Research Article

A Task Offloading Method with Edge for 5G-Envisioned Cyber-Physical-Social Systems

Jielin Jiang,^{1,2,3} Xing Zhang,¹ and Shengjun Li ⁴

¹School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China

²Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, Nanjing, China

³Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing, China

⁴School of Information Science and Engineering, Qufu Normal University, Rizhao, China

Correspondence should be addressed to Shengjun Li; qfnulsj@163.com

Received 1 June 2020; Revised 5 July 2020; Accepted 21 July 2020; Published 7 August 2020

Academic Editor: Dou Wanchun

Copyright © 2020 Jielin Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, Cyber-Physical-Social Systems (CPSS) have been introduced as a new information physics system, which enables personnel organizations to control physical entities in a reliable, real-time, secure, and collaborative manner through cyberspace. Moreover, with the maturity of edge computing technology, the data generated by physical entities in CPSS are usually sent to edge computing nodes for effective processing. Nevertheless, it remains a challenge to ensure that edge nodes maintain load balance while minimizing the completion time in the event of the edge node outage. Given these problems, a Unique Task Offloading Method (UTOM) for CPSS is designed in this paper. Technically, the system model is constructed firstly and then a multi-objective problem is defined. Afterward, Improving the Strength Pareto Evolutionary Algorithm (SPEA2) is utilized to generate the feasible solutions of the above problem, whose aims are optimizing the propagation time and achieving load balance. Furthermore, the normalization method has been leveraged to produce standard data and select the global optimal solution. Finally, several necessary experiments of UTOM are introduced in detail.

1. Introduction

For the past few years, with the perpetual progress of Big Data, Cloud Computing, Internet of things (IoT), and other technologies, traditional physical systems and new information resources are further integrated, thus forming complex systems that incorporate machines, information, and human, namely, CPSS. They enable the physical system to have the functions of computation, communication, remote cooperation, etc., and make full use of social information and computing resources to carefully coordinate the physical system. CPSS make the machine more intelligent, as well as the personnel organization operating the physical entity in a more reliable, real-time, and security manner through the cyberspace. This also makes the development of the IoT more rapid and a variety of intelligent scene applications are broader [1, 2].

Nevertheless, with the increasing development of mobile devices, IoT devices, and multiple intelligent scenes (e.g., intelligent transportation, intelligent home, and intelligent cities), an increasing number of people have higher standards for applications in these scenarios [3]. When the physical system combines network information and social information, the mass data transmission characterized by multiple types and high speed also puts forward higher requirements for network communication (i.e., higher bandwidth and lower delay). This case conflicts with people's higher requirements for high-quality, low latency, and real-time network services. Thus, academia and industry urgently need to solve the problem of how to systematically and efficiently process the data, i.e., the historical data and the local real-time data, in CPSS. However, it makes little sense to consider the service outside the context of network performance. The 5G network with the purpose of

accelerating the evolution of smart applications scene can not only enhance the data delivery rates and lessen the latency but also advance the amounts of infrastructures in intelligent applications.

Technically, the implementation of the 5G network needs the support of edge computing technique [4]. Edge computing is an inevitable development in the evolution of base stations combined with IT as well as mobile network [5]. The most intuitive benefit that edge computing brings is the ability to improve the quality of experience through high bandwidth and instant response [6]. At the same time, quality of experience is becoming more prominent among the booming new services, which have become an essential part of mobile social and entertainment [7, 8].

To offer immediate and efficient feedback for the users in CPSS, there is no doubt that edge computing, a significant paradigm, with abundant computing resources, needs to be adequately taken advantage of for users to experience a high quality of service applications in real time [9]. It makes the user close to the nodes geographically where the resource is being processed, thus significantly reducing the delay of offloading tasks. Specifically, in CPSS, the base stations are evolved into edge nodes to service the task requests and data from the users who are covered in these nodes. In addition to its advantages in offloading tasks, edge computing shortens the distance between people and processing nodes, making the traditional interception of information much less likely to cause harm to users, thus improving the user's security [10].

