Recovery Mechanism of Large-scale Damaged Edge Computing Network in Industrial Internet of Things

TIAN Hui[1], WU Hao[1], TIAN Yang[1], REN Jianyang[1], CUI Yajuan[1], AI Wen-bao[2], YUAN Jianhua[2]
1. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. School of Sciences, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Given the limited resources at the early stage of recovery, a failure recovery mechanism of edge computing networks considering both computational demands and repair costs was proposed, which intends to tackle the problem of the high probability of large-scale cascading failure caused by the interdependence between edge computing networks and other subnetworks in industrial Internet of Things (IIoT). Considering the network structure (topology and link capacities) and system dynamics (computational demands), a joint link recovery selection and computation migration optimization problem was formulated under the conservation of node computing requirements. By leveraging the Benders decomposition algorithm, the NP-hard problem was transformed into a main problem and a subproblem, which were interdependent and could be solved in polynomial time through the approximation of cutting planes. A local branching method was further introduced to guarantee the non-increasing nature of the Benders upper bound, thus accelerating the convergence of Benders decomposition. Simulation results demonstrate that the proposed mechanism outperforms the conventional topology-based recovery mechanism in system cost, and can perform well in multiple scenarios.

Keywords: industrial Internet of things, edge computing, network recovery, Benders decomposition, local branching

1 Introduction

Industrial Internet of Things (IIoT) is envisioned as an important booster for the intelligent transformation of global industrial systems. With hundreds of millions of seamlessly deployed sensors, collectors and controllers, IIoT participates in the full cycle of simulation, prediction and control of manufacturing processes [1]. As the "brain" of IIoT, edge computing networks provide more sufficient computing capability for resource-limited wireless devices, reduce processing and transmission latency, and lay a solid foundation for enterprise applications such as digital twin [2] and virtual reality [3]. Meanwhile, cables between edge computing nodes facilitate the migration of computing tasks, effectively alleviating the problem of unevenly distributed computing resources caused by the temporary- and spatial- fluctuations of computing demands in IIoT.

The stability of edge computing networks is critical for efficient operations of IIoT. Once the "brain" is damaged, IIoT will lose control of "limbs" (e.g., supply chain monitoring, data visualization analysis). However, the stability of the edge computing network in IIoT faces both internal and external challenges: 1) edge computing networks are coupled with multiple sub-networks in IIoT such as power grids and control systems, forming an interdependent network that is highly vulnerable [4]; 2) unpredictable disruptions such as natural disasters and human attacks test the robustness of edge computing networks twenty-four seven [5]. To address these challenges, on the one hand, edge computing networks can enhance its reliability to adaptively cope with various network fluctuations and prevent possible network failures; on the other hand, it is also of vital importance to explore emergent recovery mechanisms after network damage, such that the system performance can be restored to pre-damage level as soon as possible.

Although the design of recovery mechanisms is crucial for network sustainability and stability, there are limited efforts that are targeting IIoT. Fortunately, given the similarity in network topology and system dynamics, some existing network recovery strategies can still provide some references for the design of recovery mechanisms for edge computing networks. Currently, recovery studies mainly focus on local damage scenarios where only part of the systems are affected. In reference [6], Ayoubi *et al* formulated the repair of a single node or link as an integer linear programming problem, and proposed a data migration-aware recovery model to achieve well balance between service disruption rate and repair cost. Furthermore, the authors of reference [7-8] considered the problem of impaired connectivity in a multi-node failure scenario, and proposed that the connectivity of the network could be ensured by using users as relays through a device-to-device (D2D) manner [7] or by using mobile access nodes [8]. Unlike the

above-mentioned works, the authors of reference [9] considered the persistent network threat under attack, and transformed the network dynamic recovery problem into a differential game theory problem. The problem was then solved with necessary conditions for Nash equilibrium and a design of competitive strategy profile. However, the above studies for local network recovery often ignore the dynamic characteristics among global nodes/links (e.g., flow migration) and practical constraints (e.g., cable layout cannot be changed), which makes them hard to be extended to more likely large-scale network damage scenarios in IIoT.

After large-scale network damage, recovery resources (e.g., repair crews and replaceable devices) are often limited at an early stage. How to effectively balance the limited system repair resources and the need for system performance recovery is an urgent problem for IIoT. Existing research mainly focuses on the analysis of network topology. Reference [10] argued that high-degree nodes have a more important role in network connectivity and need to be repaired with priority. Similarly, study [11] considered that links with large betweenness centrality should be repaired first in case of link damage. Through real-world big data analysis, the authors of reference [12] found that weak nodes, i.e., low-degree nodes that are connected to multiple high-degree nodes, outrank the influence of other nodes in network connectivity and their repair priority should be the highest. However, all these works based on network connectivity that try to enlarge the maximal connected subgraph are static network structure analyses, ignoring the dynamic characteristics of the network. Large-scale network recovery in real environments will not form a maximal connected graph until multiple mutually independent subgraphs have been generated [13]. Therefore, engineering analysis needs to consider more device details and real transmission dynamics [14]. This type of problem is also known as the network design problem (NDP). Given network dynamics features, the complexity of NDP is high, since it is at least a NP-complete problem. Using dynamic programming algorithms to solve the problem will cause the problem of *curse of dimensionality* [15]. To address the problem, a more solvable heuristic algorithm was proposed in reference [16]. Nevertheless, it cannot provide satisfactory performance guarantees given its large variance results from the lack of macro network analysis. Analogously, both simulated annealing algorithms [17] and genetic algorithms [18] face the same dilemma as heuristics and are easily trapped at local optimal solutions. Although hill climbing algorithms [19] and gradient descent algorithms [20] can easily jump out of the local optimal point and greatly reduce the computational complexity of NDP, the optimality of solutions still cannot be ensured.

Given the above dilemmas, we propose a recovery mechanism for large-scale damaged edge computing networks in IIoT. Consider network vulnerability caused by interdependence among IIoT sub-networks, we explore structural characteristics (topology and link capacity) and system dynamics (computation demands of nodes) of large-scale damaged edge computing networks. Distinct from existing studies, a joint optimization problem of link repair selection and computation migration is designed to balance computation demands and repair costs at the early stage of network recovery. The key contributions are summarized as follows.

(1) We propose a network recovery mechanism for large-scale damaged edge computing networks. Network structure and dynamics are jointly considered to provide a repair strategy and resource scheduling framework for early-stage network recovery.

(2) Based on Benders decomposition, the original complicated mixed-integer programming (MIP) problem for recovery decisions is transformed into a master problem and a subproblem. The optimal system recovery strategies can be obtained by solving the two problems iteratively. To solve the subproblem, a virtual source node and a virtual destination node are designed, which transfer the problem into a maximum flow problem.

