Joint Collaborative Task Offloading for Cost-Efficient Applications in Edge Computing

Chaochen Ma[1(✉)], Zhida Qin[2], Xiaoying Gan[2,3], and Luoyi Fu[2]

[1] SJTU ParisTech Elite Institute of Technology, Shanghai Jiao Tong University, Shanghai 200030, China
machaochen1995@sjtu.edu.cn

[2] Department of Electronics Engineering, Shanghai Jiao Tong University, Shanghai 200030, China

[3] National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China

Abstract. Edge computing is a new network model providing low-latency service with low bandwidth cost for the users by nearby edge servers. Due to the limited computational capacity of edge servers and devices, some edge servers need to offload some tasks to other servers in the edge network. Although offloading task to other edge servers may improve the service quality, the offloading process will be charged by the operator. In this paper, the goal is to determine the task offloading decisions of all the edge servers in the network. A model is designed with different types of cost in edge computing, where the overall cost of the system reflects the performance of the network. We formulate a cost minimization problem which is NP-hard. To solve the NP-hard problem, we propose a Joint Collaborative Task Offloading algorithm by adopting the optimization process in nearby edge servers. In our algorithm, an edge server can only offload its tasks to other edge servers within a neighborhood range. Based on the real-world data set, an adequate range is determined for the edge computing network. In cases of different density of tasks, the evaluations demonstrate that our algorithm has a good performance in term of overall cost, which outperforms an algorithm without considering the influence of neighborhood range.

Keywords: Edge computing · Task offloading · Quality of service · Cost-efficiency

1 Introduction

With the continuous development of technologies such as the Internet of Things, mobile Internet, and big data, in recent years, a surge of network applications is witnessed. The numbers of websites and mobile applications grow rapidly, and different smart terminals are available online anytime, anywhere. According to

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 312, pp. 77–90, 2020.
https://doi.org/10.1007/978-3-030-41114-5_7

an in-depth market report published by Cisco, by 2022, 59.7% of the population on earth will be the user of the internet, the average devices and connections per person will be 3.6, and the average traffic per person per month will be 49.8 GB [1].

To satisfy the demand driven by the sustained and rapid traffic growth, more requirements have been put forward for the data storage and processing technology. Edge computing has been proven an efficient method to meet the increasing demands. Due to the drawbacks of traditional cloud computing, such as insufficient bandwidth, lack of mobility support and location awareness, high transmission delay, cloud data centers have been incapable to efficiently and timely process massive data generated by edge devices. Therefore, edge computing has emerged as a new data processing model without the drawbacks of cloud computing. By extending computation power from the cloud data center to the edge of the network, the local data streams will be processed by nearby edge servers rather than transported to the remote cloud data centers, the transmission delay and bandwidth cost will be greatly reduced, which satisfies quality-of-service requirements of users. Due to the above advantages, edge computing has been adopted in different fields such as augmented reality, virtual reality, video image analysis and smart traffic system.

A lot of work has been done in task offloading of edge computing network. Some research work concentrates on the incentive mechanism, energy consumption, service of quality in resource allocation and task offloading of edge computing. Chen et al. study coalition formation algorithm based on the coalition game theory, the incentive mechanism and social trust network in edge computing [2]. Li et al. focus on the placement of edge servers for reducing the overall energy consumption [3]. Lyu et al. presents a new fully distributed optimization of fog computing to minimize the time-average cost in a large-scale network [4]. Pasteris et al. propose a deterministic approximation algorithm to solve the reward maximization problem in a mobile edge computing network [5]. Pu et al. focus on minimizing the time-average energy consumption for task executions of all users in device-to-device fogging [6]. Xiao et al. investigate a task offloading problem which aims to maximize quality-of-experience of users under the given power efficiency [7].

Some research work focuses on the algorithms that optimize the system performance by task offloading under certain constraints in edge network. Wang et al. propose an online algorithm that optimally solves an edge cloud resource allocation problem with arbitrary user mobility over time, and an algorithm for an offline case of social virtual reality applications in edge computing [8,9]. Hou et al. propose a tractable online algorithm that configures edge-clouds dynamically solely based on past system history [10]. Xu et al. develop an online algorithm based on Lyapunov optimization and Gibbs sampling for the problem of dynamic service caching in dense cellular networks of mobile edge computing [11]. Zhou et al. concentrate on the proactive cost management problem by deciding the server provisioning ahead of time, based on prediction of the upcoming workload [12]. Sundar et al. design an algorithm to solve the problem with a completion

deadline allocated to each individual task by greedily optimizing the scheduling of each task subject to its time allowance [13]. To the best of our knowledge, few researchers consider energy consumption, quality of service, cost of communication simultaneously, and the influence of neighborhood range in task offloading process of edge computing has not been taken into account by the algorithm researchers.