However, in the hybrid CPSS scenario, where multiple systems are involved, offloading tasks to reasonable nodes is a complex problem [11]. Thus, how to determine the offloading node of the computing tasks is a challenge in the CPSS scenario [12]. Also, since the number of nodes is limited and each node has restricted computing resources, resource utilization needs to be taken into consideration and to be promoted as much as possible [13, 14]. Under the premise of improving performance as much as possible, load balance, as a critical indicator, should be taken into account to ensure the stability of each node, because it reflects the overall efficiency and performance of the system [15, 16].

Based on the above discussion and questions, a unique task offload method in CPSS based on the technique of edge computing, namely, UTOM, is presented in this paper to optimize the offloading strategy to get the minimum delay and achieve load balance.

Specifically, the pivotal motivations and contributions of this paper are shown below:

- (i) Few studies research on the offloading methods to pursue the minimum completion time and load balance variance with the consideration of the particular CPSS scenario. So a unique task offload method in CPSS based on the technique of edge computing is presented in this paper.
- (ii) The evolutionary algorithm, Improving the Strength Pareto Evolutionary Algorithm (SPEA2), and normalization method, Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS), are

deployed to obtain the feasible offloading strategies and select the optimal strategy.

- (iii) Sufficient experimental comparisons and assessment analysis with traditional methods confirmed the effectiveness of UTOM.

The rest of our paper is presented as follows: Section 2 shows the related work of our paper. Section 3 presents the system model based on the CPSS combined with 5G-envisioned edge computing. The process of UTOM based on the MOEA with edge is elaborated in Section 4. Section 5 shows the evaluations of UTOM and demonstrates the effectiveness of this method. Conclusion and future work are presented in Section 6.

2. Related Work

The characteristics of edge computing, i.e., sufficient memory capacity and higher computing power, are prominent in CPSS [17]. The related researches about CPSS and its prominent advantages have been extensively studied in some previous pieces of literature.

CPSS has many applications due to its real time, diversity, high reliability, and other advantages. In [18], Wang systematically described how CPS is translated into CPSS, the definition and classification as well as applications of CPSS, the contribution and significance of CPSS, and how CPSS connects and functions with different entity worlds. Relevant work has been carried out in the Internet field combining scenes such as the IoT. Han et al. proposed to introduce dynamic and manifold human behaviour into the vehicle network to make it become a CPSS system and protocolled it as a parallel vehicle network, to achieve more stable and high-efficiency traffic state and ultra-low data communication delay between vehicles [19]. Wang et al. put forward a new unified method of CPSS framework based on cloud parallel driving, which aims at collaborative online automatic driving, and developed parallel testing, learning, and reinforcement learning for this framework [20].

Given the large amount of "4V" data in the CPSS scenario, it is difficult to effectively and timely solve the demand for these data traditionally [21]. Thus, we combine the edge computing with CPSS to address and solve the mentioned problems [22, 23]. Edge computing brings the advantages of cloud computing into various application scenarios of CPSS and provides efficient services similar to cloud services on the edge of the CPSS network [24].

The offloading strategies have become more efficient and adaptive by utilizing edge computing. Mach and Becvar presented a survey, in which they divide the research of computing load into three key areas: decision-making of computing load, allocation of computing resources in MEC, and mobile management, and this survey provides relevant research direction [25]. Advanced algorithms are proposed to solve offloading problems in some literatures. A low complexity online algorithm is developed by Mao et al., which only depends on the instantaneous side information and does not need to calculate the task request distribution information, and the algorithm determines the decision of

unloading as well as the calculation of the transmission power of unloading [26]. Wang et al. proposed an innovative framework, which is developed to improve edge computing performance, and an optimal resource unloading scheme, which is designed to minimize the overall energy consumption of access nodes [27]. Wang et al. got the optimal solution based on transforming the energy consumption minimization problem into a convex problem and proposed a single variable search local optimal algorithm for the non-convex as well as non-smooth problem of delay minimization to obtain the optimal results [28]. Some unique solutions are also proposed to solve the problems in the offloading strategy. Chen and Hao reduced the task offloading problem to NP-hard problem and designed an efficient scheme to solve the task placement and resource allocation sub-problems [29].

