A Hybrid Task Offloading and Service Cache Scheme for Vehicular Edge Computing

Linyu Sun[1(✉)], Yancong Deng[2], Xingxia Dai[3], and Zhu Xiao[3]

[1] University of Southern California, Los Angeles, USA
linyusun@usc.edu
[2] University of California, San Diego, San Diego, USA
yad002@eng.ucsd.edu
[3] Shenzhen Research Institute of Hunan University, Changsha, China
zhxiao@hnu.edu.cn

Abstract. The development of 5G, IoT, and other technologies has promoted the emergence of emerging applications, including augmented reality, autonomous driving, and so on. These applications are usually delay-sensitive and energy intensive, which have strict delay constraints and need to consume a lot of computing resources. In order to ensure the quality of service of these applications, this study proposes a framework that combines task offloading and service caching in the local-edge-cloud collaboration system. In order to obtain a satisfactory offloading decision, this study first proposes a distributed task offloading algorithm based on non-cooperative game theory storage which makes the decision of local processing or offloading to the side server with the goal of minimizing system cost over time and energy consumption. Considering the limited storage resources of the edge, this study uses a 0–1 knapsack algorithm to realize dynamic service caching based on task popularity based on the original offloading decision. Based on the results of service caching, if the task initially decides to offload to the edge server and the edge server does not cache the services required by the task, the task will be unloaded locally or to the cloud.

1 Introduction

Due to the highly dynamic requirements of IoV network, a lot of studies focused on task offloading, resource management, Data scheduling, mobility management and application models. Vehicular edge computing (VEC) provides possible solutions to storage and compute resource allocation between vehicles and roadside units (RSU). The continuous growth in mobile applications has also caused an exponential increase in demand for higher computation capacity. Dedicated short-range communication (DSRC) enables vehicles' collaboration with RSU via infrastructure or infrastructure-independent communication. Meanwhile, the fast

Z. Xiao et al. (Eds.): ICECI 2022, LNICST 478, pp. 30–40, 2023.
https://doi.org/10.1007/978-3-031-28990-3_3

implementation of 5G technologies on cellular networks can use cellular vehicle-to-everything (C-V2X) and long-term evolution vehicles (LTE-V) to support low-latency and high-reliable communications for IoV.

Meanwhile, the limited cloud resources with radio access network also allows the resource-constrained mobile terminals (MT) to migrate part or all of the required tasks from the randomly arrived MTs to edge cloud for achieving spatial proximity service characterized by low latency and energy efficiency. Various intelligent transportation systems impose urgent demands on massive data transmission and intensive computing capacity such as self driving, collision warning and high resolution video crowdsourcing.

In this paper, a hybrid offloading scheme of both edge computing and service cache based on cloud computing has been proposed for vehicular network. First a distributed task offloading algorithm based on non cooperative game theory has been driven to make the decision for local processing or offloading to edge server with the goal of minimizing the total system cost over time and energy consumption. Considering the limited storage resources of the edge server, the study uses 0–1 knapsack algorithm to realize the dynamic service caching based on task popularity for the original offloading decision. Based on the result of service caching the offloading decision will be further adjusted to allow the computation mitigation with the cloud resources.

2 Related Work

Many researches have focus on fog computing paradigm for IoV to reach a better performance on time critical task [11–14]. Difference vehicle fog network (VFN) architectures have been proposed to cooperate with edge infrastructures for low-latency and high reliability services in IoV. Liu [1] investigated on adaptive offloading for time-critical tasks in heterogeneous environments and designed a vehicular fog computing based architecture to enable the cooperation among the cloud and fog nodes. Wang [2] proposed a dynamic reinforcement learning scheduling algorithm and deep dynamic scheduling algorithm to solve offloading decisions for fog computing based on Markov decision progress.

