Satisfaction Optimization in Failure-Aware Vehicular Edge Computing

Chaogang Tang*, Huaming Wu†, Chunsheng Zhu‡
*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
†Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
‡College of Big Data and Internet, Shenzhen Technology University, 518118, Shenzhen, Guangdong, China
cgtang@cumt.edu.cn, whming@tju.edu.cn, chunsheng.tom.zhu@gmail.com

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001725

***Abstract*—Vehicular edge computing (VEC) has gained worldwide attention in both academia and industry. Current works on VEC mainly focus on task offloading and resource allocation to improve the performance of VEC systems, but seldom consider the satisfaction level of vehicles. Whereas, the satisfaction level of vehicles has been playing an important role in stimulating vehicles to pursue better quality of experience by task offloading and service outsourcing operations. In the meanwhile, there is an inescapable fact, i.e., the task execution in VEC may fail due to various reasons, and thus it is important to incorporate the failure-resisted task offloading into the failure-prone VEC system. In this paper, we aim to maximize the satisfaction of all the vehicles, while considering the potential failures in VEC. Specifically, we model satisfaction optimization as a multiple knapsack problem and further put forward a greedy heuristic approach to solve this problem in polynomial time. Extensive simulation is carried out to validate the efficiency of our approach in terms of the optimal values and the running time. The simulation results have shown that our approach can achieve a better result compared to other benchmarks.**

***Index Terms*—Vehicular edge computing, service provisioning, task offloading, failure, satisfaction optimization**

I. Introduction

The computing resources are extended from the cloud center to the edge of networks, resulting in two representative computing paradigms, i.e., edge computing [1] and fog computing [2]. Both of them can provision computing resources at the network edge with the aim to satisfy the strict latency requirements of internet of things (IoT) tasks. Vehicular edge computing (VEC) as the subsystem of edge computing has also gained worldwide attention in both academia and industry. The explosive growth in vehicular tasks with regards to (w.r.t.) the number and variety has led to staggering demands for computational resources, which has posed great pressure over their own computing capabilities. To ease such pressure and further shorten the response delay, IoT tasks can be outsourced and offloaded to VEC for execution. In particular, VEC can serve the nearby vehicles in a time-sensitive fashion by deploying the edge server at RSUs [3].

Works on VEC mainly focus on task offloading and resource allocation, aiming to improve the performance of VEC systems w.r.t. response delay and energy consumption [4]–[7]. However, attention only paid to improve the performance of VEC systems is far from adequate, since we also need to consider the quality of experience (QoE) of vehicles with offloading requests, apart from the quality of services (QoS) for the computing resources provisioned at VEC systems. Similar to cloud computing, the computing resources at RSU are provisioned in a pay-as-you-go scheme. It seems reasonable to assume that, other things being equal, vehicles with task offloading requests would prefer to be served by those resource providers which bring them better QoE. The QoE usually refers to the impression of vehicles on the task execution in VEC. In particular, we assume that the QoE of vehicles in this paper only involves two aspects, i.e., the response delay and the expenditure for task offloading, respectively. The QoE of vehicles can indicate the satisfaction level of vehicles. Generally, better QoE of vehicles brings about a higher satisfaction level of vehicles. Therefore, in this paper, we pay our attention to the satisfaction optimization for the vehicles with offloading requests in VEC, and argue that it is worthy of further investigation, since drivers have a remarkable instinct of weighing advantages and disadvantages.

In the meanwhile, we notice that the existing works in VEC seldom consider an inescapable fact, i.e., the task execution in VEC may fail due to various reasons coming from either the unstable network connections or the computing resource-constrained edge nodes [8], [15]. It is necessary to consider these failures when tasks are offloaded and executed in VEC, since it usually takes time to resume task execution in VEC. Note that we only consider the failures from the edge nodes such as temporarily insufficient memory and software failures, and further assume that these failures are recoverable in this paper. The recovery time for each failure degrades not only the QoS of computing services at the edge, but also the satisfaction level of vehicles.

