Edge Computing-Based Multitasking Strategies in Smart Grids

Han Zhao[1(✉)], Mengxuan Dai[2], Kaiwen Ji[3], Wenshan Wei[4], Xinghong Jiang[2], Yong Ma[2], Yunni Xia[5], and Bingbing He[6]

[1] School of Digital Industry, Jiangxi Normal University, Shangrao, China
zhaohan@jxnu.edu.cn
[2] School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
{dmx,jxh,may}@jxnu.edu.cn
[3] Jiangxi Institute of Economic Development, Jiangxi Normal University, Nanchang, China
54745348@qq.com
[4] School of Foreign Languages, Beihang University, Beijing, China
gram594@buaa.edu.cn
[5] School of Computer Science, Chongqing University, Chongqing 400030, China
[6] Hefei Youer Electronic Technology Co., Hefei, China
hebingbing@youotech.com

Abstract. In order to promote the digital development of the smart grid, power data needs to be analyzed and processed in real time, but because the number of power grid terminal devices is vast, the massive amount of data generated will incur additional network latency and communication costs if processed directly by cloud servers, and there is also a risk of data leakage. Therefore, power data is considered to be placed on edge servers for processing to overcome many problems in the current cloud computing paradigm for power systems, such as the inability to fully realize the requirements of high bandwidth and low latency. This paper thus proposes a multitask assignment strategy (MPA) for smart grid terminals based on edge computing. The method first classifies grid end tasks into different classes based on the size of data, security level, and computational workload; it then selects a suitable edge server for the smart grid terminal tasks using the particle swarm algorithm. The simulation results show the effectiveness of the method in this paper.

Keywords: edge computing · Smart grid · task assignment · particle swarm algorithm · multi-objective constraints

Y. Cao and X. Shao (Eds.): MONAMI 2022, LNICST 474, pp. 196–211, 2023.
https://doi.org/10.1007/978-3-031-32443-7_14

1 Introduction

The Internet of Things (IoT) [1] and artificial intelligence [2] are two cutting-edge information technologies that combine to create the smart grid, a modern grid system that plays an essential role in the energy infrastructure. To collect and upload various power data in real-time, massive terminal devices need to be deployed for the smart grid. The development of technologies such as cloud computing [3] and artificial intelligence has created favorable conditions for the intelligence of grid terminal devices. Efficient analysis and process of the tasks generated by these terminals are one of the challenges facing the smart grid. Cloud computing, a technology that provides on-demand computing resources, can be a solution. However, for end devices, cloud servers are usually in a relatively centralized location. Transferring tasks to cloud servers consumes a lot of bandwidth and other resources, resulting in high communication costs. In addition, in extreme cases, it will lead to serious delay and congestion, and even cause data loss and slow processing efficiency, which will not only bring losses to the power grid company, but also reduce system QoS [4]. In order to solve the above problems, the concept of edge computing [5] has been introduced in smart grids.

Edge computing is an open platform that integrates core functions such as networking, computing, storage, and applications at the network edge near the source or data source. It can also provide nearby edge intelligence services to meet critical needs of industry digitization such as agile connectivity, real-time business, data optimization, application intelligence, security, and privacy protection. Edge computing differs from cloud computing in that edge computing allows users to offload the deployment of computing tasks to servers at the edge of the network, localizing the service, significantly reducing the amount of remote data transmission, lowering network transmission latency, decreasing the cost of equipment consumed during the transmission of computing tasks, and abating placement costs. This paper selects appropriate edge servers for grid end tasks through an edge computing-based multi-task assignment strategy (MPA) in order to process tasks from grid end devices efficiently and with almost no delay, which not only saves the transmission time of massive data to and from the cloud but also improves data transportation efficiency and ensures real-time data processing while reducing the possibility of network congestion.

The remaining chapters of this paper are organized as follows. In Sect. 2, related work on smart grids, edge computing, and particle swarm algorithms are introduced. In Sect. 3, the system model is introduced. In Sect. 4, the effectiveness of the proposed strategy is verified by simulation experiments. In Sect. 5, the paper is summarized and the future research directions are discussed.

2 Related Work

The problem of task allocation is widely discussed in edge computing, which enables resource-constrained edge servers to provide satisfactory services to end devices by solving the problem of how computational tasks generated by end

applications are distributed among edge servers [6]. It also reduces computational latency and bandwidth consumption, balances the load [7], and improves network QoS. In addition, how to introduce the edge computing paradigm into the smart power grid domain has also become a focus of research. A distributed computing architecture for smart grid IoT devices based on edge computing is proposed in the reference [8]. The architecture introduces an edge node coordinator to achieve collaborative work among smart grid edge nodes. Reference [9] proposes the security and confidentiality of communication information when users access cloud resources, combined with application security to ensure legitimate user permission to operate.

