Toward Failure-Aware Energy-Efficient Service Provisioning in Vehicular Fog Computing

Chaogang Tang*, Chunsheng Zhu†, Huaming Wu‡, Lei Ning§, Joel J. P. C. Rodrigues¶‖
*School of Computer Science and Technology, China University of Mining and Technology, 221116, Xuzhou, China
†College of Big Data and Internet, Shenzhen Technology University, 518118, Shenzhen, Guangdong, China
‡ The Center for Applied Mathematics, Tianjin University, 300072, Tianjin, China
§College of Big Data and Internet, Shenzhen Technology University, 518118, Shenzhen, Guangdong, China
¶ College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, China
‖ Instituto de Telecomunicações, Covilhã, Portugal
cg.tang@foxmail.com, chunsheng.tom.zhu@gmail.com, whming@tju.edu.cn, ninglei@sztu.edu.cn, joeljr@ieee.org

***Abstract*—The fast-growing Internet of Things (IoT) have generated a vast number of IoT tasks, and these tasks are usually featured by strict response latency requirements. To cater for the time-sensitive IoT application scenarios, vehicular fog computing (VFC) can be adopted to serve the offloading requests from the IoT devices. However, current works in VFC seldom consider the task execution failures that are actually inevitable owing to limited computing resources in VFC compared to cloud computing. Hence, we strive to enhance the VFC system by incorporating the failures for task execution into our system model, which makes task offloading more general and practical. We formulate our energy consumption optimization as a mixed integer nonlinear programming problem and further put forward an iterative algorithm to solve it. We validate our approach by extensive simulation and the experimental results have proven its advantages in terms of the optimal values.**

***Index Terms*—Vehicular fog computing, service provisioning, failure, energy consumption, task offloading**

I. Introduction

Vehicular fog computing (VFC) has gained tremendous attention in both academia and industry in the past few years. Evolved vehicles are bearing the weight of the enthusiasm for smart transportation, and such an enthusiasm is further propelled by the world's first 5G-capable electric car unveiled in China. Apart from path navigation, driving safety, and infotainment services, smart vehicles can also provision computing resources and such vehicles are termed fog vehicles (FV) in VFC [1]. The fast-growing IoT devices have increasing demands of computing resources, although their own computational capabilities are restricted by the limited physical sizes. Task offloading to the cloud center for execution usually incurs long response delay, which cannot satisfy the rigorous latency requirements of the devices. Fog vehicles can become a perfect substitute, since the computing resources are brought in close vicinity of the IoT devices.

Similar to cloud computing, the virtualization technology is also widely applied to VFC. Virtual machines (VM) created by FVs can efficiently support the scalability and flexibility of computing resource distribution, such that the computational demands from IoT devices can be satisfied on demand without additional overheads [2]. A variety of IoT applications can be performed in VFC in the form of computing tasks. It shall be noted that the amount of computing resources and energy supply in VFC is much less than that in cloud computing. Thus, the elaborate resource allocation is of vital importance to VFC for the sake of resources saving in terms of computing resource and energy consumption. For example, in order to satisfy the quality of service (QoS) requirements of IoT devices, the tasks should be accomplished before their deadline. However, to blindly shorten the response delay does not bring about better quality of experience (QoE) for IoT devices, expect for more energy consumptions. Unfortunately, existing works on VFC seldom notice that. On the other hand, it is an inescapable fact that task execution in VFC is more susceptible to failures [3], owing to its limited computing resources and energy supply. It usually takes time to resume the task execution after the failures occur. Note that we only take into consideration the recoverable failures in this paper such as temporary disconnection and software failures. Each recovery from a failure incurs extra energy consumption and calculation delay. Therefore, it is necessary to incorporate the failure-resisted task offloading into the failure-prone VFC system. To that end, we put forward an energy-aware failure-resisted service provisioning scheme in VFC in this paper. Specifically, the contributions of this paper are threefold, given below:

- Different from the traditional service provisioning which assumes that the offloaded tasks can be executed without failures in VFC, we have considered the failures for task execution in our system model, which makes our task offloading model more general and practical.
- In this paper, we aim to minimize the total energy consumption for all the fog vehicles and model it as a mixed integer nonlinear programming problem. Due to the difficulty in solving it, we put forward an iterative algorithm to solve this problem to obtain the approximately optimal solution. Specifically, we have adopted a heuristic rule to speed up the searching process.
- Extensive simulation is carried out to validate the efficien-

cy of our approach. The simulation results have shown that our approach can achieve a better result in terms of the optimal values.