(3) We develop an accelerated Benders decomposition algorithm based on local branching. With the help of Hamming distance, the algorithm can reduce the search space and improve convergence.

Simulations reveal that the proposed recovery mechanism has better convergence speed and lower system costs. Compared with existing schemes, the proposed approach outperforms benchmarks in multiple network topologies, proving its scalability and adaptability for IIoT.

2 System Model

We consider an edge network in IIoT that consists of N edge nodes, which are indexed by $\forall i \in \mathcal{N} = \{1, 2, \cdots, N\}$. Edge nodes are connected by $|E|$ cable links, where E is the set of links in the system. Cable links are often reliable and only need a small amount of complexity for channel coding, thus are considered error-free [21]. The notations used in this article are summarized in Table 1.

Note that edge computing networks in realistic IIoT is often hybrid link transmission networks consisting of wired links and wireless links. Given that wireless links are hard to impair and their repair costs

are relatively small, only recovery of wired links are considered. Nevertheless, if the deep fading of short-term wireless links is neglected, wireless links can be seen as non-damageable wired links with constant capacity (without considering the reallocation of bandwidths). Therefore, the hybrid transmission network can be simplified as an equivalent wired network, and the proposed mechanism is still applicable.

Table 1: Summary of notations

Notation	Description
N	Number of edge nodes
E	Set of links
E^0	Set of damaged links
E^1	Set of undamaged links
q	Percentage of damaged links
c_{ij}	Repair cost for link ij
c_i	Price for discarding one bit data of node i
p_i	Computed data volume at node i
$\overline{p_i}$	Computation capacity of node i
r_i	Processing demand of node i
d_i	Discarded data of node i
f_{ij}	Data migrated from node i to node j
$\overline{f_{ij}}$	Capacity of link ij
e_{ij}	Decision variable for repairing link ij
t	Number of iterations of Benders decomposition
θ^t	Optimality of objective during t -th iteration of the subproblem
Δ^t	Threshold for t -th iteration of the left-branch problem

Edge computing nodes are equipped with complicated self-protection mechanisms (e.g., overheating protection) and are hard to disrupt. Wired links, on the contrary, are often the trigger for large-scale damage. Let q denote the percentage of damaged links in the system, E^0 collect the set of those links, and $E^1=E\setminus E^0$ be the set of workable links. The damage of links leads to the re-distribution of balanced data migration between nodes. What's worse, it may result in *computing islands* (nodes that are separated from other nodes, e.g., node 1 in Figure 1), breaking the match between computing capacity and processing requirements. Restricted by the limited resources (such as number of repair crews and replaceable spares) at the early stage of recovery, restoring all damaged links simultaneously is impossible. To recover network functions as soon as possible, a set of damaged links can be repaired first, denoted by $E^r(E^r\subseteq E^0)$, according to computing demand distributions. For any link $ij\in E^0$, the cost c_{ij} for maintain, overhaul, and replace is different, given different positions and scale of damage.

Let r_i denote the processing demand of edge nod $i\in\mathcal{N}$, p_i be the corresponding amount of computed data. Nota that r_i is the total volume of data that arrived at node i at the early stage of recovery. For a given scenario, it is a constant that will not change with time, the value of which can be estimated by historical computing demands. The processed data volume of edge node i satisfies

$$0\le p_i\le \overline{p_i},\forall i\in\mathcal{N}, \tag{1}$$

where $\overline{p_i}$ is the computation capacity of node i.

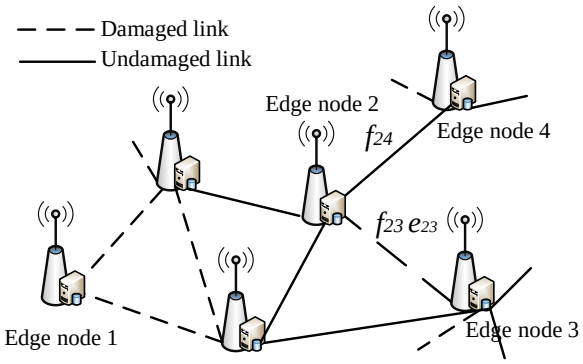

Figure 1: Topology illustration of a damaged edge computing network

Let f_{ij} be the amount of data migrated between node i and node $j\in\mathcal{N}\setminus\{i\}$. When $f_{ij}>0$, data is migrated from node i to node j; Otherwise, data flow is from node j to node i. Given the link capacity $\overline{f_{ij}}$, data transformed on link ij follows

$$\left|f_{ij}\right|\le\overline{f_{ij}},\forall ij\in E. \tag{2}$$

For damaged links, the amount of data migrated is not only determined by the link capacity but also by the repair decision, i.e.,

$$\left|f_{ij}\right|\le\overline{f_{ij}}e_{ij},\forall ij\in E^0, \tag{3}$$

where

$$\mathrm{e}_{ij}\in\{0,1\},\forall ij\in E^0, \tag{4}$$

in which $\mathrm{e}_{ij}=1$ represents that link ij is given priority in repairing (i.e., $ij\in E^r$); $\mathrm{e}_{ij}=0$ means link ij has not been restored and the actual migrated data is $f_{ij}=0$. Besides, for any link ij, the symmetry of flows leads to

$$f_{ij}=-f_{ji},\forall ij\in E. \tag{5}$$

When links in set E^r are repaired, data processed at or migrated from node i satisfies

$$p_i+\sum_{ij\in E^0}f_{ij}+\sum_{ij\in E^1}f_{ij}+d_i=r_i,\ \ i\in\mathcal{N}, \tag{6}$$

where r_i is the computing demands of node i, and

$$d_i \geq 0, \forall i \in \mathcal{N} \tag{7}$$

represents the data discarded by node i due to backlog caused by the limited computation capacity of the edge node.

