Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3918207, 15 pages
https://doi.org/10.1155/2022/3918207

Research Article

An Optimized Approach for Industrial IoT Based on Edge Computing

Hongyang Huang,[1] Mohammed Dauwed,[2] Morched Derbali,[3] Imran Khan,[4] Sun Li,[5] Kai Chen,[6] and Sangsoon Lim[7]

[1]*Graduate School of Business, Segi University, Jalan Teknologi, Kota Damansara, 47810 Petaling Jaya, Selangor, Malaysia*
[2]*Department of Medical Instrumentation Techniques Engineering, Dijlah University College, Baghdad, Iraq*
[3]*King Abdulaziz University (KAU), Faculty of Computing and Information Technology (FCIT), Jeddah, Saudi Arabia*
[4]*Department of Electrical Engineering, University of Engineering & Technology, Peshawar 814, Pakistan*
[5]*Advanced Information Research Center of Xi'an Jiaotong University, China*
[6]*Huawei Technologies, Stockholm, Sweden*
[7]*Department of Computer Engineering, Sungkyul University, Anyang 430010, Republic of Korea*

Correspondence should be addressed to Sangsoon Lim; slim@sungkyul.ac.kr

Received 17 May 2022; Accepted 24 June 2022; Published 9 July 2022

Academic Editor: Lisheng Fan

The Internet of Things (IoT) is an information network that connects gadgets and sensors to allow new autonomous tasks. The Industrial Internet of Things (IIoT) refers to the integration of IoT with industrial applications. Some vital infrastructures, such as water delivery networks, use IIoT. The scattered topology of IIoT and resource limits of edge computing provide new difficulties to traditional data storage, transport, and security protection with the rapid expansion of the IIoT. In this paper, a recovery mechanism to recover the edge network failure is proposed by considering repair cost and computational demands. The NP-hard problem was divided into interdependent major and minor problems that could be solved in polynomial time by using the Benders decomposition technique and cutting plane approximation. To ensure the nonincreasing character of the Benders upper limit, a local branching method was also added to improve the convergence. Simulation results indicated that the proposed method is superior to the existing method and has better overall performance.

1. Introduction

The Industrial Internet of Things (IIoT) is regarded as an important driver of the intelligent transformation of the global industrial system. Relying on hundreds of millions of seamlessly deployed sensors, collectors, and controllers, the Industrial Internet of Things can simulate, predict, and control the full cycle of the manufacturing process [1]. As the "brain" of the IIoT, the edge computing network provides more sufficient computing processing capabilities for wireless acquisition devices, effectively reducing processing and transmission delays, and is useful for digital twin (DT) [2], virtual reality (VR), etc. Enterprise high-level applications have laid a solid foundation [3]. At the same time, the wired link connection between edge computing nodes also makes the migration of computing tasks between nodes smoother, effectively alleviating the problem of unbalanced space-time allocation of computing resources caused by the space-time fluctuations of computing demands of the IIoT.

The normal and stable operation of the edge computing network is the key to the efficient operation of the IIoT. Once the "brain" is damaged, the IIoT system will lose effective control over the "limbs" (such as supply chain monitoring, data visualization, and analysis), bringing countless economic losses, even life-threatening. However, the stability requirements of edge computing networks face internal and external challenges. On the one hand, the edge computing network is coupled with the power grid, control network,

and other subnetworks in the IIoT, forming a highly vulnerable interdependent network [4]. Any slight fluctuations in other subnets may be transmitted to the edge computing network, causing large-scale system cascading failures. On the other hand, unpredictable events such as natural disasters and man-made attacks also test the robustness of edge computing networks at any time [5]. In order to meet the above challenges, on the one hand, the reliability of the edge computing network can be strengthened so that it can adaptively cope with various network fluctuations and prevent network failures. On the other hand, and more importantly, it is necessary to explore the rapid repair mechanism after the network is damaged, so that the network performance can be restored to the level close to before the damage as soon as possible.

Although the design of the repair mechanism is crucial for the sustainable and stable operation of the network, there is currently no research literature specifically targeting edge computing network scenarios in the IIoT. In view of the similarity of network topology, network dynamics, and other characteristics, some existing network repair strategies can still provide some reference for the design of edge computing network repair mechanisms in the IIoT. The research on the existing network repair mechanism mainly focuses on the rapid repair after the local network is damaged. Reference [6] constructs the problem of single node or link damage in the network as an integer linear programming problem and proposes a data migration-aware repair model, which achieves an effective balance between service interruption rate and repair cost. Further, in the literature [7, 8], considering the problem of network connectivity damage in multinode failure scenarios, it is proposed that users can be used as transit nodes between disconnected edge nodes in a device-to-device (D2D) way [7] or mobile devices can be used. The access node realizes that the data between network nodes can be reached everywhere [8], ensuring the connectivity of the damaged network. Different from the above studies, the reference [9] considers the continuous damage state in the case of network attack, transforms the network dynamic repair problem into a differential game theory problem, and enhances the network repair ability through Nash equilibrium necessary conditions and competitive strategy sets. However, the above-mentioned research on local network repair often ignores global network nodes, dynamic characteristics between links (such as flow migration), and practical scene constraints (such as link space layout without changes); it is difficult to effectively extend to the large-scale network damage scenarios that are more likely to occur in the IIoT.

After the network is damaged on a large scale, the initial available repair resources (such as the number of repair personnel and the number of replaceable devices) are often limited. How to effectively balance the limited system repair resources at the initial stage of repair and the urgent need for system performance recovery is an urgent problem to be solved in the current IIoT. Current research mainly focuses on the analysis of network topology. Reference [10] believes that large-degree nodes, that is, nodes with large node degrees, play a more important role in network connectivity and need to be repaired first. Similarly, in [11], considering that the link is damaged, the link with large betweenness centrality (BC) should be repaired first. Reference [12] found through the analysis of actual network data that weakly connected nodes in the network, that is, nodes with low degrees of themselves but connected to several large-degree nodes, play the most critical role in network connectivity, and their repairs are prioritized. The level should be higher than the large-degree node. However, the above schemes based on the growing maximum connected subgraph of network connectivity are all static network architecture analysis, ignoring the dynamic characteristics of the network. The large-scale network repair in the real environment often forms independent subgraphs first and then connects multiple subgraphs to form a maximum connected graph [13]. Therefore, in the engineering analysis, it is necessary to integrate more network equipment details and actual transmission dynamic analysis [14]. In order to solve the above problems, a heuristic algorithm that is easier to solve is proposed in [15]. However, it lacks the macroscopic analysis of the network, and the performance variance of the algorithm is large, so it cannot provide reliable performance guarantee. Similarly, both the simulated annealing algorithm [16] and the genetic algorithm [17] face the same dilemma as general heuristic algorithms and are easily trapped in local optimal solutions. Although the hill-climbing algorithm [18] or the gradient descent algorithm [19] can easily jump out of the local optimum and greatly reduce the computational complexity of the problem, the optimality of the solution cannot be guaranteed. This type of problem is also known as a network design problem (NDP). Due to the addition of the dynamic characteristics of the network, the complexity of the network design problem is extremely high (at least the NP-complete problem), and the traditional dynamic programming algorithm will cause the "dimension disaster" [20]. In reference [21], the problem of multihop computing task offloading in the hybrid edge cloud computing environment is studied, and the offloading method that meets the quality of service requirements is realized through the game method. In reference [22], optimization is carried out in terms of time cost and energy consumption cost to achieve the optimal allocation of large-scale green energy-saving computing resources. In reference [23], in order to meet the real-time requirements, an intelligent resource planning strategy under the hybrid computing structure is proposed. In summary, edge computing can provide more sufficient computing resources for field devices and reduce network load by deploying at the network edge closer to field devices, thereby meeting the requirements of task service quality and reducing system overhead in different scenarios.

In view of the practical dilemma faced by the current research, this paper proposes a repair mechanism for damaged edge computing networks in the IIoT scenario. Different from the existing literature, this paper deeply excavates the structural characteristics (topological relationship, link capacity) and dynamic characteristics (node computing requirements), and the link priority repair set decision and network computing migration issues are jointly considered,

in order to achieve an efficient balance between the initial computing requirements and repair costs of network repair. The main contributions of this paper are summarized as follows.

