Reliable Computation Offloading for Edge-Computing-Enabled Software-Defined IoV

Xiangwang Hou, Member, IEEE, Zhiyuan Ren, Member, IEEE, Jingjing Wang, Member, IEEE, Wenchi Cheng, Senior Member, IEEE, Yong Ren, Senior Member, IEEE, Kwang-Cheng Chen, Fellow, IEEE, and Hailin Zhang, Member, IEEE

Abstract—Internet of Vehicles (IoV) has drawn great interest in recent years. Various IoV applications have emerged for improving the safety, efficiency, and comfort on the road. Cloud computing constitutes a popular technique for supporting delay-tolerant entertainment applications. However, for advanced latency-sensitive applications (e.g., auto/assisted driving and emergency failure management), cloud computing may result in excessive delay. Edge computing, which extends computing and storage capabilities to the edge of the network, emerges as an attractive technology. Therefore, to support these computationally intensive and latency-sensitive applications in IoVs, in this article, we introduce both partial computation offloading and reliable task allocation with the reprocessing mechanism to EC-SDIoV. Since the optimization problem is nonconvex and NP-hard, a heuristic algorithm, fault-tolerant particle swarm optimization, is designed for maximizing the reliability (FPSO-MR) with latency constraints. Performance evaluation results validate that the proposed scheme is indeed capable of reducing the latency as well as improving the reliability of the EC-SDIoV.

Index Terms—Computation offloading, edge computing, Internet of Vehicles (IoV), reliability, software-defined network (SDN).

I. INTRODUCTION

INTERNET of Vehicles (IoV) [1]–[3], which becomes the cornerstone of the future intelligent transportation system (ITS) [4], integrates information processing as well as artificial intelligence-aided wireless networking technologies. By exchanging information with the vehicles and roadside-units (RSUs), transportation efficiency, accident rate, and energy consumption can be improved [5], [6].

To achieve this, various vehicular networking technologies, such as vehicle-to-infrastructure (V2I) communication, inter-roadside communications, and vehicle-to-vehicle (V2V) communications, constitute a mobile communication network of wide geographical range [7], [8], as shown in Fig. 1. Moreover, the rapid technical progress on vehicles, including computing units, electronic processing units, various sensors, radio transceivers, etc., and the improvement of RSUs’ computing, communications, and storage capabilities, makes the brand new technological frontier of IoT realizable.

The applications for IoV typically aim at three aspects: 1) safety; 2) efficiency; and 3) entertainment [9]. However, due to the limited computing and energy resources, mobile vehicles tend to offload their computation-intensive tasks generated by numerous IoV applications to other processing nodes [10], [11]. For latency- and reliability-insensitive entertainment applications, the traditional cloud-based working manner is suitable. Through offloading computation-intensive tasks to the remote cloud center, and then achieving the results of the offloaded task from the cloud, the abilities of vehicles can be greatly improved [12]. With respect to the safety and efficiency, when tasks are intensely sensitive to latency and reliability, such as emergency obstacle avoidance, path planning, road status recognition, etc., the cloud-based approach is inappropriate [13] because cloud centers are generally located far away from vehicles and resulting in high networking latency through the core network and backbone networks. To respond to these challenges mentioned above, edge computing, which provides abundant resources closer to
end users, has been introduced to IoV networks [14]–[17], as a complementary approach of cloud computing to enhance the capability of IoV.

Facing with intensive computation requests and strict latency constraints for IoV applications, we propose to utilize the available resources of nearby RSUs and vehicles to facilitate edge computing functionalities. Thus, the application-specific computational tasks can be decomposed into several subtasks, and these subtasks can be executed in parallel across multiple processing nodes. However, given the complexity of heterogeneous edge computing nodes (i.e., RSUs, mobile vehicles, etc.), efficient orchestration becomes a new technological challenge. Benefiting from the software-defined network (SDN) architecture [18]–[21], SDN controller, which has global information in an SDN control domain, is considered to orchestrate the edge computing resources [22], [23]. Furthermore, we propose an edge-computing-enabled software-defined IoV (EC-SDIoV) to enhance the capability of IoV for satisfying latency-sensitive services. Moreover, to orchestrate the computing resources satisfying various services’ requirements well, computation offloading emerges as the key challenge. Most of the literature on computation offloading focused on latency and energy consumption [24]–[31], but in the context of complex and dynamic IoV with strict latency requirements, reliability performance needs to be considered, which is closely related with the security of IoV. However, to the best of our knowledge, there is little work on reliable computation offloading in IoV. Therefore, in this article, to achieve reliable computation offloading for latency-sensitive services in IoV, we propose a reliable computation offloading scheme for EC-SDIoV, which jointly considers the partial offloading, reliability-oriented task allocation, and reprocessing mechanism. Since the optimization problem formulated is NP-hard, a heuristic algorithm with low complexity, namely, the fault-tolerant particle swarm optimization algorithm for maximizing reliability (MPSO-MR) is designed. In summary, the main contributions of this article are summarized as follows.

1) We propose the EC-SDIoV architecture to facilitate mobile vehicles and fixed roadside infrastructures as edge computing nodes for providing low-latency computing services cooperatively via SDN.

2) To the best of our knowledge, it is the first time that we focus on maximizing the reliability performance of computation offloading. Moreover, the proposed reliable computation offloading scheme jointly considers the partial offloading, reliability-oriented task allocation, and reprocessing mechanism.

3) To solve the formulated nonconvex and NP-hard optimization problem, a heuristic algorithm with low complexity, namely, the fault-tolerant particle swarm optimization algorithm is designed for maximizing the reliability (FPSO-MR) with latency constraints.

The remainder of this article is organized as follows. The related work is discussed in Section II. In Section III, the demonstration about the EC-SDIoV architecture and reliability assessment is presented. Section IV presents the theoretical modeling of the reliable computation offloading and optimization algorithm. The simulation results and analysis are summarized in Section V, followed by our conclusions in Section VI.

II. RELATED WORKS

Existing computation offloading researches generally pay attention to the optimization of latency and energy. Specifically, Chang et al. [24] investigated the energy-efficient computation offloading and optimized the offloading strategy to minimize the energy consumption with latency constraint. Le et al. [25] designed the optimization algorithm for the allocation of radio resources and computation resources to minimize execution delay, including transmission time and computation time. Moreover, Zhao et al. [26] researched the computation offloading problem in the multiuser edge computing system and modeled the energy consumption of task execution on local devices and MEC servers. Wang et al. [28] designed a three-layer traffic system to minimize the average response latency. Mao et al. [30] constructed the energy consumption minimization problem by considering the application buffer into account. Wang et al. [31] proposed an energy-optimal computation offloading scheme, relying on the maximal latency constraints. While reducing latency and saving energy consumption serve two basic characteristics of computational offloading, reliability emerges as an important indicator in IoV. In the real-world scenario, the outages of the communication link and processing node are inevitable. The consequence could be grievous for latency and reliability sensitive applications like autonomous driving. However, there exists very limited research in computation offloading concerning reliability. In the early stage, Liu and Zhang [32] considered the tradeoff between latency and reliability, and optimized edge node selection, the order of the offloading and task allocation to strike a balance between latency and reliability. However, they only considered the impact of transmission failure on reliability, but ignored the impact of computation failure, which may cause a large deviation from actual situations. Therefore, we conceived a novel reliability guarantee mechanism that integrates reliability-based computation
offloading and task allocation, both considering the reliability of the transmission link and computation node. Furthermore, the reprocessing strategy is introduced to prevent possible outages and failure cases, which enhances the fault-tolerant capability further.