Repairing as many links as possible can reduce the amount of data discarded and improve system performance, but will result in large repair costs. Recovering a small number of links, however, may cause data backlogs to increase and the amount of data discarded to surge. For quantitative analysis, network performance is measured by data discarding prices. The larger the prices are, the worse the network performance is. To balance the link repair costs and network performance (data discarding prices) at the early stage of recovery, a minimization problem of system cost can be formulated as

$$P: \min_{\boldsymbol{e},\boldsymbol{d},\boldsymbol{f},\boldsymbol{p}} \phi = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in \mathcal{N}} c_i d_i$$
$$\text{s.t.} \quad (1)\text{-}(7)$$

where c_{ij} is the repair cost for link $ij \in E^0$, c_i denotes the price for discarding a unit of data at edge node $i \in \mathcal{N}$. Link repair decision vector $\boldsymbol{e} = (e_{ij}, \forall ij \in E^0)$, data discarding decision vector $\boldsymbol{d} = (d_i, \forall i \in \mathcal{N})$, data flow distribution vector $\boldsymbol{f} = (f_{ij}, \forall ij \in E)$, and local processed data vector $\boldsymbol{p} = (p_i, \forall i \in \mathcal{N})$ are four sets of variables to be optimized. In problem P, system cost consists of two parts: link repair costs and data discarding prices. To some extent, the link repair costs $\Sigma_{j \in E^0} c_{ij} e_{ij}$ and data discarding prices $\Sigma_{i \in \mathcal{N}} c_i d_i$ are conflicting. The increase of one will lead to the decrease of the other. The combination of them can effectively balance the impacts of both in recovery. When link repair cost c_{ij} is large, the significance of data is relatively small. As a result, the system would prefer to recover a small number of links and drop the data in backlogs. Otherwise, system is inclined to increase the number of links to be repaired and keep collected data as much as possible when c_i is high.

Problem P is a MIP problem that is NP-hard. Given the large number of variables to be optimized, there is no algorithm that can solve the problem optimally within polynomial time. Currently, there are three possible methods that can be used to handle this type of problem: heuristic algorithms [22], approximation algorithms [23] and exact algorithms [24]. Heuristic algorithms respond fast and are simple to apply, but lack rigorous theoretical proof and often have large deviations from optimal solutions. Approximation algorithms, such as the relaxed variable splitting algorithm, are limited by the size of solutions. Unlike the two types of algorithms, exact algorithms, the cutting plane algorithm for example, can explore the optimal solutions well by iterative update of cutting planes and are widely used in solving MIP.

3 Algorithm design

Benders decomposition [25], a classical cutting plane algorithm, is widely used to deal with realistic MIP problems (e.g., locomotive scheduling, airline route planning). It will neither significantly increase the number of iterations with the rise of the size of variable set as in branch-and-bound method, nor occur curse of dimensionality problem as in dynamic programming. Moreover, it will not cause large variance like what heuristic methods or simulated annealing algorithms do [26]. In this section, an efficient approach is proposed to solve problem P based on Benders decomposition.

3.1 Subproblem and reformulation

By leveraging Benders decomposition, problem P can be decomposed into a master problem and a subproblem. Variables to be optimized in the master problem are called complicating variables. When complicating variables are given, the left subproblem in problem P is easier to solve. For problem P, assume the complicating variables at t-th iteration to be e_{ij}^t, then the subproblem can be written as

$$S: \min_{\boldsymbol{d},\boldsymbol{f},\boldsymbol{p}} \theta = \sum_{i \in \mathcal{N}} c_i d_i$$
$$\text{s.t. } (1), (2), (5)\text{-}(7)$$
$$|f_{ij}| \leq \overline{f_{ij}} e_{ij}^t, \forall ij \in E^0 \tag{8}$$

Since the solving of complicating variables is decomposed into master problem, variables in the above subproblem are all continuous. The problem S is equivalent to a minimum cost flow problem. Given environmental monitoring services in IIoT, edge nodes are responsible for processing data that has the same priority [3,15]. In other words, the price for discarding a unit of data is the same (i.e., $c_i = c_j, \forall i, j \in \mathcal{N}$). Therefore, the minimum cost flow problem can be further transformed into a maximum flow problem [27]. Figure 2 illustrates the equivalence through an edge network consisting of four nodes.

In Figure 2, the solid line segments are the undamaged links, and the dotted line segment between nodes 2 and 3 represents the damaged link that needs to be determined whether it can be repaired (i.e., $\{ij, ij \in E^r, E^r \subseteq E^0\}$). The source node s and des-

tination node z are two added virtual nodes. Virtual links between the source node s and the four edge nodes represent the maximum computational capacity of edge nodes, while virtual links between edge nodes and destination node z is the processing demands of edge nodes. The value on the edge of each link in Figure 2 indicates the link capacity. In this way, the problem S is equivalent to maximizing the sum of arriving flows at destination node z. The transformed maximum flow problem can then be effectively solved by Ford-Fulkerson algorithm [28].

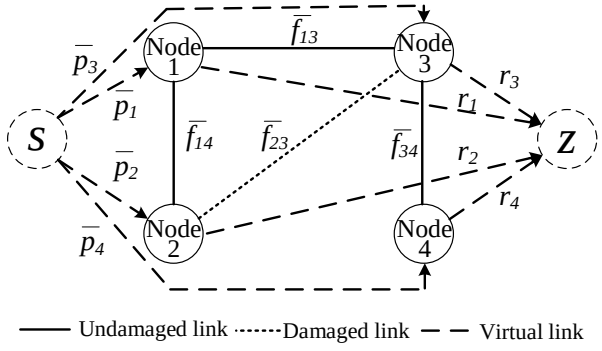

Figure 2: Illustration of the equivalent maximum flow problem of the subproblem

3.2 Optimality cuts and master problem

In Benders decomposition, solutions of the subproblem will be transferred into the linear constraints of the master problem in the form of Benders cutting planes. Consider feasibility [29], there are always feasible solutions to the subproblem for any feasible complicating variables. Benders optimality cuts can be generated by using complementary relaxation of the subproblem's dual problem [30]. The corresponding dual variables $\mu_{ij}^t, ij \in E^0$ of constraints (3) can then be extracted from the solutions of the subproblem, which represent the marginal system increment brought by repairing link ij.

Theorem 1 The Benders optimality cut at t-th iteration can be written as

$$\eta \geq \theta^t + \sum_{ij \in E^0} \mu_{ij}^t \bar{f}_{ij} (e_{ij} - e_{ij}^t) \tag{9}$$

where η is the upper bound of the objective of subprolem S, θ^t is the minimum value of objective at t-th iteration (i.e., minimal data discarding prices at t-th iteration).

Proof: Please refer to Appendix 1.

Adding obtained the Benders optimality cut into the constraints of master problem leads to

$$M: \quad \min_{\mathbf{e},\eta} \underline{\phi} = \sum_{ij \in E^0} c_{ij} e_{ij} + \eta$$
$$\text{s.t. (4), (9)}$$

where $\underline{\phi}$ is the lower bound of problem P, as master problem is a relaxation problem of P.