A network repair mechanism is proposed in the case of large-scale damage to the edge computing network. Combined with the network structure and dynamic characteristics of edge computing, an analysis framework of priority repair set decision and resource scheduling in the early stage of network repair is provided.

Based on the Benders decomposition algorithm, the complex mixed-integer problem is decomposed into two parts, the main problem and the subproblem, which are easier to solve. For the subproblems of multivariable groups, by adding virtual source nodes and destination nodes, the problem is transformed into a network maximum flow problem to solve.

A Benders decomposition acceleration algorithm based on local branching method is designed. The trust region based on the Hamming distance is used to shrink the search range of the feasible region and accelerate the convergence speed of the algorithm.

Simulation results show that the proposed algorithm has better convergence performance and lower system overhead. Compared with the existing random repair, maximum connected graph repair, and betweenness centrality sorting repair and other topology-based repair algorithms, the proposed algorithm has better performance in multiple scenarios and has good scalability and adaptability.

2. System Model

For the edge computing network in the IIoT, consider an edge computing network with N nodes, which is represented by the set $\forall i \in N$. The edge nodes are connected by wired links; the link set is represented by E. Since wired links are usually reliable, and only a small amount of channel coding complexity is required for computing tasks relative to edge nodes, the link between two edge nodes can be considered error-free [24]. The system parameters are shown in Table 1. It is worth noting that the edge computing network in the actual IIoT is often a hybrid link transmission network composed of wired links and wireless links. Since there is almost no damage to the wireless link, and its repair cost is negligible compared to the wired link, this paper only considers the case where all transmission links are wired links. Nevertheless, if the deep weakening of the short-term wireless channel is not considered, the wireless link in the static scenario (without considering the spectrum reallocation in the repair process) can be regarded as a nondestructive link with constant capacity, and the hybrid link transmission network can be considered equivalent to a pure wired link network, and the proposed algorithm will still be applicable.

The topology of the damaged edge computing network is shown in Figure 1. In the actual network, because edge nodes often have complex self-protection mechanisms (such as overheating protection), they are usually not prone to damage. Therefore, this paper does not consider node damage and focuses on link damage. In order to simulate the state of large-scale network failure caused by factors such as natural disasters, it is assumed that there are $q|E|$ wired links in the network at the initial moment, where q represents the percentage of damaged links in all links, and the set E_0 represents the damaged link. The set $E^1 = E \setminus E^0$ represents the link set that can still work normally. The damage of the link will cause the balanced computing migration flow between edge nodes to be broken and even form a computing island (computing migration cannot be performed, as shown in node 1 in Figure 1), resulting in a mismatch between computing requirements and computing capabilities, affecting the network computing performance. Constrained by limited repair resources at the initial stage of network repair (e.g., the number of repair personnel and replacement equipment inventory), it is impractical to repair all damaged links at the same time. In order to restore the network state as soon as possible, it can be distributed according to the network computing requirements, and some damaged links can be repaired preferentially, which is represented by the set E^r ($E^r \subseteq E^0$). For any link $ij \in E^0$, due to the different degree of damage, the cost c_{ij} of maintenance, repair, and replacement is also different.

Considering that the local data computing requirement of the edge node $i \in N$ is r_i, the actual local computing amount is p_i. It should be noted that r_i is the total calculated arrival amount of the ith edge node in the initial stage of network repair. For a given scenario, r_i is a fixed value that does not change with time and can be estimated more accurately based on historical computing needs. For any edge node i, the p_i satisfies

$$0 \le p_i \le \bar{p}_i, \forall i \in N. \tag{1}$$

Among them, $\bar{p}_i$ is the maximum computing power of node i.

Let the computational cost of migration between node i and node $j \in N\{i\}$ be f_{ij}. If $f_{ij} > 0$, the data computing task is migrated from node i to j. If $f_{ij} < 0$, the data computing task is migrated from node j to i. Limited by the wired link capacity $\bar{f}_{ij}$, the actual calculated migration of link ij satisfies

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

Further, for damaged links, the actual computational migration amount is not only determined by the wired link capacity, but also affected by the link repair decision, i.e.

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

where

$$e_{ij} \in \{0, 1\}, \forall ij \in E^0. \tag{4}$$

Among them, $e_{ij} = 1$ represents the priority to repair the link ij, that is, $ij \in E^r$; when $e_{ij} = 0$ means that the link ij is not repaired, it actually calculates the migration

Table 1: System parameters.

Parameter	Description
q	Damaged links as a percentage of all links
E^1	The set of all normal links in the edge computing network
E^0	The set of all damaged links in the edge computing network
E	The set of all links in the edge computing network
N	Number of edge computing network nodes
c_{ij}	Repair cost of link ij
c_i	The cost per unit of data discarded by node i
p_i	The amount of local computing data calculation of node i
$\bar{p}_i$	The maximum computing power of node i
r_i	The local data computation requirement of node i
d_i	The amount of computational data discarded by node i
f_{ij}	Computational migration from node i to node j
$\bar{f}_{ij}$	Capacity of link ij
e_{ij}	Whether to fix the decision variable of link ij
t	Number of iterations for Benders decomposition
θ^t	The optimal value of the objective function of the subproblem at iteration t
Δ^t	Left-branch problem threshold at iteration t

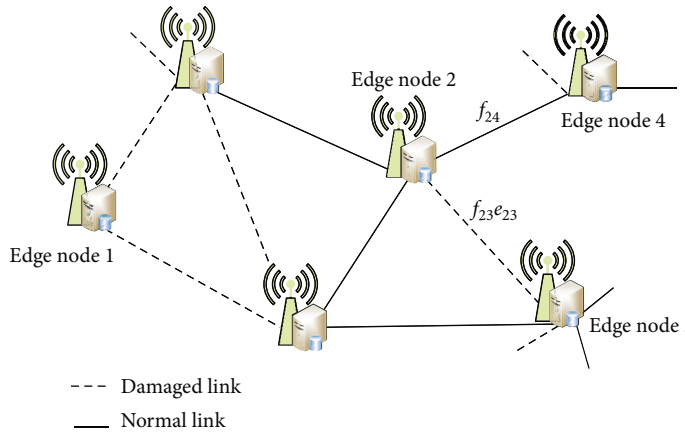

Figure 1: Damaged edge computing network topology.

amount $f_{ij} = 0$. In addition, for a single link ij, it can be known from the migration flow symmetry:

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

When the links in the set E^r are repaired, it can be known from the law of the conservation of computation of nodes:

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

Among them, r_i represents the local data computation requirement of node i; d_i represents the amount of computing tasks that edge node i has to give up due to data backlog due to limited computing power.

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

In order to repair as many links as possible, the amount of data discarded can be reduced, and the system performance can be improved, but it will cause a large system repair overhead. If there are too few repair links, the system repair overhead will be reduced, but it may cause the data to be discarded because it cannot be processed locally, which will damage the network performance. For quantitative analysis, the network performance in this paper is measured by the total cost of system data discarding. The greater the amount of data discarded, the higher

the total cost of data discarding, and the worse the network performance. Therefore, in order to balance the repair cost and network performance at the initial stage of network repair (total cost of data discarding), the following system cost minimization problem can be constructed

$$P: \min_{e,d,f,p} \varnothing = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in N} c_i d_i, \quad \text{s.t.Eqs.}(1) \sim (7). \tag{8}$$

Among them, c_{ij} represents the repair cost required to repair the link $ij \in E^0$, c_i denotes the cost of discarding each unit of data for node $i \in N$. The link repair decision vector $\boldsymbol{e} = (e_{ij}, \forall ij \in E^0)$, data discarding decision vector $\boldsymbol{d} = (d_i, \forall i \in N)$, the flow allocation vector $\boldsymbol{f} = (f_{ij}, \forall ij \in E)$, and the local actual computation vector $\boldsymbol{p} = (p_i, \forall i \in N)$. In problem P, the total system overhead consists of two parts: the total cost of system repair overhead and data discarding. As mentioned above, the system repair cost $\sum_{j \in E^0} c_{ij} e_{ij}$ and the total cost of data discarding $\sum_{i \in N} c_i d_i$ are a pair of contradictory quantities, an increase in one value will lead to a decrease in the other value, and minimizing their sum can effectively balance their effects. When the link repair cost c_{ij} is large, it indicates that the importance of computing data is relatively low, and the system tends to temporarily repair less damaged links and discard data that cannot be processed. When the cost per unit of data discarding is large, the system tends to repair more damaged links to reduce the discarding of computing data.

