Reschedulable Task Allocation Strategy in Cloud-Edge-End Cooperative Mobile Crowd Sensing

Haifeng Jiang*, Shuhao Wang*, Chaogang Tang*, Huaming Wu†, Ruidong Li‡
*School of Computer Science and Technology, China University of Mining and Technology, 221116, Xuzhou, China
†The Center for Applied Mathematics, Tianjin University, 300072, Tianjin, China
‡The Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan
jhfeng@cumt.edu.cn, ts21170095p31@cumt.edu.cn, cgtang@cumt.edu.cn, whming@tju.edu.cn, liruidong@ieee.org

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622378

***Abstract*—In centralized mobile crowd sensing (MCS), the cloud platform assigns all the tasks to participants every time. Since the cloud platform consumes a lot of computing and communication resources to provide services for participants, it will bring about high communication delay and request congestion. The cloud-edge-end architecture for service provisioning has aroused extensive attention recently, owing to its advantages in resource provisioning in close proximity to the resource requestors. Despite the advantages of this architecture, we also observe that it cannot dynamically adjust the allocation scheme when the corresponding computing services are not available to the participants after the initial task allocation. To address this issue, we put forward a re-schedulable task allocation approach in the cloud-edge-end architecture. We aim to improve the efficiency of task execution such as the maximization of task completion rate, while considering service types provided by edge servers and multiple constraints such as resource balancing on the edge servers and deadlines for the task responses. An improved Grey Wolf Optimization (GWO) algorithm is adopted for task rescheduling in this paper. Simulation results indicate that the proposed algorithm performs well in terms of task completion rates and task average response time.**

***Index Terms*—Mobile Crowd Sensing, Cloud-Edge-End Cooperative, Multi-Tasks Allocation, Task Rescheduling**

I. Introduction

Mobile Crowd Sensing (MCS) combines the advantages of crowd sourcing with mobile sensing, collects and processes data from itself or the surrounding environment through smart devices, and applies these data processing results to various fields [1]. In recent years, MCS can quickly and intelligently collect and process a large amount of valuable data, which has the advantages of lower cost of collecting data and larger data coverage. Therefore, MCS has been widely used in environmental monitoring [2], intelligent transportation [3], noise detection [4], intelligent medical treatment [5] and other fields.

In MCS, due to the explosive growth of the number of intelligent devices, the scale of MCS task expands, the amount of data and computing demand multiply, which leads to the consumption of a large number of computing and communication resources when the cloud platform provides computing services for participants, resulting in large communication delay and request congestion, and reducing the system efficiency [6] [7]. Therefore, by combining the cloud-edge-end cooperative architecture in MCS, the edge server has enough computing resources to provide computing services to participants in a timely manner, which effectively reduces communication delay and alleviates request congestion, and improves the quality of service [8]. In the cloud-edge-end cooperative MCS (CMCS), the cloud platform sends different computing models to different edge servers nearby to provide corresponding computing services for participants, so as to respond to the service requests of participants in a timely manner and effectively alleviate the pressure of computing and communication.

Most of the existing research on CMCS task allocation considers the way of one-time task allocation. When faced with the situation that participants cannot obtain computing services, CMCS cannot adjust the task allocation scheme, thus affecting the quality of task completion. Therefore, this paper constructs the task allocation method for rescheduling tasks in CMCS. Specifically, the contributions of this paper are threefold, given below:

- Different from traditional MCS task allocation, this paper considers the situation that participants cannot obtain the computing services required to complete the task, and reallocates the task through the edge server to ensure that participants can obtain the corresponding computing services when performing the task.
- In this paper, we consider the service type of edge server, take maximizing task completion rate as the optimization goal, and take edge server resource balance and task average response time as constraints, construct a resource evaluation model, and propose an Improved Grey Wolf Optimization Algorithm for Task Rescheduling (IGWOTR).
- Extensive simulations are conducted to validate the effectiveness of our approach. Simulation results demonstrate that our method can achieve improved scheme in optimizing task allocations compared to existing strategies.

