2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

Profit-Maximized Task Offloading with Simulated-annealing-based Migrating Birds Optimization in Hybrid Cloud-Edge Systems

Haitao Yuan[1], Jing Bi[2], MengChu Zhou[3], Jia Zhang[4] and Wei Zhang[5]

[1]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[2]Faculty of lnformation Technology, Beijing University of Technology, Beijing 100124, China
[3]Dept. of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA
[4]Dept. of Computer Science, Southern Methodist University, Dallas, TX 75275, USA
[5]Microsoft Redmond, WA 98052, USA

***Abstract*—As an emerging framework, edge computing achieves Internet of Things by providing computing, storage and network resources. It moves computation to edge devices located near users. Nevertheless, nodes in the edge often own limited resources and constrained energy capacities. It is impossible to entirely execute tasks in the edge due to their unsatisfied quality of service. Cloud data centers (CDCs) own almost unlimited resources yet they might cause large transmission delay and high resource cost. Consequently, it is highly needed to intelligently offload tasks between CDC and edge. This work proposes a task offloading algorithm for hybrid cloud-edge systems to achieve profit maximization of a system provider with response time bound assurance. It comprehensively investigates CPU, memory and bandwidth limits of nodes in the edge, and constraints of available energy and servers in CDC. These factors are integrated into a single-objective constrained optimization problem, which is solved by a simulated-annealing-based migrating birds optimization algorithm to yield a close-to-optimal offloading policy between CDC and the edge. Real-life data-driven experimental results show that its profit outperforms its four typical peers.**

***Index Terms*—Cloud data centers, cloud computing, edge computing, intelligent optimization, task offloading, simulated annealing, and migrating birds optimization**

I. Introduction

Smart mobile devices (SMDs) have been gaining enormous attention with advanced mobile technologies, *e.g.*, wearable devices and Internet of Things (IoTs) devices [1]. These technologies give a powerful platform for smart mobile applications, *e.g.*, speech/face recognition and health monitoring. The gap between limited computing resources and need for running applications is growing [2]. It is challenging to execute them in SMDs that own limited computation resources and battery capacities. It is shown that cloud data centers (CDCs), *e.g.*, Amazon EC2, and Microsoft Azure, provide distributed computing to efficiently tackle the limitation of battery and processing capabilities by offloading part or all computation-intensive tasks to CDCs [3]. In 2019, over 70% of calculations were completed in CDCs [4]. However, CDCs are usually far away from SMDs, and this leads to unacceptable communication delay and economic cost for utilizing resources in CDCs [5], [6].

To tackle the shortcomings of using CDCs, edge computing is an emerging architecture for the network, and it provides pervasive resources to IoT applications with strict latency needs anytime and anywhere [7]. The need for real-time and scalable data analysis in IoT devices is a major driving force for edge computing where data is generated and processed in network edge. It is shown that about 40% of IoT-produced data is stored and processed in the edge [8]. Local computing in the edge greatly reduces the response latency because waiting or communication delay between edge and CDCs is avoided. Yet, limits of energy, computation and storage resources of nodes in IoT restrict the number of tasks of resource-hungry applications [9] locally computed in the edge. It is unlikely to execute all tasks in nodes of edge (*i.e.*, local computing), and some of them have to be offloaded to CDCs to avoid energy depletion and performance degradation. Thus, it is critically important to rationally schedule all tasks between CDC and the edge, and maximize the profit of hybrid cloud-edge systems while ensuring response time limits of tasks.

To solve this issue, this work aims to maximize the profit of a system provider by smartly offloading tasks between CDC and the edge. It explicitly specifies the task service rate of servers in CDC, and determines the node for processing each task in the edge. This work proposes a fine-grained mechanism to obtain an optimal offloading strategy for tasks. It jointly considers CPU, memory and bandwidth limits, and processing capacities of nodes in the edge. In addition, it jointly considers limits of energy and servers in CDC. By investigating these factors, this work proposes a profit-maximized task offloading algorithm for maximizing the system profit and enforcing response time limits of tasks to be strictly met. Real-life data, *e.g.*, tasks in Google cluster and prices of power grid are

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61802015 and 61703011, in part by the Major Science and Technology Program for Water Pollution Control and Treatment of China under Grant 2018ZX07111005, and in part by the National Defense Pre-Research Foundation of China under Grants 41401020401 and 41401050102.

adopted to evaluate it. The experiments prove that it achieves higher profit than its typical scheduling peers.

