $event_p$ is more urgent, and the decision space which can meet the deadline constraint is smaller, so u_p should better update its offloading decision earlier. Therefore, users are given priority according to event urgency for the subsequent running process.

$$urgency_p = \frac{lim_p}{avg_p} = \frac{lim_p}{\sum_{\theta_p\in\{1,...,Y\}} EL_p/Y} \tag{14}$$

(3) Find Better Decisions Set

As the third step of the approach, we try to find the set of event offloading decisions better than current decision for each game player. Detailed steps are shown in algorithm 3.

In order to determine that new offloading decision θ_p' is better than current offloading decision θ_p for u_p, three conditions should be met. On the one hand, from the rational perspective of each individual user, u_p wants to minimize ELC_p on the premise of meeting the deadline constraint, so condition 1 and 2 should be satisfied. On the other hand, from the overall perspective of the whole system, if new decision is offloading to s_q, then users with more urgent detection events than u_p that decide to offload events to the same edge server will get less resources, which leads to longer event latency and may decrease ECR of the system. Therefore, it is necessary to restrict the rationality of users to satisfy condition 3.

- condition 1: $event_p$ can meet deadline constraint with θ_p', i.e. $EL_p' \le lim_p$.

Algorithm 3 Find set of better decisions for each user

Input: G, attributes of u_p and edge servers
Output: set of better event offloading decisions for u_p
1: **initialize**: empty set of better event offloading decision opt_p.
2: **for** q in $\{1, ..., Y\}$ **do**
3: $flag = 1$.
4: Calculate event latency EL_p' when $\theta_p' = q$.
5: **if** $EL_p' > lim_p$ **then**
6: $flag = 0$, continue.
7: Calculate event latency-cost ELC_p' when $\theta_p' = q$.
8: **if** $ELC_p' \ge ELC_p$ **then**
9: $flag = 0$, continue.
10: **for** other u_k with more urgent event and decision $\theta_k = q$ **do**
11: **if** $EL_k' > lim_k$ **then**
12: $flag = 0$, break.
13: **if** $flag == 1$ **then**
14: Append q to opt_p.
15: **return** opt_p

- condition 2: Event latency-cost of u_p under θ_p' is smaller than that of θ_p, i.e. $ELC_p' < ELC_p$.
- condition 3: It should be ensured that users with more urgent detection events than u_p that decide to offload events to the same edge server can still meet deadline constraints.

(4) Decide the Action for Each User

Since at most one player is allowed to update its decision during each game iteration, as the fourth step of the method, each player has to decide whether it has the opportunity to change its event offloading decision at the end of each iteration and the judge mechanism should be carefully designed. Detailed steps are shown in algorithm 4.

During each iteration, if user u_p has not received any message from other players, then it will broadcast a message to notify other players that it wants to update its decision in this iteration. The message includes information about event urgency $urgency_p$ and new decision θ_p'. On the contrary, if u_p has received a message from another player u_k, then it will judge whether u_k's detection event is more urgent. If u_k has a smaller event urgency value, then u_p would keep silent and keep its decision unchanged. When it comes to the end of current iteration, u_p judges whether it has the opportunity to update its decision in current iteration. u_p can update offloading decision only if it has the smallest urgency value among all users who compete for the opportunity to update. Otherwise, u_p would maintain its original decision unchanged even though it has better decisions, because only at most one update is allowed in each iteration. If u_p does not receive any message from other players, and it does not need to update its offloading decision, we can conclude that a Nash Equilibrium has been reached and the algorithm terminates.

4.3. Theoretical analysis of the GCDM method

(1) Convergence of the Method

Theorem 1. *The game G built by algorithm 1 is an exact potential game with the following potential function:*

$$\begin{aligned}\phi(\theta_p, \theta_{-p}) = &\frac{1}{2}\sum_{x=1}^{X}\sum_{v\neq x}^{X}\sum_{y=1}^{Y}\gamma_{x,t}\left(\frac{s_x}{g_y} + \frac{z_x}{h_y}\right)I(\theta_x = y)I(\theta_v = y) \\ &+ \sum_{x=1}^{X}\sum_{y=1}^{Y} O_{x,y}I(\theta_x = y)\end{aligned} \tag{15}$$

Algorithm 4 Decide the action for each user

Input: G, attributes of u_p and edge servers, set of better decisions opt_p
Output: event offloading decision θ_p

```
if opt_p is not empty then
    Select decision θ_p' from opt_p with the minimum C_p.
    if u_p has received messages from u_k and urgency_k < urgency_p then
        u_p keeps silent.
    else
        u_p broadcasts urgency_p and θ_p'.
    if current iteration ends then
        if urgency_p is the smallest among all requesting users then
            u_p updates decision to θ_p'.
        else
            u_p maintains original decision θ_p.
if no message received in current iteration then
    One NE has been reached, break.
return θ_p
```

where

$$O_{x,y} = \gamma_{x,t}(\frac{s_x}{g_y} + \frac{z_x}{h_y}) + \gamma_{x,c} z_x u_y, y \in \{1, \dots, Y\} \tag{16}$$

