Journal of Ambient Intelligence and Humanized Computing (2023) 14:14333–14347
https://doi.org/10.1007/s12652-023-04672-z

ORIGINAL RESEARCH

Offloading approach for mobile edge computing based on chaotic quantum particle swarm optimization strategy

DeGan Zhang[1] · GuiXiang Sun[1] · Jie Zhang[3] · Ting Zhang[2] · Peng Yang[1]

Received: 4 September 2022 / Accepted: 26 July 2023 / Published online: 7 August 2023

Abstract
With the development of 5 g, computing-intensive and complex applications in smart-city is growing rapidly. Due to the limited resources of mobile terminal devices in smart-city, new applications have higher requirements for delay, bandwidth, security, and energy consumption. Computation offloading in mobile edge computing (MEC) is effective to reduce delay and energy consumption of Real-time video analysis. An improved chaos quantum-behaved particle swarm optimization (ICQPSO) algorithm is proposed for multi-user and multi-MEC edge Computation offloading scenarios. Compared with other heuristic algorithms, the improved chaos quantum-behaved particle swarm optimization algorithm can effectively reduce the delay and energy consumption of edge computing offloading. Experimental results show that the improved chaotic quantum-behaved particle swarm optimization (ICQPSO) can effectively avoid premature convergence, has stronger global searchability, and can solve multi-dimensional complex NP-hard problems more efficiently.

Keywords Mobile edge computing · Computation offloading · Chaotic quantum particle swarm optimization · Real-time video analysis

1 Introduction

Over the past 50 years, the world's urbanization rate has increased rapidly (Ritchie and Roser 2018). In the World Urbanization Prospects 2018 Report, the United Nations estimated that 68 percent of the world's population will be living in cities by 2050. Rapid urbanization brings convenience to people and improves people's lives, but also brings many challenges to an urban infrastructure network, including crowd and traffic governance and management, and urban public security maintenance (Hassanein et al. 2019; Khan et al. 2020; Moorthy et al. 2020). Therefore, in recent years, smart cities have gradually become a new trend of urbanization in all countries in the world (Achmad et al. 2018). With the rapid urbanization, all kinds of criminal cases are at a high rate, which requires the public security organs to react, make decisions and deal with them quickly. To solve the social security problems caused by the population explosion in cities and improve the efficiency of social comprehensive management, urban management through video surveillance has become an important means (Karaduman et al. 2018; Hidayat 2020; Bailas et al. 2018). Real-time video analysis in smart cities is a key function in public safety applications (Zhang et al. 2020). Real-time video analysis functions include detection and classification. In traditional real-time video analysis, the camera sensors capture image data, the captured image data through wired or wireless transmission to the backend server for storage and computation analysis in some scenarios can also transfer to the central cloud server, using the cloud server the huge amounts of storage space and a strong workforce calculation and analysis for video images. However, this approach also brings many problems. First of all, the amount of video image data captured by the camera sensor is very large, which will cause huge network load pressure to the transmission network. At the same time, in real-time video analysis, not all video image data need to be calculated and analyzed. If the data can be preprocessed before the transmission of video image data so that only useful data can be

✉ GuiXiang Sun
2415181375@qq.com

[1] Tianjin Key Lab of Intelligent Computing and Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China

[2] School of Sports Economics and Management, Tianjin University of Sport, Tianjin 301617, China

[3] School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing 100093, China

transmitted, the efficiency of real-time video analysis will be greatly improved. Secondly, the mode of transmitting the data captured by the camera sensor to the back-end server for processing will produce a great delay, which is not in line with the requirements of quick response, quick decision and quick processing in the smart city. Mobile edge computing technology comes into being to solve this kind of problem. In 2014, the European Telecommunication Association defined Mobile Edge Computing (MEC) as providing an IT service environment and cloud computing capability at the edge of mobile network, emphasizing proximity to mobile users, to reduce network operation and service delay and improve user experience. MEC is characterized by proximity, low delay, locality, location awareness, and other characteristics (Wang et al. 2017; Hassan et al. 2019; Liu et al. 2017). At the same time, ultra-reliable low delay communication (URLLC) is also one of the core demands of 5 G network (Liu 2017). As a complement to traditional cloud computing, MEC is considered a key technology in 5 G heterogeneous networks (Cao et al. 2019). Computing offloading is a key technology in MEC. Combined with the characteristics of edge computing with low delay, edge servers are deployed on the edge of wireless network and user tasks are offloaded to a reasonable location for computing to provide services such as low delay and high bandwidth to reduce network load (Zhang and Zhao 2020; Dubey and Meena 2020; Yy 2020). In the real-time video analysis scenario of mobile edge computing (MEC), the video data can be processed locally on the capturing device as well as the edge server. Due to the local workforce of the energy storage equipment such as the limitation of resources, therefore can only calculate simple tasks in the local device. The advantage of local equipment offloading is low latency,and reduce network load, but also has insufficient computing capacity, limited storage space, need the disadvantage of artificial replacement battery energy running out. The edge servers have more storage space and computing power. Energy is provided by external continued, so is suitable for calculating larger and more complex tasks. Edge server has the advantages of strong computing ability, large storage space, and unlimited energy, but it brings high time delay and high network load. Therefore, how to select the appropriate offloading location for the offloading task to minimize the total time delay or energy consumption has become a hot research topic for scholars at home and abroad. In this paper, real-time video analysis in public places is used as an application scenario. To co-optimize, the time delay and energy consumption in the model, an improved chaotic quantum particle swarm optimization algorithm is proposed to find the appropriate offloading position for each user. The main contributions of this paper are as follows. 1) Real-time video analysis was modeled as a multi-user multi-MEC model, and collaborative optimization was carried out for the time delay and energy consumption of the model. 2) An improved algorithm based on chaotic quantum particle swarm optimization (QPSO) is proposed to solve the computational offloading and resource allocation problems. 3) The edge calculation offloading in a complex environment is simulated, and the simulation results are analyzed. The rest of the paper is arranged as follows: The second section introduces the research status of computational offloading at home and abroad; The third section introduces real-time video analysis scene modeling; The fourth section introduces the chaos quantum particle swarm optimization algorithm; The fifth section introduces the unloading strategy of edge computing based on chaotic quantum particle swarm; The sixth section shows the analysis of the experimental results; The seventh section is an overview of the paper.

2 Related Works

Computational offloading of MEC is the focus of research by scholars (Piao and Zhang 2020). According to different application scenarios, the goal of MEC calculation offload is different (Lin et al. 2019). Internet of vehicles has always been an important application scenario of edge computing (Zhang 2021; Ge 2019; Gundu et al. 2022; Gundu 2021). Literature (Wang et al. 2020; Gundu et al. 2021; Srinivasa Rao and Charan Arur 2022) proposed a distributed optimal response algorithm based on game theory to maximize the utility of each vehicle in the scenario of a multi-user single MEC server on the Internet of Vehicles. Literature (Zhao et al. 2019; Gundu and Anuradha 2020, 2019) studied the computing offloading strategy on the Internet of Vehicles. In the cloud-MEC collaborative computing offloading scenario, an optimization algorithm for writing computing offloading and resource allocation was proposed. According to the literature (Zhang et al. 2019; Gundu and Anuradha 2020; Gundu et al. 2020; Wang et al. 2023; Cui and Zhang 2019; Ni and Zhang 2022), vehicles with limited resources put forward intensive computing offloading requests to edge servers in the scenario of the Internet of Vehicles, and the state changes of multiple edge servers and different vehicle offloading modes make effective task offloading a huge challenge. For this reason, an effective redundant offloading algorithm is proposed to improve the offloading reliability in the case of vehicle data transmission failure.