II. Problem Formulation

This section formulates a constrained optimization problem for a cloud and edge computing system shown in Fig. 1. The framework includes three layers, *i.e.*, terminal, the edge, and CDC. Users' tasks are sent through heterogeneous SMDs, *e.g.*, iPad, smart phones, computers, sensors on the road, and all they are produced at the terminal layer, and returned results after their completion are sent back to this layer eventually. As shown in Fig. 1, the wireless infrastructure mainly includes many WiFi access points and multiple small-cell base stations (SBSs), which are mutually interconnected to transmit messages. Tasks are delivered to the edge, including a task scheduler and nodes in the edge. The scheduler smartly allocates tasks between the edge and CDC. If a task is scheduled to CDC, a macro base station (MBS) just forwards it to CDC; otherwise, an MBS needs to execute it in its node. MBS is located at the edge, and consists of many heterogeneous nodes with limited storage, computing and transmission abilities. It receives tasks from the terminal layer, and executes some of them locally and sends others to CDC through Internet backbone. It reduces tasks' latency in CDC, which owns multiple interconnected server clusters with large computing and storage capacities, and provides different cloud resources to handle tasks.

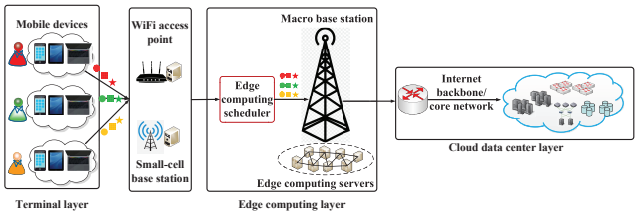

Fig. 1. Illustrative system framework.

A. Decision variables

Let $o_\tau^{i,j}$ be a binary variable. If task i is scheduled to execute in node j in the edge in time slot τ, $o_\tau^{i,j}=1$; otherwise, $o_\tau^{i,j}=0$. Let I_τ denote the number of tasks scheduled to nodes in the edge in time slot τ. Let μ_τ denote the task service rate of CDC servers in time slot τ. I_τ needs to be less than or equal to the total number of arriving tasks in time slot τ, $\lambda_\tau\gamma$, *i.e.*,

$$I_\tau \le \lambda_\tau \gamma \tag{1}$$

where λ_τ is the task service rate of CDC in time slot τ, and γ is the length of each time slot.

Let n_τ^j denote the maximum number of tasks that can be executed by node j in the edge in time slot τ. Thus,

$$\sum_{i=1}^{I_\tau} o_\tau^{i,j} \le n_\tau^j \tag{2}$$

In time slot τ, task i has to be and can only be scheduled to execute in only one node in the edge. Thus,

$$\sum_{j=1}^{J} o_\tau^{i,j} = 1 \tag{3}$$

B. Response time

The total number of arriving tasks in time slot τ is $\lambda_\tau\gamma$ according to (1). Then, the number of tasks scheduled to execute in CDC in time slot τ is $\lambda_\tau\gamma - I_\tau$. Let λ_τ^c denote the arriving rate of tasks scheduled to CDC in time slot τ, *i.e.*, $\lambda_\tau^c = \frac{\lambda_\tau\gamma - I_\tau}{\gamma}$. Let $\hat{T}_\tau$ denote the maximum response time of tasks executed in all nodes in the edge in time slot τ. Let $\widetilde{T}_\tau$ denote the average response time of all tasks executed in CDC in time slot τ. In the edge, there are J heterogeneous nodes. Let $T_\tau^{i,j}$ denote the execution time of task i ($1 \le i \le I_\tau$) on node j ($1 \le j \le J$). Let k_i denote the size of task i. Let p_j denote the processing speed of node j. Then,

$$T_\tau^{i,j} = \frac{k_i}{p_j} o_\tau^{i,j} \tag{4}$$

The maximum response time ($\hat{T}_\tau$) of all tasks executed in the edge includes transmission and computation time in CDC. Then, $\hat{T}_\tau$ is obtained as:

$$\hat{T}_\tau = \max_{j=1}^{J} \left[\sum_{i=1}^{I_\tau} T_\tau^{i,j} \right] \tag{5}$$

