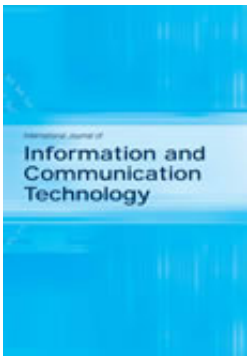
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

A meta-heuristic optimisation algorithm based method for scheduling edge computing resources

Yujie Li, Yaoyao Xu, Fangfang Cao, Xiang He

Article History:

Received:	28 September 2024
Last revised:	12 October 2024
Accepted:	12 October 2024
Published online:	28 November 2024

A meta-heuristic optimisation algorithm based method for scheduling edge computing resources

Yujie Li, Yaoyao Xu and Fangfang Cao

Nanchang Institute of Science and Technology,
Nanchang 330108, China
Email: liyujie@stu.ncpu.edu.cn
Email: xuyaoyao@stu.ncpu.edu.cn
Email: caofangfang@stu.ncpu.edu.cn

Xiang He*

Jiangxi Flight University,
Nanchang 330088, China
Email: 1992a11b11c@163.com
*Corresponding author

Abstract: Edge computing provides a viable solution to the lack of computing power in smart mobile devices (SMDs) and has received much attention in the industry. However, transferring part of the computation tasks from SMDs to edge servers brings additional transmission energy and server computation energy. To reduce energy consumption, this article suggests an edge calculating resource scheduling approach relied on meta-heuristic improvement algorithm. Firstly, the resource scheduling system model is constructed, and the SMD selects the most suitable edge server (ES) to help itself to complete the computational tasks according to the computational resources of the ES. Then the total energy consumption objective function is suggested, and an enhanced particle swarm optimisation (EPSO) algorithm is used to address this objective function. The experimental outcome indicates that when the number of SMDs is 10, the energy consumption value of the suggested method is 9.25W, which is reduced by 10%–55% compared to the other four methods.

Keywords: resource scheduling; metaheuristic optimisation algorithm; particle swarm optimisation algorithm; power law distribution function; genetic algorithm; smart mobile devices; SMDs; enhanced particle swarm optimisation; EPSO.

Reference to this paper should be made as follows: Li, Y., Xu, Y., Cao, F. and He, X. (2024) 'A meta-heuristic optimisation algorithm based method for scheduling edge computing resources', *Int. J. Information and Communication Technology*, Vol. 25, No. 9, pp.88–103.

Biographical notes: Yujie Li has graduated from the Segi University of Malaysia with a Doctorate in Management. Currently, she is a teacher of Economics and Management School of Nanchang Institute of Science and Technology. Her main research direction includes e-commerce, international trade, business administration, and business English.

Yaoyao Xu has gained her Masters degree at the Guangxi Normal University. Currently, she is a teacher of the Nanchang Institute of Science and Technology. Her research directions are in human resource management, and organisational behaviour.

Fangfang Cao has graduated from the Jiangxi Science and Technology Normal University with a Bachelors degree. Currently, she is a teacher of Nanchang Institute of Science and Technology. Her research direction is in e-commerce.

Xiang He has graduated from the Jiangxi Agricultural University with Doctors degree. Currently, he is a teacher in the Jiangxi Flight University. His research directions are in digital economy, and financial engineering.

1 Introduction

With the growth of IoT applications, the emergence of various smart mobile devices (SMDs) has led to an explosive growth in data size. Traditional cloud computing architectures face problems such as bandwidth bottlenecks, data transfer delays, and security when dealing with large amounts of data (Kumar and Goudar, 2012). To solve these problems, mobile edge computing (Mach and Becvar, 2017), as a new computing paradigm, decentralises computational power from the cloud calculating core to the edge side close to the user, so that the computational tasks are shifted to be processed on edge nodes close to the mobile devices. Under the distributed edge computing model, edge nodes can collaborate with each other to share computing resources and achieve global scheduling of resources, changing the drawbacks brought by the centralisation of resources in cloud computing centres (Lin et al., 2020). However, there are a series of new problems and challenges under the distributed edge computing scenario, and how to maximise the scheduling of edge computing resources to enhance the performance and efficiency of the overall system has become an urgent issue to be addressed.

Liu et al. (2021a) proposed a service operation chain disposition and resource governance mechanism to deploy service function chains and control resources relied on a game theory method to achieve efficient completion of service function chain deployment in edge computing networks, but with high computational energy consumption. Huang et al. (2019) proposed a mobile edge computing offloading system for multi-user cooperation, where SMDs can not only unlade calculating tasks to edge nodes, but can also choose to offload to the other SMDs. There are also research works that optimise both latency and energy consumption, but increase the network load and latency (Zhang et al., 2018), so there is a need to find a balance between the two making the edge computing system achieve an optimal result. For example, Li et al. (2020) studied the problem of minimising the energy expenditure of all SMDs and their delays in an edge computing system with multiple devices, for which a computational optimisation algorithm based on a convex function is proposed and is able to minimise the energy expenditure of all SMDs.