Therefore, it is necessary to incorporate the failure-resisted task offloading into the failure-prone VEC system, in the pursuit of satisfaction maximization for the vehicles with offloading requests. To that end, we put forward a satisfaction maximization scheme in failure-aware VEC in this paper. Specifically, the contributions of this paper are threefold, given below:

- Unlike traditional service provisioning approaches in VEC that concentrate primarily on the performance improvement for VEC systems, we turn our attention to the satisfaction optimization for vehicles, in view of its vital

978-1-6654-3540-6/22 © 2022 IEEE

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.

importance in stimulating vehicles to pursue better QoE by task offloading and service outsourcing operations.
- In this paper, we aim to maximize the satisfaction values for all the vehicles in VEC and model it as a multiple knapsack problem (MKP) that is well-known as an NP-hard problem. In the meanwhile, we generalize the traditional task offloading model by considering potential task execution failures in VEC. Furthermore, we put forward a greedy heuristic approach to solve this problem in polynomial time.
- Extensive simulation is carried out to validate the efficiency of our approach in terms of the optimal values and the running time. The simulation results have shown that our approach can achieve a better result in terms of the optimal values.

The rest of the paper is arranged as follows. We review some related works in Section II and introduce our system model and problem formulation in Section III. The algorithm design is presented in Section IV with the simulation evaluation in Section V. The conclusion finally comes in Section VI.

II. Related Works

It becomes increasingly dominating that vehicular applications and tasks are offloaded and executed outside themselves, owing to the fact that these "computers on wheels" lack sufficient computing resources to accomplish increasingly complicated tasks. For instance, to accomplish the vehicular tasks more efficiently, Ng *et al.* [6] adopted a double auction mechanism to schedule the computing resources at the edge for the nearby vehicles. Both the requirements for the vehicles and the resource pricing are considered in this paper and the simulations have proven their advantages. Considering the multiple features of VEC networks, e.g., unstable wireless links, time-varying topologies, and different move models of vehicles, it is pretty difficult to handle online task offloading efficiently. Although several promising technologies can be applied including heuristic or machine learning-related approaches, these approaches are limited by their own shortcomings, e.g., either slow convergence rate or low searching efficiency. To tackle these issues, Wang *et al.* [7] handle online task offloading based on imitation learning and try to achieve near-optimal performance from the initial stage. Particularly, they obtain the optimal scheduling policy offline based on given samples, and train agent policies online from the expert's demonstration.

In the meanwhile, there are also several works in the device-edge-cloud systems which focus on satisfaction-related issues [9]–[11]. For instance, Wang *et al.* [10] tried to optimize the satisfaction for Service-Level Agreement (SLA) in the device-edge-cloud systems. They have jointly optimized the offloading decision, task assignment and task ordering. By formulating the optimization as a binary nonlinear programming, they have adopted an integer particle swarm optimization to solve it.

On another hand, IoT tasks from end users are usually featured by delay-sensitive requirements. To depict the impression towards how these tasks are accomplished, Li *et al.* [11] strive to maximize the accumulative satisfaction values for these end devices. Specifically, they consider both the dynamic and static offloading requests, and adopt heuristic algorithms to efficiently solve the formulated problem. Simulation evaluation reveals that their algorithms are efficient to improve the user satisfaction.

In contrast, we consider not only the satisfaction of vehicles but also the potential failures in VEC. Such factors can easily lengthen the response delay and degrade the satisfaction of vehicles. To the best of our knowledge, we are the first to pay attention to the satisfaction-related issue in failure-aware VEC systems.

III. System Model

The proposed system model mainly consists of smart vehicles denoted by $\mathcal{V} = \{v_1, \cdots, v_m\}$, performance-enhanced RSU denoted by $\mathcal{R}$, and the edge nodes $\mathcal{S} = \{s_1, \cdots, s_n\}$. m and n denote the number of vehicles and computing nodes, respectively. $\mathcal{R}$ is empowered with powerful computing capabilities such that RSU has not only the networking capabilities, but also the computing capabilities. The edge nodes are deployed in the vicinity of $\mathcal{R}$ and they are interconnected with high-speed wired links (e.g., high-capacity optical fibers). Assume that these edge nodes are heterogeneous in terms of the amount of computing resources, the processing frequencies, and the computing resource pricing. The task set $\mathcal{T}$ is indexed by $\mathcal{T} = \{t_1, \cdots, t_m\}$, where t_i is the task offloaded from vehicle v_i. t_i can be expressed by a tuple $(I_i, C_i, D_i, W_i, E_i)$, where I_i is the size of task-input data, C_i is the amount of computing resources in terms of CPU cycles required for the task, D_i is the expected response delay, W_i is the dwelling time of vehicle v_i within the coverage of $\mathcal{R}$ that can be estimated easily [13], and E_i denotes the biggest expense that v_i can pay. It is well known that the computing resources at RSU are provisioned in a pay-as-you-go scheme. In VEC, vehicles need to pay for the requested computing resources and thus expect that the tasks can be accomplished satisfactorily. Generally, both shorter response delays and fewer costs can make the vehicles more satisfactory.

