SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

October 2018, Vol. 61 100303:1–100303:17
https://doi.org/10.1007/s11432-018-9499-0

Special Focus on Distributed Storage Coding

Storage and repair bandwidth tradeoff for distributed storage systems with clusters and separate nodes†

Jingzhao WANG, Tinghan WANG & Yuan LUO

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240*, China*

Received 10 March 2018/Revised 22 May 2018/Accepted 19 June 2018/Published online 20 August 2018

Abstract The optimal tradeoff between node storage and repair bandwidth is an important issue for distributed storage systems (DSSs). For realistic DSSs with clusters, while repairing a failed node, downloading more data from intra-cluster nodes than from cross-cluster nodes is effective. Therefore, differentiating the repair bandwidth from intra-cluster and cross-cluster is useful. For cluster DSSs, the tradeoff is considered with special repair assumptions where all alive nodes are used for repairing a failed node. In this paper, we investigate the optimal tradeoff for the cluster DSSs under more general storage/repair parameters. Furthermore, we propose a regenerating code construction strategy that achieves the points in the optimal tradeoff curve for the cluster DSSs with specific parameters as a numerical example. Moreover, we consider the influence of separate nodes for the tradeoff for the DSSs with clusters and separated nodes.

Keywords distributed storage system, cluster, separate node, repair bandwidth, communication cost

Citation Wang J Z, Wang T H, Luo Y. Storage and repair bandwidth tradeoff for distributed storage systems with clusters and separate nodes. Sci China Inf Sci, 2018, 61(10): 100303, https://doi.org/10.1007/s11432-018-9499-0

1 Introduction

With the expansion of data center storage, storage node failures are more prevalent [1], where distributed storage systems (DSSs) with erasure coding are widely used for ensuring data reliability [2, 3]. When a storage node in a DSS fails, to recover the failed node, a new node downloads data from others, called helper nodes. The amount of data to be downloaded is called the repair bandwidth. In [4], the tradeoff between node storage and bandwidth to repair one node is investigated for homogeneous DSSs [5] where all nodes (hard disks or other storage devices) have the same parameters (storage per node, repair bandwidth). Furthermore, regenerating codes are proposed based on the tradeoff for reducing the repair bandwidth of DSSs.

In contrast to homogeneous DSSs, in heterogeneous DSSs [5,6], nodes have different storage and repair bandwidths. In [7], the communication cost among nodes is considered, where the storage of each node is equal, but the repair bandwidth varies based on the location of the failed nodes. In practical storage systems, nodes in the same cluster (rack) can be connected to each other with cheaper and faster networks (i.e., local area networks) [8], where downloading data from each other may be faster and cheaper. For reducing the communication cost, downloading more data from intra-cluster nodes and less from cross-cluster nodes is efficient, rather than downloading the same amount of data from others.

* Corresponding author (email: yuanluo@sjtu.edu.cn)
† Invited paper

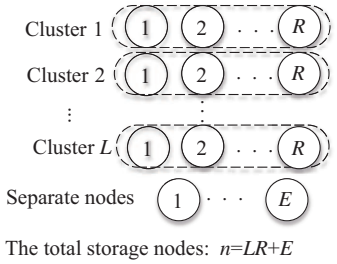

Figure 1 System model for CSN-DSS.

In [1,9,10], multi-rack (cluster) models are investigated, where there is a relay node in each rack, used for collecting data from nodes in the same rack and transmitting data to nodes in other racks, when repairing the failed nodes. The authors analyze the tradeoff between the storage and cross-rack repair bandwidth and propose coding strategies that minimize the cross-rack repair bandwidth. These strategies are deployed in hierarchical data centers, and the performance is analyzed in [1,10]. In [11], the authors also consider cluster DSSs with one relay node in each cluster with different model assumptions. The relay node is used for node repair and data collection. In [12], the tradeoff for DSSs with only two clusters (racks) and no relay nodes is analyzed. In [13], the properties of the cluster DSSs are considered without relay nodes and under an assumption that the new node downloads data from all the other alive nodes when one node has failed. Although this assumption maximizes the system storage capacity defined in [4], the reconstruction read cost is very high (all the alive node are used). Reconstruction read cost is defined in [3] as the expected number of nodes (blocks) required to serve a failed node (block).

Hence, analyzing the cluster DSSs in more general cases is beneficial, which is the main contribution of this paper. We investigate the optimal tradeoff between node storage and repair bandwidth on more flexible parameter settings for the cluster DSSs, where the new node does not download data from all the other alive nodes. The conventional homogeneous DSS model [4] and models in [12,13] are obtained by specializing parameters of our general model. By contrast, in real-world storage systems, there are many storage servers (nodes), which provokes research on heterogeneous DSSs [6]. For a general model for the cluster DSSs, considering the DSSs with clusters and separate nodes is flexible and adaptive, where the separate nodes function as supplements for the clusters in practical storage system scenarios [6,14]. We analyze the tradeoff for the DSSs with clusters and separate nodes, as well as investigate a regenerating code construction strategy for the cluster DSSs with specific parameters.

The remainder of this paper is organized as follows. Section 2 introduces the DSS model with clusters and separate node (CSN-DSS) and formulates the problem. Section 3 analyzes the properties of the cluster DSSs in two parts, which are proved in Theorems 1 and 2. In general, we analyze the DSS with clusters and one separate node in Theorem 3. Then, we characterize the tradeoff between node storage and repair bandwidth. Section 4 illustrates some numerical results, where we investigate a regenerating code construction strategy for cluster DSSs. Finally, Section 5 presents the conclusion and future work.

2 Preliminaries and problem formulation

Subsection 2.1 defines the main parameters of CSN-DSSs. As an efficient tool to analyze DSSs, Subsection 2.2 introduces the information flow graph. Subsection 2.3 formulates the problem investigated.

2.1 DSSs with cluster and separate nodes (CSN-DSS)

CSN-DSS model. The DSS with cluster and separate nodes, illustrated in Figure 1, consists of S separate nodes and L clusters each with R nodes. The total number of the storage nodes is $n = LR + E$. A data file of size $\mathcal{M}$ symbols is divided and encoded into n fragments of size α. Any k encoded fragments

out of n recover the original data file, called the (n,k) MDS[1)] property. If $\alpha = \mathcal{M}/k$ symbols, every k fragments of size $\mathcal{M}/k$ provide $\mathcal{M}$ symbols, the minimum data for recovering the file, which is optimal for storage redundancy [4].

When repairing a failed node, a newcomer located in the same cluster will download data from other alive nodes and repair the failed one. When the failed node is a separate node, the newcomer is still a separate node. It is called exact repair, when the failed nodes are exactly regenerated. In functional repair, the requirement is relaxed: the newcomer can contain different data from that of the failed node till the repaired system maintains the MDS property [15]. We consider that functional repair and MDS property holds for every repair procedure.

If a node in cluster has failed, the newcomer downloads β_I symbols each from d_I intra-cluster nodes and β_C symbols each from d_C cross-cluster nodes (including nodes from other clusters and separate nodes). Let

$$d \triangleq d_I + d_C.$$

As is illuminated by inequality (1) in [16], it is reasonable to restrict $k \leqslant d \leqslant n-1$, which is used in this paper.

In realistic distributed storage systems, the storage servers may connect to each other with local area networks or external networks. The communication between servers in the same local area network is cheaper than servers in different local area networks and connected with external networks. The servers connected in the same local area network can be seen as being in the same cluster. The separate nodes connect to cluster nodes by external networks. For reducing the bandwidth cost, downloading more data from nodes in the same cluster and less data from outer nodes, that is $\beta_I \geqslant \beta_C$, is beneficial. By contrast, when repairing failed nodes, intra-cluster nodes are usually used. Therefore, it is reasonable to assume that all intra-cluster alive nodes are used for repair, that is $d_I = R-1$.

If a separate node has failed, the newcomer downloads β_S symbols each from d other nodes, including the nodes in clusters and separate nodes, which indicates that the newcomer needs only d helper nodes to repair a failed node, regardless of the location of the failed node.

In a CSN-DSS, for simplicity, we call (n,k,L,R,E) the node parameters and $(\alpha, d_I, \beta_I, d_C, \beta_C, \beta_S)$ the storage/repair parameters. The intra-cluster and cross-cluster bandwidth of repairing a cluster node is defined as

$$\gamma_I \triangleq d_I\beta_I \quad \text{and } \gamma_C \triangleq d_C\beta_C,$$

respectively. The bandwidth of repairing a separate node is $\gamma_S \triangleq d\beta_S$. When $\beta_I = \beta_C = \beta_S$, the conventional homogeneous DSS in [4] is obtained.

2.2 Information flow graph

In [4], the information flow graph (IFG) is used for analyzing the performance of DSSs, which consists of three types of nodes: a single data source S, storage nodes x^i_{in}, x^i_{out}, and data collector DC, as shown in Figure 2(a). A physical storage node (hard disk or other storage device) is represented by a storage input node x^i_{in} and an output node x^i_{out}, where pre-computing is permitted when transmitting data. x^i_{in} and x^i_{out} are connected by a directed edge with capacity identical to the storage size α of the node. In this paper, we use x^i for presenting x^i_{in} and x^i_{out} as storage nodes.

The original data file is divided and encoded into n fragments, stored at n nodes, represented by n edges from node S to $\{x^i_{\text{in}}\}_{i=1}^n$ with infinite capacity. At the initial time, the source node S stores data to the n storage nodes. Then, S becomes inactive, and the n storage nodes become active. When node x^j fails, it becomes inactive. The repair procedure creates an active newcomer x^{n+1} to the graph as a substitute by connecting edges each with capacity β from d ($\geqslant k$) surviving active nodes, which indicates that the newcomer downloads β symbols from each of d alive nodes. The value of β varies in heterogeneous DSSs. The total number of active nodes remains n after each repair. Based on the (n,k)

1) An (n,k) maximum distance separate (MDS) code encodes k information symbols to n symbols such that any k symbols of n recover the original information symbols.