Different from the subproblem that determines data migration, computing, and discarding in the system, the master problem is responsible for deciding links to be repaired first (i.e., E^r) in damaged link set E^0. During the iterations of master problem and subproblem, the master problem is solved with given values of the amount of data migrated, computed, and discarded that obtained from the subproblem. Solutions to the master problem are then used to enhance the calculating of the subproblem. Therefore, the system can gradually approach the optimal system decisions by iteration.

3.3 Iterative path repair and computation migration algorithm based on Benders decomposition

Initializing the master problem and submitting it into the subproblem to search for the optimal solutions through iterations. If there are no decision variables that can ensure all constraints of master problem, the algorithm is terminated and problem P has no solutions. Otherwise, iterations will continue until optimal configurations are found. Detailed steps of the proposed algorithm are shown in Algorithm 1.

Algorithm 1 Iterative path repair and computation migration algorithm based on Benders decomposition

Set $t=1$ and give network parameters $\{c_{ij}, \bar{f}_{ij} \mid \forall ij \in E\}$ and $\{r_i, \bar{p}_i, c_i \mid \forall i \in \mathcal{N}\}$.

1: Initialize the upper bound of algorithm $\bar{\phi} = +\infty$, lower bound $\underline{\phi} = -\infty$, and precision ε for iteration;

2: **while** $(\bar{\phi} - \underline{\phi}) / \bar{\phi} < \varepsilon$ **do**

3: **if** Master problem M is feasible **then**

4: Obtain $\mathbf{e}^t = (e_{ij}^t, \forall ij \in E^0)$ and $\underline{\phi}$;

5: **else**

6: **break**;

7: Calculate subproblem S and obtain values of $\boldsymbol{d}^t = (d_i^t, \forall i \in \mathcal{N})$, $\boldsymbol{p}^t = (p_i^t, \forall i \in \mathcal{N})$, data migrated $\boldsymbol{f}^t = (f_{ij}^t, \forall ij \in E)$, dual variable $\boldsymbol{\mu}^t = (\mu_{ij}^t, \forall ij \in E^0)$, and minimum objective value θ^t;

8: Update $\bar{\phi} = \min\{\bar{\phi}, \sum_{ij \in E^0} c_{ij} e_{ij}^t + \theta^t\}$;

9: $t = t + 1$;

10：**end while**

When Algorithm 1 converges, the system will repair links on the basis of the results of current iteration $\boldsymbol{e}^t$. After that, the solution of the subprolem $\boldsymbol{f}^t$ is used to determine the amount of data migrated between nodes, $\boldsymbol{p}^t$ is utilized to dynamically adjust data processed at each edge node, and $\boldsymbol{d}^t$ is used to discard data that exceeds computation capacity. Note that η is a continuous variable while $\boldsymbol{e}$ is discrete, the master problem is still a MIP problem. Although it can be solved by algorithms such as the genetic algorithm with ant colony optimization, the complexity is high due to the large search space. Besides, the nature that a new cutting plane keeps monotonicity of the lower bound but not that for its upper bound. This non-monotonic bounding property will further aggravates the computational costs of Algorithm 1.

4 Accelerated Benders decomposition algorithm

To solve the problem of Algorithm 1, we further improve the algorithm with the local branching technique in the iteration process [31]. The main purpose is to find a better upper bound during each iteration to reduce the computation complexity of the master problem.

4.1 True region and Hamming distance

Algorithm 1 is not a stable algorithm as its solutions fluctuate widely in different feasible domains at the early stage of iterations, which leads to a slow convergence rate. For large-scale damage recovery in IIoT, the large number of variables to be optimized, especially the introduction of link repair vector $\boldsymbol{e}$ with an initial search space of $2^{|E^0|}$, will worsen the convergence problem. Trust region is a good way to handle the problem [32]. Given that $\boldsymbol{e}$ is a set of 0-1 variables, Hamming distance can be used to limit the distance between two iterations.

Let $(\boldsymbol{e}^t, \boldsymbol{d}^t, \boldsymbol{p}^t, \boldsymbol{f}^t)$ be the solutions of t-th iterations of problem P, where complicating variables whose values are 1 can be represented by $E^t = \{e_{ij}^t \mid e_{ij}^t = 1, e_{ij}^t \in E^0\}$. The Hamming distance between $(t+1)$-th and t-th iterations follows

$$D(e_{ij}^{t+1}, e_{ij}^t) = \sum_{ij \in E^t} (1 - e_{ij}^{t+1}) + \sum_{ij \in E^0 \setminus E^t} e_{ij}^{t+1} \tag{10}$$

which denotes the number of binary complicating variables changed during the $(t+1)$-th iteration.

To speed up convergence, the solution space can be decomposed into two mutually independent trust regions as

$$D(e_{ij}^{t+1}, e_{ij}^t) \le \Delta^{t+1}, \tag{11}$$

$$D(e_{ij}^{t+1}, e_{ij}^t) \ge \Delta^{t+1} + 1, \tag{12}$$

where Δ^{i+1} denotes the size of the trust region at the $(t+1)$-th iteration, the value of which depends on the complexity of the master problem and the fluctuating demand of search range. In local branching, equations (11) and (12) are named left branch and right branch, respectively.

4.2 Local branching

With the aid of Hamming distance, the solution space of problem P can be partitioned into two closely connected neighborhood spaces based on the solutions $(\boldsymbol{e}^t, \boldsymbol{d}^t, \boldsymbol{p}^t, \boldsymbol{f}^t)$ obtained by t–th iteration of Benders decomposition. Let $\boldsymbol{e}^k, k \in \boldsymbol{K}^t$ be the solutions (including feasible L^t （$\boldsymbol{L}^t \subseteq \boldsymbol{K}^t$） and non-feasible solutions) computed by t-th iteration of the local-branching-assisted Benders decomposition. According to Hamming distance, problem P can be branched into two independent problems, where the left-branch problem satisfies

$$P_k: \quad \min_{\boldsymbol{e},\boldsymbol{d},\boldsymbol{f},\boldsymbol{p}} \phi = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in \mathcal{N}} c_i d_i$$

s.t. (1)-(7)

$$D(e_{ij}, e_{ij}^k) \ge 1, \forall ij \in E^0, k \in \boldsymbol{K}^t \tag{13}$$

$$D(e_{ij}, e_{ij}^l) \ge \Delta^t + 1, \forall ij \in E^0, l \in \boldsymbol{L}^t \tag{14}$$

$$D(e_{ij}, e_{ij}^m) \le \Delta^t, \forall ij \in E^0 \tag{15}$$

where (13) represents that $\mathbf{e}$ that has been previously compared will not repeat again, (14) indicates that the current left branch is a sub-branch of the previous right branch, and (15) is the constraint of left branch with e_{ij}^m being its current optimal solutions. Similarly, the right branch can be written as

$$\overline{P_k}: \quad \min_{\boldsymbol{e},\boldsymbol{d},\boldsymbol{f},\boldsymbol{p}} \phi = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in \mathcal{N}} c_i d_i$$

s.t. (1)-(7), (13), (14)

$$D(e_{ij}, e_{ij}^m) \ge \Delta^t + 1, \forall ij \in E^0 \tag{16}$$

where (16) is the right branch constraint.