In addition, the information of tasks $\mathcal{T}$ can be learned by $\mathcal{R}$ from the beacon packet dissemination between vehicles and RSU. After gathering this information, $\mathcal{R}$ is responsible for distributing these tasks to different edge nodes for execution. Define $x_{i,j}$ as a binary decision variable to represent the offloading decision of task t_i. Particularly, $x_{i,j} = 1$, if task t_i is designated to s_j for execution; and $x_{i,j} = 0$, otherwise.

A. Response Delay Model

Assume that the task t_i is performed by the edge node s_j. Then, the response delay of task t_i, denoted by $d_{i,j}$, usually includes the following five parts. The first part is the transmission delay $d_{i,j}^{trs}$, i.e., the time taken to offload the task from vehicle v_i to $\mathcal{R}$, and can be calculated as $d_{i,j}^{trs} = I_i / r_i$, and r_i is the transmission rate of the wireless channel between v_i and $\mathcal{R}$,

$$r_i = B \log_2(1 + \frac{P_i G_i}{\sigma^2}) \tag{1}$$

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.

where P_i denotes the transmission power of vehicle v_i, G_i is the channel gain between vehicle v_i and $\mathcal{R}$, B is the bandwidth for the wireless channel and σ^2 is the noise power. The second part is the propagation delay $d^p_{i,j}$, i.e., the time taken to propagate the task to edge node s_j, and can be expressed as $d^p_{i,j} = I_i/H_j$, where H_j denotes the propagation rate between $\mathcal{R}$ and s_j. It shall be noted that the propagation delay could be much smaller than the transmission delay, owing to the high-speed optical fiber links between RSU and the edge nodes.

The third part is the calculation delay $d^c_{i,j}$, i.e., the time taken to accomplish the task at edge node s_j. When task t_i is allocated to s_j, the edge node will create the virtual resources for the task. However, the task execution may fail due to some technical hitches and we further assume such failures are recoverable. If there are no failures during the task execution, the calculation delay is $d^{c,no}_{i,j} = C_i/f_j$, where f_j is the processing frequency of the edge node s_j. On the other hand, if there are failures during task execution, the calculation delay needs to be discussed as follows. We assume that the failures of task t_i at s_j follow a Poisson process [14], [15] and the failure rate is $\lambda_{i,j}$ that can be estimated based on historical experience. Let $\mathcal{N}(t)$ denote the number of failures during the time $(0, t]$. Hence, the probability that k failures occur within the time interval $d^{c,no}_{i,j}$ is $\mathrm{P}\{\mathcal{N}(d^{c,no}_{i,j}) = k\} = ((\lambda_{i,j} d^{c,no}_{i,j})^k/k!)e^{-\lambda_{i,j} d^{c,no}_{i,j}}$, and $\mathbb{E}[\mathcal{N}(d^{c,no}_{i,j})] = \lambda_{i,j} d^{c,no}_{i,j}$. Generally, it will take some time (i.e., recovery time) to resume the task execution, with the help of some checkpointing and rollback/roll-forward technologies [14].

In this context, let $\mathcal{R}_k(d^{c,no}_{i,j})$ denote the recovery time of the kth failure for task t_i at s_j, and assume it follows an exponential distribution with the recovery rate $\mu_{i,j}$. Assume that the total $\mathcal{N}(d^{c,no}_{i,j})$ failures are independent of each other. So the recovery times are independent and identically distributed and the total recovery time $\mathcal{R}(d^{c,no}_{i,j}) = \sum_{k=1}^{\mathcal{N}(d^{c,no}_{i,j})} \mathcal{R}_k(d^{c,no}_{i,j})$ follows Gamma distribution with two parameters $\lambda_{i,j} d^{c,no}_{i,j}$ and $\mu_{i,j}$. Thus, $\mathbb{E}[\mathcal{R}(d^{c,no}_{i,j})] = \lambda_{i,j} d^{c,no}_{i,j}/\mu_{i,j}$. Therefore, the average calculation delay including the norm calculation delay and the recovery time, is:

$$d^c_{i,j} = d^{c,no}_{i,j} + \frac{\lambda_{i,j} d^{c,no}_{i,j}}{\mu_{i,j}} = \frac{C_i}{f_j}(1 + \frac{\lambda_{i,j}}{\mu_{i,j}}). \tag{2}$$