Proof. According to description of the exact potential game, we should prove that the equation $\Delta\phi = \Delta\mathcal{C}_p$ holds for all possible cases when a player changes event offloading decision, in which $\Delta\phi = \phi(\theta_p', \theta_{-p}) - \phi(\theta_p, \theta_{-p})$, and $\Delta\mathcal{C}_p = \mathcal{C}_p(\theta_p', \theta_{-p}) - \mathcal{C}_p(\theta_p, \theta_{-p})$. All possible cases can be summarized as $\theta_p = q \in \{1, \dots, Y\}$, $\theta_p' = k \in \{1, \dots, Y\}$, where $q \neq k$.

Firstly, the original decision of player u_p is offloading the detection event to the qth edge server s_q, i.e. $\theta_p = q$. We can calculate the original value of potential function ϕ as follows.

$$\begin{aligned}\phi(\theta_p, \theta_{-p}) &= \phi(q, \theta_{-p}) \\ &= \frac{1}{2}\sum_{x\neq p}^{X}\sum_{\substack{v\neq x \\ v\neq p}}^{X}\sum_{y=1}^{Y}\gamma_{x,t}(\frac{x_x}{g_y} + \frac{z_x}{h_y})I(\theta_x = y)I(\theta_v = y) \\ &\quad + \frac{1}{2}\gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q) + \frac{1}{2}\gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q) \\ &\quad + \sum_{x\neq p}^{X}\sum_{y=1}^{Y} O_{x,y} I(\theta_x = y) + O_{p,q}\end{aligned} \tag{17}$$

Then, the player u_p updates its offloading decision, and decides to offload its detection event to another kth edge server s_k, i.e. $\theta_p{}' = k$. We can calculate the updated value of potential function ϕ as follows.

$$\begin{aligned}&\phi(\theta_p{}', \theta_{-p}) = \phi(k, \theta_{-p}) \\ &= \frac{1}{2}\sum_{x\neq p}^{X}\sum_{\substack{v\neq x \\ v\neq p}}^{X}\sum_{y=1}^{Y}\gamma_{x,t}(\frac{s_x}{g_y} + \frac{z_x}{h_y})I(\theta_x = y)I(\theta_v = y) \\ &\quad + \frac{1}{2}\gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) + \frac{1}{2}\gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) \\ &\quad + \sum_{x\neq p}^{X}\sum_{y=1}^{Y} O_{x,y} I(\theta_x = y) + O_{p,k}\end{aligned} \tag{18}$$

The change in potential function, denoted by $\Delta\phi$, can be determined using the original and updated values of the potential function.

$$\begin{aligned}&\Delta\phi = \phi(k, \theta_{-p}) - \phi(q, \theta_{-p}) \\ &= O_{p,k} + \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) - O_{p,q} - \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q)\end{aligned} \tag{19}$$

The change in the payoff function of player u_p, indicated by $\Delta\mathcal{C}_p$, should then be obtained. The original and updated values of u_p's payout function should be acquired in advance, similar to the calculation of $\Delta\phi$. The original payoff function value is presented below.

$$\begin{aligned}&\mathcal{C}_p(\theta_p, \theta_{-p}) = \mathcal{C}_p(q, \theta_{-p}) = ELC_{p,q} \\ &= \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x=1}^{X} I(\theta_x = q) + \gamma_{p,c} z_p u_q \\ &= \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q) + \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q}) + \gamma_{p,c} z_p u_q \\ &= \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q) + O_{p,q}\end{aligned} \tag{20}$$

Similarly, the updated value of u_p's payoff function when $\theta_p = k$ is equals to

$$\begin{aligned}&\mathcal{C}_p(\theta_p{}', \theta_{-p}) = \mathcal{C}_p(k, \theta_{-p}) = ELC_{p,k} \\ &= \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x=1}^{X} I(\theta_x = k) + \gamma_{p,c} z_p u_k \\ &= \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) + \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k}) + \gamma_{p,c} z_p u_k \\ &= \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) + O_{p,k}\end{aligned} \tag{21}$$

The change in payoff function can be determined using the original and updated values of the payoff function of palyer u_p.

$$\begin{aligned}&\Delta\mathcal{C}_p = \mathcal{C}_p(k, \theta_{-p}) - \mathcal{C}_p(q, \theta_{-p}) \\ &= O_{p,k} + \gamma_{p,t}(\frac{s_p}{g_k} + \frac{z_p}{h_k})\sum_{x\neq p}^{X} I(\theta_x = k) - O_{p,q} - \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q})\sum_{x\neq p}^{X} I(\theta_x = q)\end{aligned} \tag{22}$$

According to Eqs. (19) and (22), the equation $\Delta\phi = \Delta\mathcal{C}_p$ holds for all possible cases. As a result, G is an exact potential game, which possesses finite improvement property (FIP). Therefore, the game can attain a pure-strategy Nash Equilibrium within a finite number of iterations.