Let $(\boldsymbol{e}^{k+1}, \boldsymbol{d}^{k+1}, \boldsymbol{p}^{k+1}, \boldsymbol{f}^{k+1})$ be the optimal solutions of left-branch problem P_k and ϕ_{k+1} be the corresponding objective value. The process flow of the local branching is shown in Algorithm 2.

Loop constraint $T_{\max}$ is set to avoid possible non-feasible conditions caused by large Hamming distance. In a heavily damaged edge computing network, the search space size of link repair decision $\boldsymbol{e}$ is large. A small trust region size Δ^t will be more

beneficial to improve the computation speed of the left-branch problem.

In the iteration process, a strict upper bound $\overline{\phi_k} = \min_{k \in K^t}\{\phi_k\}$ can be obtained. The difficulty of solving problem P lies in obtaining better lower bounds. By generating a series of optimality cuts during each iteration, the lower bound of Bender decomposition can be enhanced, thus accelerating the searching speed.

Algorithm 2 Local branching method

Set $k=1$ and objective value as ϕ_k. Give the values of Δ^t, maximum computing time $T_{\max}$ and iteration precision $\bar{\varepsilon}$;

1: Initialize $(\boldsymbol{e}^k, \boldsymbol{d}^k, \boldsymbol{p}^k, \boldsymbol{f}^k)=(\boldsymbol{e}^t, \boldsymbol{d}^t, \boldsymbol{p}^t, \boldsymbol{f}^t)$ and obtain corresponding objective value ϕ_k;

2: **while** time reaches $T_{\max}$ or $(\phi_k - \phi_{k+1})/\phi_k < \bar{\varepsilon}$ **do**

3: Partition the current branch into left branch P_k and right branch $\overline{P_k}$, and calculate ϕ_{k+1} as well as $(\boldsymbol{e}^{k+1}, \boldsymbol{d}^{k+1}, \boldsymbol{p}^{k+1}, \boldsymbol{f}^{k+1})$;

4: **if** $\phi_{k+1} < \phi_k$ **then**

5: Update $L^t = L^t \bigcup \{k\}$, $K^t = K^t \bigcup \{k\}$, and jump to the right branch;

6: **else**

7: $\Delta^t = \Delta^t + 1$

8: Update $K^t = K^t \bigcup \{k\}$

9: Recalculate left branch P_k, $(\boldsymbol{e}^{k+1}, \boldsymbol{d}^{k+1}, \boldsymbol{p}^{k+1}, \boldsymbol{f}^{k+1})$, and corresponding ϕ_{k+1}

10: **continue**

11: **end if**

12: $k = k+1$

13: **end while**

4.3 Accelerated Benders decomposition based on local branching

The accelerated Benders decomposition algorithm based on local branching is shown in Algorithm 3. Similar to Algorithm 1, the system determines the set of links to be repaired based on results of current iteration $\boldsymbol{e}^t$ when Algorithm 3 converges. Then, deciding the amount of data migrated, processed, and discarded according to the results of $\boldsymbol{f}^t$, $\boldsymbol{p}^t$, and $\boldsymbol{d}^t$. Different from Algorithm 1, Algorithm 3 ensures that the upper bound of problem P is strictly decreasing during iteration through local branching techniques. For heavily damaged networks, this means that the search space of link repair decision vector $\boldsymbol{e}^t$ in master problem M will gradually reduce during iterations. Therefore, the searching speed for solutions will be accelerated as the number of iterations increases.

Only the strongest cutting plane (i.e., one that minimizes the feasible search space) needs to be added to the master problem, although a large number of optimality cuts can be obtained during iterations. Given the poor performance of convergence at the initial stage of Benders decomposition, the trust region can be added only in the initial stage and be removed when iterations tend to be stable.

Algorithm 3 Iterative path repair and computation migration algorithm based on local-branching-assisted Benders decomposition

Set $t=1$ and give network parameters $\{c_{ij}, \overline{f_{ij}} \mid \forall ij \in E\}$ and $\{r_i, \overline{p_i}, c_i \mid \forall i \in \mathcal{N}\}$。

1: Initialize upper bound of algorithm $\bar{\phi} = +\infty$, lower bound $\underline{\phi} = -\infty$, and precision ε for iteration;

2: **while** $(\bar{\phi} - \underline{\phi})/\bar{\phi} < \varepsilon$ **do**

3: **if** Master problem M is feasible **then**

4: Obtain $\boldsymbol{e}^t = (\mathrm{e}_{ij}^t, \forall ij \in E^0)$ and $\underline{\phi}$;

5: **else**

6: **break**

7: Calculate subproblem S and obtain values of $\boldsymbol{d}^t = (d_i^t, \forall i \in \mathcal{N})$, $\boldsymbol{p}^t = (p_i^t, \forall i \in \mathcal{N})$, data migrated $\boldsymbol{f}^t = (f_{ij}^t, \forall ij \in E)$, dual variables $\boldsymbol{\mu}^t = (\mu_{ij}^t, \forall ij \in E^0)$, and minimum objective value θ^t;

8: Update $\bar{\phi} = \min\{\bar{\phi}, \sum_{ij \in E^0} c_{ij} e_{ij}^t + \theta^t\}$;

9: Run Algorithm 2 to obtain an augmented upper bound $\overline{\phi_k}$, and a series of Benders optimality cuts generated by different feasible solutions;

10: $t = t+1$

11: **end while**

It is worth noting that each branching problem $P_1, P_2, \cdots$ during different iterations has the same structure as problem P. Hence, Algorithm 1 can be used to solve any branching problem, the results of which can also be used for solving subsequent branching problems.

5 Simulations

In this section, the performance of the proposed

Iterative path repair and computation migration algorithm based on Benders decomposition (short as Benders decomposition) together with Iterative path repair and computation migration algorithm based on local-branching-assisted Benders decomposition (short as Accelerated Benders decomposition) are simulated and analyzed. Simulation parameters and benchmarks are discussed to illustrate the merits of the proposed approaches.

