Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 2, No. 3, 2022

Compute offloading solution to maximize server rewards

Xiaochen Zhu

College of Automation and Electrical Engineering, Dalian Jiaotong University, Dalian 116028, China
15506598522@163.com

Abstract: With the development of the Internet of Things, cloud computing can no longer meet the demand, and edge computing emerges as the times require. As one of the most critical technologies in edge computing, computing offloading has been extensively studied. The research does not aim to minimize computing delay and energy consumption, but to maximize the reward of edge computing servers. Considering the scenario of one MEC server and multiple terminal devices, the target problem obtained is the two-dimensional 0-1 knapsack problem. In view of basic sine cosine algorithm is difficult to solve the discrete target offloading problem, the basic SCA is improved to obtain ISCA, and the offloading problems with 6, 15 and 50 terminal devices are solved respectively, compared with other algorithms, the simulation results show that ISCA has better performance, better solving ability, it is not easy to fall into local optimum, and the convergence speed is also greatly improved compared with SCA.

Keywords: Edge computing; Compute offloading; Sine cosine algorithm(SCA); MEC server reward; Knapsack problem.

1. Introduction

With the development of the Internet of Things and wireless communication technologies, a large number of intelligent application devices have emerged. In many application scenarios, intelligent IoT terminal devices often need to run and process complex calculations and large amounts of data. However, most terminal devices are subject to physical constraints. Due to the limitations of size and appearance design, it only has limited computing, storage and energy resources. The traditional cloud computing solution is to offload tasks with intensive computing and high energy consumption from the terminal to the cloud for processing, but at the same time, the transmission of tasks brings high latency which makes it difficult to meet the requirements of certain tasks, and edge computing emerges as the times require. Edge computing is a supplement to cloud computing. Edge servers are closer to the terminal, and cloud services are "sinked" to the edge of the network, which can solve the problems of long delay in computing offloading and occupying too many network resources in cloud computing. Edge computing can be applied in Autonomous driving, Internet of Things and Industrial Internet, and many other fields.

With the continuous increase of user equipment and the increasing amount of computing tasks, how to use edge server nodes with limited resources to ensure the quality of service for end users has become a hot issue in edge computing. It is impossible to place all the tasks of the terminal in the edge server for processing. Therefore, it is necessary to use a reasonable computing offloading strategy to offload some computing tasks to the edge or cloud, and computing offloading becomes a key problem. Authors of elaborates the research status of computing offloading technology according to the goals of offloading decision, coarse-grained and fine-grained offloading methods, and offloading methods under the cooperation of MEC and Device-to-Device technology. The existing research results in this field are summarized.

In order to meet the challenges, mobile edge computing deploys edge servers on the WLAN side and offloads some computing-intensive tasks to edge servers, thereby shortening the distance between computing services and mobile devices, and powerful computing capabilities localize computing tasks and reduce data transmission cost becomes possible. Literature proposes EUAGame, a game theory method that expresses the edge user allocation (EUA) problem as a potential game, and designs a novel decentralized algorithm to find Nash equilibrium in the game, as the EUA problem solution. The work proposed joint downlink and uplink edge computing offloading and communication and computing resource allocation, and then transformed the offload optimization problem into minimizing the total delay time of the task while saving the energy consumption of mobile devices. Authors of conducted research on the High-Altitude Balloons (HABs) network scenario in MEC, and proposed a federated learning algorithm based on Support Vector Machine (SVM) to determine user associations and optimize each user's Service sequence and task assignment. Most of the existing literature on compute offloading is discussed with the goal of minimizing the delay and energy consumption. This paper does not consider the delay and energy consumption, and focuses on the MEC server to study how to make decisions to maximize the rewards of the server.

