REMR: A Reliability Evaluation Method for Dynamic Edge Computing Network Under Time Constraint

Liang Chen, Jianpeng Qi, Xiao Su, and Rui Wang, *Member, IEEE*

***Abstract*—Computation and/or communication-intensive collaborative services accompanied by several distributed tasks/components, such as the services in Internet of Things, can be anywhere nowadays. These services are usually used by users at the Internet edge, making cloud computing struggles with the high end-to-end latency. Thanks to edge computing which pushes resources to the edge, the goals with lower latency can be well satisfied. However, in actual scenarios especially under dynamic edge computing networks, changes exist in resources, including computing, bandwidth, and nodes. Meanwhile, data packets (or flow) among collaborative tasks/components of a service can also not be conserved. These characteristics lead the service reliability hard to be guaranteed and make existing reliability evaluation methods no longer accurate. To study the effect of distributed and collaborative service deployment strategies under such background, we propose a reliability evaluation method (REMR). We first look for the solution set which can meet the time constraints. Then, we calculate the reliability of service supported by the solution set based on the principle of inclusion–exclusion with distributions of available transmission bandwidth and computing resources. Finally, we provide an illustrative example with several real-world data sets to make REMR easy to follow. To make REMR more reliable, we also propose and implement a Monte Carlo simulation method. Experiments prove that the reliability calculated by REMR is nearly the same as the simulation results and both the latencies and the jitters are also at a lower level.**

***Index Terms*—Distributed services, dynamic networks, edge computing, reliability, sequential task.**

I. Introduction

INTERNATIONAL Data Corporation (IDC) estimates that by 2025 41.6 billion devices will be interconnected, and data volume will reach 79.4 zettabytes (ZB) [1]. Current cloud computing architectures cannot afford such an overwhelming amount of devices and data due to high latency, limited bandwidth, high carbon footprint, and poor security [2]. Many traditional services provided in the cloud but with their data generated remotely [3] are thus maintained at a high cost, e.g., the communication and computation-intensive tasks, such as image recognition and target tracking. Thus, edge computing, an accelerator of cloud computing, affords task processing and data analysis everywhere and gains widespread attention. As shown in Fig. 1, edge computing uses users' nearby resources, including smart devices, access point, intermediate servers on the data forwarding path, and so on, to provide faster service response, simultaneously realizes effective use of local resources.

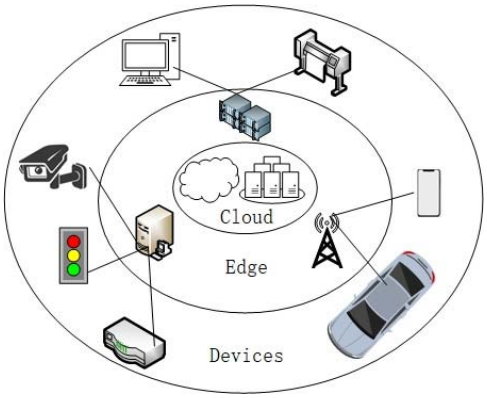


Fig. 1. Typical edge computing examples.

Compared with the centralized cloud computing resources, however, edge resources are generally incapable of executing difficult computation tasks due to their constraints [4]. Thus lots of works deploy service tasks into many intermediate edge nodes in an *edge–edge* or *edge–cloud collaborative* way to provide a high Quality of Service (QoS) [5], e.g., *in-network computing* [6]. This way, only part of intermediate data or results need to be transmitted, which greatly reduces the cost of communication and releases the limited edge resources. For example, instead of transmitting HD video to the remote cloud center, it is more efficient and secure for the camera at the gate of a community to recognize someone passing by with auxiliary components supporting computer vision technologies [7]. This job requires an additional computation-intensive VGG-Net or other model designed for face recognition. Obviously, the limited edge resources, e.g., the camera, cannot afford to finish it independently. But it is possible to perform preliminary processing on the captured pictures and videos via camera given that the preprocess operation will not require so many parameters.[1] This form of deploying a service into multiple

Manuscript received 20 September 2022; accepted 9 October 2022. Date of publication 20 October 2022; date of current version 20 February 2023. This work was supported by the National Natural Science Foundation of China under Grant 62173158 and Grant 61803391. *(Liang Chen and Jianpeng Qi are co-first authors.) (Corresponding author: Rui Wang.)*

The authors are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: wangrui@ustb.edu.cn).

Digital Object Identifier 10.1109/JIOT.2022.3216056

[1]A complete VGG16 model with 138M parameters will take up close to 500 Mb of disk space.

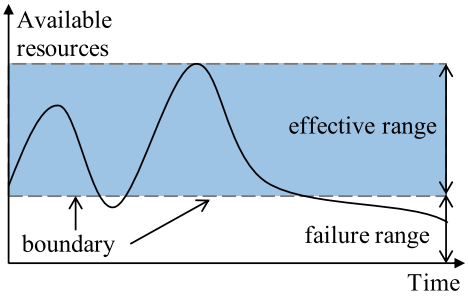


Fig. 2. Curve of available resources change.

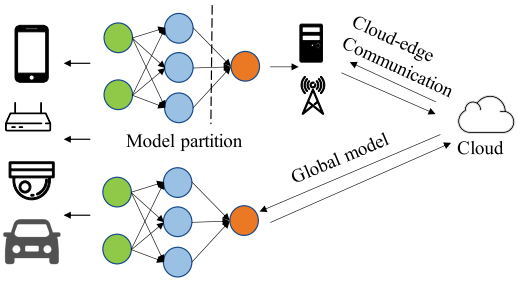


Fig. 3. Edge intelligence scenario.

tasks could improve application's scalability, portability, and availability [8].

In addition to the limited resources, there still exists an issue that the resources of edge computing networks are dynamic, highly changeable, and even unreliable [9], [10], causing the distributed collaborative services with a poor reliability. Specifically, each edge node is generally responsible for more than one task and service vendor, which means that its available resources, e.g., the computing and the transmitting resources, are changeable with time [11]. For example, short-term network congestion or insufficient machine performance will lead to a decrease in QoS. Thus, it is essential that the collaborative services could got a more reliable promise before the service provision.

In this article, we focus on how to evaluate the reliability of a service that consists of several collaborative subtasks. Conventional evaluation methods of reliability consider nothing but network-transmission latency. However, in an edge scenario, the computational latency should also be considered, which includes many dynamic characteristics.

To be more clear, as shown in Fig. 2, when the *available resources* of a node are much than the *minimal resource requirements* of a service along with time change, the service can operate normally within effective range. However, when the available resources fall below the critical line, the service becomes unavailable, leading to an obvious service jittering. Traditional approaches consider the available resources as a horizontal straight line that does not change, which leads to the execution of lots of preplanned computation even though nodes have few resources left. Thus, it is necessary to jointly consider both the computing and the transmitting resources.

To this end, this article models collaborative tasks of a service and its deployment plan in edge–edge collaborate computing scenario based on reliability, built a dynamic edge computing (DECN) scenario, and proposes a reliability evaluation method of reliability (REMR) to calculate the probability that a distributed service can be completed given a time constraint. Aiming at the unreliability of bandwidth, distribution sampling is used to represent the changeable bandwidth. For the changeable computing resources, we summarize the probability distribution based on historical data modeling. Then, we combine the characteristics of distributed tasks, and find the edge-resource-demand bound to calculate the reliability of the service. The proposed method can not only tell the credibility of the current deployment plan to complete assigned tasks but also give a theoretical support in the early stage to help service providers deploy a more perfect service.

