A Particle Swarm Optimization with Imbalance Initialization and Task Rescheduling for Task Offloading in Device-Edge-Cloud Computing

Hui Fu, Guangyuan Li, Fang Han*, Bo Wang
Faculty of Engineering, Huanghe Science and Technology College, Zhengzhou, China 450006

Abstract—**Smart devices, e.g., smart-phones, internet-of-thing device, has been prevalent in our life. How to take full advantage of the limited resources to satisfy as many as requirements of users is still a challenge. Thus, in this paper, we focus on the task offloading problem to address the challenge by device-edge-cloud computing, by PSO improved with the imbalance initialization and the task scheduling. The imbalance initialization is to increase the probability that a task is assigned to a computing node such that the node provides a longer slack time. The task scheduling is to reassign tasks with deadline violations into other nodes, to improve the number of accepted tasks for each offloading solution. Extensive experiment results show that our proposed algorithm has better performance than other ten classical and up-to-data algorithms in both the maximization of the accepted task number, the resource utilization, as well as the processing rate.**

Keywords*—*Cloud computing; edge computing; task offloading; task scheduling; particle swarm optimization

I. Introduction

In our life, smart devices, e.g., smart-phones, Internet-of-Thing (IoT) devices, are becoming popular, and they are appearing everywhere. As the life quality improves, people request a wide variety of services by their devices. But smart devices generally have limited resource capacity due to their small spaces. Thus, smart devices cannot satisfy all requirements of their users. Edge computing and cloud computing are two of commonly used ways to address this issue. Edge computing places a few servers close to devices, aiming at providing low latency services [1]. Cloud computing extends the capacity of devices by its abundant computing resources, but it usually has poor network performance [2].

Device-edge-cloud computing (DECC) combines benefits of edge and cloud computing, which provides services by not only local device resources but also low latency edge servers and abundant cloud servers [3]. How to make these heterogeneous resources cooperate well is a kind of very challenging work for high service quality and resource efficiency in DECC. Therefore, several works focused on the task offloading or scheduling to address this challenge. The task offloading problem is to decide which and how many resources for each task's processing [3]. The offloading problem has been proofed as NP-hard, because tasks and resources generally are discrete, and the task processing models and the optimization objectives are usually non-convex.

*Corresponding Authors.

To solve the task offloading problem, existing works mainly exploited two kinds of approaches, heuristics and meta-heuristics, based on their desires. Heuristics use local search strategies to provide local best solutions with a very few overheads. For example, Wang et al. [4] proposed two heuristic algorithms for the offloading problem. The first one is iteratively assigning the task to an edge server (ES), which provides the minimal response time. The second one, load balance, iteratively assigns the task to the ES such that the ES can satisfy the most tasks' requirements. Only when there is no available edge resources, these two algorithms assign tasks to the cloud for their processing. Sang et al. [5] and Chen et al. [6] presented multi-stage heuristic algorithms for task offloading in DECC, which both considers to use abundant cloud resources at first for task processing, to improve the overall accept ratio. Heuristics generally provide solutions with limited performance, because they only exploit local search strategies.

Meta-heuristics exploit global search strategies, inspired by some natural or social laws. Meta-heuristics can provide better solutions than heuristics many times, benefiting from their global search abilities, with a few costs. Such as, Wang et al. [7] and Song et al. [8] applied genetic algorithm (GA) to improve the deadline violation and the delay, respectively. Alqarni et al. [9] and Wang et al. [10] made use of Particle Swarm Optimization (PSO) for the delay minimization and the user satisfaction maximization.

