Received January 11, 2021, accepted January 23, 2021, date of publication January 27, 2021, date of current version February 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054954

An Optimal Tree-Structured Repair Scheme of Multiple Failure Nodes for Distributed Storage Systems

ANAN ZHOU, BENSHUN YI, YUSHENG LIU, AND LAIGAN LUO
School of Electronic Information, Wuhan University, Wuhan 430072, China

Corresponding author: Benshun Yi (yibs@whu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61371125, and in part by the Shenzhen Basic Research Project under Grant JCYJ20150630153917254.

ABSTRACT To reduce recovery cost of repairing multiple failed nodes, many repair schemes have been proposed for erasure codes based distributed storage systems. However, most of the existing researches ignore the network topology of storage devices. Motivated by such considerations, we combine delay repair schemes with network topology and propose a tree-structured model based on fountain codes with large value of (n, k, r) to improve the repair efficiency. More precisely, with the consideration of network topology, a new target named data recovery cost is defined to measure the efficiency of coded fragment download and source file reconstruction, and then the optimal recovery threshold is derived to minimize the average data recovery cost of general tree-structured model. Moreover, we analyze and compare the average data recovery cost of general tree-structure with different systematic parameters. To further improve the data transmission efficiency, an optimal tree-structured scheme based on improved tabu search algorithm (ITSA-ORT) is proposed. Compared with other algorithms, the ITSA-ORT scheme uses Prim algorithm to generate the initial solution and then uses special method to obtain the corresponding neighborhood structure. The experimental results show that the proposed scheme can find a globally optimal solution and obtain lower cost of data recovery. In addition, the ITSA-ORT scheme has lower computational complexity than the optimal tree-structured scheme based on particle swarm optimization algorithm (PSO-ORT) and the optimal tree-structured scheme based on firefly algorithm (FA-ORT).

INDEX TERMS Centralized communication model, fountain codes, delay repair, network topology, tree-structured model, optimization algorithms.

I. INTRODUCTION

With the arrival of the big data era, the reliability of stored data becomes a critical issue in large-scale distributed storage systems (DSSs). However, since the DSSs are usually viewed as a collection of a large number of inexpensive commercial hardware, failure events are the norm rather than exception. To guarantee the high reliability of stored data, replication and erasure coding are used to provide fault tolerance [1]. Compared to replication, erasure coding provably achieves the same degree of fault tolerance while incurring much less redundancy [2]. Because of the high storage efficiency and reliability, most current production storage systems deploy erasure coding to maintain stored data, such as Windows Azure Storage [3] and Facebook's Hadoop Distributed File System [4].

A plethora of work on the design of new types of erasure codes have appeared in recent years. The pioneering work in [5] proposed regenerating codes that optimally deal with the repair bandwidth problem. Following the [5], there has been a growing literature focused on designing new regenerating codes, such as the minimum storage regenerating (MSR) codes and the minimum bandwidth regenerating (MBR) codes [6]. Moreover, there are also several other erasure codes for DSSs [7]–[10]. The aforementioned references mainly concentrate on the single failure problem while the erasure coding schemes typically use a small value of (n, k, r), which is called small codes. The DSSs based on

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

small codes, which are called small code systems, therefore distribute encoded data for each object across a relatively small number of storage nodes. For the small code systems, it is necessary to use instantaneous repair strategy to achieve a large *Mean Time to Data Loss* (MTTDL), which is often used to characterize the durability of stored data. However, the frequent quick repair will demand a large amount bandwidth resources, and therefore result in the repair bottleneck. To the best of our knowledge, the practical DSS is usually composed of a large number of storage nodes, in which often occurs multiple nodes failed simultaneously. To solve the repair bottleneck, on the one hand, some scholars proposed to deploy large codes (i.e. the codes with large value of (n, k, r)) into large-scale DSS to meet the practical requirements. On the other hand, delay repair strategy was proposed to mitigate the repair penalty, thus avoiding some unnecessary instantaneous repairs, e.g. the repair of transient node failures.

Fountain codes (FCs) are a class of erasure codes with several advantages, including low encoding/decoding complexity, rate adaptation, near maximum distance separable (near-MDS) property, and limited feedback [11], [12]. Especially, for the FCs, the larger the message length is, the better the performance of FCs is. Therefore, when a large amount of data need to be stored, the FCs can achieve better performance than other erasure codes in improving data reliability for large-scale DSSs. As the first practical realization of FCs, Luby transform (LT) codes have been frequently used in data storage of DSS [13]–[18]. In order to improve the practicability, we take the LT codes with large value of (n, k, r) as the coding scheme in DSS, which we call large LT codes. At the same time, the delay repair strategy is used to slowly repair data lost from failed nodes.

The idea of delay repair is to decrease the repair rate, thereby reducing the required bandwidth traffic, but without significant impact on durability. Unlike the instantaneous repair, the delay repair triggers the data repair after a threshold number of storage nodes are failed, rather than triggers at the first failed node. Thus, it is a modified version of instantaneous repair. Delay repair strategies for different DSSs are investigated in many literatures recently [19]–[23]. They mainly investigate the optimal repair threshold for minimizing the communication cost of transmitting data during downloading and repairing process. However, the aforementioned references are all ignores the network topology of storage nodes, which plays an important role in large-scale DSS.

As a crucial role in DSS, network topology has a significant impact on the repair performance of failed nodes. Several works have been proposed that take network topology into account while doing repairs. Gerami *et al.* [24] and Qin and Li [25] analyzed the optimal repair cost with consideration of network topology in DSS and investigated the impact of network topology in repairs. In [26], the authors proposed a general framework to find the optimal cost feasible recovery in a dynamic network topology and investigated the gains for different codes by utilizing the network aware. In addition, there are some special network topologies which are used to promote the repair efficiency, such as in literatures [27]–[31]. Therefore, network topology plays an important role in DSSs, which inspires us to explore and exploit the relationships for failure recovery.

In this paper, we consider a large-scale distributed storage system, which achieves the reliability of stored data by deploying LT codes with a large value of (n, k, r). With the consideration of centralized repair model, the regenerative information can be transmitted to all replacement nodes by a broadcast network. A new metric named data recovery cost is defined to measure the data download and file reconstruction efficiency. Different from [20], we combine the delay repair strategy with storage nodes' topology and propose the tree-structured model to calculate the data recovery cost by using the transmitted data and network link weights of tree-structured model. In addition, the optimal repair threshold is derived to minimize the average data recovery cost of general repair tree structure in centralized communication model. Furthermore, an optimal repair tree structure is constructed by using improved approximation algorithm, since the optimal repair tree problem is NP-hard in a large-scale distributed storage system. Therefore, the data recovery cost can be reduced through the proposed scheme. To summarize, we make the following contributions.

(1) A two-dimensional Markov chain model is established to investigate the delay repair strategies in centralized communication model. In the centralized model, the leader node first reconstructs the source file by downloading coded symbols from providers, then disseminates the repaired fragments to replacement nodes by a broadcast network.

(2) For the general repair tree-structured model, the data recovery cost is calculated and the optimal repair threshold is derived to minimize the average data recovery cost per unit of time. With the Consideration of nodes' topology, the repair tree models are constructed among the surviving nodes (called providers), leader node and replacement nodes, where the replacement nodes are root nodes of the repair trees.

(3) We further propose a novel optimal repair tree-structured scheme, named improved TSA algorithm based optimal repair tree scheme (ITSA-ORT). Simulation results show that the proposed scheme has a better performance than other traditional optimization algorithms.

The remainder of the paper is organized as follows. In Section II, we present some related work and give some definitions. Section III analyzes the delay repair strategies with two-dimensional Markov chain model and then derives the optimal repair threshold for general repair tree model. In Section IV, we design the improved TSA algorithm based optimal repair tree scheme and verify its significant improvements by simulation. Finally, we conclude the paper in Section V.

II. RELATED WORK

In this section, we provide some of necessary background that are starting point of this paper, including large LT codes and

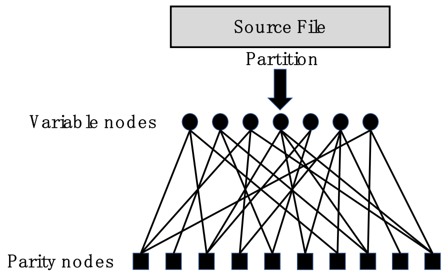

FIGURE 1. The encoding process of LT codes.

several system models. Moreover, some relative definitions are also provided.

A. LARGE LT CODES

In this subsection, we provide a brief view on large LT codes, which is used as a redundancy coding scheme. We define the large LT codes with LT($\Omega(x)$, n, k), where k is the number of input message fragments, and it is regarded as the block length; n is the number of output coded fragments, and $\Omega(x) = \sum_{d=1}^{k} \Omega_d x^d$ is the node degree distribution, where Ω_d denotes the probability that one coded fragment has a degree d. The redundancy overhead is defined as $r = n - k$. The factor graph describing the large LT encoding process is shown in Fig.1, which consists of the parity nodes (squares) and variable nodes (circles). Moreover, the receiver can recover the source file of k fragments through receiving any $(1+\varepsilon)k$ from n coded fragments, with an arbitrarily small decoding overhead of ε.

In general, the error-correction performance of LT code is directly proportional to the block length. Therefore, the LT codes with a large value of (n, k, r) can achieve high file maintenance performance in large-scale DSSs. Without any loss of generality, we assume that each storage device (i.e., storage node) stores a coded fragment. It means that the n coded fragments would be stored into n storage nodes, respectively. In this paper, we exploit the advantage of large LT codes to ensure the reliability of stored data during transmission. It must be noted that, in order to guarantee the encoding/decoding performance of large LT codes, we select the combined Poisson robust soliton distribution (CPRSD) as the degree distribution for large LT codes [32].

B. SYSTEM MODEL

1) CENTRALIZED REPAIR MODEL

We construct a centralized communication model by considering the network topology between storage nodes, where all data regenerations are done at a leader node [33], [34]. More precisely, the leader node downloads sufficient fragments from surviving nodes, named providers, and reconstructs the source file. Then, the leader node transmits repaired fragments to each of replacement nodes by the way of broadcasting. A simple example is shown in Fig.2.

In Fig.2, the green circular indicates the leader node, the red circulars denote the replacement nodes, and the black

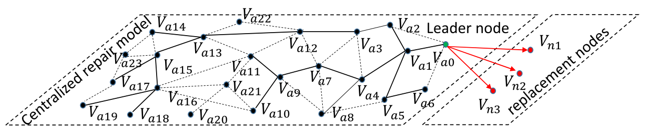

FIGURE 2. The centralized communication model.