2. System Model

In this paper, we consider an edge computing scenario consisting of an MEC server and multiple terminals. The computing resources and communication resources of the server are limited, and each terminal has different requirements for computing resources and communication resources. The size of computing resources is measured by the number of cpu cycles, and the number of communication resources is measured by the number of sub-channels. (C , B) represents the computing resources and communication resources of the MEC server. Tasks are indivisible, either executed locally or fully offloaded to the MEC server for execution. (pi , qi) represent the computing resource and

communication resource requirements of the i-th terminal task, respectively. When the terminal wants to offload the task to the server, it needs to pay a certain reward to the server. Due to the preference of the terminal to the server and other factors, the reward that each terminal is willing to pay is different. Si represents the reward that the i-th terminal is willing to pay to the MEC server. In this offloading scenario, the constraint of delay is not considered, and it is assumed that all terminals meet the general offloading conditions, that is, the offloading delay is smaller than the local execution delay. Use the variable yi to indicate whether the i-th task is offloaded, yi=1 indicates offloading, and yi=0 indicates not.

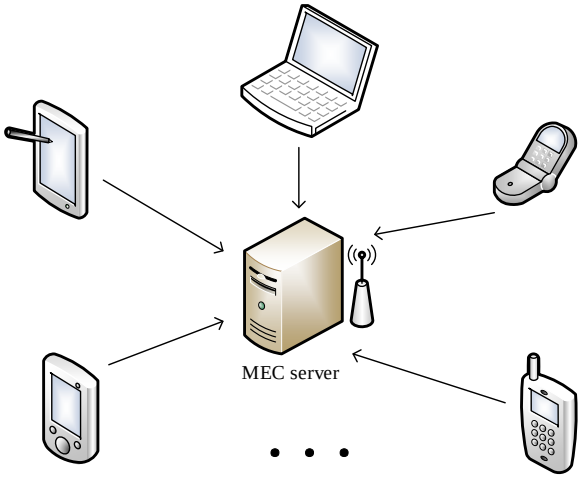

Figure 1. system architecture

3. Problem Formulation

The goal is to let the server reasonably select some terminals to obtain the maximum reward under the condition of limited resources. The total reward of the server is:

$$R = \sum_{i=1}^{L} y_i S_i \tag{1}$$

The target problem is described as:

$$\max(R = \sum_{i=1}^{L} y_i S_i) \tag{2}$$

$$\text{s.t.} \sum_{i=1}^{L} y_i p_i \le C \tag{3}$$

$$\sum_{i=1}^{L} y_i q_i \le B \tag{4}$$

The analysis shows that the objective optimization problem is a two-dimensional 0-1 knapsack problem. The method to solve the problem is discussed below.

4. Algorithm to Solve the Target Problem

4.1. Dynamic Programming

Dynamic programming is a more classical method for solving the knapsack problem. The implementation is as follows: V(L, mL, nL) indicates that the MEC server has selected the maximum value obtained by the first L mobile terminals, mL indicates the remaining communication resources of the MEC server at this time, and nL indicates the remaining computing resources of the server. Supposing a choice is to be made on the L-th terminal, there are two cases for the L-th terminal:

(1) When the resources required by the terminal are greater than the current remaining resources of the server, it is impossible for the server to select it for offloading. The total reward is the same as that of the options for the first L-1 terminals, i.e.V(L-1,mL,nL).

(2) When the remaining resources of the server meet the needs of the terminal, assuming that offload is selected, the total reward is SL+V (L-1, mL-qL,nL-pL). If you choose not to offload, the reward is V (L-1, mL,nL). The final plan is determined according to the total reward of the two options. So, the state transition equation is:

$$V(j, m_j, n_j) = \max\{V(j-1, m_j, n_j), S_j + V(j-1, m_j - q_j, n_j - p_j)\}$$

The dynamic programming method can obtain the exact solution of the problem, but the time and space complexity of the algorithm is exponential, and the recursive algorithm will lead to solving the common sub-problem more than once, which can be achieved when the amount of data is small, but when the amount of problem data is large, the solution efficiency is very low and it takes a long time to calculate.

Although the heuristic algorithm cannot guarantee that the optimal solution can be obtained, the algorithm has low complexity and can obtain the sub-optimal or optimal solution of the problem with high efficiency, which meets the requirements of the offloading system for algorithm efficiency.