Edge computing is a technology developed in the context of high-bandwidth and time-sensitive IoT integration, where the energy consumption and latency of edge computing migration are essential metrics to measure migration decisions. Reference [10] considered the singularity of optimized resource strategies in mobile edge computing and proposed a multiple resource computing migration energy consumption model based on particle swarm task scheduling algorithm to ensure sufficient reduction of edge device energy consumption under delay constraints. Reference [11] proposed an efficient asynchronous federated learning mechanism for edge network computing, which compresses the redundant communication between nodes and parameter servers during the training process according to an adaptive threshold. Reference [12] for a multi-user serial task offloading problem with latency and energy consumption as the optimization objectives, following the first-come, first-served principle, to make a near-optimal offloading decision for users. Reference [13] constructs an objective function for joint optimization of the average offload delay and resource allocation balance in the context of 3GPP long-term evolution technology application to effectively reduce the average offload delay of multiple users while balancing the workload of each mobile edge computing server. Reference [14] proposed a strategy integrating abundant computing resources of Mobile Cloud Computing (MCC) and the low transmission delay of MEC to formulate computing offloading decisions, and it provided an Iterative Heuristic MEC Resource Allocation (IHRA) scheme. The scheme extends the single-user offloading problem to multi-user offloading while integrating resource constraints and interference among multiple users to achieve collaborative execution of tasks at the edge and in the cloud and reduce task execution latency. Reference [15] presents a one-to-many task assignment problem and designs a task assignment optimization algorithm based on the Lagrange multiplier method. Reference [16] propose a novel stochas-tic approach that jointly optimizes the usage of transmission resources (e.g., bandwidth), and transcoding resources (e.g., CPU) in CLS systems that leverage the cooperation of Cloud, Edge, and Crowd technologies.

The swarm intelligence algorithm, which simulates the collaborative group search in nature, is widely used with its robust global search and optimization capability, less computational cost, and faster convergence speed. The particle swarm optimization algorithm is the first choice for solving some practical engineering problems because of its fast convergence and low setting parameters. The

formula for calculating the inertia coefficient of the particle swarm algorithm was improved in reference [17] and reference [18] by dynamically changing the inertia coefficient according to the fitness value of the particles. The second category combines particle swarm algorithms with other evolutionary algorithms. Reference [19] combined particle swarm algorithm with genetic algorithm to solve the decision problem of offloading the middle layer of deep networks. The third one uses particle swarm algorithms to solve subproblems in the task unloading problem. As in reference [20], to solve the computational offloading problem in ultra-dense heterogeneous networks, a coarse-grained search using a genetic algorithm is followed by a fine-grained search using particle a swarm algorithm to derive the optimal computational offloading decision. Reference [21] proposed a combined particle swarm and genetic algorithm for the migration decision problem of large deep neural networks and proposed a layer merging upload algorithm to solve the upload problem. Reference [22], on the other hand, decomposes the probabilistic task unloading problem into multiple unconstrained subproblems and uses a particle swarm algorithm to solve each subproblem to obtain an optimal solution to the probabilistic task unloading problem.

This paper focuses on overcoming the current problems of large data volume and complicated task processing in power systems and selecting suitable edge servers for different levels of smart grid terminal tasks for processing. A new edge computing-based multitasking strategy for smart grid terminals is proposed in this paper, which optimizes the distance of data transmission to the edge server, the cost, matching constraints, and load rate.

3 System Model

The system model scenario in this paper is an application scenario with multiple terminal devices and multiple edge servers, as shown in Fig. 1, where the terminal device is a smart meter belonging to the edge device level node [23], which can meet the real-time terminal requirements.

Suppose there are N end devices with M edge servers, and each edge server is connected through a wireless communication link. In this paper, we assume that each end-device can offload its computation of the execution task, and the task can only be offloaded to one MEC server for computation when offloading, and each end-device is within the range of the wireless connection. However, every MEC server has limited computing power and cannot accept offload requests from every endpoint at the same time, assuming that the maximum number of load tasks per edge server is h. The set of end devices is $U = \{u_1, u_2, u_3 \dots u_n\}$, The set of edge servers is $S = \{s_1, s_2, s_3 \dots s_m\}$.

3.1 Task Model

Smart grid terminal devices generate different types of tasks randomly. In order to solve the problem of task assignment in smart grid [24], this paper classifies

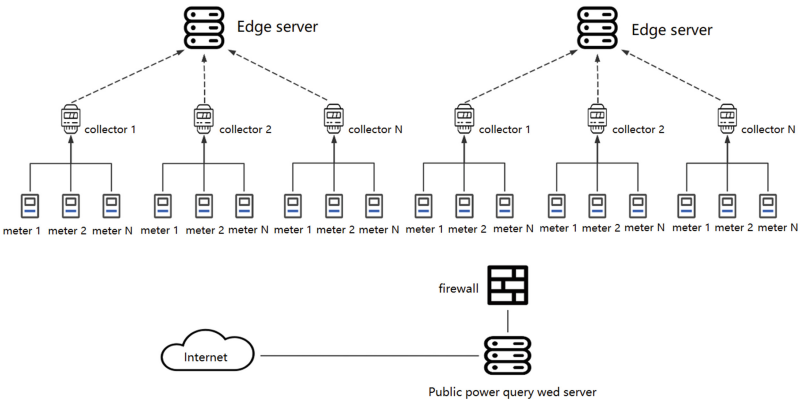

Fig. 1. System model

smart grid terminal task types into four different types according to computational workload, security level and data size, which are general tasks, predictive tasks, statisticaltasks, and top-secret tasks, and selects suitable edge servers according to different levels of smart grid terminal tasks.