Time delay and energy consumption are important indexes in edge calculation offloading decisions (Chen and Zhang 2020; Gundu and Panem 2022; Chen 2022; Cao 2022). Literature (Yang et al. 2019; Wang and Zhang 2022; Wang et al. 2023) proposed a reinforcement learning method based on Q-learning to define the system model, meet the time delay constraint and minimize the energy consumption. Literature (Huynh et al. 2020; Chen 2023; Dong and Zhang 2022) uses an alternative meta-heuristic algorithm of whale optimization algorithm (WOA) to minimize delay and energy consumption. Literature

(Wu et al. 2016; Ni and Zhang 2023; Wang and Song 2014) used a nonlinear exponential inertial weight particle swarm optimization algorithm to optimize the time delay and energy consumption in the offloading model of edge cloud collaborative multitask computing. Literature (Hu et al. 2019) adopts the Lyapunov optimization algorithm to dynamically adjust the offloading decision of the task according to the fluctuation of the current task data, to reduce the energy consumption under the condition of satisfying the delay. Literature (Ding et al. 2019) studied the problem of energy consumption optimization in a multi-user edge computing system with delay constraints. They used data compression to reduce the size of data transmission. At the same time, many scholars have also studied the problem of energy consumption minimization. Literature (Li et al. 2019), the multi-user computing offloading problem of MEC under a multi-channel wireless interference environment was studied. Meanwhile, the task offloading decision was expressed as a multi-user game with Nash Equilibrium, and a server partitioning algorithm based on clustering was proposed to reduce energy consumption. Literature (Chen et al. 2018), aiming at maximizing long-term utility performance, modeled the optimal computational shunt strategy as a Markov decision process and proposed a computational offloading algorithm based on the two-layer deep Q network by combining the Q function decomposition technology with the two-layer Q network. Literature (Li et al. 2020) the edges of the unmanned aerial vehicles are studied in auxiliary scenario calculated offloading decision-making literature through mutual optimization of UAV trajectory user transmission power and calculating load distribution to maximize the energy efficiency literature UAVs will decompose multiple subproblems, using Dinkelbach algorithm and successive convex approximation optimization technology to solve. In recent years, many scholars have studied the offloading of single-user single MEC, and the proposed method has good performance in terms of time delay or energy consumption. However, the single user single MEC scene cannot meet the requirements of low time delay, low power, and high bandwidth in Real-time video analysis. In public security video surveillance, preprocessing the video image at the edge, only upload refined structured valid data to the cloud for processing. So this article will be built into public security video monitoring multi-user multiple MEC models, joint optimization of energy consumption in time delay, and completely uninstall strategy.

3 Offloading model

With the development of 5 G and the Internet of Things, the number of wireless devices in cities is gradually increasing, and 5 G is rapidly becoming an essential technology in smart cities (Zhang et al. 2014; Liu 2020; Zhang 2019). Providing mixed network signals to edge devices can greatly reduce lossy connections (Johnson and Ketel 2019; Skouby and Lynggaard 2014; Karadimce and Marina 2018). To solve the problem of delayed response, high power consumption and high network bandwidth in Real-time video analysis, the MEC calculation was offloaded and modeled as a multi-user multi-MEC model, as shown in Fig. 1.

The offloading model co-optimizes the time delay and energy consumption. The time delay includes calculation delay and transmission delay; When the video data is collected by the surveillance camera, it can be calculated locally or transmitted to the edge server through wireless transmission such as 5 G for calculation. The time delay and energy consumption required by the calculation of offloading should be taken into account in the calculation of offloading selection. When the edge device chooses to perform the calculation locally, only the calculation delay $T_{Local_compute}$ and calculation energy $E_{Local_compute}$ consumption of the offloading task by the local device need to be considered. When the edge device chooses the edge server for computing offloading, the transmission delay required by the local task for offloading task transmission T_{trans} and the computation delay required by the edge server $T_{MEC_compute}$ for offloading task calculation should be taken into account. In the energy consumption, the transmission energy consumption required by the local device E_{trans} for offloading task transmission should be taken into account. Since the edge server is usually powered by an external power supply independently, the computing energy consumption of the edge server is not considered during edge offloading. Assume that there are m users and n servers in the model, and each user generates an offload

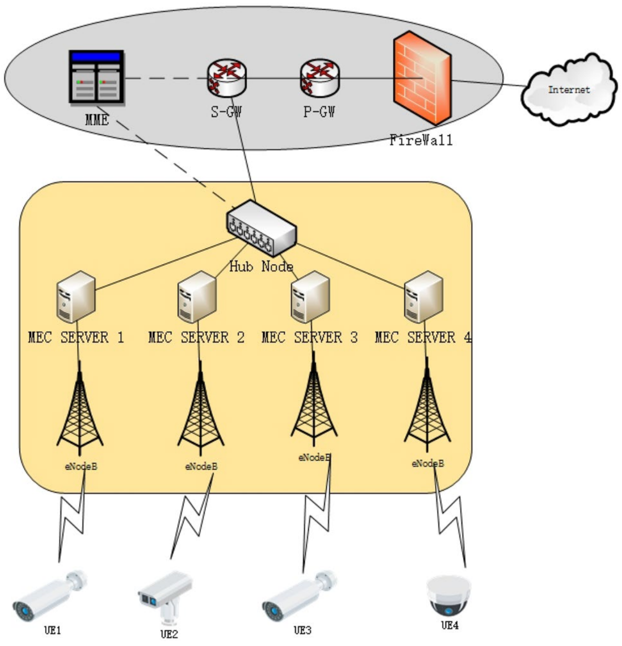

Fig. 1 Offloading model for Real-time video analysis

Task, that is, $Task = \{task_1, task_2, task_m\}$,$S = \{s_1, s_2, , s_n\}$ During the offloading process, each user can choose to perform the calculation locally or to an MEC server for calculation, i.e. $S = \{s_0, s_1, ..., s_n\}$. There are $n + 1$ offloading locations, where s_0 means to perform the calculation locally. So offload tasks offloading location to $Load = \{load_i | load_i \in S\}$, where $i = 1, 2, ..., m$. The computational offloading model is coordinated optimization for time delay and energy consumption, so the optimization objective of the computational offloading model can be expressed as

$$Fitness = \sum_{i=1}^{m} fitness_i \tag{1}$$

$$fitness_i = T_i + E_i \tag{2}$$

Where $fitness_i$ represents the fitness of the *ith* task; *Fitness* represents the fitness of the offloading model, and represents the model minimization goal. T_i is the total offloading delay of user i; E_i is the total energy consumption of offloading user i.

3.1 Local offloading model

When users choose local offload, they only need to consider local computing delay and local computing power consumption.

$$\begin{aligned} fitness_i &= T_i + E_i \\ &= T_{local} + E_{local} \end{aligned} \tag{3}$$

where

$$\begin{aligned} T_{local} &= T_{local\_compute} \\ &= \frac{B_i \times f_i}{CR_{u,i}} \end{aligned} \tag{4}$$

$$\begin{aligned} E_{local} &= E_{local\_compute} \\ &= C_i \times L^2 \times CR_{u,i} \times B_i \times f_i \end{aligned} \tag{5}$$

B_i represents the data volume of $task_i$; f_i represents the number of cycles required for user i to process each bit of data; $CR_{u,i}$ represents the CPU cycle frequency of user i; C_i represents the CPU's effective switching capacitance and L represents the voltage.

3.2 Edge offloading model

Due to the small size of the calculated result, the backhaul delay when the user chooses to offload to the MEC server can be ignored (Truong et al. 2020; Guo and Quek 2020; Jie and Chen-hao 2023)Therefore, when the user chooses to offload to the MEC server, it needs to consider the transmission delay, the calculation delay of the MEC server, and the local transmission energy consumption.

$$\begin{aligned} fitness_i &= T_i + E_i \\ &= T_{trans} + T_{MEC\_compute} + E_{local\_trans} \end{aligned} \tag{6}$$

$$\begin{aligned} T_{trans} &= T_{ij} \\ &= \frac{B_i \times f_i}{v_{ij}} \end{aligned} \tag{7}$$

Where T_{ij} represents the transmission delay of task i transmitted to the *jth* MEC server through wireless transmission technology, and v_{ij} represents the wireless transmission speed. In the calculation of transmission delay, the first thing to calculate is the wireless transmission rate.

According to Shannon's theorem, the transmission speed v_{ij} is calculated as Formula (8).

$$v_{ij} = W \times \log_2\left(1 + \frac{p_i \times H_{ij}}{W \times N_0}\right) \tag{8}$$

Where W represents the wireless channel transmission bandwidth; p_i represents the transmitting power of the local equipment of the *ith* mission; H_{ij} represents the channel gain transmitted from the *ith* task to the *jth* server; N_0 represents the noise power spectral density.

$$T_{MEC\_compute} = \frac{B_i \times f_i}{CR_{s,j}} \tag{9}$$

Where $T_{MEC_compute}$ represents the computing delay of MEC server; CR_{sj} represents the clock frequency of the *jth* MEC server.

$$E_{local\_trans} = p_i \times T_{trans} \tag{10}$$

Where E_{local_trans} represents the energy consumed by the transmission of local device i.

3.3 Optimization objectives

All the tasks were offloaded to a reasonable position to minimize the overall delay and energy consumption, and thus the optimization objective function was obtained.

$$\begin{aligned} \min : Fitness &= \sum_{i=1}^{m} fitness_i \\ &= \sum_{i=1}^{m} (T_i + E_i) \end{aligned} \tag{11}$$

Because the energy of the edge equipment is limited, the energy consumption of the edge equipment should be reduced as much as possible in the offloading process. At the same time, to coordinate the optimization of time delay and energy consumption, the final optimization objective function can be expressed as

$$\min : Fitness = \sum_{i=1}^{m} (T_i + g \times (E_i - E_{\max})) \quad s.t.\ \text{load}_i \in POS \tag{12}$$

Where g is the weight factor of time delay and energy consumption, and the importance of time delay and energy consumption can be adjusted by adjusting the weight factor. E_{max} is the maximum energy available in the edge device. Since low delay is emphasized more in Real-time video analysis scenarios, the weight factor is set to be far less than the delay factor 1. In this study, the weight factor g is set to 0.001.