III. EDGE-COMPUTING-ENABLED SOFTWARE-DEFINED IoV ARCHITECTURE AND RELIABILITY ASSESSMENT

A. Edge-Computing-Enabled Software-Defined IoV Architecture

For supporting advanced IoV applications well, we propose holistic edge computing composed of road infrastructures and vehicles to achieve latency-sensitive services in a cooperative manner. Moreover, to facilitate these heterogeneous edge computing nodes well, SDN is introduced as an orchestrator, and thus the EC-SDIoV architecture is constructed. The EC-SDIoV architecture, as shown in Fig. 2, is composed of two different planes, i.e., data plane and control plane. In the vehicular network, each network node is that one entity bears two independent modules (i.e., one for data plane, and the other for control plane) through software-defined and virtualization technologies [21], [33].

1) Mobile-edge computing nodes are mainly those vehicles near the vehicle \( s \), as shown in Fig. 2. Because of more advanced processing units and onboard computing equipment, the computing capabilities associated with vehicles are valuable computing resources. The mobile vehicles as mobile-edge nodes are controlled by the SDN controller for better resource orchestration. The users, who lease the spare communication and computing resources of their vehicles to the edge computing network, could get a certain benefit as edge computing providers [35]. Furthermore, in the future, the special server vehicle could be deployed in hotspot areas dynamically for processing computation tasks offloaded by nearby vehicles or smart devices [36].

2) Fixed edge computing nodes mainly include RSUs, etc. As a class of RSUs, the access RSU, which connects a cluster of edge nodes and SDN controller through broadband connections, could receive the service request from vehicles or other edge nodes and execute decisions of the SDN controller. Note that the access RSUs do not require special hardware, and just RSUs that achieve the network access of the vehicle (who decides to offload its task).

Control Plane and SDN Controller: The physical communication channel of the control plane is independent of the physical communication channel of the data plane, which composed of OpenFlow-based SDN controllers and network nodes. SDN controller keeps the global status information updating continuously,\(^1\) such as channel state information (CSI), match

\[^1\text{There are numerous effective approaches to reduce the overhead of collecting global information, such as decreasing the update frequency of state consistency in multi-SDN control domains [37] and making vehicles to be controlled indirectly via RSU for reducing the changes of network topology [38].}\]
fields, priority, flow tables, resources utilization states, service requirements, etc. [33]. When a network node makes a service request, in the control plane, the SDN controller will create an optimal data forwarding and allocation scheme and broadcast to network nodes at the edge. After that, in the data plane, the traffic data will be transmitted according to the given networking mechanism.

In the control plane, the work methods of mobile-edge computing nodes and fixed edge computing nodes are different.

1) **Fixed edge computing nodes** are composed of fixed edge nodes, i.e., RSUs, edge servers, etc. These nodes are connected with the SDN controller directly, and exchange the state information and service requirements with the controller continuously.

2) **Mobile-edge computing nodes** are composed of vehicles nearby the service requester. Considering the mobility and uncertainty of mobile-edge computing nodes, in order to avoid excessive latency and worsening reliability caused by the dynamic changes of network topology, a beneficial approach requires the vehicle to be controlled indirectly by the RSUs, which can reduce the number of network nodes (i.e., switches) controlled by the SDN controller. The SDN controller is connected to the RSU, which is then connected to the vehicles to form the “SDN controller-RSU-Vehicles” mode.

It has to be mentioned that the mobility of vehicles is a big challenge for achieving EC-SDIoV, as well as an open problem that draws great attention from many researchers. To mitigate the issue, He et al. [20] proposed to use trajectory prediction to estimate the possible positions to keep mobile-edge computing nodes under control. Ka et al. [33] and Li et al. [38] proposed to make vehicles to be controlled indirectly by accessing RSU for reducing the number of network nodes so as to decrease the network topology changes. Besides, taking the virtual machine migration technology [39]–[41] to achieve smooth migration of mobile vehicle crossing different access points or different SDN controllers is also an effective method.

**B. Reliability Assessment With the Reprocessing Mechanism**

Future advanced vehicular services (e.g., auto/assisted driving) usually have lots of computation-intensive and latency-sensitive tasks to cope with. However, in practice, the real-time application processing is prone to failure because of the inevitable disturbances (e.g., communication errors and outages, and insufficiency computing capability to complete the tasks). Such insufficiencies in supporting safety-critical IoV services may lead to catastrophic loss. Considering heterogeneous resources of EC-SDIoV, it is desirable to equip a reliable computation offloading scheme to support low-latency and high-reliability IoV services.

Once a vehicle requests to the access RSU, the access RSU forwards the service request to the SDN controller. Then, the SDN controller decides the offloading proportion of application-based computation task and orchestrates offloaded task and available resources among the access RSU, mobile-edge node, and fixed edge node for parallel processing. However, once any subtask in an edge node fails, the entire task could turn to failure without fault-tolerant capability. Considering the insufficient communication and computation capabilities of IoV, the reliability-oriented task allocation strategy integrated with the reprocessing mechanism is conceived to enhance the reliability of the system, instead of the utilizing redundancy technology. Different from [42] and [43], the reprocessing mechanism is the key feature in this article by considering all failed cases to form a two-stage reliability allocation strategy. Thus, the total latency of completing the offloaded task includes two aspects: 1) normal processing latency and 2) reprocessing latency. Because of the heterogeneity of the edge nodes, each node has different communication bandwidths, computing resources, failure rates, etc. For example, an edge node has a high computational capability, but it might suffer a high failure rate and poor communication link. Therefore, it is challenging to distribute appropriate subtasks over multiple edge nodes. Moreover, if an edge node responsible for a large proportion of task turns outage, other fault-free nodes need to re-execute the aborted task, which leads to long reprocessing latency. Therefore, the task allocation strategy of the offloaded task also needs to be adjusted considering the possible reprocessing or anticipatory methodology [14], [44]. Considering the actual situation of real-time IoV applications, in this article, reliability is defined as the probability of successful completion of a computing task under a predetermined latency constraint.

If only simple reprocessing after failure is considered, the execution time, including, normal processing and reprocessing is likely to exceed the latency constraint, and the reliability will be low. Therefore, we consider all the failure cases in advance, and the initial allocation has already taken into account the factors of failure and reprocessing, so that the total processing latency including that of reprocessing will not beyond the given latency constraint, thus the reliability is improved further. Therefore, we first analyze all possible failure cases. For instance, there exist two heterogeneous nodes, among them, 1 denotes the subtask assigned to the node executed successfully and 0 represents the subtask failed, thus, there are four cases in total, i.e., 00, 01, 10, and 11. Second, we calculate the execution time of each case according to the two-step strategy. Finally, by calculating the occurrence probability of each case and judging whether the execution time exceeds the latency constraint, we can accurately and comprehensively evaluate the reliability of the offloading scheme, as shown in Fig. 3. Then, the offloading scheme is adjusted and optimized to make more failure cases satisfy latency constraint for achieving high reliability. The joint reliability optimization scheme will be discussed in Section IV.