5.1 Simulation parameters and benchmarks

We consider an edge computing network consisting of $N=50$ nodes whose maximum computation capacity $\overline{p_i}$ at the early stage of recovery is independent and uniformly over [10,25] Gbits. Given the match between computing capacity and demands, the processing demand of node i (i.e., r_i) follows a uniform distribution over [10,25] Gbits. Without loss of generality, we adopt a random network for the topology before damaging [4] and ensure the connectivity of every node. Link capacity $\overline{f_{ij}}$ obeys a uniform distribution over [5,15] Gbits considering the fluctuation of available computing capacities. The percentage of damaged links, if not specified, is set as q=75% to model large-scale damage. The cost for repairing each damaged link c_{ij} is assumed to be uniformly distributed over [10^3, 2×10^3] dollars, and the price for discarding a unit of data c_i to be 10^3 dollars/Gbits.

To verify the merits of the proposed recovery mechanism, we use three benchmarks for comparison:

(1) Random recovery (RR): Damaged links are randomly chosen for repairing without any considerations of network topology and dynamics.

(2) Giant component based recovery (GCR) [16]: Prioritizing the repair of damaged links in the giant component to ensure the connectivity of the network.

(3) Betweenness centrality based recovery (BCR) [11]: Prioritizing the repair of damaged links whose betweenness centrality in pre-damage topology are higher. The number of links to be repaired is the same as the proposed approach.

5.2 Convergence rate

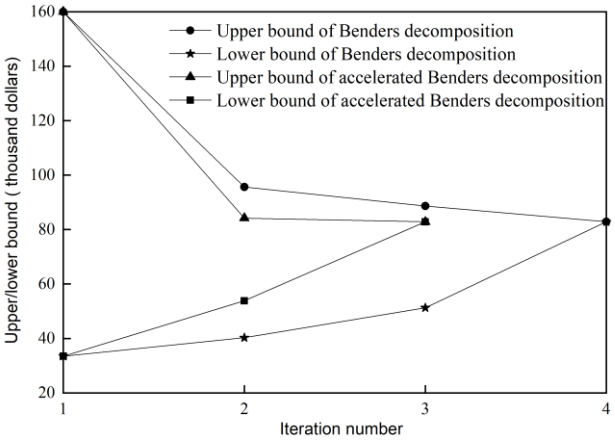

Figure 3: Convergence of the two proposed approaches.

Figure 3 shows the convergence performance of the proposed Benders decomposition algorithm and the accelerated Benders decomposition algorithm. The two algorithms can converge within a finite number of iterations. Note that the iteration number in Figure 3 is the total number of iterations of the master problem and subproblem. In each iteration, the computation complexity is $O(|\mathcal{N}||E|^2)$ for subproblem and $O(n|E^0|^3)$ for master problem, where n is the number of iterations of the inner loop for solving master problem. Although the accelerated algorithm only reduce one iteration compared with Benders decomposition, the computation time complexity decreases $O(|\mathcal{N}||E|^2+n|E^0|^3)$, which is significantly high in heavily damaged networks.

Compared to the proposed Benders decomposition algorithm, the accelerated approach, thanks to the assistance of local branching, gets stronger Benders cuts to speed up convergence and is more adaptive for large-scale damage recovery. Since the two approaches reach the same system cost, they are collectively referred to as the *proposed algorithm* in the sequel for performance comparison.

5.3 System cost analysis

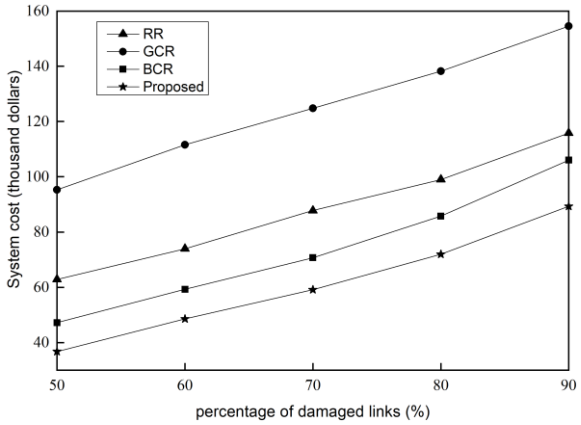

Figure 4: System cost under different levels of damage.

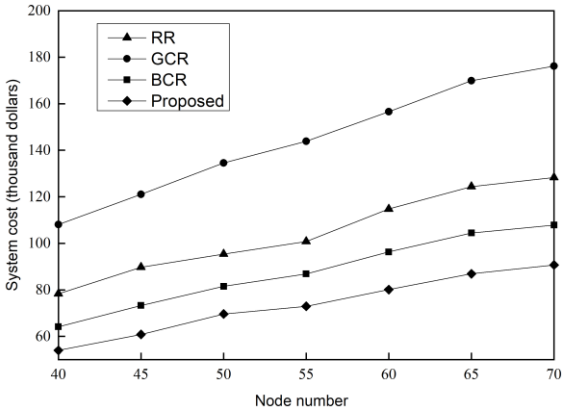

Figure 5: System cost under different network sizes.

Figure 4 and Figure 5 illustrate the system cost under different levels of damage and network sizes. In different conditions, the proposed approach outperforms benchmarks as it considers network dynamics such as computation demands and data migration. Note that GCR has the highest system cost, even higher than the RR algorithm, which is in accordance with realistic network recovery. In realistic systems, the recovery process always sees a number of independently connected clusters, and then the connection of different clusters occurs to form a giant component [13].

5.4 Applicability of multiple scenarios

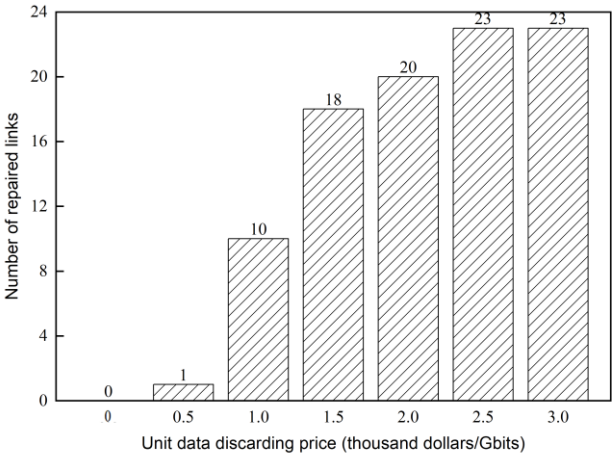

Figure 6: The number of link repaired under different unit data discarding price.