In this paper, we focus on the joint collaborative task offloading decision problem in edge computing. As edge computing and storage capabilities continue to increase, computing tasks are processed on the edge side, which decrease the distance between computing devices and users. Due to the constrained resource of edge devices, we should note that it is difficult for an edge device to support latency-sensitive applications with vast computation amount alone. Therefore, to improve the quality of service of an edge computing network, a natural idea is offloading some of the tasks of edge nodes with high computation load to other edge nodes with fewer tasks to be processed. To find out the best task offloading decision, we are faced up with following challenges: Firstly, an edge server is usually equipped with limited storage and computation capacity compared with cloud computing center. Thus an edge server should not be too crowded with computational tasks. Secondly, the edge servers will update the state of other edge servers and the data of tasks placed on the servers, the regular exchange of a large number of controlling messages may cause the congestion of the network. Thirdly, while shipping the tasks from the task-crowded edge server to other servers may improve the quality of service, the communication between edge servers will generate extra costs. The tradeoff between the quality of service and the communication cost must be carefully considered in order to obtain a better cost efficiency.

To cope with the above challenges, firstly, we model the system with four types of cost, including energy cost, placement cost, computation cost and communication cost. With these four types of cost, a lot of performance measure in reality can be represented. Then, we formulate a cost minimization problem which is NP-hard. The overall cost can evaluate the performance of network after task offloading. The solution designed for the problem is Joint Collaborative Task Offloading algorithm. We optimize each edge server by deciding which of its tasks would be offloaded to which of the other edge servers in its neighbourhood range. The above optimization process will continue until the overall cost converge. The algorithm has following advantages: (1) Each server can only offload its tasks to a small set of nearby servers. Without massive calculation, the optimization for each edge server is fast. (2) For the fact that each edge server only needs to know the task information of the nodes in its neighbourhood range, less controlling message to update the network state is needed. (3) Finally, we evaluate the algorithm performance by carrying out experiment in the large-scale network and find out a better performance compared to a typical optimization algorithm.

2 System Model

The structure of an edge computing system is illustrated in Fig. 1. The computation tasks will firstly generated by users. Each user transmits the task to the nearest edge server with computation and storage capacity. Considering the fact that some edge servers have received too many tasks from users, some of the tasks on the edge server would be offloaded to other edge servers in order to improve the service of quality with high cost-efficiency.

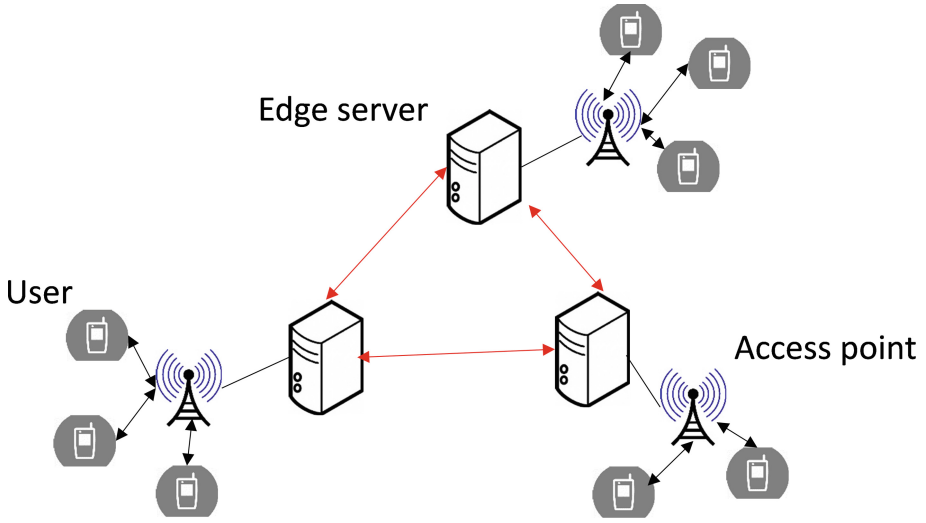

Fig. 1. Example of a wireless application in edge computing network. Users can connect to the edge computing network through the nearest access point.