The major contributions of this work are as follows.

1) We propose a brand new reliability evaluation method aiming at complex dynamic edge computing networks. To the best of our knowledge, it is the first try that network bandwidth and computing resources distribution are jointly considered to evaluate the reliability of edge computing distributed services under complex dynamic scenarios with changeable intermediate-data size.
2) Different from traditional network reliability evaluation, nonconserved data flow and computing performance of nodes are considered and implemented. For computational convenience and efficiency, we utilize data distribution and prediction model to deal with dynamic performance.
3) We implement, evaluate, and analyze the performance of the proposed REMR on a discrete-event simulation platform "EasiEI" [12], involving numerous experiments conducted on several real-world cluster data sets. And we also propose a Monte Carlo simulation method for evaluating the reliability of the collaborative services in DECN. Results show that our proposed REMR can keep both a lower latency and jitter.

The remainder of this article is as follows. Section II presents related works about reliability in network transmission and edge computing. Section III introduces theoretical support and basic experimental procedures, along with a reliability algorithm. Section IV illustrates a comprehensive example and simulation to bring more convincing details. Section V concludes this article and provides some directions in future research, focusing on the content that has not been explored in depth.

II. Related Works

Reliability analysis is popular in manufacturing, supply chain, and computer network [13], [14]. Different from the traditional data-transmitting network, computing tasks in edge computing can also be embedded in the network, i.e., the network can support running a distributed collaborative service. For example, a scheduling algorithm cuts a model into two parts based on its partitioning position and performs calculations on different nodes. Fig. 3 depicts a popular edge intelligence scenario where a light DNN model is deployed in the intermediate resource-limited and computing-capable edge

nodes to process the extracted features computed by the end devices and then to transmit the inference results to the cloud. Given that whether the tasks are embedded in the edge computing network, this section is mainly focusing on the reliability analyses on the traditional network and the edge computing network.

A. Network Transmission Reliability

Researchers have done a lot of work on the reliability of network transmission [15], [16]. Simply put, network transmission reliability is the probability of transmitting the data of a given size d from source *src* to destination *dest* within a time threshold T over a dynamic multistate network. Because the network bandwidth is dynamic, traditional methods usually first find *the minimum boundary* required for transmitting the data. Kabadurmus and Smith [17] used the concept of elasticity to combine network capacity and reliability that affect service quality. They comprehensively consider influencing factors of service reliability, uses rerouting to solve failures and splitting data to solve network congestion. Huang et al. [18] proposed, in cloud and fog system, a stochastic-flow cloud–fog computing network model to effectively evaluate the probability of successful data transmission in IoT scenario. Loh et al. [19] used an approximate Lagrangian relaxation algorithm to solve the problem of generating maximum reliability of a computer network. Combined with minimal cut vectors, Ramirez-Marquez et al. [20] proposed a *Monte Carlo simulation* method to estimate the reliability of multistate networks and achieved better numerical results than the theoretical method, and inclusion and exclusion principles. It provided an effective way to solve the problem from other directions. Other works utilizing machine learning methods, such as DNN [21] and SVM [22], trained a network reliability predicting model by considering the status of *every* network link. Since these models ignored the dynamic status of the computing resources, they need more additions to fit edge computing scenarios with distributed tasks.

In a collaborative service scenario, however, in addition to thinking of the minimum boundary required for data transmission, it is also necessary to consider minimum boundary required for computing the task when edge resources dynamically change. Meanwhile, data packets on the forwarding path are also dynamic in collaborative service, which means the intermediate data size d from *src* to *dest* can be vary. Thus, measuring the reliability of a service requires comprehensive consideration, especially when data flow, computing resources, and network bandwidth are dynamic.

B. Edge Computing Reliability

Many works use task offloading techniques to support collaborative service. They commonly distribute the tasks into several edge nodes under time, energy, or budgets constraints, lacking of reliability analysis even support.

Dong et al. [23] designed an optimal offloading strategy and allocation algorithm, considering the reliability of the task which was based on the inert shadow scheme, in the resource-constrained mobile-edge computing system. Shadow scheme, the way that relying on redundancy deployment to improve reliability is commonly used, like disaster tolerance performance of server services. Li et al. [24] studied task allocation of the edge computing system composed of edge server (ES) and edge user (EU). Their model selectively offloads part of the tasks to local edge devices, and the rest tasks are calculated by ES. However, the dynamic change of available edge resources is not taken into account. Kang et al. [25] compared pure cloud and edge–cloud collaborative computing paradigm, explained the importance of sinking part of computing to edge on lower latency and less energy consuming, and proposed a framework, which can adaptively cut the model at a reasonable division point and send them to different computing units with goal of lowest latency and energy consumption. However, supporting edge–edge or edge–cloud collaborative service in dynamic environments should also consider the service reliability or stability; otherwise, frequent scheduling may occur, which can lead to a high cost.

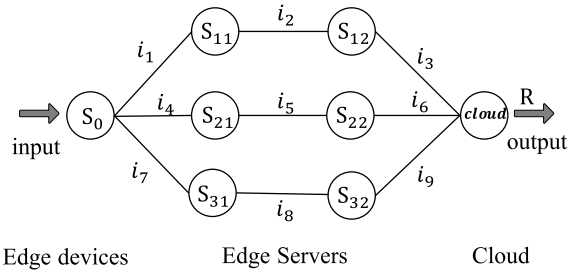


Fig. 4. Brief DECN scenario.

Compared with the above work, in a typical dynamic edge computing scenario, REMR not only combines the network transmitting and computating latency of the tasks but also realizes dynamic modeling according to the availability of nodes resources, data flow without conservation, and fluctuation of network bandwidth. REMR can effectively offer the feasibility of current collaborative service deployment plan, and it has been verified on the simulation platform EasiEI that the calculation results are almost the same as simulation results.

III. Materials and Methods

A collaborative distributed service consisting of several tasks is deployed on edge nodes that can provide computing resources according to the corresponding strategy. Fig. 4 depicts a concrete solution of the scenario presented in Fig. 3. To improve the reliability of service completion, there usually exist several deployment plans by utilizing service redundancy, e.g., two tasks are simultaneously deployed on three paths in Fig. 4. After service deployment is finished, the AI model assigned to the devices s_0 will process input data, and the intermediate data obtained will be transmitted to ES through the line for the next step of the whole service. Depending on the distribution of models and amount of calculations, different deployments may have different numbers of intermediate servers. In the end, the results of all deployments will be transmitted to the cloud that mainly plays a role of result analysis. Note that there may also exist some intermediate edge nodes for transmission only.

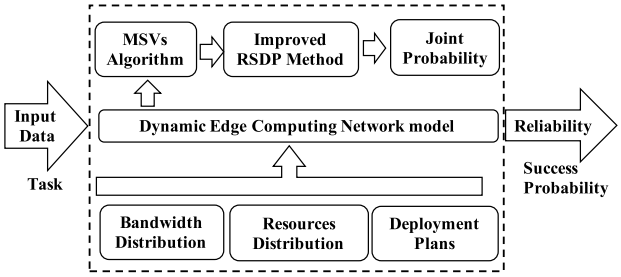


Fig. 5. Workflow of the DECN model.