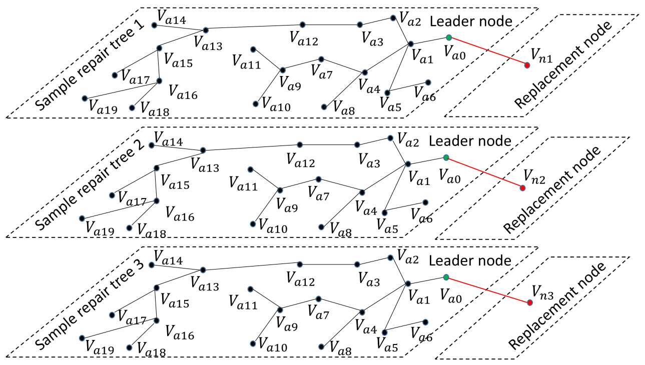

FIGURE 3. Three samples of repair tree.

circulars represent the provider nodes. In addition, the solid lines indicate the selected network links for data transmission. In such centralized scenarios, we assume only the source file of F bits is stored in the storage system. For simplicity, it is assumed that the dissemination of coded fragments among storage nodes had been finished in advance. The data are uniformly allocated, and the amount of stored data for each storage node are $\alpha = \frac{F}{k}$. We further suppose the replacement nodes have been given, and the surviving nodes are distributed in different geographical locations. It means that the network distance between surviving nodes is not always equal. Based on the above assumption, we construct a repair tree-structured model for data recovery in large LT-coded DSS by integrating network topology of storage nodes into repair procedure.

For the repair tree-structured model, the leader node is viewed as a key node, which is responsible for the regeneration and distribution of repaired fragments. In addition, the providers provide enough available coded message for the leader node and the replacement nodes are regarded as the root nodes of repair trees. In such case, the repair tree can be treated as an n-to-1 model, which means that the n independent data flows from branches are gathered at a leader and the leader node can finally transmit the repaired fragments to replacement nodes [35]. Fig.3 shows the construction of three samples of repair tree structures in centralized repair model. It can be seen that multiple repair tree structures can be built with different replacement nodes.

To sum up, the construction of the repair tree structure should satisfy the following principles:

(p1) Provider nodes should be close to the key node, i.e., leader node;

(p2) Leader node should be close to both the providers and root node;

(p3) The path from providers to the root node should be as short as possible.

2) FAILURE-REPAIR MODEL

In order to analyze the delay repair strategy more comprehensively, we define a two-dimensional continuous time discrete stochastic process $\left\{n_a(t), n_f(t)\right\}$, where $n_a(t) = \theta$, $\theta \in \{n-m, \ldots, n\}$ indicates the number of surviving nodes at time t; $n_f(t) = \varrho, \varrho \in \{0, 1, 2, \ldots, m\}$ denotes the number of failures at time t, and the m denotes the repair trigger threshold. Moreover, we indicate the system state of our storage system at time t by the two-dimensional stochastic process $\left\{n_a(t), n_f(t)\right\}$. In our system model, the storage nodes failure follows a Poisson process. Thus, the lifespan of a live node in the system can be modeled by an exponentially distributed random variable with parameter $\theta\lambda$ which denotes the expected survival rate of a node [22], [23], [36], [47]. Therefore, the probability density function (PDF) of the surviving time (lifespan) T_θ is

$$f_{T_\theta}(t) = \theta\lambda \cdot e^{-\theta\lambda t}, \quad i \geq 0, t \geq 0 \tag{1}$$

Similarly, we also can model the nodes repair time T_ϱ as an exponentially distributed random variable with parameter μ. And the PDF of T_ϱ is

$$f_{T_\varrho}(t) = \mu \cdot e^{-\mu t}, \quad \mu \geq 0, \quad t \geq 0 \tag{2}$$

We assume the repair process is performed in parallel at different nodes with the same repair rate μ.

C. SOME RELATIVE DEFINITIONS

Compared with the other ways of data transmission, i.e., from providers to replacement node directly, our repair tree-structured models transmit data through the links of network topology among the providers, leader node and replacement node. Therefore, the repair strategies of literatures [20]–[22] may be not suitable for our models. To solve this problem, some metrics are defined to more accurately indicate the repair cost of the repair tree model. In the following definitions, the V is represented as the set of storage nodes, and E is denoted as the set of edges in the repair tree model.

Definition 1: (The set of storage nodes $\mathrm{V} = \left\{V_{a\eta} | V_{n\xi} | V_{f\xi}\right\}$). Assume that there are n storage nodes in DSS. In addition to m failed nodes, there are $(n-m)$ available nodes left. The $V_{a\eta}$ denotes the surviving nodes, where $\eta = 0, 1, 2, \ldots, n-m$. The replacement nodes are denoted by $V_{n\xi}$, and the failed nodes are indicated by $V_{f\xi}$, where $\xi = 1, 2, \ldots, m$. It must be noted that the node V_{a0} denotes the leader node in centralized repair model.

Definition 2: (The edge set between storage nodes $\mathrm{E} = \left\{\left(V_{ai}, V_{aj}\right) | i, j = 0, 1, 2, \ldots, n-m\right\}$). The $\left(V_{ai}, V_{aj}\right)$ represents the edge between the node V_{ai} and V_{aj}. In addition, $\left(V_{a0}, V_{n\xi}\right)$ denotes the edge between the leader node and replacement nodes, where $\xi = 1, 2, \ldots, m$.

*Definition 3:*Link weights of the edges between storage nodes: $\mathrm{W}\left(V_{ai}, V_{aj}\right), \mathrm{W}\left(V_{a0}, V_{n\xi}\right)$). The $\mathrm{W}\left(V_{ai}, V_{aj}\right)$ represents the link weighted values among the available nodes, where $i, j = 0, 1, 2, \ldots, n-m$. The $\mathrm{W}\left(V_{a0}, V_{n\xi}\right)$ represents the link weighted values between the leader node and replacement nodes, where $\xi = 1, 2, \ldots, m$. In the repair model, we assume the network distance between two storage nodes is not always equal, which means that the link weighted values are not always the same.

Definition 4: (Data recovery cost $c(m)$). It is defined as the sum of the product between the amount of data transmitted and the corresponding link weighted values. It means that the more network links the data passes through, the more network resources it will occupy, e.g., storage nodes and links. The storage nodes and network links are important resources, which will directly influence the performance of network data transmission in DSS. Thus, the data recovery cost $c(m)$ can accurately reflect the efficiency of the failed node repairs in repair tree model.

Definition 5: (Average data recovery cost $r_c(m)$). The $r_c(m)$ is defined as the data recovery cost per unit of time. Since the repair operations can only be initiated when the number of failed nodes reaches the threshold m, it is regarded as an independent and distributed system with a repairing process in every T seconds, where T is a random variable denoting the time interval between two samples of a complete repair system.

For clarity, the summary of used notations in this paper is listed in Table 1.

III. ANALYSIS OF GENERAL REPAIR TREE STRUCTURE IN CENTRALIZED REPAIR MODEL

Let m denote the number of failed nodes in DSS, and the number of available nodes can be obtained by $n-m$. In this section, we mainly focus on the determination of optimal repair threshold m^*, with consideration of storage node's network topology, to minimize the average data recovery cost. Then, we derive the expression of MTTDL of the repair model to represent the durability of stored data.

A. THE COST OF GENERAL REPAIR TREE STRUCTURE

In the centralized repair model, the T_{tree} is used to represent the repair trees composed of a replacement node, a leader node and a certain number of provider nodes. With the given definitions, the repair tree can be denoted by $T_{tree} = \left\{V', E'\right\}$, where $V' = \{V_{a\eta}, V_{n\xi} | \eta = 0, 1, 2, \ldots, n-m, \xi = 1, 2, \ldots, m\}$ indicates the set of the leader nodes, providers, and replacement nodes. and $E' = \left\{\left(V_{ai}, V_{aj}\right)', \left(V_{a0}, V_{n\xi}\right)' | i, j = 0, 1, 2, \ldots, n-m,\right\}$ denotes the edge set of a repair tree. Noted that the E' should satisfy the following conditions.

$$\left(V_{ai}, V_{aj}\right)' = \begin{cases} 1, & if \left(V_{ai}, V_{aj}\right)' \in T_{tree} \\ 0, & otherwise \end{cases} \tag{3}$$

$$\sum\nolimits_{\left(V_{ai}, V_{aj}\right)' \in E'} \left(\left(V_{ai}, V_{aj}\right)'\right) = (1+\varepsilon) k \tag{4}$$

From the Section II, we can know that there are $(1+\varepsilon)k$ surviving nodes need to be accessed for recovering the source file in the leader node. It means that there are $(1+\varepsilon)k+2$

TABLE 1. Symbols used in this paper.

ε	The decoding overhead
F	The size of source file
α	The amount of stored data in each storage node
m	The total number of failed nodes
m^*	The optimal repair threshold of triggering the repairing operation
T_ϱ	The repair time of ϱ failed nodes
T_θ	The lifespan of θ surviving nodes
$n_a(t)$	The number of surviving nodes at time t, i.e., θ
$n_f(t)$	The number of failed nodes at time t, i.e., ϱ
$f_{T_\theta}(t)$	The probability density function of T_θ
$V_{a\eta}$	The surviving nodes (provider nodes), where $\eta = 0,1,2,\ldots,n-m$. Especially, the V_{a0} denotes the leader node.
$V_{n\xi}$	The replacement nodes, where $\xi = 1,2,\ldots,m$
$V_{f\xi}$	The failed node, where $\xi = 1,2,\ldots,m$
(V_{ai}, V_{aj})	The edge between the node V_{ai} and V_{aj}, where $i,j = 0,1,2,\ldots,n-m$
$(V_{a0}, V_{n\xi})$	The edge between the leader node V_{a0} and the replacement node $V_{n\xi}$
$c(m)$	The data recovery cost of repairing m failed nodes
$r_c(m)$	The average value of $c(m)$
T_{tree}	The repair tree
MTTDL	The *Mean Time to Data Loss*
Defined in Algorithm 1	
$\boldsymbol{\mathcal{A}}$	The search space of all solutions
$\boldsymbol{\mathcal{N}(T_{tree})}$	The neighborhood structure of repair tree $\boldsymbol{T_{tree}}$
tl_{min}	The minimum tabu list tenure
tl_{max}	The maximum tabu list tenure
Δ_{tlen}	The increment value of tabu list tenure
tl_{ten}	The tabu list tenure
$iter_{max}$	The maximum number of iteration
T_{tree}^{opt}	The optimal repair tree
T_{tree}^{cur}	The current repair tree
T_{tree}^{ropt}	The reboot-optimal repair tree, i.e., an optimal solution

nodes in one repair tree. In other words, there are a set of repair trees in centralized repair model, and the size of the repair tree set is $m * C_{n-m}^{(1+\varepsilon)k}$. It must be noted that we mainly concentrate on the analysis of general repair tree-structured model in this section, and we will discuss the optimal repair tree structure in next section.