Problem P is a mixed-integer problem, NP-hard problem, and its data scale is large, and there is no known polynomial-time algorithm to solve the above problem. The current methods for solving the above problems can be divided into three categories: Heuristic algorithm [25], approximate algorithm [26], and exact algorithm [27]. The Heuristic algorithms are fast and easy to apply, but they lack rigorous theoretical proofs, and the results often deviate significantly from the optimal solution. Approximate algorithms, such as the slack variable method, have a limited solution scale, and the solution to the slack problem cannot accurately describe the optimal solution to the original problem. Different from the above two methods, the exact algorithm, such as the cut plane algorithm, can well explore the optimal solution of the problem through the iterative update of the cut plane and is widely used in the process of solving mixed-integer problems.

3. Algorithm Design

Benders decomposition algorithm [28] is a classical cutting plane algorithm, which is widely used to deal with real mixed-integer programming problems (such as locomotive scheduling and aviation route planning). This algorithm does not significantly increase the number of iterations with the increase of operating variables like the branch and bound method, nor does it produce dimensional disaster like dynamic programming, and does not appear heuristic, simulated annealing, and other algorithms have huge variance [29]. This section will give an efficient solution to problem P based on the Benders decomposition algorithm.

3.1. Subproblem Description and Transformation. In the Benders decomposition algorithm, the original problem can be decomposed into two parts, the main problem and the subproblem, and the optimization variables in the main problem are called complex variables. When the complex variables are fixed, the remaining optimization problems (i.e., subproblems) in the original problem become relatively easy to solve. For the problem P, if the value of the complex variable e_{ij}^t in the tth iteration process is given, the subproblem can be expressed as

$$S: \min_{d,f,p} \theta = \sum_{i \in N} c_i d_i, \quad \text{s.t.Eqs.}(1) \sim (2), (5) \sim (7), \tag{9}$$

$$\left| f_{ij} \right| \le \bar{f}_{ij} e_{ij}^t, \forall ij \in E^0. \tag{10}$$

Since the 0-1 variable e_{ij} in the original problem P is decomposed into the main problem, and the optimization variables in the above subproblems are all continuous variables, this problem can be equivalent to a minimum cost flow problem. This paper considers a typical environmental monitoring scenario in the IIoT. Each edge node in the edge network is responsible for processing the environmental data collected by wireless sensors. In the environment monitoring scenario, the computing tasks at each edge node have the same computing priority [3, 20], that is, the node discards the same cost per unit of data $(c_i = c_j, \forall ij \in N)$. Thus, the problem can be further transformed into the network maximum flow problem [30]. Figure 2 illustrates the above equivalence relationship with an edge network with four nodes.

In Figure 2, the connection line between nodes 2 and 3 is the damaged link that needs to be repaired determined in the iterative process, namely $\{ij, ij \in E^r,\ E^r \subseteq E^0\}$. The source node s and the destination node z are newly added virtual nodes, the virtual link capacity of the source node s and the four edge nodes is the maximum computing power of the four edge nodes, and the virtual link between the four edge nodes and the destination node z represents the local data computation requirement of each edge node. The numerical values on each edge in Figure 2 represent the link capacity of the link, so that maximizing the total arrival flow to the destination node z is equivalent to minimizing the total data discarded. The above transformed maximum flow problem can be efficiently solved by existing algorithms, such as the Ford-Fulkerson algorithm [31].

3.2. Cut Plane Generation and Main Problem Construction. In the Benders decomposition algorithm, the solutions of the subproblems are substituted into the main problem to generate linear constraints in the main problem, namely Benders cuts. Since this paper considers the perfect resource [32], that is, for any feasible solution to the main problem, there are always feasible subproblem solutions, and there is no need to generate feasible cuts; only the optimal cut needs

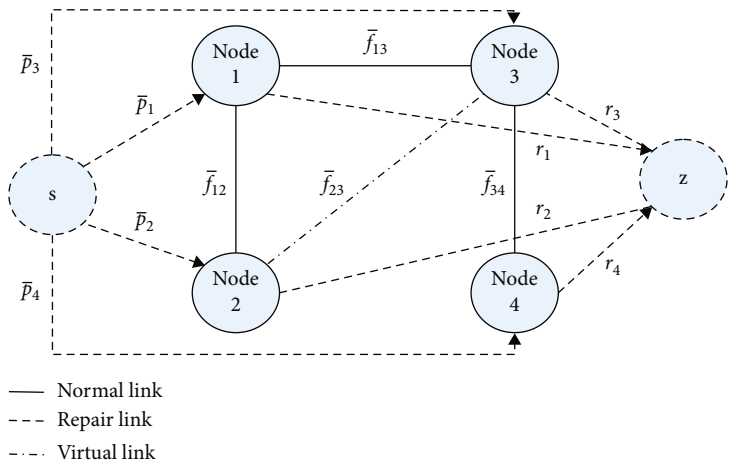

Figure 2: Subproblem equivalent maximum flow problem.

to be constructed. In order to form the optimal cut plane of Benders, it is necessary to use the complementary relaxation principle of the dual problem [33]. According to the solution of the above subproblems, extract the dual variable $\mu_{ij}^t, ij \in E^0$, corresponding to the constraint in Eq. (3), which can be repaired system increment of link ij.

Theorem 1. *The optimal cut plane of Benders in the tth iteration can be expressed as*

$$\eta \geq \theta^t + \sum_{ij \in E^0} \mu_{ij}^t \bar{f}_{ij} \left(e_{ij} - e_{ij}^t \right). \tag{11}$$

Among them, η is the upper bound of the optimal solution of the objective function of the subproblem, and θ^t is the optimal value of the objective function obtained in the tth iteration of the subproblem, that is, the minimum total cost of data discarding in the tth iteration.

Proof. See Appendix A.1. □

Substituting the above Benders optimal cutting plane into the constraints of solving the main problem, the main problem can be obtained as

$$\begin{aligned} M: \min_{e,\eta} \underline{\varnothing} &= \sum_{ij \in E^0} c_{ij} e_{ij} + \eta. \\ \text{s.t.Eqs.}(4), (9). \end{aligned} \tag{12}$$

Among them, $\underline{\varnothing}$ represents the lower bound of the optimal value of the original problem P, because the main problem only considers part of the constraints and is a relaxation problem of the original problem.

Different from the subproblem that determines the data migration, calculation, and discarding of the edge network, the main problem is responsible for determining the link set E^r that needs to be repaired preferentially in the damaged link set E^0. In the loop iteration process of the main and subproblems, the solution of the main problem is carried out given the amount of data migration, calculation, and discarding. The E^r obtained by solving the main problem is in turn used for further solving of the subproblems, thereby gradually approximating the optimal data migration, calculation, discarding, and link repair strategies of the system.

3.3. Iterative Path Repair and Computational Migration Algorithm Based on Benders Decomposition Theory. The above main problem is initialized into subproblems, and the loop iteration starts to find the optimal solution. If the decision variables of the main problem cannot satisfy all the constraints in the iterative process, the algorithm is terminated, and the original problem has no solution. Otherwise, the iterative process continues until the optimal configuration of the network is found. The specific steps of the above process are shown in Algorithm 1.

When the algorithm converges, the system will repair the link whose value is 1 according to the e^t calculated by the current iteration. After that, the solution f^t of the subproblem is used to determine the amount of data flow migration between nodes, the actual computing power of each node is adjusted according to p^t, and the data d^t that exceeds the computing power is discarded. It should be noted that in the main problem, η is a continuous variable and e is a discrete variable, and the problem is still a mixed-integer problem. Although it can be solved by algorithms such as genetic ant colony algorithm, the complexity of the algorithm is still very high due to its large search domain space. In addition, in the process of each iteration, the introduction of a new cut plane in the main problem keeps the lower bound $\underline{\varnothing}$ of the algorithm nondecreasing, but there is no similar mechanism to guarantee the monotonicity of the upper bound $\overline{\varnothing}$ of the algorithm. This nonmonotonic constraint bound property will further aggravate the computational time overhead of the above algorithm.