TABLE I
NOTATIONS AND EXPLANATIONS

S	Set of nodes in a deployment plan, $S = \{s_1, s_2, \ldots, s_n\}$
s_{jk}	kth node on deployment plan j
l_i	Lead time of ith branch, $i = 1, 2, 3, \cdots, n$
X	Bandwidth capacity vector of a deployment plan, $X = \{x_1, x_2, \ldots, x_n\}$.
x_i	Bandwidth of ith branch in a deployment plan
Y	Computing resource capacity vector of a deployment plan, $Y = \{y_1, y_2, y_3, \cdots, y_n\}$
y_i	Computing resource capacity of ith node, $i = 1, 2, 3, \cdots, n$
Δ	Ratio vector of data output to input, $\Delta = \{\delta_1, \delta_2, \cdots, \delta_n\}$
δ_i	Ratio of data output to input through ith node, $\delta_i \geq 0$
T	Time constraint for service
t	Sum time of transmission and calculation for one service
i_j	jth branch in a deployment plan, $j = 1, 2, 3, \cdots, n$
R_i	Reliability of ith solution
R_a	Reliability of deployment plan a
Q_z	Collection of vectors derived from solution set
Q_{MSV}	Collection of vectors derived from Q_z through MSV

In DECN, similar to traditional network transmission, each transmission line has a changeable capacity and corresponding probability distribution. The difference is that the traditional network transmission model is no longer applicable here because middle servers need to complete a large number of computing tasks and data packets during transmission are also changeable. Therefore, for evaluating the service reliability, additional issues including unconserved data flows among intermediate nodes, dynamic edge resources need also to be addressed. In this article, we use a probability distribution observed from the performance monitoring data set to model the dynamic computational performance.

To ensure QoS, thus, it is also important to adopt time constraint T to limit the efficiency of task completion, which is another factor considered to model in this article. Fig. 5 depicts the workflow of our proposed method. Table I lists some notations and explanations in this article.

A. DECN Scenario and Problem Formulation

Usually, a collaborative service denoted as a directed acyclic graph (DAG) can be separated and deployed to different interconnected edge nodes. A plan for deploying a collaborative service to sequential collaborative nodes $S = \{s_1, s_2, \ldots, s_n\}$ could be represented by $\{S, X, Y, \Delta\}$, where $X = \{x_1, x_2, \ldots, x_n\}$ is the bandwidth-capacity vector of the network links and for link (or *branch*) i its available bandwidth $x_i \in [0, M_{xi}]$, $Y = \{y_1, y_2, y_3, \ldots, y_n\}$ the computing resource capacity vector of the edge nodes and for edge node s_i its available computing resource $y_i \in [0, M_{yi}]$, and $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$ the ratio vector of data output to the input given an intermediate service node, where M is an ideal threshold. Because the bandwidth X, computing resource Y, and input/output ratio Δ in this scenario are not constant, we call this the dynamic edge computing network (DECN) scenario.

For a deployment plan, the time required t for transmitting a data chunk of a size C could be represented by (1), where l_i is the lead time for processing the data packets on branch i. We round up (C/x_i) for calculation convenience

$$t = \sum_{i=1}^{n} l_i + \sum_{i=1}^{n} \left\lceil \frac{C}{x_i} \right\rceil. \tag{1}$$

Equation (1) is similar to traditional network-transmission reliability research [15], which does not consider the task running time. In DECN, however, once the intermediate data reaches a node s_i running a task, the node would be immediately activated to consume that data. Presently, in this article, the running time on node s_i is proportional to the size of the input data, and inversely proportional to available computing resource y_i. Due to the blindness of the tasks amount already running, it is difficult to perceive available computing resource of the node when the computing task is coming. Thus, changeable y_i in (2) is used to represent the required computing resource

$$t = \sum_{i=1}^{n} L_i + \sum_{i=1}^{n} \left\lceil \frac{C}{x_i} \right\rceil + \sum_{i=1}^{n} \frac{C}{y_i}. \tag{2}$$

Assume the available resource for each branch x_i and computing resource y_i is independent, which means they can take any value among their range. However, the size of output data after the task has been computed would be changeable, such as dimension reduction and convolution operations. The ratio δ_i of output to the input on node s_i would be saved as a parameter and used as data change factors. Combining the data size change of the node and an initial data size C_0, the time consumed for the deployment plan can be expressed as

$$t = \sum_{i=1}^{n} L_i + \sum_{i=1}^{n} \left\lceil \frac{C_0 \prod_{k=0}^{i-1} \delta_k}{x_i} \right\rceil + \sum_{i=0}^{n} \frac{C_0 \prod_{k=0}^{i-1} \delta_k}{y_i}. \tag{3}$$

Under the predefined constraint of time T, for a time-sensitive service, meeting the user's requirements means the inequality $t \leq T$ need be satisfied, i.e., the reliability $R = \Pr\{t \leq T | S, X, Y, \Delta\}$.

B. MSV and RSDP Algorithms

For a specific deployment plan in the DECN scenario, many state vectors of X and Y can satisfy $t \leq T$. Assuming that a specific solution $(X_i, Y_i) = (x_0, x_1, x_2, \ldots, x_n, y_0, y_1, y_2, \ldots, y_n)$

satisfies $t \leq T$. By (2), we observe that t is dominated by transmission time and calculation time, affected by the available bandwidth of all deployment plans and computing resource of all nodes. This solution (X_i, Y_i) is one of the feasible solutions for the reliability of deployment plan i, which could be calculated by

$$R_i = \Pr\{t \leq T | S, X, Y, \Delta\} = \Pr\{(X_i, Y_i)\} \geq MSV\} \quad (4)$$

where minimal status vectors (MSVs) [26] are the status vectors of the minimal edge resource requirement that satisfy $t \leq T$. Due to the diversity of deployment plan bandwidth distribution and node computing resources distribution, there usually exist many solutions similar to the solutions (X_i, Y_i). Assuming that there are totally z solutions in the set of Q_z that stores every solution satisfying the inequality $t \leq T$, the global reliability of a single deployment plan can be defined as

$$R_a = \Pr\left\{\bigcup_{i=1}^{z} R_i\right\}. \quad (5)$$

However, in this case, the calculation method for the reliability of Q_z exists a lot of repeated calculations, bring a low efficiency. Thus, the MSV mining algorithm that finds the lower bound and eliminates all redundant calculations is used in this article. Vector operations are depicted as the following principles.

1) *Rule 1:* $B \leq A, (b_1, b_2, \ldots, b_n) \leq (a_1, a_2, \ldots, a_n) : b_i \leq a_i$ for each $i = 1, 2, \ldots, n$.
2) *Rule 2:* $B < A, (b_1, b_2, \ldots, b_n) < (a_1, a_2, \ldots, a_n) : B \leq A$ and $b_i < a_i$ for at least one i.

We can remove A in Q_z if $\exists B \leq A$ and $\forall A, B \in Q_z$. Then, the remaining solutions are more representative, also named MSVs. Here is a proof that the reliability of distributed tasks on a certain deployment plan can be represented by the probability calculated by all MSVs on the plan.

Proof: $\forall A, B \in Q_z$ and $A \geq B$, obeying Rules 1 and 2. Then, let

$A = (x_{a0}, x_{a1}, x_{a2}, \ldots, x_{an}, y_{a0}, y_{a1}, y_{a2}, \ldots, y_{an})$
$B = (x_{b0}, x_{b1}, x_{b2}, \ldots, x_{bn}, y_{b0}, y_{b1}, y_{b2}, \ldots, y_{bn})$
$R(A) = \prod_k Pr(x \geq x_{ak}) Pr(y \geq y_{ak})$
$R(B) = \prod_k Pr(x \geq x_{bk}) Pr(y \geq y_{bk})$.