4.2. Sine Cosine Algorithm

The sine-cosine algorithm is a meta-heuristic algorithm based on the characteristics of the sine function and cosine function in mathematics. It uses the periodic oscillation of the trigonometric function to explore the optimal solution. By embedding random parameters and adaptive amplitude adjustment, it achieves a balance between global exploration and local exploitation.

In SCA, it is assumed that the population size is N, D-dimensional solution space, and a set of random initial solutions is generated first. Each iteration, the individual updates its position according to the following formula:

$$X_i^d(t+1) = \begin{cases} X_i^d(t) + r_1 \times \sin(r_2) \times |r_3 P^d(t) - X_i^d(t)| & r_4 < 0.5 \\ X_i^d(t) + r_1 \times \cos(r_2) \times |r_3 P^d(t) - X_i^d(t)| & r_4 \ge 0.5 \end{cases} \tag{5}$$

$X_i^d(t)$ is the position of the d-th dimension of the i-th individual in the t-th generation; $P^d(t)$ is the position of the d-th dimension of the optimal individual of the t-th generation; r2, r3, r4 are random numbers subject to different uniform distributions, r2∈[0, 2π], r3∈[0,2], r4∈[0,1]; r1 is a control parameter that balances the exploration and exploitation of SCA by controlling the amplitude of the sine and cosine functions, it is adaptively adjusted by formula (6)

$$r_1 = a - t\frac{a}{T} \tag{6}$$

a is a constant, T is the maximum number of iterations, and t is the current number of iterations.

4.3. Improved Sine Cosine Algorithm

Like other intelligent optimization algorithms, the basic sine and cosine algorithm has shortcomings such as slow convergence speed and poor local exploitation ability. This paper adopts the improved sine and cosine algorithm to solve the target problem.

4.3.1. The nonlinear decreasing function adaptively adjusts the parameter r1

Change r_1 in a non-linear manner, and r_1 is decremented using formula (7). The parameter r_1 controls the search process of the algorithm. In SCA, r_1 decreases linearly in the entire algorithm iteration cycle. This reduction method is not flexible enough, which leads to a relatively rigid process from exploration to exploitation of the algorithm, and is out of balance in the face of many target problems. The algorithm will show the disadvantages of premature or too slow convergence. By changing the original linear decreasing strategy of r_1, the improved SCA algorithm has the stronger global exploration ability in the early stage of the iteration, and the reduction of r_1 is relatively slow to ensure the full exploration of the algorithm; in the middle of the iteration, the algorithm transitions from exploration to exploitation at a relatively fast speed; By the end of the iteration, the descending curve of r_1 will not be too steep, which better takes into account the exploitation of the algorithm. r_1 adopts such a parameter adjustment scheme to better balance the global exploration and local exploitation of SCA.

$$r_1 = a \cdot \exp(-10 \times (\frac{t}{T})^b) \tag{7}$$

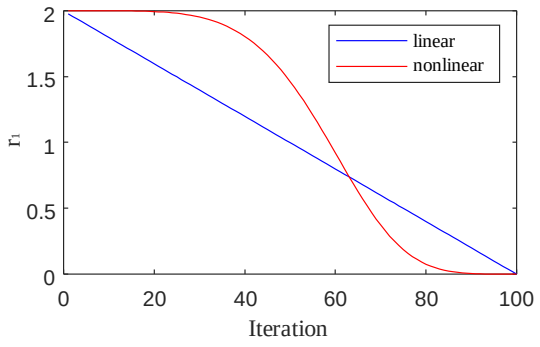

Figure 2. R1 reduction curve

Figure 2 is the reduction curve of r_1 when a=2, b=5.