In most research works, the resource scheduling problem for edge computing can be transformed into an integer nonlinear planning problem, and the traditional mathematical optimisation algorithms are unable to gain the optimal solution in a rational time, while the metaheuristic optimisation methods have been broadly adopted in edge calculating

research by virtue of their powerful global optimisation capability and faster convergence speed (Liu et al., 2021b). Chakraborty and Mazumdar (2022) used genetic algorithm (GA) for the energy optimisation issue of edge calculating model consisting of SMD and ES, but due to the early maturity and high complexity of GA, it resulted in high energy consumption of the system. Liu et al. (2023) suggested an improvement approach relied on ant colony (ACO) approach to conjointly improve the work unlading determination and resource allotment issues in edge calculating to minimise the entire consumption of all mobile users while satisfying the delay constraint. Yadav et al. (2020) and others designed an energy-effective dynamic computational unlading and resource allocation mechanism and used a simulated annealing algorithm to solve the resource allocation issue for the goal of gaining a balance between energy consumption and latency. Owing to the features of particle swarm optimisation (PSO) algorithm, for example quick convergence speed and few setup parameters, researchers have applied it to edge computing resource scheduling to improve the system performance (Zhang et al., 2020). Alfakih et al. (2021) decomposed the probabilistic task offloading issue into multiple unrestrained subproblems, and used PSO to address each of them to gain the ideal solution. Chafi et al. (2023) offered an adaptive discrete PSO approach with GA, which introduces the crossover and mutation operations of GA into particle swarm algorithm, avoiding the premature convergence of the traditional PSO algorithm, and effectively reducing the data transmission time.

However, all of the above mentioned studies only investigate how to reduce the energy consumption of SMDs and network transmission, and seldom take the energy consumption of the ES side as an optimisation goal. Inspired by this, for the problem of optimising resource allocation and reducing device energy consumption, this paper proposes a meta-heuristic optimisation algorithm based edge computing resource scheduling method, which has the following significant innovations and contributions.

1 An edge computing resource scheduling system model is constructed. the SMD can choose the most suitable ES to help it complete its calculating tasks in terms of the demands of its calculating tasks and the computing resources of ES.

2 The model takes the reduction of server energy consumption as the optimisation objective, and takes ES storage, maximum energy consumption, etc. as constraints, and establishes an objective function of total energy consumption related to mobile device computation energy, data transmission energy and ES computation energy, which is solved by using the enhanced particle swarm optimisation (EPSO) algorithm.

3 The PSO is improved by introducing a power-law distribution function relied on the linear decreasing weights of PSO to enhance the global seek capability in the early stage of PSO and focus more on the local search in the later stage. With this flexible weight adjustment, the improved method can effectively enhance the performance of the PSO algorithm.

4 Intending to the PSO algorithm, which has the issue of uneven number of allocated servers when solving, the cross operation of GA is combined so that the EPSO can be better used in edge computing resource scheduling model. The experimental results imply that the offered method has lower average energy consumption and higher resource utilisation, proving the effectiveness and feasibility of the method.

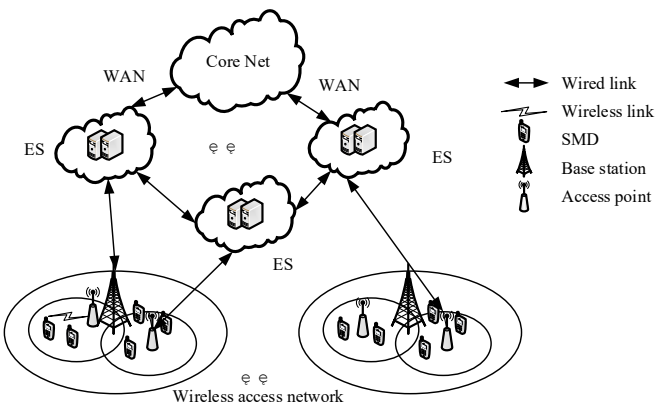
2 Relevant technologies

2.1 Edge computing technology

Edge calculating related to a public platform that combines network, calculating, and other application mental abilities at the edge of the network close to the device or data source to offer edge intelligent services in close proximity to satisfy the critical needs of industry digitisation. Compared with the centralised processing of traditional cloud computing centres, edge computing technology is with low deployment cost, low network latency and high data security.

The network system framework of edge calculating is indicated in Figure 1, the bottom layer is some SMDs, such as computers, smart phones, wearable devices, etc. which gather a large quantity of production and life data; while the middle layer generally deploys ES (such as workstations, communication base stations, etc.) and wireless access gateways, the terminal devices are accessed to the wireless network through the gateways of the layer, and the generated data are uploaded to the ES for processing; the top layer is the cloud computing platform, which can offload the data to the cloud computing platform for processing when ES is overloaded or can meet the delay requirements. The top layer is the cloud computing platform, which can offload the data to the cloud computing platform for processing when the ES is overloaded or can meet the delay requirements.

Figure 1 Network system architecture for edge computing



Energy consumption is a common optimisation metric for edge computing systems. During transmission, the transmission delay t_{trans} of a single task is V/R and the energy consumption E_k is $t_{trans} \times p_k$, where R denotes the transmission distance, p_k denotes the transmission power of the task. During the completion of a task on the server, the completion time t_e of a single task can be calculated from equation (1).

$$t_e = \frac{V_k \times S_k}{f_{pro}} \tag{1}$$

where V_k denotes the transmission rate, S_k denotes the transmission time, and f_{pro} denotes the CPU cycle frequency of the ES.

2.2 Meta-heuristic optimisation algorithm

In edge computing research, resource scheduling problems can be transformed into integer nonlinear planning problems, and meta-heuristic optimisation algorithms have greater advantages than traditional mathematical optimisation algorithms due to their global search and optimisation capabilities. Commonly used meta-heuristic optimisation algorithms include GA, PSO, ACO, etc. (Vinod Chandra and Anand, 2022). Among them, PSO has the characteristics of fast convergence speed and few setup parameters, which is the first option for addressing some practical engineering issues (Gad, 2022).