(2) PoA of the Method

Theorem 2. *PoA of the game G built by algorithm 1 satisfies:*

$$1 \leq PoA \leq \frac{\sum_{p=1}^{X} ELC_{p,q}^{max}}{\sum_{p=1}^{X} ELC_{p,q}^{min}} \tag{23}$$

Proof. When u_p offloads its detection event to the qth edge server, the communication and computing resources allocated to it satisfies:

$$\frac{g_q}{X} \leq g_{p,q} \leq g_q, \frac{h_q}{X} \leq h_{p,q} \leq h_q \tag{24}$$

The lower and upper bounds of latency-cost while offloading the event to the edge server could then be calculated.

$$ELC_p(\hat{\theta}) \leq ELC_{p,q}^{max} = X \times \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q}) + \gamma_{p,c} z_p u_q \tag{25}$$

$$ELC_p(\theta^*) \geq ELC_{p,q}^{min} = \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q}) + \gamma_{p,c} z_p u_q \tag{26}$$

Table 1
Parameters of edge servers and detection events.

Variable	Value range
Communication capacity (Mbps)	$(5, 10) \times 9.97$
Computing capacity (GHz)	$\{5, 6, 8, 9\}$
Unit price (per 1e6 cycles)	$\{0.8, 0.9, 1, 1.1, 1.2\}$
Input data size (KB)	(100,600)
Cycles required (cycles)	$(1, 6) \times 1e5$
Weight of latency	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$

Therefore, we can get

$$1 \leq PoA \leq \frac{\sum_{p=1}^{X} ELC_{p,q}^{max}}{\sum_{p=1}^{X} ELC_{p,q}^{min}} = \frac{\sum_{p=1}^{X}(X \times \gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q}) + \gamma_{p,c} z_p u_q)}{\sum_{p=1}^{X}(\gamma_{p,t}(\frac{s_p}{g_q} + \frac{z_p}{h_q}) + \gamma_{p,c} z_p u_q)} \tag{27}$$

(3) Time Complexity of the Method

The proposed method runs in an iterative manner. During each iteration, all users run algorithm 2, 3, 4 in sequence together in a distributed way. In algorithm 2, user u_p traverses all edge servers to evaluate event urgency with time complexity $O(Y)$. In algorithm 3, user u_p also need to traverse all edge servers to judge whether it can meet three conditions and whether it can be an optional choice. Among three conditions, the first and second condition can be finished in $O(1)$ time. However, when considering the third condition, u_p should visit other users with the same decision, the number of which is $O(X/Y)$ on average. Therefore, time complexity of algorithm 3 equals to $O(Y(1 + X/Y)) = O(X + Y)$. Algorithm 4 with only several judgments can be finished in $O(1)$ time. Therefore, time complexity of one iteration can be obtained from the sum of three algorithms, which equals to $O(1) + O(X + Y) + O(1) = O(X + Y)$. Besides, we denote the number of iterations to reach a Nash Equilibrium as C, which increases linearly with the number of users X. Then the total time complexity of the proposed method is $O(C(X + Y))$, which can be executed in a fast manner in practice.

5. Performance evaluation

5.1. Experiment settings

The proposed game theory-based COVID-19 close contact detecting method(GCDM) is implemented with python 3.9. Parameters of edge servers shown in Table 1 are set with reference to [22], which are claimed to be real-world values obtained from other work. The computing resources of each edge server is randomly picked from {5,6,8,9}GHz, and communication resources is $9.97R$ Mbps, where R is an integer randomly picked from $[5, 10]$. Besides, last three rows of Table 1 are parameters of detection events, where {} means selecting a random number from the set, and () indicates randomly generating an integer within the range.

To evaluate GCDM comprehensively, we simulate two scenarios by altering the values of the number of users and the number of edge servers, one for each time. As shown in Table 2, we fix the values of one parameter when the value of the other parameter varies. For the first experiment setting, we fix the number of edge servers as 25 and set the number of users as 200, 400, 600, 800, 1000 in turn to conduct experiments and comparison analysis. For the second experiment setting, we fix the number of users as 800 and set the number of edge servers as 10, 20, 30, 40, 50 to conduct experiments and comparison analysis.

In the experiments, we compare GCDM's performance to three baseline methods which have been proposed to solve task offloading problems to optimize latency or task completion rate.