The fourth part is the result propagation delay, i.e., the time taken to propagate the execution result from s_j to $\mathcal{R}$, and can be expressed as $d^{rp}_{i,j} = \tilde{O}_i/H_j$, where $\tilde{O}_i$ denotes the result size of task t_i. The fifth part is the returning delay, i.e., the time taken to transmit the execution result back to v_i from $\mathcal{R}$, and can be expressed as $d^r_{i,j} = \tilde{O}_i/\tilde{r}_i$, where $\tilde{r}_i$ is the transmission rate from $\mathcal{R}$ to vehicle v_i. Therefore, the total response delay can be expressed as:

$$d_{i,j} = d^{trs}_{i,j} + d^p_{i,j} + d^c_{i,j} + d^{rp}_{i,j} + d^r_{i,j}. \tag{3}$$

It shall be noted that the latter two parts i.e., the result propagation delay and the returning delay are often omitted, due to the fact that the size of execution result is much smaller than the size of task-input data. Usually, the response delay can affect the QoE of vehicles to a great extent. For instance, the QoE of one vehicle will decline as the response delay increases. If the response delay exceeds the dwelling time of the vehicle, the QoE may reduce to the minimum since the vehicle cannot receive the result and considers the task offloading to be unsuccessful. Therefore, we have the following definition.

Definition 1: **Delay Satisfaction Index (DSI)**. The delay satisfaction index, denoted by $U_{i,j}$, indicates the impression of the vehicle v_i on the edge node s_j that performs its task t_i. In particular, the DSI value is defined as a measurable satisfaction level of v_i towards s_j in terms of the response delay, given as:

$$U_{i,j} = \begin{cases} 1 & \text{if } d_{i,j} \le D_i, \\ \frac{W_i - \delta(d_{i,j} - D_i)}{W_i} & \text{if } D_i < d_{i,j} \le D_i + W_i/\delta, \\ 0 & \text{if } d_{i,j} > D_i + W_i/\delta. \end{cases} \tag{4}$$

Based on the above definition, we can see that the DSI value will reach the maximum (i.e., 1) if the task is accomplished within the expectation D_i. Otherwise, the value will decay linearly with a slope of $\delta(> 0)$ until it is equal to 0, as the response latency increases. Therefore, the impression of v_i on s_j declines as the DSI value decreases. In other words, the larger the DSI value, the higher the satisfaction level in terms of response delay.

B. Cost Model

The computing resources in both cloud computing and VEC are provisioned in a pay-as-you-go way. As a result, the expenditure of task offloading also has a great influence on the QoE of vehicles. Generally, the less the costs on task offloading, the better the QoE of one vehicle. On the other hand, if the expenditure of task offloading exceeds the budget of the vehicle, the QoE may reduce to the minimum, and the vehicle may consider the task offloading to be unsuccessful. Let $c_{i,j}$ denote the expenditure of vehicle v_i if the corresponding task t_i is executed at the edge node s_j, and $c_{i,j}$ can be defined as $c_{i,j} = \xi_j C_i$, where ξ_j represents the price per computing resource. Then, we have the following definition.

Definition 2: **Cost Satisfaction Index (CSI)**. The cost satisfaction index, denoted by $Q_{i,j}$, indicates the impression of v_i on the edge node s_j towards the expenditure of task offloading. In particular, the CSI value is defined as a measurable satisfaction level of v_i towards s_j in terms of the expenditure, given as:

$$Q_{i,j} = \begin{cases} \frac{E_i - c_{i,j}}{E_i} & \text{if } c_{i,j} \le E_i, \\ 0 & \text{if } c_{i,j} > E_i. \end{cases} \tag{5}$$

According to the above definition, the CSI value varies from 0 to 1. Generally, the larger the CSI value, the higher the satisfaction level towards the expenditure of task offloading. Given the two definitions 1 and 2, we have the following definition:

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.