The rest of the paper is organized as follows. Section II presents the related work in MCS research field. Section

III presents the CMCS model and the resource assessment model. In Section IV presents the programming of IGWOTR. In Section V verify the effectiveness of IGWOTR through simulation experiments. Finally, Section VI concludes this paper.

II. Related Works

Wang et al. [9] address task allocation in MCS by constructing a social relationship network through integrated multi-view social relationship inference to ensure the accuracy of task recommendations. Zhao et al. [10] introduce an innovative MUDBP prediction method based on multi-view and social network group behavior, resolving the issue of increased decision costs for participants due to information overload. Gao et al. [11] propose an automated data collection solution using unmanned aerial vehicles equipped with various sensors to complement manual data collection. Wang et al. [12] leverage a knowledge graph built on the characteristics of sensing users and sensing tasks to mine deep-link relationships between sensing participants and sensing tasks, selecting high-quality sensing users that meet task requirements. Wu et al. [13] present a method for direct collaboration in sensing data acquisition between mobile nodes, decomposing sensing tasks and allocating them to adjacent nodes.

Currently, these studies have shown their advantages in addressing the issues of task allocation in MCS to a great extent. However, with the expansion of task scale, the amount of data and computing requirements doubling, these researches are difficult to meet the current development needs of MCS in saving system resources, reducing system delay and improving system efficiency.

Li et al. [14] introduced edge servers into MCS to facilitate task allocation and user privacy protection. Xiang et al. [15] integrated edge servers into the MCS system and proposed an analytical framework for assessing the relationship between task quality and cost in an edge computing-based MCS system. Li et al. [16] presented a location privacy protection scheme for MCS in an edge environment. Although these studies introduce edge servers into the MCS system to alleviate the computing and communication pressure of MCS, they are all one-time task allocation and cannot make adjustments to the task allocation scheme.

In view of the existing research is difficult to satisfy the current mobile group of mental perception to save system resources, reduce system latency and improve the system efficiency development needs, and cannot make adjustment to the task allocation scheme. This paper combines the cloud-edge-end architecture in MCS and builds the CMCS system framework. By introducing edge servers to provide relevant computing services to participants, the pressure on computing and communication of cloud platform in CMCS is alleviated. In addition, under the framework of CMCS system, this paper proposes a task allocation method of rescheduling tasks to solve the problem that CMCS system cannot adjust the task allocation scheme.

III. System Model

The cloud platform consists of a task set $W = \{w_1, w_2, \ldots, w_N\}$, where N is the number of tasks, an edge server set $S = \{s_1, s_2, \ldots, s_K\}$ where K is the number of edge servers, and a participant set $U = \{U^1, U^2, \ldots, U^K\}$, where U^k represents the set of participants managed by edge server s_k, denoted as $U^k = \{u_1^k, u_2^k, \ldots, u_m^k\}$. Denote the task w_n by a tuple of six parameters, i.e., $w_n = (L_n, Ty_n, Rc_n, T_n, Td_n, D_n)$, where L_n is the task location, Ty_n is the requested service type, Rc_n is the resource call for, T_n is the scheduling times, Td_n is the scheduling dead time and D_n is the amount of task data. Denote the edge server s_k by a tuple of seven parameters, i.e., $s_k = (L_k, v, U^k, Rp_k, St_k, Qs_k, Qr_k)$, where L_k is the location of the edge server, v is the speed of data transfer between edge servers, U^k is the participant set, Rp_k is the edge server resources, St_k is the service type, Qs_k is the queue to be scheduled and Qs_k is the request queue. Denote the edge server u_m^k by a tuple of two parameters, i.e., $u_m^k = (L_m, W_m)$, where L_m is the participant's location and W_m is the participant's task set.

