# Capacity of Clustered Distributed Storage

Jy-Yong Sohn, *Student Member, IEEE*, Beongjun Choi, *Student Member, IEEE*, Sung Whan Yoon, *Member, IEEE*, and Jaekyun Moon, *Fellow, IEEE*

***Abstract*— A new system model reflecting the clustered structure of distributed storage is suggested to investigate interplay between storage overhead and repair bandwidth as storage node failures occur. Large data centers with multiple racks/disks or local networks of storage devices (e.g., sensor network) are good applications of the suggested clustered model. In realistic scenarios involving clustered storage structures, repairing storage nodes using intact nodes residing in other clusters are more bandwidth consuming than restoring nodes based on information from intra-cluster nodes. Therefore, it is important to differentiate between intra-cluster repair bandwidth and cross-cluster repair bandwidth in modeling distributed storage. Capacity of the suggested model is obtained as a function of fundamental resources of distributed storage systems, namely, node storage capacity, intra-cluster repair bandwidth, and cross-cluster repair bandwidth. The capacity is shown to be asymptotically equivalent to a monotonic decreasing function of number of clusters, as the number of storage nodes increases without bound. Based on the capacity expression, feasible sets of required resources which enable reliable storage are obtained in a closed-form solution. Specifically, it is shown that the cross-cluster traffic can be minimized to zero (i.e., intra-cluster local repair becomes possible) by allowing extra resources on storage capacity and intra-cluster repair bandwidth, according to the law specified in the closed form. The network coding schemes with zero cross-cluster traffic are defined as *intra-cluster repairable codes*, which are shown to be a class of the previously developed *locally repairable codes*.**

***Index Terms*— Capacity, distributed storage, network coding.**

## I. Introduction

MANY enterprises, including Google, Facebook, Amazon and Microsoft, use cloud storage systems in order to support massive amounts of data storage requests from clients. In the emerging Internet-of-Thing (IoT) era, the number of devices which generate data and connect to the network increases exponentially, so that efficient management of data center becomes a formidable challenge. However, since cloud storage systems are composed of inexpensive commodity disks, failure events occur frequently, degrading the system reliability [2].

In order to ensure reliability of cloud storage, distributed storage systems (DSSs) with erasure coding have been considered to improve tolerance against storage node failures [3]–[8]. In such systems, the original file is encoded and distributed into multiple storage nodes. When a node fails, a newcomer node regenerates the failed node by contacting a number of survived nodes. This causes traffic burden across the network, taking up significant repair bandwidth. Earlier distributed storage systems utilized the 3-replication code: the original file was replicated three times, and the replicas were stored in three distinct nodes. The 3-replication coded systems require the minimum repair bandwidth, but incur high storage overhead. Reed-Solomon (RS) codes are also used (*e.g.* HDFS-RAID in Facebook [9]), which allow minimum storage overhead; however, RS-coded systems suffer from high repair bandwidth.

The pioneering work of [10] on distributed storage systems focused on the relationship between two required resources, the storage capacity $\alpha$ of each node and the repair bandwidth $\gamma$, when the system aims to reliably store a file $\mathcal{M}$ under node failure events. The optimal $(\alpha, \gamma)$ pairs are shown to have a fundamental trade-off relationship, to satisfy the maximum-distance-separable (MDS) property (i.e., any $k$ out of $n$ storage nodes can be accessed to recover the original file) of the system. Moreover, Dimakis *et al.* [10] obtained capacity $\mathcal{C}$, the maximum amount of reliably storable data, as a function of $\alpha$ and $\gamma$. Ahlswede *et al.* [11] related the failure-repair process of a DSS with the multi-casting problem in network information theory, and exploited the fact that a cut-set bound is achievable by network coding. Since the theoretical results of [10], explicit network coding schemes [12]–[14] which achieve the optimal $(\alpha, \gamma)$ pairs have also been suggested. These results are based on the assumption of homogeneous systems, i.e., each node has the same storage capacity and repair bandwidth.