4. Benders Decomposition Acceleration Algorithm Design

In order to solve the problems existing in the iterative path repair and computational migration algorithm based on

Define: $t = 1$, given network parameters $\{c_{ij}, \bar{f}_{ij} | \forall ij \in E\}$, $\{r_i, \bar{p}_i, c_i | \forall i \in N\}$.
Initialization: Upper bound $\overline{\varnothing} = +\infty$ and algorithm lower bound $\underline{\varnothing} = -\infty$, iteration accuracy parameter ε.
1: Cycle
2: Solve the main problem M, if the main problem has no solution, terminate the algorithm, the original problem has no solution. Otherwise, $e^t = (e_{ij}^t, \forall ij \in E^0)$ and the algorithm lower bound $\underline{\varnothing}$
3: Solve the subproblem S, get $\boldsymbol{d}^t = (d_i^t, \forall i \in N)$, $\boldsymbol{p}^t = (p_i^t, \forall i \in N)$, $\boldsymbol{f}^t = (f_{ij}^t, \forall ij \in E)$, $\boldsymbol{\mu}^t = (\mu_{ij}^t, \forall ij \in E^0)$ and the minimum objective function value θ^t
4: Update algorithm upper bound $\overline{\varnothing} = \min\{\overline{\varnothing}, \sum_{ij \in E^0} c_{ij} e_{ij}^t + \theta^t\}$
5: $t = t + 1$
6: Until the algorithm converges, the convergence condition is $(\overline{\varnothing} - \underline{\varnothing})/\overline{\varnothing} < \varepsilon$

Algorithm 1: Bender iterative path repair and computational migration.

Benders decomposition theory proposed in Algorithm 1, this paper further introduces the local branching technique in the iterative solution process [34]. Its main purpose is to find a better upper bound of the problem in each iteration process, in order to realize the inward clamping of the upper and lower bounds and reduce the computational complexity of the main problem.

4.1. Trust Region and Hamming Distance. As mentioned earlier, the Benders decomposition algorithm based on cut planes is not a stable algorithm, and in the early stages of the iteration, the solution of the problem fluctuates widely in different feasible regions, resulting in a slow convergence rate. In the edge computing scenario in the IIoT considered in this paper, a large number of optimization variables introduced by large-scale network damage, especially the introduction of the repair link decision vector e with an initial search space of $2^{|E^0|}$, will make this convergence speed problems are further exacerbated. The trust region is an excellent strategy to address the above-mentioned large-scale fluctuation characteristics. Considering that e in the main problem is a set of 0-1 variables, the Hamming distance can be used to limit the distance between the two iterative solutions.

Assuming that (e^t, d^t, p^t, f^t) is the feasible solution obtained by the tth iteration of the original problem, the set of all 0-1 optimization variables with a value of 1 can be expressed as $E^t = \{e_{ij}^t | e_{ij}^t = 1, e_{ij}^t \in E^0\}$, and then the Hamming distance between the $(t+1)$th iteration and the tth iteration is

$$D\left(e_{ij}^{t+1}, e_{ij}^t\right) = \sum_{ij \in E^t} \left(1 - e_{ij}^{t+1}\right) + \sum_{ij \in E^0 \setminus E^t} e_{ij}^{t+1}. \tag{13}$$

That is, the number of binary variables e changes in the $(t+1)$th iteration relative to the tth iteration.

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

$$D\left(e_{ij}^{t+1}, e_{ij}^t\right) \le \Delta^{t+1}, \tag{14}$$

$$D\left(e_{ij}^{t+1}, e_{ij}^t\right) \ge \Delta^{t+1} + 1. \tag{15}$$

Among them, Δ^{t+1} represents the size of the trust region in the $t+1$)th iteration, and the selection of its value depends on the complexity of the main problem and the volatility requirements of the search range. In the above way, the original problem is naturally divided into two subsolution spaces, and the local branching method is based on this. In the local branching method, Equations (14) and (15) are called the left and right branches, respectively.

4.2. Local Branching. Based on the Hamming distance, the solution space of the original problem P can be divided into two closely connected neighborhood spaces according to the feasible solutions (e^t, d^t, p^t, f^t) obtained by the tth iteration of the Benders decomposition. Let $e^k, k \in K^t$ be the solutions of all e (including feasible solutions and nonfeasible solutions) calculated by the local branch method iteration during the tth iteration of Benders decomposition, where the set of feasible solutions is denoted as L^t $(L^t \subseteq K^t)$. According to the Hamming distance, the original problem P can be branched into two independent left and right branch problems. The left branch problem is

$$P_k : \min_{e,d,f,p} \varnothing = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in N} c_i d_i, \quad \text{s.t. Eqs.}(1) \sim (7), \tag{16}$$

$$D\left(e_{ij}, e_{ij}^k\right) \ge 1, \forall ij \in E^0, k \in K^t, \tag{17}$$

$$D\left(e_{ij}, e_{ij}^l\right) \ge \Delta^t + 1, \forall ij \in E^0, l \in L^t, \tag{18}$$

$$D\left(e_{ij}, e_{ij}^m\right) \le \Delta^t, \forall ij \in E^0. \tag{19}$$

Among them, Equation (17) indicates that the e value that has been compared before is not repeatedly compared, Equation (18) indicates that the current left branch should be the branch of the right branch in the previous branch, and Equation (19) indicates the left branch restriction, and e_{ij}^m is the currently obtained optimal solution. Correspondingly,

the right branch problem can be obtained as

$$\bar{P}_k : \min_{e,d,f,p} \varnothing = \sum_{ij \in E^0} c_{ij} e_{ij} + \sum_{i \in N} c_i d_i, \quad \text{s.t.Eqs.}(1) \sim (7), (15) \sim (17), \tag{20}$$

$$D\left(e_{ij}, e_{ij}^m\right) \geq \Delta^t + 1, \forall ij \in E^0. \tag{21}$$

Here, Equation (18) represents the right branch restriction.

Let $(e^{k+1}, d^{k+1}, p^{k+1}, f^{k+1})$ be the optimal solution of the left branch problem P_k, and the corresponding objective function value is $\varnothing_{k+1}$, and then there is a local branching method algorithm flow as shown in Algorithm 2.

The $T_{\max}$ in the loop condition of Algorithm 2 is to avoid the situation that the branching problem cannot be solved because the Hamming distance is set too large during the calculation process. In a large-scale damaged network, the solution space of the repair link decision e is very large, and choosing a smaller trust region size Δ^t will be more conducive to improving the calculation speed of the left branch problem.

In the iterative process, the strict upper bound $\overline{\varnothing}_k = \min_{k \in K^t}\{\varnothing_k\}$ of the original problem can be obtained. At the same time, since the main difficulty of the original problem P lies in the acquisition of the lower bound, a series of optimal cutting planes generated in each iteration process can also speed up the search speed, so that the Benders decomposition can simultaneously enhance the optimal solution of the problem in the iterative process. Search for upper and lower bounds [35].

4.3. Benders Decomposition Acceleration Algorithm Based on Local Branching Method. The Benders decomposition algorithm accelerated by the local branching method obtained by integrating the local branching algorithm into the Benders decomposition is shown in Algorithm 3.

Similar to Algorithm 1, when Algorithm 3 converges, the system will determine the set of repair links based on the et calculated by the current iteration. Then, according to the calculated values of f^t, p^t, and d^t, the migration, calculation, and discarding of data in the network are determined. Different from Algorithm 1, the Algorithm 3 ensures the strict decrease of the upper bound of the original problem P in the iterative process through the local branching technique. For large-scale damaged networks, this means that in each iteration process, the search space of the link repair decision vector e^t to be optimized for the main problem M is gradually reduced, and the search speed of the solution is gradually accelerated with the increase of the number of iterations.