To recover the lost data, we need to complete the following two steps. Firstly, the source file is reconstructed at leader node by independently downloading α symbols from $(1+\varepsilon)k$ provider nodes. Secondly, the leader node distributes the encoded fragments simultaneously to replacement nodes to finish the recovery of lost fragments. Thus,

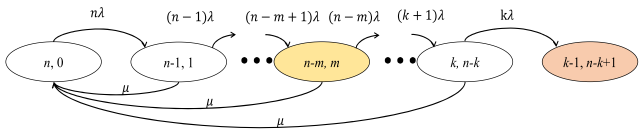

FIGURE 4. Two-dimensional markov chain model for a delay repair strategy.

the data recovery cost during the repair process can be calculated by multiplying the link weighted values and the amount of data transmitted. Finally, the data recovery cost $c(m)$ is expressed as follows.

$$c(m) = \alpha \left[\sum\nolimits_{(V_{ai}, V_{aj})' \in E'} \mathrm{W}\left(V_{ai}, V_{aj}\right)' + \sum\nolimits_{\xi=1}^{m} \mathrm{W}\left(V_{a0}, V_{n\xi}\right) \right] \quad (5)$$

Moreover, to determine the optimal repair threshold m^*, we consider minimizing the average data recovery cost $r_c(m)$, which captures the data recovery cost for maintaining n storage nodes per unit of time.

In order to calculate $r_c(m)$, we establish a two-dimensional Markov chain (TDMC) model shown in Fig.4. This model describes the periodic repair process when storage node failures occur independently, the lifespan of each node is exponentially distributed with parameter $\theta\lambda$, and the failed node recovery process is exponentially distributed with parameter μ.

From Fig.4, it can be seen that the TDMC model is composed of $n-k+2$ states indicating the number of surviving nodes and failed nodes after each node fault. The last state represents that there will not be sufficient available data to recover the source file, when the state changes to $(k-1, \mathrm{n}-\mathrm{k}+1)$. Note that we mainly concentrate on optimizing the periodic recovery cost of repairing the DSS at threshold m, where $m \in [1, n-k]$. Therefore, the system state will return to the state $(n, 0)$ directly when the repair process is initiated at state $(n-m, \mathrm{m})$. It means that the repair processes of all failed nodes are performed in parallel. For the TDMC model, we define the average data recovery cost $r_c(m)$ according to the following formula:

$$r_c(m) = \frac{c(m)}{E[T]} \quad (6)$$

where $E[T]$ denotes the average time between two repair rounds in the periodic repair processes. For T,

$$T = T_{n,0} + T_{n-1,1} +, \ldots, + T_{n-m+1,m-1} + T_{n-m,m} \quad (7)$$

where $T_{\theta,\varrho}$ indicates the time that the system stays at state $\{\theta, \varrho\}$ and $T_{n-m,m}$ denotes the expected time of finishing the repairs of m failed nodes. Moreover, the random time $T_{\theta,\varrho}$ is exponentially independent and identically distributed with parameter $\theta\lambda$, whereas $T_{n-m,m}$ is exponentially distributed with parameter μ. More precisely, $E\left[T_{\theta,\varrho}\right] = \frac{1}{\theta\lambda}$ and $E\left[T_{n-m,m}\right] = \frac{1}{\mu}$. Thus, $E[T]$ equals the sum expectation of independent exponential random variables.

$$E[T] = \sum\nolimits_{\theta=n-m+1}^{n} \frac{1}{\theta\lambda} + \frac{1}{\mu} = \frac{H_{n,n-m}}{\lambda} + \frac{1}{\mu} \quad (8)$$

where $H_{n,n-m} = \sum\nolimits_{\theta=n-m+1}^{n} \frac{1}{\theta}$. Combining (6) and (8), we can get the average data recovery cost $r_c(m)$ as follows (9), shown at the bottom of the page. It must be noted that the (9) is obtained by constructing the general repair trees in centralized model. In next section, we will investigate the corresponding data recovery cost of another special model, i.e., optimal repair tree model.

B. THE OPTIMAL REPAIR THRESHOLD m^* OF GENERAL REPAIR TREE STRUCTURE

In this subsection, we derive the optimal repair threshold m^* through minimizing the $r_c(m)$. The threshold is obtained by Proposition 1.

Proposition 1. For the general repair tree in centralized repair model, the optimal repair threshold m^* is obtained as follows (10), as shown at the bottom of the page, where $\rho = \frac{\lambda}{\mu}$.

Proof. The proof process is provided in Appendix A.

From the Proposition 1, the upper bound of ρ is derived for optimal repair threshold, in which the delay repair strategy at $m = n - k$ is more efficient than the instantaneous repair strategy at $m = 1$. The range of ρ has several implications. On the one hand, if ρ is low, it indicates that the DSSs have higher fault tolerance performance and allow to wait for multiple nodes to fail before triggering the repair process. On the other hand, if ρ is high, it denotes that the DSSs cannot tolerate multiple nodes failure at a given time. Therefore, the instantaneous repair strategy is more suitable in such cases. In addition, we also derive a lemma to prove that there is always a positive value of ρ, which makes the delay repair strategy better than the eager repair strategy.

Lemma 1. For arbitrary coding parameters used for regeneration, there is always some $\rho > 0$ that the delay repair strategy ($m^* = n - k$) is efficient than instantaneous repair strategy ($m = 1$).

Proof. The proof process is provided in Appendix B.

C. THE MTTDL OF PROPOSED REPAIR MODEL

In this subsection, we use the MTTDL of the repair rounds in TDMC model to analyze the data reliability of DSS. The MTTDL is defined as the time sum of two state transitions, i.e., the time from state $(n, 0)$ to state $(n-m, m)$, and the time from state $(n-m, m)$ to state $(n-m-1, m+1)$. In other words, we consider that the data are lost when the DSS transitions from state $(n-m, m)$ to state $(n-m-1, m+1)$ instead of state $(n, 0)$. In the Fig.4, the data are lost if the state transitions from $(k, n-k)$ to $(k-1, n-k+1)$ when $m \in [1, n-k]$. In such case, there are not sufficient available data to recover the source file. The probability from state $(k, n-k)$ to $(k-1, n-k+1)$ is denoted by p and $p = \frac{k\lambda}{k\lambda+\mu}$. Therefore, the time of transitioning to the state $(k-1, n-k+1)$ is given as follows.

Proposition 2. For a delay repair strategy triggering repair process at threshold m ($m \in [1, n-k]$), the MTTDL is given as follows.

$$MTTDL = \sum\nolimits_{x=0}^{\infty} \left(\frac{(x+1) H_{n,n-m}}{\lambda} + \frac{x}{\mu} + \frac{H_{n-m,k-1}}{\lambda} \right) (1-p)^x p \quad (11)$$

Proof. The proof process is provided in Appendix C.

D. THE SIMULATION RESULTS OF THE GENERAL REPAIR TREE STRUCTURE

We evaluate the threshold-based data recovery cost for the general repair tree structure in various system parameters and code parameters. In order to trade-off the relationship between successful decoding rate and decoding overhead, it is critical to select an appropriate value of ε [32]. For the LT code with ($n = 3000, k = 1500, r = 1500$), we set $\varepsilon = 0.2$, which can guarantee high successful decoding rate. Fig.5 shows how average data recovery cost $r_c(m)$ is affected by system parameter ρ. It can be seen that for all repair threshold m, with the decreasing of system parameter ρ, $r_c(m)$ decreases. When the repair threshold m exceeds a certain value (m^*), the $r_c(m)$ decreases to a minimum value. We can see that when the ρ is high (e.g., $\rho = 0.1$), the $r_c(m)$ cannot decrease to the minimum value at optimal threshold

$$r_c(m) = \frac{c(m)}{E[T]} = \frac{\lambda\mu\alpha \left[\sum_{(V_{ai},V_{aj})' \in E'} W(V_{ai}, V_{aj})' + \sum_{\xi=1}^{m} W(V_{a0}, V_{n\xi}) \right]}{\mu H_{n,n-m} + \lambda} \quad (9)$$

$$m^* = \begin{cases} r = n-k, & \rho \le \dfrac{\left[\sum_{(V_{ai},V_{aj})' \in E'} W(V_{ai}, V_{aj})' + W(V_{a0}, V_{n1}) \right] \cdot H_{n-1,k}}{\sum_{\xi=2}^{r} W(V_{a0}, V_{n\xi})} - \dfrac{1}{n} \\ 1, & \textit{others} \end{cases} \quad (10)$$

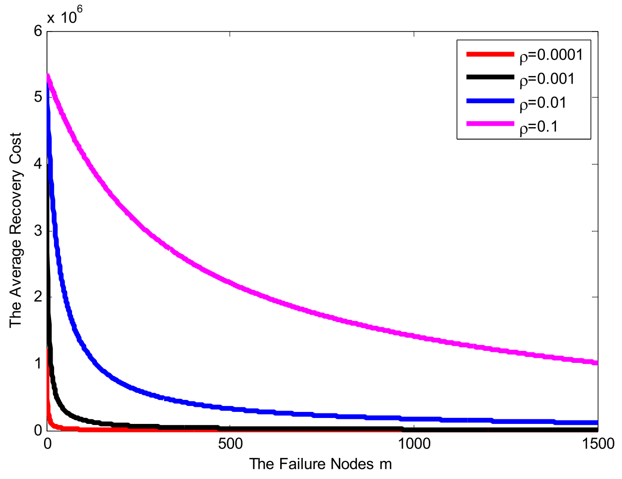

FIGURE 5. Impact of various ρ on $r_c(m)$ for general repair tree structure.

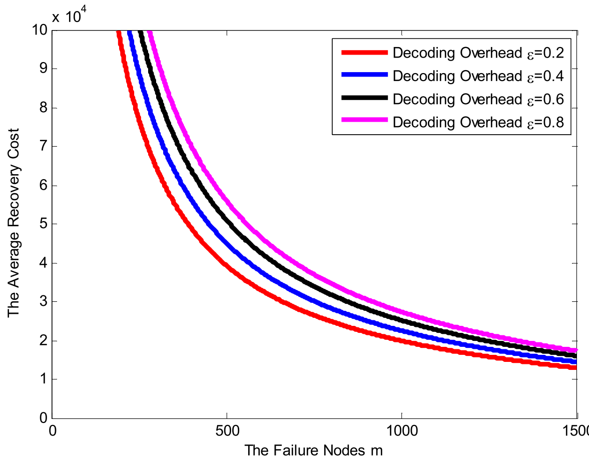

FIGURE 7. Effect of ε on $r_c(m)$ for $(n = 3000, k = 1500)$ LT codes.

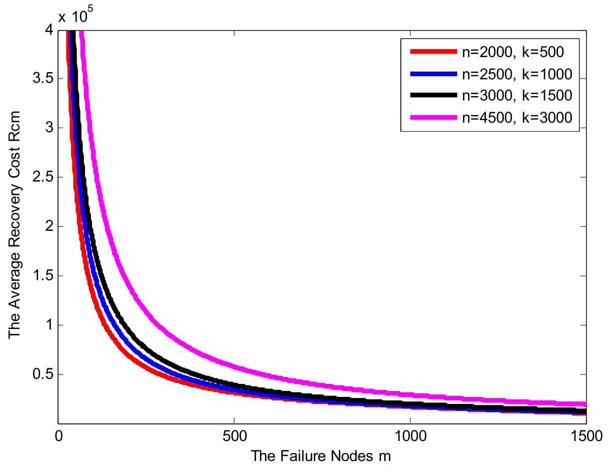

FIGURE 6. Impact of various (n, k) on $r_c(m)$ for general repair tree model.