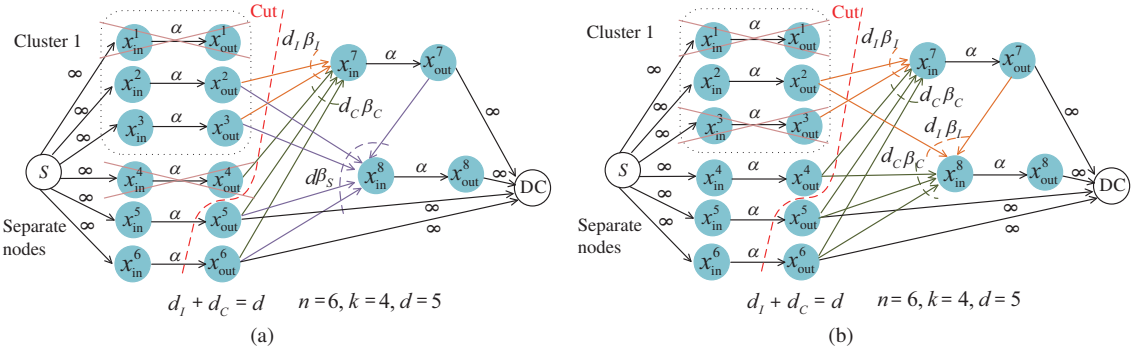

Figure 2 The IFGs of DSS with one cluster and separate nodes. (a) One node in cluster 1 and one separate node have failed; (b) two nodes in cluster 1 have failed.

MDS, data collector DC connects to arbitrary k active nodes with direct edges of infinite capacity, which indicates that any k nodes reconstructs the original data file. There are many DCs connecting different sets of k active nodes, but only one DC is drawn visually.

An example of the IFG for CSN-DSSs is illustrated in Figure 2(a), where $n = 6$ ($L = 1$, $R = 3$, $E = 3$), $k = 4$, $d = 5$. Two nodes x^1 and x^6 have failed successively. When node x^1 (a node in cluster 1) has failed, a newcomer x^7 downloads β_I symbols each from x^2 and x^3 (two intra-cluster nodes) and β_C symbols each from x^4, x^5 and x^6 (three cross-cluster nodes). When node x^6 has failed, a second newcomer x^8 is added, downloading β_S symbols from five alive nodes (nodes in clusters or separate nodes). This model only handles one node failure at a time, downloading data from d helper nodes, a subset of the $n-1$ alive nodes.

In an IFG, a cut between S and DC is defined as a subset $\mathcal{C}$ of edges, such that every directed path from S to DC contains at least one edge in $\mathcal{C}$. The min-cut is the cut between S and DC in which the total sum of the edge capacities is the smallest [4].

2.3 Problem formulation

According to [4], the reconstruction problem for every DC reduces exactly to multicasting the original data from a single source S to every DC. With relative studies on network coding [17, 18], a tradeoff between node storage and repair bandwidth is maintained by analyzing the min-cuts between S and all possible DCs.

Let $\mathcal{G}$ be the set of all possible IFGs of a CSN-DSS with node parameters (n, k, L, R, E) and storage/repair parameters $(\alpha, d_I, \beta_I, d_C, \beta_C, \beta_S)$. Consider any given finite IFG $G \in \mathcal{G}$, with a finite set of data collectors. If the minimum of the min-cuts separating the source with each data collector is larger than or equal to the data object size $\mathcal{M}$, then there exists a linear network code such that all data collectors recover the data object (see Proposition 1 in [4]). We denote the graph with minimum min-cut by G^*. The capacity of a CSN-DSS is defined as

$$\boldsymbol{C}(\mathcal{G}) \triangleq \text{min-cut of } G^*.$$

For sending data of size $\mathcal{M}$ from the source to any data collectors,

$$\boldsymbol{C}(\mathcal{G}) \geqslant \mathcal{M} \tag{1}$$

should be satisfied. Because $\boldsymbol{C}(\mathcal{G})$ depends on node parameters and storage/repair parameters, when node parameters (n, k, L, R, E) are fixed, a tradeoff between node storage α and repair bandwidth parameters $(d_I, \beta_I, d_C, \beta_C, \beta_S)$ is characterized. The set of points $(\alpha, d_I, \beta_I, d_C, \beta_C, \beta_S)$ that satisfy $\boldsymbol{C}(\mathcal{G}) \geqslant \mathcal{M}$ is feasible for reliably storing the original file of size $\mathcal{M}$.

Therefore, analyzing the capacity or tradeoff properties of CSN-DSSs requires analyzing the min-cuts of IFGs for given node parameters, which is explained in Sections 3 and 4.

3 CSN-DSSs analysis

We investigate the min-cuts of IFGs for the CSN-DSSs and prove the algorithms for generating the IFG achieving the capacity of given CSN-DSS. Some useful terms and notations are defined in Subsection 3.1. In Subsection 3.2, we consider the min-cuts of IFGs without separate selected nodes. With similar methods, we investigate the influence of one separate selected node in Subsection 3.3. The tradeoff bound of cluster DSSs is formulated in Subsection 3.4.

3.1 Terminologies and min-cut calculation

For a given IFG $G \in \mathcal{G}$, the main problem is to find the min-cuts between source S and each DC. Because there are no paths among different DCs, the min-cuts between S and each different DC can be analyzed in the same way. For simplicity, we assume the IFGs in Section 3 only contain one single DC, and the min-cut only indicates the min-cut separating S and DC. This subsection introduces important terminologies such as repair sequence, selected node distribution, cluster order, and relative location, with which the method for calculating min-cuts of IFGs is explained.

Topological order. Every directed acyclic graph has a topological order (see [19, Chapter 3]), which is an ordering of its vertices such that the existence of a path from v_i to v_j implies $i < j$. The k output nodes connected by every DC can be topologically sorted.

Min-cut between S and DC. Let $\{x_{\text{out}}^{t_i}\}_{i=1}^{k}$ be the set of output nodes connected by the data collector, which are topologically ordered. We calculate the min-cut between S and DC by cutting $\{x_{\text{out}}^{t_i}\}_{i=1}^{k}$ one by one in the topological order, which is proved in [4, Lemma 2]. Each time, cutting a node, a part of the min-cut is determined, called a part-cut value. Thus, the min-cut between S and DC is the summation of the k part-cut values. For example, in Figure 2(a), the DC connects to four output nodes x_{out}^{6}, x_{out}^{5}, x_{out}^{7}, x_{out}^{8} topologically ordered. By cutting the four nodes one by one, as shown by the red dashed line, we then obtain the four part-cut values α, α, $(\beta_C + 2\beta_I)(\leqslant \alpha)$, $(2\beta_C)(\leqslant \alpha)$. If $\beta_C + 2\beta_I \geqslant \alpha$, the cut line will be between x_{in}^{7} and x_{out}^{7}. As a result, the third part-cut value changes to α.

Repair sequence and selected nodes. When a DC connects to a newcomer instead of connecting to an original node, the part-cut value may be smaller than α. Hence, smaller min-cuts can be derived as the DC connects to more newcomers. Based on the MDS property mentioned in Subsection 2.1, a DC connects to k ($\leqslant n$) nodes. It is possible to find an IFG with a DC only connecting to k newcomers. Each newcomer corresponds to an original node failure and completes the repair procedure. Thus, the topological order of k output nodes $\{x_{\text{out}}^{t_i}\}_{i=1}^{k}$ corresponds to a repair sequence of original nodes. These original nodes contained in a repair sequence are called selected nodes.

In the homogeneous DSSs, all storage/repair parameters (α, β, d) are the same for different nodes. The repair sequence will not affect the minimum min-cut. As is proved in [4], when the DC connects to k newcomers and the newcomer x^{t_i} downloads data from all the former newcomers $\{x^{t_j}\}_{j=1}^{i-1}$, the minimum min-cut is reached.

However, in a CSN-DSS, the storage/repair parameters are different for nodes in cluster and separate nodes. Different repair sequences result in different min-cuts of the IFGs. As illustrated in Figure 2, there are two different repair sequences (x^1, x^4) in (a) and (x^1, x^3) in (b). The corresponding min-cuts are $2\alpha + 2\beta_I + \beta_C$ and $2\alpha + 2\beta_I + \beta_I$, respectively, as shown by the red dash cut lines. For simplicity, we only consider two newcomers and assume $2\beta_I$, β_C are less than α. Thus, the repair sequence directly determines the minimum min-cut.

Selected node distribution. For a given k selected nodes, without loss of generality, we assume the clusters are relabeled by the number of selected nodes in descending order. In other words, clusters 1 and L contain the most and least selected nodes, respectively. we define the selected node distribution as $\boldsymbol{s} = (s_0, s_1, s_2, \ldots, s_L)$, where s_i $(1 \leqslant i \leqslant L)$ is the number of selected nodes in cluster i, and the first component s_0 is the number of selected separate nodes. Furthermore, the set of all possible selected node

distributions is defined as

$$\mathcal{S}=\left\{\boldsymbol{s}=(s_0,s_1,s_2,\ldots,s_L):\ s_{i+1}\leqslant s_i,\ 0\leqslant s_i\leqslant R,\ \text{for } 1\leqslant i\leqslant L;\ 0\leqslant s_0\leqslant E;\ \sum_{i=0}^{L}s_i=k\right\}.$$

The selected node distribution describes the total number of selected nodes in each cluster and separate nodes. Moreover, we need to represent the topological order of k selected output nodes $\{x_{\text{out}}^{t_i}\}_{i=1}^{k}$ corresponding to k selected original nodes, called cluster order.

Cluster order. Let the cluster order $\boldsymbol{\pi}=(\pi_1,\pi_2,\ldots,\pi_k)$ denote the repair sequence, where π_i $(1\leqslant i\leqslant k)$ is the index of the cluster that contains the newcomer x^{t_i} corresponding to the failed node. If the i-th node is a separate node, π_i equals to 0. The cluster index is enough to define the repair sequence, because the storage/repair parameters for each node in the same cluster are same. For a particular selected node distribution $\boldsymbol{s}=(s_0,s_1,s_2,\ldots,s_L)$, there are different cluster orders. The set of possible cluster orders is defined as

$$\Pi(\boldsymbol{s})=\left\{\boldsymbol{\pi}=(\pi_1,\ldots,\pi_k):\ \sum_{j=1}^{k}\mathbb{I}(\pi_j=i)=s_i,\ i\in\{0,1,\ldots,L\}\right\},$$

where $\mathbb{I}(\pi_j=i)$ is an indicator function that equals 1 when $\pi_j=i$, and 0 otherwise.