In a large-scale network, we consider a set of m edge servers in the network. The set is denoted by $S = \{s_1, s_2, ..., s_m\}$. The servers are geographically dispersed in the city, in proximity to the devices of users. Each edge server contains a wireless access point which allows the connection between users and edge server. The capacity for the storage of the data generated from the tasks of users in every edge server is sufficient. The edge servers are connected through local-area-network or a metropolitan-area-network. The distance between two edge servers p, q dispersed in the network is denoted by $d(s_p, s_q)$.

In the network, we consider an application serve a set of n users with the edge servers. The set of users is denoted as $U = \{u_1, u_2, ..., u_n\}$. Each user can get access to the nearest edge server through the access point, and uploads its task to the edge network. Every computational task generated by a user is not separable, and thus it can only be processed by one single edge server. To identify in which edge server the task uploaded by the server is temporarily placed, we define binary variables $b_{i,j} \in \{0, 1\}$ for all $i \in U, j \in S$. If the task of user i is placed in server j, $b_{i,j} = 1$, else $b_{i,j} = 0$.

2.1 Edge Network Cost Model

To evaluate multiple performance measures of the network, we define four types of cost: energy cost, placement cost, computation cost and communication cost.

Energy Cost. In energy cost, we concentrate on the energy consumed by edge servers. For each edge server, its energy consumption can be divided into two parts. The first part is the energy cost to keep the edge server activated. This part of energy is static due to the maintenance of server in the activation state. The second part of energy cost is generated by the processing of the tasks placed on edge servers. For each edge server, the energy to deal with the computational tasks is proportional to the number of tasks placed in the server. We use α_i to denote the energy cost to keep edge server $i \in S$ activated, and efficiency factor β_i is denoted as the ratio of between the energy cost of processing a computational task and the workload offloaded on edge server i. The number of tasks placed in server i is defined as x_i. The total energy cost in the edge computing network is given by:

$$E_e = \sum_{i \in S, x_i > 0} \alpha_i + \sum_{i \in S} \beta_i x_i \tag{1}$$

Placement Cost. Placement cost is generated due to the communication between edge server and users. After the creation of task on a use's device, the task will be uploaded to the nearest edge server through wireless access point. The cost of placing the task of an user $n \in U$ in its nearest edge server $m \in S$ is denoted as $l_n(s_m)$ which varies due to the different locations of user. The placement cost to upload all the tasks from users to edge servers is denoted as:

$$E_p = \sum_{n \in U} l_n(s_m) \tag{2}$$

Computation Cost. Computation cost is associated to the congestion of the task placed on edge servers. As an edge server is designed with limited computational resources and large-scale parallel processing can be difficult, when the number of tasks places in an edge server increases, the computational cost increases which reflects a worse quality of service. For each server $i \in S$, its computation cost is denoted as $\gamma_i(x_i)^2 + \delta_i x_i$. x_i is the number of tasks in server i. γ_i and δ_i are parameters. We assume that in worst case, server i deal with the tasks with only one CPU, and each task needs time t to be processed. The first task needs t to be finished; the second task needs $2t$; the n-th task needs nt. We can evaluate the overall quality of service by summing up the above quadratic terms, thus $\gamma_i(x_i)^2 + \delta_i x_i$ is reasonable to represent its computation cost. If the server has multiple CPUs, γ_i and δ_i should be adjusted. The generalization of the model can be realized through altering for different types of applications or different edge servers. As a result, the total computational cost of the network is denoted as:

$$E_c = \sum_{i \in S} \gamma_i(x_i)^2 + \delta_i x_i \tag{3}$$

Communication Cost. The edge servers are connected with each other by local-area-network or metropolitan-area-network. The transmission of data between edge servers would be charged by the edge network operator considering the amount of data which has been transferred. We denote the communication fee charged by the operator for the transmission of per unit of data from server i to server k as ϵ_{ik}. The parameter ϵ_{ik}, can be approximated as the distance between two edge servers in the network. The communication cost can be divided into two parts. The first part is the communication cost to transport the task generated by user n from edge server u to edge server v, denoted as the $k(d(s_u, s_v)C_n)$ where k is the price parameter decided by the operator and C_n is the amount of data of the task of user n. The second part represent the communication cost when we forward the result of task n processed by edge server v back to edge server u, which can be denoted as $k(d(s_u, s_v)C_n\theta_n)$, where θ_n represents the data size change rate after task initialized from user n has been computed. If multiple task is shipped between two edge servers, we should multiply the cost by $\eta_{u,v}$, which is denoted as the number of task transferred from edge server u to edge server v. The communication cost of the edge network can be computed by:

$$\begin{aligned} E_{com} = k(&\sum_{n\in U}\sum_{u,v\in S} d(s_u, s_v)C_n\eta_{u,v} \\ &+ \sum_{n\in U}\sum_{u,v\in S} d(s_v, s_u)C_n\theta_n\eta_{u,v}) \end{aligned} \tag{4}$$