No heuristic or meta-heuristic has the best performance as there is no such thing as a free lunch. Thus, a promising way to achieve better performance is combining two or more heuristics and meta-heuristics for hybrid heuristics. Mahenge et al. [11] proposed a hybrid swarm intelligence offloading algorithm, by using the position update strategies of both PSO and Grey Wolf Optimizer to optimize the energy consumption of devices. To combine benefits of both swarm intelligence and evolutionary algorithm, Wang et al. [13] used GA at first, and then applied PSO with the population provided by GA. Hafsi et al. [12] sequentially employed GA and PSO on each iteration of the population evolution. Nwogbaga et al. [14] performed a mutation operator on each individual at the end of each iteration for PSO, to avoid trapping to local best solution. These works only sequentially performed two meta-heuristics without considering their integration, which provides a low combination efficiency.

Therefore, in this paper, we aim at designing a hybrid heuristic algorithm for providing a good task offloading solution in DECC, by improved PSO. Specifically, we represent task assignment solutions as integer coded positions of particles, and use PSO for the position updates. And by the earliest deadline first (EDF) heuristic algorithm for the task scheduling on each computing node, we can achieve a task offloading from each particle position. To improve the performance of PSO, we propose to use an imbalance idea for the population initialization, where the computing node for a task's processing has a possibility positively associated with its capacity. This can take full use of the easily trapping into local best position, by increasing the probability that the initialized particles are near the global best position. To improve the offloading solution, we reschedule tasks with deadline violations to other computing nodes for each assignment solution, i.e., each particle position. We conduct extensive experiments to evaluate the performance of our proposed algorithm, and the results confirm the superior performance in optimizing the number of tasks with requirement satisfactions and the resource efficiency, compared with ten of classical and up-to-data heuristics and meta-heuristics.

In the following, we state the task offloading problem of DECC in Section II. The improved PSO is illustrated in Section III. Experimental evaluations are presented in Section IV, and we conclude our work in Section V.

II. Problem Statement

In a considered DECC system, there are D devices, E ESs, and C CSs, which can be seen as $D+E+C$ computing nodes (respectively represented as n_i, $1 \leq i \leq D+E+C$). For node n_i, there are g_i computing capacity. For data transferring, the network transmission speed between n_i and $n_{i'}$ is $b_{i,i'}$. If there is no network connection between n_i and $n_{i'}$, $b_{i,i'} = 0$. And within one node, there is no latency for data transfer, i.e., $b_{i,i} = 1$.

In the DECC system, there are T tasks (t_j, $1 \leq k \leq T$) launched by these D devices for processing. We use binary constants $l_{i,j}$ to indicate the relationships between tasks and devices, where $l_{i,j} = 1$ means t_j is launched by n_i, and $l_{i,j} = 0$, otherwise. For every task, say t_i, the nodes that can accept its request for processing include the device launching it, ESs having network connections with this device, and CSs. Therefore, we can use $l_{i,j}$, $1 \leq i \leq D+E+C$, $1 \leq k \leq T$, to indicate whether n_i can used for processing t_j, where $l_{i,j} = \sum_{i'=1}^{D}(l_{i',j} \cdot (b_{i',i} \neq 0))$, for $D+1 \leq i \leq D+E+C$, $1 \leq j \leq T$.

For task t_j, the amount of required computing resources, i.e., its computing size, is c_j. The input data amount of t_j is a_j, and the deadline is d_j which means t_j must be finished before d_j. Without loss of generality, we assume $d_1 \leq d_2 \leq \cdots \leq d_T$. Then, if task t_j is offloaded into ES or CS n_i, the data transfer latency is

$$\tau_{i,j}^{D} = \frac{a_j}{\sum_{i'=1}^{D}(l_{i',j} \cdot b_{i',i})}. \tag{1}$$

where $\sum_{i'=1}^{D}(l_{i',j} \cdot b_{i',i})$ is the transmission speed between the device launching t_j and the ES or CS n_i. The computing latency of t_j in node n_i is

$$\tau_{i,j}^{C} = \frac{c_j}{g_i}. \tag{2}$$

In this paper, we ignore the transfer latency of the output data for each task, as the result generally has much less amount than the input [15], [16].