Let T^+ denote users' response time limit for tasks scheduled to execute in the edge. Thus, $\hat{T}_\tau$ cannot exceed T^+, *i.e.*,

$$\hat{T}_\tau \le T^+ \tag{6}$$

Similar to [10], [11], this work adopts an $M/M/1/\beta/\infty$ queuing system to analyze the behavior of switched-on servers in CDC. Here, β denotes the maximum number of tasks that all servers in CDC can execute. It is assumed that users' tasks arrive in a Poisson process with mean rate λ_τ^c and task service time has an exponential distribution with mean rate μ_τ. Let $\widetilde{T}_\tau$ denote the average response time of tasks scheduled to execute in CDC. Let T^c denote users' response time limit of tasks scheduled to execute in CDC. Then, $\widetilde{T}_\tau$ is obtained as:

$$\widetilde{T}_\tau = d_\tau + \frac{\Psi_\tau}{\mu_\tau (1 - Q_\tau^0)} \tag{7}$$

where

$$\Psi_\tau = \frac{\rho_\tau}{1-\rho_\tau} - \frac{(\beta+1)(\rho_\tau)^{\beta+1}}{1-(\rho_\tau)^{\beta+1}},$$

$$Q_\tau^0 = \frac{1-\rho_\tau}{1-(\rho_\tau)^{\beta+1}},$$

$$\rho_\tau = \frac{\lambda_\tau^c}{\mu_\tau},$$

d_τ is the total transmission time of input/output data to/from CDC through MBS, and Q_τ^0 is the possibility when there are

no tasks in CDC in time slot τ. In addition, to guarantee the stability of a task queue in CDC, we have:

$$\lambda_\tau^c \leq \mu_\tau \tag{8}$$

In time slot τ, $\widetilde{T}_\tau$ cannot exceed its limit T^c, *i.e.*,

$$\widetilde{T}_\tau \leq T^c \tag{9}$$

C. Profit

Then, let F_τ, $\beth_\tau$ and Γ_τ denote total revenue, total cost and total profit in time slot τ. Then,

$$\Gamma_\tau = F_\tau - \beth_\tau \tag{10}$$

We use Service Level Agreements (SLAs) [12] that specify the revenue or penalty if the response time of tasks is within or beyond its limits. Here, if the actual response time of a task is less than or equal to 0.1 seconds, its revenue is 0.5 \$; otherwise, its penalty is 0.2 \$. The revenue brought by each task scheduled to execute in the edge is $f(\hat{T}_\tau)$. Then, the total revenue brought by all tasks scheduled to the edge is $f(\hat{T}_\tau) I_\tau$. Similarly, the revenue brought by each task scheduled to CDC is $f(\widetilde{T}_\tau)$. Then, the revenue brought by all tasks scheduled to CDC is $f(\widetilde{T}_\tau)\left(\lambda_\tau \gamma - I_\tau\right)$. Hence,

$$F_\tau = f(\hat{T}_\tau) I_\tau + f(\widetilde{T}_\tau)\left(\lambda_\tau \gamma - I_\tau\right) \tag{11}$$

Let $\beth_\tau^E$ denote the execution cost of all tasks scheduled to execute in the edge in time slot τ. Let $\beth_\tau^C$ denote the energy cost of all tasks scheduled to execute in CDC in time slot τ. Then, $\beth_\tau$ consists of $\beth_\tau^E$ and $\beth_\tau^C$. Let $\eta_\tau^{i,j}$ denote the execution cost of task i scheduled to execute in node j in the edge in time slot τ. Let $\widehat{\eta}$ denote the upper limit of $\eta_\tau^{i,j}$.

$$\eta_\tau^{i,j} \leq \widehat{\eta} \tag{12}$$

$\beth_\tau^E$ is obtained as $\sum_{i=1}^{I_\tau} \sum_{j=1}^{J} \left(o_\tau^{i,j} \eta_\tau^{i,j}\right)$. Let e_τ denote the price of power grid in time slot τ. Let E_τ denote the amount of energy consumed by all tasks scheduled to execute in CDC in time slot τ. Thus, $\beth_\tau^E = e_\tau E_\tau$. Similar to [13], [14], the data transmission cost between CDC and edge is ignored. Thus,

$$\beth_\tau = e_\tau E_\tau + \sum_{i=1}^{I_\tau} \sum_{j=1}^{J} o_\tau^{i,j} \eta_\tau^{i,j} \tag{13}$$