2.2 Problem Formulation

The overall cost of the edge computing system is the sum of above four types of cost, denoted as:

$$E = E_e + E_p + E_c + E_{com} \tag{5}$$

The task offloading decision problem in the edge network system can be formulated as a cost minimization problem with several constraints:

$$\min E \tag{6}$$

$$\text{s.t. } b_{i,j} \in \{0, 1\} \quad \forall i \in U, j \in S \tag{7}$$

$$\sum_{j\in S} b_{i,j} = 1 \quad \forall i \in U \tag{8}$$

$$\sum_{v\in S} \eta_{u,v} \leq x_u \quad \forall u \in S \tag{9}$$

In some circumstances, we can alternate the parameters of each type of cost to adjust the weights of different cost so that the cost model can be designed for achieving different trade-off goals or adapting new specifications of the edge networks and applications. For example, if the system attaches great importance to the quality of service, the parameters of computation cost γ_i and δ_i can be increased to meet the demands. For the constraints in the cost minimization problem, Eq. (8) means that a task can only be placed on one edge server, Eq. (9) means that the number of tasks transferred from edge server u to edge server p would not exceed the number of tasks placed on server u. For the simplicity, constraint of the storage capacity of edge servers has not been taken into considered, and each task generated by users need the same time span to be computed by a server.

The above cost minimization problem can be found as NP-hard by converting the problem into the Simple Plant Location Problem (SPLP) which has been known as NP-hard [14]. The placement cost and the energy cost to activate the servers can be treated as the facility establishment cost in SPLP. The energy cost of task processing can represent the operating cost of SPLP. The communication cost and computation cost can be regarded as the transportation cost in SPLP. Thus, the cost minimization problem is NP-hard.

3 Algorithm Design

To solve the above task offloading decision problem, the general idea is to realize the overall cost minimization by continuous optimization process in each edge server. We find out the task offloading decision with the maximum cost decrease for a certain server, and continue the procedure for other servers in the edge network. This optimization process will be iterated until the overall cost does not decrease after an iteration. Following this idea, we propose a Joint Collab-

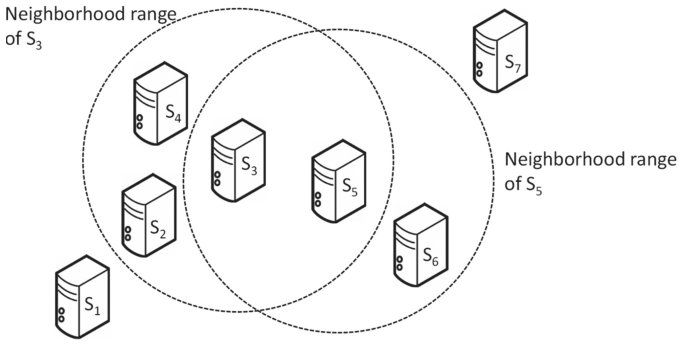

Fig. 2. An example of the neighborhood range of different edge servers in an edge computing network.

orative Task Offloading (JCTO) algorithm. The pseudo-code of JCTO is listed in Algorithm 1.

In the JCTO algorithm, a neighbourhood range is defined as the maximum range between two edge servers that one server is allowed to offload its tasks to another edge server. For the guaranty of bidirectional communication, the neighbourhood range is considered as the same for all servers in a single edge network. As is illustrated in Fig. 2, seven edge servers $S_1, S_2, ..., S_7$ are geographically distributed in an edge network. S_2, S_4, S_5 are within the neighborhood range of S_3, and S_3, S_6 are in the neighborhood range of S_5. A task originally placed at S_3 cannot be shipped directly to S_6, as S_6 is not in the neighborhood range of S_3. Multi-hop transmission is available in the network. For instance, a task placed at S_3 can be firstly shipped to S_5 then to S_6. The range is designed to restrict the tasks to be offloaded to other edge servers too far away, so that the communication cost generated by task offloading process can be reduced. In practice, neighborhood range should be decided by the size of the network and the density of edge servers.