**IV. SYSTEM MODEL OF RELIABLE COMPUTATION OFFLOADING AND ALGORITHM**

**A. Reliability Model in Computation Offloading**

Fig. 4 shows the computation offloading in EC-SDIoV, where vehicles can offload the whole or a portion of their computation tasks to the EC-SDIoV architecture. We assume the initiator vehicle (i.e., vehicle $s$) has a computation-intensive and latency-sensitive task $\Psi_s \triangleq \{C_n, D_n, T_n\}$ that needs to
TABLE I
KEY NOTATIONS AND DEFINITIONS

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Psi_n$</td>
<td>computation task</td>
</tr>
<tr>
<td>$\theta$</td>
<td>offloading coefficient</td>
</tr>
<tr>
<td>$J$</td>
<td>the reprocessing task</td>
</tr>
<tr>
<td>$T_n, \alpha_n$</td>
<td>the latency constraint, computational complexity of task $\Psi_n$</td>
</tr>
<tr>
<td>$C_n, C'_n$</td>
<td>the number of CPU cycles needed to process the task $\Psi_n$ and task $J$</td>
</tr>
<tr>
<td>$D_n, D'_n$</td>
<td>the data size of task $\Psi_n$ and task $J$</td>
</tr>
<tr>
<td>$d, d'$</td>
<td>the data size of subtask $\psi_i$ and $\psi'_i$</td>
</tr>
<tr>
<td>$k_{\text{local}}$</td>
<td>whether vehicle $s$ completes the assigned task successfully</td>
</tr>
<tr>
<td>$R(\Psi, \theta)$</td>
<td>the reliability of task $\Psi_n$ with $\Psi$ and $\theta$</td>
</tr>
<tr>
<td>$\Psi, \Psi'$</td>
<td>task allocation of offloaded-task $\theta\Psi_n$, and task $J$</td>
</tr>
<tr>
<td>$R_{\text{UL}}, R_{\text{DL}}$</td>
<td>uplink rate and downlink between edge node and the access RSU</td>
</tr>
<tr>
<td>$P_\phi(I, \Phi, I', \Phi')$</td>
<td>the probability of the $\phi$th case</td>
</tr>
<tr>
<td>$\tau_{\text{lost}}, \tau_{\text{TR}}, \tau_{\text{Re}}$</td>
<td>the transmission time of the subtask from the access RSU to other fixed edge nodes, mobile edge nodes, and itself</td>
</tr>
</tbody>
</table>

Fig. 3. Reliability assessment.

be completed, where $C_n$ represents the total number of CPU cycles required to accomplish the task $\Psi_n$, $D_n$ is the input data size of $\Psi_n$. $T_n$ is the latency constraint of task $\Psi_n$. We define that dividing $C_n$ by $D_n$ is computational complexity $\alpha_n$, i.e., cycles/byte, meeting $C_n = \alpha_n D_n$ [46]. All the subtasks of task $\Psi_n$ share the same computational complexity [32]. The access RSU, which achieves the access of vehicle $s$, could receive computation offloading requests of vehicles in a certain area, and then requests the SDN controller to obtain the offloading proportion and task allocation scheme of an offloaded task according to the FPSO-MR algorithm in Section IV-C. The access RSU tells vehicle $s$ about the offloaded proportion of the computation task. Then, the access RSU receives an offloaded task and decomposes it to multiple subtasks. After that, these subtasks are distributed to multiple edge computing nodes for parallel execution. When the assigned subtasks are completed, the results of them are returned to the access RSU. Next, the access RSU integrates the results and sends them back to vehicle $s$. It is noted that partition a task is an open issue which needs to be studied further. Therefore, for convenience, it can be assumed that the task can be partitioned into several independent subtasks sharing common computation complexity [32]. Each edge node will process the assigned subtask instantly with maximum CPU frequency.

For easy reference, Table I summarizes the key notations and their definitions used in this article.

In this article, we only consider the single user scenario, in fact, for multiple users, it can be decoupled into the independent single-user problem, based on the slicing technology of the network and computation sources [45]. The common assumption can be found in similar works [29], [31].
The coordinate of the access RSU is \((x_0, y_0)\), while \(K = \{(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\}\) are 2-D coordinates of nearby available vehicles which prepare to receive tasks from the access RSU. Moreover, \((x_s, y_s)\) is the coordinate of the initiator vehicle \(s\). In EC-SDIoV, the access RSU connects all available edge nodes and vehicle \(s\) directly. The SDN controller keeps global state information up to date, including positions, directions, velocities, network states, etc. Without loss of generality, we consider vehicle \(i\) travels at a constant speed \(v_i\) in a fixed direction. The distance between vehicle \(i\) and the access RSU can be represented as

\[
g_{ir} = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}.
\]

From [36], the uplink rate \(R_{UL}(i, \text{RSU})\) and downlink rate \(R_{DL}(\text{RSU}, i)\) between vehicle \(i\) and the access RSU are given by

\[
\begin{align*}
R_{UL}(i, \text{RSU}) &= W_{UL} \log_2 \left( 1 + \frac{\sigma_i (g_{ir}^{-\alpha} |h_0|^2)}{N_0} \right) \quad (2a) \\
R_{DL}(\text{RSU}, i) &= W_{DL} \log_2 \left( 1 + \frac{\sigma_R (g_{ir}^{-\alpha} |h_0|^2)}{N_0} \right) \quad (2b)
\end{align*}
\]

where \(W_{UL}\) and \(W_{DL}\) are the uplink and downlink channel bandwidths between vehicle \(i\) and the access RSU, respectively, while \(\sigma_i\) and \(\sigma_R\) are the transmission powers of vehicle \(i\) and access RSU, respectively. Moreover, \(\alpha\) is the path-loss exponent which ranges from [2, 5], and \(h_0\) is the complex Gaussian channel coefficient that follows the complex normal distribution \(CN(0, 1)\). \(N_0\) is the additive white Gaussian noise (AWGN).

When vehicle \(s\) chooses to offload task \(\Psi_n\), it can offload a portion or whole of its task, i.e., \(\theta \Psi_n\), where \(\theta\) is the offloading coefficient. \(\theta = 0\) represents that the vehicle \(s\) chooses to deal with the task locally, while \(0 < \theta < 1\) represents that the vehicle \(s\) chooses to offload a portion. \(\theta = 1\) represents that the vehicle \(s\) chooses to offload the whole of the task to EC-SDIoV. We consider vehicles as mobile-edge nodes, belonging to \(L_m = \{l_1, l_2, \ldots, l_p\}\), RSUs as fixed edge nodes, belonging to \(L_f = \{l_{p+1}, l_{p+2}, \ldots, l_{p+q}\}\), and \(L_R\) as the access RSU. Hence, \(L_m, L_f,\) and the access RSU compose all edge nodes \(L = \{l_1, l_2, \ldots, l_{p+q}\}\). \(Z_{fi}\) is defined as the CPU frequency of the fixed edge node \(j\), while \(Z_{mi}\) denotes the CPU frequency of the mobile-edge node \(i\). Moreover, \(Z_R\) and \(Z_S\) represent the CPU frequencies of the access RSU and of vehicle \(s\), respectively.