A. Cloud-Edge-End Cooperative MCS Architecture

In this paper, the design of CMCS architecture as shown in Fig. 1. The architecture consists of task initiators, the cloud platform, edge servers, and task participants. Firstly, the task initiator is responsible for submitting the task requirements into the CMSC system. Secondly, the cloud platform sends the computing model to the edge server and generates the global distribution information table of the model, and assigns tasks to participants through the edge server. Then, participants request the edge server to provide computing services after the data collection is completed. When the edge server cannot provide computing services for participants, the task is reallocated to let other edge servers provide computing services for the task. Finally, when the participants completed the task, the cloud platform normalized the task results into available data and returned them to the task initiator.

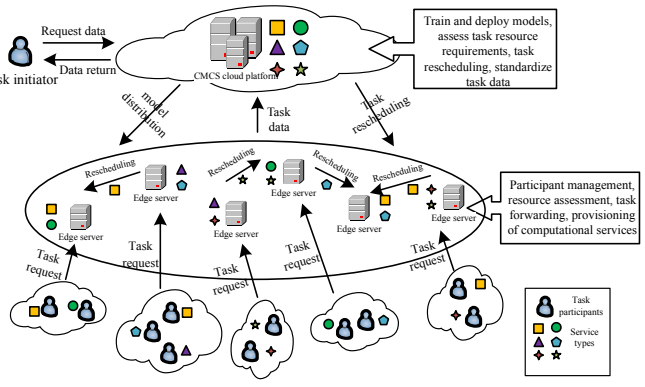

Fig. 1. Cloud-edge-end cooperative MCS architecture.

B. Resource Assessment Model

Edge servers need to provide services for tasks, and balancing the load of edge servers can make the system respond to more requests at the same time and be more efficient. This paper primarily considers CPU frequency (f), the number of CPU cores (c), GPU throughput (v), available running memory (h) and available storage memory (q). Grey Relation Analysis is used to construct the resource assessment model. The specific process is as follows:

Data processing: Construct a decision matrix $B = (b_{ij})_{n\times 5}$, where each column represents an evaluation index, and each row represents a sample to be evaluated:

$$B = \begin{bmatrix} f_1 & c_1 & v_1 & h_1 & q_1 \\ f_2 & c_2 & v_2 & h_2 & q_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_n & c_n & v_n & h_n & q_n \end{bmatrix} \tag{1}$$

We adopt the Z-score standardization method to transform the matrix $B = (b_{ij})_{n\times 5}$ into the matrix $Z = (z_{ij})_{n\times 5}$:

$$z_{ij} = \frac{b_{ij}}{\sqrt{\sum_{i=1}^{n} {b_{ij}}^2}} \tag{2}$$

Therefore, the standardized matrix Z is computed as follows:

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & z_{n3} & z_{n4} & z_{n5} \end{bmatrix} \tag{3}$$

Constructing Virtual Samples: The optimal value corresponding to each index is taken to construct the virtual sample Y:

$$Y = [y_1, y_2, y_3, y_4, y_5] \tag{4}$$

where

$$y_j = max\left(z_{1j}, z_{2j}, \cdots, z_{nj}\right) \tag{5}$$

Calculate the correlation coefficient: The difference matrix be denoted as K:

$$K = \begin{bmatrix} |z_{11} - y_1| & |z_{12} - y_2| & \cdots & |z_{15} - y_5| \\ |z_{21} - y_1| & |z_{22} - y_2| & \cdots & |z_{25} - y_5| \\ \vdots & \vdots & \vdots & \vdots \\ |z_{n1} - y_1| & |z_{n2} - y_2| & \cdots & |z_{n5} - y_5| \end{bmatrix} \tag{6}$$

Therefore, the correlation coefficients μ and φ are calculated as follows:

$$\mu = \min_i \min_j |k_{ij}| \tag{7}$$

$$\varphi = \max_i \max_j |k_{ij}| \tag{8}$$

Calculate indicator correlation: Taking ρ = 0.5:

$$\xi_{ij} = \frac{\mu + \rho \times \varphi}{|k_{ij}| + \rho \times \varphi} \tag{9}$$

Therefore, the grey correlation degree of the ith object, that is, which is the evaluation Q_i:

$$Q_i = \sum_{j=1}^{5} \theta_j \times \xi_{ij} \tag{10}$$

where θ_j represents the weight of indicator j.