Definition 3: **Overall Satisfaction Index (OSI)**. The overall satisfaction index, denoted by $\mathcal{O}_{i,j}$, indicates the overall impression of v_i on the edge node s_j towards both the response delay and the expenditure of task offloading. In particular, the OSI value is defined as a measurable satisfaction level of v_i towards s_j, given as:

$$\mathcal{O}_{i,j} = w_d U_{i,j} + w_c Q_{i,j}, \tag{6}$$

where $w_d, w_c \in (0,1)$ and $w_d + w_c = 1$. The overall satisfaction index is the weighted sum of the delay satisfaction index and cost satisfaction index. The preference towards OSI can be tuned by adjusting the weights. For example, if vehicle v_i cares about the response delay more than the expenditure, it can achieve this goal by increasing w_d.

C. Problem Formulation

The goal of this work is to maximize the OSI values of all the vehicles in the failure-aware VEC system. Define $\mathcal{O}(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}\mathcal{O}_{i,j}$ as the sum of OSI values for all the vehicles, where $\boldsymbol{x}$ is the $m \times n$ matrix of which the element $x_{i,j}$ is the task offloading decision for the task t_i at the edge node s_j. Therefore, the optimization problem in this paper is formulated as below:

$$(P1) \quad \max_{\boldsymbol{x}} \mathcal{O}(\boldsymbol{x})$$

$$\text{s.t.}: \quad \sum_{j=1}^{n} x_{i,j} = 1, \quad 1 \le i \le m \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} \le h_j, \quad 1 \le j \le n \tag{8}$$

$$x_{i,j} \in \{0,1\}, \quad 1 \le i \le m, 1 \le j \le n \tag{9}$$

where the constraint (7) guarantees that the task t_i from v_i can be offloaded to only one edge node for execution. Due to the limited computing resources in VEC compared to cloud computing, we assume that none of the edge nodes can perform the tasks from all the vehicles in the optimization period. Such a constraint can be satisfied by the inequation (8), and h_j denotes the maximal number of tasks s_j can perform in the optimization period.

Proposition 1: The satisfaction maximization for all the vehicles in VEC (i.e., the problem $P1$) is *NP-Hard*.

Proof: The above satisfaction optimization problem $P1$ is actually an instance of the MKP that is well-known as an NP-hard problem [16]. Considering two sets of knapsacks $\mathcal{K}$ and items $\mathcal{I}$, respectively, each knapsack $k \in \mathcal{K}$ can pack finite items based on its knapsack capacity a_k and each item $i \in \mathcal{I}$ has a size s_i and profit p_i. MKP tries to maximize the total profits of the items when they are packed into the knapsacks, i.e.,

$$(P2) \quad \max \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}} x_{k,i} p_i \tag{10}$$

$$\text{s.t.}: \quad \sum_{i\in\mathcal{I}} x_{k,i} s_i \le a_k, \quad \forall k \in \mathcal{K} \tag{11}$$

$$x_{k,i} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{I} \tag{12}$$

Actually, problem $P1$ is equivalent to $P2$, which can be proven as follows. The set of edge nodes $\mathcal{S}$ and the set of vehicles $\mathcal{V}$ in the system model can be considered to be $\mathcal{K}$ and $\mathcal{I}$ in $P2$, respectively. The constraint (8) in our problem $P1$ corresponds to the constraint (11) in the sense that each task has a size 1 which is corresponding to the item size. Each edge node has a constraint h_j that is corresponding to the backpack capacity a_k. The number of tasks offloaded to each edge node should not exceed the maximal number of tasks the node can support. This constraint actually corresponds to the knapsack constraint shown in Eq. (11). The OSI value for each task executed at the edge can be regarded as the profit of each item in problem $P2$. Therefore, $P1$ is equivalent to $P2$ in the sense that m items are packed to n knapsacks for OSI value maximization. Hence, the satisfaction optimization problem $P1$ is *NP-Hard*. □

IV. Algorithm Design

It usually takes exponential time to find the optimal solution to our satisfaction optimization problem, which is prohibitively costly and thus the exhaustive search does not perfectly suit our application scenario in which the offloading decision is supposed to be made in almost real time. As a result, we propose a greedy heuristic approach to seek the approximate solution to the problem $P1$, and hope to find the solution in polynomial time to better cater to the rigorous latency requirement of the decision making on task offloading in VEC.