PSO simulates the process of discovery of food location through exploration and cooperation between flocks of birds in nature, where each particle maintains velocity $v = (v_i^1, v_i^2 ..., v_i^D)$ and position $x = (x_i^1, x_i^2 ..., x_i^D)$, in a D-dimensional seek space, where i denotes the amount of the particle. The velocity vector of a particle determines the direction and rate of motion of the particle, which changes the position information of the particle, while the position vector represents the position of the particle in the seek space, which can estimate the current fitness value of the particle.

The PSO algorithm also requires each particle to maintain a position vector of historical optimal adaptation values $pBest_i$, and a global optimal position $gBest_i$. During the evolutionary process, the equations for updating the velocity and position information of the i^{th} particle are shown in equation (2) and equation (3), respectively.

$$v_i^d = uv_i^d + a_1 r_1 \left(pBest_i - x_i^d \right) + a_2 r_2 \left(gbest^d - x_i^d \right) \tag{2}$$

$$x_i^d = x_i^d + v_i^d \tag{3}$$

where u is the inertia weight, which commonly takes the value in the range [0, 1]. a_1 and a_2 are acceleration factors, while r_1 and r_2 are random numbers generated in the interval [0,1].

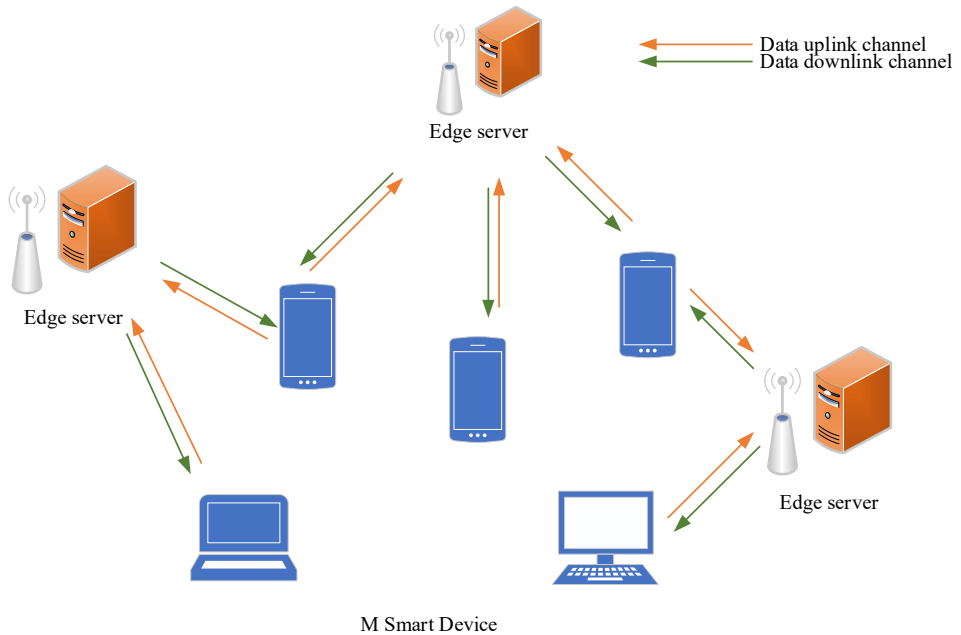
Although PSO has the features of quick convergence speed and few parameters, it is also with the issues of easy to fall into the local optimum and low improvement accuracy, so the simple PSO algorithm can not be directly used in the optimisation of the edge computing problem, this article will enhance the PSO algorithm, for the goal of improving the optimisation performance of the PSO algorithm.

3 Edge computing resource scheduling modelling

3.1 System model

Considering the complexity of the edge network deployment situation, the existing single server resource scheduling method cannot satisfy today's huge and complex network system, this paper designs a multi-server edge computing resource scheduling model, as implied in Figure 2.

Figure 2 Edge computing resource scheduling model (see online version for colours)



Assuming that there are M SMDs and N ESs in the edge computing network, these M SMDs need to deal with multiple computation tasks, and each task has a different amount of data, so each device can send its own computation data and receive the computation results from the ESs through the upstream and downstream channels through the algorithm. The scheduling optimisation algorithm determines the most suitable server and resource allocation scheme for each device to offload, based on the hardware of the SMDs and the different devices among the servers, as well as the computational task requirements of the program and the relevant constraints, so as to consume the least amount of computational energy of the SMDs, the transmission energy, and the computational energy of the ES, provided that the constraints are satisfied.

3.2 Optimised modelling of system energy consumption

For the goal of achieving the correction of erroneous movements in continuous sports dance, the acquired images are first pre-processed and the template combination equations for binocular stereo vision imaging of the images are calculated.

The total energy consumption of SMDs and multiple ESs is composed of three aspects: the local computational energy consumption E_m^L of each SMD, the computational energy expenditure $E_m^{B_i}$ produced through its offloaded edge servers (ESs) B_i to process the computational data uploaded by SMDs, and the transmittance energy expenditure $\mu_m^{B_i}$ produced through the process of transmitting the computational tasks to B_i, as shown below.

$$\varphi = \sum_{m=1}^{M} \left(E_m^L + E_m^{B_i} + \mu_m^{B_i} \right) \tag{4}$$

The local calculative energy expenditure E_m^L of each SMD is $P_m^L \cdot t_m^L$, where P_m^L denotes the computational power of a single SMD and t_m^L denotes the working hours of a single SMD. Assuming that each device needs to handle K computational tasks, the formula for calculating the working hours of SMDs can be obtained as follows.

$$t_m^L = \frac{\sum_{i=1}^{\rho_m K} \beta G_i^m}{h_m} \tag{5}$$

where G_i^m denotes the amount of data to be processed by the j^{th} computational task of the m^{th} SMD, β denotes the complexity of all computational data, ρ_m denotes the ratio of the amount of local computational tasks to the total amount of computations of the m^{th} SMD, and h_m denotes the computational speed of the m^{th} SMD.