II. Related Works

Cloud computing residing in the remote core network can provide unlimited computing resources at the expense of long response delay and a fault-tolerated mechanism is required for reducing the failures of task execution [4]. Authors in [5] put forward an efficient QoS-aware and fault-tolerated architecture which incorporates the software-defined vehicular networks to tackle the above issues. Then heuristic algorithms are proposed to solve the formulated problem. Authors in [6] envision an application scenario where the fog nodes can process the tasks by creating VMs using the rented computing resources. Furthermore, they have taken into consideration the reliability of the fog computing system, e.g., measuring it by the number of failures of task execution. The goal is to seek the tradeoff between the system reliability maximization and system costs minimization.

In a VFC scenario where RSU is deployed along the rural highways, the performance of RSU is usually limited by the energy supply, which requires that the task scheduling and resource allocation should be more efficient. For example, how to optimize the energy consumption is crucial, because the task offloading incurs both the computation and communication costs, thus bringing more energy consumptions of RSU. Hence, authors in [7] put forward a strategy for energy consumption optimization which can meet the constraints including the task deadline and resource availability. Authors in [8] strive to optimize the energy efficiency for internet of vehicles. To that end, they propose a non-orthogonal multiple access (NOMA)-based fog computing vehicular network architecture and the corresponding optimization problem. Then the resource optimization problem is divided into two subproblems for solving the problem more efficiently.

The sensing capability has been playing an important role in autonomous driving, but one single autonomous vehicle has limited sensing coverage. To avoid misdetecting the dead zones, a VFC architecture which combines greedy and SVM algorithms is proposed to enhance the sensing capability via cooperation among fog vehicles [9]. They further use the distributed deep learning for trajectory prediction. The simulation shows that the sensing capability can be greatly enhanced in terms of the sensing coverage and accuracy. Authors in [10] try to optimize the response delay and energy consumption by modeling the problem as a multi-objective optimization problem. Then an efficient offloading strategy is used for task execution at FVs, and solved based on differential evolution algorithm.

It is challenging to motivate FVs to contribute the computing resources and guarantee the service availability. A task offloading strategy is proposed to encourage FVs to share the resources considering vehicle mobility, task priority and service availability, and the problem is formulated as Markov decision process [11]. The deep reinforcement learning-based approach is used for solving this problem. The topology of vehicular networks fluctuates a lot due to the mobility of vehicles. The service provisioning in VFC is supposed to consider this factor. Specifically, a joint optimization problem is proposed which tries to optimize both task scheduling and resource allocation while considering the effects of vehicle mobility [12]. Due to the difficulty in solving this non-convex problem, authors divide it to two sub-problems.

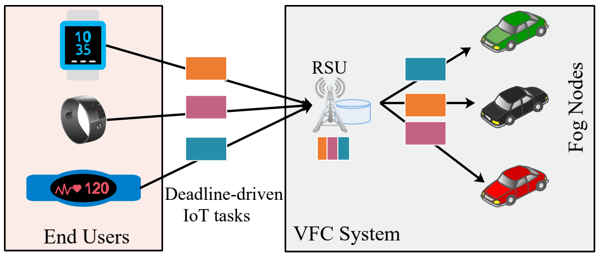

Fig. 1. Failure-aware VFC architecture for deadline-driven IoT tasks

Other works such as [13]–[17] also focus on the performance improvement of VFC systems with regards to response latency, energy consumption, privacy and so on. However, due to the limitation of space, we do not detail them any longer. Different from the above works, we consider a more general and practical task offloading model in VFC, and try to jointly optimize the task offloading and resource allocation in VFC.

III. System Model

The failure-aware VFC architecture for deadline-driven IoT tasks is shown in Fig. 1, which consists of three entities, i.e., the IoT devices, FVs and RSU. RSU acting as the fog server can serve the FVs within its wireless coverage. For example, RSU can take charge of task assignment for them. There are M IoT tasks, indexed by $\mathcal{M} = \{1, ..., M\}$, and N FVs, indexed by $\mathcal{N} = \{1, ..., N\}$, respectively, in the proposed system. The task $i (\in \mathcal{M})$ can be represented by the tuple (I_i, O_i, D_i), where I_i is the size of the task-input data, O_i is the amount of computing resources needed to accomplish task i, and D_i is the deadline for task i. The FV $j (\in \mathcal{N})$ can be represented by the tuple $(f_{j,max}, h_j, g_j,)$, where $f_{j,max}$ is the maximal processing frequencies of FV j, h_j is the maximal number of VMs that FV j can simultaneously support, and g_j is the dwelling time of FV j. The dwelling time of FV j within the coverage of RSU can be easily estimated [18]. Assume that the FVs information such as the VM information and the dwelling time is known to RSU. According to these information, the M tasks can be assigned to different FVs for execution. Define x_{ij} as a binary decision variable to denote whether task i is performed by FV j. Particularly, $x_{ij} = 1$, if task i is performed by FV j; and $x_{ij} = 0$, otherwise.