For multiple tasks assigned to one computing node, the data transfers and the computing are processed sequentially. It has been proofed that EDF yields an optimal schedule for minimizing the number of task deadline violations in a computing node [17], which is the major objective in the paper. Therefore, we can deduce the finish time of each task on every node with EDF processing order. Then, the earliest complete time (ft_j^D) of the data transfer for a task can be deduced by Eq. (3). Where $x_{i,j}$ is the binary variable to indicate where t_j is assigned to n_i for its processing (1 is yes, and 0 is no).

$$ft_j^D = \sum_{i=1}^{D+E+C}(x_{i,j} \cdot \sum_{j' \leq i}(x_{i,j'} \cdot \tau_{i,j'}^{D})), 1 \leq j \leq T. \tag{3}$$

For a task's computing on a node, it can be started only when its data transfer finishes and the node is available, where the node is available only when the its previous task finishes its computing. Thus, the finish time of a task's computing, i.e., its finish time, can be calculated iteratively by Eq. (4).

$$\begin{aligned} ft_j = &\max\{ft_j^D, \sum_{i=1}^{D+E+C}(x_{i,j} \cdot \max_{j'<j}\{x_{i,j'} \cdot ft_{j'}\})\} \\ &+ \sum_{i=1}^{D+E+C}(x_{i,j} \cdot \tau_{i,j}^{C}), 1 \leq j \leq T. \end{aligned} \tag{4}$$

Then, based on above formulations, the task offloading problem in the DECC system can be expressed as

$$\text{Maximizing } N = \sum_{i=1}^{D+E+C}\sum_{j=1}^{T} x_{i,j}, \tag{5}$$

subject to

$$ft_j \leq d_i, 1 \leq j \leq T, \tag{6}$$
$$x_{i,j} \in \{0,1\}, 1 \leq i \leq D+E+C, 1 \leq j \leq T. \tag{7}$$

Where the objective (5) is maximizing the number of tasks whose deadlines are met, and the constraints mainly include the deadline requirements (Eq. 6) and the atomicity of every task (Eq. 7). In this work, we consider the hard deadline, where if the deadline of a task is satisfied, the task is accepted for its processing, and it is rejected, otherwise. Due to the discrete decision variables ($x_{i,j}$) and the non-convex constraints (see Eq. 4 and 6), the task offloading problem generally is hard to solve. Existing tools, e.g., lpsolve [18] and MathWorks [19], can provide exact solutions, but has exponential complexity at worst. Therefore, in the following section, we present a hybrid heuristic offloading algorithm to solve the problem with a polynomial time.

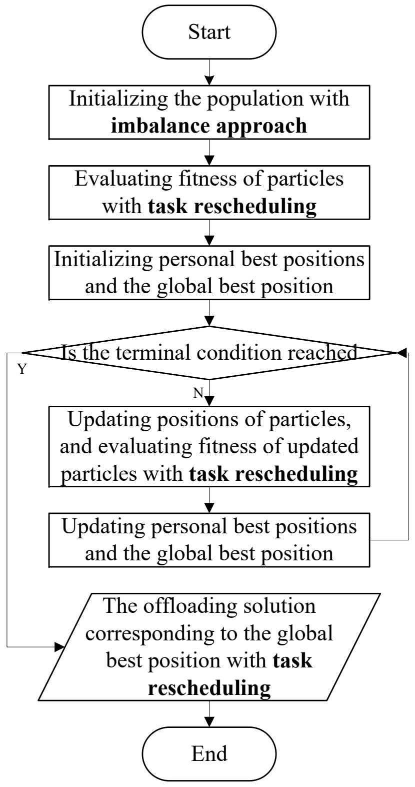

Start
Initializing the population with **imbalance approach**
Evaluating fitness of particles with **task rescheduling**
Initializing personal best positions and the global best position
Is the terminal condition reached
Y
N
Updating positions of particles, and evaluating fitness of updated particles with **task rescheduling**
Updating personal best positions and the global best position
The offloading solution corresponding to the global best position with **task rescheduling**
End

Fig. 1. The flow chart of PSOINRS.