4.3.2. Modify the Location Update Formula

The sine function sin(r2) and cosine function cos(r2) have the same probability of obtaining a positive value as that of obtaining a negative value, so the removal of the absolute value sign in formula (5) will not affect the overall position update, and it can be reduced the complexity of the algorithm, so the position update formula is rewritten as:

$$X_i^d(t+1) = \begin{cases} X_i^d(t) + r_1 \times \sin(r_2) \times r_3 P^d(t) - X_i^d(t) & r_4 < 0.5 \\ X_i^d(t) + r_1 \times \cos(r_2) \times r_3 P^d(t) - X_i^d(t) & r_4 \geq 0.5 \end{cases} \tag{8}$$

4.3.3. Crossover Strategy

Inspired by the chromosomal crossover in genetic algorithm, the crossover operation is introduced into ISCA. In each iteration, two individuals are randomly selected, and the selected two individuals are crossed according to the crossover ratio pc. Crossover realizes the exchange of information between individuals and improves the diversity of the population.

4.3.4. Discretization and Greedy Repair

The optimization space of most intelligent optimization algorithms is continuous, which is suitable for solving numerical continuous optimization problems. The target problem of this research is a two-dimensional 0-1KP problem, which is a discrete problem. Each dimension of the feasible solution can only take 1 or 0, so the continuous solution in the original algorithm needs to be converted into a discrete solution, that is, the individual should use binary coding.

The continuous solution is transformed into a discrete solution in the following way: a candidate solution in the solution space is represented by an ordered vector pair (X, Y), where X=(x1,x2,...,xd) is a D-dimensional real vector, represents a continuous solution, and Y=(y1,y2,…,yd) is the corresponding binary vector, which is the candidate solution of the target problem. Taking $X \in [-s,s]$ as the continuous solution space of the problem to be sought, we take s=5; $Y \in \{0, 1\}$ is the binary solution space corresponding to the target problem, and the encoding method of converting the continuous solution into the discrete solution is as follows :

$$y_i = \begin{cases} 1 & if \ \text{sig}(x_i) \geq rand \\ 0 & if \ \text{sig}(x_i) < rand \end{cases} \tag{9}$$

Sig () is the sigmoid function, the expression is:

$$\text{sig}(x_i) = \frac{1}{1+e^{-x_i}} \tag{10}$$

rand$\in(0,1)$ is a random number.

When the intelligent optimization algorithm solves 0-1KP, there will be individuals that do not meet the constraints in the iterative process, and methods need to be used to repair these individuals. The abnormal individuals are transformed by the greedy repair method: the mobile terminals are arranged in ascending order of value density, and then the mobile terminals that request the MEC server to be offloaded are discarded in order, that is, the abnormal individuals are checked in order, and the corresponding dimension 0 becomes 1 until the individual satisfies the constraints.

4.3.5. Scout bee mutation strategy

The scout bee strategy in the artificial bee colony algorithm, when the individual's continuous stay times n to a certain threshold G, still cannot find a better position, it indicates that they are trapped in a local optimum. The role of the observer bee will transform into a scout bee and a new location will be generated randomly. Inspired by this, a mutation strategy of scout bees is proposed. When the number of consecutive times n that the individual position does not become better reaches G, the algorithm mutates it, that is, the original 0 in each dimension becomes 1, and 1 becomes 0. The mutation ratio is pb.

4.4. Algorithm Flow

Step1: Initialization parameters, including population size N, space dimension D, number of consecutive stays n, threshold G, maximum number of iterations T, and constant a, b.

Step 2: Randomly generate a binary population, and repair the individuals that do not meet the constraints.

Step 3: Calculate the value density of each mobile terminal, and sort the mobile terminals according to the calculated value density.

Step4: Calculate the target function value of each individual and record the global optimal solution.

Step5: Calculate r1 according to formula (7).

Step6: Update the position of the individual according to formula (8).

Step7: Execute the crossover strategy.

Step8: Execute the scout bee mutation strategy.

Step9: Greedy Repair.

Step10: Update the global optimal solution.

Step11: Is the termination condition met? If satisfied,

output the optimal solution; otherwise, turn to Step5.