Because, $A \geq B$, assuming that $x_{a0} \geq x_{b0}$, $(x_{a1}, x_{a2}, \ldots, x_{an}) = (x_{b1}, x_{b2}, \ldots x_{bn})$, and $(y_{a0}, y_{a1}, y_{a2}, \ldots, y_{an}) = (y_{b0}, y_{b1}, y_{b2}, \ldots, y_{bn})$.

Then, $R(B) = R(B\{x_0 > x_{b0}\}) + R(B\{x_0 = x_{b0}\})$ which means small values are more representative and could include the probability of large values on demand of "$\geq$." ■

So $R\{A, B\}$ exists duplicate calculations without properly process on the solution set. So the MSVs method is used to remove the solution set that will be recalculated.

If there exists z MSVs in total called Q_{MSVs}, the reliability of deployment plan is

$$R = \Pr\left\{\bigcup_{j=1}^{z} R_j\right\}. \quad (6)$$

Equation (6) is a joint probability problem of multivariate probabilities. To avoid repeated calculations, the principle of inclusion and exclusion is a feasible idea for solving this problem. The recursive sum disjoint product algorithm (RSDP) mentioned in [26] provides an efficient recursive way. In the case where all solutions $R_i = \Pr\{\cdot\}$ can be found, the reliability of Q_z which contains a total of z solutions could be answered by $R = RSDP(Q_z)$.

The reason we introduced MSV and RSDP in this study is computing complexity. It is observed that the existing approaches are not applicable to complex and dynamic scenarios, leading to a large impact on QoS when latency optimization-based deployment plan reaches a critical value of resource and cannot guarantee the operation of the service. Facing this dynamic challenge, to avoid the expensive communication cost associated with global awareness, the approach we come up with is to use distribution to disaggregate the continuous problem. This encounters another problem. After discretization, a solution space is obtained, containing many solutions however. There is a lot of repetitive computation in computing the reliability of this solution space, so we combined the MSV algorithm, based on the minimum cut idea, to find a small fraction of solutions among them that can represent whole solution space to avoid erroneous results from redundant computation. RSDP is then used for efficient solution of joint probabilities problem; otherwise, the result will be difficult to calculate when the number of solutions reaches a certain level.

C. REMR Algorithm

The REMR means a reliability evaluation method with MSV and RSDP algorithm, which returns the reliability of a distributed service in the DECN scenario. In REMR, latency is a main concern which would be affected by three quantitative factors: data packet storing and sending time, data transmission time and the calculation time of running tasks on edge node, especially while the last two are in dynamic scenarios. For a specific collaborative service, input data and time constraint are usually identified, the service reliability of a single and global deployment plan can be obtained by Algorithm 1. Line 25 gave the reliability R_k of a single deployment plan k, while global reliability could be calculated by line 27.

Algorithm 1 is mainly composed of four parts. Lines 4–10 are to find a solution set that meets requirements, and lines 11–24 are to find all MSVs in this solution set. Then, the RSDP method combined with MSVs is used to solve the reliability of the corresponding deployment plan in line 25. Finally, the reliability of the entire scheme is calculated by joint probability formulas.

Assuming that there are a total of m initial vector in the solution set, the complexity of the algorithm for finding MSVs is about $O(m^2)$. If there are a total of n MSVs for a deployment plan, the complexity of the RSDP method would be $O(n^3)$. In the last step of the whole algorithm, joint probability only has $O(k)$ complexity if there exist k plans in a DECN.

IV. Experiments

We first give an illustrative example of REMR in Section IV-A. Then, we compare the proposed REMR method

Algorithm 1 Multideployment Plans Reliability

Input:
Initial $DECN : S, I, X, Y, P$; Input $input\_data$, T;
Solve: $Q_{MSVs}, RSDP(Q_{MSVs})$;
Output:
R: reliability of whole DECN;
$t = \sum leadtime + transtime + computingtime$;
Suppose there are K deployment plans;
for $k = 1 : K$ **do**
  Create a traversal table Q_{init} with N vectors for X,Y;
  Create two empty collections Q_z, Q_{MSVs}
  for $n = 1 : N$ in Q_{init} **do**
    if $t\_n \leq T$ **then**
      $Q_z = Q_z \cup Vector : n$
    end if
  end for
  Suppose there are Z solutions in Q_z, L MSVs in Q_{MSVs};
  for $z = 1 : Z$ in Q_z **do**
    for $l = 1 : L$ in Q_{MSVs} **do**
      if $MSV : l \geq Solution : z$ **then**
        Remove $MSV : l$ from Q_{MSVs};
      end if
    end for
    for $l = 1 : L$ in Q_{MSVs} **do**
      if $Solution : z \geq MSV : l$ **then**
        Break and next z;
      end if
    end for
    $Q_{MSVs} = Q_{MSVs} \cup solution : z$
  end for
  $R_k = RSDP(Q_{MSVs})$;
end for
$R = \bigcup_{k=1}^{K} R_k$;

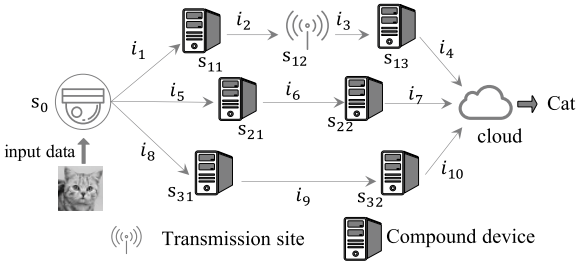


Fig. 6. Illustrative DECN example.

and prior work [25] with model partition application in Section IV-B. Also, we embed a real dynamic performance trace data set, filtered from Goolge's cluster [27], into this example in Section IV-C. To make REMR more reliable, we also propose a Monte Carlo simulation way in Section IV-D. Simply put, when the solution space is huge, Monte Carlo simulation is another practical method to compute the reliability of a collaborative service in DECN.

A. Illustrative Example

Fig. 6 shows the example used in the experiments, which includes three deployment plans. Each plan in this example contains two intermediate ESs running a collaborative task. We use it to explain the REMR procedures.

To help the readers touch the reason of why we select this example, our explanations are as follows. During the local execution of task, that the subtask amount a task will be divided

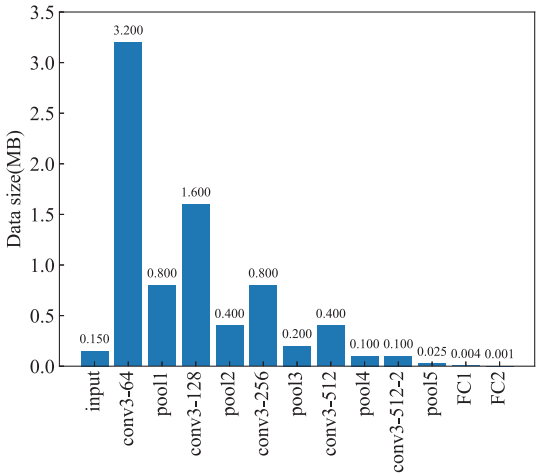


Fig. 7. Output data size of different layers.