- LFC [23]: Events are firstly evaluated and assigned priorities by evaluating deadlines. Then, each user chooses the optimal server to meet the deadline.
- DCO [24]: A standard potential game model is built and run to determine the offloading decisions without taking deadlines into account.
- Dedas [25]: This method adopts a heuristic idea. If multiple edge servers can finish the event before the corresponding deadline, then the one with the greatest difference between deadline and event latency is chosen. Besides, if all edge servers fail to meet its deadline constraint, then the one that causes the least number of successful events to fail will be selected.

Table 2
Experiment environment settings.

Set ID	Total number of edge servers	Total number of users
1	25	200, 400, 600, 800, 1000
2	10, 20, 30, 40, 50	800

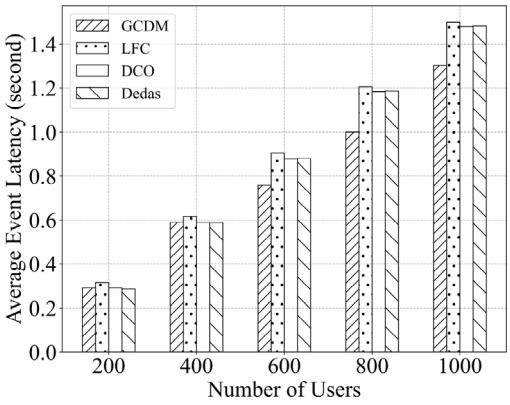

Fig. 3. Average EL via number of users.

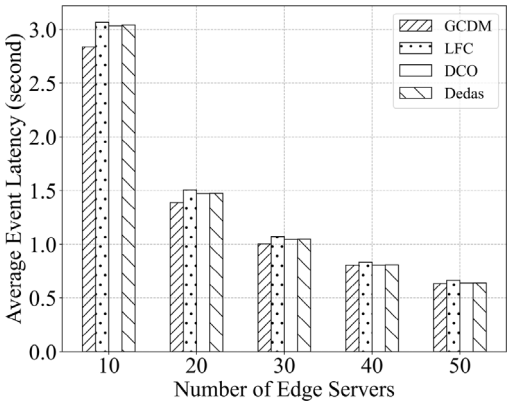

Fig. 4. Average EL via number of edge servers.

5.2. Comparison analysis

Based on the principles of control variable method shown in Table 2, we compare GCDM with aforementioned three methods, and we mainly discuss four metrics, including event latency, event cost, event latency-cost and event completion rate.

(1) Event Latency (EL)

"Average event latency" refers to the average latency to finish all users' COVID-19 close contact detection events. Results of two experiment sets about average event latency are shown in Fig. 3 and Fig. 4.

Fig. 3 demonstrates that the average event latency of all four techniques grows as the number of users increases, which is easy to anticipate and understand. More users indicates more detection events

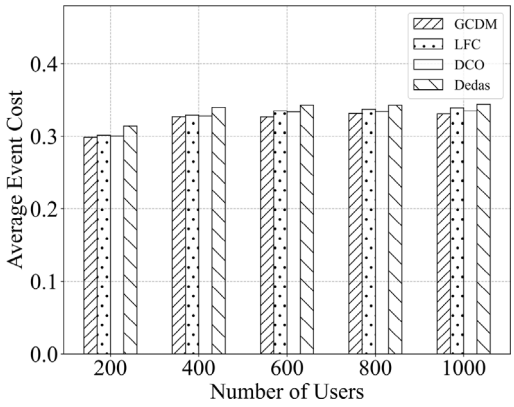

Fig. 5. Average EC via number of users.

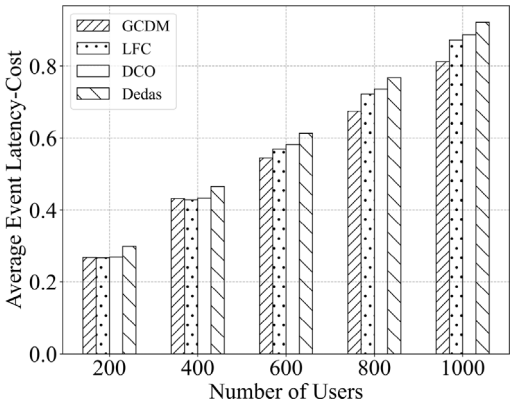

Fig. 7. Average ELC via number of users.

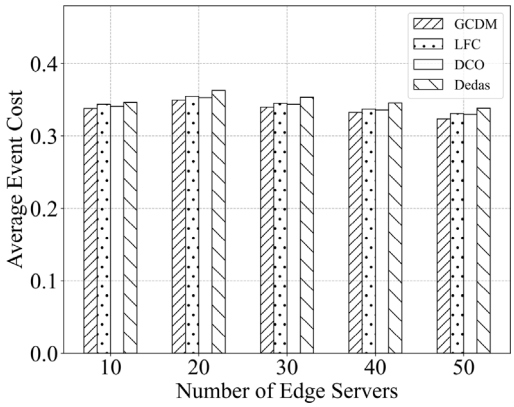

Fig. 6. Average EC via number of edge servers.