D. Energy consumption

$\widehat{P}$ and $\widetilde{P}$ denote the peak and idle power of each server in CDC, respectively. σ denotes the number of tasks processed by each powered-on server per time in CDC, and χ denotes the power usage effectiveness value [15] of CDC. Following [16], energy consumption E_τ is calculated as:

$$E_\tau = \frac{\nu \mu_\tau + \psi \lambda_\tau^c \left(1 - q\left(\lambda_\tau^c, \mu_\tau\right)\right)}{\sigma} \gamma \tag{14}$$

$$\nu = \widetilde{P} + (\chi - 1)\, \widehat{P}$$

$$\psi = \widehat{P} - \widetilde{P}$$

$$\delta(\lambda_\tau^c, \mu_\tau) = \frac{1 - \rho_\tau}{1 - (\rho_\tau)^{\beta+1}} (\rho_\tau)^\beta$$

Let Δ denote the maximum amount of the total available energy in CDC. Then, E_τ cannot exceed Δ, *i.e.*,

$$\frac{\nu \mu_\tau + \psi \lambda_\tau^c \left(1 - q\left(\lambda_\tau^c, \mu_\tau\right)\right)}{\sigma} \gamma \leq \Delta \tag{15}$$

The execution cost of all tasks scheduled to execute in the edge needs to be less than or equal to the energy cost of all tasks scheduled to execute in CDC in time slot τ, *i.e.*,

$$\sum_{i=1}^{I_\tau} \sum_{j=1}^{J} o_\tau^{i,j} \eta_\tau^{i,j} \leq e_\tau E_\tau \tag{16}$$

Let ϖ denote the maximum number of servers in CDC. Then, the number of powered-on servers is μ_τ / σ, satisfying:

$$\frac{\mu_\tau}{\sigma} \leq \varpi \tag{17}$$

E. Resource limits in the edge

This work considers three important resources including CPU, memory and bandwidth in the edge [17]. $C_\tau^{u,j}$, $M_\tau^{u,j}$ and $B_\tau^{u,j}$ denote the utilization of CPU, memory and bandwidth resources of node j in time slot τ, respectively, and they cannot exceed 1, satisfying:

$$C_\tau^{u,j} = \frac{\sum_{i=1}^{I_\tau} o_\tau^{i,j} \varsigma_i}{\widetilde{C}} \leq 1 \tag{18}$$

$$M_\tau^{u,j} = \frac{\sum_{i=1}^{I_\tau} o_\tau^{i,j} \varphi_i}{\widetilde{M}} \leq 1 \tag{19}$$

$$B_\tau^{u,j} = \frac{\sum_{i=1}^{I_\tau} o_\tau^{i,j} \xi_i}{\widetilde{B}} \leq 1 \tag{20}$$

where ς_i, φ_i and ξ_i denote the amount of CPU, memory and bandwidth resources needed by task i, respectively. $\widetilde{C}$, $\widetilde{M}$ and $\widetilde{B}$ are the maximum amount of CPU, memory and bandwidth resources in node j, respectively.

F. Profit maximization problem

The objective is to maximize Γ_τ, *i.e.*,

$$\underset{o_\tau^{i,j}, I_\tau, \mu_\tau}{\textbf{Max}} \quad \{\Gamma_\tau\} \tag{21}$$

subject to (1), (2), (3), (6), (8), (9), (12), (15), (16), (17), (18)–(20) and (22).

$$I_\tau \in N^+,\ \mu_\tau \geq 0,\ o_\tau^{i,j} \in \{0, 1\} \tag{22}$$

III. Simulated-annealing-based Migrating Birds Optimization (SMBO)

To solve (21), we can use many nature-inspired metaheuristic-based optimization algorithms [18], *e.g.*, tabu search, genetic algorithm and particle swarm optimization, to find near optimal solutions. We adopt a metaheuristic algorithm named Migrating Birds Optimization (MBO) is proposed in [19] as a basic algorithm. It is inspired and derived from an effective V flight formation of migrating birds (solutions),

which brings benefits in energy saving because birds in other positions get benefit from the birds in front. Fig. 2 shows a 7-solution V flight formation where bird 1 is the leading bird.