Consider a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$, where n is the total number of input tasks in the edge computing system. Each task is denoted by $t_i = \{D_i, W_i, \delta_i, S_i\}$, where D_i denotes the computational workload of task t_i, W_i denotes the security level of the data generated by task t_i, δ_i denotes the data size of task t_i, and S_i denotes the task level.

Table 1. provides a detailed breakdown of the types of tasks randomly generated by the smart grid terminals

Task Type	Calculating workload	Security Level	Data Size	Task Level
General tasks	1	1	1	1
Predictive tasks	3	2	2	2
Statistical tasks	2	2	3	3
Top-Secret Tasks	4	3	2	4

General tasks refer to the collection of power data, which is less computationally intensive and generally does not have security issues; however, the direct delivery of power data to the cloud center will lead to congestion in the communication channel, and standard edge servers can be selected for distribution to reduce costs. Predictive tasks are slightly more computationally intensive and have a higher security level than regular tasks. Generally, predictive tasks are based onhistorical electricity consumption data to predict future electricity consumption at a specific time. A statistical task is usually to count a large amount

of electricity data and find the electricity consumption pattern, which can be used for abnormal data detection, such as electricity theft analysis. The statistical task is not heightened in computational power but very large in data volume, so selecting an edge server that can receive a large amount of data is necessary. Top-secret tasks generally involve power data critical military institutions and state-owned enterprises.

After determining the level of tasks generated by the smart grid terminals, the security level of the edge servers also needs to be considered. If the task level is smaller than the edge server's encryption level, the data security is at risk, and if the task level is higher than the encryption level of the edge server, it will cause additional resource waste because that a higher encryption level means more computational power, storage capacity, and bandwidth capacity is required to encrypt and decrypt the original data. Based on this, a more suitable edge server needs to be matched to suit the level of the task. The security level corresponding to task t of edge server number m is selected and noted as $safe_{mt}$, the target constraint for the overall security level is expressed as follows.

$$\text{Max}\left(O_{\text{safe}}\right) = \sum_{j=1}^{M} \sum_{n=1}^{h} \text{safe}_{jn} \tag{1}$$

$$\text{s.t. } 0 < \text{safe}_{jn} \tag{2}$$

3.2 Load Model

The edge server needs to consider four aspects in its operating state: processor utilization, disk read/write rate, memory utilization, and bandwidth occupancy. The server state is evaluated by combining these four factors, representing the busy level of the server CPU, the throughput of data operations, the system operating state, and the amount of received data, respectively. The set of edge servers is known to be $S = \{s_1, s_2, s_3 \ldots s_m\}$, and the overall performance of its servers is denoted as $C^{S_m} = \left(C^{S_m}_{\text{cpu}}, C^{S_m}_{\text{i/0}}, C^{S_m}_{\text{mem}}, C^{S_m}_{\text{band}}\right)$, where $C^{S_m}_{\text{cpu}}$, $C^{S_m}_{\text{i/o}}$, $C^{S_m}_{\text{mem}}$, $C^{S_m}_{\text{band}}$ denote the maximum processor utilization, disk read/write rate, memory utilization, and bandwidth occupancy that edge server m can carry, respectively. The load state of the edge server m is $\text{Load} = \left(\text{Load}^{S_m}_{\text{cpu}}, \text{Load}^{S_m}_{\text{i/o}}, \text{Load}^{S_m}_{\text{mem}}, \text{Load}^{S_m}_{\text{band}}\right)$, where $\text{Load}^{S_m}_{\text{cpu}}$,$\text{Load}^{S_m}_{\text{i/o}}$,$\text{Load}^{S_m}_{\text{mem}}$,$\text{Load}^{S_m}_{\text{band}}$ denote the current processor utilization, disk read/write rate, memory utilization, and bandwidth usage of server m, respectively. $F^{S_m}_{\text{cpu}}$, $F^{S_m}_{\text{i/o}}$, $F^{S_m}_{\text{mem}}$, $F^{S_m}_{\text{band}}$ denote the percentages of the four metrics of processor utilization, disk read/write rate, memory utilization, and bandwidth occupancy of server m, respectively. Then:

$$\text{Load}^{S_m}_{\text{cpu}} = F^{S_m}_{\text{cpu}} C^{S_m}_{\text{cpu}} \tag{3}$$

$$\text{Load}^{S_m}_{\text{i/o}} = F^{S_m}_{\text{i/o}} C^{S_m}_{\text{i/o}} \tag{4}$$

$$\mathrm{Load}_{\mathrm{mem}}^{S_m} = F_{\mathrm{mem}}^{S_m} C_{\mathrm{mem}}^{S_m} \tag{5}$$

$$\mathrm{Load}_{\mathrm{band}}^{S_m} = F_{\mathrm{band}}^{S_m}\, C_{\mathrm{band}}^{S_m} \tag{6}$$