However, in real data centers, storage nodes are dispersed into multiple clusters (in the form of disks or racks) [7], [8], [15], allowing high reliability against both node and rack failure events. In this clustered system, repairing a failed node gives rise to both intra-cluster and cross-cluster repair traffic. While the current data centers have abundant intra-rack communication bandwidth, cross-rack communication is typically limited. According to [16], nearly a 180TB of cross-rack repair bandwidth is required everyday in the Facebook warehouse, limiting cross-rack communication for

Manuscript received September 25, 2017; revised March 9, 2018 and April 30, 2018; accepted April 30, 2018. Date of publication May 17, 2018; date of current version December 19, 2018. This work was supported in part by the National Research Foundation of Korea under Grant 2016R1A2B4011298 and in part by the ICT Research and Development Program of MSIP/IITP (Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion) under Grant 2016-0-00563. This paper was presented at the 2017 IEEE Conference on Communications [1].

The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: jysohn1108@kaist.ac.kr; bbzang10@kaist.ac.kr; shyoon8@kaist.ac.kr; jmoon@kaist.edu).

Communicated by M. Schwartz, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2018.2837860

foreground map-reduce jobs. Moreover, surveys [17]–[19] on network traffic within data centers show that cross-rack communication is *oversubscribed*; the available cross-rack communication bandwidth is typically $5-20$ times lower than the intra-rack bandwidth in practical systems. Thus, a new system model which reflects the imbalance between intra- and cross-cluster repair bandwidths is required.

### A. Main Contributions

This paper suggests a new system model for *clustered DSS* to reflect the clustered nature of real distributed storage systems wherein an imbalance exists between intra- and cross-cluster repair burdens. This model can be applied to not only large data centers, but also local networks of storage devices such as the sensor networks or home clouds which are expected to be prevalent in the IoT era. This model is also more general in the sense that when the intra- and cross-cluster repair bandwidths are set to be equal, the resulting structure reduces to the original DSS model of [10]. This paper only considers recovering a single node failure at a time, as in [10]. The main contributions of this paper can be seen as twofold: one is the derivation of a closed-form expression for capacity, and the other is the analysis on feasible sets of system resources which enable reliable storage.

*1) Closed-Form Expression for Capacity:* Under the setting of functional repair, storage capacity $\mathcal{C}$ of the clustered DSS is obtained as a function of node storage capacity $\alpha$, intra-cluster repair bandwidth $\beta_I$ and cross-cluster repair bandwidth $\beta_c$. The existence of the cluster structure manifested as the imbalance between intra/cross-cluster traffics makes the capacity analysis challenging; Dimakis' proof in [10] cannot be directly extended to handle the problem at hand. We show that symmetric repair (obtaining the same amount of information from each helper node) is optimal in the sense of maximizing capacity given the storage node size and total repair bandwidth, as also shown in [20] for the case of varying repair bandwidth across the nodes. However, we stress that in most practical scenarios, the need is greater for reducing cross-cluster communication burden, and we show that this is possible by trading with reduced overall storage capacity and/or increasing intra-repair bandwidth. Based on the derived capacity expression, we analyzed how the storage capacity $\mathcal{C}$ changes as a function of $L$, the number of clusters. It is shown that the capacity is asymptotically equivalent to $\underline{C}$, some monotonic decreasing function of $L$.

*2) Analysis on Feasible $(\alpha, \beta_I, \beta_c)$ Points:* Given the need for reliably storing file $\mathcal{M}$, the set of required resource pairs, node storage capacity $\alpha$, intra-cluster repair bandwidth $\beta_I$ and cross-cluster repair bandwidth $\beta_c$, which enables $\mathcal{C}(\alpha, \beta_I, \beta_c) \geq \mathcal{M}$ is obtained in a closed-form solution. In the analysis, we introduce $\epsilon = \beta_c/\beta_I$, a useful parameter which measures the ratio of the cross-cluster repair burden (per node) to the intra-cluster burden. This parameter represents how scarce the available cross-cluster bandwidth is, compared to the abundant intra-cluster bandwidth. We here stress that the special case of $\epsilon = 0$ corresponds to the scenario where repair is done only locally via intra-cluster communication, i.e., when a node fails the repair process requires intra-cluster traffic only without any cross-cluster traffic. Thus, the analysis on the $\epsilon = 0$ case provides a guidance on the network coding for data centers for the scenarios where the available cross-cluster (cross-rack) bandwidth is very scarce.