In MBO, there is a leader bird and two lines of other birds follow it. The leader bird becomes tired after flying for a certain time, it then flies to the end of a line and one of other following birds becomes a new leader. Each solution compares itself with a number of its own neighbors, and several best neighbors of the front solution. It is replaced by the best of them if it becomes worse. Then, the leader solution goes to the last position, and one of the second solutions goes to the first position. In MBO, the neighborhood within more promising solutions is explored in more details. After several iterations, these solutions might move to different directions if they are improved along their ways. However, finally most solutions of MBO may converge to one or several local optima. Simulated annealing (SA) can conditionally jump from local optima by moving to some worse solutions with the Metropolis acceptance rule [20]. SA can thus obtain global optima with large probability in theory, and is widely used to find high-quality solutions to different problems. This work proposes a hybrid algorithm called SA-based Migrating Birds Optimization (SMBO) by combining SA's Metropolis acceptance rule and MBO. Specifically, this work adopts the SA-based update mechanism to change a solution. The flowchart of SMBO is shown in Fig. 3.

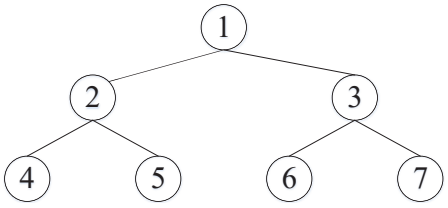

Fig. 2. A 7-solution V flight formation.

IV. Performance Evaluation

This work evaluates the proposed SMBO with real-life data. SMBO is implemented with MATLAB 2017 in a computer with an Intel Xeon CPU with 2.4 GHz and a 32-GB memory.

A. Parameter setting

This work uses realistic tasks collected from Google cluster trace[1] to evaluate the proposed method. Fig. 4 illustrates the task arriving rate in one day. In addition, this work uses realistic price of power grid collected from the New York Independent System Operator[2]. Fig. 5 illustrates the price of power grid in one day. Besides, the length of one time slot is 300 seconds, *i.e.*, γ=300 seconds.

The following parameters are set as follows. J=30, $\widetilde{B}$=3000 MB/s, $\widetilde{M}$=1024 MB and $\widetilde{C}$=2048 MIPS. In addition, ξ_i, φ_i and ς_i are randomly produced in (0,0.1). In addition, σ=0.05 tasks/second, ϖ=2×10^3, Δ=1×10^5 WH, $\widehat{P}$=600 W, $\widetilde{P}$=300 W, χ=1.6 and $\widehat{\eta}$=0.01 \$. Besides, d_τ is set to $\frac{1}{5}\Psi_\tau/\left(\mu_\tau\left(1-Q_\tau^0\right)\right)$. $\eta_\tau^{i,j}$ is randomly produced in (0,0.01). T^c=T^+=0.1 seconds, β=30, and n_τ^j=1000. Besides, $k_i\in(1\times10^6,\ 8\times10^6)$. $p_j\in(1\times10^{11},\ 2\times10^{11})$. The parameters

[1]https://github.com/google/cluster-data
[2]https://www.nyiso.com/

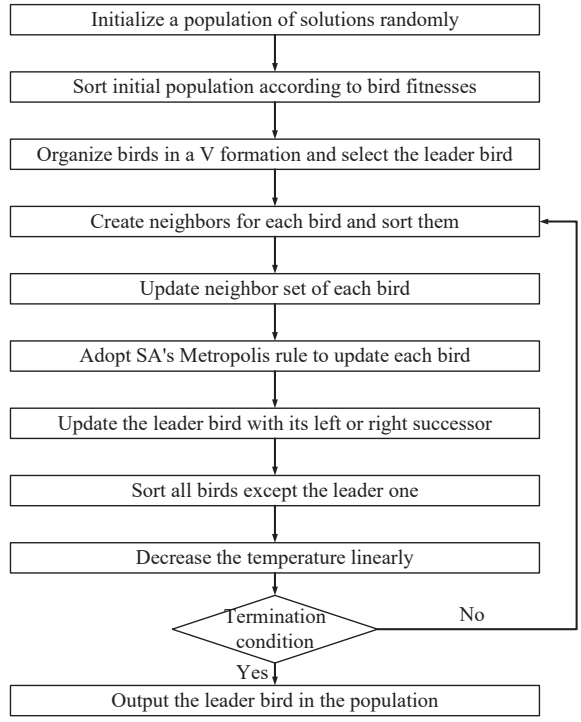

Fig. 3. Flowchart of SMBO.