However the task assignment and resource allocation proposed for fog computing were designed based on the assumption of static distributed users. The uncertainty of mobility-driven user makes the task schedule more complex in the presence of heterogeneous radio, computing cost and cache resources. Considering the mobility of the vehicles, Saleem [3] designed a Genetic Algorithm-based evolutionary scheme to minimize the total execution latency. Then to obtain an effective task assignment with low complexity, a heuristic named mobility-aware task schedule has been proposed for the evaluation. The cooperative abilities among user terminals also plays a vital rule for the system performance. Peng [4] constructed an online resource coordinating and allocating scheme to minimize the network-wide response latency and energy consumption simultaneously by exploiting Lyapunov optimization theory, which offers partial offloading, cooperative scheduling and computation allocation for practical implementation.

The work in [9,10] studied task scheduling by replication strategy. Sun [5] designed a learning-based task replication algorithm with combinatorial multi-armed bandit theory, which considered the task failure probability as a constraint to guarantee reliability. Mohamed [6] considered multiple sets of users forming service groups due to the participation of each group in the same synchronized activities and formulated the problem as an integer non-linear problem. Then a task offloading and service replication scheme is proposed to minimize the response time while satisfying the group delay requirement.

While researches which are mentioned above are based on cooperation between edge nodes and fog nodes, there are also a number of studies investigated on task offloading and resource allocation based on the vehicular ad-hoc network (VANET). Despite the variety of studies for vehicle to vehicle communication(V2V), the algorithms can be roughly divided into two categories: topology based routing and geographic based routing. In V2V, nodes have high mobility and the topology changes frequently with the time, which means the topology based routing protocols are not suitable in the real situation [7]. Wang [8] investigated geographic based routing protocol and proposed a V2V hybrid model based on the Greedy Perimeter Stateless Routing algorithm, which leads to a better path decision and transmission efficiency.

3 System Model

In this section, first the algorithm structure of hybrid task offloading and service caching scheme is described. Then the computation models of two decision making stage are presented. All the notations used in the system model are listed in (Table 1).

Table 1. Frequently Used Notations

Term	Description
$\mathcal{T}$	total offloading task set
$\mathcal{O}$	task set
d_i	edge execution cost
l_i	local execution cost
s_i	task strategy cost
$\mathcal{G}$	offloading strategy combination set
$\mathcal{H}$	cost compare vector
$\mathcal{Q}$	loss vector
$\mathcal{R}$	potential decision set
C	cache capacity

Figure 1 shows the proposed Hybrid Task Offloading and Service Cache Scheme for Vehicular Edge Computing, which consists of cloud pool, edge server and mobile device set $\mathcal{N} = \{1, 2, ..., N\}$. For the task to be unloaded to the

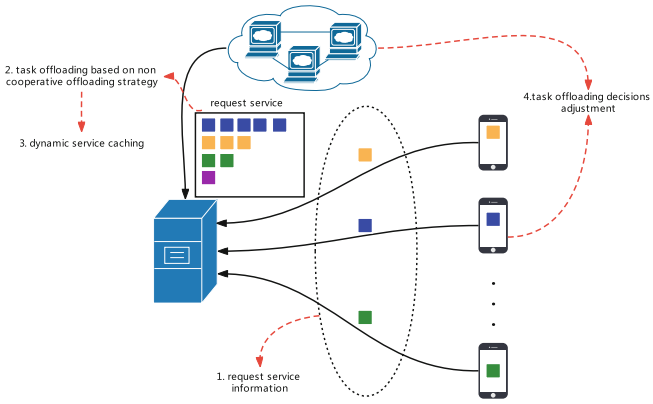

Fig. 1. Illustration of hybrid task offloading and service caching scheme

side server, the IoT device first notifies the side server of the service information requested by the task before the task is transmitted. After receiving the information, the edge server determines the popularity of the service according to the frequency of each service request. The more requests, the higher the popularity, so there are more opportunities to cache in the edge server. Considering the popularity of the service and the limited storage space of the edge server, the edge server downloads the service from the cloud to the cache. If the service requested by the task is cached in the edge server, the task can be processed in the edge; When the requested service is not cached in the edge server, the IoT device needs to adjust the offloading decision of the task, that is, decide whether to unload the task to the cloud or process it locally.