4 Chaotic quantum particle swarm optimization strategy

4.1 Quantum particle swarm optimization algorithm

In 1995, J. Kennedy and R. Eberhart proposed a particle swarm optimization algorithm (PSO) inspired by the foraging behavior of birds (Kennedy and Eberhart 1995). The particle swarm optimization algorithm regards the individual as a massless and volumeless particle, and the particle searches in the solution space at a certain speed. The historical optimal position in the particle search process is *pbest*. The optimal historical location in the particle swarm is taken as the global optimal solution *Gbest*. In the elementary particle swarm optimization algorithm, particles converge at a finite speed and in a specific direction, so the elementary particle swarm optimization algorithm is prone to prematurity and falls into the local optimum (Luo and Li 2009; Gao et al. 2008). To overcome the limitations of the elementary particle swarm optimization algorithm, Sun proposed the quantum particle swarm optimization algorithm (QPSO) from the perspective of quantum mechanics (Sun et al. 2004). Based on the DELTA well theory of quantum mechanics, the quantum properties of particles make it possible to search in the whole feasible solution space, which greatly improves the global search ability of PSO. The QPSO algorithm describes the state of the particle through the wave function and solves the Schrodinger equation to obtain the probability density function of the occurrence of the particle at a certain point in space. Finally, the position equation of the particle is obtained through Monte Carlo stochastic simulation. The position update equation of particles can be expressed as.

$$X_i(t+1) = P_i \pm \beta \left| mbest - X_i(t) \right| \times \ln(\frac{1}{u}) \tag{13}$$

$$P_i = \omega Pbest_i + (1-\omega) Gbest \tag{14}$$

$$mbest = \sum_{i=1}^{pop} \frac{Pbest_i}{pop} \tag{15}$$

$$\begin{aligned} u &= random(0,1) \\ \omega &= random(0,1) \end{aligned} \tag{16}$$

Where P_i is used to update the position of the particle at the next moment; β represents the expansion coefficient; $Pbest_i$ represents the historical optimal position of particle i; *Gbest* represents the global optimal position; *pop* represents particle population size; *mbest* is the average of all the positions.

Compared with the elementary particle swarm optimization algorithm, the quantum particle swarm optimization algorithm (QPSO) has a stronger global search ability and stronger ability to jump out of the local optimal solution. Therefore, the quantum particle swarm optimization algorithm has been applied by scholars in various fields. However, QPSO also has its disadvantages, such as insufficient population diversity, weak global search ability, weak ability to escape from a local optimal solution and poor performance in the face of complex multidimensional NP-hard problems. To improve the shortcomings of QPSO algorithm, many scholars have done a lot of research on it Zhang (2018), WANG and Hong-rui (2020).

4.2 Chaos quantum particle swarm optimization algorithm

4.2.1 Measure the state of particle aggregation

Lemma 4.1 *When the particle swarm optimization algorithm falls into the local optimum or reaches the global optimum, the particle swarm will converge to several positions in the search space, and the fitness variance of the particle swarm* σ^2 *is equal to* 0 (*Van Den Bergh* 2007).

When the particle swarm aggregates in the global extreme value, it is reflected in the fitness function, that is, the fitness value of the particle swarm is the same. Variance is a measure that reflects the degree of dispersion of a set of data. Therefore, the fitness variance σ^2 can reflect the aggregation state of particle swarm fitness, and the fitness of particle swarm is determined by its final convergence position. Therefore, the aggregation state of particles can be measured by the fitness variance σ^2 of the particle swarm. The definition of fitness variance $\sigma^2$is given below, as shown in Formula

$$\begin{aligned} \sigma^2 &= \frac{1}{pop} \sum_{i=1}^{pop} \left(\frac{f_i - f_{avg}}{f} \right)^2 \\ f &= \max\{1, \max\{|f_i - f_{avg}|\}\} \\ i &= 1, 2, \ldots, pop \end{aligned} \tag{17}$$

Where f_i represents particle fitness; f_{avg} represents the average fitness of particles; f is the normalized factor, which is used to limit the size of σ^2. Literature (Lin et al. 2008) shows that the smaller σ^2 is, the particle swarm tends to converge, and on the contrary, it is in a random search state. In particular, when σ^2 is equal to 0, it indicates that the particle swarm is locally or globally optimal. When the fitness variance σ^2 is less than a certain small value, the particle swarm is forced to jump out of the local optimum by adding disturbance to the particle swarm position and achieve a better global convergence effect.

4.2.2 chaos theory

Chaos is a seemingly random motion, which obtains random states from deterministic equations. Chaos theory holds that in a chaotic system, a small change in the initial conditions will lead to a huge change in the state of the future system after continuous amplification. Because of the inherent sensitivity, randomness, scale law, universality, and other characteristics of a chaotic system, it is suitable for algorithm optimization. The classical Logistic chaotic equation is used in this paper, and its expression is shown as Formula

$$\begin{aligned} Z_{n+1} &= \mu Z_n(1 - Z_n) \\ Z_n &\in (0, 1) \end{aligned} \tag{18}$$

When $\mu = 4$, the system is in a state of complete chaos. In Formula (18), Z_n represents the chaotic particle of the n generation. The combination of quantum particle swarm optimization (QPSO) and chaos theory is used to improve the global search ability of the algorithm

When the QPSO is detected to fall into the local optimal solution (that is, the σ^2 value is small), chaos disturbance is added to the particle local optimal solution to make the QPSO jump out of the local optimal solution. The disturbance formula can be expressed as

$$\begin{aligned} y_i &= Pbest_i \pm \lambda \times Z_i \ (i = 1, 2, ...\text{pop}) \\ \lambda &= \lambda \times 0.5 \end{aligned} \tag{19}$$

Where $Pbest_i$ represents the historical best position of particle i; Z_i represents the chaos particle. As λ decreases linearly, the search of the algorithm becomes more and more refined; The fitness of $Pbest_i$ and $Gbest$ was compared with that of y_i and updated.

The offloading locations in the computational offloading are discrete and $Z_i \in (0, 1)$. therefore the chaos perturbation formula requires to be changed for offloading. The final formula of chaos disturbance can be expressed as

$$\begin{aligned} y_i &= Pbest_i \pm \lambda(Z_i\ (\text{S}_{\max}\text{-S}_{\min})\text{+S}_{\min}) \\ i &= 1, 2, ...\text{pop} \\ \lambda &= \lambda \times 0.5 \end{aligned} \tag{20}$$

Where S_{max} and S_{min} represent the position constraints of particles.

5 Improved Offloading Approach for MEC Based on CQPSO Strategy

Chaotic quantum particle swarm optimization algorithm (CQPSO) has slow convergence speed and poor global search ability when dealing with complex high-dimensional functions. Therefore, an improved algorithm based on chaotic quantum particle swarm Optimization (ICQPSO) is proposed in this paper.

5.1 Analysis of initial population diversity

Due to the sensitivity of chaotic system to the initial state, the larger the population diversity of the initial state of particle swarm is, the more helpful the algorithm is to search for the global optimal solution. The formula of population diversity measurement can be expressed as

$$\begin{aligned} \text{var} &= \sum_{i=1}^{m} \frac{POS_i - avgpos}{m} \\ avgpos &= \sum_{i=1}^{m} \frac{POS_i}{m} \end{aligned} \tag{21}$$

The variable *var* is used to measure the value of population diversity. POS_i is the position of the particle; *avgpos* represents the average population position; m is the population size.

Lemma 5.1 *Population diversity is the key factor affecting the convergence of the algorithm, and is also the key index to measure the performance of the algorithm. The lack of population diversity in the quantum particle swarm optimization algorithm is the essence of the algorithm falling into the local optimal solution.*

To enhance the global search ability of chaotic quantum particle swarm optimization (CQPSO), the exchange operator and mutation operator of the Genetic Algorithm are added based on chaotic quantum particle swarm optimization (CQPSO). The exchange operator and mutation operator can effectively improve the diversity of the population, increase the coverage of the search space, and improve the global search ability of the algorithm. The pseudocodes of the switching algorithm and mutation algorithm are shown in the algorithm.