The access RSU decomposes offloaded task \(\theta \Psi_n\) into several subtasks, i.e., \(\Psi = \{(\psi_1, \psi_2, \ldots, \psi_s, \psi_{s+1}, \ldots, \psi_{s+p+q})\\}\), where \(\psi_s\) represents the task allocation vector. \(\psi_i \triangleq [c_i, d_i]\) represents the assigned subtask on edge node \(l_i\), where \(c_i\) denotes the required CPU cycles, while \(d_i\) denotes input data size. Similarly, \(\psi_R \triangleq [c_R, d_R]\) represents the assigned subtask on the access RSU. The \((1 - \theta)\psi_n \triangleq \{(1 - \theta)c_n, (1 - \theta)d_n\}\) is the task which is decided to execute on vehicle \(s\) locally. To facilitate the discussion, we consider there is no overlap between any subtasks in satisfying \(\sum_{i=1}^{p} c_i + \sum_{j=p+1}^{p+q} c_j + c_R = \theta c_n\) and \(\sum_{i=1}^{p} d_i + \sum_{j=p+1}^{p+q} d_j + d_R = \theta d_n\), i.e., the subtasks on mobile-edge nodes, fixed edge nodes, and access RSU can add up to the offloaded task \(\theta \Psi_n\).

The latency of vehicle \(s\) transmitting the offloaded task \(\theta \Psi_n\) to the access RSU is defined as

\[
T_{UL} = \frac{\theta D_n}{R_{UL}(s, \text{RSU})}.
\]

Then, the access RSU divides offloaded task \(\theta \Psi_n\) into \(\Psi = \{(\psi_1, \psi_2, \ldots, \psi_s, \psi_{s+1}, \ldots, \psi_{s+p+q}, \psi_R)\}\) and distributes them to multiple edge nodes for distributed computing, according to the result of the FPSO-MR algorithm on SDN controller. The transmission latency from the access RSU to mobile-edge nodes, fixed edge nodes, and itself can be denoted by

\[
\begin{align*}
T_{i}^{DL} &= \frac{d_i}{R_{UL}(\text{RSU}, i)}, & i & \in L_m, \\
T_{j}^{TR} &= \frac{d_j}{R_{UL}(\text{RSU}, j)}, & j & \in L_f, \\
T_{R}^{TR} &= 0, & R & \in L_R
\end{align*}
\]

respectively. After received the assigned subtasks from the access RSU, these edge nodes will process the subtasks immediately. The computation latency of these subtasks on fixed edge nodes, mobile-edge nodes, and the access RSU can be denoted by

\[
\begin{align*}
T_{i}^{Comp} &= \frac{c_i}{Z_{mi}}, & i & \in L_m, \\
T_{j}^{Comp} &= \frac{c_j}{Z_{fj}}, & j & \in L_f, \\
T_{R}^{Comp} &= \frac{c_R}{Z_R}, & R & \in L_R
\end{align*}
\]

respectively. According to the task allocation vector \(\Psi\), each edge node executes different subtasks in parallel. The latency of edge nodes returning the results to the access RSU, and the latency of the access RSU returning the results back to vehicle \(s\) are negligible, because the size of computation results are generally much smaller than the input data [47]. Therefore, the latency for edge nodes completing the offloaded part \(\theta \Psi_n\) can be given by

\[
T_{Norm} = \max_{i \in L_m, j \in L_f, R \in L_R} \left\{ T_{i}^{DL} + T_{i}^{Comp} + T_{j}^{TR} + T_{j}^{Comp} + T_{R}^{TR} + T_{R}^{Comp} \right\}.
\]

The latency of vehicle \(s\) computing the task \((1 - \theta)\psi_n\) locally is denoted as

\[
T_{Local} = \frac{(1 - \theta)c_n}{Z_S}.
\]

Therefore, the total execution latency of task \(\psi_n\) relies on the maximum between local execution latency and offloaded execution latency, which is given by

\[
T_{Total} = \max \left\{ T_{Local}, T_{UL} + T_{Norm} \right\}.
\]

Initially, without considering system reliability, the task allocation \(\Psi = \{(\psi_1, \psi_2, \ldots, \psi_s, \psi_{s+1}, \ldots, \psi_{s+p+q}, \psi_R)\}\) and the offloading coefficient \(\theta\) only need to be optimized to meet
corresponding failure probability. Denote the failure rates of mobile-edge node and its offload-
during the offloading latency. Specifically, under a task allocation scheme considering reprocessing in the EC-SDIoV scenario is proposed. This section aims to find the optimal task allocation and offloading coefficient $\theta$, which can satisfy the latency constraint as well as accommodate more failures. Since it is impossible to predict the specific situation of failure in advance, all possible cases and corresponding occurrence probability are discussed.

Given that 1 represents that both the edge computing node and the corresponding link are fault free, and 0 represents that the node or/and the corresponding link is faulted. Thus, for $p$ mobile-edge nodes, $q$ fixed edge nodes and local vehicle $s$, there are $2p+q+1$ cases according to whether subtasks are completed or not, as shown in Fig. 5.

Considering the 8th case, we denote that the mobile and fixed edge nodes which complete assigned subtasks, belong to sets $\Gamma$ and $\Phi$, respectively. In contrast, the mobile and fixed edge nodes which fail to execute assigned subtasks, belong to sets $\Gamma'$ and $\Phi'$, respectively. Let $k_{\text{local}} = 1$ represent that vehicle $s$ successfully completes its task $\Psi_n$ in the worst case (by the end of the task, the vehicle fault occurs and the task need to be re-executed, i.e., twice the local execution time) [52], which is given as

$$T^\text{Local} = \frac{T^\text{Local}}{2\theta}, \quad \text{if } k_{\text{local}} = 1,$$

$$T^\text{Local} = \frac{T^\text{Local}}{2\theta}, \quad \text{if } k_{\text{local}} = 0.$$
The total latency of offloading execution depends on four parts: task transmission latency $T^{UL}$ (i.e., transmission latency from vehicle $s$ to the access RSU); normal processing latency $T^{Norm}$ on fault-free edge nodes according to (6), where the processing nodes change from all nodes to fault-free nodes, as shown in (18); reprocessing latency $T^{Re}$ on fault-free edge nodes which includes both retransmission and recomputing latency; and the time lost $T^{Lost}$ caused by the reprocessing procedure, such as queuing latency. The total execution time $T_{Re}^\text{Total}$ of task $\Psi_n$ depends on the maximum latency between local execution and offloading execution, which is given by

$$T_{Re}^\text{Total} = \max\left\{ T_{Re}^\text{Local}, T^{UL} + T^{Norm} + T^{Lost} + T^{Re} \right\}. \quad (17)$$

Under the task allocation $\Psi$ and offloading coefficient $\theta$, $T^{Norm}$ is defined as the maximal execution time of fault-free subtasks, which is formulated as

$$T_{Re}^\text{Norm} = \max_{i \in \Gamma, j \notin \Phi, \Re, L} \left\{ T_{i}^{DL} + T_{i}^{Comp}, T_{j}^{TR}, T_{j}^{Comp}, T_{R}^{TR}, T_{R}^{Comp} \right\}. \quad (18)$$