The maximum amount of energy that each SMD can use for the calculation is denoted by $E_{\max}^L$. Therefore, the energy consumed through all SMDs in processing the data needs to satisfy the following equation.

$$k_L \left(h_m\right)^2 \sum_{i}^{\rho_m K} \beta G_i^m \le E_{\max}^L \tag{6}$$

Similar to the computational energy expenditure of the SMD locally, the computational energy expenditure of the ES is calculated by the product of its power P and computation time t_m^B. Since the m^{th} SMD will keep the $\rho_m K$-bar computational tasks to be computed at the local device, the computational tasks that B_i needs to process are from subscripts $\rho_m K$+1 to K.

$$E_m^{B_i} = k_{B_i} \left(h_{B_i}\right)^2 \sum_{i=\rho_m K+1}^{K} \beta G_i^m \tag{7}$$

where k_{B_i} is a constant relevant with the ES chip framework, h_{B_i} denotes the ES' computational speed, and $E_{\max}^{B_i}$ is used to denote the maximum energy expenditure restrict of the i^{th} ES, thus, the maximum energy expenditure required by each ES to process the computational tasks has to satisfy the following equation.

$$\sum_{m=1}^{M_{B_i}} \left(k_{B_i} \left(h_{B_i}\right)^2 \sum_{i=\rho_m K+1}^{K} \beta G_j^m \right) \le E_{\max}^{B_i} \tag{8}$$

where M_{B_i} denotes the number of SMDs that offload computational tasks to B_i.

Assuming that the communication links between different servers do not interfere with each other, the data transmittance energy expenditure $\mu_m^{B_i}$ produced through the m^{th} SMD in sending data to B_i is computed as bellow.

$$\mu_m^{B_i} = \left(P_L^I + \lambda_m k_L^T P_m^T \right) t_U + P_L^C t_C \tag{9}$$

where P_L^I denotes the idle power, k_L^T denotes the enhancement factor of the conveyed data, P_m^T denotes the power of the transmitted data, P_L^C denotes the power of the downloaded data, and L denotes the relevant parameters. λ_m denotes the ratio of the bandwidth engaged in the m^{th} SMD in the transmittance channel, and the value ranges from [0, 1]. $t_U^{B_i}$ is the time required for uploading the data.

Most applications on SMDs are with related latency demands, and the maximum accomplishment time consists of the following aspects. The first aspect is that the time t_m^L calculated locally by the SMD, as shown in equation (5). On the other hand, it is the time for SMD to offload part of the data to B_i to complete the computation t_{B_i}. Since these two aspects of the computation are carried out simultaneously, the final accomplishment time of the computation task requires to satisfy the following formula.

$$\max\left\{t_m^L, t_{B_i}\right\} \le L_{\max} \tag{10}$$

where $L_{\max}$ denotes the maximum completion time of SMD processing data. The time t_{B_i} required for B_i to complete the calculation task transferred from the m^{th} SMD is calculated as shown in equation (11), which includes the data uploading time $t_U^{B_i}$, the data downloading time $t_U^{B_i}$, and the time $t_m^{B_i}$ for the server to process the calculation task from the SMD.

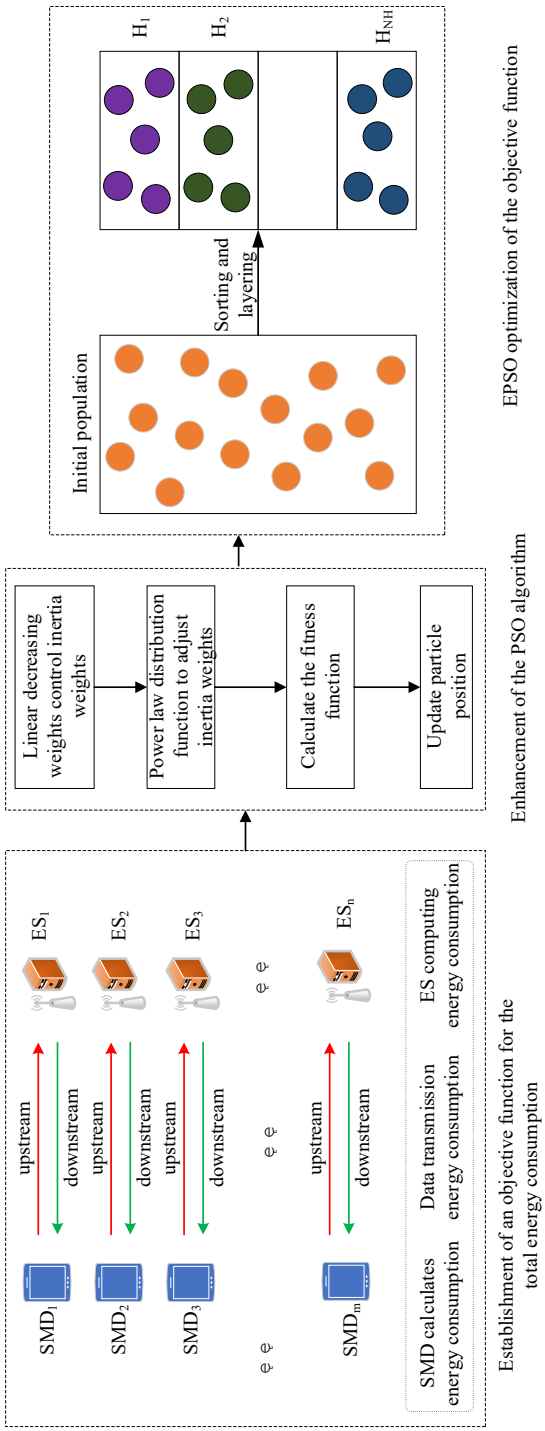
$$t_{B_i} = t_U^{B_i} + t_D^{B_i} + t_m^{B_i} \tag{11}$$