The optimization process for the network is performed in the sequence of edge servers. For each edge server, it is allowed to offload the task to its neighborhood servers in its optimization process. We apply the last in first out regulation for the tasks in an edge server to be offloaded. For a task x placed in edge server j, where $b_{x,j} = 1$, the change of cost if task x is shipped to a neighborhood server k of edge server j is denoted as:

$$\Delta E_{x,j,k} = \Delta E_{\Delta b_{x,j}} + \Delta E_{\Delta b_{x,k}} + \Delta E_{com,x,j,k} \tag{10}$$

Intuitively, $\Delta b_{x,j}$ is a variable with respect to the alternation of $b_{x,j}$, where $\Delta b_{x,j} = -1, \Delta b_{x,j} = 1$ in case x is shipped from server j to server k. The first and second terms of Eq. (10) represent respectively the change of computation cost and energy cost of server j and k after offloading task x. The third term represents the communication cost incurred by the transmission of task x from server j to k. The above calculation of Eq. (10) will be done for all the edge servers within the neighborhood range of server j, and the neighbor server with a minimum cost $\Delta E_{x,j,k_1}$ would be found, meaning that the placement of task x is decided to be offloaded to server k_1 if $\Delta E_{x,j,k_1} < 0$. The process continue for the task in server j until the minimum cost change is superior to 0 which means any task offloading originated from server j can not decrease the overall cost of the network. After the optimization of server j is finished, the optimization process of next edge server would be carried out. We will traverse all the servers during the optimization process in one iteration, and the process does not stop until the overall cost converges after several times of iterations.

4 Performance Evaluation

4.1 Dataset and Settings

We obtain a dataset of the metro stations in Beijing from the map. It contains the longitude and latitude of each metro station of a total 337 stations which are

Algorithm 1. Joint Collaborative Task Offloading Algorithm

Initialize $E, b_{i,j}$ for all $i \in U, j \in S$
Initialize $E_{min} = E$
while $True$ **do**
  for all server $j \in S$ **do**
    for all task x in task sequence of edge server j **do**
      Calculate $\Delta E_{x,j,k_1}, \Delta E_{x,j,k_2}..\Delta E_{x,j,k_y}$, where $k_1, k_2..k_y$ are neighbor servers of server j;
      $\Delta Cost = \min\{0, \Delta E_{x,j,k_1}, \Delta E_{x,j,k_2}..\Delta E_{x,j,k_y}\}$;
      if $\Delta Cost < 0$ **then**
        $E = E + \Delta Cost$, update $b_{x,j}$ for all $j \in S$;
      else
        Break;
      end if
    end for
  end for
  Compare E and E_{min} ;
  if $E_{min} = E$ **then**
    Break
  end if
end while
Output total cost E_{min} after the optimization

regarded as the locations where edge servers are placed. The user is distributed randomly in proximity of edge servers. Every user generates one task. For each edge server, the number of users in its proximity is randomly distributed in $[0, z]$ where z is the max number of users initialized near each server which represents the initial maximum number of task in an edge server, and z can be altered to adjust the total number of tasks in the edge network. In energy cost, the energy of activation and the efficiency factor for computation follow normal distribution. For the placement cost, the cost is calculated by the extent of proximity of randomly generated users. For the computation cost, by assuming that every edge server in the edge network has the same computation capacity for simplicity, the parameters are set as the same for all edge servers. For the communication cost, the amount of data of task n follows normal distribution. The data size change rate is set as 0.5. The price parameter k is set with 3 levels which will be evaluated in the experiment.

4.2 Results

Firstly, we study the influence of altering neighborhood range which is a vital parameter in the JCTO algorithm. The performance of the algorithm is evaluated by the cost minimized ratio, which represents the ratio between the overall cost after optimization and the initial overall cost. A smaller ratio means a better performance of the algorithm. In Fig. 3, it is shown that with the increase of the total number of tasks, the edge network becomes more congested and it is much