Figure 3 illustrates the relationship between the selected node distribution and cluster order, where the selected nodes are numbered. The selected node distribution $\boldsymbol{s}=(1,4,3,1)$ indicates that, in the IFG of this CSN-DSS, the DC connects 1 separate node, 4 nodes from cluster 1, 3 nodes from cluster 2 and 1 node from cluster 3. The cluster order $\boldsymbol{\pi}(\boldsymbol{s})=(1,2,3,1,2,1,2,1,0)$ is a possible repair sequence for $\boldsymbol{s}$.

As shown in Figure 3, the selected nodes are labeled from 1 to k, although it is enough to record the cluster number in the cluster order as the nodes in one cluster are undifferentiated. For the nodes in a cluster order $\boldsymbol{\pi}$, it is also needed to identify the precedence of selected nodes in each cluster. We assume the i-th node in cluster order $\boldsymbol{\pi}$ is the $h_{\boldsymbol{\pi}}(i)$-th node in its cluster. We call $h_{\boldsymbol{\pi}}(i)$ the relative location of the i-th node and

$$h_{\boldsymbol{\pi}}(i)=\sum_{j=1}^{i}\mathbb{I}(\pi_j=\pi_i),\tag{2}$$

where $1\leqslant i\leqslant k$. For example, in Figure 3, the cluster order is $\boldsymbol{\pi}=(1,2,3,1,2,1,2,1,0)$ and $h_{\boldsymbol{\pi}}(4)=\mathbb{I}(\pi_1=\pi_4)+\mathbb{I}(\pi_2=\pi_4)+\mathbb{I}(\pi_3=\pi_4)+\mathbb{I}(\pi_4=\pi_4)=1+0+0+1=2$ where $\pi_4=1$. The corresponding sequence of $h_{\boldsymbol{\pi}}(i)$ is then $(1,1,1,2,2,3,3,4,1)$.

Calculating the min-cut between S and DC **of an IFG.** When calculating the min-cut of an IFG G, it is to cut the output nodes contacted by the DC k times in topological order. We will consider two disjoint sets U and $\overline{U}$ of the nodes in G. Assume S and the original nodes x^i $(1\leqslant i\leqslant n)$ are contained in U and DC is contained in $\overline{U}$ at the beginning. Every time we cut G, some nodes are added into U and $\overline{U}$ respectively. When G^* is cut k times, all the nodes of G^* are contained in U or $\overline{U}$ and the set of edges emanating from U to $\overline{U}$ is a cut between S and DC. Let $\mathcal{C}$ denote the edges in the cut set, i.e., the set of edges going from U to $\overline{U}$.

As illustrated in Figure 4, the k selected nodes $\{x^{t_i}\}_{i=1}^{k}$ are in topological order. When cutting node x^{t_1}, there are two possible cases.

• If $x_{\text{in}}^{t_1}$ is in U, the edge $(x_{\text{in}}^{t_1},x_{\text{out}}^{t_1})$ is contained in $\mathcal{C}$. The part-cut value equals α.

• If $x_{\text{in}}^{t_1}$ is in $\overline{U}$, since $x_{\text{in}}^{t_1}$ has an in-degree of $d=d_I+d_C$ and it is the topologically first newcomer in $\overline{U}$, all the incoming edges of $x_{\text{in}}^{t_1}$ must be in $\mathcal{C}$, which consists of d_I edges from intra-cluster nodes and d_C edges from cross-cluster nodes. The part-cut value equals $d_I\beta_I+d_C\beta_C$.

When cutting node x^{t_2}, the first case is similar to x^{t_1} and the part-cut value is also α. For the second case, if x^{t_2} is in the same cluster with x^{t_1}, the incoming edges of $x_{\text{in}}^{t_2}$ consist of d_I-1 edges from intra-cluster nodes and d_C edges from cross-cluster nodes. The part-cut value then equals to $(d_I-1)\beta_I+d_C\beta_C$. By contrast, if x^{t_2} is in different clusters with x^{t_1}, the incoming edges of $x_{\text{in}}^{t_2}$ still

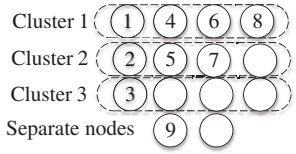

Figure 3 The numbered nodes are selected nodes and the selected node distribution is $\boldsymbol{s} = (1, 4, 3, 1)$. A corresponding cluster order is $\boldsymbol{\pi} = (1, 2, 3, 1, 2, 1, 2, 1, 0)$ for the CSN-DSS with $n = 15$ nodes, $k = 9$ selected nodes.

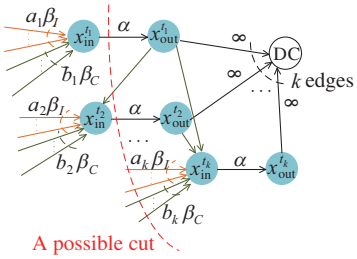

Figure 4 The min-cut IFG G without separate nodes in selected nodes.

contain d_I edges from intra-cluster nodes but $d_C - 1$ edges from cross-cluster nodes, and the part-cut value equals $d_I\beta_I + (d_C - 1)\beta_C$.

Then we consider node x^{t_i} $(1 \leqslant i \leqslant k)$, the i-th newcomer:

• If $x_{\text{in}}^{t_i} \in U$, the edge $(x_{\text{in}}^{t_i}, x_{\text{out}}^{t_i})$ must be in $\mathcal{C}$.

• If $x_{\text{in}}^{t_i} \in \overline{U}$, node $x_{\text{in}}^{t_i}$ has $d = d_I + d_C$ incoming edges consisting of two parts: edges from nodes in U and edges from nodes in $\overline{U}$. Cut set $\mathcal{C}$ only includes the first part of incoming edges, among which a_i denote the number of edges from intra-cluster nodes and b_i denote edges from cross-cluster nodes. It is obvious that $0 \leqslant a_i \leqslant d_I$ and $0 \leqslant b_i \leqslant d_C$. Note that when i increases by 1, either a_i or b_i will decrease by 1 and will not decrease when a_i or b_i equals 0. As at most $i - 1$ incoming edges of $x_{\text{in}}^{t_i}$ can be from $x_{\text{out}}^{t_j}$ $(1 \leqslant j \leqslant i - 1)$ already contained in $\overline{U}$,

$$a_i + b_i \geqslant d - (i - 1), \tag{3}$$

for $1 \leqslant i \leqslant k$. Equality holds if a_i and b_i will not decrease to 0 as i increases.

If the i-th selected node is a separate node, let c_i denote the number of incoming edges of $x_{\text{in}}^{t_i}$ from U and

$$c_i = d - (i - 1).$$

The respective values of a_i and b_i depend on the repair sequence of original nodes, the selected node distribution $\boldsymbol{s}$ and cluster order $\boldsymbol{\pi}$. The sum of the capacity of these edges is called the i-th part incoming weight

$$w_i(\boldsymbol{s}, \boldsymbol{\pi}) = \begin{cases} a_i\beta_I + b_i\beta_C, & \text{if the } i\text{-th selected node is a cluster node,} \\ c_i\beta_S, & \text{if the } i\text{-th selected node is a separate node.} \end{cases} \tag{4}$$

If the selected node distribution $\boldsymbol{s}$ or cluster order $\boldsymbol{\pi}$ is fixed beforehand, $w_i(\boldsymbol{s}, \boldsymbol{\pi})$ can be written as $w_i(\boldsymbol{\pi})$ or w_i for simplicity. On the other hand, a_i, b_i, c_i can be written as $a_i(\boldsymbol{\pi})$, $b_i(\boldsymbol{\pi})$, $c_i(\boldsymbol{\pi})$ for specific $\boldsymbol{\pi}$, respectively.

For a fixed selected node distribution $\boldsymbol{s}$, the min-cut varies for different cluster orders $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$. The min-cut for $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_k)$ is defined as

$$\text{MC}(\boldsymbol{s}, \boldsymbol{\pi}) \triangleq \sum_{i=1}^{k} \min\{w_i(\boldsymbol{\pi}), \alpha\}. \tag{5}$$

With the above definitions, for an IFG G with specified selected node distribution and cluster order, the min-cut can be figured out. In Subsection 3.2, we analyze the min-cuts of cluster DSSs without separate nodes.

3.2 The min-cuts of IFGs without separate selected nodes

In this subsection, we assume the selected nodes are all cluster nodes, namely, $\boldsymbol{s} = (s_0 = 0, s_1, \ldots, s_L)$, in which case, our CSN-DSS model is considered as a cluster DSS model. For given node parameters

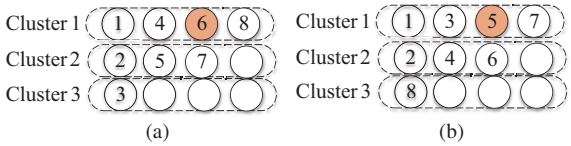

Figure 5 (Color online) The numbered nodes are selected nodes. There are two cluster orders $\boldsymbol{\pi}^*$ and $\boldsymbol{\pi}$ for a selected node distribution $\boldsymbol{s} = (0, 4, 3, 1)$. (a) $\boldsymbol{\pi}^* = (1, 2, 3, 1, 2, 1, 2, 1)$; (b) $\boldsymbol{\pi} = (1, 2, 1, 2, 1, 2, 1, 3)$.

(n, k, L, R, E), finding the IFG G^* with the minimum min-cut among all possible IFGs is equivalent to finding the corresponding selected node distribution $\boldsymbol{s}$ and cluster order $\boldsymbol{\pi}$. As $\boldsymbol{s}$ and $\boldsymbol{\pi}$ both influence the min-cuts, the analysis comprises of two steps:

(1) Fixing the selected node distribution $\boldsymbol{s}$ and analyzing the min-cuts for different cluster orders $\boldsymbol{\pi}$ (see the proof of vertical-order algorithm in Theorem 1);

(2) Fixing the cluster order generating algorithm and analyzing the min-cuts for different $\boldsymbol{s}$ (see the proof of horizontal selection algorithm in Therorm 2).