Since RSU is interconnected with the edge nodes using the ultra-fast optical fiber networks, we assume that RSU can learn the status of the edge nodes in real time, e.g., by beacon packets dissemination. RSU can construct two $m \times n$ matrixes $\mathcal{U}$ and $\mathcal{Q}$. $\mathcal{U}$ is the DSI matrix with each element $U_{i,j}$ denoting the DSI value of vehicle v_i towards the edge node s_j. $\mathcal{Q}$ is the CSI matrix with each element $Q_{i,j}$ denoting the CSI value of vehicle v_i towards the edge node s_j. Then the OSI matrix $\mathcal{O}$ can be formed, i.e., $\mathcal{O} = w_d\mathcal{U} + w_c\mathcal{Q}$. The optimization problem $P1$ can be transformed to maximize the sum of m elements in $\mathcal{O}$ in the condition that m elements come from m different rows in $\mathcal{O}$ and the constraint (8) is satisfied.

The proposed greedy heuristic-based algorithm is shown in Alg. 1. At the beginning, the algorithm does some initialization such as constructing matrixes $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{x}$, $\mathcal{U}$, $\mathcal{Q}$, and $\mathcal{O}$. Then, a list l_j for each column j in $\mathcal{Q}$ (i.e., each edge node s_j) is maintained for recording the tasks which s_j is responsible for performing. Then we begin to traverse each row i in $\mathcal{Q}$ that is actually corresponding to each task t_i. The heuristic rule adopted here is to select the maximal element in each row. However, the corresponding element may be invalid, due to the fact that the number of tasks each edge node can support is limited. Therefore, we need to check whether the task can be performed by the edge node, i.e., whether the maximal element can be added into the list (lines 7-10). The element is added to the list and we update the offloading decision (e.g., x_{i,idx_j}) if the element passes the validation. Then, we calculate the sum of OSI values for all the vehicles, and return it as well as the offloading decision matrix $\boldsymbol{x}$.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Greedy heuristic-based algorithm for satisfaction maximimzation

Input: $\mathcal{V}$, $\mathcal{S}$, $\mathcal{T}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, δ, w_d, w_c, $\boldsymbol{h}$, $\boldsymbol{P}$, $\boldsymbol{G}$, σ^2
Output: Optimal objective value, and $\boldsymbol{x}$

Estimate the failure rate matrix $\boldsymbol{\lambda}$, recovery rate matrix $\boldsymbol{\mu}$;
Initialize the matrix of offloading decision $\boldsymbol{x}$;
Construct two matrixes $\mathcal{U}$ and $\mathcal{Q}$;
Construct matrix $\mathcal{O}$ by $\mathcal{O} = w_d\mathcal{U} + w_c\mathcal{Q}$;
Initialize a list l_j for each column j in $\mathcal{O}$;
for each row i in $\mathcal{O}$ **do**
    Find the index of the largest element in row i, let $idx\_j = \arg\max_j\{\mathcal{O}_{i,j}|\mathcal{O}_{i,j} \in \mathcal{O}[i]\}$;
    while $len(l_{idx\_j}) \geq h_{idx\_j}$ **do**
        Find the index of the largest element from the remaining elements in row i, denoted by idx_j;
    end
    $x_{i,idx\_j} = 1$;
    Store $\mathcal{O}_{i,idx\_j}$ to l_{idx_j};
end
$S = 0$;
for each list l_j **do**
    $S+ = sum(l_j)$;
end
return S, $\boldsymbol{x}$;

TABLE I
PARAMETER SETTINGS

Parameter	Value	Parameter	Value
m	[20, 70]	n	10
I_i	[1, 50]	C_i	[40, 70]
h_j	[5,8]	f_j	[10000,12000]
w_d	0.6	w_c	0.4
E_i	[50,90]	ξ_j	(0,1)
$\lambda_{i,j}$	[10,15]	$\mu_{i,j}$	[1,10]
δ	2	$Size$	50
cp	0.2	mp	[300, 1000]

The complexity of the above algorithm can be analyzed as follows. First, it takes time of $O(mn)$ to construct the matrixes including $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{x}$, $\mathcal{U}$, $\mathcal{Q}$, and $\mathcal{O}$. For the worst case, the algorithm always finds the suitable edge node for performing the task at the end when traversing each row in $\mathcal{O}$. In the meanwhile, it will take time of $n \log n$ to find the largest element in each row. Hence, the total time taken to find the appropriate edge node for each task is $O(mn^2 \log n) + O(\sum_{j=1}^{n} h_j) = O(mn^2 \log n)$. As a result, the approximately optimal solution can be found in polynomial time.