A. Networking Model

Task offloading to VFC instead of the cloud center for execution aims to better cater for the strict latency requirements of IoT devices. The corresponding transmission delay of the computation tasks needs to be calculated as follows. For instance, the transmission delay of task i offloaded to FV j, denoted by $d_{i,j}^{trs}$, is given as $d_{i,j}^{trs} = I_i/r_{i,j}$, where $r_{i,j}$ is the transmission rate of the wireless channel between device i and vehicle j, and can be calculated as:

$$r_{i,j} = \mathcal{B}\log_2(1+\frac{P_i\mathbb{H}_{ij}}{\sigma^2}) \tag{1}$$

where P_i denotes the transmission power of device i, $\mathbb{H}_{ij}$ is the channel gain between device i and vehicle j, $\mathcal{B}$ is the bandwidth for the wireless channel and σ^2 is the noise power.

B. Failure-aware Calculation Model

When the task i is allocated to FV j for execution, j will create a VM for i by designating computing and storage resources. It shall be noted that the execution may fail when the corresponding VM processes task i. Then, the calculation delay denoted by $d_{i,j}^{clt}$ will be discussed based on the following two cases. If the task execution is successful, the calculation delay, denoted by $d_{i,j}^{s}$, can be expressed as:

$$d_{i,j}^{s} = \zeta_j + \frac{O_i}{f_j} \tag{2}$$

where ζ_j that is independent of any offloaded tasks denotes the initialization time of the VM; f_j is the processing frequency of FV j. On the other hand, if the task execution fails, the VM will restart the task after some recovery time, e.g., by using checkpointing and rollback/roll-forward technologies [3]. Note that the recovery time here is different from the initialization time of the VM (i.e., ζ_j), in the sense that the principles and technologies behind them are not the same. Following the works [3], [6], we also assume that the failures of task i at FV j follow a Poisson process with the failure rate λ_j. Then, define $\mathcal{N}(t)$ as the number of failures during the time $(0, t]$. Hence, the probability that there are k failures within the time interval lasting $d_{i,j}^{s}$ seconds can be expressed as:

$$\mathrm{P}\{\mathcal{N}(d_{i,j}^{s}) = k\} = \frac{(\lambda_j d_{i,j}^{s})^k}{k!}e^{-\lambda_j d_{i,j}^{s}} \tag{3}$$

and $\mathbb{E}[\mathcal{N}(d_{i,j}^{s})] = \lambda_j d_{i,j}^{s}$. Define $\mathcal{R}_k(d_{i,j}^{s})$ as the recovery time of the kth failure for task i at FV j, and we further assume it follows an exponential distribution with the recovery rate μ_j. Meanwhile, the total $\mathcal{N}(d_{i,j}^{s})$ failures for task i at FV j are assumed to be independent of each other. Then, the recovery times are independent and identically distributed (i.i.d.) random variables, and the total recovery time $\mathcal{R}(d_{i,j}^{s}) = \sum_{k=1}^{\mathcal{N}(d_{i,j}^{s})}\mathcal{R}_k(d_{i,j}^{s})$ follows Gamma distribution, i.e., $\mathcal{R}(d_{i,j}^{s}) \sim \Gamma(\lambda_j d_{i,j}^{s}, \mu_j)$. Hence, the mean of the total recovery time is $\lambda_j d_{i,j}^{s}/\mu_j$. The failure-aware calculation delay, which consists of the norm calculation delay and the recovery time, can be expressed as:

$$d_{i,j}^{clt} = d_{i,j}^{s} + \frac{\lambda_j d_{i,j}^{s}}{\mu_j} = (1+\frac{\lambda_j}{\mu_j})(\zeta_j + \frac{O_i}{f_j}) \tag{4}$$

The response delay $d_{i,j}^{rsp}$ for the task i offloaded to FV j can be expressed as: $d_{i,j}^{rsp} = d_{i,j}^{trs} + d_{i,j}^{clt}$. Here, we omit the returning delay owing to the fact that the size of execute results is negligible compared to the task-input data size.