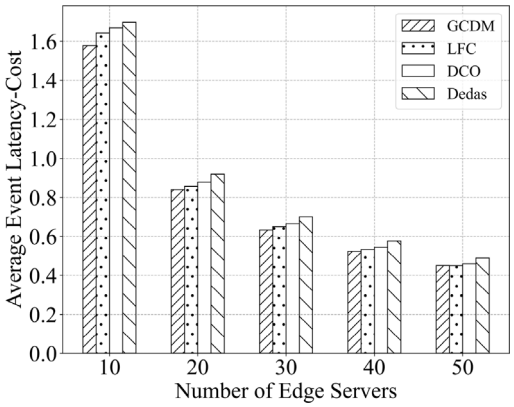

Fig. 8. Average ELC via number of edge servers.

to be offloaded. However, number of edge servers remains unchanged, so resources allocated to each user decreases and incurs longer latency. In addition, another conclusion can be found from the figure when we compare the performance of four methods. When the number of users is in a low level, average event latency of four methods are relatively close because system resources are rich enough to complete all detection events in time. However, with the number of users increasing, system resources become more and more limited, and our method GCDM gradually shows it advantage over other three techniques.

On the contrary, Fig. 4 shows that as the number of edge servers rises, average event latency of all four methods declines. More edge servers can provide more resources while number of users' detection events remains unchanged, so each event can be allocated more resources and be completed in a shorter time. Similar to the conclusion in Fig. 3, our method GCDM presents more obvious advantages over other three methods in face of edge resources shortage. Apart from these, another phenomenon in the figure is that when the number of edge servers increases from 10 to 20, EL of all four methods experience a sharp decline. After that, the downward trend gradually slows down.

(2) Event Cost (EC)

"Average event cost" refers to the average cost paid by all users in the system for consuming computing resources. The results of two experiment sets about average event cost are shown in Fig. 5 and Fig. 6.

We can find a slight increasing trend and a slight decreasing trend in these two figures separately, but the trends are not too obvious. Users are charged by edge servers in terms of CPU cycles used, which is only affected by fixed value of CPU cycles needed to finish the detection event. Besides, the difference between charging standard of different edge servers may not be two large. Therefore, average event cost only experiences a slight fluctuation when the number of users or edge servers changes. Furthermore, compared with LFC and Dedas, our method GCDM and the other game theory-based method DCO achieves relatively low EC in that cost is considered as an important factor when building utility functions.

(3) Event Latency-Cost (ELC)

"Average event latency-cost" refers to the average weighted sum of latency and cost of all users' COVID-19 close contact detection events. Results of two experiment sets about average latency-cost are shown in Figs. 7 and 8.

These two figures show similar ascending and descending trends to Figs. 3 and 4 about average EL, because ELC is the weighted sum of EL and EC, and the fluctuation of average EC is slight. With the number of users decreasing or the number of edge servers increasing, each event can be allocated more resources with similar unit price, which incurs much shorter event latency and relatively smaller event cost, resulting in smaller event latency-cost.

Through comparison, it can be found that GCDM has obvious advantage over the other three methods in face of system resource shortage when the number of users is large or the number of edge servers is small. Besides, the four methods depict the same order in terms of ELC for two different experiment settings, which indirectly verify the effectiveness of the results of our experiments. Through calculation, we can find that GCDM decreases average event latency-cost by 2.52%, 3.45% and 6.68% respectively compared with LFC, DCO and Dedas for the first experiment set and decreases average event latency-cost by 2.13%, 3.84% and 7.18% respectively for the second experiment set.

(4) Event Completion Rate (ECR)

"Event completion rate" refers to the proportion of detection events that can be completed before the given deadline to all of the events in the system. Improving event completion rate is one of the optimization

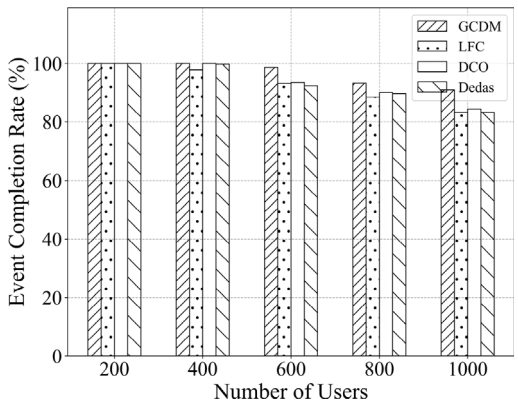

Fig. 9. ECR via number of users.