Resource State Assessment: The edge server s_k uses the computing resource evaluation model to evaluate its own computing resource status:

$$Rp_k = Q_k \tag{11}$$

The computational resource requirements of the tasks are evaluated to obtain Rc_n:

$$Rc_n = Q_n \tag{12}$$

C. Problem Formulation

IGWOTR takes the edge server service type into consideration, and takes the Task Completion Rate (TCR) as the optimization objective under the constraint of edge server resource balance and task average response time. The response time T_n of the task is the sum of the time taken for task forwarding and the time for task queuing to get the response, as follows:

$$T_n = \frac{D_n \times |L_k - L_i|}{v} + \sum_{j=1}^{|Qr_i|} \frac{Rc_j}{Rp_i} \tag{13}$$

where i is the target edge server s_i, $\forall i \in K$, and j is the jth task in Qr_i of the target edge server s_i. The TCR is as follows:

$$TCR = 1 - \frac{\sum_{k=1}^{K} |Qs_k|}{N} \tag{14}$$

where $|Qs_k|$ is the number of tasks in the set of s_k tasks to be scheduled at the edge server. Based on these descriptions, the optimization problem in this paper is formulated as below:

$$max \;\; TCR$$

$$\text{s.t.} \;\; Ty_n \subseteq St_k \quad \forall n \in N, \forall k \in K \tag{15}$$

$$T_n \leq Td_n \;\; \forall n \in N \tag{16}$$

where the $max \;\; TCR$ is taken as the optimization objective. The constraint (15) ensures that the requested service type Ty_n of w_n is among the service types St_k that s_k can provide. The constraint (16) guarantees that the total response time T_n should not exceed the deadline Td_n of w_n.

IV. Algorithm Design

A. Improved Grey Wolf Optimization algorithm

GWO has the advantages of strong convergence performance, adaptively adjusted convergence factor and information feedback mechanism. Therefore, GWO is selected for task rescheduling and combined with particle swarm optimization algorithm to improve GWO. The strategy of nonlinear adjustment of convergence factor a is proposed by using the

linear differential decreasing strategy of inertia weight, and ω is allowed to update its position according to its own optimum and the global optimum. The position updating model is shown in Fig. 2. The main process of IGWO is as follows:

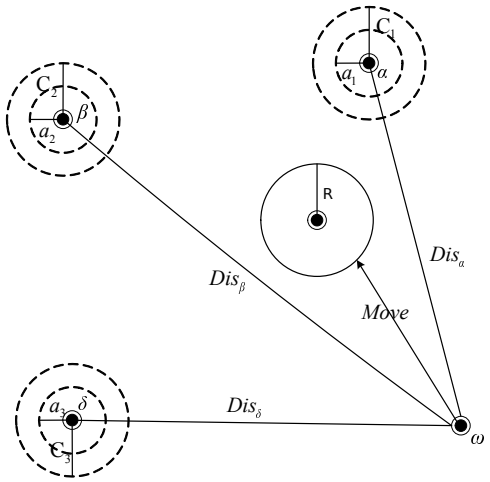

Fig. 2. Improved Grey Wolf Optimizer location update model.

Tracking the prey: In this stage, a mathematical model is built by calculating the distance between prey and wolf, as follows:

$$Dis = |C \times X_p(t) - X(t)| \tag{17}$$

where Dis represents the calculated distance between prey and grey wolf, $X(t)$ represents the position of the gray Wolf at a given time interval t, and $X_p(t)$ represents the position of the prey at a given time interval t. C is a constant coefficient, as follows:

$$C = 2 \times r_1 \tag{18}$$

where r_1 is a random number in the range $[0, 1]$.