In the new cuts obtained by each iteration of the above algorithm, not all cuts need to be added to the main problem solving process, and only the deepest cuts can be added (even the cuts with the smallest feasible region). Considering the convergence requirements of the algorithm and the poor performance of the initial stage of the Benders decomposition, it is possible to add trust region constraints only in the initial iteration stage. When the iterations become stable, this restriction is lifted.

It is worth noting that in each branch process, the branch problem $P_1, P_2, \cdots$ has the same structure as the original problem P. Therefore, for each branch problem, Algorithm 1 can be used to solve it. The cut plane obtained from the previous branching problem can also be used to solve the subsequent branching problem.

5. Simulation Results

In this section, the proposed iterative path repair and computational migration algorithm based on Benders decomposition theory (hereinafter referred to as Benders decomposition algorithm) and the iterative path repair and computational migration algorithm based on Benders decomposition acceleration theory based on local branching method (hereinafter referred to as Benders decomposition acceleration performance of the proposed algorithm) are simulated and tested. The performance advantages of the proposed algorithm compared with other benchmark algorithms are verified and analyzed.

5.1. Simulation Parameters and Comparison Algorithms. This paper considers an edge computing network with $N = 50$ nodes, and the maximum computing power $\bar{p}_i$ of each node in the initial stage of network repair is independent of each other and evenly distributed in (10, 25) Gbit. Considering the matching of computing power and computing requirements, the computing requirements r_i of nodes in the initial stage of network repair also obey the uniform distribution on (10, 25) Gbit. Without loss of generality, it is assumed that the number of wired links in the original topology of the edge network is $|E| = 2N$, the network topology is the same as the random network [4], and the data transmission between nodes is reachable everywhere. Considering the fluctuation range of processing power of computing nodes, the link capacity $\bar{f}_{ij}$ obeys a uniform distribution on (5, 15) Gbit. In order to simulate the large-scale damage of the network, in the following, unless otherwise specified, the link damage ratio $q = 75\%$. Assuming that the cost of repairing each damaged link c_{ij} is evenly distributed in $(1 \times 10^4, 2 \times 10^4)$ pesos, the cost c_i of discarding data due to insufficient computing power is 10^4 peso/Gbit.

In order to verify the effect of the damaged network repair mechanism, the proposed algorithm is compared with the following benchmark algorithms.

(1) Random repair algorithm. The damaged links in the network are randomly selected for repair, regardless of the specific topology of the network and the dynamic characteristics of network computing flow migration

(2) Maximum connected graph repair algorithm [15]. Repair all damaged links in the maximum connected graph of the network to ensure that all nodes in the maximum connected graph of the network can reach a strong connected state

Define: $k = 1$, let the initial feasible solution of local branch iteration $(e^k, d^k, p^k, f^k) = (e^t, d^t, p^t, f^t)$, and the corresponding objective function value is $\varnothing_k$. Given Δ^t, the maximum allowable computation time $T_{\max}$ and the iteration accuracy parameter ε
1: Cycle
2: Divide the current branch into left branch P_k and right branch $\bar{P}_k$, calculate $\varnothing_{k+1}$ and $(e^{k+1}, d^{k+1}, p^{k+1}, f^{k+1})$
3: If $\varnothing_{k+1} < \varnothing_k$ holds
4: Then update $L^t = L^t \bigcup \{k\}$, and jump to the right branch
5: Otherwise, $\varnothing_{k+1} \geq \varnothing_k$ or P_k has no solution
6: Let $\Delta^t = \Delta^t + 1$, add $D(e_{ij}, e_{ij}^k) \geq 1, \forall ij \in E^0$ to the constraint Eq. (15), and update $K^t = K^t \bigcup \{k\}$. Recalculate $(e^{k+1}, d^{k+1}, p^{k+1}, f^{k+1})$ and $\varnothing_{k+1}$ of the left branch problem P_k, go back to step 3)
7: $k = k + 1$
8: Until the algorithm computation time reaches $T_{\max}$ or meets the accuracy requirement $(\varnothing_k - \varnothing_{k+1})/\varnothing_k < \bar{\varepsilon}$

Algorithm 2: Local branch scheme.

Define: $t = 1$, given network parameters $\{c_{ij}, \bar{f}_{ij} | \forall ij \in E\}$, $\{r_i, \bar{p}_i, c_i | \forall i \in N\}$.
Initialization: Upper bound $\overline{\varnothing} = +\infty$ and algorithm lower bound $\underline{\varnothing} = -\infty$, iteration accuracy parameter ε
1: Cycle
2: Solve the main problem M, if the problem has no solution, terminate the algorithm, and the original problem has no solution. Otherwise, $e^t = (e_{ij}^t, \forall ij \in E^0)$ and the algorithm lower bound $\underline{\varnothing}$
3: Solve the subproblem S, and get $d^t = (d_i^t, \forall i \in N)$, $p^t = (p_i^t, \forall i \in N)$, $f^t = (f_{ij}^t, \forall ij \in E)$, the dual variable $\mu^t = (\mu_{ij}^t, \forall ij \in E^0)$ and the minimum objective function value θ^t
4: Update upper bound $\overline{\varnothing} = \min\{\overline{\varnothing}, \sum_{ij \in E^0} c_{ij} e_{ij}^t + \theta^t\}$
5: if $(\overline{\varnothing} - \underline{\varnothing})/\overline{\varnothing} < \varepsilon$
6: Determine the optimal solution, then terminate the algorithm
7: Otherwise, generate a new Benders optimal cutting plane
8: Run Algorithm 2 to obtain an enhanced upper bound $\overline{\varnothing}_k$, and a series of Benders optimal cuts generated by different feasible solutions
9: $t = t + 1$
10: Until the algorithm converges

Algorithm 3: Iterative path repair and computational migration.

(3) Betweenness centrality sorting repair algorithm [11]. The topological structure of the edge network before the damage is analyzed, the betweenness centrality of each wired link is sorted, and the damaged link with large betweenness centrality is preferentially repaired. The number of repaired links is the same as the proposed algorithm

5.2. Algorithm Convergence. Figure 3 shows the convergence comparison between the proposed Benders decomposition algorithm and the Benders decomposition acceleration algorithm. It can be seen from Figure 3 that the two algorithms can achieve fast iteration within a limited number of times and have good convergence. It should be noted that the number of iterations represents the total number of loop iterations of the main problem and subproblems in Algorithm 1 and Algorithm 3. In each iteration process, the computational complexity of the subproblem is $\mathcal{O}(|N||E|^2)$, and the computational complexity of the main problem is $\mathcal{O}(|N||E^0|^3)$, where n is the inner loop iteration for solving the main problem frequency. Although the Benders decomposition acceleration algorithm reduces the total number of iterations only once compared to the Benders decomposition algorithm, its actual computational time complexity is reduced by $(|N||E|^2 + n|E^0|^3)$. In a large-scale compromised network environment, the improvement of the computational time complexity is still considerable. From the comparison of the upper and lower bounds of the algorithm under the same number of iterations, it can be seen that, compared with the Benders decomposition algorithm, the local branch method-assisted Benders decomposition acceleration algorithm has a significantly improved convergence speed due to the introduction of deeper cutting planes and can be better adapt to the application requirements of large-scale networks. In addition, the total system cost of the Benders decomposition algorithm and the Benders decomposition acceleration algorithm tend to be consistent after convergence. Therefore, in the following, for the convenience of comparison with the benchmark algorithm, the Benders decomposition algorithm and the Benders decomposition acceleration algorithm are collectively referred to as the proposed algorithm.