However, most of existing studies only focus on one point in CPSS scenario and offloading method without considering them together. Indeed, compared with the previous works, this paper designed a CPSS-based offloading strategy named UTOM, whose purpose is offloading the tasks from the users in the condition of minimum time consumption and load balance variance.

3. System Model

Firstly, the framework of the task offloading model based on CPSS in 5G scene is presented in Figure 1. Secondly, the propagation delay model and load balance model for offloading strategies are designed. Thirdly, the task offloading problem has been defined as a multi-objective problem. Table 1 shows some key terms and their descriptions.

3.1. Task Offloading Resource Model. In a 5G scene based on CPSS, base stations are arranged to offer efficient services for task requesters. In general, the coverage of base stations includes some micro base stations with the aim of improving access speed and service efficiency. The micro base stations receive service or data request tasks from mobile handheld devices by using wireless signals.

As shown in Figure 1, the diagram briefly describes two scenarios: the online social scenario and the actual commuting scenario. In online social scenario, social contact data, app data, and IoT device data will be generated. Traffic data, motion data, trajectory data, and so on will be generated in the actual commuting scenario. In these scenarios, there exist a huge number of users around the base station to receive data. The purpose of this paper is to study how to offload the service data to the appropriate edge server with the purpose of minimizing the delay and keeping the load balance of the server.

It is assumed that there are quantifiable servers in the framework of task offloading in this section. Denote the task requester collections as $R = \{r_1, r_2, \dots, r_M\}$, where M represents that there are M hypothetical task requesters in this scene. The scene assumes that each requester has only one computing task waiting to be processed. Denote the base station collection as $MB = \{mb_1, mb_2, \dots, mb_N\}$, where N

represents the number of base stations. Each base station only accepts task requests within its coverage. Then, the base station transfers its received tasks to the edge server to which it belongs. The edge server collection is denoted as $ES = \{es_1, es_2, \dots, es_K\}$, where K represents that there are K edge servers in this framework.

3.2. Propagation Delay Model. This paper assumes that the overall propagation delay consists of four parts in all. VB_n^k is defined to judge whether the n -th ($n = 1, 2, \dots, N$) base station mb_n is combined with the k -th ($k = 1, 2, \dots, K$) edge server es_k .

$$VB_n^k = \begin{cases} *20c1, & \text{if } mb_n \text{ combine with } es_k, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The first part of overall propagation delay is propagation time for transferring task from the base station to the target edge server, which is calculated by

$$PT_n(X) = (1 - VB_n^k) \frac{ST_n \cdot \lambda_n}{TR}, \quad (2)$$

where ST_n represents the size of the task request coming from the base station mb_n in the coverage of edge servers es_k . Besides, λ_n signifies the number of the passing base stations in the process of task propagation, and TR denotes the propagation rate between base stations.

The second part is execution time for coping with the task requests coming from mb_n , which is defined as

$$CT_n(X) = \frac{ST_n}{AV_n \cdot PV}, \quad (3)$$

where AV_n represents the resource units VMs demanded by the task of mb_n and PV represents the processing power of each unit in VM.

The third part is average wait time of the task request in edge server, which is calculated by

$$AT_n(X) = \frac{WL}{AR}, \quad (4)$$

where AR represents the arrival rate of task and WL represents average wait length of the task request.

The fourth part is the return time of processing result coming from es_k , which is obtained by

$$RT_n(X) = \frac{ST'_n}{AV}, \quad (5)$$

where ST'_n signifies the task size of the results offloaded from mb_n .

The total propagation delay for responding to one certain task request is defined as

$$OT_n(X) = PT_n(X) + CT_n(X) + AT_n(X) + RT_n(X). \quad (6)$$

The average delay for responding to all the task requests is calculated by

$$AVE = \frac{1}{N} \sum_{n=1}^N OT_n(X). \quad (7)$$

FIGURE 1: A task offloading framework based on CPSS with edge.

TABLE 1: Key terms and relevant descriptions.