5. Simulation Results and Discussions

5.1. Simulation Environment

The simulation was performed on a laptop with 64bit windows10 operating system, Intel(R) Core (TM) i7-6700HQ CPU @ 2.6GHz processor, and 8G memory configuration. The simulation code was written and run with MATLAB R2020b.

5.2. Problems and Parameter Settings

Table 1. Parameters of each algorithm

algorithm	parameters
SCA	a=2
ISCA	a=2,b=5,G=4,pc=0.3,pb=0.1
PSO	W=0.9,$c1$=$c2$=1,Vmax=6
GA	cp=0.5,mp=0.1

The population size is N=90, the maximum number of iterations is T=100, and each algorithm runs independently for 50 times. Particle Swarm Optimization (PSO) prososed in and Genetic Algorithm (GA) are used as comparison algorithms.

Table 2. Problem 1: 6 mobile terminals

(The total amount of computing and communication resources of the MEC server are respectively 80,20)

terminal	pay	computing resource requirements	wireless resource requirements
1	100	8	3
2	600	12	6
3	1200	13	4
4	2400	64	18
5	500	22	6
6	2000	41	4

5.3. Results

In Table 6, the evaluation indicators include the average optimal value, standard deviation, worst value, best value and success rate obtained by running several algorithms independently for 50 times. Among them, the average optimal value can reflect the solution accuracy and convergence speed of the algorithm, the standard deviation is an index to evaluate the stability and robustness of the algorithm, and the best value and the worst value represent the upper and lower limits of the algorithm's ability to solve problems to a certain extent, the success rate reflects the execution efficiency of the algorithm to achieve the target result.

According to Table 6, from the average optimal value obtained by several algorithms, on the offloading problem with different numbers of terminals, ISCA obtains the largest average optimal value. Compared with basic SCA, ISCA has a significant convergence speed and solution accuracy. In terms of standard deviation, the standard deviation obtained by ISCA is the smallest among the three target problems. In contrast, the standard deviation of basic SCA has a considerable gap, which greatly improves the stability of the algorithm; All of them have the ability to solve the problem and get the theoretical optimal value. From the worst value obtained by each algorithm solving 3 offloading problems independently for 50 times, the performance of basic SCA is the worst, it's worst value is the smallest, and the worst value of ISCA is the largest. ISCA is more capable of obtaining higher quality feasible solutions.

For simple problems, that is, when the number of mobile terminals is small, several algorithms can obtain the optimal solution with a success rate higher than 90%, and the success rate of ISCA and GA directly reaches 100%. When the number of mobile terminals increases to 15, the success rate of each algorithm decreases, and when the number of mobile terminals is 50, the success rate further decreases, and the complexity of the target problem has a significant impact on the success rate of the algorithm. The success rate of ISCA is always the highest among several algorithms, and the reliability of solving the target problem is much better than that of SCA. The success rate is not 0, and each algorithm has a certain ability to obtain the theoretical optimal solution.