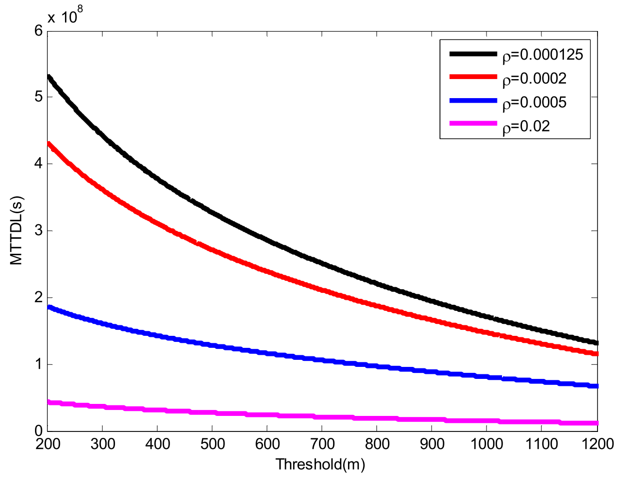

FIGURE 8. Effect of ρ on MTTDL.

$m^* = n - k = 1500$. It means that the delay repair strategy is inefficient in such cases. In addition, when the ρ is low (e.g., $\rho = 0.001$), the $r_c(m)$ can decrease to the minimum value at optimal threshold $m^* = n - k = 1500$. Thus, there is always an appropriate regime of ρ, which make the delay repair scheme more efficient.

Fig.6 shows the effect of various values of (n, k) on average data recovery cost $r_c(m)$, and we set the system parameters $\rho = 0.001$ and $\varepsilon = 0.2$. From Fig.6, we can see that with the increasing of m, the drop trend of $r_c(m)$ with different values of (n, k) tends to be consistent, which means that they converge to a same optimal repair threshold $m^* = n - k = 1500$.

Fig.7 shows the relationship between the average data recovery cost $r_c(m)$ and the repair threshold m at different decoding overhead ε. We set the system parameters $\rho = 0.001$ and $(n = 3000, k = 1500)$. From the Fig.7, it can be seen that the $r_c(m)$ of $\varepsilon = 0.8$ is highest while it is lowest when $\varepsilon = 0.2$.

Fig.8 shows the relationship between MTTDL and repair threshold at different system parameter ρ, and we set $(n = 3000, k = 1500)$. For all repair threshold m, with the increasing of ρ, the MTTDL decreases. As expected, the MTTDL is a decreasing function of repair threshold m due to the corresponding increase in departure rate from state $(n - m, m)$ with the value of m. In the high ρ regime, the MTTDL becomes impractical due to the node fails frequently before repair process can be finished.

IV. THE OPTIMAL REPAIR TREE IN CENTRALIZED REPAIR MODEL

We have discussed the delay repair strategy of centralized repair model in Section III and derived the optimal repair threshold of general repair tree structure to minimize the average data recovery cost. However, the optimal utilization of the network resource in large-scale DSS is expected while the general repair tree model cannot meet the requirement. Therefore, it is necessary to investigate the optimal repair tree-structured model for fully utilizing the node resources in large-scale DSS. As a combinatorial optimization problem, the optimal tree problem has been verified to be a NP problem [37]. To solve discrete combinatorial optimization problems, many algorithms have been proposed, such as exact algorithms, approximate algorithms, heuristic algorithms, metaheuristic algorithms and so on [38]–[40]. In this section, we use an improved tabu search algorithm to construct the optimal repair tree in the centralized model. Next, we first introduce the tabu search algorithm, and then describe our optimal repair tree construction scheme in detail.

A. TABU SEARCH ALGORITHM

Tabu search algorithm (TSA) is a very efficient method in the field of combinatorial optimization [41]. As a local-search metaheuristic algorithm, tabu search drives solution space to avoid local optima and cycling which as the weakness of heuristic approaches within a set of neighborhoods of solutions. Briefly, the TSA approach explores the search space by moving from a feasible solution to the best solution in a set of neighborhoods with each iteration. Therefore, the primary part of TSA is to generate an initial solution as the current solution by a random approach, then searches several solutions in the neighboring set of the current solution and finds the best solution. To escape from the local optima, the tabu list is introduced to record the historical information of the local optima. Specially, the length of tabu list plays an important role, which will affect the performance of TSA. Thus, the initial solution and the length of tabu list greatly affect the efficiency of the TSA. Due to the excellent performance, TSA has been widely used to solve combinatorial optimization problems [42]–[45].

The simple implementation steps of TSA to solve the combinatorial optimization problem are as follows.

Step 1: Generate an initial solution s by a random method and initialize the tabu list.

Step 2: The initial solution is viewed as the current solution s^{cur} and its neighboring set $\mathcal{N}(s^{cur})$ is constructed by a special principle. Then, the neighborhood of current solution is searched and choose the candidate solutions $\{s^{can}\}$ which meets the tabu requirements. If the multiple criteria are met, then turn to the step 4.

Step 3: Select the best solution s^{can_best} among the candidate solutions $\{s^{can}\}$, and let $s^{cur} = s^{can\_best}$, update the tabu list and turn to the step 2.

Step 4: Output the global optimal solution s^{opt} and stop the operation.

B. MODELING OF OPTIMAL REPAIR TREE PROBLEM

Before introducing our scheme, it is necessary to model the optimal repair tree problem in large-scale DSS. Without loss of generality, an edge-weighted node graph $G(V, E, W)$ is used to denote the storage node network scenario, where V indicates the set of storage nodes, E denotes the edge set of nodes, and W is the set of edge weights. For simplicity, we assume that the amount of data transmitted between nodes is equal. According to previous description, the Lemma 2 is obtained as follows.

Lemma 2. In the set of general repair tree structure T_{tree}, the optimal repair tree structure T_{tree}^{opt} is defined as the tree which minimizes the sum of edge weights.

Proof: From the formula (5), we can see that the data recovery cost $c\,(m)$ achieves the minimum when the $\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai}, V_{aj}\right)' + \sum_{\xi=1}^{m} \mathrm{W}\left(V_{a0}, V_{n\xi}\right)'\right]$ equals the minimum value, while the amount of data transmitted α is a constant. Since the replacement nodes have been given in advance, the $c\,(m)$ can achieve the minimum when the $\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai}, V_{aj}\right)'$ is minimum. Obviously, it can be proved that the repair tree with the minimum sum of edge weights is the optimal repair tree.

According to Lemma 2, the optimal repair tree problem can be defined as follows.

$$\text{Minimize } W_{sum} = \sum\nolimits_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai}, V_{aj}\right)' \tag{12}$$

$$\text{Subject to} \sum\nolimits_{(V_{ai},V_{aj})'\in E'} \left(V_{ai}, V_{aj}\right)' = (1+\varepsilon)\,k, \tag{13}$$

where $\varepsilon > 0, i, j = 0, 1, 2, \ldots, n-m$.

Under such scenario, there are $C_{n-m}^{(1+\varepsilon)k}$ selections of provider nodes with different data transmission costs for constructing a repair tree. Therefore, it is very meaningful to find an appropriate method to construct optimal repair tree.

C. THE IMPROVED TSA-BASED OPTIMAL REPAIR TREE (ITSA-ORT)

As two classical algorithms, Prim algorithm and Kruskal algorithm are always used to tackle the traditional minimum spanning tree problems [46]. However, they cannot escape from the local optimal trap. In most practical applications, it is usually necessary to find a group of optimal or suboptimal solutions as the candidate solutions for evaluation and selection. Therefore, for our optimal repair tree problem in large-scale DSSs, we adopt an improved tabu search algorithm (ITSA) to deal with this problem.

From the previous subsection a, we can know that the tsa would have better performance and faster convergence speed when there is a good initial solution and neighborhood structure. Inspired by tsa and prim, we propose an improved tsa by combining the prim algorithm in this paper. more precisely, the initial solution of tsa is generated by prim algorithm. Then the corresponding neighborhood structure is obtained by special method. The detailed description is provided as follows.

1) THE DESIGN OF NEIGHBORHOOD STRUCTURE OF ITSA-ORT SCHEME

Let $\boldsymbol{\mathcal{A}}$ denotes the search space. The neighborhood structure $\boldsymbol{\mathcal{N}} : \boldsymbol{\mathcal{A}} \longmapsto \mathbf{2}^{\boldsymbol{\mathcal{A}}}$ is a function that provides a neighboring set for each solution $\mathbf{s} \in \boldsymbol{\mathcal{A}}$. The neighborhood structure we design for optimal repair tree problem is very simple and intuitive. The repair tree can be denoted by $\boldsymbol{T_{tree}}$, and the corresponding neighborhood structure is indicated by $\boldsymbol{\mathcal{N}}\,(\boldsymbol{T_{tree}})$. The generation process of $\boldsymbol{\mathcal{N}}\,(\boldsymbol{T_{tree}})$ can be described as follows: Firstly, a new repair tree $\boldsymbol{T_{tree\_1}}$ can be generated by removing a tree edge $\boldsymbol{e}$ and adding another edge from the edge set $\boldsymbol{E}\left(\boldsymbol{T_{tree\_1}}\right) \setminus \{\boldsymbol{e}\}$, where the $\boldsymbol{E}\left(\boldsymbol{T_{tree\_1}}\right)$ is defined as follows:

$$E\left(T_{tree_1}\right) = e = \left(V_{ai}, V_{aj}\right)' \in E | V_{ai} \in V\left(T_{tree_1}\right) \mathbf{XOR} V_{aj} \in V\left(T_{tree_1}\right) \qquad (14)$$

where $\boldsymbol{E}$ indicates the edge set of a graph $\boldsymbol{G}$. It can be seen intuitively from (14) that the edge $\boldsymbol{E}\left(\boldsymbol{T}_{\boldsymbol{tree\_1}}\right)$ consists of all edges which do not belong to the repair tree $\boldsymbol{T}_{\boldsymbol{tree\_1}}$ and have exactly one end-point in $\boldsymbol{T}_{\boldsymbol{tree}\ \mathbf{1}}$. We are going to use the proposed neighborhood structure in the improved tsa for performing moves.

2) THE IMPLEMENTATION OF ITSA-ORT SCHEME

To obtain the best solution by implementing the ITSA-ORT scheme, we need two tabu lists, denoted by RemList and AddList, respectively. They store the edges which were recently removed and added, respectively. Each deletion and addition of an edge in the current repair tree indicate a round, i.e., a move operation. The RemList denotes the list that keep storage of the removed edges, and the AddList indicates the list that store the added edges. Moreover, as a critical parameter, the tenure of tabu list is a period for which it forbids edges in the list from removing or adding. In our scheme, an adaptive method is adopted to adjust the tabu tenure. We denote the minimum tabu list tenure by tl_{min}, indicate the maximum tabu list tenure by tl_{max} and denote an increment value by Δ_{tlen}. In detail, we set an initial value of tabu tenure tl_{ten} to tl_{min} at the beginning of each reboot phase (i.e., a new initial solution is generated while the current solution is removed). If the reboot-optimal solution T_{tree}^{ropt} was not modified for a maximum number of $iter_{max}$, the tabu list tenure tl_{ten} is increased by Δ_{tlen} for diversifying the search process. Whenever the T_{tree}^{ropt} is modified, the tabu list tenure tl_{ten} is set back to tl_{min} for strengthening the search process around T_{tree}^{ropt}. In addition, if $tl_{ten} + \Delta_{tlen} > tl_{max}$, the reboot is performed. This situation can be regarded as an escaping mechanism. The implementation of our ITSA-ORT scheme is shown in Algorithm 1.