The total load can be expressed as:

$$\mathrm{Load}_{\mathrm{all}} = \alpha_{\mathrm{cpu}}\, \mathrm{Load}_{\mathrm{cpu}}^{S_m} + \alpha_{\mathrm{i/o}}\, \mathrm{Load}_{\mathrm{i/o}}^{S_m} + \alpha_{\mathrm{mem}}\, \mathrm{Load}_{\mathrm{mem}}^{S_m} + \alpha_{\mathrm{band}}\, \mathrm{Load}_{\mathrm{band}}^{S_m} \tag{7}$$

$$\alpha_{\mathrm{cpu}} + \alpha_{\mathrm{i/o}} + \alpha_{\mathrm{mem}} + \alpha_{\mathrm{band}} = 1 \tag{8}$$

Among them, α_{cpu} ,$\alpha_{\mathrm{i/o}}$,α_{mem} and α_{band} denote the weights occupied by each index of processor utilization, disk read/write rate, memory utilization, and bandwidth occupancy, respectively, and higher values indicate that the proportion of corresponding resources is more significant and more dependent.

The edge server can maintain stable service support when the total load is low enough during the overall task migration. When the load on a single edge server exceeds 1, tasks may not be completed as scheduled, negatively impacting user perception. By denoting Load_{jn_l} the load generated when the smart grid terminal distributes task n to the jth edge server for processing, then the target constraint for the total load can be expressed as:

$$\mathrm{Min}\,(O_{\mathrm{Load}}) = \sum_{i=1}^{M} \sum_{n=1}^{h} \mathrm{Load}_{jn\_l} \tag{9}$$

$$\text{s.t. } 0 < \sum_{n=1}^{h} \mathrm{Load}_{jn\_l} \leq \mathrm{Load}_{all} \tag{10}$$

3.3 Communication Distance Model

Smart grid end devices allocate their computing tasks to edge servers with more abundant computing resources to reduce task execution latency and save cost and energy. Smart grid end devices are connected to the edge server via a wireless channel, and the transmission rates are as follows:

$$\mathrm{R}_{u,s} = W \log_2 \left(1 + \frac{G_{u,s} P_{u,s}}{\sigma^2}\right) \tag{11}$$

$$\text{s.t. } 0 < P_{u,s} \leq P_{u,s}^{\max} \tag{12}$$

$G_{u,s}$ denotes the channel gain between a grid terminal device u and an edge server s [25] in dB;W denotes the bandwidth of the link; σ^2 denotes Gaussian white noise; $P_{u,s}$ is the transmission power between a grid terminal device u and an edge server s, and defines the maximum transmission power of device m as $P_{u,s}^{\max}$. The communication transmission distance is as follows:

$$Dis_{u,s} = \mathrm{R}_{u,s} t_{u,s} \tag{13}$$

$t_{u,s}$ is the transmission time between the grid terminal device u and a certain edge server s.

$$\operatorname{Min}\left(O_{dis}\right)=\sum_{j=1}^{M}\sum_{n=1}^{h}Dis_{jn\_dis} \tag{14}$$

$$\text{s.t. } 0<\mathrm{D}is_{jn\_dis}\leq \mathrm{D}is_{max} \tag{15}$$

3.4 Cost Model

Grid terminal tasks incur costs when computed on the edge server side, and grid companies prefer to choose edge service nodes with lower costs per unit as offload targets to reduce costs. Assume that each edge server has a different price and that each edge server has a standby cost (As long as there is a successful migration of services and standard calculation will incur standby costs, this cost will only be calculated once). Set the standby cost as Cost_Standby, The collection of service prices is as follows:

$$\mathrm{Cost}_s=\{cost_{s1},cost_{s2},cost_{s3}\ldots,cost_{sk}\} \tag{16}$$

Record the active status of the edge server as ac:

$$\begin{cases} ac_m=1 \\ ac_m=0 \end{cases} \tag{17}$$

Among them, $\mathrm{ac}_m=1$ means that edge server m is active;$\mathrm{ac}_m=0$ means that edge server m is inactive. The set of its state is $\mathrm{Activate}=\{\mathrm{ac}_1,\mathrm{ac}_2,\mathrm{ac}_3\ldots,\mathrm{ac}_k\}$, the task security level is Tsafe_n, which is denoted as the security level of the nth task of the jth base station, and the set of security levels for edge servers is as follows:

$$\mathrm{Bsafe}=\{\mathrm{Bsafe}_1,\ \mathrm{Bsafe}_2,\ \mathrm{Bsafe}_3\ldots,\ \mathrm{Bsafe}_m\} \tag{18}$$