Key terms	Relevant descriptions
ES	Edge server set, $ES = \{es_1, es_2, \dots, es_K\}$
MB	Base station set, $MB = \{mb_1, mb_2, \dots, mb_N\}$
R	A set of task requesters, $R = \{r_1, r_2, \dots, r_M\}$
K	The size of the edge servers
N	The size of the base stations
M	The size of the task requesters
PV	The processing power of each unit in VM
C_{vm}	The number of the VMs in edge servers
TR	The propagation rate between base stations
AR	The arrival rate of task requests

3.3. Load Balance Model. This paper aims to investigate which edge server is the best node to offload. During the process of searching offloading strategies, the load balance is an important factor which assesses the reliability of the designed model. By utilizing of virtualized technique, the usage of virtual machine (VM) instances could be leveraged to obtain the load balance variance for all the edge servers in the offloading strategy.

SB_k is defined to estimate whether es_k has been occupied, which is acquired by

$$SB_k = \begin{cases} 1, & \text{if } es_k \text{ has been occupied,} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

Besides, MB_n^k is defined to estimate whether the task in mb_n has been offloaded to es_k . It is acquired by

$$MB_n^k = \begin{cases} 1, & \text{if } mb_n \text{ offloads the task to } es_k, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Thus, the number of running edge servers is defined as

$$A_s = \sum_{n=1}^N MB_n^k. \quad (10)$$

The corresponding resource utilization of es_k is calculated by the utilization number of VM, which is defined as

$$R_k = \frac{1}{C_{vm}} \sum_{n=1}^N MB_n^k \cdot \varepsilon_n, \quad (11)$$

where ε_n represents the number of VMs required by the task requests in mb_n .

Then, the overall average resource utilization of the edge servers is obtained by

$$AU_R = \frac{1}{A_s} \sum_{k=1}^K R_k. \quad (12)$$

According to the different resource utilization in each edge server, the load balance variance of es_k is calculated by

$$BV_{ave} = (R_k - AU_R)^2. \quad (13)$$

At last, the overall load balance variance of the occupied edge servers in the offloading scene is calculated by

$$LB = \frac{1}{A_s} \sum_{k=1}^K SB_k \cdot BV_{ave}. \quad (14)$$

3.4. Problem Formulation. The objective functions of this system model have been presented in (7) and (14), which is expected to improve the overall effectiveness in the task offloading scenario. The objective problems are formulated as follows:

$$\min LB, \quad (15)$$

$$\min AVE, \quad (16)$$

$$\text{s.t. } AV_n \leq C_{vm} (n \in \{1, 2, \dots, N\}). \quad (17)$$

The constraint that the number of VMs requested in any task must be less than the number of VMs in the edge server has been defined in (17).

4. The Tasks Offloading Method

As shown in (7) and (14), in this segment, a unique task offloading means based on the MOEA with CPSS, called UTOM, is designed with the aim of minimizing the propagation delay and the load balance variance. The process of UTOM, i.e., searching the feasible solutions, normalizing the solutions, and selecting the optimal strategy, is presented in detail.

4.1. Offloading Strategy Option by SPEA2. The multi-objective offloading model based on SPEA2 is presented in problem formulation. SPEA2 utilizes an advanced fitness allocation strategy, which takes into account the number of individuals controlled by each individual [30]. Besides, it integrates the nearest neighbour density estimation technology, allowing more accurate guidance in the search process. Given these advantages, SPEA2 is utilized to be adopted in this method to figure out the double-objective optimization problems. The related fitness functions and constraints are encoded first in this process. Then, the selection process including environment selection and pairing selection is applied. Finally, the advanced evolutionary operators are performed to generate solutions.

- (1) Encoding: First of all, we need to map the problem to be solved, i.e., minimizing the time consumption in (7) and the load balance variance in (14), into a mathematical problem. The solution of the problem is represented by a coded string of numbers, and the genetic operators operate on the string directly. There are many coding ways, and here floating-point coding is selected. Because we have high precision requirements for the results to be generated, the solution space will increase dramatically when utilizing integer coding. In addition, the floating-point coding method is easy to deal with the complex constraints of decision variables, which have been presented in the model part.
- (2) Fitness functions and constraints: In the genetic algorithm, fitness function plays the role of selecting excellent individuals. According to the fitness value of the individual, it selects the individuals to be inherited to the next generation. In this method, the fitness function is transformed according to the objective functions (7) and (14). The practical constraint has been shown in (17), which represents that the number of VMs requested for any task must be less than the number of VMs in the edge servers.
- (3) Selection operator: Selection refers to the operation of selecting excellent individuals from the group and eliminating the inferior ones. It is based on the evaluation of fitness. The larger the fitness, the greater the possibility of being selected, the number