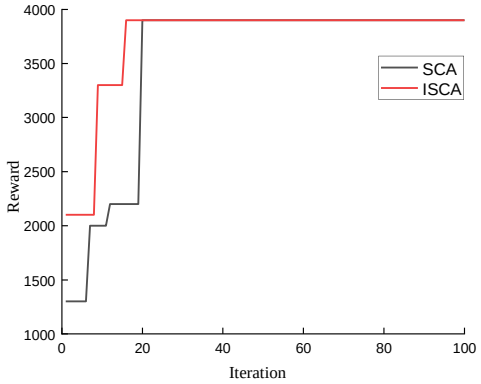

Figure 3. Convergence curves of problem 1

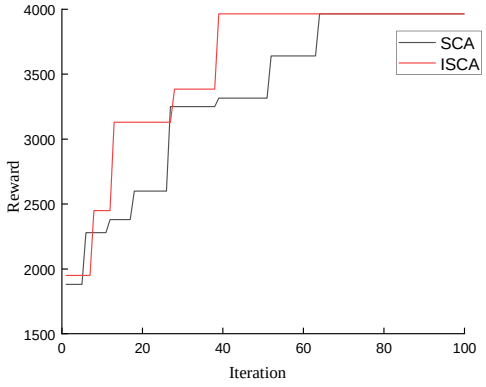

Figure 4. Convergence curves of problem 2

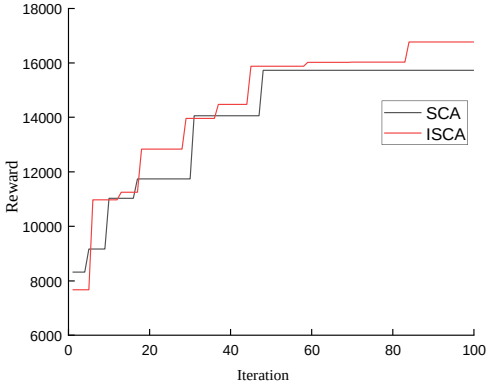

Figure 5. Convergence curves of problem 3

In the above convergence curve, the abscissa is the number of iterations, and the ordinate is the server reward. From the convergence curve of SCA and ISCA, the convergence speed of ISCA is faster than that of SCA. The ability of the algorithm to jump out of local optimum in late iterations is improved.

Table 3. Problem 2: 15 mobile terminals
(The total amount of computing and communication resources of the MEC server are respectively 500,110)

terminal	pay	computing resource requirements	wireless resource requirements
1	100	8	3
2	220	24	6
3	90	13	4
4	400	80	20
5	300	70	20
6	400	80	30
7	205	45	8
8	120	15	3
9	160	28	12
10	580	90	14
11	400	130	40
12	140	32	6
13	100	20	3
14	1300	120	20
15	650	40	5

Table 4. Problem 3: 50 mobile terminals
(The total amount of computing and communication resources of the MEC server are respectively 800,650)

terminal	pay	computing resource	Wireless resource	terminal	pay	computing resource	Wireless resource
1	560	40	16	26	103	5	5
2	1125	91	92	27	215	10	10
3	300	10	41	28	81	8	6
4	620	30	16	29	91	2	4
5	2100	160	150	30	26	1	7
6	431	20	23	31	49	3	4
7	68	3	4	32	420	10	12
8	328	12	18	33	316	42	8
9	47	3	6	34	72	6	4
10	122	18	2	35	71	4	3
11	322	9	12	36	49	8	8
12	196	25	8	37	108	5	10
13	41	1	2	38	116	10	6
14	25	1	1	39	90	1	6
15	425	10	3	40	738	40	28
16	4260	280	200	41	1811	86	93
17	416	10	20	42	430	11	9
18	115	8	6	43	3060	120	30
19	82	1	2	44	215	8	22
20	22	1	1	45	58	3	33
21	631	49	70	46	296	32	36
22	132	8	9	47	620	28	45
23	420	21	22	48	418	13	13
24	86	6	4	49	47	2	2
25	42	1	1	50	81	4	2

In short, the optimization capability of ISCA has been greatly improved compared with the basic SCA. When SCA solves discrete computational offloading problems, it lacks the ability to jump out of the local optimum, and it is easy to stagnate, making it difficult to achieve the goal of finding a better solution. ISCA overcomes this deficiency very well. When solving the corresponding offloading problem, it can always jump out of the local optimum, the convergence speed is also improved, and it can approach the optimal solution at a faster speed. When the number of mobile terminals is small, the solution results of various algorithms are less different, they can solve the problem with a high success rate, while when the number of mobile terminals is large, the superior performance of ISCA can be more clearly reflected.