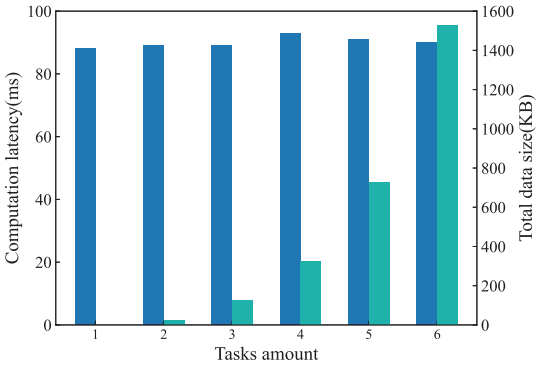


Fig. 8. Calculation latency and transmission data size when the number of subtasks is different.

into would not be determined by the number of nodes on actual link, but by the algorithm strategy. However, when there are too many subtasks, intermediate data needs to transmit is also increasing, making the total transmission latency increases. To understand, we implement an image recognition model partition algorithm, aiming to run a part of model on different nodes. In this example, the partitioning strategy is consistent, i.e., the partitions are fixed. Fig. 7 shows the output data size of each layer. To simplify the calculating procedures, our first assumption is that all computing nodes have the same computing resources. Under this assumption, in Fig. 8 we can see that the total computation latency of the task nearly does not change as expected, because the total cost of computation does not change. But data that need to be transferred among different partitions increase obviously, which results in an increase in transmission latency. The second assumption is more realistic which would limit the computing resource of the front-end device. In this experiment, the computing resource of the front-end device is fixed to half of the follow-up nodes, by reducing the main frequency of CPU to half while other parameters are consistent. Then, the total computation latency increases slowly as the number of subtasks increases as shown in Fig. 9. Moreover, only the transmission time is calculated in the experiment, the time for sending and receiving data in network transmission also needs to be considered. The size of

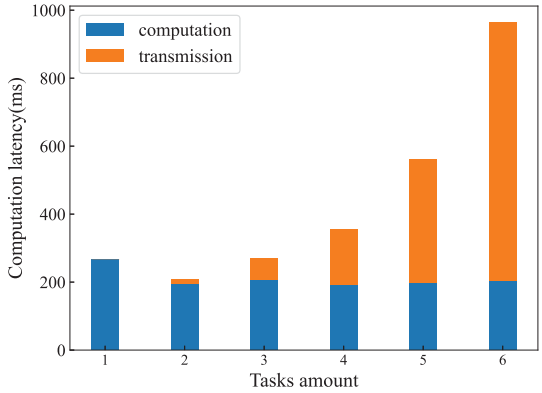


Fig. 9. Calculation latency and transmission latency when the number of subtasks is different.

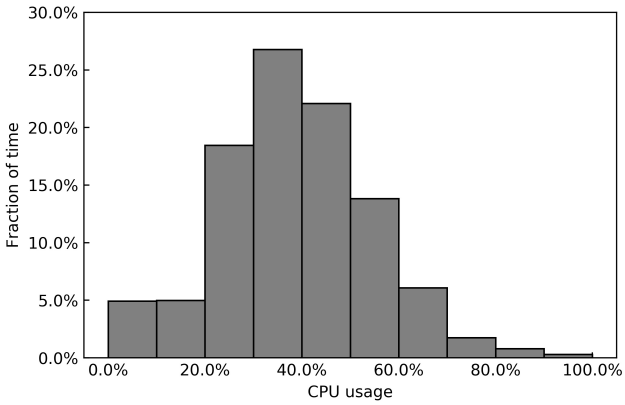


Fig. 10. CPU usage distribution with time in Alibaba servers cluster.

TABLE II
BANDWIDTH CAPACITY PROBABILITY

branch	lead time	capacity					
		0	1	2	3	4	5
i_1	2	0.04	0.01	0	0.02	0	0.93
i_2	1	0.03	0.01	0.02	0.94	0	0
i_3	1	0.02	0.01	0	0.04	0.93	0
i_4	1	0.03	0.01	0	0.06	0	0.9
i_5	2	0.02	0.01	0.01	0	0.96	0
i_6	1	0.04	0.01	0.95	0	0	0
i_7	1	0.03	0.01	0	0.02	0.94	0
i_8	1	0.03	0.01	0.01	0.95	0	0
i_9	2	0.03	0	0.01	0.03	0.93	0
i_{10}	1	0.02	0	0.01	0.02	0.95	0

TABLE III
AVAILABLE NODES CAPACITY PROBABILITY

nodes	O/I	capacity						
		0	1	2	3	4	5	6
s_0	0.8	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{11}	1.2	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{12}	1.0	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{13}	$\rightarrow 0$	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{21}	1.2	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{22}	$\rightarrow 0$	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{31}	1.2	0.0	0.01	0.09	0.26	0.37	0.20	0.07
s_{32}	$\rightarrow 0$	0.0	0.01	0.09	0.26	0.37	0.20	0.07

TABLE IV
CHANGES OF DATA SIZE IN DEPLOYMENT PLANS

	C_0	C_1	C_2	C_3	C_4
DP1	15	12	14.4	14.4	0.01
DP2	15	12	14.4	0.01	
DP3	15	12	14.4	0.01	

transmission data is increasing as the amount of time required for repetitive sending and receiving operations increasing, indicating that the number of subtasks needs to be limited. Thus, in this study, we limit the number of collaborative tasks run on edge device, ESs, and the cloud no more than 4. Based on this example, three deployment plans are designed. Node s_0 means an edge device where the task data from and the cloud is the terminal for deployment plan. The compound nodes will play a role in transmitting intermediate data and computing collaborative task simultaneously, while the transit nodes, such as s_{12}, only account for transmission data.

We use a synthetic data set to show REMR procedures. Inspired by bandwidth distribution [15], we adopt a similar way to characterize computing resources, considering that discrete probabilities could easily and quickly obtain the availability of computing resources. Table II depicts the probability distribution of available bandwidth on each branch and Table III depicts the probability distribution of available compute resources on each server and each device. The capacity of bandwidth and compute resources is set to be a multiple of integer for simplicity. The bandwidth distribution based on historical data is derived from [28], used to describe the probability of each branch's available bandwidth. The distribution of computing resources (Table III) comes from the analysis of the Alibaba servers data set [29]. While this data set includes CPU usage about server clusters as shown in Fig. 10, we convert it into available CPU resources, i.e., CPU frequencies [30]. Table III represents the ratio of output data calculated by the server to the size of the input data received by the server, i.e., O/I, which is used to calculate the change degree in deployment plan.

Assuming that $C_0 = 15$ units of data would be input of three deployment plans with time constraint $T = 25$ s, the reliability of the task could be presented by $R = \{R_a, R_b, R_c\}$, where $a = \{s_0, s_{12}, s_{13}, s_{11}, \text{cloud}\}$, $b = \{s_0, s_{21}, s_{22}, \text{cloud}\}$, and $c = \{s_0, s_{31}, s_{32}, \text{cloud}\}$. To explain this indicator output/input with more detail, Table IV is drawn where the DP represented different deployment lines. They have one initial edge device in common. Therefore, the initial input C_0 is all 15 units. Taking DP 1 as an example, the indicator of the device is 0.8, so the amount of data send to the first node C_1 of DP 1 becomes $15 \times 0.8 = 12$. Here, the indicator is 1.2, so the amount of data C_2 becomes 14.4. Since s_{12} in DP 1 is a transmission node without computing, output C_3 is totally the same as input, also 14.4, the node S_{13} would finish the final computing task and send a result to the cloud. Usually, the output here would be very small, so the indicator is set to be 0.01, to present a tiny computing task.