γ_{jn} is the task cost multiplier, and the individual task cost is calculated as follows:

$$\gamma_{jn}=\begin{cases} 1,\ \mathrm{Tsafe}_j\leq \mathrm{Bsafe} \\ \mathrm{Bsafe}_j-\ \mathrm{Tsafe}_n+1,\ \mathrm{Tsafe}_j>\ \mathrm{Bsafe}_n \end{cases} \tag{19}$$

$$\mathrm{TaskCost}(j,n)=\mathrm{cost}_{sj}*\left(\mathrm{Load}_{\mathrm{cpu}}^{S_m}+\mathrm{Load}_{\mathrm{i/o}}^{S_m}+\mathrm{Load}_{\mathrm{mem}}^{S_m}+\mathrm{Load}_{\mathrm{band}}^{S_m}\right)*\gamma_{jn} \tag{20}$$

The formula for calculating the total service cost is as follows:

$$\mathrm{Cost}\ =\sum_{j=1}^{M}\sum_{n=1}^{h}\mathrm{TaskCost}(j,n)+\sum_{j=1}^{M}ac_j*\ \mathrm{Cost\_Standby} \tag{21}$$

The target constraint for the total service cost is expressed as follows:

$$\operatorname{Min}\left(O_{\text{Cost}}\right)=\sum_{j=1}^{M}\sum_{n=1}^{h}\text{cost}_{sj}*\sum_{j=1}^{M}\sum_{n=1}^{h}\text{Load}_{jn\_l}+\sum_{j=1}^{M}ac_j*\text{Cost\_Standby} \quad (22)$$

$$\text{s.t. } 0\leq cost_{sj} \quad (23)$$

3.5 Multi-objective Constraint Conversion

In the computing situation mentioned in this paper, multiple task offload requests are issued by grid end devices simultaneously. Due to the limited resources of the edge server, there is no guarantee that the task will be assigned to the most appropriate edge server under any circumstance. In this case, it is necessary to consider the optimal overall benefit and enable the grid end devices to utilize the edge end resources equitably and maximally [26]. Based on this, the four influencing factors of consumption cost, communication distance, edge server load factor, and security level are combined and defined as the combined cost to evaluate whether the task allocation can maximize the use of the edge server resources. A joint optimization model is established, and the normalized transformation function of the cost is obtained using the min-max normalization process [27] as follows:

$$\overline{\text{cost}}_{i,j}=\frac{\text{cost}_{i,j}-\text{cost}_{\min}}{\text{cost}_{\max}-\text{cost}_{\min}} \quad (24)$$

$\text{cost}_{\max}$ and $\text{cost}_{\min}$ are the maximum and minimum values of the consumption cost, respectively, and $\overline{\text{cost}}_{i,j}$ is the normalized consumption cost. The communication distance is normalized to obtain the following:

$$\overline{\text{dis}}_{i,j}=\frac{\text{dis}_{i,j}-\text{dis}_{\min}}{\text{dis}_{\max}-\text{dis}_{\min}} \quad (25)$$

where dis $_{\max}$ and dis $_{\min}$ are the maximum and minimum values of the communication distance, respectively, and $\overline{\text{dis}}_{i,j}$ is the normalized communication distance value. In order to minimize the cost, transmission distance, and load factor, it is only needed to minimize the cost affiliation function value, transmission distance function value, and load factor function value, where the security level of the task corresponding to the base station should be as high as possible, and the specific transformation process is expressed as follows:

$$\text{Min}(O)=\mu_1\,\text{MinO}_{\text{Cost}}+\mu_2\,\text{MinO}_{dis}+\mu_3\,\text{MinO}_{\text{load}}+\mu_4\,\text{Max}\,O_{\text{safe}} \quad (26)$$

where: $\text{Min}(O)$ is the minimum expected objective, μ_1 is the cost weight, μ_2 is the distance weight, μ_3 is the load weight, μ_4 is the matching restriction weight, and the following equation is satisfied:

$$\sum_{i=1}^{4}\mu_i=1 \quad (27)$$

For different needs, it can be satisfied by changing the values of μ_1, μ_2, μ_3 and μ_4.

4 Particle Swarm Algorithm

The particle swarm optimization (PSO) algorithm is a swarm intelligence algorithm proposed by Dr. Eberhart and Dr. Kennedy by studying the predatory behavior of birds [28]. Bird flock search for food is based on information sharing and collaborative cooperation among individual members of the flock. The PSO algorithm is relatively simple to implement and does not require many parameters to be tuned. It has been widely used for solving optimization problems and training neural networks [29].

The PSO algorithm designs a flock of massless particles to simulate the behavior of a flock of birds, and each particle has two essential properties, namely, position X and velocity V, where the position of the particle represents a feasible solution to the problem to be solved in the search space, and the velocity of the particle represents the rate and orientation of the particle to be flown. Each particle adjusts its position according to the individual extremum pbest and the global extremum gbest, thus approaching the optimal position in the search space, where pbest is the best position found so far for the particle itself and gbest is the best position found so far for the particle population. In addition, each particle calculates the corresponding fitness value based on its current position and uses this value to evaluate the individual extremum pbest and the global extremum gbest. Iterative operation on particle swarm (the primary purpose is to calculate the velocity and position of the particle, update the two poles), finally, all particles will converge to the optimal position, thus finding the optimal solution of the problem to be solved.