of his “offspring” in the next generation, and the selected individuals will be put into the matching database. This method selects a roulette operator, which ensures the individuals whose fitness function is better would be selected into the next generation as far as possible while ensuring that all individuals are likely to be selected.

- (4) Crossover and mutation operators: The purpose of the crossover is to improve the searchability of the genetic algorithm of leap in the next generation of the new individual through the crossover operation. Crossing is an important method of genetic algorithm to obtain excellent individuals. The probability of crossover operation is in accordance with the random selection of two individuals in the library, and the cross-location is random. Indeed, single-point crossover is applied in UTOM.

The basic process of mutation operation is as follows: generate a random number $rand$ between 0 and 1 and mutation probability pm . If $rand > pm$, the mutation operation would be performed. Mutation operator itself is a kind of local random search, and it has the ability to avoid some of the permanent loss of information due to selection and crossover operators together with the selection and crossover operators. It makes the genetic algorithm maintain the population diversity, while avoiding premature convergence. In mutation operation, the probability of mutation should not be too large. If $pm > 0.5$, the genetic algorithm will degenerate into random search.

4.2. Optimal Strategy Selection by TOPSIS. Figure 2 shows how to utilize TOPSIS to derive the optimal strategy based on the strategies generated by SPEA2. TOPSIS generates results which accurately reflect the gap between the evaluation schemes [31]. Then, the best result could be gained by comparing these gaps. The symbols utilized in the flowchart are described as follows [32].

The initial strategies produced by SPEA2 are represented by P , and these strategies form two sets, i.e., propagation time strategies T and load balance variance strategies L , where $T = \{T_1, T_2, \dots, T_P\}$ and $L = \{L_1, L_2, \dots, L_P\}$.

The standardized values of propagation time and load balance variance are defined as $CT_p(X)$ and $CL_p(X)$. Two weights of the indicators are defined as μ_T and μ_L . Then, the standardized weight decision values are presented as WT_p and WL_p . Afterwards, the degree of closeness between alternatives and the best solution as well as the worst solution are measured as OS_p and WS_p . Next, the comprehensive evaluation value of the best solution and the worst solution is presented as C_r . At last, the best solution S_{idea} would be obtained from all the strategies.

4.3. The Overview of UTOM. The purpose of UTOM method is realizing the optimization of the objective functions presented in the system model. The overview of our UTOM is presented in Algorithm 1. In this algorithm, the scale of the population is I , the maximum amount of inheritance is J ,

FIGURE 2: The flowchart of utilizing TOPSIS to derive the optimal strategy.

```

Require  $J, I$  Ensure BS
Obtain tasks from task requesters
Initialize strategies  $P$ 
For ( $j = 1, 2, \dots, J$ )
   $i = 1$ 
  While ( $i \leq I$ )
    Execute the crossover, selection, and mutation operators to produce offspring
    For (individuals in population)
      Obtain total propagation time from (7)
      Obtain load balance variance from (14)
    End For
    Execute environmental selection operator
     $i = i + 1$ 
  End While
  Acquire the utility values by TOPSIS
  Obtain the best solution BS
End For
Return BS
  
```

ALGORITHM 1: Obtaining the best strategy by utilizing UTOM.

and the exportation of UTOM is the best strategy BS. Firstly, the initial strategies are produced randomly and presented as P . Then, feasible solutions are generated after J iterations by SPEA2 through fitness functions. Finally, TOPSIS is applied to calculate standardized values and select optimal value.

5. Experimental Evaluation

Our paper utilizes the MECHREVO-Ti2 as the experimental station. The computer parameters are as follows: the CPU is Intel i7-6700H @ 2.6 GHz, the RAM is 8 GB, and the hard disk is 1T. Some experimental parameters and their values used in this section are shown in Table 2. To prove the effectiveness of this method, some traditional methods, i.e.,

TABLE 2: Experimental variable setting.