Vertical-order algorithm for $d_I = R - 1$. When the selected node distribution $\boldsymbol{s} = (s_0, s_1, \ldots, s_L)$ is fixed, the cluster order $\boldsymbol{\pi}^* = (\pi_1^*, \pi_2^*, \ldots, \pi_k^*)$ generated by the vertical-order algorithm (Algorithm 1) achieves the minimum min-cut among all the possible IFGs, which is proved in Theorem 1. In [13], the above conclusion is considered based on the special assumption that all the alive nodes are used for repairing the failed node, that is, $d_I = R - 1$ and $d_C = n - R$. Although this assumption maximizes the system storage capacity defined in [4], the reconstruction read cost under this assumption is very high (all the alive node are used). Reconstruction read cost is defined in [3] as the expected number of nodes (blocks) required to serve a failed node (block). Hence, we investigate and prove this problem in more general settings. The number of helper nodes from cross-cluster, d_C, varies from $k - R + 1$ to $n - R$ and does not need to be $n - R$ of [13], following from the condition that $k \leqslant d_I + d_C \leqslant n - 1$ and $d_I = R - 1$ (see Subsection 2.1).

Algorithm 1 Vertical-order algorithm

Require: $\boldsymbol{s} = (s_0, s_1, \ldots, s_L)$. Initial cluster label $j \leftarrow 1$.
Ensure: $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_k^*)$.
1: **for** $i = 1$ to k **do**
2: **if** the i-th selected node is a separate node **then**
3: $\pi_i^* \leftarrow 0$; continue;
4: **end if**
5: **if** $s_j = 0$ **then**
6: $j \leftarrow 1$;
7: **else**
8: $\pi_i^* \leftarrow j$; $s_j \leftarrow s_j - 1$; $j \leftarrow (j \bmod L) + 1$;
9: **end if**
10: **end for**

An example of Algorithm 1 is illustrated in Figure 5(a), where $\boldsymbol{s} = (0, 4, 3, 1)$. After three iterations, s_3 equals to 0 and $\pi_4^* = 1$ in the next iteration. The final output of the algorithm is $\boldsymbol{\pi}^* = (1, 2, 3, 1, 2, 1, 2, 1)$. When the selected node distribution $\boldsymbol{s}$ is fixed, a cluster order determines an IFG and the min-cut $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$ can be calculated. Note that $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$ depends on the i-th part incoming weight $w_i(\boldsymbol{\pi}) = a_i(\boldsymbol{\pi})\beta_I + b_i(\boldsymbol{\pi})\beta_C$ $(1 \leqslant i \leqslant k)$ and a useful property for $a_i(\boldsymbol{\pi})$ (the coefficient of β_I) is proved in Lemma 1.

Lemma 1. For a given selected node distribution $\boldsymbol{s} = (0, s_1, s_2, \ldots, s_L)$ of the system model in Figure 1, the multi-set[2)] $[a_i(\boldsymbol{\pi})]_{i=1}^k = [a_1(\boldsymbol{\pi}), a_2(\boldsymbol{\pi}), \ldots, a_k(\boldsymbol{\pi})]$ consists of the same elements for all the different cluster orders $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ and $a_i(\boldsymbol{\pi}) = d_I + 1 - h_{\boldsymbol{\pi}}(i)$ for $1 \leqslant i \leqslant k$.

Proof. Assume $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_k^*)$ and $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ are two cluster orders for the same selected node distribution $\boldsymbol{s} = (s_1, \ldots, s_L)$. For example, in Figure 5, the selected node distribution is $\boldsymbol{s} = (4, 3, 1)$

2) A multi-set is a generalization of the concept of a set that, unlike a set, allows multiple instances of the multi-set's elements.

and the corresponding two cluster orders are $\boldsymbol{\pi}^* = (1,2,3,1,2,1,2,1)$ and $\boldsymbol{\pi} = (1,2,1,2,1,2,1,3)$. The colored node 6 in Figure 5(a) is the third selected node in cluster 1. When cutting node 6, two intra-cluster nodes are contained in $\overline{U}$ (nodes 1 and 3), which are not counted in the part-cut value. Then, $a_6(\boldsymbol{\pi}^*) = d_I - 2 = R - 1 - 2 = 1$. Now consider cluster order $\boldsymbol{\pi}$ in Figure 5(b). Although the colored node 5 is the 5th node in $\boldsymbol{\pi}$, it is the third selected node in cluster 1 and $a_5(\boldsymbol{\pi}) = d_I - 2 = R - 1 - 2 = 1$. We observe that the value of $a_i(\boldsymbol{\pi}^*)$ only depends on the number of selected nodes in the same cluster before repairing the current node, which is defined by the relative location $h_{\boldsymbol{\pi}^*}(i)$, namely, $a_i(\boldsymbol{\pi}^*) = d_I + 1 - h_{\boldsymbol{\pi}^*}(i)$ for $1 \leqslant i \leqslant k$.

When $\boldsymbol{s} = (0, s_1, \ldots, s_L)$ is fixed, the set of $h_{\boldsymbol{\pi}}(i)$ for nodes in cluster l is $\{1, 2, \ldots, s_l\}$ for $1 \leqslant l \leqslant L$, no matter where the nodes are located in the cluster orders. For all cluster orders $\pi \in \Pi(\boldsymbol{s})$, the multi-set $[a_i(\pi)]_{i=1}^k$ consists of L sets $\{d_I + 1 - 1, d_I + 1 - 2, \ldots, d_I + 1 - s_l\}$ for $1 \leqslant l \leqslant L$.

When the selected node distribution $\boldsymbol{s}$ is fixed, a property of the min-cuts of IFGs for different cluster orders is proved in Theorem 1.

Theorem 1. For the given node parameters (n, k, L, R, E) and any given selected node distribution $\boldsymbol{s} \in \mathcal{S}$ with $s_0 = 0$, the vertical cluster order $\boldsymbol{\pi}^*$ obtained by the vertical-order algorithm achieves the minimum min-cut among all the possible IFGs with $\boldsymbol{s}$. In other words,

$$\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi}^*) \leqslant \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$$

holds for arbitrary $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$. $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$ is defined by (5).

Proof. Assume $(w_{u_1}(\boldsymbol{\pi}), \ldots, w_{u_k}(\boldsymbol{\pi}))$ is a non-increasing order of elements in multi-set $[w_i(\boldsymbol{\pi})]_{i=1}^k$, namely, $w_{u_1}(\boldsymbol{\pi}) \geqslant \cdots \geqslant w_{u_k}(\boldsymbol{\pi})$. This proof consists of two parts. In Part 1, we will prove that

$$\sum_{i=k-t+1}^{k} w_{u_i}(\boldsymbol{\pi}^*) \leqslant \sum_{i=k-t+1}^{k} w_{u_i}(\boldsymbol{\pi}), \tag{6}$$

for any $1 \leqslant t \leqslant k$. Note that Eq. (6) indicates that the sum of the minimum t elements in multi-set $[w_i(\boldsymbol{\pi}^*)]_{i=1}^k$ is no more than the sum of the minimum t elements in $[w_i(\boldsymbol{\pi})]_{i=1}^k$ for any $1 \leqslant t \leqslant k$. With the help of (6), Part 2 completes the proof by considering the relationship between α and $w_i(\boldsymbol{\pi}^*)$ in (5).

Part 1. As $w_i(\boldsymbol{\pi}) = a_i(\boldsymbol{\pi})\beta_I + b_i(\boldsymbol{\pi})\beta_C$ (see (4)), we consider the coefficient of β_I and β_C, and let

$$\phi_i(\boldsymbol{\pi}) \triangleq a_i(\boldsymbol{\pi}) + b_i(\boldsymbol{\pi})$$

for simplicity. In the following part, we will compare $\phi_i(\boldsymbol{\pi}^*)$ and $\phi_i(\boldsymbol{\pi})$ one by one and proved the inequality (7), which is required to prove (6).

For any cluster order $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$, let sequence $(\phi_{t_1}(\boldsymbol{\pi}), \ldots, \phi_{t_k}(\boldsymbol{\pi}))$ denote the non-increasing order of the elements of multi-set $[\phi_i(\boldsymbol{\pi})]_{i=1}^k$, namely, $\phi_{t_1}(\boldsymbol{\pi}) \geqslant \cdots \geqslant \phi_{t_k}(\boldsymbol{\pi})$. Based on Algorithm 1, it is easy to verify that the sequences $(\phi_1(\boldsymbol{\pi}^*), \ldots, \phi_k(\boldsymbol{\pi}^*))$, $(b_1(\boldsymbol{\pi}^*), \ldots, b_k(\boldsymbol{\pi}^*))$ and $(w_1(\boldsymbol{\pi}^*), \ldots, w_k(\boldsymbol{\pi}^*))$ are all non-increasing. Assume p $(1 \leqslant p \leqslant k)$ is the integer that satisfies the following conditions:

$$b_p(\boldsymbol{\pi}^*) = 0 \text{ and } b_{p-1}(\boldsymbol{\pi}^*) > 0$$

meaning that exactly d_C selected cross-cluster nodes are already cut when cutting the p-th node by the cluster order $\boldsymbol{\pi}^*$. Then $\phi_i(\boldsymbol{\pi}^*) = a_i(\boldsymbol{\pi}^*) + b_i(\boldsymbol{\pi}^*) = d - i + 1$ for $1 \leqslant i \leqslant p$. For the remaining nodes in $\boldsymbol{\pi}^*$, $b_i(\boldsymbol{\pi}^*) = 0$ and $\phi_i(\boldsymbol{\pi}^*) = a_i(\boldsymbol{\pi}^*)$ $(i > p)$.

• When $1 \leqslant i \leqslant p$, as $\phi_i(\boldsymbol{\pi}) \geqslant d - i + 1$ (see (3)), $\phi_{t_i}(\boldsymbol{\pi}) \geqslant \phi_i(\boldsymbol{\pi}) \geqslant d - i + 1 = \phi_i(\boldsymbol{\pi}^*)$.

• When $p + 1 \leqslant i \leqslant k$, as $a_i(\boldsymbol{\pi}^*) \geqslant a_{i+1}(\boldsymbol{\pi}^*) \geqslant \cdots \geqslant a_k(\boldsymbol{\pi}^*)$, at most $k - i$ elements of multi-set $[a_i(\boldsymbol{\pi}^*)]_{i=1}^k$ are less than $a_i(\boldsymbol{\pi}^*)$.