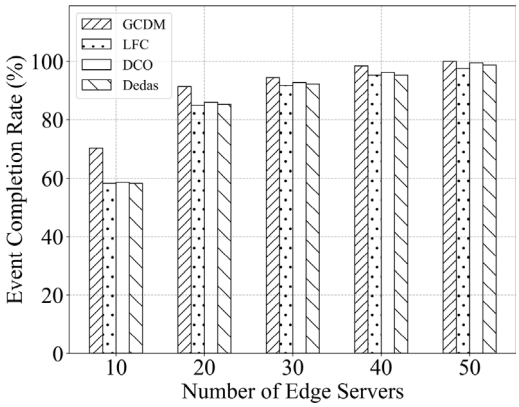

Fig. 10. ECR via number of edge servers.

goal of our method. Obviously, methods with higher ECR are better. Results of two experiment sets in terms of event completion rate are shown in Figs. 9 and 10, in which the trends are on the contrary to those in Figs. 7 and 8, which is very reasonable, because more resources means shorter latency but higher possibility to be finished in time.

In Fig. 9, when the number of users is on a low level, edge resources in the system is sufficient to finish all detection events before deadlines, so event completion rate of all four methods can reach 100%. After that, with more users existing in the system, each event can be allocated less resources, which may not meet the need to be finished in time. Therefore, event completion rate of all four methods gradually decline. Besides, our method GCDM shows more obvious advantage over other three methods in face of system resources shortage and similar conclusions can also be obtained from Fig. 10. Additionally, an obvious increase can be found in Fig. 10 when number of edge servers increases from 10 to 20, followed by a more moderate trend.

It can also be found that the four methods follow the same order in terms of ECR for two different experiment settings, which indirectly verify the effectiveness of the results of our experiments. Through calculation, we can find that GCDM increases event completion rate by 4.02%, 2.98% and 3.56% respectively compared with LFC, DCO and Dedas for the first experiment set and increases event completion rate by 5.33%, 4.3% and 4.95% respectively for the second experiment set.

6. Related work

The rapid spread of disease in most nations throughout the world has had a significant impact on society, the economy, and the healthcare system and many works have been done to tackle these problems [26]. For COVID-19, a lot of works have also been carried out to deal with the threat of the COVID-19 from different kinds of perspectives. Wang et al. [27] proposed an rapid diagnosis method of COVID-19. They use the deep learning-based model to implement the automatic diagnosis of COVID-19 which is helpful to counter the outbreak of SARS-CoV-2. Without the necessity for annotating the lesions for training, the technique can reliably estimate the COVID-19 infection probability and detect lesion locations in chest CT. The deep learning method is simple to train and performs well in identifying COVID-19 patients. Oh et al. [28] proposed a patch-based convolution neural network method. The method can solve the problem of the difficulty of collecting CXR data set for deep neural network training which is inspired by the CXR radiographs' potential imaging biomarkers. Roy et al. [29] developed an deep learning based assisted diagnosis method of lung diseases. The method relies on uninorms for effective video-level frame score aggregation. They also offer a fresh fully-annotated dataset of LUS pictures gathered from different Italian hospitals, as well as a deep network built using Spatial Transformer Networks.

Long latency between mobile devices and cloud computing infrastructures makes it difficult to fulfill the latency-sensitive tasks' requirements. Edge computing has emerged as a new computing paradigm aimed at addressing the problem. It has been widely used to address a variety of issues in the edge computing environment. For example, Zhou et al. [30] introduced the A-YONet deep neural network model for tracking multitarget detection in smart IoT devices in an edge computing environment. A-YONet is built by merging the benefits of YOLO and MTCNN. They also created a detection technique based on an anchor box preadjusting scheme and a multilayer feature fusion mechanism. Wang et al. [31] proposed Div_PreAPI, a MF-based recommendation method to improve the diversity of the recommendation. To provide diverse and tailored API suggestions, Div_PreAPI incorporates a weighting technique and neighborhood information into matrix factorization (MF). Based on locality-sensitive hashing, Kong et al. [32]proposed a unique multitype health data privacy-aware prediction approach. The technique can forecast missing health data while avoiding the problems of privacy disclosure and diverse data kinds. It can also achieve a decent balance between prediction accuracy and privacy protection.

Users are usually geographically distributed and the method used under the wireless distributed scenario is required to be decentralized. There exist many classic techniques which can fulfill the requirements like block chain [33–35] and game theory [36]. As a powerful tool to evaluate the interaction between multiple different players according to their individual interest and rationality, game theory has been successfully applied in many fields [37]. Refs. [22,24,38] all adopt potential game theory to determine an offloading decision for each device to jointly minimize task delay and energy consumption, in which [24] adopts cloud computing model, [38] uses edge computing model, and [22] adopts end–edge–cloud collaboration mode. These works only take into account each device's individual rationality, ignoring the mutual effect between the task completion of different devices, which leads to many task failures and a low task completion rate when system resources are not sufficient.

To the best of our knowledge, few work focus on the close contact detection problem for COVID-19 through the view point of edge computing. Inspired by the above studies, a game theory-based COVID-19 close contact detecting method, named GCDM, is developed. The GCDM can provide cost-effective strategies for the COVID-19 close contact detection problem with a high close contact detection completion rate.