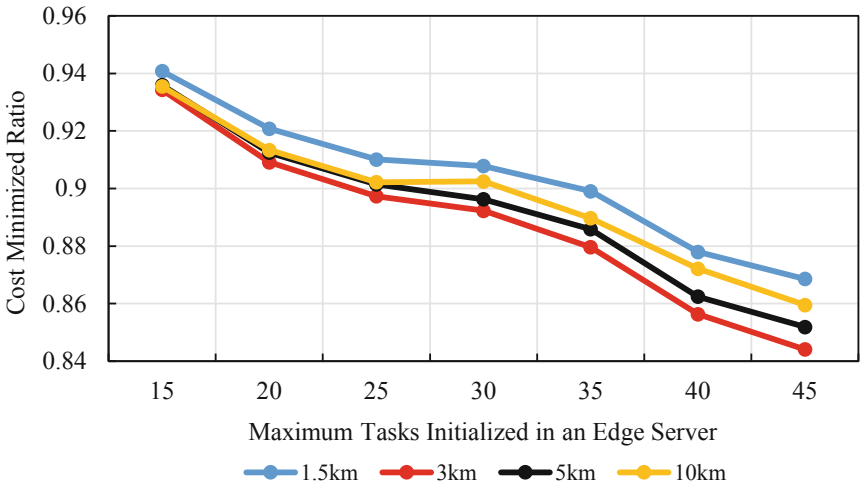

Fig. 3. Cost minimized ratio of JCTO algorithm with different neighborhood ranges.

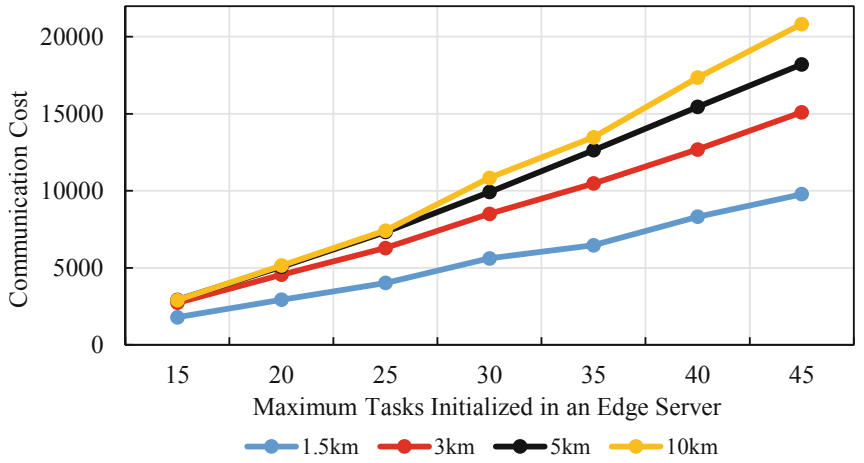

Fig. 4. Communication cost of JCTO algorithm with different neighborhood ranges.

more beneficial to offload the tasks from a crowded edge server to other servers for the reduction of the computation cost. Comparing the performances when the neighborhood range of each edge server is set as 1.5 km, 3 km, 5 km and 10 km, it is found that the neighborhood range of 3 km has the best performance. This is because: (1) When the neighborhood range is too small, such as 1.5 km, the number of neighbor servers for a certain edge server where it can directly place its tasks is small. It will be difficult to find an adequate server to offload the task from the source server for the reduction of overall cost. (2) As is shown in Fig. 4, the communication cost of the network will increase if the total number of tasks

increase, and it is obvious as more tasks will be offloaded to other servers in the optimization process. The communication cost will also increase when the neighborhood range increases as a task is allowed to be shipped to further edge servers. In our experiment, while the neighborhood range is set as 5 km, 10 km, the communication cost is much higher than the case of 3 km. Thus, it results in a better overall cost minimization performance when the range is set as 3 km. In short, the setting of neighborhood range is crucial for a good performance of Joint Collaborative Task Offloading Algorithm. The range should be carefully decided in case of different datasets.

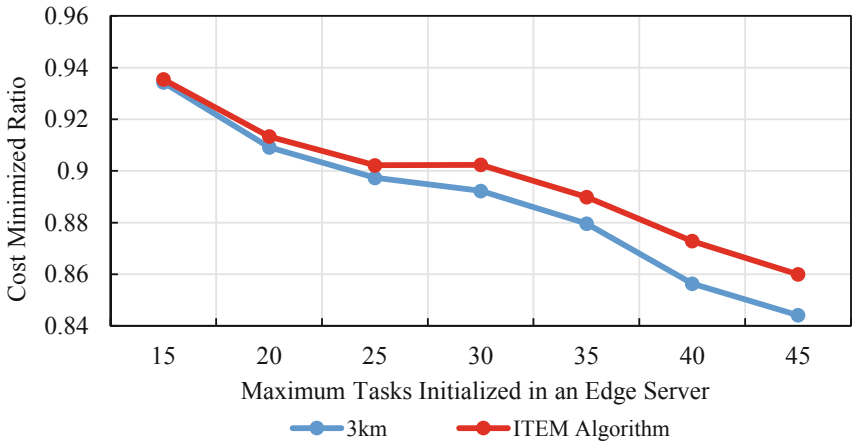

Fig. 5. Comparison between JCTO algorithm with neighborhood range set as 3 km and an optimization algorithm ITEM.