Encircling the prey until it stops moving: In this stage, the α, β, and δ to detect prey, while ω according to the equation with the interval change position $t + 1$, as follows:

$$X\left(t+1\right) = X_p(t) - Dis \times A \tag{19}$$

where A is a coefficient constant whose value is affected by a, as follows:

$$A = 2 \times a \times r_2 - a \tag{20}$$

where r_2 is a random number in the range $[0, 1]$. According to the linear differential decline strategy of inertia weight, a strategy of nonlinear adjustment of convergence factor a is proposed, as follows:

$$a = a_{max} - (a_{max} - a_{min}) \times \frac{t^2}{t_{max}^2} \tag{21}$$

where t denotes the current iteration, t_{max} denotes the maximum iteration, and a_{max} and a_{min} are the initial and final values of a, respectively. In the initial iteration, a changes slowly, which is conducive to finding the local optimal value that meets the conditions in the initial iteration. When the number of iterations is close to the maximum, a changes faster, and quickly converges to the global optimal value after finding the local optimal value.

Chasing down the prey: In this stage, α, β, and δ are used to guide the movement of ω so as to achieve global optimization. Update the positions of all grey wolves using the positions X_α, X_β, and X_δ of α, β, and δ, as follows:

$$\begin{cases} X_1 = X_\alpha - A_1 \times |C_1 \times X_\alpha - X| \\ X_2 = X_\beta - A_2 \times |C_2 \times X_\beta - X| \\ X_3 = X_\delta - A_3 \times |C_3 \times X_\delta - X| \end{cases} \tag{22}$$

where X_α, X_β, and X_δ represent the current positions of the three wolves, X represents the current position of the grey wolf, and X_1, X_2, and X_3 represent the positions that ω individuals need to adjust under the influence of α, β, and δ, respectively. Combined with Particle Swarm Optimization algorithm, ω individuals are allowed to learn from the global optimal position and individual historical optimal position at the same time, as follows:

$$H = c_1 \times (X_{best} - X(t)) + c_2 \times (X_1 - X(t)) \tag{23}$$

where H is the learning result of ω.

$$X\left(t+1\right) = \eta \times \frac{X_1 + X_2 + X_3}{3} + H \tag{24}$$

where X_{best} and X_1 are the individual historical best position and the global best position, respectively. c_1 and c_2 are the individual learning factor and the global learning factor, respectively. η is the inertia weight, as follows:

$$\eta = \eta_{min} + (\eta_{max} - \eta_{min}) \times \frac{t_{max} - t}{t_{max}} \tag{25}$$

where η_{min} is the final inertia weight and η_{max} is the initial inertia weight.

Attacking the prey: When the value of a is large, the grey wolf will move away from the prey, hoping to find a more suitable prey, thus prompting the wolves to search globally. If the value of a is small, the grey wolf will move closer to the prey, prompting the wolves to search locally.

B. Improved Gray Wolf Optimization algorithm for Task Rescheduling

In the rescheduling process, in order to avoid the increase of task delay, the edge server generates two queues for tasks: one is the task service request queue, and the other is the task scheduling request queue. When the edge server cannot provide corresponding computing services for participants, the task is rescheduled through the cloud platform. In this paper, the resource utilization $load\left(s_k\right)$ of the edge server is used as the resource load index, as follows:

$$load\left(s_k\right) = \frac{\sum_{i=1}^{|Qr_k|} Rc_i}{|Qr_k| \times Rp_k} \tag{26}$$

where $|Qr_k|$ is the number of tasks allocated to the edge server s_k, and Rc_i is the resource requirement of the ith service request task allocated to the edge server s_k.