Based on the above analysis, this paper proposes the pseudo-code of edge computing offloading method of multi-user and multi-MEC server based on chaotic quantum particle swarm optimization strategy, as shown in algorithm 1.

```
Algorithm 1: ICQPSO ALGORITHM
Data: population $pop$,Number of initializations $N$ , Expansion
      coefficient $\beta_{max}$ and $\beta_{min}$, chaos search threshold $C$ , Number of
      chaos searches $C\_max$ ,Position constraint,Delay threshold
      $\alpha$,Energy max $E_{max}$,Memory threshold $\delta$,Threshold of success
      rate $\gamma$,Threshold of success rate $SR$ , Rounds
Result: Optimum offloading location $Gbest$
1  Initialize particle with $pop$ $N$ times for max $var$;
   /* The formula (21)                          */
2  Initialize $Pbest$ and $Gbest$;
   /* As step 2 to step 5 in Algorithm 1        */
3  while $T \le \alpha$ && $M \le \delta$ && $SR \le \gamma$ && $E \le E_{\max}$ && $round \le Rounds$/* theorem 1   */
4   do
5      $\beta = \beta_{\max} - \frac{round}{rounds}(\beta_{\max} - \beta_{\min})$;
6      Calculate $mbest$;
       /* The formula (15)                      */
7      for $i$=0 to $POP$ do
8          Calculate $P_i$;
           /* The formula (14)                  */
9          Update particle position $X_i$;/* The formula (13)   */
10     end
11 end
12 update $Pbest$ and $Gbest$;
   /* As step 18 to step 25 in Algorithm 1      */
13 Exchange and mutation operator;
   /* Algorithm 2                               */
14 Calculate $\sigma^2$ ;
   /* The formula (17)                          */
15 if $\sigma^2 < C$ then
16     Initialize Chaos particle;
17     for $i$=0 to $C\_max$/* chaos search */
18      do
19         Calculate $y_i$ ;
           /* The formula (20)                  */
20         Chaos particle evolve;
           /* The formula (17)                  */
21         foreach $y_i$ do
22             if $fitness(y_i) < fitness(Pbest_i)$ then
23                 $Pbest_i = y_i$
24             end
25             if $fitness(y_i) < fitness(Gbest)$ then
26                 $Gbest = y_i$
27             end
28         end
29     end
30 end
```

5.2 Algorithm complexity analysis

The complexity of the algorithm is usually expressed as a function whose domain is the size of the input data and whose value ranges from the number of steps to be performed (time complexity) or the storage space required (space complexity). Time complexity is the amount of time it takes to solve a problem and is usually measured by calculating the deployment.

Corollary 5.1 *The ICQPSO time complexity is* $O(R * P * M)$.

Proof Suppose that the particle population size is: P; The number of iterations is: R; The number of offloading tasks is: M. In the offloading problem of edge calculation, each iteration of ICQPSO algorithm has to go through

(1) several particle initializations;
(2) Update the position of particles;
(3) Update *pbest* and *Gbest*;
(4) Exchange and variation;
(5) Calculate the particle swarm aggregation state σ^2;
(6) Chaos search.

In this paper, particle initialization times and chaotic search times are set as fixed values, which are far less than the time complexity of population number and particle dimension algorithm as follows.

$$\begin{aligned} O(P,R,M) &= O(P*M) + R*(O(P*M) + O(P)+ \\ &\quad O(P*M) + O(P) + O(P*M)) \\ &\approx O(P*M) + 3O(R*P*M) \approx O(R*P*M) \end{aligned}$$

It can be seen from the formula that the time complexity of ICQPSO algorithm is proportional to the number of iterations, population size and particle dimension. And the time complexity is in the same order of magnitude as other algorithms(Lin et al. 2008). □

Corollary 5.2 *The ICQPSO time complexity is* $O(P * M)$.

Proof The space complexity of the algorithm is the amount of storage space required by the algorithm. In ICQPSO algorithm, the data to be stored are particle population, *pbest* and several auxiliary variables. Since the storage capacity of the particle population is related to the particle dimension of the particle population, and the size of *pbest* is consistent with the size of the particle population, the spatial complexity of the ICQPSO algorithm is proportional to the number of the particle population, proportional to the particle dimension, and the spatial complexity is $O(P * M)$. □

6 SIMULATION RESULTS

6.1 Experiments settings

In this paper, Python language is used to conduct four groups of simulation experiments. Experiment 1: Compare the convergence of the ICQPSO algorithm with other algorithms in fitness, time delay, and energy consumption. Experiment 2: Compare the influence of the different number of edge devices on delay. Experiment 3: Compare the performance of each algorithm under the different number of MEC servers. Experiment 4: Compare the effects of different data quantities on time delay and energy consumption.

This paper studies the edge computing offloading method of multi-user and multi-MEC servers in mobile edge computing. In the multi-user and multi-MEC server Computation offloading scenario, the experimental computing task selects the appropriate offload location among local devices and edge servers in the way of complete offload. Multi-user multi-MEC server computing offloading scenarios have been widely used in many fields, such as real-time video analysis, industrial Internet of Things, Internet of vehicles, and so on. The experimental parameters are shown in Table (51). In this paper, real-time video analysis as the application scenario, before the simulation experiment has been carried out the relevant test work.

Through four groups of experiments, the performance of the ICQPSO algorithm in an edge computing environment is measured from different aspects. At the same time, the effects of the number of edge devices, the number of edge servers, and the amount of offloading task data on the delay and energy consumption of edge computing task offloading are analyzed. In the four sets of experiments, the improved chaotic quantum swarm particle swarm optimization (ICQPSO) proposed in this paper compares genetic algorithm (GA), differential evolution algorithm (DE), immune algorithm (IA), quantum particle swarm optimization (QPSO) and chaotic quantum particle swarm optimization (CQPSO). The results show that improved Chaos Quantum Swarm Particle Swarm Optimization (ICQPSO) has better performance.

6.2 Algorithm convergence comparison

In this paper, the global search ability and local fine searchability of the traditional chaotic quantum particle swarm optimization algorithm are improved. To measure the convergence of the ICQPSO algorithm proposed in this paper, a set of convergence comparative experiments are set up.

When the number of edge devices is set to 250 and the number of MEC servers is set to 10, the data volume of offloading task is set to 3MB. To simulate edge calculation offloading in a complex environment, the other parameter settings are shown in Table 1. The average fitness changes of the improved chaotic quantum particle swarm optimization algorithm (ICQPSO) and other heuristic algorithms with the number of iterations are shown in Fig. 2. The average delay of each algorithm changes with the number of iterations, as shown in Fig. 3. The average energy consumption of each algorithm varies with the number of iterations, as shown in Fig. 4. From these figures, we can see that the improved chaotic quantum particle swarm optimization algorithm (ICQPSO) tends to converge and approaches the global optimal solution after 4500 iterations, immune optimization algorithm (IA) and genetic algorithm (GA) converge the fastest and perform the second, and differential evolution algorithm (DE) and chaotic quantum particle swarm optimization algorithm (CQPSO) converge faster than the ICQPSO, but the effect is not as good as the ICQPSO, Quantum particle swarm optimization (QPSO) converges quickly but has the worst effect.

From the analysis of the average fitness, average delay, and average energy consumption in Figs. 2, 3 to 4, several heuristic algorithms compared in this paper are all convergent.

GA, IA, and QPSO have the fastest convergence speed. After about 600 computational iterations, the algorithm is close to the global optimal solution, while the DE, CQPSO, and ICQPSO approach the global optimal solution after 4500 iterations.

The ICQPSO proposed in this paper has the smallest global optimal solution and the strongest global searchability. The global searching ability of GA and IA is next. The QPSO algorithm has the weakest global search capability. According to the analysis of fitness variance σ^2 (Lemma 4.1) and population diversity (Lemma 5.1) in this paper, the size of the global optimal solution can reflect the ability of the algorithm to jump out of the local optimal solution and the size of population diversity in the algorithm. The ICQPSO in this paper has the strongest global searchability, which can reflect that the ICQPSO algorithm has a stronger ability to jump out of the local optimal solution and greater population diversity. This is because in the ICQPSO algorithm, by looking for particle swarm with larger population diversity as the initial particle swarm, the exchange operator and mutation operator of the genetic algorithm are introduced to improve the population diversity of the ICQPSO, to improve the ICQPSO algorithm's global searchability. At the same time, the convergence state of the algorithm is measured by the fitness variance σ^2, and the ergodicity of the chaotic system is used to improve the ability of the algorithm to jump out of the local optimal solution.