Assuming during one specific time slot every mobile device $n \in \mathcal{N}$ sends one task offloading request to the edge server. The total offloading task set is $\mathcal{T} = \{t_1, t_2 \ldots, t_N\}$. Task set $\mathcal{O} = \{a, b, \ldots\}$ contains all possible task where $a, b, \ldots$ are different task type. $t_i \in \mathcal{O}, \forall i \in \mathcal{N}$.

3.1 Task Offloading Model

Figure 2 shows the algorithm flow for the proposed scheme. The algorithm contains three steps. First, a non-cooperation offloading method is adopted to minimize the total system cost at the edge server. The total system cost is a combination of both local operation cost and geographic-based cost, which is dependent on the offloading strategy chosen for each task.

Each task will be decided whether it will be offloaded to the edge server or reserved for the local process based on its local operation cost l and edge operation cost d, which both are the weighted sum of process time cost and energy cost. Thus the task strategy cost s can be represented as

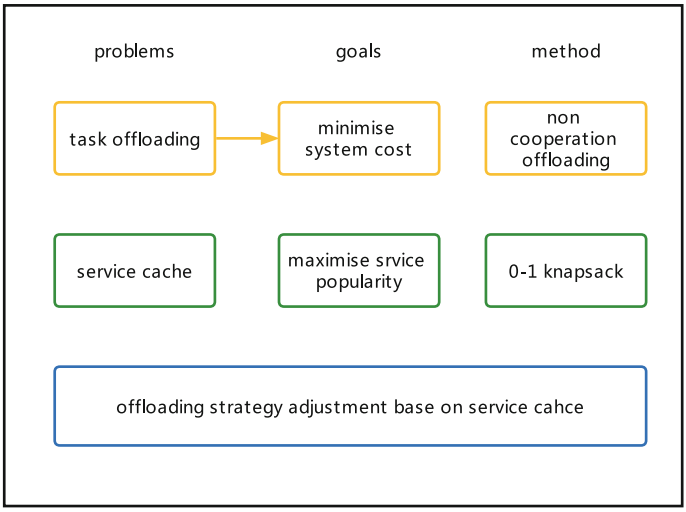

Fig. 2. Illustration of algorithm structure

$$s_i = \begin{cases} l_i & \text{local execution} \\ d_i & \text{edge execution} \end{cases} \quad \forall i \in \mathcal{N} \tag{1}$$

Since the offloading choice of each task can be different in every offloading strategy set. The weighted cost of k^{th} strategy set $g^k \in \mathcal{G}$ can be represented as:

$$g^k = \sum_{i=1}^{N} s_i^k + b \tag{2}$$

where s_i^k stands for the i^{th} task cost for k^{th} strategy set and b is a constant and $\mathcal{G}$ stands for all possible combination of N strategies.

The non-cooperative task offloading strategy is based on the iteration process. During the first iteration process, the task offloading strategy will be chosen randomly for the initialization, assuming the set cost is c^j. The cost compare vector based on j^th strategy choice is $\mathcal{H} = \{h_1, h_2, \ldots h_N\}$ where $h_i = g^j - g^i$. If the current offloading choice j has less cost, the value of h would be larger.

Thus, the loss vector $\mathcal{Q}$ for e^{th} iteration process can be represented as:

$$\mathcal{Q}_e = \begin{cases} \mathcal{H}_e & e = 1 \\ \mathcal{Q}_{e-1} + \frac{1}{e*(\mathcal{H}_e - \mathcal{Q}_{e-1})} & e \neq 1 \end{cases} \tag{3}$$

where $q_{i-e} \in \mathcal{Q}_e$ is the loss score of i^{th} offloading strategy compared with current strategy during the e^{th} iteration process. Thus the offloading strategy for e^{th} iteration process can be chosen from the decision set $\mathcal{R}_e \subseteq \mathcal{Q}_e$, where all elements q_i belong to $\mathcal{R}_e$ have positive value.