C. Energy Consumption Model

When task i is performed by the FV j, the energy consumption of FV j needs to be calculated depending upon the following two cases. If there are no failures during the task processing, the energy consumption $e_{i,j}^{s}$ can be calculated as:

$$e_{i,j}^{s} = \chi_j + \vartheta\varsigma O_i f_j^2 \tag{5}$$

where χ_j is the static power consumption incurred by VM initialization, regardless of the workload of tasks, ϑ is the effective switched capacitance coefficient, and ς is the number of cycles needed to perform one task-input bit at j. Obviously, the larger the processing frequency, the more the energy consumptions. As a result, it is advised to reduce the extra overheads on energy consumptions by adjusting the processing frequency f_i, as long as the deadline of the IoT tasks is satisfied. On the other hand, if there are failures for task execution, the extra energy consumptions can be brought about during the task processing recovery. Assume that the energy consumption for each failure recovery is fixed, denoted by ξ_j. Since the failures of task i at FV j follow a Poisson process with the failure rate λ_j and $\mathbb{E}(\mathcal{N}(d_{i,j}^{s})) = \lambda_j d_{i,j}^{s}$, the extra energy consumption for FV j is $\lambda_j d_{i,j}^{s}\xi_j$. Thus, the average total energy consumptions for task i performing at FV j can be expressed as:

$$\begin{aligned} e_{i,j} &= \chi_j + \vartheta\varsigma O_i f_j^2 + \lambda_j d_{i,j}^{s}\xi_j \\ &= \vartheta\varsigma O_i f_j^2 + \lambda_j\xi_j\frac{O_i}{f_j} + \lambda_j\xi_j\zeta_j + \chi_j \end{aligned} \tag{6}$$

D. Problem Formulation

The goal in this paper is to minimize the energy consumptions for all the fog vehicles in the failure-resisted VFC system. In particular, we define $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f}) = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}e_{i,j}$ as our optimization objective, where $\boldsymbol{x}$ is the $M \times N$ matrix of which the element x_{ij} is the task offloading decision for the task i at FV j, and $\boldsymbol{f} = (f_1, ..., f_N)$ is a vector to denote the processing frequencies of all the FVs allocated to the tasks offloaded from IoT devices. Based on these descriptions, the optimization problem in this paper is formulated as below:

$$(P1) \quad \min_{\boldsymbol{x},\boldsymbol{f}} \mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$$

$$\text{s.t.} \quad d_{i,j}^{rsp} \le D_i \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \tag{7}$$

$$d_{i,j}^{rsp} \le g_j \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \tag{8}$$

$$\sum_{i=1}^{M} x_{ij} \le h_j \quad \forall j \in \mathcal{N} \tag{9}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad \forall i \in \mathcal{M} \tag{10}$$

$$f_j \le f_{j,max} \quad \forall j \in \mathcal{N} \tag{11}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \tag{12}$$

where the constraint (7) guarantees that the total response delay $d_{i,j}^{rsp}$ should not exceed the deadline of the task i, and on the other hand the constraint (8) guarantees that the total response delay $d_{i,j}^{rsp}$ should not exceed the dwelling time of FV j, either. Since an arbitrary FV j can support maximal h_j VMs at the same time, the total number of tasks offloaded to the FV j shouldn't exceed h_j, which can be guaranteed by the constraint (9). In the meanwhile, an arbitrary task can be offloaded to at most one FV for execution, which can be guaranteed by the constraint (10). Constraints (11) and (12) ensure that the two variables should not violate their own constraints.

IV. Algorithm Design

To obtain the optimal solution to $P1$ requires exponential time, e.g., by the exhaustive search among the potential N^M solutions. Such a prohibitively costly way cannot perfectly suit our deadline-driven IoT situation. Furthermore, $P1$ is a mixed integer nonlinear programming which is pretty difficult to solve, due to the coexistence of discrete variable (i.e., $\boldsymbol{x}$) and continuous variable (i.e., $\boldsymbol{f}$) as well as the nonlinearity of the optimization objective. Accordingly, we put forward an iterative algorithm to solve this problem. In particular, we try to search the optimal decision pair $(\boldsymbol{x}^*, \boldsymbol{f}^*)$ in an iterative fashion.