In Fig. 5, we make a comparison between Joint Collaborative Task Offloading Algorithm and another optimization algorithm: Iterative Expansion Moves algorithm (ITEM) [7]. In each iteration of ITEM algorithm, for the edge server i being optimized all the tasks placed in other edge servers are allowed to be placed in server i. By solving a minimum cut problem formed by the cost in the edge network, a set of tasks would be found and be offloaded to server i with a maximum cost decrease for the network and then continue to the next server until the overall cost converges. However, ITEM has not considered the influence of neighbourhood range. By comparison, we find out that the JCTO algorithm with the neighbor range set as 3 km has a better performance than ITEM. The reason is that the tasks throughout the network can be shipped to an edge server in the optimization process which results in a bigger communication cost compared to JCTO. Besides, ITEM needs to update regularly task informational in the whole network for each server, while JCTO only needs the information in a small neighbor range for an edge server. Thus, compared to ITEM, JCTO requires less control message with a better optimization speed.

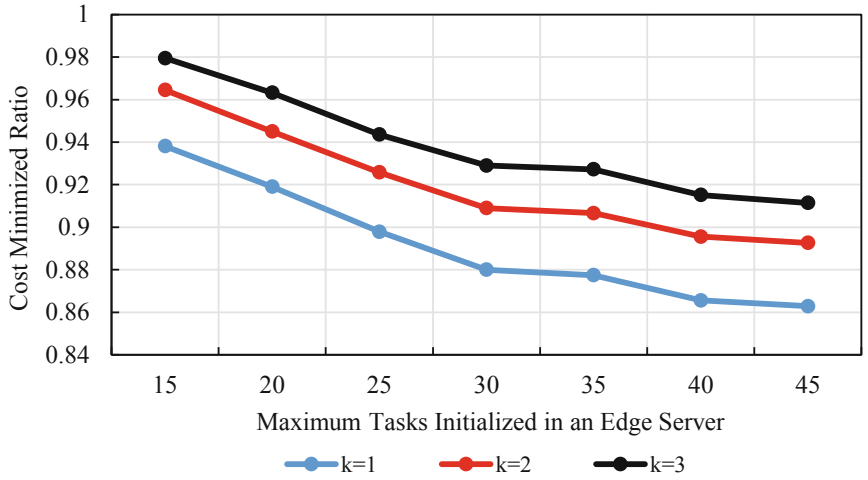

Fig. 6. Comparison of different price parameter k for communication cost in term of cost minimized ratio.

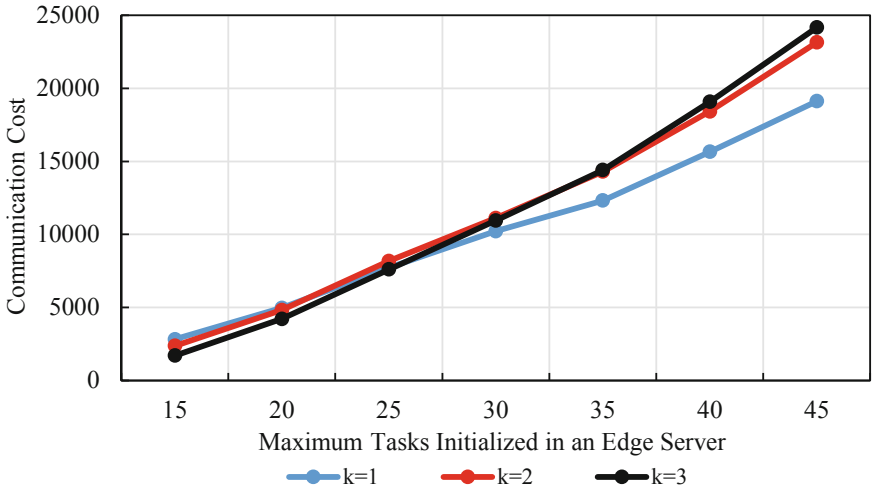

Fig. 7. Comparison of different price parameter k for communication cost in term of communication cost.