Firstly, calculate the resource utilization $load(s_k)$ of all edge servers. Then, the initial population was generated for the criterion of selecting the edge server with the lowest resource utilization $load(s_k)$ among the edge servers that could provide the required computing services for the task. Finally, IGWOTR was used to select the optimal task allocation scheme. The proposed algorithm is shown in Alg. 1.

Algorithm 1 Improved Gray Wolf Optimization algorithm for Task Rescheduling

Input: $S = \{s_1, s_2, \ldots, s_k\}$, population size G, a, t_{max}, C, A.
Output: Task rescheduling allocation scheme X_α.
Initialize: Rp_k, Rc_n, $load(s_k)$, The location of all the wolves.
for $b \leftarrow 1$ to G **do**
    Calculate the TCR for each gray wolf.
end for
The best three grey wolves are selected as X_α, X_β, X_δ.
for $t \leftarrow 1$ to t_{max} **do**
    Update the position of each gray wolf.
    Calculate the TCR for each gray wolf.
    Update X_α, X_β, X_δ.
    Update C, A, a.
end for

V. Simulation Experiments

A. Experimental Setup

This paper uses random data sets and the T-Drive Taxi Trajectories (T-Drive) dataset generated by the Microsoft T-Drive project to collect the trajectory data of more than ten thousand taxis in Beijing for a week. This paper mainly verifies the performance of IGWOTR in two aspects of TCR and task average response time (TAT) under different task sizes and edge server sizes. In this paper, three algorithms including Genetic Algorithm (GA), Simulated Annealing Algorithm (SA) and Tabu Search Algorithm (TS) are selected for comparative experiments with IGWOTR. GA is a stochastic global search optimization method, which can obtain high-quality solutions through random selection, crossover and mutation operations. SA is a stochastic optimization algorithm based on Monte-Carlo iterative solution strategy. TS is a global neighborhood search algorithm for global stepwise optimization. The main parameter ranges are shown in Table. I.

TABLE I
Ranges of Main Parameters

Parameters	Range
Number of users	50
Number of tasks	[60,100]
Number of edge servers	[6,10]
Population size G	30
Maximum number of iterations t_{max}	1000
a_{max}	2
a_{min}	0
c_1,c_2	0.5

B. Analysis of Experimental Results

1) Task Completion Rate: This section mainly analyzes the performance of different algorithms in terms of TCR under different task sizes and edge server sizes.

Fig. 3 shows the variation of the corresponding TCR under different task sizes using random datasets and T-Drive. It can be seen that the change trend of TCR is basically the same, and they all decrease with the increase of task sizes. Because under the premise of the same number of edge servers is unchanged, with the increase of the number of tasks, the task service request queue and task scheduling request queue of each edge server also increase, so that the task scheduling time increases, and the TCR decreases.

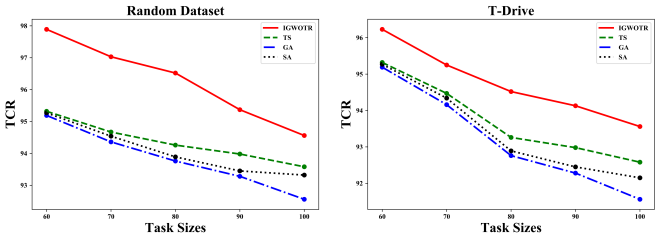

Fig. 3. TCR for different task sizes.

Fig. 4 shows the variation of the corresponding TCR under different task sizes using random datasets and T-Drive. It can be seen that the change trend of TCR under different edge server sizes is basically the same, and they all increase with the increase of edge server sizes. Because under the premise of the number of tasks is unchanged, with the increase of the number of edge servers, the task service request queue and task scheduling request queue of each edge server are reduced, and the tasks can be responded in time, so that the TCR is increased.