Experimental variable	Value
The scale of tasks	{50, 100, 150, 200, 250}
The size of edge services	[0.5, 0.8]
The scale of running VMs	[1, 6]
TR	620 MB/s
AR	120 MB/s

Benchmark, First Fit Decreasing-based task offloading with time-saving and resource utilization optimization (FFD), and Best Fit Decreasing-based task offloading with time saving and resource utilization optimization (BFD), are utilized in this section. Benchmark supposes that the VM in

the initial edge node falls short with regard to the requirement of task; the certain task would not be coped with the other node. FFD gives the required amount of the VM in the task, and the tasks will be ranked in an order. Then, the initial task will be offloaded to the initial node. BFD will sort all the tasks by descending order. Afterwards, the initial task will be offloaded to the initial node. Benchmark, FFD, and BFD will be uniformly referred to as the “three classical methods.”

5.1. Comparison of Experimental Results on the Employed Number of Edge Servers. The number of the employed edge servers by Benchmark, FFD, BFD, and UTOM is presented in Figure 3. It is obvious that the number of the employed edge servers is smaller than that of the other three methods. When the number of the evaluated tasks equals 100, the gap between the four methods is relatively little. The difference between the four means it begins to increase with the increasing of the task amount. It means that the UTOM has a better performance in higher number of tasks. These data show that the performance of our UTOM is the best in the comparison, while the performance of Benchmark is the worst.

5.2. Comparison of Experimental Results on the Average Propagation Time of Tasks. The average time equals the entire time divided by the number of tasks. Correspondingly, the average propagation time of tasks is calculated. The average propagation time keeps increasing with the enlarging of the task amount. The time calculated by UTOM is lower than the other three methods. The average propagation time of UTOM in different scale of tasks is 0.13, 0.22, 0.37, 0.48, and 0.59 (s) when the number of the tasks equals 50, 100, 150, 200, and 250. Besides, the difference of the three classical methods between UTOM is shown in Figure 4.

5.3. Comparison of Experimental Results on the Overall Propagation Time of Tasks. The entire time consists of four parts, which are the propagation time, the wait time, the execution time, and the return time. The total time represents the satisfaction of the users. It is conducted that the total time obtained by our UTOM is lower than the other methods by analysing Figure 5. The total time of UTOM is 6.49, 22.00, 56.10, 95.50, and 147.94 (s) when the number of the tasks equals 50, 100, 150, 200, and 250.

5.4. Comparison of Experimental Results on the Average Resource Utilization. The average resource utilization is not the objective function in our paper, while this index is another crucial evaluation parameter in the experimental comparison. This index represents the employed amount of the VM in the edge servers and is expected to get a high value in the experiment. Figure 6 presents the performance comparison of average resource utilization by utilizing of UTOM and three classical methods severally. The Benchmark performs worst in the comparison, and UTOM performs better than the other two means.

FIGURE 3: Experimental results on the employed scale of edge servers by UTOM and three classical methods.

FIGURE 4: Experimental results on the average propagation time of tasks by UTOM and three classical methods.

FIGURE 5: Experimental results on the entire propagation time of tasks by UTOM and three classical methods.

5.5. Comparison of Experimental Results on the Load Balance Variance. The load balance variance is the objective value in this experiment. It is summarized that the variance begins to

FIGURE 6: Experimental results on the average resource utilization by UTOM and three classical methods.

FIGURE 7: Experimental results on the load balance variance by UTOM and three classical methods.

FIGURE 8: Experimental results on utility value by UTOM and three classical methods.

enlarge with the increasing of the amount of the tasks by analysing Figure 7. The lower value represents the better offloading strategy which the method obtained. The load balance variance of UTOM is 0.14, 0.35, 0.47, 0.59, and 0.70 when the number of the tasks equals 50, 100, 150, 200, and 250.