Communication cost is the fee charged by the operator for the transmission of data between edge servers. For the operator, its goal is to maximize its income, that is the communication cost. We alter the price parameter k of communication cost which is originally set as 1 to determine its influence while the neighborhood range set as 5 km. As is shown in Fig. 6, with the increase of k, the cost ratio increases because of the rise of communication cost. For the communication cost, as is illustrated in Fig. 7, when the maximum tasks initialized in a server z is set

as 15 or 20, we obtain the maximum communication cost with $k = 1$. However, $k = 2$ is proved to be a better choice when z equals to 25, 30, and $k = 3$ is the optimal solution for the case when $z = 35, 40, 45$. This is because: when the total number of task is small, it is not congested in the edge servers. An edge server prefers to handle the tasks by itself if the price for the transmission of data is high. Thus, setting a high price parameter is not beneficial. By contrary, when it is extremely congested in the servers, although the communication price is high, it is profitable by offloading the tasks to other edge servers to reduce the computation cost and improve the quality of service. Thus, for an operator, setting a price parameter such as $k = 3$ will result in a greater communication cost when it is extremely congested. In short, operator should decide the price parameter k dynamically according to the density of tasks and users in order to maximize its income incurred by the data transmission between edge servers.

5 Conclusion

In this paper, we study the task offloading decision problem in an edge computing network. We characterize the major challenges and convert the task offloading problem to a cost minimization problem. To solve the problem which is found to be NP-hard, an algorithm JCTO is proposed with the concept of the optimization of edge servers based on the neighborhood range. We evaluate the algorithm and the influence of different parameters by extensive experiments. Compared to the ITEM optimization algorithm, JCTO has a better performance in term of the cost minimized ratio after the optimization, which can be widely adopted for the edge computing networks with computational capacity constraint of edge servers.

Acknowledgement. This paper was partly supported by National Key RD Program of China Grant (No. 2018YFB2100302, No. 2017YFB1003000), NSFC Grant (No. 61671478, No. 61672342, No. 61532012, No. 61602303), the Science&Technology Innovation Program of Shanghai Grant (No. 17511105103, No. 18510761200) and the open research fund of National Mobile Communications Research Laboratory, Southeast University under Grant (No. 2018D06).

References

1. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper. Technical report (2019)
2. Chen, L., Xu, J.: Socially trusted collaborative edge computing in ultra dense networks. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, p. 9. ACM (2017)
3. Li, Y., Wang, S.: An energy-aware edge server placement algorithm in mobile edge computing. In: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 66–73. IEEE (2018)
4. Lyu, X., Ren, C., Ni, W., Tian, H., Liu, R.P.: Distributed optimization of collaborative regions in large-scale in homogeneous fog computing. IEEE J. Sel. Areas Commun. **36**(3), 574–586 (2018)

5. Pasteris, S., Wang, S., Herbster, M., He, T.: Service placement with provable guarantees in heterogeneous edge computing systems. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 514–522. IEEE (2019)
6. Pu, L., Chen, X., Xu, J., Fu, X.: D2D fogging: an energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. IEEE J. Sel. Areas Commun. **34**(12), 3887–3901 (2016)
7. Xiao, Y., Krunz, M.: QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
8. Wang, L., Jiao, L., He, T., Li, J., Mühlhäuser, M.: Service entity placement for social virtual reality applications in edge computing. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 468–476. IEEE (2018)
9. Wang, L., Jiao, L., Li, J., Mühlhäuser, M.: Online resource allocation for arbitrary user mobility in distributed edge clouds. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1281–1290. IEEE (2017)
10. Hou, I., Zhao, T., Wang, S., Chan, K., et al.: Asymptotically optimal algorithm for online reconfiguration of edge-clouds. In: Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 291–300. ACM (2016)
11. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 207–215. IEEE (2018)
12. Zhou, Z., Chen, X., Wu, W., Wu, D., Zhang, J.: Predictive online server provisioning for cost-efficient IoT data streaming across collaborative edges. In: Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 321–330. ACM (2019)
13. Sundar, S., Liang, B.: Offloading dependent tasks with communication delay and deadline constraint. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 37–45. IEEE (2018)
14. Krarup, J., Pruzen, P.M.: The simple plant location problem: survey and synthesis. Eur. J. Oper. Res. **12**, 36–81 (1983)