From the Algorithm 1, we can get the optimal repair tree T_{tree}^{opt} and its sum of edge weights. To analyze the data transmission cost of T_{tree}^{opt}, we substitute the formula (9) from the result of Algorithm 1. Thus, the average data recovery cost of T_{tree}^{opt} can be calculated as follows.

$$r_{c\_opt}(m) = \frac{\lambda\mu\alpha\left[Min(W_{sum}) + \sum_{\xi=1}^{m} \mathrm{W}\left(V_{a0}, V_{n\xi}\right)\right]}{\mu H_{n,n-m} + \lambda} \qquad (15)$$

$$Min\left(W_{sum}\right) = \sum\nolimits_{\left(V_{ai}, V_{aj}\right)' \in E'\left(T_{tree}^{opt}\right)} \mathrm{W}\left(V_{ai}, V_{aj}\right)' \qquad (16)$$

where $E'(T_{tree}^{opt})$ denotes the edge set of T_{tree}^{opt}.

Therefore, the optimal repair threshold of T_{tree}^{opt} can be also obtained as follows.

$$m_{opt}^{*} = \begin{cases} r = n-k, & \rho \leq \frac{[Min(W_{sum}) + \mathrm{W}(V_{a0}, V_{n1})] \cdot H_{n-1,k}}{\sum_{\xi=2}^{r} \mathrm{W}(V_{a0}, V_{n\xi})} - \frac{1}{n} \\ 1, & others \end{cases} \qquad (17)$$

Algorithm 1: The Optimal Repair Tree Based on Improved Tabu Search Algorithm (ITSA-ORT)

```
Input  : a weighted complete graph $G(V, E, W)$
Output: $T_{tree}^{opt}$ &
        $Min(W_{sum}) \sum_{(V_{ai}, V_{aj})' \in E'(T_{tree}^{opt})} \mathrm{W}(V_{ai}, V_{aj})'$
1:  Initialize Parameters
    ($tl_{min}, tl_{max}, \Delta_{tlen}, tl_{ten}, iter = 0, iter_{max}$)
2:  Initialize Tabulists ($RemList, AddList, tl_{ten}$)
3:  $T_{tree}^{cur} \leftarrow$ Generate initial solution by **Prim algorithm**
4:  $T_{tree}^{opt} \leftarrow T_{tree}^{cur}, T_{tree}^{ropt} \leftarrow T_{tree}^{cur}$
5:  **While** *termination conditions not met* **do**
6:      $T_{tree}^{new} \leftarrow$ Neighborhood
          ($\mathcal{N}(T_{tree}^{cur}), Remlist, Addlist$)
7:      **If** $T_{tree}^{new} \neq NULL$ **then**
8:          Update Tabulists ($T_{tree}^{cur}, T_{tree}^{new}, Remlist, Addlist$)
9:          $T_{tree}^{cur} \leftarrow T_{tree}^{new}$
10:         Update ($T_{tree}^{cur}, T_{tree}^{ropt}, T_{tree}^{opt}, iter$)
11:         **If** $iter > iter_{max}$ **then**
12:             **If** $tl_{ten} + \Delta_{tlen} > tl_{max}$ **then**
13:                 Perform Restart ()
14:                 else
15:                 $tl_{ten} \leftarrow tl_{ten} + \Delta_{tlen}$
16:             end if
17:         end if
18:         else
19:         Perform Restart ()
20:     end if
21: end while
22: The final solution is the best solution $T_{tree}^{opt}$ found so far
```

Compared with the formula (9), it can be seen intuitively from (15) that the average data recovery cost of optimal repair tree T_{tree}^{opt} is lower than that of the general tree T_{tree}. The reason is that the T_{tree}^{opt} provides the lowest value of links' weight sum.

In addition, comparing formula (10) and (17), we can find that although the T_{tree}^{opt} and the T_{tree} have the same optimal repair threshold, the corresponding upper bounds of ρ are different. It means that the T_{tree}^{opt} has a lower upper bound of ρ.

D. THE COMPUTATIONAL COMPLEXITY ANALYSIS OF ITSA-ORT

In order to fully understand the performance of our scheme, we now analyze its time complexity and space complexity for ITSA-ORT algorithm. For the time complexity, since the initial solution is generated by Prim algorithm, the generation of initial solution can be done in $O\left(|V|^2\right)$, where $|V|$ denotes the number of nodes in graph G. Then, constructing neighborhood for the current solution runs in $O(|V|)$, and checking whether a solution of the neighborhood is feasible or not can be done in $O\left(|V|^2\right)$. Sorting all solutions of the neighborhood in non-descending order of the average data recovery cost runs in $O(|V| \, log\, |V|)$. With the tabu list tenure

TABLE 2. Comparison of different optimal repair tree generation schemes.

Graph	k		The weights sum of optimal repair tree				
			Kruskal	Prim	PSO	FA	ITSA-ORT
$\lvert V\rvert = 3000$ $\varepsilon = 0.2$ $\rho = 0.001$ 5000* 5000	200	Best	129822.2	130600.4	128621.3	125900.1	**122450.8**
	500	Best	183002.3	181250.1	177434.1	171338.4	**168789.2**
	800	Best	211434.8	210560.8	208577.4	201398.2	**199845.6**
	1000	Best	232977.2	232500.4	230045.2	227398.6	**224850.1**
	1500	Best	293121.6	290815.5	288734.1	282924.6	**279886.4**
	1800	Best	357330.1	355380.3	354255.3	348184.2	**345523.8**
$\lvert V\rvert = 3000$ $\varepsilon = 0.4$ $\rho = 0.001$ 5000* 5000	200	Best	137936.6	137376.3	135151.4	131978.7	**128649.9**
	500	Best	199004.7	196491.1	192122.3	185011.1	**182732.9**
	800	Best	232189.1	230700.5	228470.6	220094.8	**218979.8**
	1000	Best	257336.9	256312.0	253531.3	250446.7	**248169.2**
	1500	Best	325522.1	324362.5	322018.0	315242.5	**312393.5**
	1800	Best	400461.8	399718.1	398489.8	391409.0	**389000.9**
$\lvert V\rvert = 3000$ $\varepsilon = 0.6$ $\rho = 0.001$ 5000* 5000	200	Best	145751.0	143852.2	142621.5	137557.3	**134825.0**
	500	Best	215563.9	211432.1	207750.5	198183.8	**196652.6**
	800	Best	253500.2	250540.2	249303.8	238291.4	**238090.0**
	1000	Best	282253.4	279823.6	277957.4	272994.8	**271464.3**
	1500	Best	360193.3	357609.5	356241.9	347060.4	**344876.6**
	1800	Best	445864.2	443755.9	443664.3	434133.8	**432454.0**
$\lvert V\rvert = 3000$ $\varepsilon = 0.8$ $\rho = 0.001$ 5000* 5000	200	Best	150435.1	149354.5	146934.3	142493.9	**139858.9**
	500	Best	228982.8	225399.5	220221.4	210714.5	**209431.1**
	800	Best	271671.0	269406.3	266979.7	258846.0	**256058.9**
	1000	Best	304029.6	302361.6	300216.2	297900.9	**293618.1**
	1500	Best	391724.2	389882.9	387298.5	381236.3	**376218.4**
	1800	Best	488126.3	486820.1	485671.5	479216.6	**474765.8**

tl_{ten}, the selection of a best current solution can be done in $O(tl_{ten})$. Therefore, each iteration of ITSA-ORT algorithm can be done in $O\left(|V|^2\right) + O(|V|) + O(|V|\log|V|) + O\left(|V|^2\right) + O(tl_{ten})$, which can be simplified as $O\left(|V|^2\right)$. To sum up, the time complexity of the proposed scheme is $O\left(iter_{max}\cdot|V|^2\right)$, where $iter_{max}$ denotes the maximum size of iterations.

The space complexity of initial solution's generation and parameters' initialization is $O(|V|)$. Then, the space complexity of constructing the neighborhood set for current solution is $O(|V|)$. After that, the space complexity of updating the neighborhood set and current solution is $O(2|V|)$. Finally, the space complexity for the storage of best solution (i.e., the optimal average data recovery cost) is $O(|V|)$. Therefore, the final space complexity of ITSA-ORT scheme is $O(2|V|)$.

E. THE SIMULATION RESULTS ANALYSIS AND COMPARISON

In this subsection, we compare the proposed scheme of optimal repair tree with several other optimization algorithms. The parameter settings are as follows, $|V| = 3000$, $\rho = 0.001$, and $\varepsilon = \{0.2, 0.4, 0.6, 0.8\}$. In addition, some initial parameters of improved TSA, such as $(tl_{min}, tl_{max}, \Delta_{tlen}, tl_{ten}, iter = 0, iter_{max})$, are set as recommended in [40]. The storage nodes are deployed in the range of 5000 * 5000.

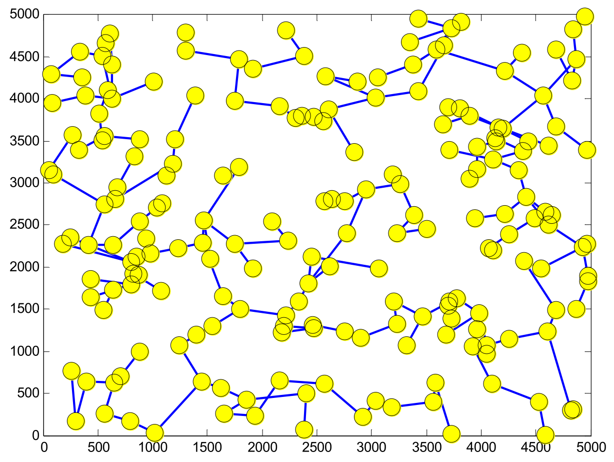

FIGURE 9. The sample of optimal repair tree generated by ITSA scheme.