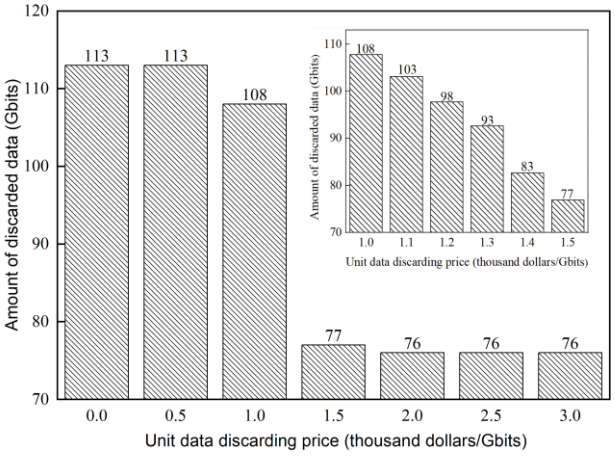

Figure 7: The amount of data discarded under different unit data discarding price.

Figure 6 and Figure 7 show detailed system cost information under different data discarding prices. Given that system cost balances the costs for link repair and data discard, the number of repaired links and the amount of discarded data appear an inverse relationship as data discarding price c_i increases. When c_i increases, the system tends to restore more links to meet computation demands. If c_i is between [1,1.5] $\times 10^3$ dollars/Gbits, as shown in Figure 7, the system is sensitive to the change in the discarding price. A small change of c_i will have a relatively large impact on system balance. Once c_i exceeds 2×10^3 dollars/Gbits, the marginal gain brought by link repair decreases. In this case, the cost for link repair and the price for data discard remain stable. Figure 6 and Figure 7 together demonstrate good scalability and adaptability under different scenarios with distinct data significance.

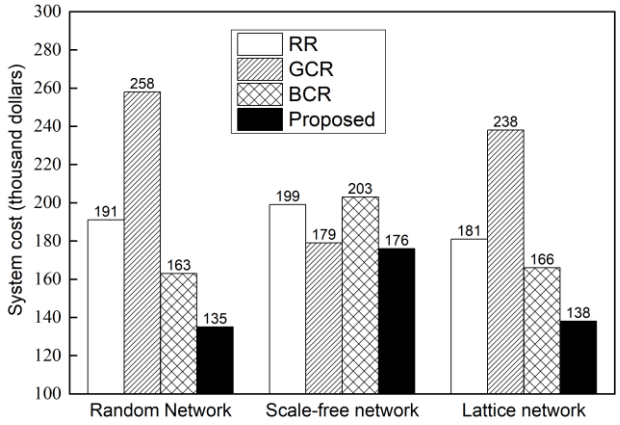

Figure 8: System cost in different network topologies (N=100).

To analyze the algorithm performance for different network topologies, we extend the topology from random networks to lattice networks and scale-free networks. Different from random networks, the degrees of nodes in lattice networks are the same, and scale-free networks follow the power law [33]. As can be seen from Figure 8, the proposed recovery mechanism outperforms benchmarks in all the topologies, although fluctuating as topology changes. In other words, the proposed approach can be applied to multiple scenarios thanks to the joint analysis of network topology and system dynamics.

6 Conclusions

In this article, we propose a recovery mechanism for large-scale damage in edge computing networks of IIoT. A joint optimization method of link repair and data migration is presented, which alleviates the contradiction between limited repair resources and large data computing demands at the early stage of network recovery. Given the complexity for solving the NP-hard problem, Benders decomposition is applied to transform the problem into a master problem and a subproblem, through the iterations of which we can gradually approach the optimal solutions. With the aid of local branching techniques, an accelerated algorithm is designed to improve the speed of convergence for large-scale network analysis. Simulations reveal that the proposed approach can achieve a lower system cost in recovery compared with existing schemes.

We consider that data processed at different edge nodes have the same priority, i.e., the same data discarding price, out of fairness concerns. Even so, the proposed algorithm can be easily extended to scenarios with different discarding prices by substituting the Ford-Fulkerson algorithm with a minimum cost flow algorithm (e.g., the Dinic algorithm).

In this article, we consider an edge computing network connected by cable links. For scenarios with dynamic wireless links, unmanned aerial vehicles (UAVs) for example, channel capacities change with the positions of UAVs. It will necessitate the joint consideration of UAV trajectory and spectrum allocation, which makes the current complicated problem even more difficult to solve and thus is left for future research.

Appendix 1: Proof of Benders optimality cuts

For given $\mathrm{e}_{ij}^t, ij \in E^0$, from problem P, we have

$$\theta(\mathrm{e}_{ij}^t) = \min_{\boldsymbol{d},\boldsymbol{f},\boldsymbol{p}} \{\sum_{i\in\mathcal{N}} c_i d_i \mid (1)-(3), (5)-(7)\}. \tag{17}$$

The objective of its equivalent dual problem can be written as

$$\theta(\mathrm{e}_{ij}^t) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\lambda},\boldsymbol{\mu}} \{\sum_{i\in\mathcal{N}} (\alpha_i r_i + \beta_i \overline{p_i}) + \sum_{ij\in E^1} \lambda_{ij} \overline{f_{ij}} + \sum_{ij\in E^0} \mu_{ij} \overline{f_{ij}} e_{ij}^t\}, \tag{18}$$

where variables $\boldsymbol{\alpha} = (\alpha_i, \forall i \in \mathcal{N})$, $\boldsymbol{\beta} = (\beta_i, \forall i \in \mathcal{N})$, $\boldsymbol{\lambda} = (\lambda_{ij}, \forall ij \in E^1)$, and $\boldsymbol{\mu} = (\mu_{ij}, \forall ij \in E^0)$ are four sets of dual variables to be optimized in (18). Assume that the solutions of the dual problem at $t-th$ iteration to be $\boldsymbol{\alpha}^t = (\alpha_i^t, \forall i \in \mathcal{N})$, $\boldsymbol{\beta}^t = (\beta_i^t, \forall i \in \mathcal{N})$, $\boldsymbol{\lambda}^t = (\lambda_{ij}^t, \forall ij \in E^1)$, and $\boldsymbol{\mu}^t = (\mu_{ij}^t, \forall ij \in E^0)$, the optimal object to be $\theta^t = \theta(e_{ij}^t)$, we have

$$\theta(\mathrm{e}_{ij}) \geq \sum_{i\in\mathcal{N}} (\alpha_i^t r_i + \beta_i^t \overline{p_i}) + \sum_{ij\in E^1} \lambda_{ij}^t \overline{f_{ij}} + \sum_{ij\in E^0} \mu_{ij}^t \overline{f_{ij}} e_{ij}. \tag{19}$$

Operating a linearization in the vicinity of $\mathrm{e}_{ij}^t, ij \in E^0$ yields

$$\begin{aligned} &\eta \geq \theta(\mathrm{e}_{ij}) \\ &\geq \sum_{i\in\mathcal{N}} (\alpha_i^t r_i + \beta_i^t \overline{p_i}) + \sum_{ij\in E^1} \lambda_{ij}^t \overline{f_{ij}} \\ &\quad + \sum_{ij\in E^0} \mu_{ij}^t \overline{f_{ij}} e_{ij}^t + \sum_{ij\in E^0} \mu_{ij}^t \overline{f_{ij}} (e_{ij} - e_{ij}^t) \\ &= \theta^t + \sum_{ij\in E^0} \mu_{ij}^t \overline{f_{ij}} (e_{ij} - e_{ij}^t), \end{aligned} \tag{20}$$

which concludes the proof.