The proposed algorithm is shown in Alg. 1. The algorithm works as follows. At the beginning, IAFE will do some initialization on the required parameters including the failure rate vector $\boldsymbol{\lambda}$ and recovery rate vector $\boldsymbol{\mu}$, where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_N)$ and $\boldsymbol{\mu} = (\mu_1, ..., \mu_N)$. Such parameters can be easily estimated based on the statistical histories. One IoT task, say i, is randomly selected, and then change its offloading decision $\boldsymbol{x}_i$ to $\tilde{\boldsymbol{x}}_i$ by randomly altering its offloading decision (e.g., $x_{ik} = 1$ to $x_{ip} = 1$). Then, we need to check whether $\tilde{\boldsymbol{x}}_i$ is feasible. For instance, if there are no violation of the constraints such as the inequations (7)–(9), $\tilde{\boldsymbol{x}}_i$ is valid. The new task offloading decision $\tilde{\boldsymbol{x}}$ is formed based on $\tilde{\boldsymbol{x}}_i$ (line 9). Given the offloading decision $\tilde{\boldsymbol{x}}$, the algorithm calculates the optimal resource allocation $\tilde{\boldsymbol{f}}$ by minimizing the optimization objective $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$. Since there is only one task (i.e., i) changing its offloading decision, $\tilde{\boldsymbol{f}}_{-i} = \boldsymbol{f}_{-i}$, i.e., the energy consumptions of other tasks except i is the same. We only need to update the energy consumption of task i. In the meanwhile, the objective value $\mathcal{O}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}}) = \mathcal{O}(\boldsymbol{x}, \boldsymbol{f}) + \triangle e_i$, where $\triangle e_i = e_{ip} - e_{ik}$. Therefore, we can speed up the search process to a great extent.

Iteration-based approach for this mixed discrete-continuous optimization may be trapped in a local optimum. To tackle this issue, we have adopted a control factor θ to help escape the local optimum. Particularly, the exploration continues with the probability of θ and stops with the probability of $1-\theta$, when $\mathcal{O}$ is very close to $\tilde{\mathcal{O}}$. The process repeats until the stopping criterion is satisfied, e.g., the maximal number of steps has been reached.

Lemma 1: Given the offloading decision $\boldsymbol{x}$ for all the IoT tasks $\mathcal{M}$, there exists an optimal resource allocation scheme $\boldsymbol{f}$ which can minimize the optimization objective $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$.

Algorithm 1: Iterative Algorithm for Failure Resisted Energy Optimization (IAFE)

```
Input: $\mathcal{M}$, $\mathcal{N}$, $\boldsymbol{\zeta}$, $\boldsymbol{\chi}$, $\vartheta$, $\varsigma$, $\boldsymbol{\xi}$, $\mathcal{B}$, $\boldsymbol{P}$, $\mathbb{H}$, $\sigma^2$
Output: Optimal decision pair $(\boldsymbol{x}^*, \boldsymbol{f}^*)$
1  Predict the failure rate vector $\boldsymbol{\lambda}$, recovery rate vector $\boldsymbol{\mu}$;
2  Construct an initial task offloading decision $\boldsymbol{x}$;
3  Calculate the optimal resource allocation $\boldsymbol{f}$ by minimizing $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$;
4  Record the optimal objective value $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$;
5  repeat
6      Randomly pick an offloading decision $\boldsymbol{x}_i$ of task $i$;
7      Change $\boldsymbol{x}_i$ to $\tilde{\boldsymbol{x}}_i$ by altering the offloading decision of task $i$ randomly;
8      if $\tilde{\boldsymbol{x}}_i$ is valid then
9          $\tilde{\boldsymbol{x}} \leftarrow (\boldsymbol{x}_{-i}, \tilde{\boldsymbol{x}}_i)$;
10         Obtain $\tilde{\boldsymbol{f}}$ by minimizing $\mathcal{O}(\tilde{\boldsymbol{x}}, \boldsymbol{f})$;
11         Record $\tilde{\mathcal{O}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}})$;
12         if $|\tilde{\mathcal{O}} - \mathcal{O}| \leq \delta$ then
13             Generate a random value $\theta (\in (0,1))$;
14             With probability $\theta$, update: $\mathcal{O} \leftarrow \tilde{\mathcal{O}}$
               $(\boldsymbol{x}, \boldsymbol{f}) \leftarrow (\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}})$ ;
15         else
16             $\mathcal{O} \leftarrow \tilde{\mathcal{O}}$ ;
17             $(\boldsymbol{x}, \boldsymbol{f}) \leftarrow (\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{f}})$;
18         end
19     end
20 until stopping criterion satisfied;
```