Fig.9 shows the sample of optimal repair tree which is generated by improved TSA scheme, where the number of k is 200. From the Fig.9, it can be seen that the network distance

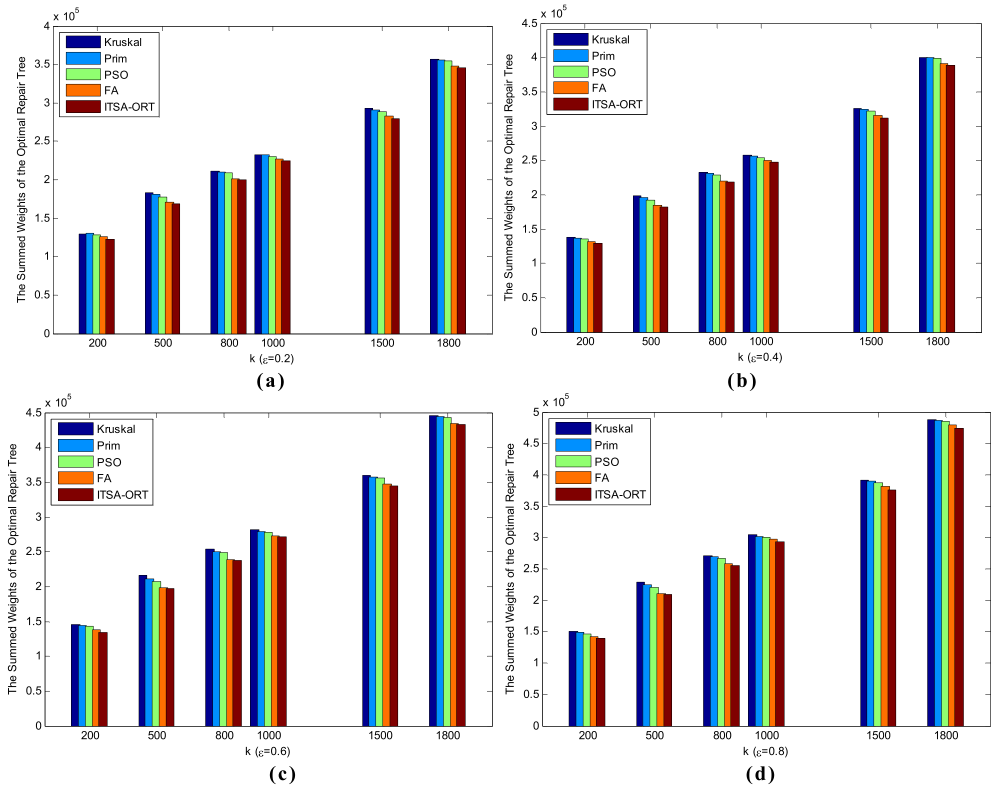

FIGURE 10. **The summed weights comparison of optimal repair tree for different algorithms. (a)$\varepsilon = 0.2$; (b)$\varepsilon = 0.4$; (c)$\varepsilon = 0.6$; (d) $\varepsilon = 0.8$.**

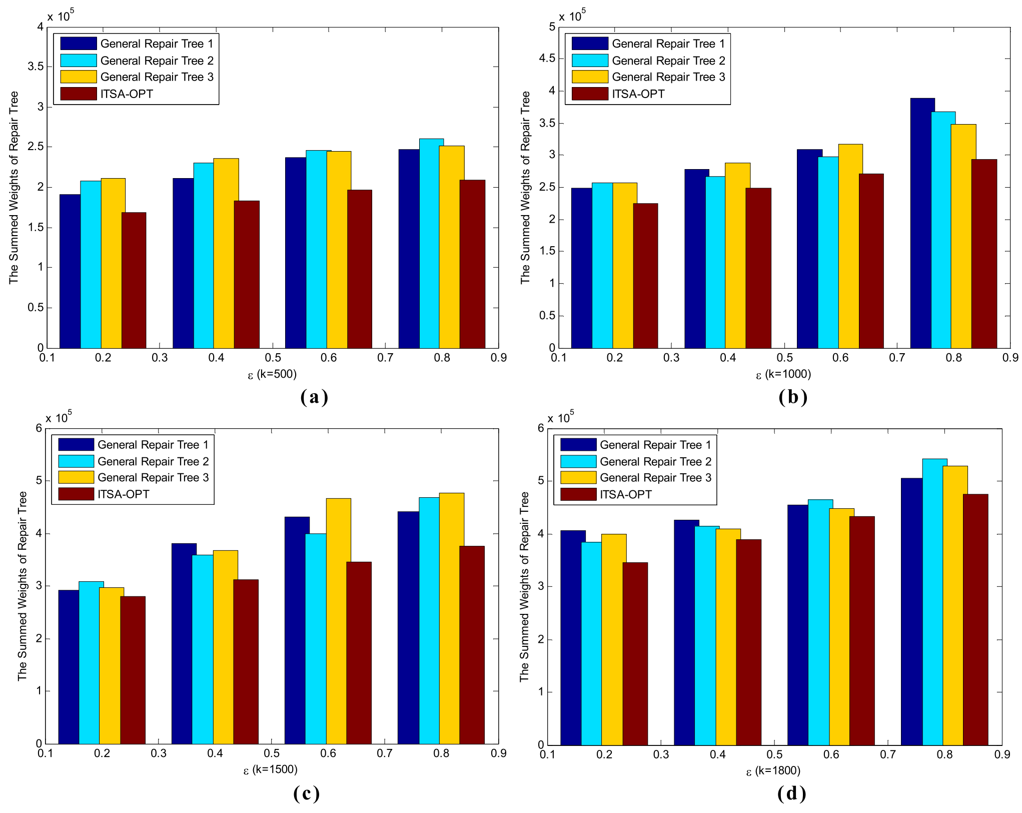

FIGURE 11. **The summed weights comparison between general repair trees and ITSA-ORT scheme with different parameters, i.e., $k = \{500, 1000, 1500, 1800\}$ and $\varepsilon = \{0.2, 0.4, 0.6, 0.8\}$.**

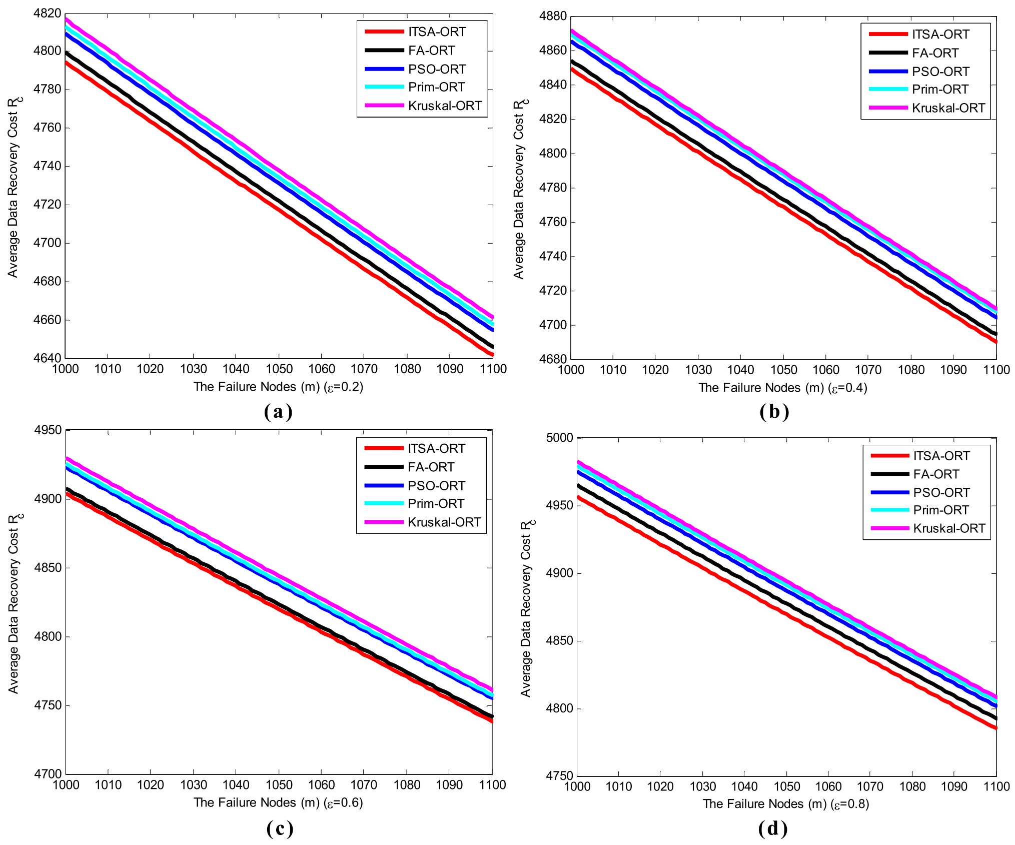

FIGURE 12. The average data recovery cost of different optimal repair tree generation schemes. (a)$\varepsilon = 0.2$; (b)$\varepsilon = 0.4$; (c)$\varepsilon = 0.6$; (d) $\varepsilon = 0.8$.

between surviving nodes is not always equal. In our scheme, the network distance between nodes is regarded as the edge weight of corresponding nodes.

Table 2 provides computational results of the weights sum of optimal repair trees for different optimization algorithms. From Table 2, the columns 4-8 give the optimal weights sum of different algorithms. It can be seen that the proposed ITSA-ORT represents the best performance in different k. In addition, the particle swarm optimization algorithm (PSO) and firefly algorithm (FA) have better values than traditional minimum spanning tree algorithms, i.e., Kruskal and Prim algorithms. In addition, it can be observed that the weights sum of optimal repair trees is proportional to the decoding overhead ε.

Fig.10 shows a more intuitive comparison of different optimal repair tree generation algorithms. Fig.10 (a), (b), (c) and (d) represent the weights sum of several optimal algorithms with different k for decoding overhead $\varepsilon = 0.2, 0.4, 0.6$ and 0.8, respectively. From the Table 2 and Fig.10, we can observe that with the increasing of ε and k, the weights sum of the optimal repair trees of different algorithms are increases.

Fig.11 shows the results of weights sum for several repair trees with different parameters. Fig.11 (a), (b), (c) and (d) indicate the weights sum of different repair trees with different values of ε for $k = 500, 1000, 1500,$ and 1800, respectively. As a matter of fact, for all repair tree models, the sum of weights is increased along with the increasing value of decoding overhead $\boldsymbol{\varepsilon}$, and the same result also can be obtained with the increasing of k. As shown in Fig.11, the ITSA-ORT scheme outperforms general repair tree schemes for the values of k and $\boldsymbol{\varepsilon}$ in terms of saving cost of data transmitted.

Fig.12 shows the efficiency of various optimization algorithms on average data recovery cost $r_c(m)$, and we set the system parameters $\rho = 0.001, \varepsilon = \{0.2, 0.4, 0.6, 0.8\}$, and $(n = 3000, k = 1500)$. It can be seen from Fig.12, taking m from 1000 to 1100 as an example, as the decoding overhead ε increases, the average data recovery cost increases at any number of failure nodes. Moreover, compared with other optimal schemes, the proposed ITSA-ORT scheme has best performance in terms of $r_c(m)$.