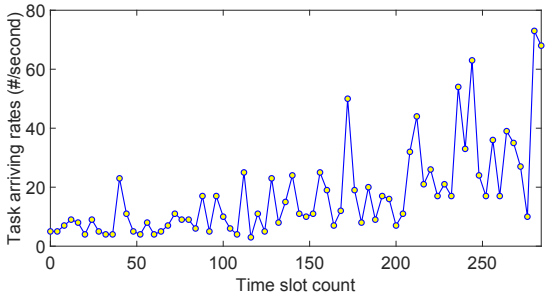

Fig. 4. Task arriving rate.

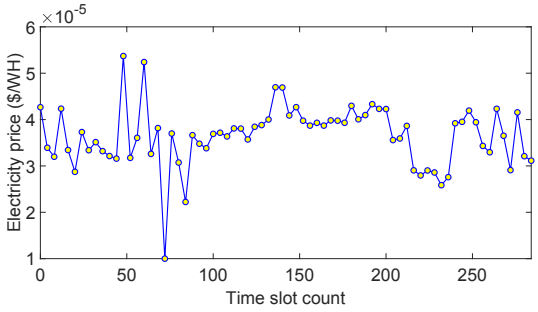

Fig. 5. Price of power grid in one day.

of SMBO are set as follows. The population size is 51. The number of neighbors for the leader solution is 3. The total number of iterations of SBA is 132651. The initial temperature of SMBO is 5×10^6 and its cooling rate is 0.985.

B. Experimental results

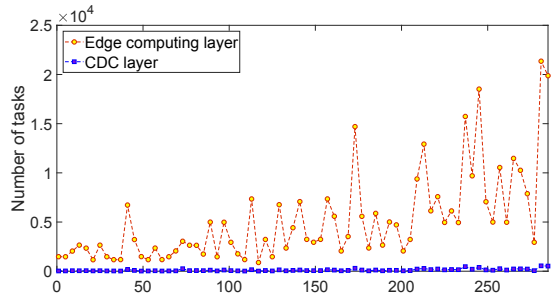

Fig. 6. Number of tasks scheduled to edge and CDC.

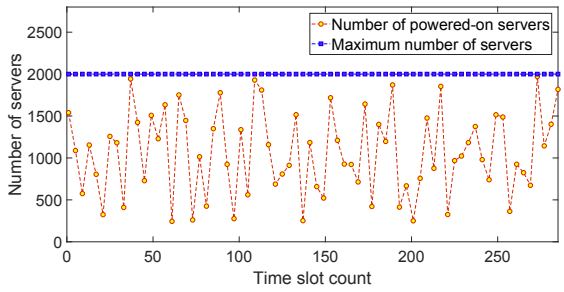

Fig. 7. Number of powered-on servers in CDC.

Fig. 6 shows the number of tasks scheduled to the edge and CDC. It is shown that the number of tasks scheduled to CDC is much lower than that scheduled to the edge in each time slot. The reason is that nodes in the edge are much closer to users and can avoid the transmission delay of input/output data to/from CDC through MBS. Fig. 7 shows the number of powered-on servers in CDC. The maximum number of servers in CDC is 2000, *i.e.*, ϖ=2000. It is shown that the number of powered-on servers in CDC is less than its limit in each time slot.

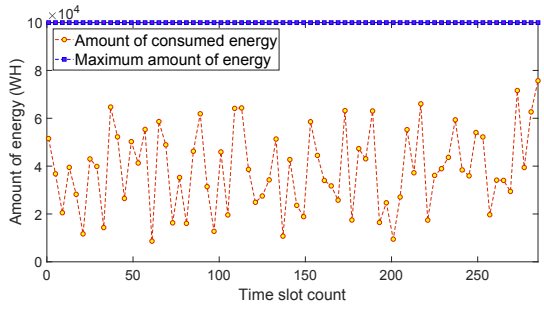

Fig. 8. Amount of energy consumed in CDC.

Fig. 8 shows the amount of energy consumed in CDC in each time slot. It is shown that the amount of energy consumed in CDC is less than its limit in each time slot. Fig. 9 shows the total response time of each task executed in CDC and the edge. Therefore, it is shown that the total response time of each task executed in CDC and the edge is less than its limit in each time slot, *i.e.*, (6) and (9) are both met in each time slot. Hence, the execution results with our obtained schedule can strictly meet response time limits of tasks.