V. PERFORMANCE EVALUATION

A. Parameter Settings

In this section, we validate the efficiency of our approach by conducting extensive simulations. In particular, the parameters involved in the simulation are all generated in a random way. The main parameters involved in the simulation are shown in Table I. For instance, the failure rate and the recovery rate in the simulation vary from 5 to 10 and from 1 to 10, respectively.

In the meanwhile, we compare our approach with three benchmarks which are detailed in what follows. *RND*: As the simplest approach, the random approach allocates the tasks to the edge nodes randomly, as long as the constraints are satisfied. We denote this approach by RND. *EN-optimal*: From the perspective of the edge node, each node tends to select the task with the maximal OSI value each time, and they repeat the procedure until the constraint is violated. *GA*: The genetic algorithm (GA) tries to find the optimal task offloading decision iteratively. The running time of GA mainly depends upon the number of iterations in the simulation. Specifically, the parameter settings of GA are also shown in Table I. For instance, the population size is 50 and the number of iterations is 100. The crossover probability and mutation probability are set to 0.2 and 0.02, respectively.

B. Simulation Results and Analysis

We first validate the efficiency of our approach compared to the other three benchmarks and the simulation results are shown in Fig. 1.

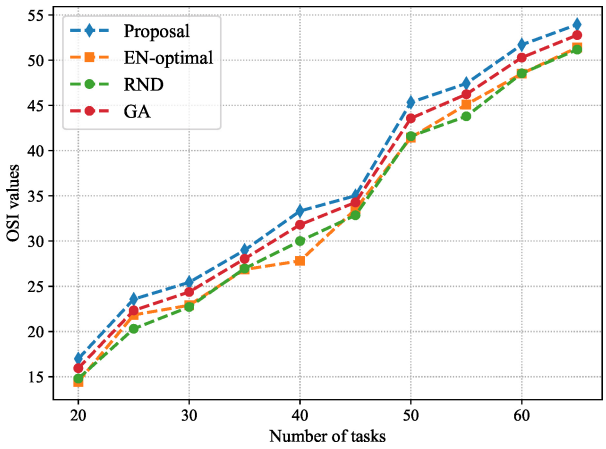

Fig. 1. The performance comparison with different failure rates

First of all, our approach can achieve the best results no matter how the number of tasks varies. Second, as the number of tasks increases, the objective values (i.e., OSI values) also increase for all the approaches. Third, GA can achieve the second best results among the four approaches, and there are no determinate relationships between the random approach and EN-optimal approach. For instance, when the number of tasks is 40, the former is better than the latter. However, when the number of tasks is 55, the latter is better than the former. The main reason is that the random approach fluctuates more frequently than other approaches. Sometimes it may achieve better results and sometimes may not. Generally, our approach is the best among the four approaches in terms of the optimal values.

The algorithm is actually run by RSU, when RSU receives the beacon information from nearby vehicles and learns the

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.

status of the edge nodes. As a result, the time taken to make the offloading decision by RSU is supposed to be almost real time. Such a strict latency requirement helps improve both the QoE of the vehicles and the QoS of the edge nodes. Thus, it is necessary to evaluate the running times of these approaches. The simulation results are shown in Fig. 2. Obviously, apart from GA, the other three approaches can achieve real-time response, i.e., they can make decisions on task offloading in real time. GA usually takes seconds to make the task offloading decisions, and thus GA is not suitable for our application scenario. Comparatively speaking, our approach is the best among the four approaches in terms of both the optimal values and running time.

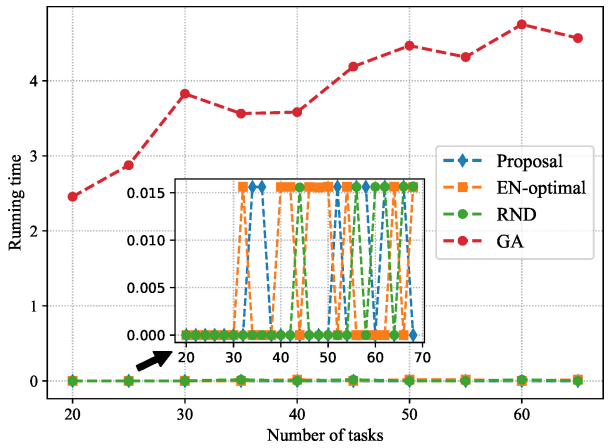

Fig. 2. The performance comparison with different failure rates

VI. Conclusion

In view of the importance of satisfaction of vehicles in stimulating them to pursue better QoE through task offloading and service outsourcing operations, we strive to optimize the satisfaction values for all the vehicles in VEC. Further to this, we have generalized the traditional task offloading model by considering potential failures in VEC. A satisfaction maximization scheme is put forward in the failure-aware VEC system. Due to the difficulty in solving this formulated MKP problem, we adopt a greedy heuristic approach to speed up the searching process, so as to find the near-optimal solution in polynomial time. The simulation results have shown that our approach can outperform other benchmarks.