Similar to the non-clustered case of [10], the required node storage capacity and the required repair bandwidth show a trade-off relationship. In the trade-off curve, two extremal points - the minimum-bandwidth-regenerating (MBR) point and the minimum-storage-regenerating (MSR) point - have been further analyzed for various $\epsilon$ values. Moreover, from the analysis on the trade-off curve, it is shown that the minimum storage overhead $\alpha = \mathcal{M}/k$ is achievable if and only if $\epsilon \geq \frac{1}{n-k}$. This implies that in order to reliably store file $\mathcal{M}$ with minimum storage $\alpha = \mathcal{M}/k$, sufficiently large cross-cluster repair bandwidth satisfying $\epsilon \geq \frac{1}{n-k}$ is required. Finally, for the scenarios with the abundant intra-cluster repair bandwidth, the minimum required cross-cluster repair bandwidth $\beta_c$ to reliably store file $\mathcal{M}$ is obtained as a function of node storage capacity $\alpha$.

### B. Related Works

Several researchers analyzed practical distributed storage systems with a goal in mind to reflect the non-homogeneous nature of storage nodes [20]–[31]. A heterogeneous model was considered in [20] and [21] where the storage capacity and the repair bandwidth for newcomer nodes are generally non-uniform. Upper/lower capacity bounds for the heterogeneous DSS are obtained in [20]. An asymmetric repair process is considered in [22], coining the terms, cheap and expensive nodes, based on the amount of data that can be transfered to any newcomer. Shah *et al.* [23] considered a flexible distributed storage system where the amount of information from helper nodes may be non-uniform, as long as the total repair bandwidth is bounded from above. The view points taken in these works are different from ours in that we adopt a notion of cluster and introduce imbalance between intra- and cross-cluster repair burdens.

Recently, some researchers considered the clustered structure of data centers [1], [24]–[31]. Some recent works [24]–[27] provided new system models for clustered DSS and shed light on fundamental aspects of the suggested system. In [24], the idea of [22] is developed to a two-rack system, by setting the communication burden within a rack much lower than the burden across different racks, similar to our analysis. However, Gastón *et al.* [24] only considered systems with two racks, while the current paper considers a general setting of $L$ racks (clusters), and provides mathematical analysis on how the number of clusters (i.e., the dispersion of nodes) affects the capacity of clustered distributed storage. Similar to the present paper, Prakash *et al.* [25], [26] obtained the capacity of clustered distributed storage, and provided capacity-achieving regenerating coding schemes. However, the coding schemes considered in [25] and [26] do not satisfy the MDS property, and the capacity expression is obtained for limited scenarios when intra-cluster repair bandwidth $\beta_I$ is set to its maximum value. In contrast, the current paper provides the capacity expression for general values of $\beta_I, \beta_c$

parameters, and analyzes the behavior of capacity as a function of the ratio $\epsilon = \beta_c/\beta_I$ between intra- and cross-cluster repair bandwidths. Moreover, unlike in the previous work, the capacity-achieving coding schemes (whose existence is shown) here satisfy the MDS property. In [27], the security issue in clustered distributed storage systems is considered, and the maximum amount of securely storable data in the existence of passive eavesdroppers is obtained.

There also have been some recent works [28]–[31] on network code design appropriate for clustered distributed storage. Motivated by the limited available cross-rack repair bandwidth in real data centers, the work of [28] provides a network coding scheme which minimizes the cross-rack bandwidth in clustered distributed storage systems. However, the suggested coding scheme is applicable for some limited $(n, k, L)$ parameters and the minimum storage overhead ($\alpha = \mathcal{M}/k$) setting. On the other hand, the current paper provides the capacity expression for general $(n, k, L, \alpha)$ setting, and proves the existence of capacity-achieving coding scheme. Calis and Koyluoglu [29] proposed coding schemes tolerant to rack failure events in multi-rack storage systems, but have not addressed the imbalance between intra- and cross-cluster repair burdens in node failure events, which is an important aspect of the current paper. Ye and Barg [30] considers the scenario of having grouped (clustered) storage nodes where nodes in the same group are more accessible to each other, compared to the nodes in other groups. However, the focus is different to the present paper: [30] focuses on the code construction which has the minimum amount of accessed data (called the optimal access property), while the scope of the present paper is on finding the optimal trade-off between the node storage capacity and the repair bandwidth, as a function of the ratio $\epsilon$ of intra- and cross-cluster communication burdens. Finally, a locally repairable code which can repair arbitrary node within each group is suggested in [31]; this code can suppress the inter-group repair bandwidth to zero. However, the coding scheme is suggested for the $\epsilon = 0$ case only, while the present paper provides the capacity expression for general $0 \leq \epsilon \leq 1$, and proves the existence of an optimal coding scheme.