Based on the above analysis, the ICQPSO algorithm proposed in this paper has a slower convergence speed, but has

Table 1 parametric description

Param	Meaning	Value
B_i	Data volume of $task_i$	7–40 Mb
f_i	Cycles to process each bit of data	800–1200 c/b
$CR_{u,i}$	CPU frequency	1–3GHz
C_i	CPU's efffective switching capacitance	4–12
L	Voltage	1.35V
W	Wireless channel transmission bandwidth	1MHz
p_i	Transmitting power of local devices	0.1–0.5 W
$H_{i,j}$	Channel gain transmitted i task to j server	2×10^{-10}-2×10^{-6}
N_0	The noise power spectral density	1×10^{-9}
$CR_{s,j}$	Clock frequency of jth server	4–8GHz
g	Weight factor	0.001
pop	Particle population	50
N	Times of initializations	10
β_{max}	Maximum expansion coefficient	0.9
β_{min}	Minimum expansion coefficient	0.4
C	Chaotic search threshold	0.1
C_{max}	Chaos search times	10

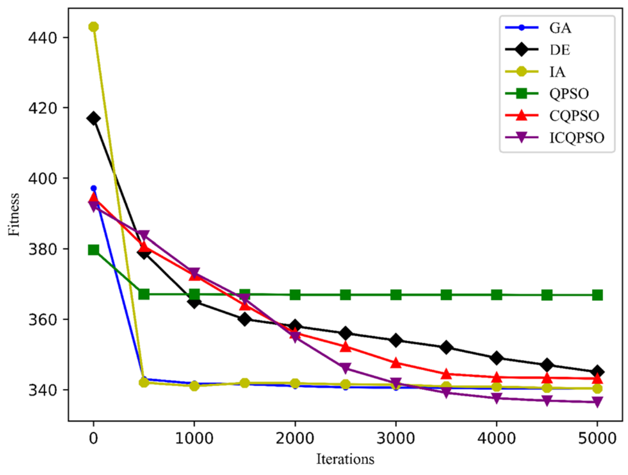

Fig. 2 Fitness convergence comparison with different algorithms

stronger global searchability, and can obtain results closer to the theoretically optimal solution. At the same time, the contradiction between the convergence speed and the global search ability of the ICQPSO algorithm also conforms to the NFL theorem (No Free Lunch)(Wolpert and Macready 1997). The NFL theorem says that an algorithm that is stronger in one area is weaker in another. In the edge computing task offload, the task offload decision is calculated in the scheduling server in the edge computing. The scheduling server has stronger computing resources, and the final

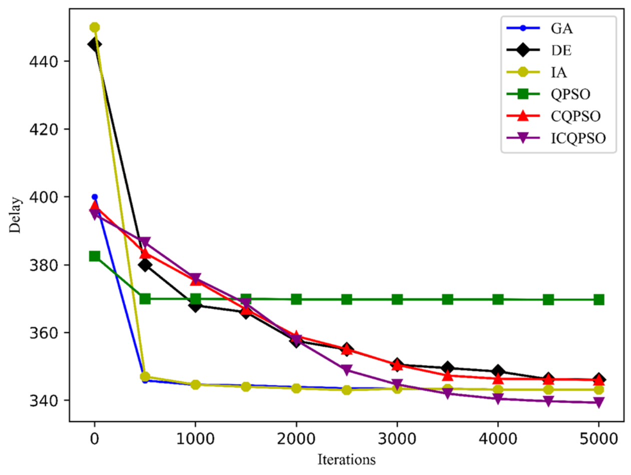

Fig. 3 Comparison delay with different algorithms

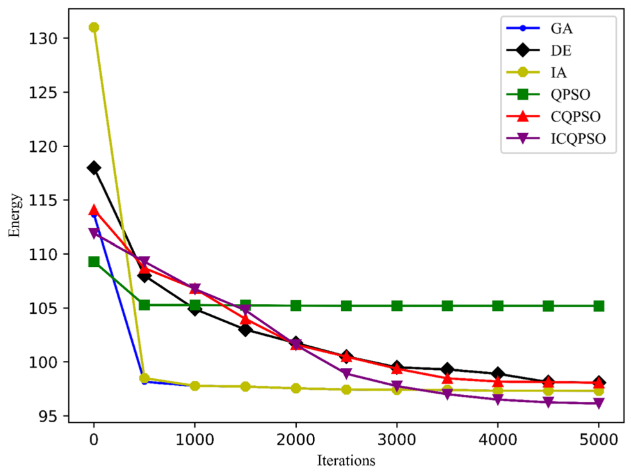

Fig. 4 Comparison of energy consumption convergence with different algorithms

computation result is more important to the task offloading decision than the computation speed.

Therefore, the ICQPSO algorithm proposed in this paper is more suitable for the combined optimization of time delay and energy consumption in MEC calculation offloading.

6.3 The effect of the number of edge devices on offloading

In edge computing task offloading, the number of edge devices influences the complexity, delay, and energy consumption of the task offloading decision. To measure the performance of the ICQPSO algorithm under the different number of edge devices, set the number of devices as 50,100,150,200,250. The number of MEC servers is 10, and the amount of offloading task data is 3MB. Other parameter settings are shown in Table 1. The average fitness under the different number of edge devices is shown in Fig. 5. The average delay is shown in Fig. 6. And the average energy consumption is shown in Fig. 7. From Figs. 5, 6 and 7, it can be seen that the number of edge devices has an impact on energy consumption and time delay in the calculation of offloading. With the increase in the number of edge devices, the adaptability and delay of calculating offloading also increase correspondingly. From the vertical axis, the ICQPSO algorithm proposed in this paper has better performance than other heuristic algorithms.

QPSO algorithm is most affected by the number of edge devices, and its average fitness and average delay increase fastest. The CQPSO algorithm, GA algorithm, IA algorithm, and ICQPSO algorithm has similar trends in the average fitness, average delay, and average energy consumption affected by the number of edge devices, while the DE algorithm is more affected by the number of edge devices. ICQPSO algorithm is the lowest in average fitness, average delay, and average energy consumption, and the range of change is the least. The performance of the IA algorithm GA algorithm is better than the CQPSO algorithm, but inferior to the ICQPSO algorithm. The performance of the four algorithms is consistent with the convergence comparison experiment in Sect. 6.2 of this paper. In the experiment in Sect. 6.2, the experimental results and algorithm are analyzed in detail. Because the proposed ICQPSO algorithm has greater population diversity, stronger ability to jump out of the local optimal solution, and stronger global searchability.

From the analysis of the experimental results of the influence of the number of edge devices on energy consumption in Fig. 7, it can be seen that the changing trend of energy consumption, fitness, and delay is not consistent. The energy consumption decreases with the increase in the number of edge equipment. According to the analysis of the energy consumption model in the offloading model in Sect. 3 (Formula 5 and Formula 10), with the increase of the number of edge devices, edge offloading will be more than local offloading for the offloading task. In the local offloading model, the energy consumption is the local computing energy consumption, while in the edge offloading model, the energy consumption is the transmission energy consumption. In the case of the same amount of data, the transmission energy consumption is less than the local computing energy consumption. As a result, energy consumption is on the decline.

According to the above analysis, the ICQPSO algorithm is superior to other heuristic algorithms in terms of time delay and energy consumption under different offloading task numbers. The oscillation amplitude of the ICQPSO algorithm is the smallest in the delay curve of the different number of devices, which indicates that the ICQPSO algorithm has a better global search ability when facing multi-dimensional complex NP-hard problems and can jump out of the local optimum advantage.

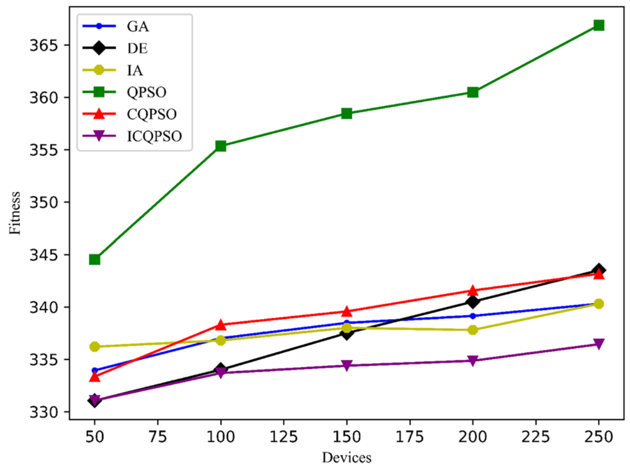

Fig. 5 Fitness for numbers of devices

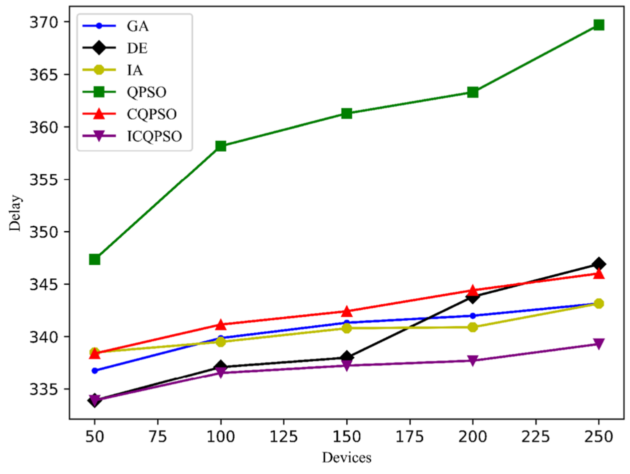

Fig. 6 Delay for numbers of devices

6.4 The impact of the number of edge servers on offloading

To measure the performance of the ICQPSO algorithm under different numbers of MEC servers, the number of MEC servers was set as 10,15,20,25,30, the numbers of edge devices was set as 150, and the amount of offloading task data was set as 3MB. Other parameters were shown in Table 1. The average fitness under the different number of MEC servers is shown in Fig. 8. The average delay is shown in Fig. 9. The average energy consumption is shown in Fig. 10.