III. The PSO with Imbalance Initialization and Task Rescheduling

PSO has been applied for solve optimization problem in various fields, due to its easy implementation and good performance. But PSO has some drawbacks which prevent its more wide application. Therefore, in this section, we integrate two improvement strategies on PSO to achieve a better offloading solution for DECC, and proposed a hybrid heuristic offloading algorithm, PSOINRS (PSO with imbalance INitialization and task ReScheduling). The algorithm flow chart is shown in Fig. 1.

In PSOINRS, a position of particles represents a task assignment solution. The dimension number of particles is identical to the task number, and the value in a dimension indicates the node that the corresponding task is assigned to.

At first, for the population initialization of PSO, we propose an imbalance approach to improve the particle density in the possible best positions. Intuitively, a task has more possibility to be processed by the node providing longer slack time (the closeness to the deadline). Therefore, PSOINRS initializes the position of a particle by setting the possibility of a node positively associated with the deadline tightness that the node provides, where the probability that t_j is assigned to n_i is

$$p_{i,j} = \frac{s_{i,j}}{\sum_{j'=1}^{T} s_{i',j}}, \tag{8}$$

where $s_{i,j} = d_j - ft_j$ if t_j is assigned to n_i and $ft_j \leq d_j$, and $s_{i,j} = 0$, otherwise.

After the population initialization, PSOINRS evaluates the fitness of each particle according to its position. Given a position, we can easily achieve a task assignment solution. By EDF ordering scheme, we achieve a offloading solution from the task assignment solution. Then, we proposed to use the task rescheduling in PSOINRS to improve the solution quality for each position. The task rescheduling is scheduling tasks with deadline violations to other nodes by EDF to improve the optimization objective (5).

Except the imbalance initialization and the task rescheduling for mapping a position into a task offloading solution, PSOINRS updates the positions of particles identical to PSO, as shown in Fig. 1.

IV. Performance Evaluation

For evaluating the performance of our proposed algorithm presented in the previous section, we simulate a DECC environment based on related works and the reality. In a simulated DECC, there are 10 devices, 5 ESs, and 10 CS types. For each device, there are [1.8, 2.5]GHz computing capacity. Each device is randomly connected with an ES. Each ES or CS type has [1.8, 3.0]GHz computing capacity. The data transfer rate of a device to an ES or CS is set as [80, 120]Mbps, and [10, 20]Mbps, respectively. There are 1000 tasks randomly launched by these devices. Each task has [0.5, 1.2] GHz computing size, and [1.5, 6] MB input data. The deadline of a task is set as [1, 5]s.

We compare our proposed algorithm with First Fit (FF), First Fit Decreasing (FFD), EDF, Short Job First (SJF), random (RAND), GA, GAR [20], PSO, PSOM [14], and GAPSO [13]. RAND is randomly initializing a population, and provide the best individual. We use the following performance metrics for the evaluation of each algorithm, the number of tasks with deadline satisfactions, the overall computing resource utilization, the computing rate, and the data processing rate. The rate of computing and data processing is the computing size and the input data amount of tasks with deadline satisfactions.

Fig. 2 shows the number of tasks with deadline satisfactions when applying different offloading algorithms, which is one of the most commonly used metrics for evaluating the user satisfaction or the quality of service (QoS). From this figure, we can see that PSOINRS has 8.73%–36.8% better performance than others. This verifies the performance superiority of our proposed algorithm. The benefits of PSOINRS are mainly the imbalance initialization and the task rescheduling, which will be both evaluated and illustrated in the followings.

Fig. 3 gives the resource utilizations achieved by these offloading algorithms, which is one of the most frequently used metrics for the quantification of the resource efficiency. As shown in the figure, heuristics (FF, FFD, EDF, and SJF) has better utilizations than meta-heuristics (GA, GAR, PSO, PSOM, GAPSO, and PSOINRS). This is mainly because heuristics prioritise processing tasks locally or in low-latency edge resources, while meta-heuristics pursues the global optimization objective for maximizing the number of accepted tasks, because the task processing with a low data transfer latency has a high computing resource utilization, which is mainly decided by the ratio between the data transfer latency and the computing delay.