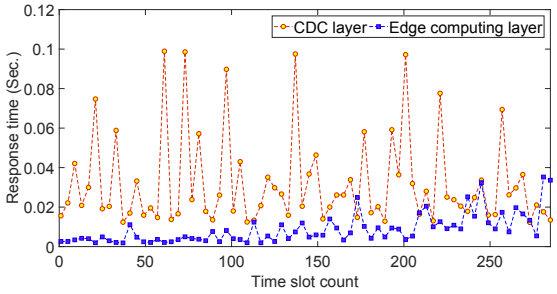

Fig. 9. Total response time of each task in CDC and edge.

C. Comparison results

This work compares it with firefly algorithm (FA) [21] and Genetic Learning Particle Swarm Optimization (GL-PSO) [22], local computing and entire offloading. SMBO, FA and GL-PSO are repeated independently for 30 times to produce their respective results.

1) FA [21]: FA can efficiently find high-quality optima of multimode functions. Its convergence is fast, but it suffers from a premature convergence problem.
2) GL-PSO [22]: GL-PSO applies a learning mechanism of a genetic algorithm to construct exemplars hybridized with particle swarm optimization in a cascade manner.
3) Local computing. All tasks are executed by nodes in the edge.
4) Entire offloading. All tasks are offloaded to CDC.

The comparison between SMBO and FA can demonstrate the former's convergence speed. The comparison between it and GL-PSO can demonstrate its solution accuracy. SMBO, FA and GL-PSO stop their search processes if their solutions are not improved in 10 consecutive iterations.

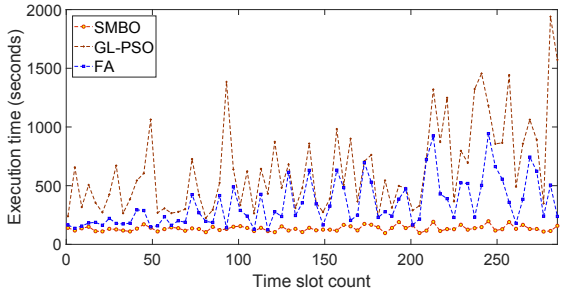

Fig. 10. Convergence time comparison of SMBO, FA and GL-PSO.

Fig. 10 shows the convergence time comparison of SMBO, FA and GL-PSO in each time slot. It is shown that compared to FA and GL-PSO, SMBO's average convergence time of all time slots is reduced by 49.26% and 72.21% on average, respectively. Therefore, SMBO increases the profit in less time and fewer iterations than FA and GL-PSO. Fig. 10 proves that the adoption of SA's Metropolis acceptance rule increases the diversity of population and search performance.

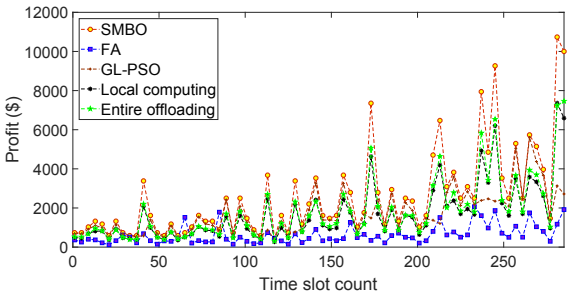

Fig. 11. Profit of SMBO, FA, GL-PSO, local computing and entire offloading.

Fig. 11 illustrates the profit comparison of SMBO, FA, GL-PSO, local computing and entire offloading. It is shown that compared with its four peers, its profit is increased by 66.83%, 21.32%, 34.81% and 30.22% on average, respectively. The solution accuracy of SMBO is higher than those of FA, GL-PSO, and therefore the profit of SMBO is larger than those of FA and GL-PSO. Battery energy, CPU, memory and bandwidth resources are all limited in the edge, and energy, servers, *etc.* in CDC are also limited. Thus, tasks scheduled with local and entire offloading need to wait for execution and suffer from higher latency than SMBO, resulting in bad user experience and low profit. The results validate that the proposed offloading method outperforms four peers.

V. Conclusion

The emerging edge computing is widely implemented because of its quick response and local processing capacities. Nevertheless, its computing, storage and network resources, and energy are not enough t execute all tasks. Cloud data centers (CDCs) have sufficient resources for users to utilize but they are far away from users in the edge. Thus, it causes additional latency and consumes energy to deliver data between CDC and edge. Therefore, edge computing provides an alternative way to partially offload its tasks to CDC for achieving profit maximization of hybrid cloud-edge systems while meeting tasks' response time requirements. To achieve it, this work designs a profit-maximized task offloading algorithm. In each time slot, a single-objective constrained optimization problem is given and further addressed by a newly designed Simulated-annealing-based Migrating Birds Optimization (SMBO) algorithm. Real-world data-driven results demonstrate that the obtained offloading strategy yields higher profit than its four benchmark methods.