From the analysis of the experimental results in Figs. 8, 9 and 10, when the number of edge servers is the same, the ICQPSO algorithm proposed in this paper has the minimum

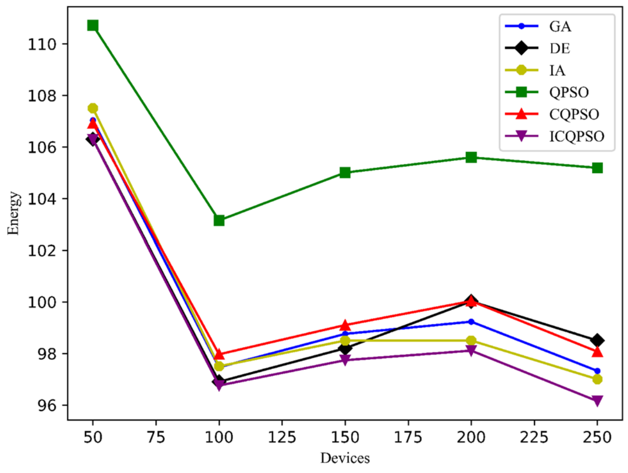

Fig. 7 Energy for numbers of devices

time delay and energy consumption. The delay and energy consumption of the GA algorithm, DE algorithm, and IA algorithm is similar to the ICQPSO algorithm. The time delay and energy consumption performance of the CQPSO algorithm is between the genetic algorithm (GA algorithm, DE algorithm, IA algorithm) and QPSO algorithm. The experimental conclusion is consistent with the previous two conclusions.

As the number of edge servers increases, the average fitness, average delay, and average power consumption do not decrease linearly. Analyzed from the offloading model and experimental parameters, the only variable of the number of edge servers in this experiment. In the edge server, the factor that affects the offloading model is the server CPU clock frequency. To simulate the edge calculation offloading decision in a complex environment, the clock frequency parameters of the server CPU in this experiment were randomly set between 4GHz-8GHz, as shown in Table 1. When the CPU cycle frequency parameter of the edge server is set low, even an increase of the number of edge servers may lead to an increase of delay and energy consumption. From Figs. 8, 9 and 10, in addition to the experiment in which the number of edge servers in the first group was 10, the number of edge servers in the other four groups was set to 15, 20, 25, and 30 respectively, presenting a trend of gradual decrease. This phenomenon is consistent with the perceptual perception that power consumption and delay gradually decrease as edge servers increase. The experiment of the first group with the number of edge servers 10 showed a small-time delay and energy consumption. According to the above analysis, the reason is that the average value of CPU clock frequency randomly set in this group of servers is larger.

According to the above analysis, the ICQPSO algorithm proposed in this paper is superior to other heuristic algorithms, which is consistent with the conclusions drawn from the previous two experiments. Meanwhile, the influence of the number of edge servers on the calculation of offloading is also roughly consistent with the analysis of the offloading model in this paper.

6.5 The effect of offloading task data volume on offloading

The task to measure data volume and the relationship between the time delay set the amount of data to 10 MB, respectively, 20 MB, 30 MB, 40 MB, 50 MB, offloading task number 50, edge server number 10. The average fitness was shown in Fig. 11. The average delay is shown in Fig. 12. The average energy consumption is shown in Fig. 13. From Fig. 11, 12 to 13, it can be seen that with the increase in the amount of task data, the delay gradually increases. When the amount of data is small, the delay of each algorithm has little difference. When the amount of data increases gradually, the performance

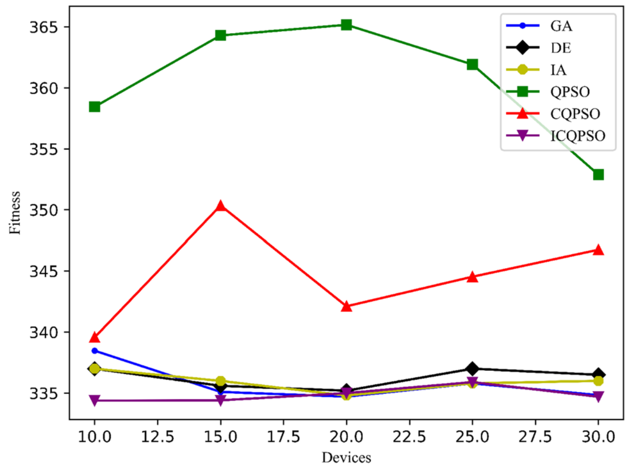

Fig. 8 Fitness for numbers of servers

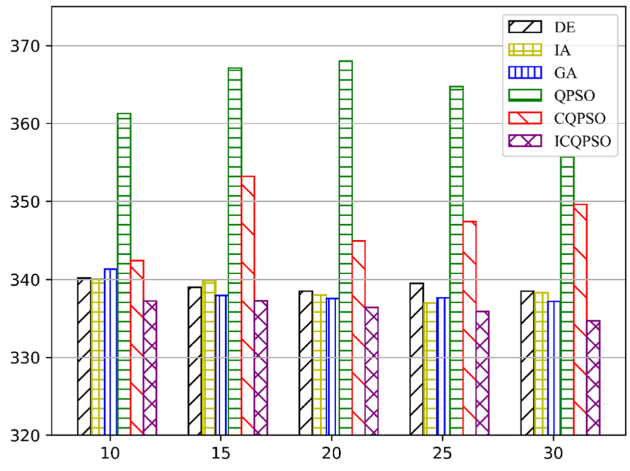

Fig. 9 Delay for numbers of servers

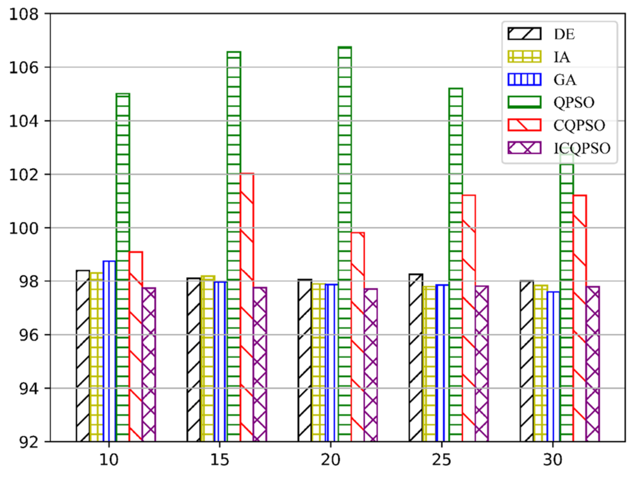

Fig. 10 Energy for numbers of servers

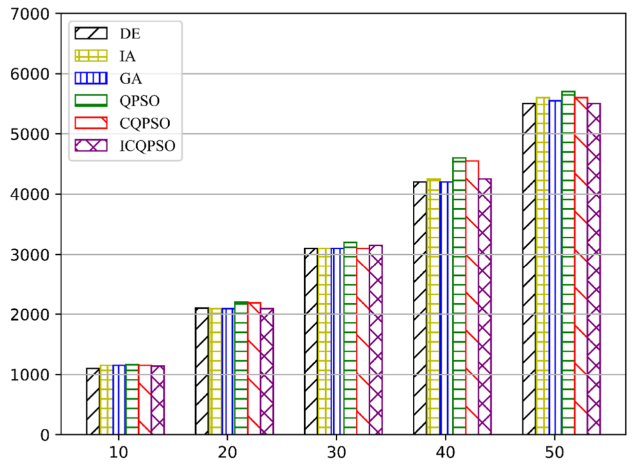

Fig. 12 The effect of task data volume on delay

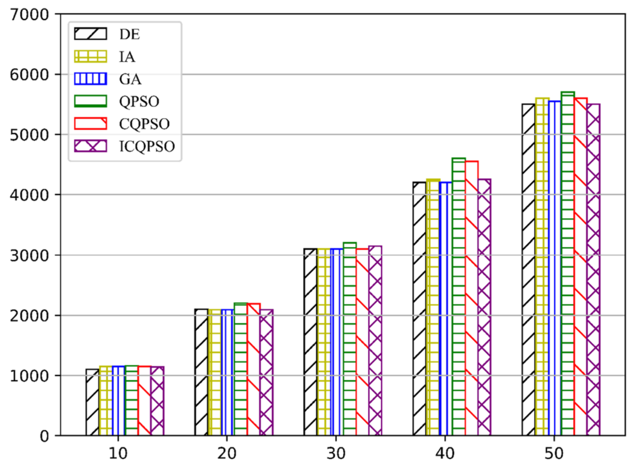

Fig. 11 The effect of task data volume on fitness

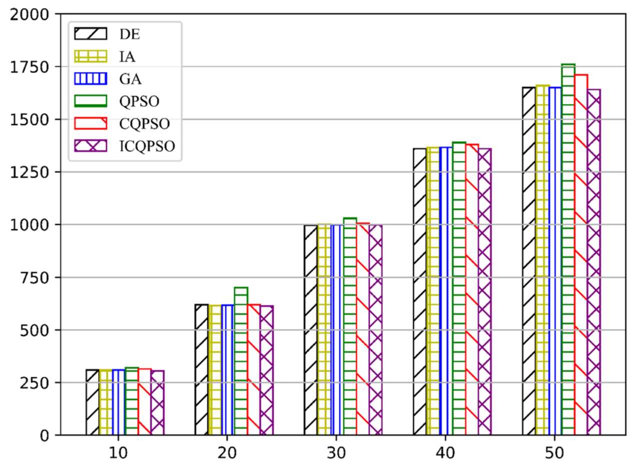

Fig. 13 The effect of task data volume on energy

gap between the algorithms appears. When the data volume is 50MB, the ICQPSO algorithm has the smallest delay and the largest delay gap with other algorithms.