7. Conclusion

Using the location data of the mobile phones is a promising way to detect the close contact patient of COVID-19. However, too much patient privacy data uploaded to the cloud center will overwhelm the network and the privacy of user information cannot be guaranteed. To tackle these problems, we propose a game theory-based COVID-19 close contact detecting method with edge computing collaboration called

GCDM. GCDM adopts exact potential game theory with edge computing paradigm, which can maximize close contact detection completion rate while minimizing the cost of the evaluation process in a decentralized way. Extensive experiments were carried out to compare the GCDM with three other methodologies, demonstrating that the GCDM works effectively.

CRediT authorship contribution statement

Yue Shen: Made a key contribution to the revision of the article, Designed the study, Originally drafted the manuscript. **Bowen Liu:** Reviewed the manuscript, Designed the study, Originally drafted the manuscript. **Xiaoyu Xia:** Processed the data, Designed the experiment. **Lianyong Qi:** Gave theoretical proof in the manuscript. **Xiaolong Xu:** Did the experiment, Visualized the experimental result. **Wanchun Dou:** Designed the study, Reviewed and edited the manuscript.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, "A Game Theory-Based COVID-19 Close Contact Detecting Method with Edge Computing Collaboration".

Data availability

No data was used for the research described in the article.

Acknowledgment

This work is supported by the Open Research Fund from Guangdong Laboratory ofArtificial Intelligence Digital Economy (SZ) No. GML-KF-22-20 and Dou Wanchun Expert Workstation of Yunnan Province No. 202105AF150013.

References

[1] A. Castiglione, P. Vijayakumar, M. Nappi, S. Sadiq, M. Umer, COVID-19: automatic detection of the novel coronavirus disease from CT images using an optimized convolutional neural network, IEEE Trans. Ind. Inform. 17 (9) (2021) 6480–6488.

[2] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He, W. Dou, DisCOV: distributed COVID-19 detection on X-ray images with edge-cloud collaboration, IEEE Trans. Serv. Comput. 15 (3) (2022).

[3] S. Kumar, R. Viral, Effect, challenges, and forecasting of COVID-19 situation in India using an ARMA model, IEEE Trans. Comput. Soc. Syst. 8 (4) (2021) 955–963.

[4] J. Ma, Y. Dong, Z. Huang, D. Mietchen, J. Li, Assessing the causal impact of COVID-19 related policies on outbreak dynamics: A case study in the US, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 2678–2686.

[5] https://covid19.who.int/.

[6] G. Wang, X. Liu, C. Li, Z. Xu, J. Ruan, H. Zhu, T. Meng, K. Li, N. Huang, S. Zhang, A noise-robust framework for automatic segmentation of COVID-19 pneumonia lesions from CT images, IEEE Trans. Med. Imaging 39 (8) (2020) 2653–2663.

[7] D. Antweiler, D. Sessler, M. Rossknecht, B. Abb, S. Ginzel, J. Kohlhammer, Uncovering chains of infections through spatio-temporal and visual analysis of COVID-19 contact traces, Comput. Graph. (2022).

[8] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu, J. Chen, A correlation graph based approach for personalized and compatible web APIs recommendation in mobile APP development, IEEE Trans. Knowl. Data Eng. (2022) http://dx.doi.org/10.1109/TKDE.2022.3168611.

[9] S. Tuli, S. Tuli, R. Tuli, S.S. Gill, Predicting the growth and trend of COVID-19 pandemic using machine learning and cloud computing, Internet Things 11 (2020) 100222.

[10] X. Xia, F. Chen, Q. He, G. Cui, J.C. Grundy, M. Abdelrazek, X. Xu, H. Jin, Data, user and power allocations for caching in multi-access edge computing, IEEE Trans. Parallel Distrib. Syst. 33 (5) (2022) 1144–1155.

[11] B. Liu, X. Jiang, X. He, L. Qi, X. Xu, X. Wang, W. Dou, A deep learning-based edge caching optimization method for cost-driven planning process over IIoT, J. Parallel Distrib. Comput. 168 (2022) 80–89.

[12] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, H. Jin, Online collaborative data caching in edge computing, IEEE Trans. Parallel Distrib. Syst. 32 (2) (2021) 281–294.

[13] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, Q. He, Constrained app data caching over edge server graphs in edge computing environment, IEEE Trans. Serv. Comput. 15 (5) (2022) 2635–2647.

[14] R. Schlegel, S. Kumar, E. Rosnes, A.G. i Amat, Privacy-preserving coded mobile edge computing for low-latency distributed inference, IEEE J. Sel. Areas Commun. 40 (3) (2022) 788–799.

[15] X. Jiang, Y. Sun, B. Liu, W. Dou, Combinatorial double auction for resource allocation with differential privacy in edge computing, Comput. Commun. 185 (2022) 13–22.