Since the reprocessing latency affects total execution time, $T^{Re}$ should be as small as possible to reduce $T_{Re}^\text{Total}$, hence, all fault-free nodes are introduced to reprocessing. Moreover, for fault-free mobile-edge nodes, the offloading transmission rate between the access RSU and mobile nodes are changed due to the mobility of vehicles. We consider vehicle $i$ traveling along the $x$-axis at the speed of $\tau_i$, and it is within the communication coverage of the access RSU during the execution of task $\Psi_n$. The coordinate of vehicle $i$ is defined as $(x_i, y_i)$ at the initial offloading. Considering the mobility of the vehicle, the coordinate of vehicle $i$ is changed to $(x_i + \tau_i T_{Re}^\text{Norm}, y_i)$ at the reprocessing offloading. The new distance between vehicle $i$ and the access RSU is calculated as

$$g_i' = \left[ (x_i + \tau_i T_{Re}^\text{Norm} - x_0)^2 + (y_i - y_0)^2 \right]^{\frac{1}{2}}. \quad (19)$$

Therefore, the communication rate between the access RSU and vehicle $i$ during reprocessing is changed to

$$R^{DL}(\text{RSU}, i) = W^{DL} \log_2 \left( 1 + \frac{\sigma_{RX}(g_i')^{-\alpha}|h_0|^2}{N_0} \right). \quad (20)$$

Under a reprocessing task allocation $\Psi' = (\psi'_1, \psi'_2, \ldots, \psi'_j, \ldots, \psi'_{R})$, $\psi'_i \triangleq (c'_i, d'_i)$, according to (4a)–(6), the reprocessing $T^{Re}$ both considering retransmission and recomputing latency can be obtained by

$$T^{Re} = \max_{i \in \Gamma, j \notin \Phi, \Re, L} \left\{ \frac{d_i'}{R^{DL}(\text{RSU}, i)} + \frac{c'_i}{Z^M_j}, \frac{d'_j}{R^{UL}(\text{RSU}, j)} + \frac{c'_j}{Z^F_j}, \frac{c'_R}{Z^R_j} \right\}. \quad (21)$$

Whether the computation task $\Psi_n$ is reliable depending on whether the execution time $T_{Re}^\text{Total}$ is less than the latency constraint $T_n$. $O_\delta$ is introduced to indicate whether the $\delta$th case is reliable, i.e.,

$$O_\delta = \begin{cases} 1, & \text{if } T_{Re}^\text{Total} \leq T_n \\ 0, & \text{if } T_{Re}^\text{Total} > T_n. \end{cases} \quad (22)$$

When task $\Psi_n$ is with the allocation $\Psi$ and offloading coefficient $\theta$, considering all possible cases and the corresponding occurrence probability $P_\delta(\Gamma, \Phi, \Gamma', \Phi')$, the reliability of that the EC-SDiVo completing the task successfully can be represented as

$$R(\Psi, \theta) = \sum_{\delta=0}^{2^{p+q+1}-1} P_\delta(\Gamma, \Phi, \Gamma', \Phi') O_\delta. \quad (23)$$

Therefore, the optimization problem which maximizes the reliability within latency constraint considering the reprocessing mechanism can be modeled as follows:

$$\mathcal{P}1: (\Psi, \theta) = \arg \max \left\{ \sum_{\delta=0}^{2^{p+q+1}-1} P_\delta(\Gamma, \Phi, \Gamma', \Phi') O_\delta \right\} \quad (24a)$$

s.t. \begin{align*}
& \sum_{i=1}^{p} c_j + \sum_{j=p+1}^{p+q} c_j + c_R = \theta C_n \\
& \sum_{i=1}^{p} d_i + \sum_{j=p+1}^{p+q} d_j + d_R = \theta D_n \\
& \sum_{i \in \Gamma} c_i + \sum_{j \notin \Phi} c_j = C'_n \quad (24b) \\
& \sum_{i \in \Gamma} d_i + \sum_{j \notin \Phi} d_j = D'_n \quad (24c) \\
& 0 \leq \theta \leq 1 \quad (24d) \\
& T_{Re}^\text{Total} \leq T_n. \quad (24g)
\end{align*}

C. Fault-Tolerant Particle Swarm Optimization Algorithm for Maximizing Reliability

Due to the existence of the binary functions, i.e., (22), the optimization problem $\mathcal{P}1$ is nonconvex, and cannot be transformed into the convex problem easily. Besides, the problem is also NP-hard, which can be proved in the following.

Theorem 1: If one of the subproblem of a problem is NP-hard, the problem is NP-hard.

Proposition 1: $\mathcal{P}1$ is an NP-hard problem.

Proof: In fact, $\mathcal{P}1$ aims to maximize the sum of the reliability of $2^{p+q+1}$ cases. Therefore, if we only consider the best environment, which means that all nodes are fault free [i.e., the $(2^{p+q+1}-1)$th case in Fig. 5], the problem $\mathcal{P}1$ is equivalent to $\mathcal{P}2$

$$\mathcal{P}2: (\Psi, \theta) = \arg \max \left\{ \prod_{i=1}^{p} R_{m_i} \prod_{j=p+1}^{p+q} R_{j_j} \right\} \quad (25a)$$

s.t. \begin{align*}
& (24b), (24c), (24f) \quad (25b)
\end{align*}

The special case is the so-called distributed system reliability (DSR) problem, and has already been shown to be NP-hard [53]. Hence, so is $\mathcal{P}1$. ■

Since the problem formulated in (24a) is nonconvex and NP-hard, the heuristic algorithms, which have no requirement to the convexity of the problem as well as known with fast search speed in the tackling NP-hard problem, are a good choice to solve problem $\mathcal{P}1$. The particle swarm optimization
algorithm (PSO) which known as its efficient global search ability, achieves great success in numerous fields, e.g., image processing, neural network training, etc. [54]. Hence, in this article, we propose a fault-tolerant algorithm based on PSO, for maximizing the reliability, named FPSO-MR.

In FPSO-MR, there are several solutions working in parallel in the iteration process, each solution of the optimization problem is called a particle in the search space, and the optimal solution of the problem corresponds to the “best solution.” In each iteration, particles in the population can learn from their own “experience” (i.e., historical location), and also according to the experience of the optimal particle in the population, so as to determine how to adjust and change the direction and speed of flight in the next iteration. Therefore, with gradual iteration, the whole population of particles will converge to the optimal solution, eventually.