The results show that the average fitness, average delay, and average energy consumption increase linearly with the increase of offloading task data. According to the analysis of the offloading model in Sect. 3, the offloading task data amount is linearly correlated with the time delay and energy consumption. There is also a linear correlation between fitness and time delay and energy consumption. The experimental results agree with the analysis of the offloading model in this paper. At the same time, the ICQPSO algorithm proposed in this paper is better than other heuristic algorithms under the same offloading task data volume. This experimental conclusion is consistent with the previous three experimental conclusions. With the increase of offloading task data, the performance gap between the ICQPSO algorithm and other algorithms gradually begins to appear. When the offloading task data volume is 10 MB, the time delay and energy consumption of all the heuristic algorithms tested in this paper are almost the same. However, when the offloading task data volume is 50MB, the time delay and energy consumption of the four algorithms begin to show obvious differences.

According to the above analysis, this paper proposes that the improved chaotic quantum particle swarm optimization algorithm has better performance in the face of multi-dimensional complex functions, and is more suitable for the offloading of edge computing tasks in complex environments.

7 Conclusion

To synergistically optimize time delay and energy consumption in a complex edge computing task offloading environment, the offloading model is modeled in this paper. The performance of chaotic quantum particle swarm optimization (CQPSO) is analyzed in detail. In this paper, we analyze the reason that the general heuristic algorithm is easy to fall into the local optimal solution and the global search ability is not good. Meanwhile, the Improved Chaos quant-pour Particle Swarm Optimization algorithm (ICQPSO) is proposed to solve these problems. In this paper, four groups of experiments are carried out to test the influence of different variables on edge offloading. In each experiment, the results were analyzed in detail. The experimental results show that the improved chaotic quantum particle swarm optimization algorithm is superior to the quantum particle swarm optimization algorithm (QPSO) in the face of multi-dimensional complex functions. The improved chaotic quantum particle swarm optimization algorithm (ICQPSO) has a stronger global search ability and can effectively jump out of the local optimal solution. The ICQPSO algorithm performs well in the collaborative optimization of time delay and energy consumption in the multi-user and multi-MEC server scenarios of mobile edge computing. In the multi-user multi-MEC server scenario, it is also necessary to study the separability of tasks and the collaborative offloading of edge cloud computing, which is the content to be studied in the future.

References

Achmad KA, Nugroho LE, Djunaedi A, et al. (2018) Smart City Model: a Literature Review[C] 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)

Bailas C, Marsden M, Zhang D, et al. (2018) Performance of video processing at the edge for crowd-monitoring applications[C] 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). IEEE, 482-487

Cao LX (2022) Task Offloading Method of Edge Computing in Internet of Vehicles Based on Deep Reinforcement Learning. Clust Comput 25(1):1175–1187. https://doi.org/10.1007/s10586-021-03532-9

Cao B, Zhang L, Li Y et al (2019) Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework[J]. IEEE Commun Mag 57(3):56–62

Chen L (2022) An Approach of Flow Compensation Incentive based on Q-Learning Strategy for IoT User Privacy Protection. AEU-Int J Electron C 2022(148):1–20. https://doi.org/10.1016/j.aeue.2022.154172

Chen L (2023) A Novel Offloading Approach of IoT User Perception Task Based on Quantum Behavior Particle Swarm Optimization. Futur Gener Comput Syst 2023(141):577–594. https://doi.org/10.1016/j.future.2022.12.016

Chen L, Zhang J (2020) A multi-path routing protocol based on link lifetime and energy consumption prediction for mobile edge computing[J]. IEEE Access 8(1):69058–69071. https://doi.org/10.1109/ACCESS.2020.2986078

Chen X, Zhang H, Wu C et al (2018) Optimized Computation Offloading Performance in Virtual Edge Computing Systems via Deep Reinforcement Learning[J]. IEEE Internet of Things Journal 1–1

Cui YY, Zhang T (2019) New Quantum-Genetic Based OLSR Protocol (QG-OLSR) for Mobile Ad hoc Network. Appl Soft Comput 80(7):285–296. https://doi.org/10.1016/j.asoc.2019.03.053

Ding X, Li Q, Zhu H. (2019) Energy-Saving Computation Offloading by Joint Data Compression and Resource Allocation for Mobile-Edge Computing[J]. IEEE Communications Letters, PP(99):1-1

Dong WM, Zhang T (2022) New Computing Tasks Offloading Method for MEC Based On Prospect Theory Framework. IEEE Transactions on Computational Social Systems 12:1–13. https://doi.org/10.1109/TCSS.2022.3228692

Dubey S, Meena J. (2020) Computation Offloading Techniques in Mobile Edge Computing Environment: A Review[C]//2020 International Conference on Communication and Signal Processing (ICCSP). IEEE, 1217-1223

Gao Y, An X, Liu J (2008) A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation[C] 2008 international conference on computational intelligence and security. IEEE 1:61–65

Ge H (2019) New Multi-hop Clustering Algorithm for Vehicular Ad Hoc Networks[J]. IEEE Trans Intell Transp Syst 20(4):1517–1530. https://doi.org/10.1109/TITS.2018.2853165

Gundu SR, Anuradha T (2020) Digital Data Growth and the Philosophy of Digital Universe in View of Emerging Technologies. International Journal of Scientific Research in Computer Science and Engineering 8(2):59–64

Gundu SR, Panem CA (2022) Cloud Computing and its Service Oriented Mechanism. Akinik Publications, New Delhi

Gundu Srinivasa Rao, Panem Charan Arur, Satheesh S. (2022) High-Performance Computing-Based Scalable Cloud Forensics- as-a-Service Readiness Framework Factors-A Review. scrivener publishing(2022).ISBN: 978 1119812494 (Series: Advances in Cyber Security)

Gundu Srinivasa Rao (2021). OBSERVED ISSUES IN CLOUD-BASED WEB COMMERCE ADOPTION FOR THE FINANCIAL TRANSACTIONS IN HYDERABAD. JOURNAL OF MECHANICS OF CONTINUA AND MATHEMATICAL SCIENCES. 16. https://doi.org/10.26782/jmcms.2021.09.00001

Gundu Srinivasa Rao, Panem Charan Arur, Thimmapuram A, et al. (2021) Emerging computational challenges in cloud computing and RTEAH algorithm based solution. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-021-03380-w

Gundu Srinivasa Rao, Anuradha T. (2020) “Improved Virtual Machine Load Balance using RTEAH Algorithm”, Journal of Mechanics of Continua and Mathematical Sciences (JMCMS), VOLUME 15,ISSUE 2 - February, pp.200-211, https://doi.org/10.26782/jmcms.2020.02.00018

Gundu Srinivasa Rao, Anuradha T. (2019). Improved Hybrid Algorithm Approach based Load Balancing Technique in Cloud Computing. Global journal of computer science and technology

Gundu Srinivasa Rao, Panem Charan Arur, Thimmapuram A. (2020) "Hybrid IT and Multi Cloud an Emerging Trend and Improved Performance in Cloud Computing". SN COMPUT. SCI. 1, 189. https://doi.org/10.1007/s42979-020-00277-x

Guo K, Quek TQS (2020) On the Asynchrony of Computation Offloading in Multi-User MEC Systems[J]. IEEE Trans Commun 68(12):7746–7761

Hassan N, Yau KLA, Wu C (2019) Edge computing in 5G: A review[J]. IEEE Access 7:127276–127289

Hassanein H, Zorba N, Han S et al (2019) Crowd Management[J]. IEEE Commun Mag 57(4):18–19

Hidayat F. (2020) Intelligent Video Analytic for Suspicious Object Detection: A Systematic Review[C] 2020 International Conference on ICT for Smart Society (ICISS). IEEE, 1-8