3.2 Dynamic Service Caching and Task Offloading Decision Adjustment

Algorithm 1 two stages task offloading algorithm

for $t_i \in \mathcal{T}$ **do**
  calculate the local execution cost l_i and edge execution cost e_i
end for
form the weighted strategy cost set $\mathcal{G}$ based on equation (2)
for $e = 1$,$e \leq$iteration time **do**
  update the comparison vector $\mathcal{H}_e$ based on $h_i = g^j - g^i$
  update the loss vector $\mathcal{Q}_e$ based on the equation (3)
  form the decision set $\mathcal{R}_e$, $r_i = q_i$ when q_i is greater or equal than 0, $r_i = 0$ when q_i is less than 0. For each r_i, assign a pickup probability based on $r_i / \sum_{k=1}^{N} r_k$. The next offloading strategy will be generated based on the probability distribution of $\mathcal{R}_e$
end for
based on the required service type for each task, from the storage cost set $\mathcal{W}$ and popularity profit set $\mathcal{P}$, which are corresponded to every task in set $\mathcal{N}$
follows the recursion of equation (5), find the $f(p_i, C)$, which would lead to its corresponding dynamic caching strategy.
based on the offloading strategy and dynamic caching strategy, there will be a further adjustment if a task is offloaded to the edge server and the edge server does not have a cache of it.
the edge server will compare the task strategy cost for both local and cloud execution to decide to final choice for the noncached task.

Assuming the edge server has cache capacity C. The task set $\mathcal{T}$ has been offloaded to the edge server during the specific time slot. Assuming each task $t_i \in \mathcal{T}$ has a storage cost w_i and popularity profit p_i. The relation between the summation of all task weights and the cache capacity can be expressed as $\sum_{i=1}^{N} w_i > C$ where we assume that $w_i < C, i \in \mathcal{N}$.

Based on the task offloading results the 0–1 knapsack is used to maximize the service popularity by caching the popular tasks in the edge server with limited storage space.

Introducing the binary decision variables x_i, with $x_i = 1$ if task is selected, and $x_i = 0$ otherwise. The dynamic service caching problem (DSCP) can be summarised as:

$$\begin{aligned} &\text{maximize} \sum_{i=1}^{N} p_i x_i \\ &\text{subject to} \sum_{i=1}^{N} w_i x_i \leq C, x_i \in \{0,1\}, i \in \mathcal{N} \end{aligned} \tag{4}$$

The optimization problem is solved by a dynamic programming approach. The state $f(p_i, \widetilde{C})$ represents the maximum popularity profits of storage cost $\widetilde{C}$ considering all tasks from 1 to i.

The recursion process for DSCP can be denoted as:

$$f(p_i, \widetilde{C}) = \begin{cases} f(p_{i-1}, \widetilde{C}) & w_i > \widetilde{C} \\ \max(f(p_{i-1}, \widetilde{C}), & \\ f(p_{i-1}, \widetilde{C} - w_i) + p_i) & w_i \leq \widetilde{C} \end{cases} \tag{5}$$

The profit of each task is determined by the number of tasks of the same type in the total offloading task set $\mathcal{T}$. The more tasks of the same kind, the higher the profit of the task, and the greater the chance of being cached by the edge server.

If the task is not cached in edge, the server will compare the cloud processing cost with the local processing cost to determine the further adjustment policy for the offloading strategy.

The complete workflow for both non-cooperative tasks offloading scheme and dynamic service caching along with the final decision adjustment scheme is described in Algorithm 1 as follows:

4 Simulation Results and Analysis

The simulation model is built based on the system architecture proposed in Sect. 3. In the default setting, assume the total number of devices N = 5. Task set $\mathcal{O}$ has 5 different service types. The consumption of CPU computation resources and the storage space of every task type is also different. tasks in set $\mathcal{N}$ will be initialized with a random type from set $\mathcal{O}$.

To compare the performance of the proposed scheme, three solutions have also been implemented, which are described as follows:

1. *TOCS:* This algorithm comprehensively considers task offloading and service caching in the mobile edge computing network, without cloud server assistance.
2. *TO:* This algorithm considers the offloading problem in the mobile edge computing network, without dynamic service caching.
3. *LP:* This algorithm processes all tasks in local IoT devices.