Table 5. computational results of deterministic algorithms

problem	dynamic programming	greedy algorithm
problem 1	3900	3800
problem 2	3965	3925
problem 3	16761	16590

Table 6. Comparison of heuristic algorithm results

problem	algorithm	average optimum	standard deviation	best	worst	Success rate /%
problem1	SCA	3886	85.738	3900	3300	96
	ISCA	3900	0	3900	3900	100
	PSO	3898	14.142	3900	3800	98
	GA	3900	0	3900	3900	100
problem 2	SCA	3814.6	328.403	3965	2665	80
	ISCA	3963.4	7.918	3965	3925	96
	PSO	3899	157.529	3965	3315	84
	GA	3931.8	109.739	3965	3465	90
problem 3	SCA	15595.5	1779.677	16761	12070	60
	ISCA	16695.96	223.828	16761	15873	92
	PSO	16465.48	439.7	16761	15710	68
	GA	16625.68	296.967	16761	15770	82

6. Conclusion

In this paper, we consider a computing offloading scenario of a MEC server and multiple terminal devices, aiming at the maximum reward of the server, modeling the target problem as a knapsack problem, and improving the sine cosine algorithm to solve it without using the traditional solution method with high complexity. The computational offloading problem is solved with an improved sine cosine algorithm and compared with two classical heuristics. The simulation results show that when solving discrete offloading problems, the improved sine cosine algorithm has the best performance. In the results, the average optimal value and standard deviation of ISCA rank first among all algorithms. Stability, solution speed, and solution success rate of ISCA are greatly improved over the basic sine SCA. We'll consider applying the ISCA other compute offloading scenarios in future work.

References

[1] Liu S, Liu L, Tang J, et al. Edge computing for autonomous driving: Opportunities and challenges[J]. Proceedings of the IEEE, 2019, 107(8): 1697-1716.

[2] Premsankar G, Di Francesco M, Taleb T. Edge computing for the Internet of Things: A case study[J]. IEEE Internet of Things Journal, 2018, 5(2): 1275-1284.

[3] Li H, Li X, Xiong Q, et al. Edge Computing Enabling Industrial Internet: Architecture, Applications and Challenges[J]. COMPUTER SCIENCE, 2021, 48(1): 1-10. (In Chinese)

[4] Zhang Y, Liang Y, Yin M,et al.Survey on the Methods of Computation Offloading in Mobile Edge Computing [J]. Chinese Journal of Computers,2021, 44(12):2406-2430.(In Chinese)

[5] Lin L, Liao X, Jin H, et al. Computation offloading toward edge computing[J]. Proceedings of the IEEE, 2019, 107(8): 1584-1607.

[6] Hu H, Jin F, Lang S. Survey of research on computation offloading technology in mobile edge computing environment [J]. Computer Engineering and Applications, 2021, 57(14):60-74. (In Chinese)

[7] He Q, Cui G, Zhang X, et al. A game-theoretical approach for user allocation in edge computing environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 31(3): 515-529.

[8] Zheng J, Gao L, Wang H, et al. Joint downlink and uplink edge computing offloading in ultra-dense HetNets [J]. Mobile Networks and Applications, 2019, 24(5): 1452-1460.

[9] Wang S, Chen M, Yin C, et al. Federated Learning for Task and Resource Allocation in Wireless High-Altitude Balloon Networks [J]. IEEE Internet of Things Journal, 2021, 8(24): 17460-17475.

[10] Abdel-Basset M, Mohamed R, Mirjalili S. A binary equilibrium optimization algorithm for 0-1 knapsack problems [J]. Computers & Industrial Engineering, 2021, 151: 106946.

[11] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems[J]. Knowledge-based systems, 2016, 96: 120-133.

[12] Mirjalili S. Genetic algorithm[M]//Evolutionary algorithms and neural networks. Springer, Cham, 2019: 43-55.

[13] ZHAN S,WANG L,ZHANG Z,et al. Noising methods with hybrid greedy repair operator for 0-1 knapsack problem [J]. Memetic Computing, 2020, 12(1): 37-50.

[14] Wang H, Wang W, Xiao S, et al. Improving artificial bee colony algorithm using a new neighborhood selection mechanism [J]. Information Sciences, 2020, 527: 227-240.

[15] Wang F, Zhang H, Zhou A. A particle swarm optimization algorithm for mixed-variable optimization problems[J]. Swarm and Evolutionary Computation, 2021, 60: 100808.