5.6. The Analysis of Experimental Results on the Utility Value. The different utility values of all the strategies are obtained by TOPSIS method. The certain strategy with the maximum value among the entire strategies is our best strategy. It is shown that the optimal utility value in different amount will reduce with the decreasing of the task scale by analysing Figure 8. From Figure 8, we intuitively deduce that UTOM obtains lower utility value with the improvement of task number. After the detailed statistics, the utility value of UTOM is 0.73, 0.78, 0.81, 0.83, and 0.86 when the number of the tasks equals 50, 100, 150, 200, and 250.

6. Conclusions

We devote ourselves to the problem of task offloading based on CPSS, in which edge computing technology is reasonably combined. The offloading problem is defined as an optimization problem of the minimizing of the propagation consumption and load balance variance. Furthermore, a method named UTOM is presented in this paper to optimize the offload strategy to get the minimum propagation delay and load balance variance. Besides, the normalization technique named TOPSIS is also utilized in combination to obtain standardized data. The experimental results show that the UTOM method has sufficient effectiveness and correctness. We intend to apply this method to real datasets based on CPSS to discuss the applicability in practice in future work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Jielin Jiang conceived and designed the study. Shengjun Li performed the simulations. Xing Zhang wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61601235 and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20160972.

References

- [1] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6172–6181, 2019.
- [2] W. Yu, W. Wang, P. Jiao, H. Wu, Y. Sun, and M. Tang, "Modeling the local and global evolution pattern of community structures for dynamic networks analysis," *IEEE Access*, vol. 7, pp. 71350–71360, 2019.

- [3] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.
- [4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
- [6] H. Wu, Z. Han, K. Wolter, Y. Zhao, and H. Ko, "Deep learning driven wireless communications and mobile computing," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 4578685, 2 pages, 2019.
- [7] L. Wang, L. Jiao, J. Li, and J. Gedeon, "Moera: mobility-agnostic online resource allocation for edge computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2019.
- [8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.
- [9] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.
- [10] M. Wen, K. Ota, H. Li, J. Lei, C. Gu, and Z. Su, "Secure data deduplication with reliable key management for dynamic updates in CPSS," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 137–147, 2015.
- [11] X. Xu, S. Fu, L. Qi et al., "An iot-oriented data placement method with privacy preservation in cloud environment," *Journal of Network and Computer Applications*, vol. 124, pp. 148–157, 2018.
- [12] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407–1419, 2019.
- [13] H. Yu, H. Qi, and K. Li, "CPSS: A study of cyber physical system as a software-defined service," *Procedia Computer Science*, vol. 147, pp. 528–532, 2019.
- [14] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A qos-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.
- [15] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled iot," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2622–2629, 2020.
- [16] L. Qi, Q. He, F. Chen et al., "Finding all you need: web apis recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.
- [17] X. Xu, Y. Li, T. Huang et al., "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.
- [18] F. Wang, "The emergence of intelligent enterprises: from CPS to CPSS," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 85–88, 2010.
- [19] S. Han, X. Wang, J. J. Zhang, D. Cao, and F.-Y. Wang, "Parallel vehicular networks: a CPSS-based approach via multimodal big data in iot," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1079–1089, 2018.
- [20] F.-Y. Wang, N.-N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, "Parallel driving in CPSS: a unified approach for transport automation and vehicle intelligence," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 577–587, 2017.
- [21] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for iot-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.
- [22] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: a survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.
- [23] N. Moustafa, K.-K. R. Choo, I. Radwan, and S. Camtepe, "Outlier dirichlet mixture mechanism: adversarial statistical learning for anomaly detection in the fog," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 1975–1987, 2019.
- [24] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
- [25] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
- [26] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.
- [27] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.
- [28] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, p. 1, 2016.
- [29] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.
- [30] K. Giannakoglou, D. Tsahalis, J. Periaux et al., "Spea2: improving the strength pareto evolutionary algorithm for multiobjective optimization," 2001.
- [31] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "Trust-oriented iot service placement for smart cities in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2019.
- [32] V. T. Lokare and P. M. Jadhav, "Using the AHP and topsis methods for decision making in best course selection after HSC," in *Proceedings of the 2016 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–6, Coimbatore, India, January 2016.