4 Edge computing resource scheduling method based on meta-heuristic optimisation algorithm

4.1 Establishment of the objective function

Based on the resource scheduling model established in the previous section, the total energy consumption objective function with various performance constraints is established, and the optimised PSO is used to quickly obtain the optimal solution that satisfies the constraints and has lower energy consumption. Firstly, a power-law distribution function is introduced on the basis of linearly decreasing weights of PSO to improve the performance of PSO. Then the combination of GA crossover operation compensates the shortcoming of PSO algorithm in optimising integer discrete variables, which makes EPSO more suitable for solving energy consumption optimisation issues. The flow of EPSO optimised edge computing resource scheduling is indicated in Figure 3.

Figure 3 Architecture of EPSO for optimising edge computing resource scheduling (see online version for colours)



The improvement goal in this chapter is to minimise the sum of the computational energy expenditure of the whole devices, the computational task transfer energy expenditure, and the computational energy expenditure of all servers, while satisfying various constraints, such as latency constraints, server storage constraints, and so on. To optimise the entire energy expenditure and various constraints at the same time, all the constraints are converted into a penalty operation, the smaller the value of the penalty function, the better the optimisation result, so the optimisation objective of the final algorithm is to minimise the whole energy expenditure and the value of the penalty operation, and the objective operation is established as follows.

$$\underset{x}{Min}\{\varphi' = \varphi + N\varepsilon\} \tag{12}$$

where ε stands for the sum of whole penalty operation values, N is a natural number with a large value, φ represents the penalty operation's weight, x stands for the approach of the issue. ε is computed as bellow.

$$\varepsilon = \sum\nolimits_{p=1}^{5} \left(\max\{0, h_p(x)\}\right)^{y_1} + \left|h_q(x)\right|^{y_2} \tag{13}$$

where y_1 and y_2 are exponential invariant values, $f_p(x)$ is the penalty operation for flexible values, p = 1, 2, …, 5, f_q denote the penalty function for equality constraints.

$$f_1(x) = k_L (h_m)^2 \sum\nolimits_{i}^{\rho_m K} \beta G_j^m - E_{\max}^L \tag{14}$$

$$f_2(x) = \left\{\sum\nolimits_{m=1}^{M_{B_i}} \left(k_{B_i} (f_{B_i})^2 \sum\nolimits_{i=\rho_m K+1}^{K} \beta G_i^m\right\} - E_{\max}^B \right. \tag{15}$$

$$f_3(x) = \left\{\sum\nolimits_{m=1}^{M_{B_i}} \sum\nolimits_{i=\rho_m K+1}^{K} \beta G_j^m\right\} - A_{\max} \tag{16}$$

$$f_4(x) = \left\{\sum\nolimits_{m=1}^{M_{B_i}} \sum\nolimits_{i=\rho_m K+1}^{K} \zeta G_j^m\right\} - I_{\max} \tag{17}$$

$$f_5(x) = \max\{t_m^L, t_{B_i}\} - L_{\max} \tag{18}$$

Equations (14) to equation (18) are various types of penalty functions transformed by the maximum energy expenditure constraints of SMDs, maximum energy expenditure constraints of servers, CPU computation cycle constraints of servers, storage constraints of servers, latency constraints, and bandwidth constraints, respectively, with $A_{\max}$ being the maximum amount of ES's CPU cycles, $I_{\max}$ being the maximum memory storage space of the ES, and ζ denoting the ES's memory space.

4.2 Improvement of particle swarm optimisation algorithm

Intending to the issue that traditional PSO algorithms tend to fall into local optimality and have limited search performance, this paper combines the linear decreasing weights (Mazahery et al., 2013) and the power law distribution function (Mitzenmacher, 2004) to calculate the inertia weights in PSO algorithms, which is firstly used to control the overall trend of the inertia weights and then combined with the power law distribution function to adjust the specific weights of each particle. A power law distribution function

is then used to adjust the specific weights of each particle. In this way, we can balance the effect of global seek and local seek, and improve the performance of PSO algorithm. The steps in detail are as bellow.

1 Initialisation, setting the maximum and minimum inertia weights $u_{\max}$ and $u_{\min}$, and the maximum amount of iterations $iter_{\max}$.