Hu Y, Cui T, Huang X, et al. (2019) Task Offloading Based on Lyapunov Optimization for MEC-assisted Platooning[C]//2019 11th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 1-5

Huynh LNT, Pham QV, Nguyen TDT, et al. (2020) A Study on Computation Offloading in MEC Systems using Whale Optimization Algorithm[C] 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM). IEEE, 1-4

Jie Zhang, Chen-hao Ni (2023) New Method of Edge Computing Based Data Adaptive Return in Internet of Vehicles. IEEE Transactions on Industrial Informatics 6:1–11. https://doi.org/10.1109/TII.2023.3285301

Johnson D, Ketel M. (2019) IoT: The Interconnection of Smart Cities[C] 2019 SoutheastCon. IEEE, 1-2

Karadimce A, Marina N. (2018) Smart Mobile City Services in the 5G Era[C]//2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE, 1-6

Karaduman G, Karakose M, Akin E. (2018) Determination of the Optimum Number of Cameras for Monitoring In Smart Cities[C] 2018 International Conference on Artificial Intelligence and Data Processing (IDAP). Computer Engineering Department, Firat University, Elazig, Turkey

Kennedy J, Eberhart R (1995) Particle swarm optimization[C] Proceedings of ICNN'95-international conference on neural networks. IEEE 4:1942–1948

Khan LU, Yaqoob I, Tran NH et al (2020) Edge-Computing-Enabled Smart Cities: A Comprehensive Survey[J]. IEEE Internet Things J 99:1–1

Li G, Lin Q, Wu J et al (2019) Dynamic computation offloading based on graph partitioning in mobile edge computing[J]. IEEE Access 7:185131–185139

Li M, Cheng N, Gao J, et al. (2020) Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization[J]. IEEE Transactions on Vehicular Technology, PP(99):1-1

Lin L, Liao X, Jin H et al (2019) Computation offloading toward edge computing[J]. Proc IEEE 107(8):1584–1607

Lin X, Feng B, Sun J. (2008) Chaos quantum-behaved particle swarm optimization algorithm[J]. Computer Engineering and Design, 10

Liu S (2017) Novel Unequal Clustering Routing Protocol Considering Energy Balancing Based on Network Partition & Distance for Mobile Education[J]. J Netw Comput Appl 88(15):1–9. https://doi.org/10.1016/j.jnca.2017.03.025

Liu S (2020) Adaptive Repair Algorithm for TORA Routing Protocol based on Flood Control Strategy[J]. Comput Commun 151(1):437–448. https://doi.org/10.1016/j.comcom.2020.01.024

Liu H, Eldarrat F, Alqahtani H et al (2017) Mobile edge cloud system: Architectures, challenges, and approaches[J]. IEEE Syst J 12(3):2495–2508

Luo Y, Li L. (2009) Chaos quantum-behaved particle swarm optimization algorithm with hybrid discrete variables[C] 2009 International Conference on Artificial Intelligence and Computational Intelligence. IEEE, 1: 535-539

Moorthy R, Upadhya V, Holla VV, et al. (2020) Challenges Encountered in Building A Fast and Efficient Surveillance System: An Overview[C] 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 731-737

Ni CH, Zhang J (2022) A Kind of Novel Edge Computing Architecture Based on Adaptive Stratified Sampling. Comput Commun 2022(183):121–135. https://doi.org/10.1016/j.comcom.2021.11.012

Ni CH, Zhang J (2023) New Method of Vehicle Cooperative Communication Based on Fuzzy Logic and Signal Game Strategy. Futur Gener Comput Syst 2023(142):131–149. https://doi.org/10.1016/j.future.2022.12.039

Piao MJ, Zhang T (2020) New Algorithm of Multi-Strategy Channel Allocation for Edge Computing[J]. AEUE - International Journal of Electronics and Communications 126(11):1–15. https://doi.org/10.1016/j.aeue.2020.153372

Ritchie H, Roser M. (2018) Urbanization[J]. Our world in data

Skouby KE, Lynggaard P. (2014) Smart home and smart city solutions enabled by 5G, IoT, AAI and CoT services[C] 2014 International Conference on Contemporary Computing and Informatics (IC3I). IEEE, 874-878

Srinivasa Rao G, Charan Arur P (2022) Cloud Computing and its Service Oriented Mechanism. Akinik Publications, New Delhi

Sun J, Feng B, Xu W. (2004) Particle swarm optimization with particles having quantum behavior[C] Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753). IEEE, 1: 325-331

Truong TP, Tran AT, Masood A, et al. (2020) Delay-Sensitive Task Offloading for Internet of Things in Nonorthogonal Multiple Access MEC Networks[C]//2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 597-599

Van Den Bergh F. (2007) An analysis of particle swarm optimizers[D]. University of Pretoria

WANG JX, Hong-rui F (2020) New Method of Traffic Flow Forecasting Based on Quantum Particle Swarm Optimization Strategy for Intelligent Transportation System[J]. Int J Commun Syst 33(10):1–13. https://doi.org/10.1002/dac.4647

Wang Y, Lang P, Tian D et al (2020) A game-based computation offloading method in vehicular multiaccess edge computing networks[J]. IEEE Internet Things J 7(6):4987–4996

Wang X, Song XD (2014) A Novel Approach to Mapped Correlation of ID for RFID Anti-collision[J]. IEEE Trans Serv Comput 7(4):741–748. https://doi.org/10.1109/TSC.2014.2370642

Wang S, ZHANG J, Zhu HL (2023) A Content Distribution Method of Internet of Vehicles Based on Edge Cache and Immune Cloning Strategy. Ad Hoc Netw 138(2023):1–12. https://doi.org/10.1016/j.adhoc.2022.103012

Wang JX, Zhang J (2022) New Method of Fuzzy Mutli-criteria Routing in Vehicle Ad-Hoc Network. IEEE Transactions on Computational Social Systems 6:1–15. https://doi.org/10.1109/TCSS.2022.3193739

Wang S, Zhang X, Zhang Y et al (2017) A survey on mobile edge networks: Convergence of computing, caching and communications[J]. Ieee Access 5:6757–6779

Wang WJ, Zhang J, Zhang T (2023) Novel Edge Caching Approach Based on Multi-agent Deep Reinforcement Learning for Internet of Vehicles. IEEE Trans Intell Transp Syst 24(6):1–16. https://doi.org/10.1109/TITS.2023.3264553

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization[J]. IEEE Trans Evol Comput 1(1):67–82

Wu J, Cao Z, Zhang Y, et al. (2019) Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC[C] 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 959-962

Yang Y, Chen X, Chen Y, et al. (2019) Green-oriented offloading and resource allocation by reinforcement learning in MEC[C] 2019 IEEE International Conference on Smart Internet of Things (SmartIoT). IEEE, 378-382

Yy CUI (2020) Novel Method of Mobile Edge Computation Offloading Based on Evolutionary Game Strategy for IoT Devices[J]. AEU-International Journal of Electronics and Communications 118(5):1–13. https://doi.org/10.1016/j.aeue.2020.153134

Zhang T (2018) Novel Optimized Link State Routing Protocol Based on Quantum Genetic Strategy for Mobile Learning[J]. J Netw Comput Appl 2018(122):37–49. https://doi.org/10.1016/j.jnca.2018.07.018

Zhang T (2019) Novel Self-Adaptive Routing Service Algorithm for Application of VANET[J]. Appl Intell 49(5):1866–1879. https://doi.org/10.1007/s10489-018-1368-y

Zhang T (2021) A New Method of Data Missing Estimation with FNN-Based Tensor Heterogeneous Ensemble Learning for Internet of Vehicle[J]. Neurocomputing 420(1):98–110. https://doi.org/10.1016/j.neucom.2020.09.042

Zhang DG, Li G, Zheng K (2014) An energy-balanced routing method based on forward-aware factor for Wireless Sensor Network[J]. IEEE Trans Industr Inf 10(1):766–773

Zhang Y, Liu JH, Wang CY et al (2020) Decomposable Intelligence on Cloud-Edge IoT Framework for Live Video Analytics[J]. IEEE Internet Things J 7(9):8860–8873

Zhang J, Zhao X. (2020) An Overview of User-Oriented Computation Offloading in Mobile Edge Computing[C] 2020 IEEE World Congress on Services (SERVICES). IEEE, 75-76

Zhang K, Zhu Y, Leng S, et al. (2019) Deep Learning Empowered Task Offloading for Mobile Edge Computing in Urban Informatics[J]. IEEE Internet of Things Journal, 1-1

Zhao J, Li Q, Gong Y et al (2019) Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks[J]. IEEE Trans Veh Technol 68(8):7944–7956

Zhou XM. Research on offloading strategy in energy-saving mobile edge computing system [D]. Beijing University of Posts and Telecommunications