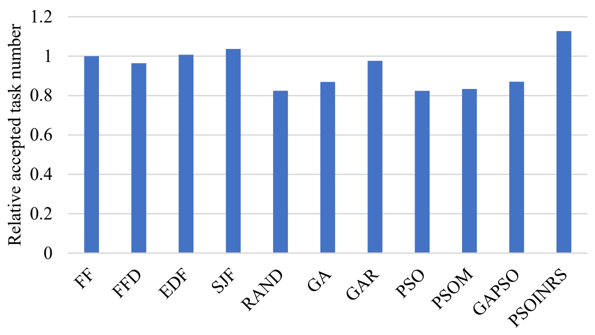


Fig. 2. The accepted task numbers achieved by various methods.

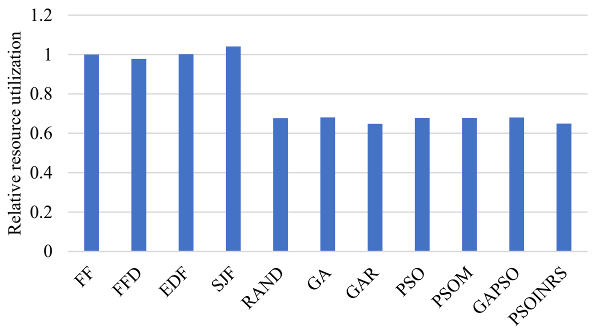


Fig. 3. The resource utilizations achieved by various methods.

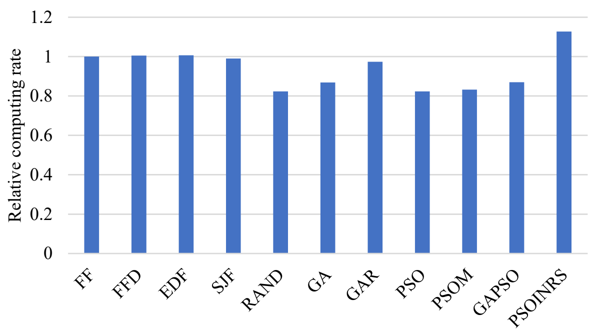


Fig. 4. The computing rates achieved by various methods.

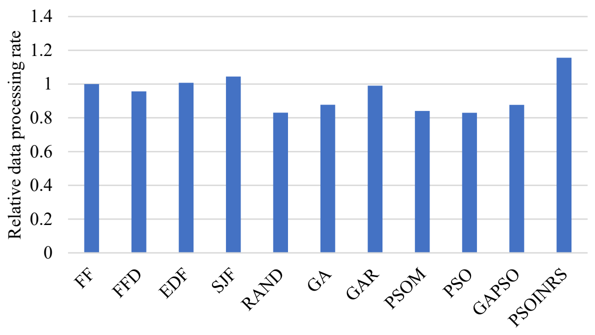


Fig. 5. The data processing rates achieved by various methods.

Fig. 4 and 5 presents the rates of the computing and the data processing when applying these offloading methods. From these figures, we can see that our proposed algorithm has the fast processing rate. Our algorithm has 12.0%–36.9% faster computing rate, and 10.7%–39.6% faster data processing rate than others.

Next, we evaluate the performance of our improvement strategies, the imbalance initialization and the task rescheduling, and the experimental results are shown in Fig. 6 and 7, where PSOIN and PSORS are respectively PSO with the imbalance initialization and the task rescheduling. From these figures, we can see that the imbalance initialization can improve GA by 1.73% and PSORS 12.1%, which verifies the validity of the imbalance initialization. In Fig 7, we can see that offloading algorithms with the task rescheduling can achieve 18.5%–189% better performance then that without it in processing rate. Thus, the task rescheduling strategy is deserved to be integrated into offloading algorithms.