In the FPSO-MR, for a particle swarm with $u$ particles, when the algorithm is executed to the $i$ generation, the location of each particle $i$ is denoted as $X_{i}(I) = \{x_{i1}, x_{i2}, \ldots, x_{ip}, x_{i}^{r}, x_{i}^{\theta}\}$, which represents an offloading scheme $(\Psi_{i}, \theta) = \{\psi_{i1}, \psi_{i2}, \ldots, \psi_{ip}, \psi_{i}^{r}, \psi_{i}^{\theta}\}$. The velocity of each particle $i$ is $V_{i}(I) = \{v_{i1}, v_{i2}, \ldots, v_{ip}, v_{i}^{r}, v_{i}^{\theta}\}$ representing the moving velocity that changes its position. The best position of each particle $i$ is $B_{i}(I) = \{b_{i1}, b_{i2}, \ldots, b_{ip}, b_{i}^{r}, b_{i}^{\theta}\}$. The best position of the swarm is denoted as $\text{Best}(I) = \{\xi_{i1}, \xi_{i2}, \ldots, \xi_{ip}, \xi_{i}^{r}, \xi_{i}^{\theta}\}$. Equations (26) and (27) show the updating formulas of particle’s velocity and position, respectively. We employ a constriction factor $\chi$ in the velocity updating formula, which is proposed by Clerc and Kennedy [55]

\[
V_{i}(I+1) = \chi \cdot (w \cdot V_{i}(I) + \mu_1 \cdot r_1 \cdot (B_{i}(I) - X_{i}(I)) + \mu_2 \cdot r_2 \cdot (\text{Best}(I) - X_{i}(I)))
\]

(26)

\[
X_{i}(I+1) = X_{i}(I) + V_{i}(I+1)
\]

(27)

where $w$ is inertia weight, while $\mu_1$ and $\mu_2$ are learning factors, and $r_1$ as well as $r_2$ are random numbers between $[0, 1]$, according to [23].

The constriction factor $\chi$ is given by

\[
\chi = \frac{2}{\sqrt{2 - \psi - \sqrt{\psi^2 - 4\psi}}}
\]

(28)

where $\psi = \mu_1 + \mu_2$. The constriction factor guarantees the convergence of population and prevents explosion of particle’s velocity.

The optimization problem formulated in (24a) is the constrained problem, however, PSO is unable to deal with the constrained problem directly. Thus, the exterior penalty function method [56] is adopted to transform the original problem into an unconstrained problem. The fitness function of particle $i$ is reconstructed as follows:

\[
\Theta(X_{i}) = \left\{ \begin{array}{ll}
R(X_{i}), & \text {if } X_{i} \in H \\
R(X_{i}) + r \sum_{j=1}^{p+q+3} R_{j}(X_{i}) + \phi(X_{i}, I), & \text {if } X_{i} \in N
\end{array} \right.
\]

(29)

\[
R_{j}(X_{i}) = \left\{ \begin{array}{ll}
\max(0, -x_{ij}), & 1 \leq j \leq p + q + 2 \\
|\sum_{k=1}^{p+q} x_{ik} + x_{ir}^{r} + x_{i}^{\theta} - \Psi_{n}|, & j = p + q + 3
\end{array} \right.
\]

(30)

\[
\phi(X_{i}, I) = \text{Wor}(I) - \max_{X_{i} \in N} \left\{ R(X_{i}) + r \sum_{j=1}^{p+q+3} R_{j}(X_{i}) \right\}
\]

(31)

\[
\text{Wor}(I) = \max \{ \text{Wor}(I-1), \max R(X_{i}) \}
\]

(32)

$\Theta(X)$ is the reconstructed target function, in which, $r$ is the penalty factor. $H$ and $N$ are the feasible region and infeasible region, respectively. $R_{j}(X)$ is the constraint violation value of the $j$th constraint for the infeasible solutions. $\phi(X, I)$ is the additional heuristics value to accelerate the convergence of the FPSO-MR. Moreover, $\text{Wor}(I)$ records the maximum fitness value of the feasible particles after $I$-generation iterations.

Algorithm 1 summarizes the FPSO-MR.

The time complexity of FPSO-MR is $O(u \times MaxG \times 2^{p+q+1})$, where $MaxG$ represents the maximum evolution generation, while $2^{p+q+1}$ is the number of failure cases in the context of $p$ mobile-edge nodes, $q$ fixed edge nodes as well as local vehicle $s$. FPSO-MR seems with relatively high complexity, because the algorithm traverses possible cases of the task execution to assess and improve reliability comprehensively. But in fact, traversing all of these cases is to obtain the upper bound of optimal reliability performance. In practice, to achieve a tradeoff between performance and utility, on the one hand, we can select appropriate nodes instead of all nodes, which is expected to reduce the complexity while maintaining high reliability. On the other hand, we can also consider that only one or two subtasks cannot be completed to reduce the complexity, because the occurrence probability of more failures is quite a bit lower.

V. Performance Evaluation

In this section, we evaluate the performance of the FPSO-MR. Referring to [36], [45], [57], and [58], $\sigma_{e} = 20 \text{ dBm}$, $\sigma_{R} = 46 \text{ dBm}$, $W_{DL} = W_{UL} = 1 \text{ MHz}$, $\alpha = 4$, $N_{0} = -100 \text{ dBm}$, and $Z_{S} = 2.2 \text{ GHz}$. Moreover, the CPU frequency of fixed edge nodes and mobile-edge nodes is assumed to be uniformly distributed, i.e., $Z^{F} \sim \text{Unif}([6, 7])$ MHz and $Z^{M} \sim \text{Unif}([1, 3])$ MHz. The computation capability of the access RSU is $6.25 \text{ GHz}$.

The basic parameter settings of the FPSO-MR are given as: the number of particles is 100; the maximum number of iterations is 50; learning factors $\mu_1$ and $\mu_2$ are both equal to...
Algorithm 1 FPSO-MR Algorithm

Input: a particle swarm with $u$ particles and $w$, $\mu_1$, $\mu_2$, $r_1$, $r_2$, $r$, $MaxG$

Output: $Gbest$, $\Theta(Gbest)$

1: for each particle $i$ do
2: Initialize $X_i(0), V_i(0)$
3: Set current position as the best position $B_i(0)$ of particle $i$
4: end for
5: Choose the particle position with the best fitness of all $u$ particles as the global best position $Gbest(0)$
6: while 1 to $MaxG$ do
7: for the $i$th particle do
8: Update the velocity of particle $i$ using Eq. (26)
9: Update the position of particle $i$ using Eq. (27)
10: calculate the occurrence probability of the $\delta$th case using Eq. (15)
11: Calculate the total processing latency considering reprocessing using Eq. (17)
12: Estimate whether the $\delta$th case is reliable using Eq. (22)
13: end for
14: Evaluate fitness function of particle $i$ using Eq. (23) and Eq. (29-32)
15: if $\Theta(X_i(l+1)) > \Theta(B_i(l))$ then
16: $B_i(l+1) = X_i(l+1)$
17: end if
18: if $\Theta(X_i(l+1)) > \Theta(Gbest(i))$ then
19: $Gbest(l+1) = X_i(l+1)$
20: end if
21: end for
22: end while
23: return $Gbest$, $\Theta(Gbest)$

2.05; inertia weight $w$ is 0.9; the penalty factor $r$ is 10; and $Wor(0)$ is $10^6$.

In order to further illustrate the impact of the proposed offloading scheme on the latency performance of the EC-SDIoV architecture, we compare it with other offloading schemes.