[16] B. Liu, S. Meng, X. Jiang, X. Xu, L. Qi, W. Dou, A QoS-guaranteed online user data deployment method in edge cloud computing environment, J. Syst. Archit. 118 (2021) 102185.

[17] X. Zhou, X. Yang, J. Ma, I. Kevin, K. Wang, Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT, IEEE Internet Things J. 9 (16) (2021) 14988–14997.

[18] H. Cho, Y. Cui, J. Lee, Energy-efficient cooperative offloading for edge computing-enabled vehicular networks, IEEE Trans. Wireless Commun. (2022).

[19] X. Chen, H. Zhao, H. Sun, S. Zhen, A. Al Mamun, Optimal adaptive robust control based on cooperative game theory for a class of fuzzy underactuated mechanical systems, IEEE Trans. Cybern. 52 (5) (2020) 3632–3644.

[20] L. Yao, Z. Chen, H. Dai, G. Wu, Exploiting non-cooperative game against cache pollution attack in vehicular content centric network, IEEE Trans. Dependable Secure Comput. 19 (6) (2021) 3873–3886.

[21] X. Nian, F. Niu, Z. Yang, Distributed Nash equilibrium seeking for multicluster game under switching communication topologies, IEEE Trans. Syst. Man Cybern. 53 (7) (2021) 4105–4116.

[22] Y. Ding, K. Li, C. Liu, K. Li, A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing, IEEE Trans. Parallel Distrib. Syst. 33 (6) (2021) 1503–1519.

[23] K. Fizza, N. Auluck, A. Azim, Improving the schedulability of real-time tasks using fog computing, IEEE Trans. Serv. Comput. 15 (1) (2019) 372–385.

[24] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2015) 2795–2808.

[25] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, B. Li, Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 2287–2295.

[26] X. Zhou, Y. Li, W. Liang, CNN-RNN based intelligent recommendation for online medical pre-diagnosis support, IEEE/ACM Trans. Comput. Biol. Bioinform. 18 (3) (2020) 912–921.

[27] X. Wang, X. Deng, Q. Fu, Q. Zhou, J. Feng, H. Ma, W. Liu, C. Zheng, A weakly-supervised framework for COVID-19 classification and lesion localization from chest CT, IEEE Trans. Med. Imaging 39 (8) (2020) 2615–2625.

[28] Y. Oh, S. Park, J.C. Ye, Deep learning COVID-19 features on CXR using limited training data sets, IEEE Trans. Med. Imaging 39 (8) (2020) 2688–2700.

[29] S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, C. Saltori, I. Huijben, N. Chennakeshava, F. Mento, A. Sentelli, et al., Deep learning for classification and localization of COVID-19 markers in point-of-care lung ultrasound, IEEE Trans. Med. Imaging 39 (8) (2020) 2676–2687.

[30] X. Zhou, X. Xu, W. Liang, Z. Zeng, Z. Yan, Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart IoT, IEEE Internet Things J. 8 (16) (2021) 12588–12596.

[31] F. Wang, L. Wang, G. Li, Y. Wang, C. Lv, L. Qi, Edge-cloud-enabled matrix factorization for diversified apis recommendation in mashup creation, World Wide Web 25 (5) (2022) 1809–1829.

[32] L. Kong, L. Wang, W. Gong, C. Yan, Y. Duan, L. Qi, LSH-aware multitype health data prediction with privacy preservation in edge environment, World Wide Web 25 (5) (2022) 1793–1808.

[33] W. Wang, H. Xu, M. Alazab, T.R. Gadekallu, Z. Han, C. Su, Blockchain-based reliable and efficient certificateless signature for IIoT devices, IEEE Trans. Ind. Inform. 18 (10) (2022) 7059–7067.

[34] Z. Lian, Q. Zeng, W. Wang, T.R. Gadekallu, C. Su, Blockchain-based two-stage federated learning with non-IID data in IoMT system, IEEE Trans. Comput. Soc. Syst. (2022) 1–10.

[35] W. Wang, Q. Chen, Z. Yin, G. Srivastava, T.R. Gadekallu, F. Alsolami, C. Su, Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks, IEEE Internet Things J. 9 (11) (2022) 8883–8891.

[36] Y. Yang, W. Wang, Z. Yin, R. Xu, X. Zhou, N. Kumar, M. Alazab, T.R. Gadekallu, Mixed game-based AoI optimization for combating COVID-19 with AI bots, IEEE J. Sel. Areas Commun. 40 (11) (2022) 3122–3138.

[37] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, Y. Yang, A game-theoretical approach for user allocation in edge computing environment, IEEE Trans. Parallel Distrib. Syst. 31 (3) (2019) 515–529.

[38] T. Fang, F. Yuan, L. Ao, J. Chen, Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach, IEEE Internet Things J. 9 (5) (2021) 3226–3237.