2 Linear decreasing weight calculation, in each iteration, the current base inertia weight u_{base} is calculated according to the linear decreasing method.

$$u_{base} = u_{\max} - (u_{\max} - u_{\min}) * (iter / iter_{\max}) \tag{19}$$

3 The power law distribution function is calculated and for each particle, its weight factor u_{factor} is calculated according to the fitness ordering and the power law distribution function.

$$u_{factor} = (rank / swarm_{size})^{\wedge} p \tag{20}$$

4 To update the position of the particles, multiply u_{base} and u_{factor} to get the inertia weight u of each particle as shown in equation (21), and then substitute u into equation (2) with the acceleration constants a_1 and a_2 to update the present position $x_i(t + 1)$ of the particles.

$$u = (u_{\max} - (u_{\max} - u_{\min}) * (iter / iter_{\max})) * (rank / swarm_{size})^{\wedge} p \tag{21}$$

where *rank* is the rank of the particle in the population, $swarm_{size}$ is the size of the population, p is a positive real number, and plays a critical role in the power-law distribution function, affecting the weight factor u_{factor} of each particle.

4.3 Edge computing resource scheduling based on EPSO algorithm

The solution x of EPSO algorithm represents the current resource allocation of all SMDS, which is composed of four parameters related to SMD: calculating speed h_m, power expenditure of downloaded data P_m^T, calculating data offloading percentage ρ_m, remaining network bandwidth ratio λ_m and uninstalled server location B_m. Assume that the number of SMDS is M, the length of x is 5M, and M shitter percentage μ_m is stored, the parameter b_m is stored in the last M positions. The encoding mode of the final individual code x is as bellow.

$$x = (h_1, h_2, \ldots h_M, P_1^T, \ldots, P_M^T, \mu_1, \ldots, \mu_M, \lambda_1 \ldots, \lambda_M, b_1 \ldots, b_M) \tag{22}$$

Each particle maintains a velocity v, which represents the range of resource allocation changes. The length of v is $4M$, and different velocity values corresponding to the first four parameters of each SMD are stored respectively. The variable x output of the final algorithm is the resource allocation method and server allocation scheme with the lowest energy consumption.

After x is encoded, edge computing resources are scheduled. First, the particle is initialised, randomly generating a floating point number in the different value ranges of five parameters, corresponding to the values of $h_m \in [0, h_{\max}]$, $P_m^T \in [0, P_{\max}]$, $\mu_m \in [0, 1]$, $\lambda_m \in [0, 1]$, and $b_m \in [1, B$. After initialisation, λ_m is converted to the true

bandwidth ratio τ_m^b based on the server subscript b to which each SMD is assigned, as shown below.

$$\tau_m^b = \begin{cases} \left(1 - \sum_{i=1}^{M_B - 1} \tau_i^b\right) \cdot \lambda_m & 1 \le m < M_B \\ 1 - \sum_{i=1}^{M_B - 1} \tau_i^b & m = M_B \end{cases} \tag{23}$$

where M_B is the amount of SMDS allocated to server b.

After conversion, using the idea of power law distribution function, the population was first sorted according to the fitness value and stratified. After initialisation, the population is sorted from smallest to largest according to the value of the objective function, and then each Q_S particle is divided into a layer in order, in which the first Q_S particles are divided into the first layer Q_1, and the last Q_S particles are divided into the last layer Q_N.

Then, the individual updating operation is carried out. Assuming that the j^{th} particles x_{ij}, $1 < i \leqslant Q_N$ and $1 < j \leqslant Q_S$ located in the i layer, the two particles x_{k_1,l_1} and x_{k_2,l_2} with a higher level will be selected as the template for the evolution of x_{ij} at first, where k_1 and k_2 are randomly generated within the range [1, Q_S]. l_1 and l_2 are generated randomly in the range of [1, i), and the update operation is required after the selection of $l_1 < l_2$ evolution template, and the final optimisation result $x_{i,j}^k$ is output.

$$v_i^k = ux_{i,j}^k + a_1\left(x_{k_1,l_1}^k - x_{i,j}^k\right) + \varphi a_2\left(x_{k_2,l_2}^k - x_{i,j}^k\right), k \in [0, 4M) \tag{24}$$

$$x_{i,j}^k = v_i^k + x_{i,j}^k k \in [0, 4M) \tag{25}$$

where u represents dynamic weight. As shown in equation (21), subscripts 0 to 4M-1 in x and v store four continuous variable optimisation parameters h_m, P_m^T, μ_m and λ_m of SMD. As for the positional parameter b_m, it is an integer variable. If the update operation is carried out according to the above method, the update result is a floating point number rather than an integer. If the update result is rounded by rounding, the final server allocation result set and a server cannot play a role in reducing energy consumption. Therefore, for the update of b_m, the cross operation of two evolutionary template x_{k_1,l_1} and x_{k_2,l_2} is carried out by referring to the cross operation of GA, and the corresponding parameter position is assigned to x_{ij}, so as to complete the update.

5 Experimental results and analyses

To evaluate the effects of the experiment, GA (Chakraborty and Mazumdar, 2022), ACO (Liu et al., 2023) and PSO, which are commonly used in edge computing resource scheduling methods, are selected respectively. 2021), GAPSO (Chafi et al., 2023) and the proposed method EPSO were compared. The simulation experiment in this paper is implemented using Python 3.7 under the environment of Windows 10 operating system with 16 GB memory, Inter Core i5-4460 processor and 3.2 GHz frequency, and simulates ES and SMD by creating a virtual machine. Simulation using the publicly available real dataset MNIST, which contains 60,000 training data and 10,000 test data. The dataset size is power-law distribution, and each device has a dataset range of 500 to 5,000. In this

paper, the dataset is divided into training set and test set in the ratio of 7:3. The simulation parameters of this experiment are indicated in Table 1.