- **Offload-Proportion-to-M.&F.:** Offload a certain proportion of task to mobile-edge nodes and fixed edge nodes.
- **Offload-Whole-to-M.&F.:** Offload the whole task to mobile-edge nodes and fixed edge nodes.
- **Offload-Whole-to-RSU:** Offload the whole task to the access RSU.
- **Local-Processing:** User’s vehicle executes the whole task directly.
- **Offload-Proportion-to-M.:** Offload a certain proportion of task to mobile-edge nodes.

A. Latency Performances of Different Schemes Versus Computational Complexity

In this part, we analyze the five schemes’ latency performances by setting one access RSU, two fixed edge nodes, and three mobile-edge nodes.

Fig. 6 shows the latency performances of different schemes versus computational complexity. With the increasing of the computational complexity, i.e., cycles/byte, the task becomes increasingly difficult to compute, meaning the more computation latency is needed. Since the scheme offload-whole-to-M.&F. and the scheme offload-whole-to-RSU need to offload the whole task to the edge computing layer, there is still with a certain offloading latency even though the computational complexity is close to 0. In the scheme offload-proportion-to-M.&F., scheme Local-processing, and scheme offload-proportion-to-M., the whole task can be accomplished in the user vehicle $s$, so there is no offloading latency when the task’s computational complexity is close to 0. However, when the computational complexity exceeds 2000, the latency of the scheme offload-whole-to-M.&F. and offload-whole-to-RSU is both obviously lower than the latency of the scheme Local-processing, because the user vehicle $s$ has a weaker computing ability than the scheme offload-whole-to-M.&F. and the scheme offload-whole-to-RSU, which has auxiliary edge nodes. Therefore, despite with higher offloading latency, the total execution time of the scheme offload-whole-to-M.&F. and the scheme offload-whole-to-RSU is also lower than the latency of the scheme Local-processing. Similarly, the latency of the scheme offload-whole-to-M.&F. is lower than the latency of the scheme offload-proportion-to-M. when the computational complexity exceeds 5000. Since the computing ability of the access RSU is weaker than multiple edge nodes, the latency of the scheme offload-whole-to-RSU is higher than that of the scheme offload-proportion-to-M.&F., scheme offload-whole-to-M.&F. and scheme offload-proportion-to-M. Due to the partial offloading and enormous computing power, the latency of the scheme offload-proportion-to-M.&F. is always lower than other schemes.

B. Latency Performances of Different Schemes Versus Input Data Size

In this part, we analyze the five schemes’ latency performances versus input data size, where the computational complexity $\alpha_n$ is set as 7000 cycles/B.

Fig. 7 shows the influences of the input data size on the latency performance. With the increase of input data size, the latency of each schemes increases because more input data brings more computation latency and offloading latency. According to Fig. 6, the relationship of latency
size between five schemes is: scheme offload-proportion-to-M.&F. < scheme offload-whole-to-M.&F. < scheme offload-proportion-to-M. < scheme offload-whole-to-RSU < scheme Local-processing, when the computational complexity is 7000. Although the input data size changes in Fig. 7, the ratio between the needed CPU cycles and the input data size, i.e., computational complexity has not changed. Therefore, the relationship of latency size between five schemes is consistent with 7000 cycles/B of Fig. 6. Compared to offloading latency, the computation latency is dominant when computational complexity is 7000 cycles/B. Therefore, the total execution time of user vehicle s and single access RSU is considerably larger than the other three schemes. Scheme offload-proportion-to-M.&F., which has the maximum computing ability and lower transmission latency, has the best latency performance.

C. Latency Performance Comparison of Random Tasks

We evaluate the latency performance among different tasks. Referring to face recognition application, the input data size and the total number of CPU cycles are Gaussian distributions, $D_n \sim N(400, 100)$ (kB) and $C_n \sim N(1000,10)$ (Megacycles) [26]. Fig. 8 shows the latency results of ten random tasks. From Fig. 8, the latency performance of the scheme offload-proportion-to-M.&F. is improved by 26.5%, 55.8%, 72.4%, and 14.4% compared with the scheme offload-whole-to-M.&F., scheme offload-whole-to-RSU, scheme Local-processing, and scheme offload-proportion-to-M., respectively. Moreover, from the input data size and the total number of CPU cycles, we can infer that the computational complexity of the face recognition application is about 2500. Therefore, the relationship of latency size between five schemes is consistent with 2500 cycles/B of Fig. 6.

D. Impact on Reliability Performance of the Reprocessing Mechanism in Different Environments

Fig. 9 shows the comparison between reliable computation offloading with the reprocessing mechanism and that without the reprocessing mechanism in different environments. In the good environment, the failure rates of edge nodes and links are assumed to be uniformly distributed, i.e., $\lambda = \epsilon \sim \text{Unif}(0, 0.03)$, while in the poor environment, we have $\lambda = \epsilon \sim \text{Unif}(0.1, 0.5)$. In the mixed environment, some nodes and links have $\lambda = \epsilon \sim \text{Unif}(0, 0.03)$, and other nodes and links have $\lambda = \epsilon \sim \text{Unif}(0.1, 0.5)$. From Fig. 9, we can see that, with the increase of the computational complexity, the reliability of both two schemes decreases in varying degrees. Because the increase of the computational complexity leads to the increase of computation latency in each node, which can reduce the reliability of the system. However, we can see that the scheme relying on reprocessing is substantially beneficial in terms of improving the reliability performance in comparison to the scheme without the reprocessing mechanism, especially in the poor environment. The reason is that the reprocessing mechanism makes it possible to successfully complete the task within the latency constraint even if some nodes or links fail, while the conventional reliability-oriented task allocation strategy without considering reprocessing, is bound to fail completing the task, once any failure occurs.
TABLE II
SIMULATION PARAMETERS OF EDGE NODES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Master RSU</th>
<th>Fixed Edge Node</th>
<th>Mobile Edge Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>Node 1</td>
<td>Node 2</td>
<td>Node 3, Node 4, Node 5, Node 6, Node 7</td>
</tr>
<tr>
<td>Computation Capability</td>
<td>6.25GHz</td>
<td>7GHz</td>
<td>2GHz, 2.2GHz, 1.2GHz, 2.3GHz, 1.8GHz</td>
</tr>
<tr>
<td>Failure Rate</td>
<td>0</td>
<td>0.012</td>
<td>0.033, 0.056, 0.017, 0.047, 0.035</td>
</tr>
</tbody>
</table>

Fig. 10. Reliability performance comparison of different numbers of edge nodes.

E. Reliability Performance Comparison of Different Numbers of Edge Nodes

In IoV, due to the mobility of vehicles as well as the instability of links, the number of available nodes is uncertain. Hence, in this part, we evaluate the impact of the number of available nodes on system reliability. The detailed simulation parameters of all edge nodes are set in Table II.