References

[1] WU H, TIAN H, NIE G F, et al. Wireless Powered Mobile Edge Computing for Industrial Internet of Things Systems[J]. IEEE Access, 2020, 8:101539-101549.

[2] Xie G Q, Yang K H, Xu C, et al. Digital twinning based adaptive development environment for automotive cyber-physical systems[J] IEEE Transactions on Industrial Informatics, 2021, to be published.

[3] WU H, LYU X C, TIAN H. Online optimization of wireless powered mobile-edge computing for heterogeneous industrial internet of things[J]. IEEE Internet of Things Journal, 2019, 6(6): 9880-9892.

[4] XING L. Cascading Failures in Internet of Things: Review and Perspectives on Reliability and Resilience[J]. IEEE Internet of Things Journal, 2020, 8(1): 44-64.

[5] PROKHORENKI V, BABAR M A. Architectural resilience in cloud, fog and edge systems: A survey[J]. IEEE ACCESS, 2020, 8: 28078-28095

[6] AYOUBI S, ASSI C, CHEN Y, et al. Restoration methods for cloud multicast virtual networks[J]. Journal of Network and Computer Applications, 2017, 78: 180-190.

[7] SATRIA D, PARK D, JO M. Recovery for overloaded mobile edge computing[J]. Future Generation Computer Systems, 2017, 70: 138-147.

[8] TENG R, LI H B, MIURA R. Dynamic recovery of wireless multi-hop infrastructure with the autonomous mobile base station[J]. IEEE Access, 2016, 4: 627-638.

[9] LI P D, YANG X F. On dynamic recovery of cloud storage system under advanced persistent threats[J]. IEEE Access, 2019, 7: 103556-103569.

[10] BAXTER G J, Timár G, MENDES J F F. Targeted damage to interdependent networks[J]. Physical Review E, 2018, 98(3): 032307.

[11] BRUMMITT C D, D'SOUZA R M, LEICHT E A. Suppressing cascades of load in interdependent networks[J]. Proceedings of the National Academy of Sciences, 2012, 109(12): E680-E689.

[12] MORONE F, MAKSE H A. Influence maximization in complex networks through optimal percolation[J]. Nature, 2015, 524(7563): 65-68.

[13] RUDNICK H, MOCARQUER S, ANDRADE E, et al. Disaster management[J]. IEEE Power and Energy Magazine, 2011, 9(2): 37-45.

[14] PUNZO G, TEWARI A, VASILE M, et al. Engineering resilient complex systems: The necessary shift toward complexity science[J]. IEEE System Journal, 2020, 14(3): 3865-3874

[15] WU H, TIAN H, FAN S S, et al. Data Age Aware Scheduling for Wireless Powered Mobile-Edge Computing in Industrial Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2020, 17(1): 398-408.

[16] SMITH A M, Pósfai M, ROHDEN M, et al. Competitive percolation strategies for network recovery[J]. Scientific reports, 2019, 9(1): 1-12.

[17] QIN J, MIAO L, Combined Simulated Annealing Algorithm for Logistics Network Design Problem[C]// 2009 International Workshop on Intelligent Systems and Applications, Wuhan, China, 2009: 1-4,

[18] KHELIFI M, SAIDI M Y and BOUDJIT S, Genetic algorithm based model for capacitated network design problem[C]// 2016 24th In-ternational Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 2016: 1-6

[19] LIAN H B, Network Design Problems, Formulations and Solutions[D] Order No. 3507762 The University of Texas at Dallas, 2012. Ann Arbor: ProQuest. Web. 3 Mar. 2021.

[20] LI D Q, ZHANG Q, ZIO E, et al. Network reliability analysis based on percolation theory[J]. Reliability Engineering & System Safety, 2015, 142:556–562.

[21] LYU X C, REN C S, NI W L, et al. Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 574-586.

[22] ZHAO P T, TIAN H, QIN C, et al. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing[J]. IEEE Access, 2017, 5: 11255-11268.

[23] ZHAO P T, TIAN H, CHEN K C, et al. Context-aware TDD configuration and resource allocation for mobile edge computing[J]. IEEE Transactions on Communications, 2019, 68(2): 1118-1131.

[24] CERISOLA LOPEZ DE HARO S, Ramos Galán A. A finite Benders decomposition algorithm for mixed integer problems, resolution through parametric Branch and Bound[J]. 2016.

[25] RAHMANIANI R, CRAINIC T G, GENDREAU M, et al. The Benders decomposition algorithm: A literature review[J]. European Journal of Operational Research, 2017, 259(3): 801-817.

[26] WU H, TIAN H, NIE G F. Energy-efficient inter-frequency small cell discovery in dense urban environments[J]. IEEE Wireless Communications Letters, 2019, 8(1): 41-44.

[27] PEREIRA M V F, PINTO L, CUNHA S H F, et al. A decomposition approach to automated generation/transmission expansion planning[J]. IEEE Transactions on Power Apparatus and Systems, 1985 (11): 3074-3083.

[28] ERICKSON J. Algorithms[M]. Illinois: Independently Published, 2019.

[29] OLIVEIRA F, GROSSMANN I E, HAMACHER S. Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain[J]. Computers & Operations Research, 2014, 49: 47-58.

[30] Gan Y. Operations Research [M]. Beijing: Tsinghua University Press, 2013.

[31] FISCHETTI M, LODI A. Local branching[J]. Mathematical programming, 2003, 98(1-3): 23-47.

[32] SANTOSO T, AHMED S, GOETSCHALCKX M, et al. A stochastic programming approach for supply chain network design under uncertainty[J]. European Journal of Operational Research, 2005, 167(1):

96-115.

[33] YU A, Wang N, WU N. Scale-free networks: Characteristics of the time-variant robustness and vulnerability[J]. IEEE Systems Journal, 2020, to be published.