Compared to the conference version [1] of the current work, this paper provides the formal proofs for the capacity expression, and obtains the feasible $(\alpha, \gamma)$ region for $0 \leq \epsilon \leq 1$ setting[1] (only $\epsilon = 0$ is considered in [1]). The present paper also shows the behavior of capacity as a function of $L$, the number of clusters, and provides the sufficient and necessary conditions on $\epsilon = \beta_c/\beta_I$, to achieve the minimum storage overhead $\alpha = \mathcal{M}/k$. Finally, the asymptotic behaviors of the MBR/MSR points are investigated in this paper, and the connection between what we call the intra-cluster repairable codes and the existing locally repairable codes [32] is revealed.

### C. Organization

This paper is organized as follows. Section II reviews preliminary materials about distributed storage systems and the information flow graph, an efficient tool for analyzing DSS.

[1]The reason why the present paper considers this regime is provided in Section III-B.

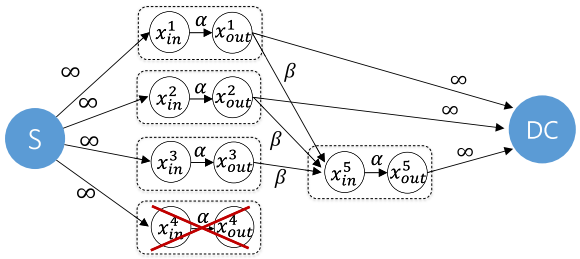


Fig. 1. Information flow graph ($n = 4, k = 3, d = 3$).

Section III proposes a new system model for the clustered DSS, and derives a closed-form expression for the storage capacity of the clustered DSS. The behavior of the capacity curves is also analyzed in this section. Based on the capacity expression, Section IV provides results on the feasible resource pairs which enable reliable storage of a given file. Further research topics on clustered DSS are discussed in Section V, and Section VI draws conclusions.

## II. Background

### A. Distributed Storage System

Distributed storage systems can maintain reliability by means of erasure coding [33]. The original data file is spread into $n$ potentially unreliable nodes, each with storage size $\alpha$. When a node fails, it is regenerated by contacting $d < n$ helper nodes and obtaining a particular amount of data, $\beta$, from each helper node. The amount of communication burden imposed by one failure event is called the repair bandwidth, denoted as $\gamma = d\beta$. When the client requests a retrieval of the original file, assuming all failed nodes have been repaired, access to any $k < n$ out of $n$ nodes must guarantee a file recovery. The ability to recover the original data using any $k < n$ out of $n$ nodes is called the maximal-distance-separable (MDS) property. Distributed storage systems can be used in many applications such as large data centers, peer-to-peer storage systems and wireless sensor networks [10].

### B. Information Flow Graph

Information flow graph is a useful tool to analyze the amount of information flow from source to data collector in a DSS, as utilized in [10]. It is a directed graph consisting of three types of nodes: data source S, data collector DC, and storage nodes $x^i$ as shown in Fig. 1. Storage node $x^i$ can be viewed as consisting of input-node $x^i_{in}$ and output-node $x^i_{out}$, which are responsible for the incoming and outgoing edges, respectively. $x^i_{in}$ and $x^i_{out}$ are connected by a directed edge with capacity identical to the storage size $\alpha$ of node $x^i$.