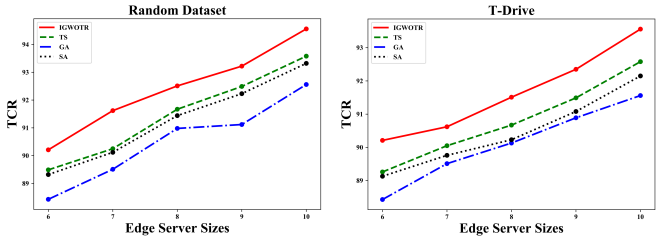

Fig. 4. TCR for different edge server sizes.

2) Task average response time: This section mainly analyzes the performance of different algorithms in terms of TAT under different task sizes and edge server sizes.

Fig. 5 shows the variation of the corresponding TAT under different task sizes using random datasets and T-Drive. It can be seen that the change trend of TAT is basically the same, and they all increase with the increase of task sizes. Because under the premise that the number of edge servers is unchanged, with the increase of the number of tasks, the task request scheduling

queue length of each edge server also increases, so that the TAT increases.

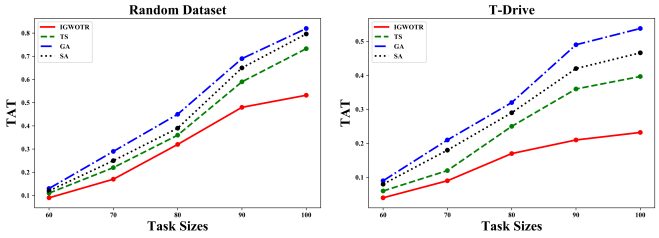

Fig. 5. TAT for different task sizes.

Fig. 6 shows the variation of the corresponding TAT under different edge server sizes using random datasets and T-Drive. It can be seen that the change trend of the TAT is basically the same, and it decreases with the increase of the scale of edge servers. Because under the premise of the same number of tasks, with the increase of the number of edge servers, the task service request queue and task scheduling request queue of each edge server are reduced, so that the tasks can be responded in time.

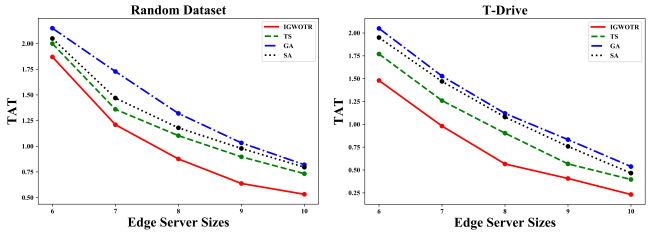

Fig. 6. TAT for different edge server sizes.

3) Analysis: In summary, the proposed IGWOTR is 1.57%, 2.02% and 1.26% higher than GA, SA and TS in TCR, and 40.56%, 34.88% and 25.58% lower than GA, SA and TS in TAT, respectively. The reason for IGWOTR's good performance is that IGWOTR combines the advantages of cloud-edge-end architecture, uses the cloud platform to release the global task scheduling scheme, and then uses the cooperation between edge servers to quickly reschedule tasks. In addition, IGWOTR has strong global search ability and updates the position according to its own optimum and the global optimum by improving the convergence factor a, so that the task can obtain the requested computing service in time, so that IGWOTR can further optimize the task allocation results quickly and accurately.

VI. Conclusion

In this paper, aiming at the task allocation problem of MCS combined with cloud-edge-end cooperative, considering the task rescheduling after the completion of the first task allocation, a task allocation strategy based on task rescheduling is proposed. The simulation results show that the proposed IGWOTR has a significant optimization effect on the first allocation results of tasks, and the TCR and TAT are significantly improved. In future research work, the energy consumption during task rescheduling will be further studied to optimize the energy consumption during rescheduling.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Number 62071327, 62271486 and 62071470. Huaming Wu is the corresponding author.

References

[1] Z. Jin, M. Yao, and D. Tao, "Implicit authentication with sensor normalization and multi-modal domain adaption based on mobile crowd sensing," CCF Trans. Pervasive Com. Interact., vol. 4, no.4, pp. 370–380, Dec. 2022.