The PSO algorithm is inspired by the flock of birds foraging in nature. When a bird is foraging for food, in addition to following its target flight, it has to refer to the flight path of other birds in the flock, especially the one close to the food.

In particle swarm algorithms, each particle is a candidate solution, and each particle has N dimensions, which depend on the particular research problem. In our research problem, N is the task to be assigned, and assigning N tasks to M data centers minimizes the processing time, transmission time, processing cost, and transmission cost. Each particle is iteratively updated until the desired result is obtained or the iterative set-point is reached. The process of implementing PSO is shown in Algorithm 1.

1. Initialize a population array of particles with random positions and velocities on D dimensions in the search space.
2. loop.
3. For each particle, evaluate the desired optimization fitness function in D variables.
4. Compare the particle's fitness evaluation with its pbesti.
5. Identify the particle in the neighborhood with the best success, and assign its index to the variable g.

6. Change the velocity and position of the particle according to the following equation:

$$\begin{cases} v_i^{(t+1)} = wv_i^t + c_1 r\left(\mathrm{pb}_i^{(t)} - x_i^{(t)}\right) + \\ \qquad\qquad c_2 R\left(\mathrm{gb}^{(t)} - x_i^{(t)}\right) \\ x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \end{cases} \tag{28}$$

7. If a criterion is met (usually a sufficiently good fitness or a maximum number of iterations), exit loop.
8. end loop

where v_i^t and $x_i^{(t)}$ denote the velocity and position of the particle,$\mathrm{pb}_i^{(t)}$ denotes the individual historical optimum of the ith particle, $\mathrm{gb}^{(t)}$ denotes the global optimum of the whole population, r and R are random values within [0,1], respectively, and $\boldsymbol{w}$, c_1, and c_2 are the weight values.

5 Experiments

5.1 Parameter Setting

To verify the effectiveness of the MPA algorithm, the authors of this paper compare five algorithms and simulate the randomly generated tasks of electricity meters as data sets for simulation experiments. The four aspects, namely cost, communication distance, edge server load factor, and matching constraint, are analyzed respectively.

Most of the code is implemented in Python 3.8, and the experiments were conducted on a server with a 3.8 GHZ AMD CPU and 32 G of RAM.

5.2 Comparison of Algorithms and Metrics

Table 2 shows the algorithms considering combined factors and multiple algorithms considering single factors.

The proposed algorithms are tested and compared using the following four evaluation metrics.

1. Cost: represents consumption; the lower the indicator, the lower the cost.
2. Communication Distance: represents the communication distance metric, and the lower the metric, the closer the edge server selected for the task.
3. Load Rate: represents the edge server load factor; the lower the metric, the more stable the edge server is.
4. Matching Constraints: represents the matching constraints metric of tasks and edge servers; the lower the metric, the more edge servers the task can choose.

Table 2. Task classification

Algorithm	Specific description
MPA	The algorithm proposed in this paper considers a combination of factors cost, communication distance, load factor, and matching restriction parameters to select a suitable edge server to handle the grid terminal device tasks.
MLA	A greedy algorithm that prioritizes edge server load factor to select edge servers
MDA	A greedy algorithm that prioritizes communication distance to select edge servers.
MCA	A greedy algorithm that prioritizes consumption costs to select edge servers.
MRA	Randomly select edge servers for tasks.
MTA	Consider both communication distance and consumption cost (Reference [15]).

5.3 Analysis

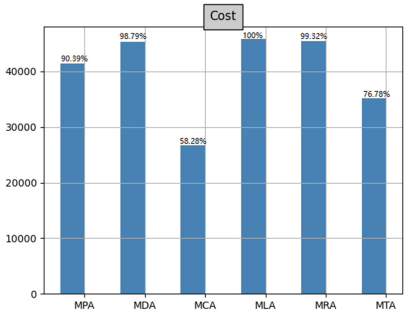

Fig. 2. Cost

Comparative Cost Analysis: A particular cost is consumed in assigning grid terminal tasks to the appropriate servers. Figure 2 indicates that algorithm MLA consumes the highest cost, while algorithm MCA consumes the lowest cost and accounts fTor only 58.28% of the cost generated by the MLA algorithm. Because that the MLA algorithm gives preference to edge servers with a lower load, which incur relatively higher costs, while the MCA algorithm focuses on selecting the edge servers with the lowest selection cost. Algorithm MTA considers both distance and cost, so the consumed cost is between the range of cost consumed by algorithm MCA and MDA. In contrast, MDA, MPA, and MRA, all three algorithms consume relatively high costs because they consider only one factor. The cost of the MPA algorithm proposed in this paper accounts for 90.39% of

the cost of the MLA algorithm. This is because the MPA algorithm takes into account the cost and also integrates other factors, and finally selects an edge server with moderate cost.