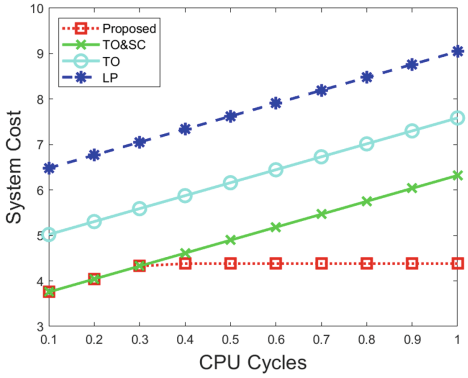

Fig. 3. Comparison of system cost under different CPU cycles

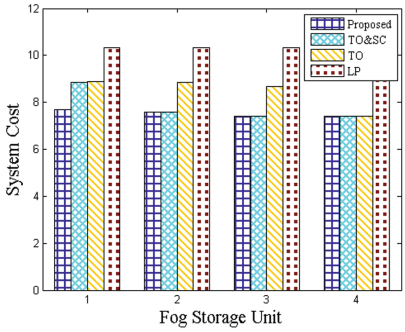

Fig. 4. Comparison of system cost under different storage unit

The experiment changed the number of CPU computing resources required for the fifth task from range (0.1GHz:1GHz). Meanwhile, the local energy consumption and the task size remain the same for tasks. Figure 3 shows the comparison of system cost between the proposed scheme and three different schemes.

With the growth of the CPU cycles, the system cost of *LP* and *TO* grows linearly. The *LP* has the highest cost among all CPU cycles since the scheme does not involve any offloading technique.

Figure 4 compares the system cost for all scheme under assumption of different storage unit. During the first stage all schemes share the same initial condition for task offloading. During the dynamic service caching stage the capacity of storage unit on fog side has been changed in order to compare the system performance.

LP maintains the highest system cost. The proposed scheme has the best performance when the capacity of fog storage unit is small. It has the same performance with other two scheme when the storage capacity is large, generally the proposed scheme has better system cost performance for all fog storage capacities.

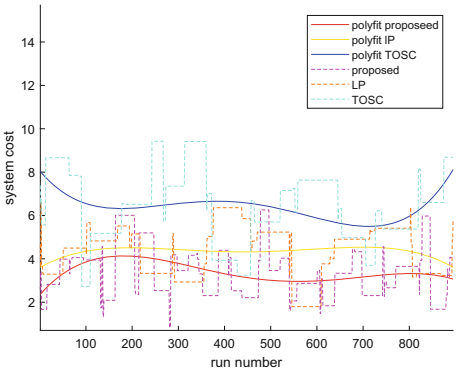

Fig. 5. Comparison of system performance under random task set

Figure 5 shows the difference of system cost under the initialization of random task set. Mobile device set $\mathcal{N} = \{1, 2, ..., 5\}$, number of task type O for task set $\mathcal{O}$ is initialized with random setting where $O \geq N$.

All three schemes share the same initialization condition. The experiment was run 900 times in order to show the statistical performance of all schemes. the dotted line from the figure shows the real time system cost and the solid line are the polynomial fits of the real time system cost.

From the figure it is clear that LP has the highest system cost both in real time and statistical performance. The real time performance of TOSC and proposed scheme are very close but judged by the polynomial fit the proposed scheme has a general better performance on system cost.

5 Conclusion

In this paper, a hybrid offloading scheme of both edge computing and service cache based on cloud computing has been proposed for vehicular network. During each iteration process, the task offloading decisions will be made based on the evaluation of the last decision sets. The complete computation model and the Algorithm process of the two stages task offloading strategy are proposed. Finally, a simulation model is built to compare the performance of the proposed scheme along with three different solutions. The results prove that the proposed scheme has better performance on system cost under different CPU cycles and different storage unit in the edge server. In future directions, Since MEC and D2D offloading are two promising paradigms in the industrial Internet of Things [15]. With the insperation of Heterogeneous Small Cell Networks, heterogeneous tasks with diverse delay requirements, data size, and reliability constraints should be considered [16,17]. Also the task offloading can be improved by incorporating mmWave, massive MIMO. These challenging and interesting extensions are yet left for the future work.