5.3. Algorithm Performance. Figures 4 and 5 are the total system cost curves of the proposed algorithm, the random repair algorithm, the maximum connected graph repair

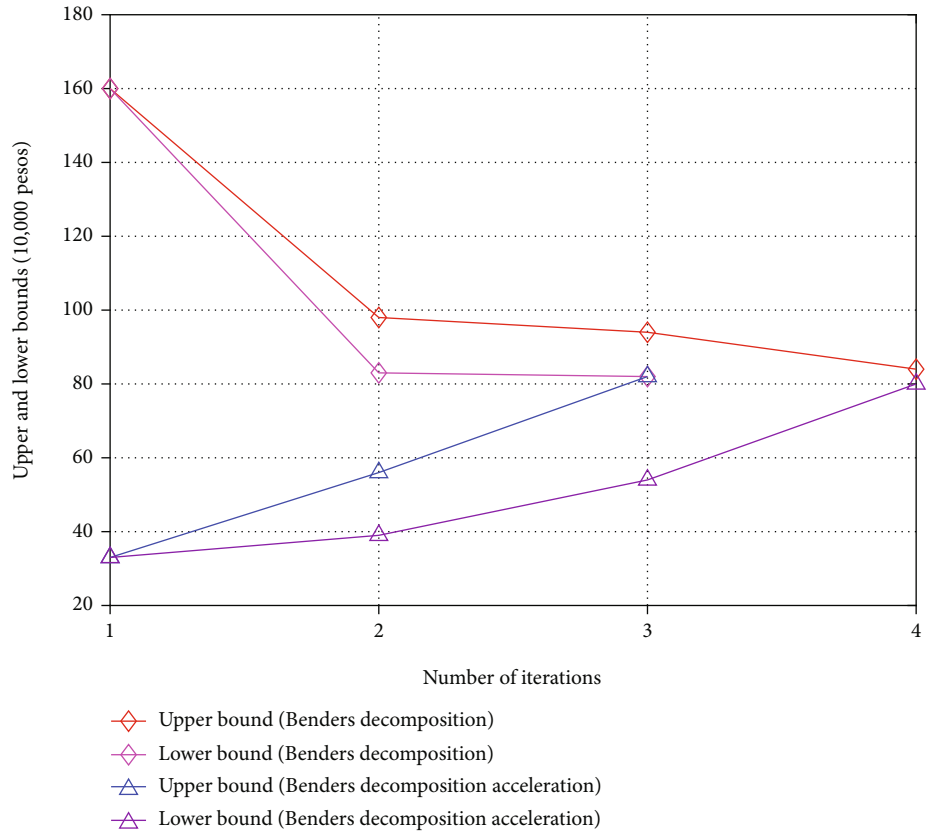

Figure 3: Convergence comparison between the proposed Benders decomposition algorithm and the Benders decomposition acceleration algorithm.

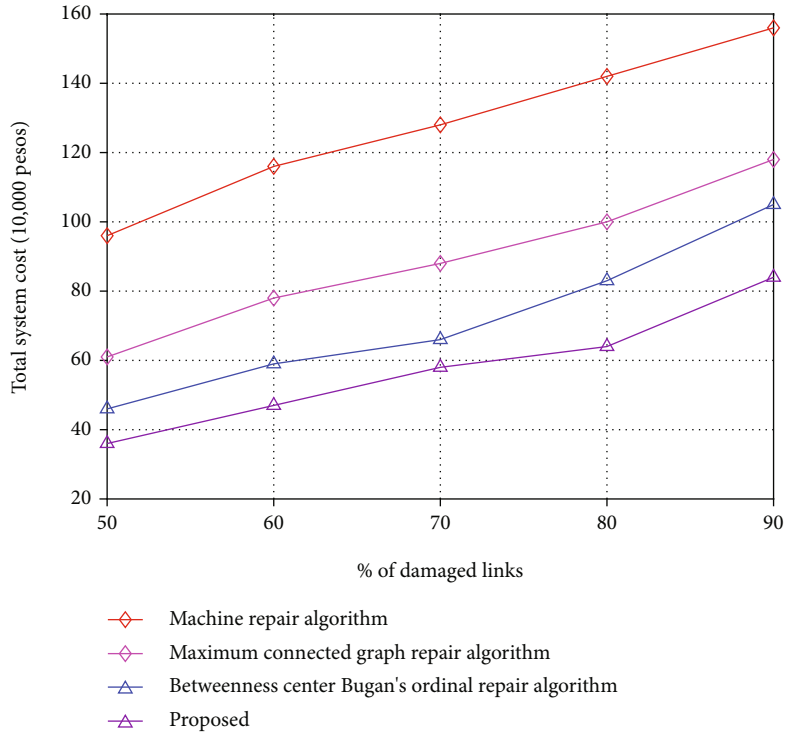

Figure 4: Algorithm performance comparison under different network damage levels.

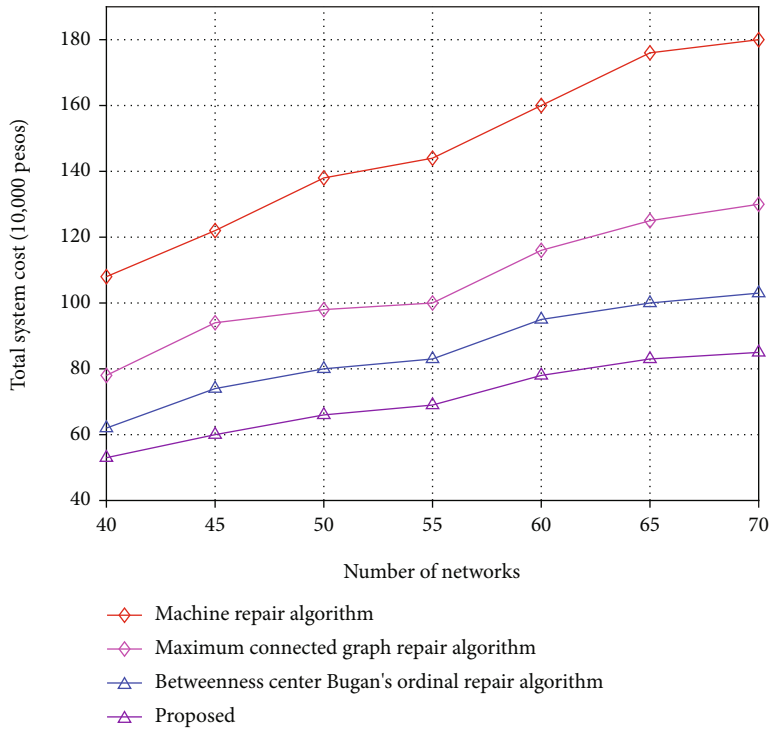

Figure 5: Algorithm performance comparison under different network scales.

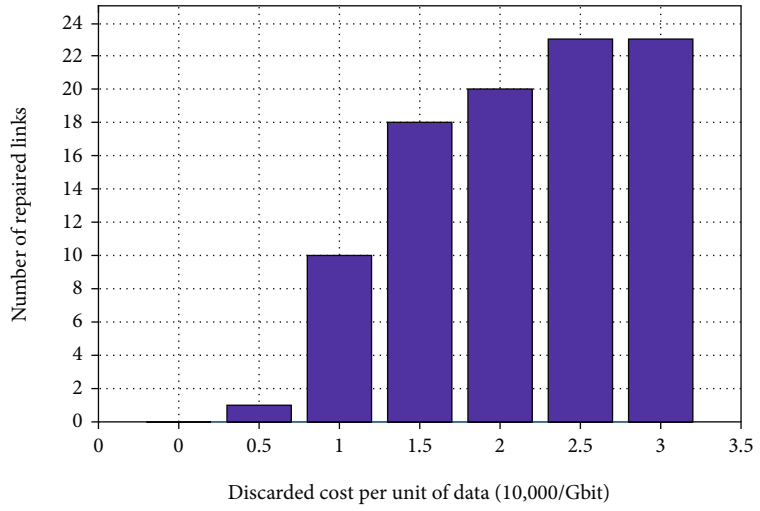

Figure 6: The number of damaged links repaired by the proposed algorithm under different unit data discarding costs.

algorithm, and the betweenness centrality sorting repair algorithm under different network damage degree q and different network scales N, respectively. It can be seen from Figures 4 and 5 that under different network damage degrees and network scales, the performance of the proposed algorithm is excellent due to the consideration of the network dynamic characteristics such as the computing demand and actual computing power of the nodes in the network and link capacity limitations on the benchmark algorithm. It is worth noting that the system overhead performance of the maximum connected graph repair algorithm is the worst among all algorithms, even weaker than the random repair algorithm, which is in good agreement with the state of network repair in real networks. In the real network, the repair process always makes the large and small independent clusters form in the network first, and at the end of the repair, the clusters are connected to form the maximum connected graph [13].