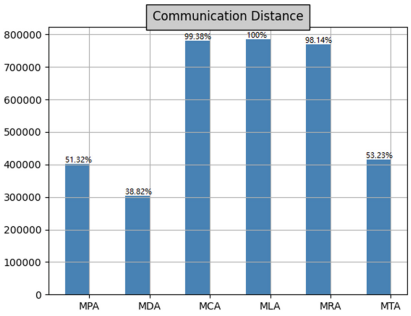

Fig. 3. Communication Distance

Comparative Analysis of Communication Distance: As shown in Fig. 3, the MLA algorithm prioritizes the edge server load, in assigning the grid terminal tasks to the appropriate servers. The distance of the selected edge server is not the first factor to be considered by the algorithm, resulting in a long distance. In contrast the MDA algorithm prioritizes the distance of the edge server and selects the closest edge server for the terminal tasks, with a distance of only 38.82% of the distance selected by the MLA algorithm. The MTA algorithm considers the cost as well as the distance, and the distance accounts for 53.23% of the distance selected by the MLA algorithm. The MPA algorithm proposed in this paper, after considering comprehensive factors, selects an edge server distance for the end task that is farther than the distance selected by the MDA algorithm but closer than the MCA, MLA, MTA and MRA algorithms which is a moderate distance.

Comparative Load Rate Analysis: The lower the edge server load ratio, the more stable the edge server runs, so when assigning edge servers to grid terminal tasks, the load ratio must be considered so that some nodes do not stop responding because of excessive load or remain idle and cause a waste of resources. Figure 4 reveals that the edge server selected by the MCA algorithm is overloaded, which is because the MCA algorithm only considers the cost, and the lower the cost, the higher the load rate; the MLA algorithm gives priority to the server with lower load, which only accounts for 33.36% of the load rate of the edge server selected by the MCA algorithm. After considering the comprehensive factors, the load rate of the edge server selected by the MPA algorithm proposed in this paper only accounts for 38.02% of the load rate of the edge servers selected by the MCA algorithm, which is in a moderate range.

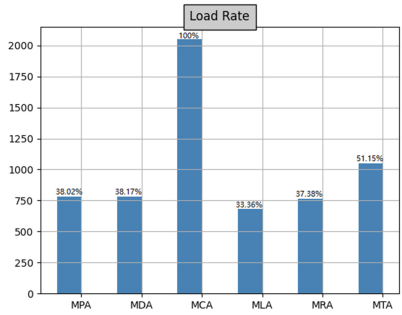

Fig. 4. Load Rate

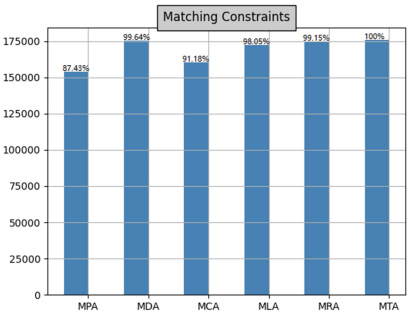

Fig. 5. Matching Constraints

Comparative Analysis of Matching Constraints: The tasks generated by the smart grid terminal have a certain security level. When assigning tasks to appropriate edge servers, the security level of tasks cannot be higher than the security level of edge servers, and tasks with a low-security level can be offloaded to high-security level servers. In contrast, tasks with high security levels cannot be offloaded to low-security level servers. Otherwise, the data will have security risks. The lower the matching constraint parameter, the more edge servers the task can be assigned to. As shown in Fig. 5, the matching constraints of the proposed MPA algorithm in this paper is the lowest among all experimental algorithms, indicating that the grid end tasks can choose to offload more edge servers.

6 Conclusion and Outlook

With the development of the IoT industry, the data from grid terminals has increased dramatically. When transferring data to cloud servers for computing, there are problems such as increased transmission costs, network delays, and data loss or distortion, and edge computing technology can effectively solve these problems. Selecting the appropriate edge server for grid terminal tasks becomes

a challenge. The authors propose an edge-computing-based smart grid terminal multitasking assignment strategy (MPA) to solve this problem. Compared with traditional task assignment methods, the MPA algorithm divides smart grid terminal tasks into different classes. It selects appropriate edge servers for tasks by considering a combination of factors such as cost, communication distance, server load, and matching constraints so that the total energy consumed by all terminal devices and edge servers is minimum. In order to verify the performance of the algorithm MPA, the authors conducted simulation experiments, and the experimental results show that the MPA algorithm proposed in this paper can effectively select the appropriate edge server for different levels of grid terminal tasks, reduce the network transmission delay, and reduce the various costs consumed in the process of task transmission.

Nevertheless, the MAP algorithm proposed in this paper still has some limitations. In the task allocation scenario considered in this paper, although the edge server acts as the main resource provider for task computing, the computing resources are still limited, and for tasks with more data volume and computation and lower latency, the MPA algorithm in this paper can be considered to be integrated with cloud computing architecture in the future.