Table 1 Parameters of the simulation experiment

Parameter	*Value*	*Parameter*	*Value*
Total system bandwidth	20 MHZ	CPU frequency of ES	10 GHz
Noise power	–120 dBm	Systematic training rounds	100
Transmission power	20 dBm	Batch size	128
CPU frequency of SMD	2 GHz	Training learning rate	0.01

The data in Table 2 are the comparison of ES average energy consumption value (AEV), average penalty value (APV) and feasible solution probability (FSP) of the five methods running independently for 100 times under different number of SMDS (Aslanpour et al., 2020). Each method stops improvement when the objective function evaluation reaches 2,000 times, and FSP is used to indicate that each method gets a feasible solution value of 0 in 100 independent runs.

Table 2 Improvement outcome of five methods with various numbers of SMDs

Number of SMDs	*Indicator*	*GA*	*ACO*	*PSO*	*GAPSO*	*EPSO*
$M = 5$	AEV	4.5521	4.8357	4.4726	4.2591	4.1327
	APV	0	0	0	0	0
	FSP	100	100	100	100	100
$M = 10$	AEV	11.9253	13.6951	10.8197	9.8152	8.6288
	APV	1.28	2.49	0.72	0	0
	FSP	48	35	72	89	100
$M = 15$	AEV	19.7425	23.1589	18.9112	15.6286	13.1473
	APV	2.53	4.36	1.19	0.08	0
	FSP	35	13	65	77	100

From Table 2, the energy consumption improvement outcome of EPSO is better than another four methods, and the feasible solution probability of EPSO can reach 100% under all different SMD quantities. When the amount of SMDS is 5, each method can find possible approaches. The AEV of GAPSO is close to that of EPSO, but EPSO has the lowest energy consumption optimisation result. When the number of SMDS increases, such as $M = 10$ and $M = 15$, the possible approaches' probabilities of GA and ACO are both lower than 50%, implying that they are no longer applicable in the improvement model with a large amount of SMDS. The feasible solution probabilities of PSO and GAPSO are lower than 90%, while the possible approaches' probabilities of EPSO are still 100%.

Figure 4 implies the distribution of energy consumption optimisation results for 100 independent runs of the five methods when the number of SMDS is 10. As can be seen from the figure, the energy consumption of EPSO is 9.25 W, which is reduced by 53%, 42%, 39% and 17% respectively compared with the other four methods. The optimisation results are the best and the distribution is the most concentrated, indicating that the performance of EPSO is stable, and the average and median energy consumption are lower than that of the other four algorithms. The outcome of PSO and GAPSO were

similar. However, the distribution of optimisation outcome of GA and ACO algorithms is poor, and the distribution range of optimisation outcome of ACO in different numbers of mobile devices is larger than that of other algorithms, which indicates that the optimisation results of ACO are unstable.

Figure 4 Comparison of optimised distribution results for different methods (see online version for colours)

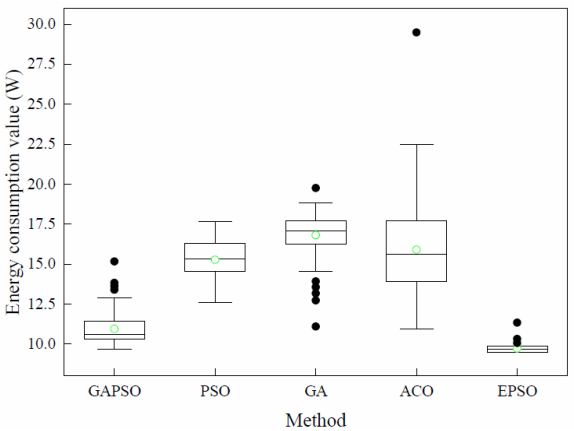
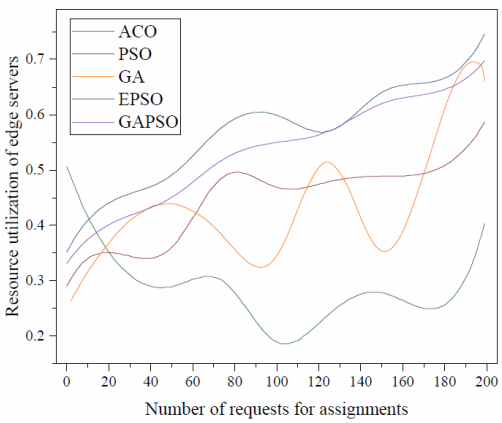


Figure 5 Comparison of resource utilisation rates of different methods (see online version for colours)



The comparison of resource utilisation of ES under different request tasks is shown in Figure 5. When the number of task requests is 200, the resource utilisation of GA, ACO, PSO, GAPSO and EPSO are 58%, 40%, 67%, 69% and 78%, respectively. Since ACO allocates ES resources based on the sequential search of incoming request tasks, it will lead to strong randomness and irregularity of ES resource utilisation. Like ACO, GA will also cause the imbalance of ES resource utilisation, but the resource utilisation rate of GA is better than ACO in most cases. Since PSO can perform task scheduling and ES resource management, PSO can achieve high resource utilisation while maintaining

relatively stable fluctuation amplitude. Although EPSO algorithm's fluctuation range is second only to GAPSO algorithm, its resource utilisation rate is the highest compared with other algorithms. To sum up, EPSO can efficiently and adaptively allocate computing resources of ES, and effectively optimise resource utilisation of ES.