This section first takes the processes of computing R_a as an example to describe a specific calculation process. According to (6), all MSVs need to be found. Combining (3), given initial input data size C_0 and time threshold T, the implicit enumeration method is utilized in

TABLE V
MSVs FOR A DEPLOYMENT PLAN

No	$X(x_1, x_2, x_3, x_4)$	$Y(y_1, y_2, y_3)$
1	(2, 3, 4, 1)	(5, 1, 6)
2	(2, 3, 4, 1)	(6, 1, 5)
3	(3, 3, 3, 1)	(4, 1, 5)
4	(3, 3, 3, 1)	(5, 1, 4)
5	(3, 3, 4, 1)	(4, 1, 4)
.	.	.
.	.	.
.	.	.
14	(4, 3, 4, 1)	(4, 1, 3)

TABLE VI
NUMERICAL EXPERIMENTS

	C=14	C=15	C=16
T=20	0.94930	0.87567	0.75312
T=21	0.97163	0.94656	0.82634
T=22	0.98670	0.97449	0.91367
T=24	0.99606	0.99438	0.98362
T=25	0.99729	0.99653	0.99207

the above-mentioned DECN to gain all solutions for the inequality

$$t = 4 + \sum_{i=1}^{4} \left\lceil \frac{C_0 \prod_{k=0}^{i-1} \delta_k}{x_i} \right\rceil + \sum_{i=0}^{4} \frac{C_0 \prod_{k=0}^{i} \delta_k}{y_i} \leq 25s$$

without thinking of repetition. The changes in the amount of input data after node processing are shown in Table IV. C_0 represents the initial input, and the amount of data would always change after going through different nodes of the deployment plan, similarly, C_i represents the amount of data output after the ith node, and the rest is the same. Implicit enumeration refers to the removal of attempts to impossible answers on the basis of enumeration. First 2460 solutions were found in solving the inequality. After filtering the solution set, 14 MSVs remain, while part of them were listed in Table V.

Then, the reliability of deployment plan a can be obtained by the improved RSDP algorithm. The reliability of $R_a = 0.82344$. Similarly, the reliability of $R_b = 0.87679$ and $R_c = 0.90955$. Finally, the global reliability of $R = R_a \bigcup R_b \bigcup R_c$. Considering that the three reliabilities are redundant and independent, thus

$$R = R_a + R_b * (1 - R_a) + R_c * (1 - R_a) * (1 - R_b). \quad (7)$$

The obtained reliability of the example is 0.99653, which presents the probability of finishing the service perfectly. More similar experiments based on the same DECN were carried out and the results were recorded in Table VI.

B. Comparison of Prior Work and REMR

REMR was combined with specific application and compared with existing prior work in this section.

1) Description of Prior Work and REMR: We combined AI services and model partition to implement inference process for image classification task in the edge. The experiments compared the service execution in a dynamic scenario with lowest latency deployment plan and REMR deployment plan. Specifically, we selected a partition point in the classification model that satisfies the requirement of the lowest execution latency when dynamics are not considered. Meanwhile, another partition point was chosen with the same amount of current data as the previous partition point but at a different position. The same data size ensures that time consumed by two partition points in data transmission will be similar.

TABLE VII
SIMPLIFIED RESOURCE DISTRIBUTION

	100%	75%	50%	25%	0%
S1	0.1	0.2	0.4	0.2	0.1
S2	0.1	0.2	0.4	0.2	0.1

TABLE VIII
SOLUTIONS SET FOR TWO PARTITION POINTS

No	$S1(left, right)$	$S2(left, right)$
1	(4, 4)	(4, 4)
2	(4, 3)	(4, 3)
3	(4, 2)	(4, 2)
4	(3, 4)	(4, 1)
5	(3, 3)	(3, 4)
6	(3, 2)	(3, 3)
7	(2, 4)	(3, 2)
8	(2, 3)	(3, 1)
9	(2, 2)	(2, 4)
10	(1, 4)	(2, 3)
11	(1, 3)	(2, 2)
12		(1, 4)
13		(1, 3)

2) Data Set Used for Comparison: We assume that the distribution of available resources of two computing nodes $s1$, $s2$ obeys the distribution in the Alibaba data set, and simplify it as shown in Table VII for computational convenience. We also used linear regression for predicting the running time of different layers of the AI model, for which the input data include CPU frequency, memory, and input data size. Training data were obtained from repeated experiments based on docker. It is specified that the execution time of this inference task should not exceed 4 s, i.e., $T_{s1} + T_{s2} < 4$ s. Since the data size is the same, we removed it from calculation. Therefore, it is the requirement that the total computation time of both nodes does not exceed 4 s, which is considered to have an acceptable quality of service.

The solutions set was got shown in Table VIII, utilizing the prediction model to obtain the sum of computation time of two nodes in different resource states. By combining the probability distributions, we can obtain the magnitude of the reliability of two nodes under all distributions. $R_{s1} = 0.54$ and $R_{s2} = 0.61$, which is the reliability of two partition positions, and indicates the predictive result of their QoS in a dynamic environment.

To verify this result, we simulated the nodes with different resource states using docker and obtained the results shown in Fig. 11. The four subgraphs correspond to the operational state of node S1, and each subgraph depicts the relationship between computational latency and available resources of the S2 node in the current S1 node state. And the standard deviation of the data in each subplot is calculated and plotted as Fig. 12. From Fig. 11, we can see that the traditional method

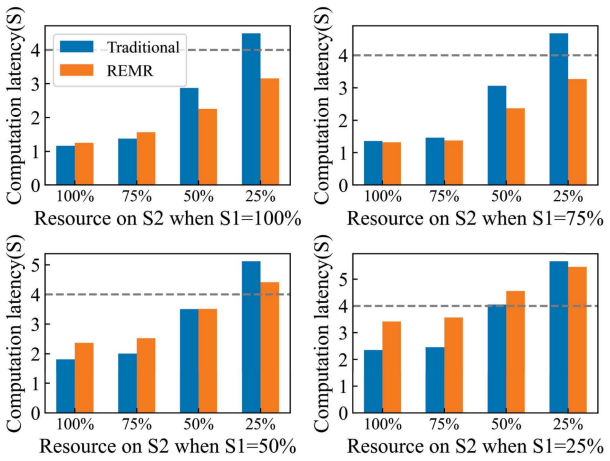


Fig. 11. Computation latency of two plans under different resource states. Here, S1 = 100% means that its resource is totally available.