References

[1] J. Bi, H. Yuan and S. Duanmu, "Genetic Particle Swarm Optimization-based Energy-optimized Partial Computation Offloading in Mobile Edge Computing," in *Proc. The 3rd IFAC Workshop on Cyber-Physical & Human Systems*, Shanghai, China, 2020, pp. 1–6.

[2] P. Wang, C. Yao, Z. Zheng, G. Sun and L. Song, "Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2872–2884, Apr. 2019.

[3] L. Liu, Z. Chang, X. Guo, S. Mao and T. Ristaniemi, "Multiobjective Optimization for Computation Offloading in Fog Computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[4] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Trans. on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[5] Y. Lin, E. T. -. Chu, Y. Lai and T. Huang, "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds," *IEEE Systems Journal*, vol. 9, no. 2, pp. 393–405, Jun. 2015.

[6] H. Yuan, J. Bi, W. Tan and B. H. Li, "Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds," *IEEE Trans. Aut. Sci. and Eng.*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[7] M. Gaggero and L. Caviglione, "Model Predictive Control for Energy-Efficient, Quality-Aware, and Secure Virtual Machine Placement," *IEEE Trans. Automation Sci. and Eng.*, vol. 16, no. 1, pp. 420–432, Jan. 2019.

[8] X. T. R. Kong, S. X. Xu, M. Cheng and G. Q. Huang, "IoT-Enabled Parking Space Sharing and Allocation Mechanisms," *IEEE Trans. Automation Sci. and Eng.*, vol. 15, no. 4, pp. 1654–1664, Oct. 2018.

[9] B. Bai, Y. Fan, W. Tan, and J. Zhang, "DLTSR: A Deep Learning Framework for Recommendation of Long-tail Web Services", *IEEE Trans. on Services Computing*, vol. 13, no. 1 pp. 73–85, Feb. 2020.

[10] J. Cao, K. Hwang, K. Li and A. Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Trans. on Parallel and Distributed Sys.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[11] J. Yao, H. Guan, J. Luo, L. Rao and X. Liu, "Adaptive Power Management through Thermal Aware Workload Balancing in Internet Data Centers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2400–2409, Sept. 2015.

[12] A. Barri and A. Dooms, "Data-Driven Modules for Objective Visual Quality Assessment Focusing on Benchmarking and SLAs," *IEEE Journal of Selected Topics in Signal Proc.*, vol. 11, no. 1, pp. 196–205, Feb. 2017.

[13] O. Muñoz, A. Pascual-Iserte and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Trans. on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[14] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Trans. on Comm.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[15] J. Conejero, O. Rana, P. Burnap, J. Morgan, B. Caminero, C. Carrión, "Analyzing Hadoop Power Consumption and Impact on Application QoS," *Future Generation Computer Sys.*, vol. 55, pp. 213–223, Feb. 2016.

[16] A. Kiani and N. Ansari, "Profit Maximization for Geographically Dispersed Green Data Centers," *IEEE Trans. on Smart Grid*, vol. 9, no. 2, pp. 703–711, Mar. 2018.

[17] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen and L. Xiao, "Learning Driven Computation Offloading for Asymmetrically Informed Edge Computing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1802–1815, Aug. 2019.

[18] Y. Du, C. Xu and D. Tao, "Matrix Factorization for Collaborative Budget Allocation," *IEEE Trans. on Automation Science and Engineering*, vol. 15, no. 4, pp. 1471–1482, Oct. 2018.

[19] E. Duman, M. Uysal, A. F. Alkaya, "Migrating Birds Optimization: A New Metaheuristic Approach and Its Performance on Quadratic Assignment Problem," *Information Sciences*, vol. 217, pp. 65–77, Dec. 2012.

[20] F. Javidrad, M. Nazari, "A New Hybrid Particle Swarm and Simulated Annealing Stochastic Optimization Method," *Applied Soft Computing*, vol. 60, pp. 634–654, Nov. 2017.

[21] K. Jagatheesan, B. Anand, S. Samanta, *etc.*, "Design of a Proportional-integral-derivative Controller for an Automatic Generation Control of Multi-area Power Thermal Systems Using Firefly Algorithm," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 503–515, Mar. 2019.

[22] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi, and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Trans. on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.