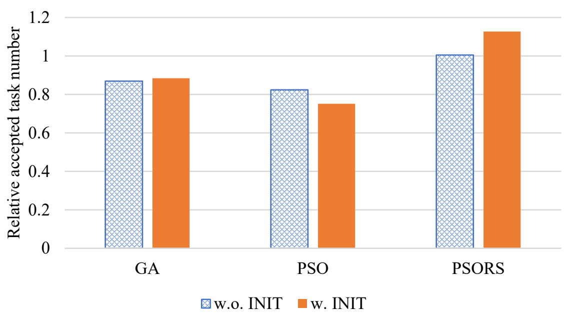
From Fig. 6 and 7, we also can see that the imbalance initialization can decrease the performance of PSO, while the task rescheduling can make up the degradation and improve both the accepted task number, the utilization and the processing rate. This inspires us that the combination of multiple improvement strategies may produce good solutions, even though a signal improvement strategy degrades the overall performance.

V. Conclusion

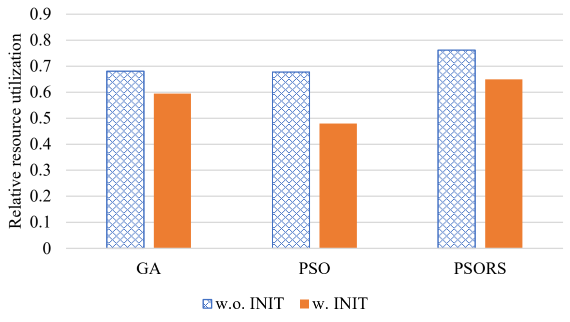
In this paper, we focus on the task offloading problem in DECC systems. We first formulate the problem as a discrete non-convex optimization model, which is hard to be solved. Then, we propose a PSO-based algorithm to solve the task offloading problem, where we use the imbalance initialization and the task rescheduling to improve the performance of PSO on the solving offloading problem in DECC. Extensive experiments are conducted and results verify the superior performance of our proposed algorithm.

Acknowledgment

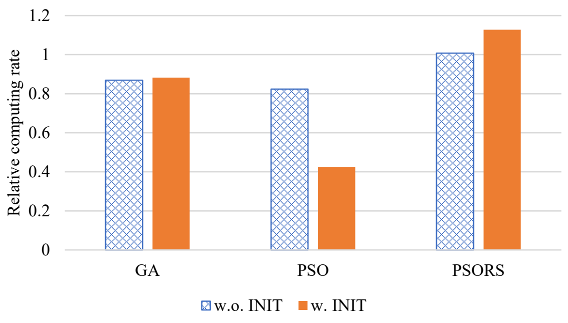
The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The research was supported by the key scientific and technological projects of Henan Province (Grant No. 232102211084, 232102210023, 232102210125), the Key Scientific Research Projects of Henan Higher School (Grant No. 22A520033), Zhengzhou Basic Research and Applied Research Project (ZZSZX202107) and China Logistics Society (2022CSLKT3-334).