6 Conclusions

Due to issues such as slow CPU operation speed and limited battery capacity, SMD itself is unable to handle applications with high computing demands, necessitating the use of edge computing technology to alleviate the hardware requirements on mobile devices. However, offloading certain computational tasks from SMD to ES results in additional power consumption. To address these challenges, this paper designs an edge computing resource scheduling method based on meta-heuristic optimisation algorithms, offering the following advantages.

1 Establish edge computing resource scheduling system model, and model SMD calculation energy consumption, transmission energy consumption and ES calculation energy consumption according to different equipment conditions among servers.

2 The objective function of total energy consumption optimisation of the system was established, and the computational speed and data transmission power consumption were taken as constraint conditions. Since the total energy consumption optimisation problem was a complex nonlinear programming problem, PSO was adopted for optimisation.

3 A power law distribution function is introduced to optimise the PSO algorithm, which gradually decreases according to the nonlinear law of the power law function, thus enhancing the global and local search capabilities of PSO.

4 Intending to the issue that PSO cannot achieve better energy consumption optimisation results when solving the objective function, the proposed method combines the crossover operation of GA to make up for the shortcoming of PSO algorithm in optimising integer discrete variables. It is also proved through comparative experiments that the suggested method obtains solutions with lower energy consumption than other methods.

In the future research, this article will design novel seek patterns for use in methods to deal with large-scale issues, enhancing the capability of methods to arrive at attainable solutions when the amount of SMDs and ESs is high.

Acknowledgements

This work is supported by the Science and Technology Research Project of Jiangxi Provincial Department of Education in 2024 (GJJ2402804) entitled 'A meta-heuristic optimisation algorithm based method for scheduling edge computing resources'.

References

Alfakih, T., Hassan, M.M. and Al-Razgan, M. (2021) 'Multi-objective accelerated particle swarm optimization with dynamic programming technique for resource allocation in mobile edge computing', *IEEE Access*, Vol. 9, pp.167503–167520.

Aslanpour, M.S., Gill, S.S. and Toosi, A.N. (2020) 'Performance evaluation metrics for cloud, fog and edge computing: a review, taxonomy, benchmarks and standards for future research', *Internet of Things*, Vol. 12, p.100273.

Chafi, S-E., Balboul, Y., Fattah, M. et al. (2023) 'Enhancing resource allocation in edge and fog-cloud computing with genetic algorithm and particle swarm optimization', *Intelligent and Converged Networks*, Vol. 4, No. 4, pp.273–279.

Chakraborty, S. and Mazumdar, K. (2022) 'Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing', *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 4, pp.1552–1568.

Gad, A.G. (2022) 'Particle swarm optimization algorithm and its applications: a systematic review', *Archives of Computational Methods in Engineering*, Vol. 29, No. 5, pp.2531–2561.

Huang, P-Q., Wang, Y., Wang, K. et al. (2019) 'A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing', *IEEE Transactions on Cybernetics*, Vol. 50, No. 10, pp.4228–4241.

Kumar, S. and Goudar, R. (2012) 'Cloud computing-research issues, challenges, architecture, platforms and applications: a survey', *International Journal of Future Computer and Communication*, Vol. 1, No. 4, p.356.

Li, S., Sun, W., Sun, Y. et al. (2020) 'Energy-efficient task offloading using dynamic voltage scaling in mobile edge computing', *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 1, pp.588–598.

Lin, R., Zhou, Z., Luo, S. et al. (2020) 'Distributed optimization for computation offloading in edge computing', *IEEE Transactions on Wireless Communications*, Vol. 19, No. 12, pp.8179–8194.

Liu, J., Yang, P. and Chen, C. (2023) 'Intelligent energy-efficient scheduling with ant colony techniques for heterogeneous edge computing', *Journal of Parallel and Distributed Computing*, Vol. 172, pp.84–96.

Liu, J., Yang, T., Bai, J. et al. (2021a) 'Resource allocation and scheduling in the intelligent edge computing context', *Future Generation Computer Systems*, Vol. 121, pp.48–53.

Liu, Y., Shang, X. and Yang, Y. (2021b) 'Joint SFC deployment and resource management in heterogeneous edge for latency minimization', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 8, pp.2131–2143.

Mach, P. and Becvar, Z. (2017) 'Mobile edge computing: a survey on architecture and computation offloading', *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 3, pp.1628–1656.

Mazahery, A., Shabani, M.O., Alizadeh, M. et al. (2013) 'Concurrent fitness evaluations in searching for the optimal process conditions of Al matrix nanocomposites by linearly decreasing weight', *Journal of Composite Materials*, Vol. 47, No. 14, pp.1765–1772.

Mitzenmacher, M. (2004) 'A brief history of generative models for power law and lognormal distributions', *Internet Mathematics*, Vol. 1, No. 2, pp.226–251.

Vinod Chandra, S.S. and Anand, H.S. (2022) 'Nature inspired meta heuristic algorithms for optimization problems', *Computing*, Vol. 104, No. 2, pp.251–269.

Yadav, R., Zhang, W., Kaiwartya, O. et al. (2020) 'Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing', *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 12, pp.14198–14211.

Zhang, J., Hu, X., Ning, Z. et al. (2018) 'Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching', *IEEE Internet of Things Journal*, Vol. 6, No. 3, pp.4283–4294.

Zhang, Y., Liu, Y., Zhou, J. et al. (2020) 'Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing', *Future Generation Computer Systems*, Vol. 112, pp.148–161.