5.4. Multiscenario Deployment. Figures 6 and 7, respectively, show the number of damaged links repaired by the proposed algorithm and the amount of data discarded under different

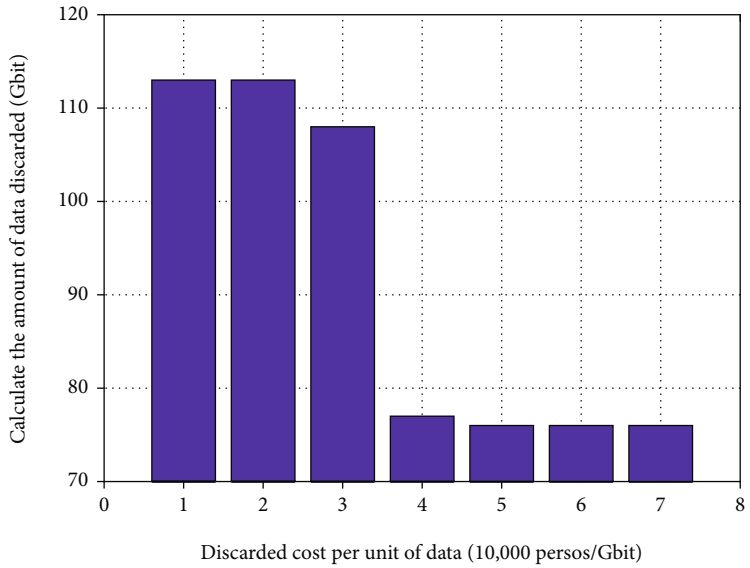

Figure 7: The amount of data discarded under different unit data discarding costs of the proposed algorithm.

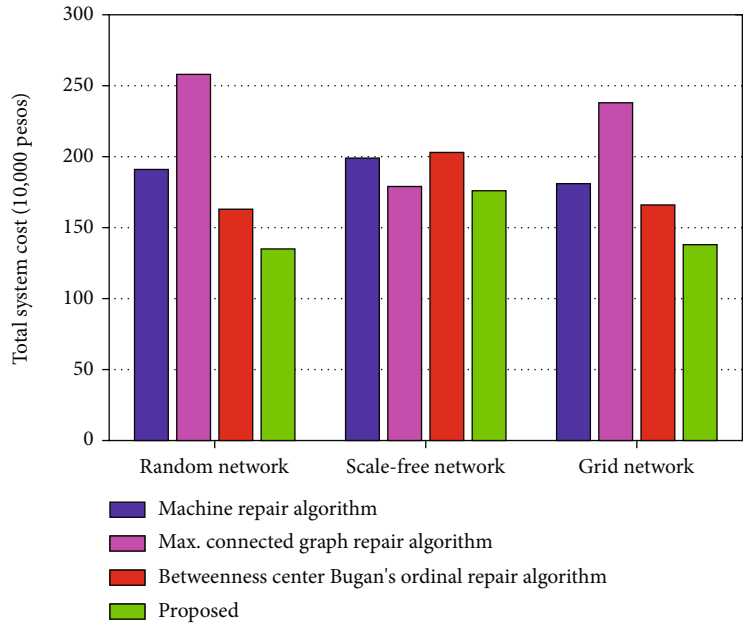

Figure 8: Comparison of algorithms under different network topologies with $N = 100$.

unit data discarding cost scenarios. Considering the double consideration of the total system overhead of the proposed algorithm for link repair cost and data discarding, with the increase of unit data discarding cost c_i, the number of repaired links and the amount of data discarding show an opposite increase or decrease relationship. Consistent with experience, as c_i increases, i.e., the cost of data discarding increases, the network tends to repair more links so that the computational demands can be met. When the value of c_i is (1, 1.5) million/Gbit, the amount of data discarded in Figure 7 begins to drop significantly, because within this range, the system is just near the sensitive critical point. Around this value, the system has the strongest ability to balance the cost of repairing with the overhead of discarding data. Any small change in the c_i value may bring about a larger change in equilibrium decision-making. After the c_i exceeds 20,000 pesos/Gbit, the marginal benefit brought by continuing to repair damaged links is reduced, and the number of network repaired links and the amount of computing data discarded remain stable. It can be seen that the proposed algorithm can be well applied to a variety of different data discarding cost scenarios and has good scalability and adaptability.

In order to further analyze the performance of the algorithm in this paper in different network topology scenarios, this paper extends the network form from a single random network scenario to a grid network and a scale-free network [36]. Different from the random network, the degree of each node of the grid network is exactly the same, and the degree of the nodes of the scale-free network obeys a power-law distribution. As can be seen from Figure 8, under the three types of networks, the performance of each algorithm will fluctuate to varying degrees, but the overall system overhead performance of the algorithm in this paper is better than the benchmark algorithm. Thanks to the joint analysis of network topology and dynamic features, the algorithm in this paper has good adaptability to multiple scenarios.

In summary, it can be seen from the simulation results that the proposed method outperforms the existing methods in various parameters evaluation and different scenarios. This provides solid basis that the proposed approach has practical value in IIoT deployment.

6. Conclusion

Aiming at the vulnerable characteristics of edge computing networks in the IIoT, this paper proposes a repair mechanism after large-scale damage to the network and gives a joint optimization method of priority repair link set decision and computing migration configuration, which effectively alleviates the problem of the initial stage of network repair; there is a conflict between limited repair resources and a large amount of data computing requirements. Considering the difficulty of solving the original problem, the Benders decomposition algorithm is used to transform it into a main problem and subproblems that are easier to solve. Furthermore, combined with the local branching method, a Benders decomposition acceleration algorithm is designed, which effectively improves the convergence speed of the algorithm. The simulation results show that the proposed algorithm has better system repair performance compared to the existing topology-based repair algorithms.

Although for the sake of data fairness, this paper considers that the computing data has the same processing priority, that is, the cost per unit of data discarded by different nodes is the same. The subproblem solution can be replaced with the existing minimum cost flow algorithm (such as Dinic's algorithm).

The proposed algorithm is mainly aimed at the situation where the nodes are connected by wired links. For scenarios with dynamic wireless links, such as the scenario where the UAV acts as a mobile edge node, the channel capacity changes with the location of the UAV, and issues such as the UAV's trajectory and wireless spectrum allocation need to be further considered jointly. This will make the already complex network repair problem more difficult to solve, which is beyond the scope of this paper and is reserved for follow-up research.

Appendix

A.1. Benders Optimal Cut Proof

When $e_{ij}^t, ij \in E^0$ is given, from the original problem P, we can get:

$$\theta\left(e_{ij}^t\right) = \min_{d,f,p}\left\{\sum_{i\in N} c_i d_i \mid (1) \text{ to } (3), (5) \text{ to } (7)\right\}. \tag{A.1}$$

The objective function of the equivalent dual problem of the above problem can be written as:

$$\theta\left(e_{ij}^t\right) = \max_{\alpha,\beta,\lambda,\mu}\left\{\sum_{i\in N}(\alpha_i r_i + \beta_i \bar{p}_i) + \sum_{ij\in E^1}\lambda_{ij}\bar{f}_{ij} + \sum_{ij\in E^0}\mu_{ij}\bar{f}_{ij}e_{ij}^t\right\} \tag{A.2}$$

Among them, $\alpha = (\alpha_i, \forall i \in N), \beta = (\beta_i, \forall i \in N), \lambda = (\lambda_{ij}, \forall i\, j \in E^1)$ optimizes variables (dual variables) for the four sets of dual problems. In the tth iteration process, it is assumed that the solution of the above dual variable set is $\alpha = (\alpha_i^t, \forall i \in N), \beta^t = (\beta_i^t, \forall i \in N), \lambda^t = (\lambda_{ij}^t, \forall ij \in E^1)$, the optimal solution of the problem $\theta^t = \theta(e_{ij}^t)$, and then we have

$$\theta(e_{ij}) \geq \sum_{i\in N}\left(\alpha_i^t r_i + \beta_i^t \bar{p}_i\right) + \sum_{iij\in E^1}\lambda_{ij}^t \bar{f}_{ij} + \sum_{ij\in E^0}\mu_{ij}^t \bar{f}_{ij}e_{ij}. \tag{A.3}$$