[2] J. Chen, J. Yang, "Maximizing coverage quality with budget constrained in mobile crowd-sensing network for environmental monitoring applications," Sensors, vol. 19, no.10, p. 2399, 2019.

[3] H. Li, X. Wu, L. Hou U, and K. Pang Kou, "Near-Optimal Fixed-Route Scheduling for Crowdsourced Transit System," in 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece: IEEE, Apr. 2021, pp. 2273–2278.

[4] P. Huang, X. Zhang, L. Guo, and M. Li, "Incentivizing Crowdsensing-Based Noise Monitoring with Differentially-Private Locations," IEEE Trans. on Mobile Comput., vol. 20, no. 2, pp. 519–532, Feb. 2021.

[5] R. P. Thati, A. S. Dhadwal, P. Kumar, and S. P, "A novel multi-modal depression detection approach based on mobile crowd sensing and task-based mechanisms," Multimed Tools Appl, vol. 82, no. 4, pp. 4787–4820, Feb. 2023.

[6] M. Marjanovic, A. Antonic, and I. P. Zarko, "Edge Computing Architecture for Mobile Crowdsensing," IEEE Access, vol. 6, pp. 10662–10674, 2018.

[7] K. Cao, S. Hu, Y. Shi, A. Colombo, S. Karnouskos, and X. Li, "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems," IEEE Trans. Ind. Inf., vol. 17, no. 11, pp. 7806–7819, Nov. 2021.

[8] Y. Fu, X. Zhang, X. Qin, Q. Meng, and B. Huang, "Data collection of multi-player cooperative game based on edge computing in mobile crowd sensing," Computer Networks, vol. 222, p. 109551, Feb. 2023.

[9] J. Wang, Z. Zhang, and G. Zhao, "Task recommendation method for fusion of multi-view social relationship learning and reasoning in the mobile crowd sensing system," Computer Communications, vol. 206, pp. 60–72, Jun. 2023.

[10] G. Zhao, X. Wang, J. Wang, and J. Liu, "Task recommendation for mobile crowd sensing system based on multi-view user dynamic behavior prediction," Peer-to-Peer Netw. Appl., vol. 16, no. 3, pp. 1536–1550, May 2023.

[11] H. Gao, J. Feng, Y. Xiao, B. Zhang, and W. Wang, "A UAV-Assisted Multi-Task Allocation Method for Mobile Crowd Sensing," IEEE Trans. on Mobile Comput., vol. 22, no. 7, pp. 3790–3804, Jul. 2023, doi: 10.1109/TMC.2022.3147871.

[12] J. Wang, J. Liu, and G. Zhao, "Dynamic link prediction method of task and user in Mobile Crowd Sensing," Computer Communications, vol. 189, pp. 110–119, May 2022.

[13] W. Yafeng, S. Yina, F. Yu, and L. Yazhi, "A Utility-Based Subcontract Method for Sensing Task in Mobile Crowd Sensing," IEEE Trans. Ind. Inf., vol. 18, no. 2, pp. 1210–1219, Feb. 2022.

[14] Z. Li, Z. Song, and X. Chen, "Privacy-Preserving Cost Minimization in Mobile Crowd Sensing Supported by Edge Computing," IEEE Access, vol. 8, pp. 121920–121928, 2020.

[15] Z. Xiang, S. Deng, and Y. Zheng, "Activate Cost-Effective Mobile Crowd Sensing with Multi-access Edge Computing," in 2020 Communications and Networking: 15th EAI International Conference, Shanghai, China, Nov. 2021, pp. 78–97.

[16] M. Li, Y. Li, and L. Fang, "ELPPS: An Enhanced Location Privacy Preserving Scheme in Mobile Crowd-Sensing Network Based on Edge Computing," in 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China: IEEE, Dec. 2020, pp. 475–482.