References

1. Ashton, K.: That 'internet of things' thing. RFID J. **22**(7), 97–114 (2009)
2. Charniak, E.: Introduction to Artificial Intelligence. Pearson Education India (1985)
3. Sadiku, M.N.O., Musa, S.M., Momoh, O.D.: Cloud computing: opportunities and challenges. IEEE Potentials **33**(1), 34–36 (2014)
4. Kumar, K., Liu, J., Lu, Y.H., et al.: A survey of computation offloading for mobile systems. Mobile Netwo. Appl. **18**(1), 129–140 (2013)
5. Shi, W., Cao, J., Zhang, Q., et al.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
6. Wang, S., Zafer, M., Leung, K.K.: Online placement of multi-component applications in edge computing environments. IEEE Access **5**, 2514–2533 (2017)
7. Dai, Y., Xu, D., Maharjan, S., et al.: Joint load balancing and offloading in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4377–4387 (2018)
8. Wang, P., Liu, S., Ye, F., et al.: A fog-based architecture and programming model for IoT applications in the smart grid. arXiv preprint arXiv:1804.01239 (2018)
9. Bo, Y., Ma, Y., Ma Z., Shao, S., Yang, S., Wang, M.: Research on key technologies for secure access management of resource pools. J. Jiangxi Normal Univ. (Nat. Sci. Edn.), **44**(06), 639–643 (2020). https://doi.org/10.16357/j.cnki.issn1000-5862.2020.06.16
10. Tang, L., He, S.: Multi-user computation offloading in mobile edge computing: a behavioral perspective. IEEE Netw. **32**(1), 48–53 (2018)
11. Shahidinejad, A., Farahbakhsh, F., Ghobaei-Arani, M., et al.: Context-aware multi-user offloading in mobile edge computing: a federated learning-based approach. J. Grid Comput. **19**(2), 1–23 (2021)
12. Zhou, S., Jadoon, W.: The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment. Comput. Netw. **178**, 107334 (2020)

13. Yang, X., Yu, X., Huang, H., et al.: Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems. IEEE Access **7**, 117054–117062 (2019)
14. Huang, P.Q., Wang, Y., Wang, K., et al.: A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. IEEE Trans. Cybern. **50**(10), 4228–4241 (2019)
15. Jianbin, X., Yaning, A.: A novel task offloading and resource allocation strategy based on edge computing. Comput. Eng. Sci. **42**(06), 959–965 (2020)
16. Chen, X., Xu, C., Wang, M., Wu, Z., Zhong, L., Grieco, L.A.: Augmented queue-based transmission and transcoding optimization for livecast services based on cloud-edge-crowd integration. IEEE Trans. Circuits Syst. Video Technol. **31**(11), 4470–4484 (2021). https://doi.org/10.1109/TCSVT.2020.3047859
17. Deng, X., Sun, Z., Li, D., et al.: User-centric computation offloading for edge computing. IEEE Internet Things J. **8**(16), 12559–12568 (2021)
18. Wang, Y., Wu, L., Yuan, X., et al.: An energy-efficient and deadline-aware task offloading strategy based on channel constraint for mobile cloud workflows. IEEE Access **7**, 69858–69872 (2019)
19. Chen, X., Zhang, J., Lin, B., et al.: Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments. IEEE Trans. Parallel Distrib. Syst. **33**(3), 683–697 (2021)
20. Zhou, T., Qin, D., Nie, X., et al.: Energy-efficient computation offloading and resource management in ultradense heterogeneous networks. IEEE Trans. Veh. Technol. **70**(12), 13101–13114 (2021)
21. Xue, M., Wu, H., Li, R., et al.: EosDNN: an efficient offloading scheme for DNN inference acceleration in local-edge-cloud collaborative environments. IEEE Trans. Green Commun. Netw. **6**(1), 248–264 (2021)
22. Liu, Z., Dai, P., Xing, H., et al.: A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing. IEEE Trans. Syst. Man, Cybern. Syst. (99), 1–14 (2021)
23. Dinh, T.Q., Tang, J., La, Q.D., et al.: Offloading in mobile edge computing: task allocation and computational frequency scaling. IEEE Trans. Commun. **65**(8), 3571–3584 (2017)
24. Wang, P., Yao, C., Zheng, Z., et al.: Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems. IEEE Internet Things J. **6**(2), 2872–2884 (2018)
25. Diao, X., Zheng, J., Wu, Y., et al.: Joint computing resource, power, and channel allocations for D2D-assisted and NOMA-based mobile edge computing. IEEE Access **7**, 9243–9257 (2019)
26. Aki, H.: Better than net benefits: rethinking the FERC v. EPSA test to maximize value in grid-edge electricity markets. Ecology Law Q. **44**(2), 419–444 (2017)
27. Robinson, M.D., Oshlack, A.: A scaling normalization method for differential expression analysis of RNA-SEQ data. Genome Biol. **11**(3), 1–9 (2010)
28. Xinchao, Z.: A perturbed particle swarm algorithm for numerical optimization. Appl. Soft Comput. **10**(1), 119–124 (2010)
29. Singh, N., Singh, S.B., Houssein, E.H.: Hybridizing SALP swarm algorithm with particle swarm optimization algorithm for recent optimization functions. Evol. Intell. 1–34 (2020)
30. Tsai, H.C., Lin, Y.H.: Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior. Appl. Soft Comput. **11**(8), 5367–5374 (2011)