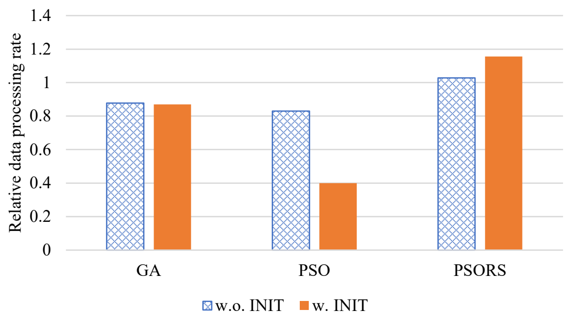
(a) Case I



(b) Case II

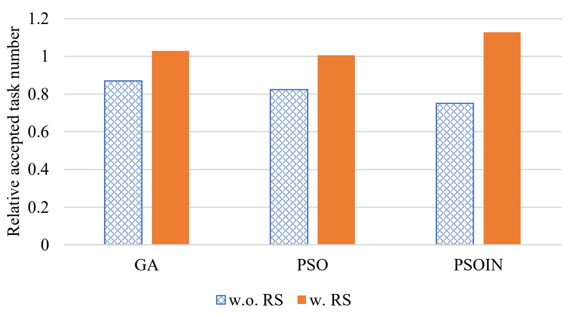


(c) Case III

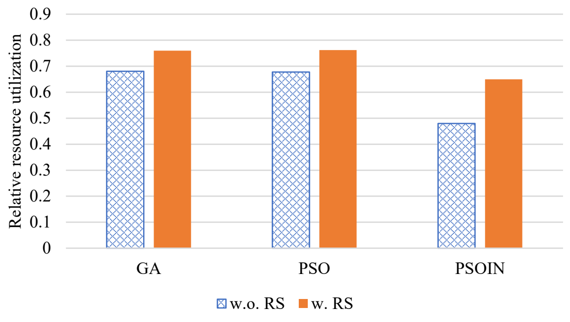


(d) Case IV

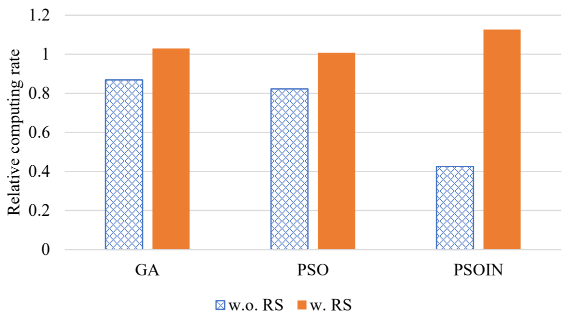
Fig. 6. The improvement of the imbalance initialization.



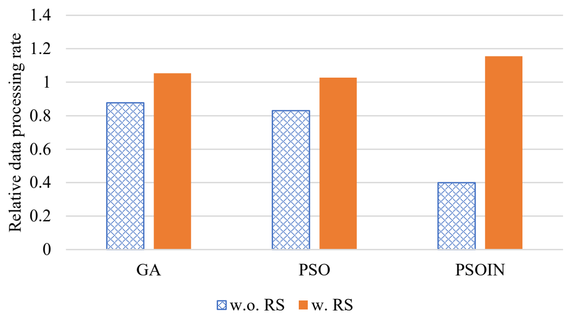
(a) Case I



(b) Case II



(c) Case III



(d) Case IV

Fig. 7. The improvement of the task rescheduling.

References

[1] K. Cao, Y. Liu, G. Meng and, Q. Sun. An Overview on Edge Computing Research. IEEE Access, 2020, 8: 85714-85728. Doi: 10.1109/ACCESS.2020.2991734.

[2] N. Fernando, S. W. Loke, and W. Rahayu. Mobile cloud computing: A survey. Future Generation Computer Systems, 2013, 29(1): 84-106. Doi: 10.1016/j.future.2012.05.023.

[3] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin. A Survey and Taxonomy on Task Offloading for Edge-Cloud Computing. IEEE Access, 2020, 8: 186080-186101. Doi: 10.1109/ACCESS.2020.3029649.

[4] C. Wang, R. Guo, H. Yu, Y. Hu, C. Liu, and C. Deng. Task offloading in cloud-edge collaboration-based cyber physical machine tool, Robotics and Computer-Integrated Manufacturing, 2023, 79: 102439. Doi: 10.1016/j.rcim.2022.102439.

[5] Y. Sang, J. Cheng, B. Wang, and M. Chen. A three-stage heuristic task scheduling for optimizing the service level agreement satisfaction in device-edge-cloud cooperative computing. PeerJ Computer Science, 2022, 8:e851. Doi: 10.7717/peerj-cs.851.