Proof: First, the minimal value of $e_{i,j}$ exists, i.e.,

$$
\begin{aligned}
e_{i,j} &= \vartheta\varsigma O_i f_j^2 + \lambda_j \xi_j \frac{O_i}{f_j} + \lambda_j \xi_j \zeta_j + \chi_j \\
&\overset{\ddagger}{\geq} (\frac{1}{4}\vartheta\varsigma O_i^3 \lambda_j^2 \xi_j^2)^{1/3} + \lambda_j \xi_j \zeta_j + \chi_j
\end{aligned}
$$

where the equal sign of $\ddagger$ holds, if and only if $\vartheta\varsigma O_i f_j^2 = \frac{1}{2}\lambda_j \xi_j \frac{O_i}{f_j}$, i.e., $f_{j,1} = (\frac{1}{2}\lambda_j \xi_j / \vartheta\varsigma)^{1/3}$, which is independent of the task-input data O_i. Second, there are other two critical values we need to check. One is the energy consumption of $e_{i,j}$ for the response latency equal to the deadline D_i; the other is the energy consumption for the response latency equal to d-welling time of FV j g_j. Let $d_{min} = \min\{D_i, g_j\}$, and we can obtain $\tilde{f}_{j,2} = (\lambda_j + \mu_j) O_i / (\mu_j (d_{min} - d_{i,j}^{trs}) - \zeta_j (\lambda_j + \mu_j))$, and let $f_{j,2} = \max\{\tilde{f}_{j,2} | j \in \mathcal{N}\}$. As a result, given the decision $\boldsymbol{x}$, the minimal value of $\mathcal{O}(\boldsymbol{x}, \boldsymbol{f})$ exists by checking the three potential processing frequencies $\{f_{j,1}, f_{j,2}, f_{j,max}\}$, respectively. $\square$

V. Simulation Evaluation

In this section, we validate the efficiency of our approach in terms of the optimal values by conducting extensive simulation. Specifically, the involved parameters are initialized in a random fashion. For instance, the number of fog vehicles in VFC is 10, and the number of IoT tasks varies from 20 to 70.

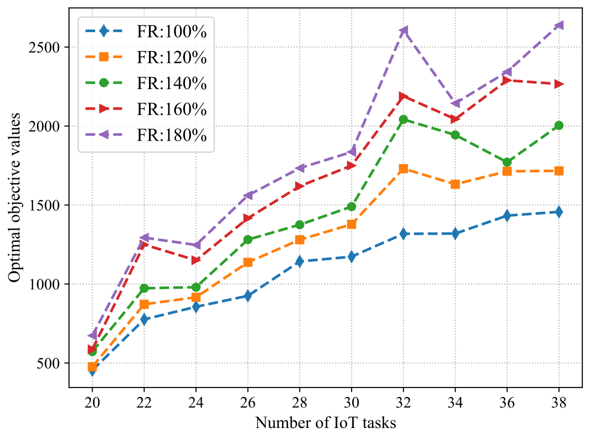

Fig. 2. The performance comparison with different failure rates

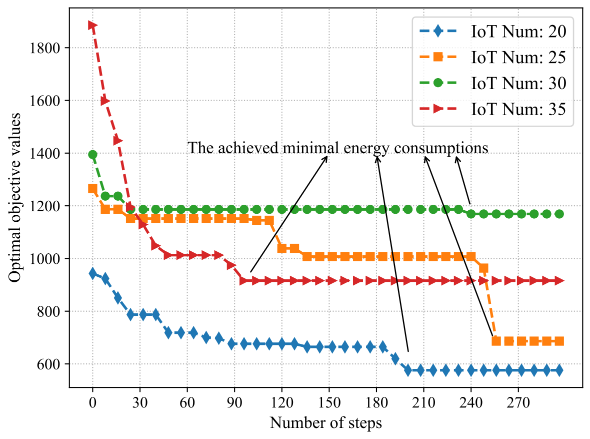

Fig. 3. The evaluation of maximal steps before achieving the optimal values

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 IAFE with three benchmarks which are mainly dependent upon the heuristic rules. *PF-optimal*: Different FVs have different maximal allowed processing frequencies (i.e., $f_{j,max}$). This approach always allocates the IoT tasks to the unoccupied FV with the maximal processing frequency. *VM-optimal*: Different FVs have different maximal number of VMs, and this approach always allocates the IoT tasks to the unoccupied FV with the maximal number of VMs. *Failure-optimal*: Different FVs have different failure rates, and this approach always allocates the IoT tasks to the unoccupied FV with minimal failure rate.