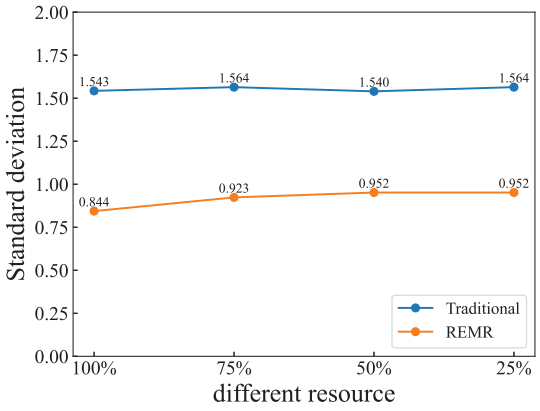


Fig. 12. Standard deviation of experimental result of two schemes.

has the highest gain when the resources of both nodes are in a relatively sufficient state, which is reflected in its low latency. But when we consider the dynamic factor, i.e., the resources drop to a certain level, the time latency of the split point chosen by the traditional method grows faster, while the REMR method grows slower accordingly. This can also be seen from the variance curves in Fig. 12. The variance of the time latency data of the traditional method is significantly higher than that of the REMR method, which can be explained by the degree of matching the computational workload with the available computational resources. After the nodes and models are deployed, the selection of the segmentation points does not change again. However, because of the existence of resource dynamics, we can allocate the computational load to the nodes with more sufficient resources, so that the nodes with more current computational power can take up more computational load which can effectively improve the time efficiency.

C. Experiments on Google Cluster Trace Data Set

The Google cluster data trace represents 29 days' worth of Borg cell information from May 2011, on a cluster of about 12.5k machines. According to the analysis of the data set in

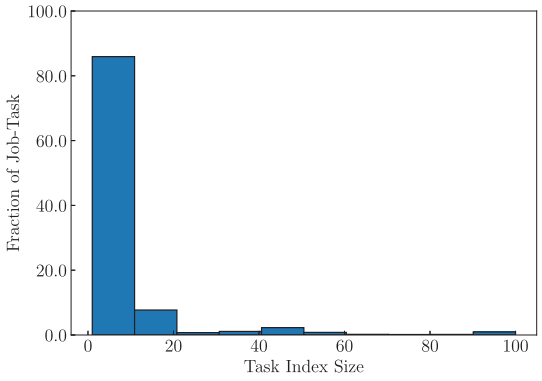


Fig. 13. Distribution of subtasks number in part of the Google cluster trace data set.

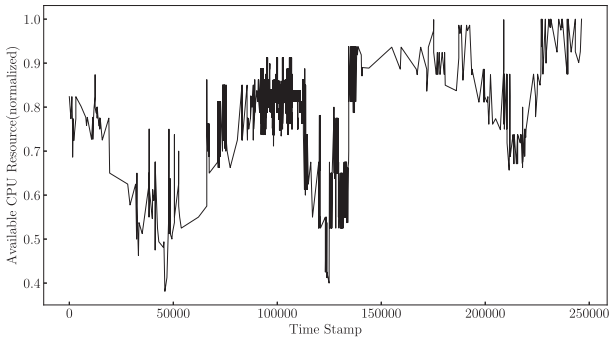


Fig. 14. Curve of available CPU resource with time in the Google cluster trace data set.

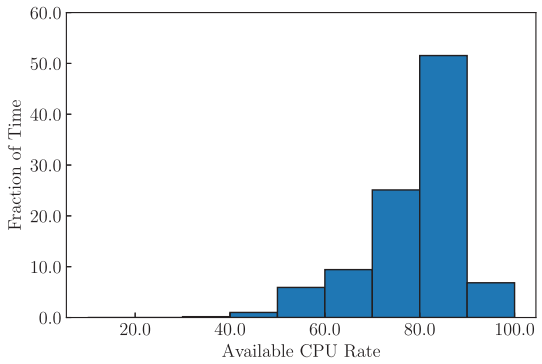


Fig. 15. Available CPU resource distribution in part of the Google cluster trace data set.

this study, it was found that most of the services were decomposed into several subservices. The distribution of subservices number can be statistically obtained as shown in Fig. 13 by counting the frequency of event type 0 in each job. Obviously, most of the services are decomposed into no more than ten subservices. This experiment selected a service with three subtasks in data set for evaluation. These three subservices run on different machines based on the time sequence, so this study monitored the available resources on these three machines based on three days of data. Fig. 14 showed the situation of one machine of three, and it was translated into a probability distribution shown in Fig. 15. Since the cluster data set has

Algorithm 2 Monte Carlo Simulation

Input:
Initial $DECN : S, I, X, Y, P$;
Input C_0, T;
Output:
R: Probability of 4000 experiments;
$t = \sum leadtime + transtime + computingtime$;
for $k = 1 : 4000$ in a whole deployment plan **do**
  Create C_0 at the first node
  Once need transmission
  choose a x_i according probability in Tab.2;
  Once need computing
  choose a y_i according probability in Tab.3;
  .....
  until finish the whole deployment
  Assuming cost t seconds in total
  if $t \leq T$ **then**
    $Success+ = 1$
  else
    $Defeat+ = 1$
  end if
end for
$R = \frac{Success}{Success+Defeat}$;

no bandwidth distribution data, the bandwidth data set used in the last experiment was adopted again. The reliability is calculated to be 0.98657, while the input data size is 14 and the time constraint is 20. Although possibly affected by bandwidth, the success rate of its task completion is still very high which could be explained by the high spare rate of machines.

D. EasiEI and Monte Carlo Simulation

Modeling the formal reliability calculating procedures is hard especially when the collaborative tasks are too many. Thus, we also propose and implement a Monte Carlo simulation for calculating the reliability of a collaborative service.

The Monte Carlo simulation is a computational method based on random numbers, in which discrete random sampling of input is performed by computer simulation [20]. The probability distribution of input would be converted into the possibility of being selected to perform specific time calculations of output, and the actual distribution of output can be predicted under the known probability distribution of input. In the simulation, experiments are carried out according to Algorithm 2.

Nodes on the simulation platform would be set according to the given link. When a task starts, a value would be selected for each node and each branch as its actual ability according to the probability distribution of bandwidth and computing resource. Then, the probability of completing the task within the time T would be calculated as reliability after repeating a certain number of times.

We implement Algorithm 2 on our proposed open-source simulator EasiEI [12]. EasiEI can simulate dynamic edge computing environment and allows us to make the node resource and network bandwidth changeable. The nodes and links are first reproduced in EasiEI according to the deployment plan, then the bandwidth distribution of each link and computing resource distribution of each node are assigned to it. In addition to the predefined distribution, EasiEI also supports replaying a real-world dynamic workload data set. By this, EasiEI has a more powerful and reasonable ability to model the dynamic production environment. Since distributions are derived from historical data, discrete probabilities are used as confidence level. A concrete computing resource is chosen according to the magnitude of probability to calculate the time performing task.

TABLE IX
EASIEI VERSUS REMR IN NUMERICAL EXPERIMENTS

NS3	C=14	C=15	C=16
T=21	0.85650	0.76025	0.67125
T=22	0.91300	0.86800	0.82575
T=24	0.97600	0.94875	0.94000
T=25	0.98050	0.97550	0.97100
REMR	C=14	C=15	C=16
T=21	0.86887	0.80861	0.76274
T=22	0.87471	0.83771	0.81734
T=24	0.88253	0.87199	0.86887
T=25	0.89346	0.87709	0.87471

The numerical experiments in Table IX were reproduced by Monte Carlo and REMR, which are based on the second deployment plan in Fig. 6. It is clear that the results of the Monte Carlo simulation and REMR are relatively consistent. Since that simulation of Monte Carlo utilizes random numbers, deviation exists.