[6] X. Chen, T. Gao, H. Gao, B. Liu, M. Chen, and B. Wang. A multi-stage heuristic method for service caching and task offloading to improve the cooperation between edge and cloud computing. PeerJ Computer Science, 2022, 8:e1012. Doi: 10.7717/peerj-cs.1012.

[7] B. Wang, B. Lv, and Y. Song. A Hybrid Genetic Algorithm with Integer Coding for Task Offloading in Edge-Cloud Cooperative Computing. IAENG International Journal of Computer Science, 2022, 49(2): 503-510.

[8] S. Song, S. Ma, J. Zhao, F. Yang, and L. Zhai. Cost-efficient multi-service task offloading scheduling for mobile edge computing. Applied Intelligence, 2022, 52(4): 4028-4040. Doi: 10.1007/s10489-021-02549-2.

[9] M. A. Alqarni, M. H. Mousa, and M. K. Hussein. Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing. Journal of King Saud University - Computer and Information Sciences, 2022, 34(10-Part B): 10356-10364. Doi: 10.1016/j.jksuci.2022.10.026.

[10] B. Wang, J. Cheng, J. Cao, C. Wang, and W. Huang. Integer particle swarm optimization based task scheduling for device-edge-cloud cooperative computing to improve SLA satisfaction. PeerJ Computer Science, 2022, 8: e893. Doi: 10.7717/peerj-cs.893.

[11] M. P. John Mahenge, C. Li, and C. A. Sanga. Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications. Digital Communications and Networks, 2022, 8(6): 1048-1058. Doi: 10.1016/j.dcan.2022.04.001.

[12] H. Hafsi, H. Gharsellaoui, and S. Bouamama. Genetically-modified Multi-objective Particle Swarm Optimization approach for high-performance computing workflow scheduling. Applied Soft Computing, 2022, 122: 108791. Doi: 10.1016/j.asoc.2022.108791.

[13] B. Wang, P. Wu, and M. Arefzaeh. A new method for task scheduling in fog-based medical healthcare systems using a hybrid nature-inspired algorithm. Concurrency and Computation: Practice and Experience, 2022, 34(22): e7155. Doi: 10.1002/cpe.7155.

[14] N. E. Nwogbaga, R. Latip, L. S. Affendey, and A. R. Abdul Rahiman. Attribute reduction based scheduling algorithm with enhanced hybrid genetic algorithm and particle swarm optimization for optimal device selection. Journal of Cloud Computing, 2022, 11: 15. Doi: 10.1186/s13677-022-00288-4.

[15] H. Zhou, Z. Zhang, D. Li and Z. Su. Joint Optimization of Computing Offloading and Service Caching in Edge Computing-based Smart Grid. IEEE Transactions on Cloud Computing, 2022, In Press. Doi: 10.1109/TCC.2022.3163750.

[16] H. Hu, and P. Wang. Computation Offloading Game for Multi-Channel Wireless Sensor Networks. Sensors, 2022, 22(22):8718. Doi:10.3390/s22228718

[17] M. L. Pinedo. Scheduling Theory, Algorithms, and Systems, Fifth Edition. Springer International Publishing AG Switzerland, 2016.

[18] lpsolve: Mixed Integer Linear Programming (MILP) Solver. 2021. Available online: https://sourceforge.net/projects/lpsolve/ (accessed on 31 March 2023).

[19] MathWorks, Inc. Optimization Toolbox: Solve Linear, Quadratic, Conic, Integer, and Nonlinear Optimization Problems. 2022. Availabe online: https://ww2.mathworks.cn/en/products/optimization.html (accessed on 31 March 2023).

[20] A. A. Hussain and F. Al-Turjman. Hybrid Genetic Algorithm for IOMT-Cloud Task Scheduling. Wireless Communications and Mobile Computing, 2022, 2022: 6604286. Doi: 10.1155/2022/6604286