The comparison of computational complexity for different algorithms is shown in Table 3, where $|E|$ denotes the edge number in graph G and M is the population size. It can be seen that the ITSA-ORT scheme is worse than the Prim-ORT scheme and Kruskal-ORT scheme while better than the PSO-ORT scheme and FA-ORT scheme in terms of computational complexity. Although the Prim-ORT scheme and

TABLE 3. **The comparison of computational complexity for different optimal algorithms.**

	Time complexity	Space complexity
Kruskal-ORT	$O(\|E\|log\|E\|)$	$O(\|E\|)$
Prim-ORT	$O(\|V\|^2)$	$O(\|V\|)$
PSO-ORT	$O(iter_{max}\cdot M\|V\|^2)$	$O(\|V\|^2)$ $+O(M\|V\|)$
FA-ORT	$O(iter_{max}\cdot M\|V\|^2)$	$O(\|V\|^2)$ $+O(M\|V\|)$
ITSA-ORT	$O(iter_{max}\cdot \|V\|^2)$	$O(2\|V\|)$

Kruskal-ORT scheme have lower computational complexity, they generate only one optimal repair tree, which is very easy to get trapped in the local solution. Therefore, they cannot find a globally optimal solution for the large scale DSSs. On the contrary, the PSO-ORT scheme, FA-ORT scheme and the ITSA-ORT scheme can all find a globally optimal solution by comparing multiple optimal repair trees. In addition, compared with the PSO-ORT scheme and FA-ORT scheme, the ITSA-ORT scheme has lower computational complexity and better performance in minimizing the cost of data recovery.

V. CONCLUSION

In this work, we considered a problem of utilizing nodes' topology to analyze the delay repair strategies in centralized model, where the data content is stored by LT codes with a large value of (n, k, r). In particular, under the consideration of storage nodes' topology, repair tree models are established to calculate the data recovery cost of repairing failed nodes. Then, we constructed a two-dimensional Markov chain model to investigate threshold-based repair strategies and derived the optimal repair thresholds for minimizing the average data recovery cost of general repair trees. The numerical results show that optimal thresholds are dependent on edge weights of nodes, code parameters and node failure-to-repair rate ratio (i.e., ρ). Only under low ρ value, the delay repair is optimal in terms of average data recovery cost. Inspired by Prim algorithm and TSA algorithm, we proposed an improved TSA to tackle the optimal repair tree problem (i.e., ITSA-ORT scheme), and analyzed the average data recovery cost and optimal repair thresholds for the proposed optimal repair tree scheme. Simulation results proved that the proposed ITSA-ORT scheme was promising to lower the data transmission cost than common repair tree and other optimal tree algorithms. In the meantime, the ITSA-ORT scheme had lower computational complexity than the PSO-ORT scheme and the FA-ORT scheme.

APPENDIX A

Proof: In order to determine m^*, we compare $r_c(r)$ with average data recovery cost at all other possible storage nodes' repair states $r-\delta$, where $1 \le \delta \le r-1$, and $r = n-k$. Firstly, we check if

$$r_c(r) \le r_c(r-\delta), \forall\delta \in [1, r-1] \tag{18}$$

Next, substituting $r_c(m)$ from (9) to (18) yields,

$$\frac{\lambda\mu\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} W\left(V_{ai},V_{aj}\right)' + \sum_{\xi=1}^{r} W\left(V_{a0},V_{n\xi}\right)\right]}{\mu H_{n,n-r}+\lambda} \le \frac{\lambda\mu\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} W\left(V_{ai},V_{aj}\right)' + \sum_{\xi=1}^{r-\delta} W\left(V_{a0},V_{n\xi}\right)\right]}{\mu H_{n,n-r+\delta}+\lambda} \tag{19}$$

Let $\rho = \frac{\lambda}{\mu}$, and then we can obtain formula (20) (21), as shown at the bottom of the next page.

The equality of formula (21) is achieved when $m = r = n-k$. Now we examine the monotonicity of the right side in (21) as a function of δ. Similar to the Lemma 3 of literature [20], we inspect the monotonicity of function $f(n-r,\delta) = \frac{H_{n-r+\delta,n-r}}{\sum_{\xi=r-\delta+1}^{r} W(V_{a0},V_{n\xi})}$, which has the same monotonicity as the right side in (21). The result shows that $f(n-r,\delta)$ is monotonically decreasing with δ. Thus, the ρ is also monotonically decreasing with δ. Finally, we substitute the maximum δ (i.e., $\delta = r-1$) to formula (21) yields a bound as follows.

$$\rho \le \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} W\left(V_{ai},V_{aj}\right)' + W\left(V_{a0},V_{n1}\right)\right]\cdot H_{n-1,k}}{\sum_{\xi=2}^{r} W\left(V_{a0},V_{n\xi}\right)} - \frac{1}{n} \tag{22}$$

Next, we need to calculate the range of ρ which make the average data recovery cost $r_c(m)$ minimize at $m = 1$. We consider $r_c(1) \le r_c(1+\delta)$, where $1 \le \delta \le r-1$. By substituting from (9), it follows that

$$\frac{\lambda\mu\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} W\left(V_{ai},V_{aj}\right)' + W\left(V_{a0},V_{n1}\right)\right]}{\mu H_{n,n-1}+\lambda} \le \frac{\lambda\mu\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} W\left(V_{ai},V_{aj}\right)' + \sum_{\xi=1}^{1+\delta} W\left(V_{a0},V_{n\xi}\right)\right]}{\mu H_{n,n-\delta-1}+\lambda} \tag{23}$$

Let $\rho = \frac{\lambda}{\mu}$, and then we can obtain formula (24) (25), as shown at the bottom of the next page. Similar to Lemma 4 of literature [20], we also can obtain a result that the function $g(n-1,\delta) = \frac{H_{n-1,k}}{\sum_{\xi=2}^{r} W(V_{a0},V_{n\xi})}$ is monotonically increasing with δ. It means that the right side of (25) has the same monotonicity as the function g. Thus, the lower bound of ρ for inequality $r_c(1) \le r_c(1+\delta)$ can be obtained. Finally, the proof is complete.

APPENDIX B

Proof: In order to prove Lemma 1, it needs to show that the upper bound of ρ for which $m^* = n-k$ is strictly positive for any coding parameters. We examine this fact by analyzing the boundary of $\frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} W(V_{ai},V_{aj})' + W(V_{a0},V_{n1})\right]\cdot H_{n-1,k}}{\sum_{\xi=2}^{r} W(V_{a0},V_{n\xi})}$.

For our purposes, we have assumed that the replacement nodes are given, which means that the link weights $\mathrm{W}\left(V_{a0}, V_{n\xi}\right)$ from leader node to replacements have been known for us. Therefore, without loss of generality, we suppose that the link weights $\mathrm{W}\left(V_{a0}, V_{n\xi}\right)$ are same and denote by W_l. Then we can obtain the expression of $\frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}(V_{ai},V_{aj})'+\mathrm{W}_l\right]\cdot H_{n-1,k}}{(n-k-1)\cdot \mathrm{W}_l}$.

Let $f(x,\delta) = \frac{H_{x+\delta,x}}{\delta}$ for $x > 0$, $\delta > 0$, then we can get that

$$\frac{H_{x+\delta,x}}{\delta} = \sum_{i=1}^{x+\delta}\frac{1}{i} - \sum_{i=1}^{x} i = \frac{\frac{1}{x+1}+\frac{1}{x+2}+\ldots+\frac{1}{x+\delta}}{\delta} \tag{26}$$

There would be a lower bound and an upper bound for (26) as follows.

The lower bound:

$$> \frac{\frac{1}{x+\delta}+\frac{1}{x+\delta}+\ldots+\frac{1}{x+\delta}}{\delta} = \frac{\frac{1}{x+\delta}\cdot\delta}{\delta} = \frac{1}{x+\delta} \tag{27}$$

The upper bound:

$$< \frac{\frac{1}{x}+\frac{1}{x}+\ldots+\frac{1}{x}}{\delta} = \frac{\frac{1}{x}\cdot\delta}{\delta} = \frac{1}{x} \tag{28}$$

Then,

$$\frac{1}{x+\delta} < \frac{H_{x+\delta,x}}{\delta} < \frac{1}{x} \tag{29}$$

Therefore, we can obtain formula (30) by substituting (29) from $\frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}(V_{ai},V_{aj})'+\mathrm{W}_l\right]\cdot H_{n-1,k}}{(n-k-1)\cdot \mathrm{W}_l}$ as follows.

$$\begin{aligned}&\frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\mathrm{W}_l\right]}{(n-1)\cdot \mathrm{W}_l}\\ &\le \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\mathrm{W}_l\right]\cdot H_{n-1,k}}{(n-k-1)\cdot \mathrm{W}_l}\\ &< \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\mathrm{W}_l\right]}{k\cdot \mathrm{W}_l}\end{aligned} \tag{30}$$

Thus, the result of equation $\rho = \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}(V_{ai},V_{aj})'+\mathrm{W}(V_{a0},V_{n1})\right]\cdot H_{n-1,k}}{\sum_{\xi=2}^{r}\mathrm{W}(V_{a0},V_{n\xi})} - \frac{1}{n}$ is strictly larger than zero.

APPENDIX C

Proof: To obtain MTTDL, it needs to consider two phases' time. The first stage's time is the cumulative sum of transition time from state $(n,0)$ to state $(n-m,m)$, and equals $\sum_{i=n-m+1}^{n}\frac{1}{i\lambda} = \frac{H_{n,n-m}}{\lambda}$, where $H_{n,n-m} = \sum_{i=n-m+1}^{n}\frac{1}{i}$. The time of the second phase equals the cumulative sum of transition time from state $(n-m,m)$ to $(n-m-1,m+1)$, i.e., it equals $\sum_{i=k}^{n-m}\frac{1}{i\lambda} = \frac{H_{n-m,k-1}}{\lambda}$, where $H_{n-m,k-1} = \sum_{i=k}^{n-m}\frac{1}{i}$. We define a state variable, namely repair round, which denotes a cycle process of $\{(n,0) \longrightarrow \text{repair process} \longrightarrow (n,0)\}$. Therefore, the MTTDL can be calculated by following steps:

When undergo 0 repair round:

$$\left(\frac{H_{n,n-m}}{\lambda}+\frac{H_{n-m,k-1}}{\lambda}\right)p;$$

When undergo 1 repair round:

$$\left(\frac{2H_{n,n-m}}{\lambda}+\frac{1}{\mu}+\frac{H_{n-m,k-1}}{\lambda}\right)(1-p)\,p;$$

When undergo 2 repair rounds:

$$\left(\frac{3H_{n,n-m}}{\lambda}+\frac{2}{\mu}+\frac{H_{n-m,k-1}}{\lambda}\right)(1-p)^2\,p$$

$\vdots$

When undergo i repair rounds:

$$\left(\frac{(i+1)\,H_{n,n-m}}{\lambda}+\frac{i}{\mu}+\frac{H_{n-m,k-1}}{\lambda}\right)(1-p)^i\,p;$$

Then,

$$\mathrm{MTTDL} = \sum_{i=0}^{\infty}\left(\frac{(i+1)\,H_{n,n-m}}{\lambda}+\frac{i}{\mu} + \frac{H_{n-m,k-1}}{\lambda}\right)(1-p)^i\,p \tag{31}$$

$$\frac{\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\sum_{\xi=1}^{r}\mathrm{W}\left(V_{a0},V_{n\xi}\right)\right]}{H_{n,n-r}+\rho} \le \frac{\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\sum_{\xi=1}^{r-\delta}\mathrm{W}\left(V_{a0},V_{n\xi}\right)\right]}{H_{n,n-r+\delta}+\rho} \tag{20}$$

$$\rho \le \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\sum_{\xi=1}^{r}\mathrm{W}\left(V_{a0},V_{n\xi}\right)\right]\cdot H_{n-r+\delta,n-r}}{\sum_{\xi=r-\delta+1}^{r}\mathrm{W}\left(V_{a0},V_{n\xi}\right)} - H_{n,n-r} \tag{21}$$

$$\frac{\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\mathrm{W}\left(V_{a0},V_{n1}\right)\right]}{H_{n,n-1}+\rho} \le \frac{\alpha\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\sum_{\xi=1}^{1+\delta}\mathrm{W}\left(V_{a0},V_{n\xi}\right)\right]}{H_{n,n-\delta-1}+\rho} \tag{24}$$

$$\rho \ge \frac{\left[\sum_{(V_{ai},V_{aj})'\in E'} \mathrm{W}\left(V_{ai},V_{aj}\right)'+\mathrm{W}\left(V_{a0},V_{n1}\right)\right]\cdot H_{n-1,k}}{\sum_{\xi=2}^{r}\mathrm{W}\left(V_{a0},V_{n\xi}\right)} - \frac{1}{n} \tag{25}$$

REFERENCES

[1] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. Int. Workshop Peer-to-Peer Syst.*, Mar. 2002, pp. 328–337.