V. CONCLUSION

Due to the lack of consideration of dynamic characteristics in edge computing scenarios, the deployed collaborative services face the problem of QoS degradation and invalidation caused by latency jitter. We propose a new reliability evaluation method REMR in a dynamic edge computing network scenario under a time constraint. Through the MSVs and RSDP algorithm, we design and implement the reliability evaluation method based on the solution set. Then, we use an example to explain how REMR works. And we utilize REMR as the basis for model partition. Compared with the model partition with lowest latency, we find that REMR has better resiliency for dynamic resource scenarios. Finally, we take EasiEI as an experiment simulation, which is in line with the proposed method almost. The main contribution of this study is the consideration of dynamic factors in edge scenarios, the combination of MSV algorithm and RSDP algorithm to solve the reliability calculation and the verification with public data set on simulation. With the method, experiments show that REMR is convenient to tell how reliable a distributed collaborative service is and how to make service more reliable.

Some directions worthy of further research in the future are listed as follows. The first RSDP Algorithm could be improved on time as shown in [31]. The use of distributed information has advantages, but time information is lost. In future work, we are going to combine time-series information and optimize the expression of available resources. Current research only considered serial collaborative services, but there are many applications that allow task parallelism in practical applications [32]. So we need to improve the current evaluation method to be suitable for parallel computing scenarios

too. More importantly, hoping to provide a framework, so that the proposed reliability evaluation method could be used to segment a distributed service of the DNN model rather than manual assignments.

REFERENCES

[1] M. Shirer and C. MacGillivray, *The Growth in Connected IoT Devices Is Expected to Generate 79.4 zb of Data in 2025, According to a New IDC Forecast*, IDC, Needham, MA, USA, 2019.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] C. Nguyen, A. Mehta, C. Klein, and E. Elmroth, "Why cloud applications are not ready for the edge (YET)," in *Proc. 4th ACM/IEEE Symp. Edge Comput. (SEC)*, 2019, pp. 250–263.

[4] J. King and A. I. Awad, "A distributed security mechanism for resource-constrained IoT devices," *Informatica*, vol. 40, no. 1, pp. 1–12, 2016.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[6] N. Hu, Z. Tian, X. Du, and M. Guizani, "An energy-efficient in-network computing paradigm for 6G," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 4, pp. 1722–1733, Dec. 2021.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[8] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive resource efficient microservice deployment in cloud-edge continuum," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1825–1840, Aug. 2022.

[9] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Resource management for latency-sensitive IoT applications with satisfiability," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2982–2993, Sep./Oct. 2022.

[10] Z. Fan, W. Yang, F. Wu, J. Cao, and W. Shi, "Serving at the edge: An edge computing service architecture based on ICN," *ACM Trans. Internet Technol.*, vol. 22, no. 1, pp. 1–27, 2021.

[11] R. Milocco, P. Minet, E. Renault, and S. Boumerdassi, "Evaluating the upper bound of energy cost saving by proactive data center management," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1527–1541, Sep. 2020.

[12] X. Su. "EASIEI." Accessed: Mar. 22, 2022. [Online]. Available: https://gitlab.com/Mirrola/ns-3-dev

[13] A. Hoyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*. Hoboken, NJ, USA: Wiley, 2009.

[14] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*. New York, NY, USA: CRC Press, 2016.

[15] C.-F. Huang, D.-H. Huang, and Y.-K. Lin, "Network reliability evaluation for a distributed network with edge computing," *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106492.

[16] S. Chakraborty, N. K. Goyal, S. Mahapatra, and S. Soh, "Minimal path-based reliability model for wireless sensor networks with Multistate nodes," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 382–400, Mar. 2020.

[17] O. Kabadurmus and A. E. Smith, "Evaluating reliability/survivability of capacitated wireless networks," *IEEE Trans. Rel.*, vol. 67, no. 1, pp. 26–40, Mar. 2018.

[18] C.-F. Huang, D.-H. Huang, and Y.-K. Lin, "Reliability evaluation of a cloud–fog computing network considering transmission mechanisms," *IEEE Trans. Rel.*, vol. 71, no. 3, pp. 1355–1367, Sep. 2022.

[19] R. C. Loh, S. Soh, and M. Lazarescu, "Addressing the most reliable edge-disjoint paths with a delay constraint," *IEEE Trans. Rel.*, vol. 60, no. 1, pp. 88–93, Mar. 2011.

[20] J. E. Ramirez-Marquez and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Rel. Eng. Syst. Safety*, vol. 87, no. 2, pp. 253–264, 2005.

[21] C.-H. Huang, D.-H. Huang, and Y.-K. Lin, "A novel approach to predict network reliability for multistate networks by a deep neural network," *Qual. Technol. Quant. Manag.*, vol. 19, no. 3, pp. 362–378, 2021.

[22] M. L. Gámiz, F. J. Navas-Gómez, and R. Raya-Miranda, "A machine learning algorithm for reliability analysis," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 535–546, Jun. 2021.

[23] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, and D. Z. Du, "Reliability-aware offloading and allocation in multilevel edge computing system," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 200–211, Mar. 2021.

[24] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Oct. 2019.

[25] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Architect. News*, vol. 45, no. 1, pp. 615–629, 2017.

[26] M. J. Zuo, Z. Tian, and H. Huang, "An efficient method for reliability evaluation of multistate networks given all minimal path vectors," *IIE Trans.*, vol. 39, no. 8, pp. 811–817, 2007.

[27] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google Inc., Mountain View, CA, USA, White Paper, 2011.

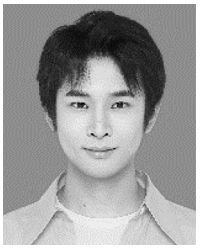
[28] C.-F. Huang, Y.-K. Lin, and L. C.-L. Yeng, "Routing scheme of a multi-state computer network employing a retransmission mechanism within a time threshold," *Inf. Sci.*, vols. 340–341, pp. 321–336, May 2016.

[29] S. Luo et al., "Characterizing microservice dependency and performance: Alibaba trace analysis," in *Proc. ACM Symp. Cloud Comput.*, 2021, pp. 412–426.

[30] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in *Proc. Int. Symp. Qual. Service*, 2019, pp. 1–10.

[31] G. Bai, M. J. Zuo, and Z. Tian, "Ordering heuristics for reliability evaluation of multistate networks," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1015–1023, Sep. 2015.

[32] J. Qi and R. Wang, "R2: A distributed remote function execution mechanism with built-in metadata," *IEEE/ACM Trans. Netw.*, early access, Aug. 19, 2022, doi: 10.1109/TNET.2022.3198467.



Liang Chen received the B.S. degree from the Advanced Engineering School, University of Science and Technology Beijing, Beijing, China, in 2020, where he is currently pursuing the M.S. degree in computer technology, under the guidance of Prof. R. Wang.

His research interests include edge computing, reliability, and machine learning.

Jianpeng Qi received the B.S. and M.S. degrees in computer software and theory from Yantai University, Yantai, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree in computer science and technology with the University of Science and Technology Beijing, Beijing, China, under the guidance of Prof. R. Wang.

His research interests include resource management, reliability, and edge intelligence.

Xiao Su received the B.S. degree in information and computational science from Beijing Information Science and Technology University, Beijing, China, in 2020. He is currently pursuing the M.S. degree in computer technology with the University of Science and Technology Beijing, Beijing, under the guidance of Prof. R. Wang.

His research interests include edge computing, simulation, and machine learning.

Rui Wang (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligent systems from Northwestern Polytechnical University, Xi'an, China, in 2007.

He is currently a Professor with the Department of Computer Science and Technology, School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China. His research interests include edge intelligence, mobile and ubiquitous computing, and distributed systems.