▶ If $a_{t_i}(\boldsymbol{\pi}) < a_i(\boldsymbol{\pi}^*)$, we first assume $a_{t_j}(\boldsymbol{\pi}) < a_j(\boldsymbol{\pi}^*)$ for all $i+1 \leqslant j \leqslant k$ and will derive a contradiction. It is obvious that at least $k - i + 1$ elements of multi-set $[a_{t_i}(\boldsymbol{\pi})]_{i=1}^k$ are less than $a_i(\boldsymbol{\pi}^*)$. As is proved in Lemma 1, $[a_{t_i}(\boldsymbol{\pi})]_{i=1}^k$ and $[a_i(\boldsymbol{\pi})^*]_{i=1}^k$ contain the same elements, $[a_i(\boldsymbol{\pi})^*]_{i=1}^k$ then contains at least $k - i + 1$ elements of multi-set $[a_{t_i}(\boldsymbol{\pi})]_{i=1}^k$ are less than $a_i(\boldsymbol{\pi}^*)$, which a contradiction. There then exists at least one $a_{t_j}(\boldsymbol{\pi})$ $(i + 1 \leqslant j \leqslant k)$ not less than $a_i(\boldsymbol{\pi}^*)$, and $\phi_{t_i}(\boldsymbol{\pi}) \geqslant \phi_{t_j}(\boldsymbol{\pi}) \geqslant a_{t_j}(\boldsymbol{\pi}) \geqslant a_i(\boldsymbol{\pi}^*) = \phi_i(\boldsymbol{\pi})$.

▶ If $a_{t_i}(\boldsymbol{\pi}) \geqslant a_i(\boldsymbol{\pi}^*)$, $\phi_{t_i}(\boldsymbol{\pi}) = a_{t_i}(\boldsymbol{\pi}) + b_{t_i}(\boldsymbol{\pi}) \geqslant a_{t_i}(\boldsymbol{\pi}) > a_i(\boldsymbol{\pi}^*) = \phi_i(\boldsymbol{\pi}^*)$.

Then it can be proved that

$$\sum_{i=k-t+1}^{k} \big(a_i(\boldsymbol{\pi}^*) + b_i(\boldsymbol{\pi}^*)\big) = \sum_{i=k-t+1}^{k} \phi_i(\boldsymbol{\pi}^*) \leqslant \sum_{i=k-t+1}^{k} \phi_{t_i}(\boldsymbol{\pi}) \overset{\text{(a)}}{\leqslant} \sum_{i=k-t+1}^{k} \big(a_{u_i}(\boldsymbol{\pi}) + b_{u_i}(\boldsymbol{\pi})\big). \tag{7}$$

Note that $\sum_{i=k-t+1}^{k} a_{u_i}(\boldsymbol{\pi}) + b_{u_i}(\boldsymbol{\pi}) = \sum_{i=k-t+1}^{k} \phi_{u_i}(\boldsymbol{\pi})$ is the sum of t elements of multi-set $[\phi_i(\boldsymbol{\pi})]_{i=1}^{k}$. Inequality (a) is based on the fact that $\sum_{i=k-t+1}^{k} \phi_{t_i}(\boldsymbol{\pi})$ is the sum of the minimum t elements of $[\phi_i(\boldsymbol{\pi})]_{i=1}^{k}$, not greater than the sum of any t elements of $[\phi_i(\boldsymbol{\pi})]_{i=1}^{k}$.

With the above consequence, it can be proved that

$$\begin{aligned}
\sum_{i=k-t+1}^{k} w_{u_i}(\boldsymbol{\pi}^*) &= \sum_{i=k-t+1}^{k} w_i(\boldsymbol{\pi}^*) = \sum_{i=k-t+1}^{k} (a_i(\boldsymbol{\pi}^*)\beta_I + b_i(\boldsymbol{\pi}^*)\beta_C) \\
&= \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\beta_I + \left(\sum_{i=k-t+1}^{k} (a_i(\boldsymbol{\pi}^*) + b_i(\boldsymbol{\pi}^*)) - \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\right)\beta_C \\
&\overset{\text{(b)}}{\leqslant} \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\beta_I + \left(\sum_{i=k-t+1}^{k} (a_{u_i}(\boldsymbol{\pi}) + b_{u_i}(\boldsymbol{\pi})) - \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\right)\beta_C \\
&\overset{\text{(c)}}{\leqslant} \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\beta_I + \left(\sum_{i=k-t+1}^{k} a_{u_i}(\boldsymbol{\pi}) - \sum_{i=k-t+1}^{k} a_i(\boldsymbol{\pi}^*)\right)\beta_I + \sum_{i=k-t+1}^{k} b_{u_i}(\boldsymbol{\pi})\beta_C \\
&= \sum_{i=k-t+1}^{k} a_{u_i}(\boldsymbol{\pi})\beta_I + \sum_{i=k-t+1}^{k} b_{u_i}(\boldsymbol{\pi})\beta_C = \sum_{i=k-t+1}^{k} w_{u_i}(\boldsymbol{\pi}),
\end{aligned}$$

where (b) is based on inequality (7) and (c) is because of $\beta_I \geqslant \beta_C$.

Part 2. Assume there are t_1 elements in $[w_i(\boldsymbol{\pi}^*)]_{i=1}^{k}$ and t_2 elements in $[w_i(\boldsymbol{\pi})]_{i=1}^{k}$ greater than α.

• If $t_1 < t_2$, $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi}^*) = t_1\alpha + \sum_{i=t_1+1}^{t_2} w_i(\boldsymbol{\pi}^*) + \sum_{i=t_2+1}^{k} w_i(\boldsymbol{\pi}^*) \leqslant t_2\alpha + \sum_{i=t_2+1}^{k} w_{u_i}(\boldsymbol{\pi}) = \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$.

• If $t_1 = t_2$, it is easy to prove $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi}^*) \leqslant \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$, using (6).

• If $t_1 > t_2$, $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi}^*) = t_2\alpha + \sum_{i=t_2+1}^{t_1} \min\{w_i(\boldsymbol{\pi}^*), \alpha\} + \sum_{i=t_1+1}^{k} w_i(\boldsymbol{\pi}^*) \leqslant t_2\alpha + \sum_{i=t_2+1}^{k} w_i(\boldsymbol{\pi}^*) \leqslant t_2\alpha + \sum_{i=t_2+1}^{k} w_{u_i}(\boldsymbol{\pi}) = \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$.

The consequence of Theorem 1 can be verified by Algorithm 1 and the numerical examples illustrated in Figure 5(a) and (b). To analyse the influence of selected node distribution, for any input $\boldsymbol{s}$, let

$$\boldsymbol{\pi}^*(\boldsymbol{s}) = (\pi^*(\boldsymbol{s})_1, \pi^*(\boldsymbol{s})_2, \ldots, \pi^*(\boldsymbol{s})_k) \tag{8}$$

denote the unique cluster order generated by the vertical-order algorithm. In the following part, we investigate the min-cuts for different selected node distributions $\boldsymbol{s}$, where the cluster orders are $\boldsymbol{\pi}^*(\boldsymbol{s})$.

Horizontal selection algorithm for $d_I = R - 1$. The vertical-order algorithm generates the cluster order achieving the minimum min-cut for any given selected node distribution $\boldsymbol{s}$. In this part, we assume all the cluster orders are generated by the vertical-order algorithm and analyse the min-cuts for different selected node distributions. In Theorem 2, it is proved that the minimum min-cut among possible IFGs is achieved by the selected node distribution $\boldsymbol{s}^* = (s_0^*, s_1^*, \ldots, s_L^*)$ generated by the horizontal selection algorithm (Algorithm 2) and the cluster order $\boldsymbol{\pi}^*(\boldsymbol{s}^*)$ generated by the vertical-order algorithm.

In this subsection, we consider the situation without separate nodes, namely, $s_0^* = 0$, in which case, Algorithm 2 reduces to the algorithm proposed in [13]. We will investigate Algorithm 2 in more general cases. An example of this algorithm is illustrated in Figure 6(a), where $k = 8$, $R = 4$. Based on the horizontal algorithm, $s_1^* = R = 4$, $s_2^* = R = 4$ and $s_3 = k - 2R = 0$. Another property of $a_i(\boldsymbol{\pi})$, the coefficients of β_I, is proved in Lemma 2, when the horizontal selected algorithm is used.

Lemma 2. For the given node parameters (n, k, L, R, S), the coefficients of β_I satisfies that

$$a_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant a_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$$

Algorithm 2 Horizontal selection algorithm

The horizontal selected node distribution is $\boldsymbol{s}^* = (s_0^*, s_1^*, \ldots, s_L^*)$ $(\sum_{i=0}^{L} s_i^* = k)$, where

$$s_i^* = \begin{cases} R, & i \leqslant \left\lfloor \dfrac{k - s_0^*}{R} \right\rfloor, \\ k - \left\lfloor \dfrac{k - s_0^*}{R} \right\rfloor R, & i = \left\lfloor \dfrac{k - s_0^*}{R} \right\rfloor + 1, \\ 0, & i > \left\lfloor \dfrac{k - s_0^*}{R} \right\rfloor + 1. \end{cases}$$

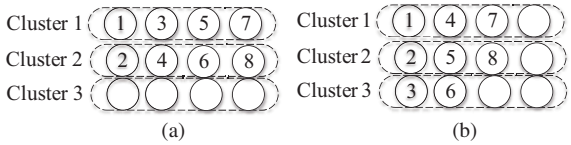

Figure 6 The numbered nodes are selected nodes. There are two selected node distributions $\boldsymbol{s}^*$ and $\boldsymbol{s}$ for the CSN-DSS ($n = 12$, $k = 8$, $L = 3$, $R = 4$, $E = 0$). (a) $\boldsymbol{s}^* = (0, 4, 4, 0)$; (b) $\boldsymbol{s} = (0, 3, 3, 2)$.

for $1 \leqslant i \leqslant k$, where $\boldsymbol{s}^*$ is the selected node distribution generated by the horizontal selection algorithm and $\boldsymbol{s} \in \mathcal{S}$ with $s_0 = 0$. Note that $\boldsymbol{\pi}^*(\cdot)$ is defined by (8).

Proof. As is proved in Lemma 1, the coefficient of β_I, $a_i(\boldsymbol{\pi}^*(\boldsymbol{s})) = d_I + 1 - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$. We will analyse the relative location $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$ with jumping points defined in (9) and prove that $h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) \geqslant h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$ for $1 \leqslant i \leqslant k$.