For future work, we plan to consider more factors when evaluating the satisfaction value of vehicles and design more efficient schemes and strategies to optimize the satisfaction of vehicles in VEC.

Acknowledgement

This work is partially supported by the National Natural Science Foundation of China under Grant Number 62071327. Huaming Wu is the corresponding author.

References

[1] Q. Luo, S. Hu, C. Li, G. Li and W. Shi, "Resource Scheduling in Edge Computing: A Survey," in IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2131–2165, Fourthquarter 2021.

[2] I. Martinez, A. S. Hafid and A. Jarray, "Design, Resource Management, and Evaluation of Fog Computing Systems: A Survey," in IEEE Internet of Things Journal, vol. 8, no. 4, pp. 2494–2516, 15 Feb.15, 2021.

[3] C. Tang, C. Zhu, X. Wei, H. Wu, Q. Li and J. J. P. C. Rodrigues, "Intelligent Resource Allocation for Utility Optimization in RSU-Empowered Vehicular Network," in IEEE Access, vol. 8, pp. 94453–94462, 2020.

[4] S. Wang, J. Li, G. Wu, H. Chen and S. Sun, "Joint Optimization of Task Offloading and Resource Allocation Based on Differential Privacy in Vehicular Edge Computing," in IEEE Transactions on Computational Social Systems, vol. 9, no. 1, pp. 109–119, Feb. 2022.

[5] C. Tang, H. Wu, "Joint optimization of task caching and computation offloading in vehicular edge computing," in Peer-to-Peer Netw. Appl., vol. 15, no. 2, pp. 854–869, 2022.

[6] J. S. Ng, W. Y. B. Lim, Z. Xiong, D. Niyato, C. Leung and C. Miao, "A Double Auction Mechanism for Resource Allocation in Coded Vehicular Edge Computing," in IEEE Transactions on Vehicular Technology, vol. 71, no. 2, pp. 1832-1845, Feb. 2022.

[7] X. Wang, Z. Ning, S. Guo and L. Wang, "Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing," in IEEE Transactions on Mobile Computing, vol. 21, no. 2, pp. 598-611, 1 Feb. 2022.

[8] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," The Journal of Supercomputing, vol. 65, no. 1, pp. 426–444, Jul. 2013.

[9] Y. Sang, J. Cheng, B. Wang, M. Chen, "A three-stage heuristic task scheduling for optimizing the service level agreement satisfaction in device-edge-cloud cooperative computing," PeerJ Comput. Sci., 8: e851, 2022.

[10] B. Wang, J. Cheng, J. Cao, C. Wang, W. Huang, "Integer particle swarm optimization based task scheduling for device-edge-cloud cooperative computing to improve SLA satisfaction," PeerJ Comput. Sci., 8: e893, 2022.

[11] J. Li et al., "Maximizing User Service Satisfaction for Delay-Sensitive IoT Applications in Edge Computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 5, pp. 1199-1212, 1 May 2022.

[12] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," IEEE Transactions on Vehicular Technology, vol. 69, no. 9, pp. 9364–9375, 2020.

[13] C. Tang, S. Xia, Q. Li, W. Chen, and W. Fang, "Resource pooling in vehicular fog computing, " Journal of Cloud Computing, 10, 19, 2021.

[14] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," The Journal of Supercomputing, vol. 65, no. 1, pp. 426–444, Jul. 2013.

[15] J. Yao and N. Ansari, "Fog Resource Provisioning in Reliability-Aware IoT Networks," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 8262–8269, Oct. 2019.

[16] B. Korte and J. Vygen, Combinatorial Optimization: Theory and Algorithms, 1st ed. Springer, 2000.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on January 12,2023 at 13:53:17 UTC from IEEE Xplore. Restrictions apply.