We first investigate the effect of the failure rates upon our approach. Particularly, the performance comparison with different failure rates is shown in Fig. 2, where the x-coordinate denotes the number of IoT tasks and the y-coordinate denotes the optimal energy consumptions for five cases. "FR:100%" denotes the original case where the failure rate for each FV is set randomly. The next four cases, denoted by "FR:120%", "FR:140%", "FR:160%", and "FR:180%", mean that each case improves the failure rate by 20%, 40%, 60% and 80%, respectively compared to the original case. First of all, higher failure rates lead to larger energy consumptions which can be easily observed from the figure. For example, we can randomly pick two from five cases, say "FR:120%" and "FR:160%", and the minimal optimal values for "FR:160%" are always larger than the optimal values for "FR:120%", no matter how the number of IoT tasks varies. Second, the optimal energy consumptions increase as the number of IoT tasks increases, which can be observed from all the five cases. More IoT tasks mean more energy consumptions taken to accomplish them, no matter which FVs are assigned. Note that the task information (e.g., I_i, O_i and D_i) is generated randomly, it is understandable that the performance may fluctuate a lot, e.g., when the number of IoT tasks is equal to 32 compared to the number of tasks equal to 34.

In the next, we investigate the effects of iteration steps upon the approach. It is well known that the iterative algorithms are time-consuming. Due to the difficulty in solving our mixed integer nonlinear programming problem, we adopt the iterative algorithm to solve it and we need to check whether it can well suit our time-driven IoT application scenario. The simulation results for the evaluation of maximal steps before achieving the optimal values are shown in Fig. 3, where the x-coordinate denotes the number of iteration steps and the y-coordinate denotes the optimal energy consumptions. We study the four cases in the simulation, i.e., the number of IoT tasks are 20, 25, 30 and 35, respectively. In the simulation, we set the maximal number of iteration steps to 500. From the figure, we can see that the optimal values (i.e., energy consumptions) can achieve the best within 300 iteration steps. For instance, for the case with the number of IoT tasks equal to 20, the optimal value of energy consumptions does not decrease after the number of iteration steps reaches about 200; for the case with the number of IoT tasks equal to 35, the optimal value of energy consumptions does not decrease after the number of iteration steps reaches 90. Similarly, for the other two cases, the optimal values can also be obtained when the number of iteration steps are equal to 240 and 250, respectively. Note that in the simulation we evaluate the number of iteration steps instead of the running time, since the latter can also be greatly influenced by the hardware of computing nodes apart from the algorithms. Generally, it is more objective to evaluate the performance of approach using the former in the simulation.

Last, we evaluate our approach with three benchmarks in terms of the optimal values of energy consumptions. The simulation results are shown in Fig. 4. Several conclusions can be drawn from the figure. First, our approach can achieve the best performance among the four approaches no matter how the number of IoT tasks varies. Second, there are actually no comparable relationships among three benchmarks. For instance, the performance of *PF-optimal* is the best among the three approaches when the number of IoT tasks equals about 32, while it has the worst performance compared to the other two approaches when the number of IoT tasks equals 40.

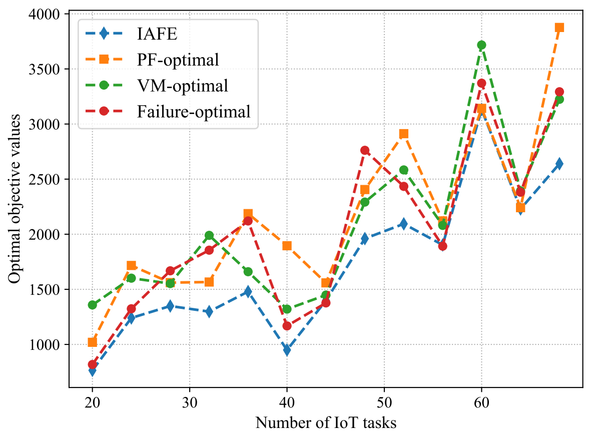

Fig. 4. The performance comparison of four approaches

The similar situations can also be observed for other number of IoT tasks in the figure. Third, the four approaches fluctuate a lot, since the task information is generated randomly, which is still acceptable in our opinion. To sum up, our approach is the best compared to the three benchmarks in terms of the optimal values.