Based on the vertical-order algorithm, its easy to verify the following two properties of $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$:

- $1 \leqslant h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) \leqslant R$ for $1 \leqslant i \leqslant k$;
- $0 \leqslant h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i+1) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) \leqslant 1$ for $1 \leqslant i \leqslant k-1$;

meaning that $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$ is non-decreasing and will increase one time at most by 1. For example, the sequence of relative location $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$ is $(1, 1, 1, 2, 2, 3, 3, 4, 4)$ in Figure 6(a) and when $i = 3, 5$ or 7, the value of $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)$ increases by 1 for the next time. These values of i are called jumping points, denoted by

$$J(\boldsymbol{\pi}^*(\boldsymbol{s})) = (j_0(\boldsymbol{\pi}^*(\boldsymbol{s})), j_1(\boldsymbol{\pi}^*(\boldsymbol{s})), \ldots, j_{s_1-1}(\boldsymbol{\pi}^*(\boldsymbol{s})), j_{s_1}(\boldsymbol{\pi}^*(\boldsymbol{s}))), \tag{9}$$

which depends on $\boldsymbol{s} = (s_0, s_1, \ldots, s_k)$. We set $j_0(\boldsymbol{\pi}^*(\boldsymbol{s})) = 0$ and $j_{s_1}(\boldsymbol{\pi}^*(\boldsymbol{s})) = k$ as the beginning and ending of the jumping point vector, respectively, then

$$j_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_{i-1}(\boldsymbol{\pi}^*(\boldsymbol{s})) = \#\{t | h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(t) = i, 1 \leqslant t \leqslant k\}$$

for $1 \leqslant i \leqslant s_1$. Based on the definition of cluster order, it is obvious that

$$j_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_{i-1}(\boldsymbol{\pi}^*(\boldsymbol{s})) \geqslant j_{i+1}(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_i(\boldsymbol{\pi}^*(\boldsymbol{s})) \tag{10}$$

for $1 \leqslant i \leqslant s_1 - 1$.

We will use the induction method for proving $j_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ for $1 \leqslant i \leqslant s_1 - 1$. Based on the vertical-order algorithm, $j_1(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_1(\boldsymbol{\pi}^*(\boldsymbol{s}))$. Assume $j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_t(\boldsymbol{\pi}^*(\boldsymbol{s}))$, it is needed to prove $j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}))$.

There are $k - j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*))$ nodes remaining after the jumping point $j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*))$. Based on the horizontal selection algorithm,

$$j_{i+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) = j_1(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \text{ or } j_1(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - 1$$

for $1 \leqslant i \leqslant R - 1$. Then

$$k - j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \geqslant (R - t - 1)(j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - 1). \tag{11}$$

Assume

$$j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) > j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})). \tag{12}$$

As $j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_t(\boldsymbol{\pi}^*(\boldsymbol{s}))$, then

$$\begin{aligned} &j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s})) < j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \\ &\Rightarrow j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s})) \leqslant j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - 1. \end{aligned} \tag{13}$$

Then

$$\begin{aligned} k - j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})) &\overset{(a)}{\leqslant} (s_1 - t - 1)(j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}))) \\ &\overset{(b)}{\leqslant} (s_1 - t - 1)(j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - 1) \\ &\leqslant (R - t - 1)(j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - j_t(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) - 1) \\ &\overset{(c)}{\leqslant} k - j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)), \end{aligned}$$

where (a) is based on (10), (b) is because of (13) and (c) results from (11). Hence,

$$j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})^*) \leqslant j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s})),$$

contradicting assumption (12), and we prove that $j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_{t+1}(\boldsymbol{\pi}^*(\boldsymbol{s}))$. Since $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) = t$ for $j_{t-1}(\boldsymbol{\pi}^*(\boldsymbol{s})) \leqslant i \leqslant j_t(\boldsymbol{\pi}^*(\boldsymbol{s}))$ $(t = 1, 2, \ldots, s_1)$ and $j_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant j_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ for $1 \leqslant i \leqslant s_1 - 1$, we prove that

$$h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) \geqslant h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) \tag{14}$$

for $1 \leqslant i \leqslant k$. Hence, $a_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant a_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ for $1 \leqslant i \leqslant k$.

Theorem 2. For the given node parameters (n, k, L, R, E), when the selected node distribution $\boldsymbol{s}^*$ is generated by the horizontal selection algorithm and the corresponding cluster order is generated by the vertical-order algorithm, the min-cut of this IFG is not greater than any IFGs. In other words,

$$\mathrm{MC}(\boldsymbol{s}^*, \boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi}^*(\boldsymbol{s}))$$

for all $\boldsymbol{s} \in \mathcal{S}$ with $s_0 = 0$. Note that $\boldsymbol{\pi}^*(\cdot)$ is defined by (8). $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$ is defined by (5).

Proof. Based on the vertical-order algorithm, sequence $(w_1(\boldsymbol{\pi}^*(\boldsymbol{s})), \ldots, w_k(\boldsymbol{\pi}^*(\boldsymbol{s})))$ is non-increasing for all $\boldsymbol{s} \in \mathcal{S}$ and $s_0 = 0$. Similarly to the proof of Theorem 1, we will prove that

$$w_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant w_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$$

for $1 \leqslant i \leqslant k$. From the definition of $a_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$, $b_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ and $h_{\boldsymbol{\pi}}(i)$ (see (2)), we observe that

$$a_i(\boldsymbol{\pi}^*(\boldsymbol{s})) = d_I + 1 - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) \tag{15}$$

for $1 \leqslant i \leqslant k$. When $d_C - (i - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)) \geqslant 0$,

$$b_i(\boldsymbol{\pi}^*(\boldsymbol{s})) = d_C - (i - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)). \tag{16}$$

We assume $b_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ decreases to 0 when $i = i^*(\boldsymbol{s})$, where $i^*(\boldsymbol{s})$ is a function of $\boldsymbol{s}$. It will not decrease anymore and $w_i(\boldsymbol{\pi}^*(\boldsymbol{s})) = a_i(\boldsymbol{\pi}^*(\boldsymbol{s}))\beta_I$ for $i \geqslant i^*(\boldsymbol{s})$, where

$$i^*(\boldsymbol{s}) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i^*(\boldsymbol{s})) = d_C. \tag{17}$$

As is proved in Lemma 2 (14), $h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i) \leqslant h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i)$ for $1 \leqslant i \leqslant k$, then $i^*(\boldsymbol{s}^*) \geqslant i^*(\boldsymbol{s})$ based on (17).

• When $1 \leqslant i \leqslant i^*(\boldsymbol{s})$,

$$\begin{aligned} w_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - w_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) &= \big(a_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - a_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\big)\beta_I + \big(b_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - b_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\big)\beta_C \\ &\overset{(a)}{=} \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)\beta_I - \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)\beta_C \\ &= \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)(\beta_I - \beta_C) \overset{(b)}{\geqslant} 0. \end{aligned}$$

Note that (a) is based on (15) and (16). (b) comes from (14) and $\beta_I \geqslant \beta_C$. Then $w_i(\boldsymbol{\pi}^*(\boldsymbol{s})) \geqslant w_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)$.

• When $i^*(\boldsymbol{s})+1 \leqslant i \leqslant i^*(\boldsymbol{s}^*)$, $b_i(\boldsymbol{\pi}^*(\boldsymbol{s}))$ equals to 0 and will not decrease with i increasing, but

$$d_C - (i - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)) \leqslant 0. \tag{18}$$

Hence,

$$\begin{aligned} w_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - w_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*)) &= \big(a_i(\boldsymbol{\pi}^*(\boldsymbol{s})) - a_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\big)\beta_I - b_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\beta_C \\ &\overset{(c)}{\geqslant} \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)\beta_I - b_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\beta_C - (d_C - (i - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)))\beta_C \\ &= \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)\beta_I - \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)\beta_C \\ &= \big(h_{\boldsymbol{\pi}^*(\boldsymbol{s}^*)}(i) - h_{\boldsymbol{\pi}^*(\boldsymbol{s})}(i)\big)(\beta_I - \beta_C) \geqslant 0, \end{aligned}$$

where (c) bases on (18).

• When $i^*(\boldsymbol{s}^*)+1 \leqslant i \leqslant k$, $w_i(\boldsymbol{\pi}^*(\boldsymbol{s})) = a_i(\boldsymbol{\pi}^*(\boldsymbol{s}))\beta_I \geqslant a_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))\beta_I = w_i(\boldsymbol{\pi}^*(\boldsymbol{s}^*))$.

The consequence of Theorem 2 can be verified by Algorithm 2 and the numerical examples are illustrated in Figure 6(a) and (b). As is proved by Theorems 1 and 2, the capacity of a cluster DSS with given node parameters and storage/repair parameters is $\mathrm{MC}(\boldsymbol{s}^*, \boldsymbol{\pi}^*(\boldsymbol{s}^*))$. Based on (1), the tradeoff between node storage and repair bandwidth can be characterized, which is illustrated in Section 4 for specific numerical parameters.

3.3 The min-cuts of IFGs with separate selected nodes

As proved in preceding subsection, the horizontal selection algorithm and vertical-order algorithm generate the selected node distribution $\boldsymbol{s}^*$ and corresponding cluster order $\boldsymbol{\pi}^*(\boldsymbol{s}^*)$, achieving the minimum min-cut of the IFGs without separate selected nodes. In this subsection, we analyze the min-cuts of IFGs with one separate selected node in Theorem 3, namely, $s_0 = 1$ in $\boldsymbol{s}$.

Note that Theorem 1 focuses on the influence of cluster order $\boldsymbol{\pi}$ where the selected node distribution $\boldsymbol{s}$ is fixed. By contrast, Theorem 2 analyzes different selected node distributions when the cluster order generating algorithm is fixed. Theorem 3 combines these two aspects and investigates the situation where there is one separate selected node.

Theorem 3. For given node parameters (n, k, L, R, E), assume the j-th selected node is a separate node. The minimum min-cut of all the IFGs is achieved by the selected node distribution $\boldsymbol{s}^*$ generated by the horizontal selection algorithm with cluster order $\boldsymbol{\pi}^*(\boldsymbol{s}^*)$ generated by the vertical-order algorithm. In other words,

$$\mathrm{MC}(\boldsymbol{s}^*, \boldsymbol{\pi}^*(\boldsymbol{s}^*)) \leqslant \mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$$

for all $\boldsymbol{s} \in \mathcal{S}$ with $s_0 = 1$ and $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ with $\pi_j = 0$. Note that $\boldsymbol{\pi}^*(\cdot)$ is defined by (8). $\mathrm{MC}(\boldsymbol{s}, \boldsymbol{\pi})$ is defined by (5).