In Fig. 10, we compare the reliability performance with 0.4-s latency constraint in four different numbers of edge nodes: 1) four edge nodes including nodes 1–4; 2) five edge nodes including nodes 1–5; 3) six edge nodes; and 4) seven edge nodes have similar definition. (Node 1 is the access RSU, node 2 is the fixed edge node, and nodes 3–7 are mobile-edge nodes.) From Fig. 10, we can find that seven edge nodes have the highest reliability. It is because the system has more computing power, which brings less computation latency. Therefore, the system could tolerate more failures of subtasks to achieve higher reliability. However, more edge nodes do not always mean better. For instance, the reliability of four edge nodes is close to that of seven edge nodes when computational complexity is 4500. For cost reason, the four edge nodes are a better choice when computational complexity is 4500. We can observe that the reliability gap between seven edge nodes and four edge nodes becomes evident with the increase in computational complexity, especially in 8000 cycles/B. The reliability of four edge nodes and five edge nodes is equal to 0 because the computing power of them is unable to process the tasks within the given latency constraint. Therefore, it is worth studying further to choose the appropriate number of edge nodes.

Fig. 11. Reliability performance comparison of different algorithms.

F. Reliability Performance Comparison of Different Algorithms

Fig. 11 shows the reliability performance of FPSO-MR, state-of-the-art heuristic algorithms (i.e., genetic algorithm (GA) [60] and simulated annealing algorithm (SA) [61]), and typical load balancing algorithms (i.e., weighted round-robin algorithm (WRR) [62] and greedy load balancing algorithm (Greedy-LB) [63]), versus latency constraints. We can see that the reliability performance is negatively correlated with latency constraint. The reason is that the stronger the latency constraint, the more likely the failure case occurs, thus the reliability is lower. Obviously, the FPSO-MR and other state-of-the-art heuristic algorithms are superior to the typical load balancing algorithms. It is because that the WRR and Greedy-LB algorithms only consider the computing capabilities and failure rates of edge nodes but ignore the impact of the capabilities and failure rates of transmission links, which may result in the mismatch between the optimal solution with the real situation. As a contrast, the heuristic algorithms which jointly consider computing and communication capabilities as well as failure rates and achieve relatively high-reliability performance all along. Moreover, benefit by the stronger global optimization ability, the reliability performance of FPSO-MR is obviously superior to the GA and SA after the same number iterations.

VI. Conclusion

In this article, we proposed the EC-SDIoV architecture, which was expected to support reliable latency-sensitive
applications. Our target is to ensure the high probability of executing the application successfully, i.e., high reliability in the EC-SDIoV architecture. To achieve it, the reliable computation offloading strategy, which jointly considering partial offloading, task allocation, and reprocessing mechanism was proposed to improve the reliability of latency-sensitive applications. Due to the potential failure of processing nodes and communication links, the reliable offloading strategy traverses possible failure cases of the subtask execution to better assess and optimize reliability. Furthermore, to solve the optimization problem formulated, the FPSO-MR algorithm was designed. The simulation results demonstrated that the FPSO-MR could provide high reliability for latency-sensitive applications in the EC-SDIoV network. In future work, we plan to study how to reduce the complexity of the proposed FPSO-MR algorithm, such as the designing node selection algorithm to select the appropriate number of nodes instead of using all of them.

REFERENCES


Wenchi Cheng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in telecommunication engineering from Xidian University, Xi’an, China, in 2008 and 2013, respectively. He is a Professor with Xidian University. He was a Visiting Scholar with the Networking and Information Systems Laboratory, Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA, from 2010 to 2011. He has published more than 80 international journal and conference papers in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE MAGAZINES, IEEE INFOCOM, GLOBECOM, and ICC. His current research interests include 5G wireless networks and orbital-angular-momentum-based wireless communications.

Prof. Cheng received the URSI Young Scientist Award in 2019, the Young Elite Scientist Award of CAST from 2016 to 2018, the Best Dissertation (Rank 1) of China Institute of Communications, the Best Paper Award for IEEE ICC 2018, the Best Paper Nomination for IEEE GLOBECOM 2014, and the Outstanding Contribution Award for Xidian University. He has served or served as an Associate Editor for IEEE COMMUNICATIONS LETTERS and IEEE ACCESS, the IoT Session Chair for IEEE 5G Roadmap, the Wireless Communications Symposium Co-Chair for IEEE GLOBECOM 2020, the Publicity Chair for IEEE ICC 2019, the Workshop Chair for IEEE INFOCOM 2020 IWECN, the Next Generation Networks Symposium Chair for IEEE ICC 2019, and the Workshop Chair for the IEEE ICC 2019 Workshop on Intelligent Wireless Emergency Communications Networks, the IEEE ICCC 2017 Workshop on Internet of Things, and the IEEE GLOBECOM 2019 Workshop on Intelligent Wireless Emergency Communications Networks.

Yong Ren (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Harbin Institute of Technology, Harbin, China, in 1984, 1987, and 1994, respectively. He worked as a Postdoctoral Fellow with the Department of Electronics Engineering, Tsinghua University, Beijing, China, from 1995 to 1997, where he is currently a Professor with the Department of Electronics Engineering and the Director of the Complexity Engineered Systems Lab. He holds 12 patents, and has authored or coauthored more than 100 technical papers in the behavior of computer network, P2P network, and cognitive networks. His current research interests include complex systems theory and its applications to the optimization and information sharing of the Internet, Internet of Things and ubiquitous network, cognitive networks, and cyber–physical systems.

Prof. Ren has served as a reviewer for the IEICE Transactions on Communications, Digital Signal Processing, Chinese Physics Letters, the Chinese Journal of Electronics, the Chinese Journal of Computer Science and Technology, and the Chinese Journal of Aeronautics.

Kwang-Cheng Chen (Fellow, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 1987 and 1989, respectively. From 1987 to 1998, he worked with SSE, Perth, U.K.; COMSAT, Bethesda, MD, USA; IBM Thomas J. Watson Research Center, Ossining, NY, USA; and National Tsing Hua University, Hsinchu, Taiwan, working on the mobile communications and networks. From 1998 to 2016, he was a Distinguished Professor with National Taiwan University, where he also served as the Director of the Graduate Institute of Communication Engineering and Communication Research Center, and an Associate Dean for Academic Affairs, College of Electrical Engineering and Computer Science from 2009 to 2015. Since 2016, he has been a Professor of electrical engineering with the University of South Florida, Tampa, FL, USA. His recent research interests include wireless networks, artificial intelligence and machine learning, IoT and CPS, social networks and data analytics, and cyber security.

Prof. Chen has received a number of awards, including the 2011 IEEE COMSOC WTC Recognition Award, the 2014 IEEE Jack Neubauer Memorial Award, and the 2014 IEEE COMSOC AP Outstanding Paper Award. He has been actively involving in the organization of various IEEE conferences as the General/TPC chair/co-chair, and has served in editorships with a few IEEE journals. He also actively participates in and has contributed essential technology to various IEEE 802, Bluetooth, LTE and LTE-A, 5G-NR, and ITU-T FG ML5G wireless standards.

Hailin Zhang (Member, IEEE) received the B.S. and M.S. degrees from Northwestern Polytechnic University, Xi’an, China, in 1985 and 1988, respectively, and the Ph.D. degree from Xidian University, Xi’an, in 1991. In 1991, he joined the School of Telecommunications Engineering, Xidian University, where he is a Senior Professor and the Dean. He is also currently the Director of Key Laboratory of telecommunications and information system with Xidian University, and an Associate Director of the National 111 Project. He has published more than 150 papers in journals and conferences. His current research interests include key transmission technologies and standards on broadband wireless communications for 5G and 5G-beyond wireless access systems.