VI. Conclusion

The explosive growth in the number of IoT tasks has posed great pressure on the back-haul links when they are offloaded to the cloud center for execution. The resulting long response delay may degrade both QoS and QoE of IoT devices to some extent. In this context, VFC can become a tempting choice for executing the IoT tasks, since FVs in VFC can provision computing resources in close vicinity of the IoT devices and thus better cater for the strict latency requirement of IoT tasks. In this paper, we improve the performance of the VFC system by incorporating the failures for task execution. Particularly, we consider to optimize the energy consumptions of all the FVs and formulate it as a mixed integer nonlinear programming problem and further solve it by an an iterative algorithm. By conducting extensive simulation, we have proven that the proposed strategy in this paper can better optimize the energy consumptions compared to other benchmarks.

Acknowledgement

This work is supported by the Scientic Research Capacity Improvement Project from Guangdong Province (No. 2021ZDJS109), and the National Natural Science Foundation of China under Grant Number 62071327. This work is also partially supported by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/50008/2020, and by Brazilian National Council for Research and Development (CNPq) via Grant No. 313036/2020-9. Chunsheng Zhu is the corresponding author.

References

[1] 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.
[2] M. M. Sayed, M. S. Kashkoush and M. Azab, “Towards Resilient Adaptive Vehicular Fog Computing,” 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2020, pp. 0681-0685.
[3] 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.
[4] B. Senapati, P. Khilar, R. Swain, “Composite fault diagnosis methodology for urban vehicular ad hoc network,” Veh. Commun. , no. 29, pp. 100337, 2021.
[5] S.A. Syed, et al, “QoS Aware and Fault Tolerance Based Software-Defined Vehicular Networks Using Cloud-Fog Computing, ” Sensors, vol. 22, no. 1, pp. 401, 2022.
[6] 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.
[7] S. Vemireddy, R. R. Rout, “Fuzzy Reinforcement Learning for energy efficient task offloading in Vehicular Fog Computing,” Comput. Networks, vol. 199, pp. 108463, 2021.
[8] Y. Liu, H. Zhang, K. Long, H. Zhou and V. C. M. Leung, “Fog Computing Vehicular Network Resource Management Based on Chemical Reaction Optimization,” in IEEE Transactions on Vehicular Technology, vol. 70, no. 2, pp. 1770–1781, Feb. 2021.
[9] H. Du, S. Leng, F. Wu, X. Chen and S. Mao, “A New Vehicular Fog Computing Architecture for Cooperative Sensing of Autonomous Driving,” IEEE Access, vol. 8, pp. 10997–11006, 2020.
[10] M. Hussain, M. Alam, M. M. Sufyan Beg, N. Akhtar, “Towards minimizing delay and energy consumption in vehicular fog computing (VFC),” J. Intell. Fuzzy Syst., vol. 38, no. 5, pp. 6549–6560, 2020.
[11] J. Shi, J. Du, J. Wang, J. Wang and J. Yuan, “Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning,” IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.
[12] X. Wu, S. Zhao, R. Zhang and L. Yang, “Mobility Prediction-Based Joint Task Assignment and Resource Allocation in Vehicular Fog Computing,” 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020, pp. 1-6.
[13] A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei and R. M. Mehmood, “Mobility Aware Blockchain Enabled Offloading and Scheduling in Vehicular Fog Cloud Computing,” in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4212–4223, July 2021.
[14] Y. Wu, J. Wu, L. Chen, G. Zhou and J. Yan, “Fog Computing Model and Efficient Algorithms for Directional Vehicle Mobility in Vehicular Network,” in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 5, pp. 2599–2614, May 2021.
[15] M. T. Hossain and R. E. De Grande, “Cloudlet Dwell Time Model and Resource Availability for Vehicular Fog Computing,” 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2021, pp. 1–8.
[16] S. Lee and S. Lee, “Resource Allocation for Vehicular Fog Computing Using Reinforcement Learning Combined With Heuristic Information,” in IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10450–10464, Oct. 2020.
[17] C. Lin, G. Han, X. Qi, M. Guizani and L. Shu, “A Distributed Mobile Fog Computing Scheme for Mobile Delay-Sensitive Applications in SDN-Enabled Vehicular Networks,” in IEEE Transactions on Vehicular Technology, vol. 69, no. 5, pp. 5481–5493, May 2020.
[18] C. Tang, S. Xia, Q. Li, W. Chen, and W. Fang, “Resource pooling in vehicular fog computing, ” Journal of Cloud Computing, 10, 19, 2021.