Due to space constraints, we just sketch the proof of Theorem 3 in the following part.

Cluster order assignment. Let $\overline{\boldsymbol{\pi}} = (\overline{\pi}_1, \overline{\pi}_1, \ldots, \overline{\pi}_k)$ denote a cluster order without separate selected nodes in its corresponding selected node distribution $\overline{\boldsymbol{s}}$. When the j-th selected node is a separate node, we assume $\boldsymbol{\pi}$ denote the new cluster order with

$$\pi_i = \begin{cases} \overline{\pi}_i, & \text{if } 1 \leqslant i < j, \\ 0, & \text{if } i = j, \\ \overline{\pi}_{i-1}, & \text{if } j < i \leqslant k. \end{cases}$$

As illustrated in Figure 7(b) and (c), node 3 is a separate selected node in $\boldsymbol{\pi}$, then $\pi_1 = \overline{\pi}_1$, $\pi_2 = \overline{\pi}_2$, $\pi_3 = 0$ and $\pi_i = \overline{\pi}_{i-1}$ $(i = 4, 5, 6, 7, 8)$. In Figure 7(a), the optimal selected node distribution is $\boldsymbol{s}^* = (1, 4, 3, 0)$, and the optimal cluster order is $\boldsymbol{\pi}^* = (1, 2, 0, 1, 2, 1, 2, 1)$ generated by the vertical-order algorithm. Note that the third node in the cluster order is a separate node, which is fixed beforehand.

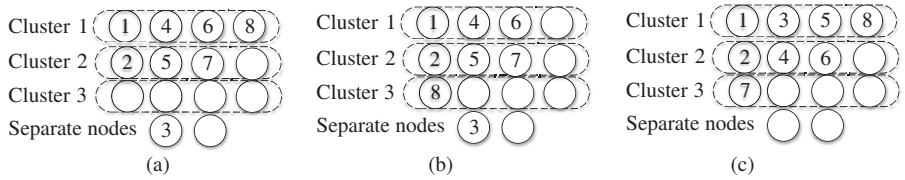

Figure 7 The numbered nodes are selected nodes. There are three selected node distributions $\boldsymbol{s}^* = (1,4,3,0)$, $\boldsymbol{s} = (1,3,3,1)$, and $\overline{\boldsymbol{s}} = (0,4,3,1)$, with cluster orders (a) $\boldsymbol{\pi}^* = (1,2,0,1,2,1,2,1)$, (b) $\boldsymbol{\pi} = (1,2,0,1,2,1,2,3)$, and (c) $\overline{\boldsymbol{\pi}} = (1,2,1,2,1,2,3,1)$, respectively, for CSN-DSS with node parameters ($n = 12$, $k = 9$, $L = 3$, $R = 4$, $S = 2$).

Main idea of the proof. For any cluster order $\boldsymbol{\pi}$ whose j-th node is a separate node, there always exists a cluster order $\overline{\boldsymbol{\pi}}$ without separate selected nodes satisfying the definition of $\boldsymbol{\pi}$. By investigating the relationship between $\overline{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}$, we reduce the proof of Theorem 3 to similar problems of Theorems 1 and 2, which is omitted here.

Theorem 3 is an essential property of min-cuts of IFGs with one separate selected node. More studies must be conducted for analyzing the minimum min-cut when separate nodes are considered. In addition, the location of the separate selected node (the value of j in Theorem 3) is important. Moreover, the relationship among β_I, β_C, and β_S must be considered for characterizing the final tradeoff. A situation with multiple separate selected nodes can be investigated using similar methods, which is not introduced here due to the space limitation.

3.4 Tradeoff between storage and repair bandwidth for cluster DSSs

Based on Theorems 1 and 2, the minimum min-cut of a cluster DSS with given node parameters (n,k,L,R,E) can be determined. As explained in Subsection 2.3, a tradeoff between node storage α and repair bandwidth parameters $(d_I,\beta_I,d_C,\beta_C)$ will be characterized with (1), namely,

$$\mathrm{MC}(\boldsymbol{s}^*,\boldsymbol{\pi}^*(\boldsymbol{s}^*)) = \sum_{i=1}^{k} \min\{w_i^*,\alpha\} \geqslant \mathcal{M}, \tag{19}$$

where $\mathcal{M}$ is the size of original file defined in (1), and let $w_i^* \triangleq w_i(\boldsymbol{s}^*,\boldsymbol{\pi}^*(\boldsymbol{s}^*))$ for simplicity. With Algorithms 1 and 2, we determine w_i^* as

$$w_{k-i+1}^* = \begin{cases} \left(R - \left\lceil \dfrac{i}{\lfloor \frac{k}{R} \rfloor + 1} \right\rceil\right)\beta_I + \left(d_C - i + \left\lceil \dfrac{i}{\lfloor \frac{k}{R} \rfloor + 1} \right\rceil\right)\beta_C, & 1 \leqslant i \leqslant \left(\left\lfloor \dfrac{k}{R} \right\rfloor + 1\right)\left(k - \left\lfloor \dfrac{k}{R} \right\rfloor R\right), \\ \left(\left\lfloor \dfrac{k-i}{\lfloor \frac{k}{R} \rfloor} \right\rfloor\right)\beta_I + \left(d_C + R - i - \left\lfloor \dfrac{k-i}{\lfloor \frac{k}{R} \rfloor} \right\rfloor\right)\beta_C, & \left(\left\lfloor \dfrac{k}{R} \right\rfloor + 1\right)\left(k - \left\lfloor \dfrac{k}{R} \right\rfloor R\right) < i \leqslant k. \end{cases} \tag{20}$$

We set the subscript of w^* in (20) as $k-i+1$ for ensuring $w_1^* \leqslant w_2^* \leqslant \cdots \leqslant w_k^*$ for simplifying the following formulas without changing the value of $\mathrm{MC}(\boldsymbol{s}^*,\boldsymbol{\pi}^*(\boldsymbol{s}^*))$. Let α^* denote the minimum α satisfying (19). By comparing w_i^* and α, we obtain the tradeoff bound (i.e., Figure 8) as follows. For $i = 2,\ldots,k$,

$$\alpha^* = \frac{\mathcal{M} - \sum_{j=1}^{i-1} w_j^*}{k-i+1}, \tag{21}$$

for $\mathcal{M} \in (\sum_{j=1}^{i-1} w_j^* + (k-i+1)w_{i-1}^*, \sum_{j=1}^{i} w_j^* + (k-i)w_i^*]$. When $i = 1$, $\alpha^* = \mathcal{M}/k$ for $\mathcal{M} \in [0, w_1^*]$.

Based on (20), w_i^* can be represented by the bandwidth parameters β_I and β_C. Hence, the tradeoff bound can be illustrated with (20) and (21). The detailed analysis is omitted here due to space limitation. In Section 4, some numerical examples are investigated in Figures 8 and 9. When separate nodes are considered, the tradeoff is different as the value of w_i^* changes, which will be investigated in future works.

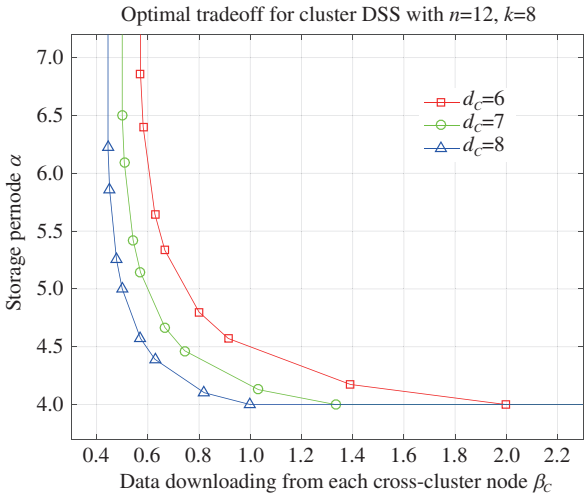

Figure 8 Optimal tradeoff curves between storage per node α and data downloading from each cross-cluster node β_C, for cluster DSS with $n = 12$, $k = 8$ and $\beta_I = 2\beta_C$. There are three curves for different number of cross-cluster helper nodes, namely, $d_C = 6, 7, 8$. The original file size is $\mathcal{M} = 32$.

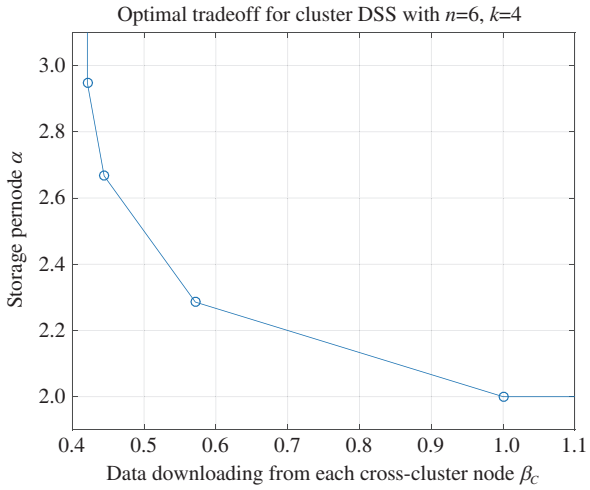

Figure 9 Optimal tradeoff curves between storage per node α and data downloading from each cross-cluster node β_C, for the cluster DSS with $n = 6$, $k = 4$, and $\beta_I = 2\beta_C$, $d_C = 3$. The original file size is $\mathcal{M} = 8$.

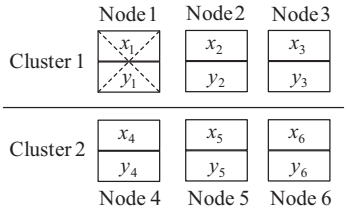

Figure 10 MSR construction for the cluster DSS with parameters $n = 6$, $k = 4$, $\beta_I = 2\beta_C$, $d_C = 3$ and $\mathcal{M} = 8$.

4 Numerical results and code constructions for cluster DSSs

In this section, Figure 8 illuminates some numerical capacity bounds for the cluster DSSs without separate nodes. As mentioned in [15, 20], interference alignment is an important method in regenerating code constructions, which is also applicative in thecluster DSSs. A code construction strategy with interference alignment is investigated for a cluster DSS with specific parameters as an example (see Figures 9 and 10). The code constructions for general cases can utilize similar methods.