References

1. Liu, C., Liu, K., Guo, S., Xie, R., Lee, V.C.S., Son, S.H.: Adaptive offloading for time-critical tasks in heterogeneous internet of vehicles. IEEE Internet Things J. **7**(9), 7999–8011 (2020). https://doi.org/10.1109/JIOT.2020.2997720
2. Wang, Y., Wang, K., Huang, H., Miyazaki, T., Guo, S.: Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. IEEE Trans. Industr. Inf. **15**(2), 976–986 (2019). https://doi.org/10.1109/TII.2018.2883991
3. Saleem, U., Liu, Y., Jangsher, S., Li, Y., Jiang, T.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wireless Commun. **20**(1), 360–374 (2021). https://doi.org/10.1109/TWC.2020.3024538
4. Peng, J., Qiu, H., Cai, J., Xu, W., Wang, J.: D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC. IEEE Trans. Wireless Commun. **20**(8), 4858–4873 (2021). https://doi.org/10.1109/TWC.2021.3062616
5. Sun, Y., Zhou, S., Niu, Z.: Distributed task replication for vehicular edge computing: performance analysis and learning-based algorithm. IEEE Trans. Wireless Commun. **20**(2), 1138–1151 (2021). https://doi.org/10.1109/TWC.2020.3030889
6. Mohamed, S.A., Sorour, S., Hassanein, H.S.: Group-delay aware task offloading with service replication for scalable mobile edge computing. In: GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–6 (2020). https://doi.org/10.1109/GLOBECOM42002.2020.9348241
7. Wu, T.-Y., Wang, Y.-B., Lee, W.-T.: Mixing greedy and predictive approaches to improve geographic routing for VANET. Wireless Commun. Mob. Comput. **12**(4), 367–378 (2012)
8. Wang, X., Deng, Y., Wang, T., Zhang, Y., Jiang, H.: A hybrid vehicle-to-vehicle transmission model for vehicular networks. In: 2021 the 11th International Conference on Information Communication and Management (ICICM 2021), pp. 34–40. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3484399.3484404
9. Saeik, F., et al.: Task offloading in edge and cloud computing: a survey on mathematical, artificial intelligence and control theory solutions. Comput. Netw. **195**, 108177 (2021) ISSN 1389-1286
10. Chen, C.-L., Brinton, C.G., Aggarwal, V.: Latency minimization for mobile edge computing networks. IEEE Trans. Mob. Comput. 1 (2021). https://ieeexplore.ieee.org/document/9557844
11. Siriwardhana, Y., Porambage, P., Liyanage, M., Ylianttila, M.: A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects. IEEE Commun. Surv. Tutor. **23**(2), 1160–1192 (2021)
12. Liu, L., et al.: Vehicular edge computing and networking: a survey. Mob. Netw. Appl. **26**(3), 1145–1168 (2021)
13. Peng, J., Qiu, H., Cai, J., Xu, W., Wang, J.: D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC. IEEE Trans. Wireless Commun. **20**(8), 4858–4873 (2021)
14. Raza, S., Liu, W., Ahmed, M., et al.: An efficient task offloading scheme in vehicular edge computing. J. Cloud Comput. **9**, 28 (2020). https://doi.org/10.1186/s13677-020-00175-w

15. Dai, X., et al.: Task co-offloading for D2D-assisted mobile edge computing in industrial internet of things. IEEE Trans. Industr. Inform. (2022). https://doi.org/10.1109/TII.2022.3158974
16. Jiang, H., Xiao, Z., Li, Z., Xu, J., Zeng, F., Wang, D.: An energy-efficient framework for internet of things underlaying heterogeneous small cell networks. IEEE Trans. Mob. Comput. **21**(1), 31–43 (2022)
17. Jiang, H., Dai, X., Xiao, Z., Iyengar, A.: Joint task offloading and resource allocation for energy-constrained mobile edge computing. IEEE Trans. Mob. Comput. (2022). https://doi.org/10.1109/TMC.2022.3150432