[2] J. Li and B. Li, "Erasure coding for cloud storage systems: A survey," *Tsinghua Sci. Technol.*, vol. 18, no. 3, pp. 259–272, Jun. 2013.

[3] Amazon EC. *Amazon Web Services*. Accessed: Nov. 2012. [Online]. Available: http://aws.amazon.com/es/ec2/

[4] K. Nansai, X. Chen, S. Chen, and J. Zhang. *HDFS Erasure Coding in Production*. Accessed: Jun. 10, 2019. [Online]. Available: https://blog.cloudera.com/hdfs-erasure-coding-in-production/

[5] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[6] X. Y. Zhang and Y. C. Hu, "Efficient storage scaling for MBR and MSR codes," *IEEE Access*, vol. 8, pp. 78992–79002, Apr. 2020.

[7] W. Q. Li, Z. Y. Wang, and H. Jafarkhani, "Repairing Reed–Solomon codes over GF(2l)," *IEEE Commun. Lett.*, vol. 24, no. 1, pp. 34–37, Jan. 2020.

[8] H. X. Hou and P. P. C. Lee, "Binary MDS array codes with optimal repair," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1405–1422, Mar. 2020.

[9] E. Yavari and M. Esmaeili, "Locally repairable codes: Joint sequential-parallel repair for multiple node failures," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 222–232, Jan. 2020.

[10] B. Zhu, K. W. Shum, and H. Li, "Fractional repetition codes with optimal reconstruction degree," *IEEE Trans. Inf. Theory*, vol. 66, no. 2, pp. 983–994, Feb. 2020.

[11] D. J. C. MacKay, "Fountain codes," *IEEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[12] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[13] M. Luby, "LT codes," in *Proc. IEEE Symp. Found. Comp. Sci.*, Nov. 2002, pp. 271–280.

[14] Z. N. Kong, S. A. Aly, and E. Soljanin, "Decentralized coding algorithms for distributed storage in wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 261–267, Feb. 2010.

[15] N. Cao, S. C. Yu, Z. Y. Yang, W. J. Lou, and Y. T. Hou, "LT codes-based secure and reliable cloud storage service," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 693–701.

[16] A. H. F. Raouf, J. Abouei, M. Jaseemuddin, and M. Uysal, "Delivery phase in cache-based wireless networks with modified LT codes," *Phys. Commun.*, vol. 42, pp. 1–11, Jul. 2020.

[17] X. Ye, J. Li, W.-T. Chen, and F. Tang, "LT codes based distributed coding for efficient distributed storage in wireless sensor networks," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.

[18] B. Kong, G. X. Zhang, F. L. Dong, L. Cheng, and X. L. Zhou, "Efficient distributed storage based on LT codes in wireless sensor networks," in *Proc. Int. Conf. Wireless Commun. Signal Proc. (WCSP)*, Oct. 2015, pp. 1–5.

[19] N. N. Wang, Y. Wang, S. S. Gu, Q. Y. Zhang, and W. Xiang, "Adaptive data storage system for mobile satellite-terrestrial IoT," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2019, pp. 106–111.

[20] G. Calis, S. Shivaramaiah, O. O. Koyluoglu, and L. Lazos, "Repair strategies for mobile storage systems," *IEEE Trans. Cloud Comput.*, early access, May 2, 2019, doi: 10.1109/TCC.2019.2914436.

[21] J. Pedersen, A. G. I. Amat, I. Andriyanova, and F. Brannstrom, "Optimizing MDS coded caching in wireless networks with device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 286–295, Jan. 2019.

[22] S. S. Gu, J. Li, Y. Wang, N. N. Wang, and Q. Zhang, "DR-MDS: An energy-efficient coding scheme in D2D distributed storage network for the Internet of Things," *IEEE Access*, vol. 7, pp. 24179–24191, Feb. 2019.

[23] M. Luby, R. Padovani, T. J. Richardson, L. Minder, and P. Aggarwal, "Liquid cloud storage," *ACM Trans. Storage*, vol. 15, no. 1, pp. 1–49, Apr. 2019.

[24] M. Gerami, M. Xiao, M. Skoglund, K. W. Shum, and D. Lin, "Optimal-cost repair in multi-hop distributed storage systems with network coding," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 11, pp. 1539–1549, Nov. 2016.

[25] S. W. Qin and Z. Li, "Network topology impacts on repair cost in distributed storage system with network coding," in *Proc. IEEE ICECE*, Dec. 2018, pp. 102–109.

[26] M. Sipos, J. Gahm, N. Venkat, and D. Oran, "Network-aware feasible repairs for erasure-coded storage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1404–1417, Jun. 2018.

[27] J. Xia, D. Guo, L. Luo, and G. Cheng, "Topology-aware data placement strategy for fault-tolerant storage systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 4296–4307, Sep. 2020.

[28] J. Li, S. Yang, X. Wang, X. Y. Xue, and B. C. Li, "Tree-structured data regeneration with network coding in distributed storage systems," in *Proc. IEEE 17th Int. Workshop Quality Service*, Jul. 2009, pp. 1–9.

[29] S. Weidong, W. Yijie, and P. Xiaoqiang, "Tree-structured parallel regeneration for multiple data losses in distributed storage systems based on erasure codes," *China Commun.*, vol. 10, no. 4, pp. 113–125, Apr. 2013.

[30] P. F. You, Y. X. Peng, Z. Huang, and C. J. Wang, "Repairing multiple data losses by parallel max-min trees based on regenerating codes in distributed storage systems," in *Proc. 14th Int. Conf. Algorithms Archit. Parallel*, Aug. 2014, pp. 325–338.

[31] H. Bao, Y. J. Wang, and F. L. Xu, "Reducing network cost of data repair in erasure-coded cross-data center storage," *Future Gener. Comput. Syst.*, vol. 102, pp. 494–506, Jan. 2020.

[32] W. Yao, B. Yi, W. Li, T. Huang, and Q. Xie, "CPRSD for LT codes," *IET Commun.*, vol. 10, no. 12, pp. 1411–1415, Aug. 2016.

[33] M. Ye and A. Barg, "Cooperative repair: Constructions of optimal MDS codes for all admissible parameters," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1639–1656, Mar. 2019.

[34] M. Zorgui and Z. Wang, "Centralized multi-node repair regenerating codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4180–4206, Jul. 2019.

[35] H. Zhang, H. Li, and S.-Y.-R. Li, "Repair tree: Fast repair for single failure in erasure-coded distributed storage systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1728–1739, Jun. 2017.

[36] J. Pedersen, A. G. I. Amat, I. Andriyanova, and F. Brannstrom, "Distributed storage in mobile wireless networks with device-to-device communication," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4862–4878, Sep. 2016.

[37] R. Castro de Andrade, "New formulations for the elementary shortest-path problem visiting a given set of nodes," *Eur. J. Oper. Res.*, vol. 254, no. 3, pp. 755–768, Nov. 2016.

[38] S. Muthuraman and V. P. Venkatesan, "A comprehensive study on hybrid meta-heuristic approaches used for solving combinatorial optimization problems," in *Proc. World Congr. Comput. Commun. Technol. (WCCCT)*, Feb. 2017, pp. 185–190.

[39] L. Zhou, A. T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, no. 2, pp. 1–23, Jun. 2020.

[40] J. Puchinger and G. R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification," in *Proc. IWINAC*, in Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2005, pp. 41–53.

[41] V. K. Prajapati, M. Jain, and L. Chouhan, "Tabu search algorithm (TSA): A comprehensive survey," in *Proc. 3rd Int. Conf. Emerg. Technol. Comput. Eng. Mach. Learn. Internet Things (ICETCE)*, Feb. 2020, pp. 222–229.

[42] A. H. El-Maleh, "A probabilistic tabu search state assignment algorithm for area and power optimization of sequential circuits," *Arabian J. Sci. Eng.*, vol. 45, no. 8, pp. 6273–6285, Aug. 2020.

[43] Y. Chen, S. Li, and H. Liu, "Dynamic frequency reuse based on improved tabu search in multi-user visible light communication networks," *IEEE Access*, vol. 7, pp. 35173–35183, Feb. 2019.

[44] H. Katagiri, T. Hayashida, I. Nishizaki, and Q. Q. Guo, "A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5681–5686, Apr. 2012.

[45] E. L. da Silva, V. A. A. da Silva, L. B. Goncalves, L. L. O. Moreno, and S. S. R. F. Soares, "An efficient algorithm for the tabu clustered traveling salesman problem," in *Proc. 8th Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2019, pp. 287–292.

[46] P. Ayegba, J. Ayoola, E. Asani, and A. Okeyinka, "A comparative study of minimal spanning tree algorithms," in *Proc. Int. Conf. Math., Comput. Eng. Comput. Sci. (ICMCECS)*, Mar. 2020, pp. 1–4.

[47] O. Elishco and A. Barg, "Capacity of dynamical storage systems," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 329–346, Jan. 2021.

ANAN ZHOU was born in 1991. He received the M.S. degree from the Wuhan University of Technology, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Electronic Information School, Wuhan University, Wuhan. His research interests include distributed storage and fountain codes.

BENSHUN YI was born in 1965. He received the Ph.D. degrees in electrical engineering from the Huazhong University of Science and Technology. He is currently a Professor of the Electronic Information School, Wuhan University, Wuhan, China. His research interests fall in power quality, wireless networks, and channel coding. He has published extensively in these areas.

YUSHENG LIU was born in 1993. He received the M.S. degree from the Kunming University of Science and Technology, Kunming, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electronic Information School, Wuhan University, Wuhan, China. His research interests include distributed storage and federated learning.

LAIGAN LUO was born in 1995. He received the M.S. degree from the Wuhan University of Technology, Wuhan, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electronic Information School, Wuhan University, Wuhan. His research interests include deep learning and computer vision.