Model configurations. We assume the node parameters are ($n = 12$, $k = 8$, $L = 3$, $R = 4$, $E = 0$), as shown in Figure 6. Note that different relationships between β_I and β_C result in different tradeoffs between storage per node α and bandwidth to repair one node. We consider a specific situation that $\beta_I = 2\beta_C$. As proved before, the cluster order illustrated in Figure 6(a) achieves the capacity of this DSS. Based on the bound constraint in (1), for specific values of $k - R + 1 \leqslant d_C \leqslant n - k$, the tradeoff between α and β_C is shown in Figure 8, where the file size $\mathcal{M}$ is set to be 32 for simplicity.

As illuminated in Figure 8, the tradeoff curve moves to the left as the number of cross-cluster helper nodes increases. Since α is the storage per node and β_C corresponds to the repair bandwidth, when the storage per node is fixed, the more helper nodes are utilized, the less bandwidth will be achieved, which is consistent with the consequences of the DSS without clusters in [4]. When $d_C = 8$, the point ($\alpha = 4$, $\beta_C = 2$) achieves the minimum storage, and the corresponding code constructions is called minimum-storage regenerating (MSR) codes, as defined in [4]. We will investigate an MSR construction for the cluster DSS with less nodes as an example.

We will consider the cluster DSS with $n = 6$, $k = 4$ in Figure 10. It can be verified in Figure 9 that

the MSR point of this system is ($\alpha = 2$, $\beta_C = 1$). Hence, the amount of data downloading from each intra-cluster node is $\beta_I = 2\beta_C = 2$.

Encoding procedure. As the original file size is $\mathcal{M} = 8$, we assume x_i ($1 \leqslant i \leqslant 4$) and y_i ($1 \leqslant i \leqslant 4$) are the original file symbols storing from nodes 1 to 4 as Figure 10 illustrates. Two (6,4)-MDS codes are used to encode symbols (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4), respectively. Let

$$\begin{aligned}(x_1, x_2, x_3, x_4, x_5, x_6) &= (x_1, x_2, x_3, x_4)\,[\boldsymbol{I}_{4\times4}|\boldsymbol{g}\ \boldsymbol{h}],\\ (y_1, y_2, y_3, y_4, y_5, y_6) &= (y_1, y_2, y_3, y_4)\,[\boldsymbol{I}_{4\times4}|\boldsymbol{g}'\ \boldsymbol{h}'],\end{aligned} \tag{22}$$

where $\boldsymbol{I}_{4\times4}$ is an identity matrix and $\boldsymbol{g} = (g_1, g_2, g_3, g_4)^t$, $\boldsymbol{h} = (h_1, h_2, h_3, h_4)^t$, $\boldsymbol{g}' = (g'_1, g'_2, g'_3, g'_4)^t$, $\boldsymbol{h} = (h'_1, h'_2, h'_3, h'_4)^t$. Then

$$\begin{aligned}x_5 &= g_1x_1 + g_2x_2 + g_3x_3 + g_4x_4, \quad y_5 = g'_1y_1 + g'_2y_2 + g'_3y_3 + g'_4y_4,\\ x_6 &= h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4, \quad y_6 = h'_1y_1 + h'_2y_2 + h'_3y_3 + h'_4y_4.\end{aligned}$$

Repair procedure. Assuming that node 1 has failed, based on the tradeoff in Figure 9 ($\alpha = 2$ and $\beta_C = 1$, $\beta_I = 2\beta_C = 2$), we will download 2 symbols each from nodes 2 and 3 respectively and download 1 symbols each from nodes 4 to 6. As the values of x_2, y_2, x_3, y_3 are known by downloading from nodes 2 and 3, to calculate x_1 and y_1, interference alignment can be used to eliminate x_4 and y_4. For example, the symbols downloading from nodes 4–6 are

$$\begin{aligned}&\text{symbol}_4 = l_4x_4 + l'_4y_4,\\ &\text{symbol}_5 = m_5x_5 + m'_5y_5 = m_5(g_1x_1 + g_2x_2 + g_3x_3 + g_4x_4) + m'_5(g'_1y_1 + g'_2y_2 + g'_3y_3 + g'_4y_4),\\ &\text{symbol}_6 = n_6x_6 + n'_6y_6 = n_6(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) + n'_6(h'_1y_1 + h'_2y_2 + h'_3y_3 + h'_4y_4),\end{aligned}$$

respectively. Hence,

$$\text{symbol}_4 = l_4x_4 + l'_4y_4, \tag{23}$$

$$\text{symbol}_5 - m_5(g_2x_2 + g_3x_3) - m'_5(g'_2y_2 + g'_3y_3) = m_5g_1x_1 + m'_5g'_1y_1 + m_5g_4x_4 + m'_5g'_4y_4, \tag{24}$$

$$\text{symbol}_6 - n_6(h_2x_2 + h_3x_3) - n'_6(h'_2y_2 + h'_3y_3) = n_6h_1x_1 + n'_6h'_1y_1 + n_6h_4x_4 + n'_6h'_4y_4. \tag{25}$$

Note that the left parts of (23)–(25) are real values. If the coefficients of x_1, y_1 and x_4, y_4 satisfy that

$$\text{rank}\left(\begin{bmatrix}m_5g_1 & m'_5g'_1\\ n_6h_1 & n'_6h'_1\end{bmatrix}\right) = 2 \ \text{ and } \ \text{rank}\left(\begin{bmatrix}l_4 & l'_4\\ m_5g_4 & m'_5g'_4\\ n_6h_4 & n'_6h'_4\end{bmatrix}\right) = 1, \tag{26}$$

x_4 and y_4 can be eliminated; in addition, x_1 and y_1 can be solved, finishing the repair of node 1.

As proved in [15], there exists MDS codes satisfying the condition (26). When other nodes have failed, similar methods can be utilized to generate the parameters of corresponding MDS codes. More future works are needed to generalize the construction of MDS codes to all the node failures for the cluster DSS with general parameters.

5 Conclusion and future work

In this paper, we investigate the DSSs with clusters and separate nodes, where the tradeoff between node storage and repair bandwidth is characterized on more flexible parameter settings. When a node in the cluster DSSs has failed, the number of helper nodes varies based on the practical storage system demands. The influence of separate nodes is also analysed for DSSs with clusters and separated nodes. Moreover, a regenerating code construction strategy is proposed for cluster DSSs with specific parameters as a numerical example, achieving the points in the optimal tradeoff curve.

More general and practical regenerating codes for the cluster DSSs need to be investigated further. Moreover, more works are needed to characterize the influence of separate selected nodes for the min-cuts of the CSN-DSSs more explicitly, when analyzing the optimal tradeoff between node storage and repair bandwidth. Furthermore, the constructions of more flexible regenerating codes for the CSN-DSSs are also useful for practical storage systems.

Acknowledgements This work was partially supported by National Natural Science Foundation of China (Grant No. 61571293), China Program of International S&T Cooperation (Grant No. 2016YFE0100300), and SJTU-CUHK Joint Research Collaboration Fund 2018. The authors would like to thank Prof. Kenneth W. Shum for the help in improving this paper.

References

1 Hu Y C, Li X L, Zhang M, et al. Optimal repair layering for erasure-coded data centers. ACM Trans Storage, 2017, 13: 1–24
2 Zhang H Y, Li H, Li R S. Repair tree: fast repair for single failure in erasure-coded distributed storage systems. IEEE Trans Parallel Distrib Syst, 2017, 28: 1728–1739
3 Huang C, Chen M H, Li J. Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems. In: Proceedings of the 6th IEEE International Symposium on Network Computing and Applications, Cambridge, 2007
4 Dimakis A G, Godfrey P B, Wu Y N, et al. Network coding for distributed storage systems. IEEE Trans Inf Theory, 2010, 56: 4539–4551
5 Ernvall T, Rouayheb S E, Hollanti C, et al. Capacity and security of heterogeneous distributed storage systems. IEEE J Sel Areas Commun, 2013, 31: 2701–2709
6 Yu Q, Shum K W, Sung C W. Tradeoff between storage cost and repair cost in heterogeneous distributed storage systems. Trans Emerg Telecommun Technol, 2015, 26: 1201–1211
7 Akhlaghi S, Kiani A, Ghanavati M R. Cost-bandwidth tradeoff in distributed storage systems. Comput Commun, 2010, 33: 2105–2115
8 Ford D, Labelle F, Popovici F I, et al. Availability in globally distributed storage systems. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, Vancouver, 2010
9 Hu Y C, Lee P P C, Zhang X Y. Double regenerating codes for hierarchical data centers. In: Proceedings of IEEE International Symposium on Information Theory (ISIT), Barcelona, 2016. 245–249
10 Hou H X, Lee P P C, Shum K W, et al. Rack-aware regenerating codes for data centers. 2018. ArXiv:1802.04031
11 Prakash N, Abdrashitov V, Médard M. The storage vs repair-bandwidth trade-off for clustered storage systems. IEEE Trans Inf Theory, 2017. ArXiv:1701.04909
12 Pernas J, Yuen C, Gastón B, et al. Non-homogeneous two-rack model for distributed storage systems. In: Proceedings of IEEE International Symposium on Information Theory (ISIT), Istanbul, 2013. 1237–1241
13 Sohn J Y, Choi B, Yoon S W, et al. Capacity of clustered distributed storage. In: Proceedings of IEEE International Conference on Communications (ICC), Paris, 2017
14 Kubiatowicz J, Weimer W, Wells C, et al. OceanStore: an architecture for global-scale persistent storage. SIGPLAN Notice, 2000, 35: 190–201
15 Dimakis A G, Ramchandran K, Yunnan Wu K, et al. A survey on network codes for distributed storage. Proc IEEE, 2011, 99: 476–489
16 Rashmi K V, Shah N B, Kumar P V. Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction. IEEE Trans Inf Theory, 2011, 57: 5227–5239
17 Li S Y R, Yeung R W, Ning Cai R W. Linear network coding. IEEE Trans Inf Theory, 2003, 49: 371–381
18 Ahlswede R, Cai N, Li R S, et al. Network information flow. IEEE Trans Inf Theory, 2000, 46: 1204–1216
19 Smith D K, Ahuja R K, Magnanti T L, et al. Network flows: theory, algorithms, and applications. J Oper Res Soc, 1994, 45: 1340
20 Goparaju S, Fazeli A, Vardy A. Minimum storage regenerating codes for all parameters. IEEE Trans Inf Theory, 2017, 63: 6318–6328