From linearization operation around $e_{ij}^t, ij \in E^0$, we get:

$$\begin{aligned}\eta \geq \theta(e_{ij}) &\geq \sum_{i\in N}\left(\alpha_i^t r_i + \beta_i^t \bar{p}_i\right) + \sum_{iij\in E^1}\lambda_{ij}^t \bar{f}_{ij} + \sum_{ij\in E^0}\mu_{ij}^t \bar{f}_{ij}e_{ij} \\ &+ \sum_{ij\in E^0}\mu_{ij}^t \bar{f}_{ij}\left(e_{ij} - e_{ij}^t\right) = \theta^t + \sum_{ij\in E^0}\mu_{ij}^t \bar{f}_{ij}\left(e_{ij} - e_{ij}^t\right).\end{aligned} \tag{A.4}$$

Data Availability

The data used for the findings of this study is available upon from the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

Authors' Contributions

Conceptualization was contributed by Hongyang Huang and Imran Khan; data curation was contributed by M. Dauwed and M. Derbali; formal analysis was contributed by Sun Li and Kai Chen; funding acquisition was performed by Sangsoon Lim; supervision was performed by Imran Khan and Sangsoon Lim.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2021R1F1A1063319).

References

[1] J. Lu, L. Chen, J. Xia et al., "Analytical offloading design for mobile edge computing-based smart internet of vehicle," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, no. 1, p. 10, 2022.

[2] L. Zhang, W. Zhou, J. Xia et al., "DQN-based mobile edge computing for smart internet of vehicle," *EURASIP Journal on Advances in Signal Processing*, vol. 3, 10 pages, 2022.

[3] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: a survey," *Journal of Network and Computer Applications*, vol. 202, no. 5, article 103366, 2022.

[4] L. Xing, "Cascading failures in internet of things: review and perspectives on reliability and resilience," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 44–64, 2021.

[5] S. Ayoubi, C. Assi, Y. Chen, T. Khalifa, and K. B. Shaban, "Restoration methods for cloud multicast virtual networks," *Journal of Network and Computer Applications*, vol. 78, no. 3, pp. 180–190, 2017.

[6] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Compute Systems*, vol. 70, no. 8, pp. 138–147, 2017.

[7] R. Teng, H. Li, and R. Miura, "Dynamic recovery of wireless multi-hop infrastructure with the autonomous mobile base station," *IEEE Access*, vol. 4, pp. 627–638, 2016.

[8] P. Li and X. Yang, "On dynamic recovery of cloud storage system under advanced persistent threats," *IEEE Access*, vol. 7, pp. 103556–103569, 2019.

[9] G. J. Baxter, G. Timár, and J. F. F. Mendes, "Targeted damage to interdependent networks," *Physical Review E*, vol. 98, no. 3, pp. 3328–3339, 2018.

[10] C. D. Brummitt, R. M. D'Souza, and E. A. Leicht, "Suppressing cascades of load in interdependent networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. 12, pp. 680–689, 2012.

[11] F. Morone and H. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.

[12] H. Rudnick, S. Mocarquer, E. Andrade, E. Vuchetich, and P. Miquel, "Disaster management," *IEEE Power and Energy Magazine*, vol. 9, no. 2, pp. 37–45, 2011.

[13] G. Punzo, A. Tewari, E. Butans et al., "Engineering resilient complex systems: the necessary shift toward complexity science," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3865–3874, 2020.

[14] J. Moon, M. Yang, and J. Jeong, "A novel approach to the job shop scheduling problem based on the deep Q-network in a cooperative multi-access edge computing ecosystems," *Sensors*, vol. 21, no. 4, pp. 1–17, 2021.

[15] M. Oudani, "A simulated annealing algorithm for intermodal transportation on incomplete networks," *Applied Sciences*, vol. 11, no. 10, article 4467, 2021.

[16] A. Hamed, M. Alkinani, and M. Hassan, "A genetic algorithm to solve capacity assignment problem in a flow network," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1579–1586, 2020.

[17] A. Ranjbari, M. Hickman, and Y. Chiu, "A network design problem formulation and solution procedure for intercity transit services," *Transportmetrica A: Transport Science*, vol. 16, no. 3, pp. 1156–1175, 2017.

[18] D. Li, Q. Zhang, E. Zio, S. Havlin, and R. Kang, "Network reliability analysis based on percolation theory," *Reliability Engineering & Systems Safety*, vol. 142, no. 5, pp. 556–562, 2015.

[19] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 574–586, 2018.

[20] A. Smith, M. Posfai, and M. Rohden, "Competitive percolation strategies for network recovery," *Scientific Reports*, vol. 9, no. 1, pp. 965–983, 2019.

[21] Z. Hong, W. Chen, W. Huang, S. Guo, and Z. Zheng, "Multi-Hop cooperative computation offloading for industrial IoT-edge-cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[22] Y. Yu, X. Bu, K. Yang, Z. Wu, and Z. Han, "Green large-scale fog computing resource allocation using joint benders decomposition, Dinkelbach algorithm, ADMM, and branch-and-bound," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4106–4117, 2019.

[23] X. Li, J. Wan, H. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[24] Z. H. Abbas, Z. Ali, G. Abbas et al., "Computational offloading in mobile edge with comprehensive and energy efficient cost function: a deep learning approach," *Sensors*, vol. 21, no. 10, article 3523, 2021.

[25] M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, and S. Papavassiliou, "ENERDGE: distributed energy-aware resource allocation at the edge," *Sensors*, vol. 22, no. 2, p. 660, 2022.

[26] G. Codato and M. Fischetti, "Combinatorial Benders' cuts for mixed-integer linear programming," *Operations Research*, vol. 54, no. 4, pp. 756–766, 2006.

[27] R. Rahmaniani, T. Crainic, M. Gendreau, and W. Rei, "The Benders decomposition algorithm: a literature review," *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, 2017.

[28] J. Tanveer, A. Haider, R. Ali, and A. Kim, "An overview of reinforcement learning algorithms for handover management in 5G ultra-dense small cell networks," *Applied Sciences*, vol. 21, no. 1, pp. 1–14, 2022.

[29] M. F. Pereira, L. V. G. Pinto, S. F. Cunha, and G. Oliveira, "A decomposition approach to automated generation/transmission expansion planning," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-104, no. 11, pp. 3074–3083, 1985.

[30] S. Latif, M. Driss, W. Boulila et al., "Deep learning for the industrial internet of things (IIoT): a comprehensive survey of techniques, implementation frameworks, potential applications, and future directions," *Sensors*, vol. 21, no. 22, article 7518, 2021.

[31] F. Oliveira, I. Grossmann, and S. Hamacher, "Accelerating Benders stochastic decomposition for the optimization under

uncertainty of the petroleum product supply chain," *Computers & Operations Research*, vol. 49, no. 6, pp. 47–58, 2014.

[32] E. Vega, R. Soto, B. Crawford, J. Peña, and C. Castro, "A learning-based hybrid framework for dynamic balancing of exploration-exploitation: combining regression analysis and metaheuristics," *Mathematics*, vol. 9, no. 16, article 1976, 2021.

[33] M. Fischetti and A. Lodi, "Local branching," *Mathematical Programming*, vol. 98, no. 1-3, pp. 23–47, 2003.

[34] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro, "A stochastic programming approach for supply chain network design under uncertainty," *European Journal of Operational Research*, vol. 167, no. 1, pp. 96–115, 2005.

[35] M.-X. Lu, G.-Z. Du, and Z.-F. Li, "Multimode gesture recognition algorithm based on convolutional long short-term memory network," *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 4068414, 9 pages, 2022.

[36] A. Yu, N. Wang, and N. Wu, "Scale-free networks: characteristics of the time-variant robustness and vulnerability," *IEEE Systems Journal*, vol. 3, no. 99, pp. 1–11, 2020.