Data from source $S$ is stored into $n$ nodes. This process is represented by $n$ edges going from S to $\{x^i\}_{i=1}^n$, where each edge capacity is set to infinity. A failure/repair process in a DSS can be described as follows. When a node $x^j$ fails, a new node $x^{n+1}$ joins the graph by connecting edges from $d$ survived nodes, where each edge has capacity $\beta$. After all repairs are done, data collector DC chooses arbitrary $k$ nodes to retrieve data, as illustrated by the edges connected from

$k$ survived nodes with infinite edge capacity. Fig. 1 gives an example of information flow graph representing a distributed storage system with $n = 4, k = 3, d = 3$.

### C. Notation Used in the Paper

This paper requires many notations related to graphs, because it deals with information flow graphs. Here we provide the definition of each notation used in the paper. For the given system parameters, we denote $\mathcal{G}$ as the set of all possible information flow graphs. A graph $G \in \mathcal{G}$ is denoted as $G = (V, E)$ where $V$ is the set of vertices and $E$ is the set of edges in the graph. For a given graph $G \in \mathcal{G}$, we call a set $c \subset E$ of edges as *cut-set* [34] if it satisfies the following: every directed path from S to DC includes at least one edge in $c$. An arbitrary cut-set $c$ is usually denoted as $c = (U, \overline{U})$ where $U \subset V$ and $\overline{U} = V \setminus U$ (the complement of $U$) satisfy the following: the set of edges from $U$ to $\overline{U}$ is the given cut-set $c$. The set of all cut-sets available in $G$ is denoted as $C(G)$. For a graph $G \in \mathcal{G}$ and a cut-set $c \in C(G)$, we denote the sum of edge capacities for edges in $c$ as $w(G, c)$, which is called the *cut-value* of $c$.

A vector is denoted as $\mathbf{v}$ using the bold notation. For a vector $\mathbf{v}$, the transpose of the vector is denoted as $\mathbf{v}^T$. A set is denoted as $X = \{x_1, x_2, \cdots, x_k\}$, while a sequence $x_1, x_2, \cdots, x_N$ is denoted as $(x_n)_{n=1}^N$, or simply $(x_n)$. For given sequences $(a_n)$ and $(b_n)$, we use the term "$a_n$ is asymptotically equivalent to $b_n$" [35] if and only if

$$\lim_{n\to\infty} \frac{a_n}{b_n} = 1. \tag{1}$$

We utilize a useful notation:

$$\mathbb{1}_{i=j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

For a positive integer $n$, we use $[n]$ as a simplified notation for the set $\{1, 2, \cdots, n\}$. For a non-positive integer $n$, we define $[n] = \emptyset$. Each storage node is represented as either $x^t = (x^t_{in}, x^t_{out})$ as defined in Section II-B, or $N(i, j)$ as defined in (A.2). Finally, important parameters used in this paper are summarized in Table I.

TABLE I
PARAMETERS USED IN THIS PAPER

| | |
|---|---|
| $n$ | number of storage nodes |
| $k$ | number of DC-contacting nodes |
| $L$ | number of clusters |
| $n_I = n/L$ | number of nodes in a cluster |
| $d_I = n_I - 1$ | number of intra-cluster helper nodes |
| $d_c = n - n_I$ | number of cross-cluster helper nodes |
| $\mathbb{F}_q$ | base field which contains each symbol |
| $\alpha$ | storage capacity of each node |
| $\beta_I$ | intra-cluster repair bandwidth (per node) |
| $\beta_c$ | cross-cluster repair bandwidth (per node) |
| $\gamma_I = d_I\beta_I$ | intra-cluster repair bandwidth |
| $\gamma_c = d_c\beta_c$ | cross-cluster repair bandwidth |
| $\gamma = \gamma_I + \gamma_c$ | repair bandwidth |
| $\epsilon = \beta_c/\beta_I$ | ratio of $\beta_c$ to $\beta_I$ $(0 \le \epsilon \le 1)$ |
| $\xi = \gamma_c/\gamma$ | ratio of $\gamma_c$ to $\gamma$ $(0 \le \xi < 1)$ |
| $R = k/n$ | ratio of $k$ to $n$ $(0 < R \le 1)$ |

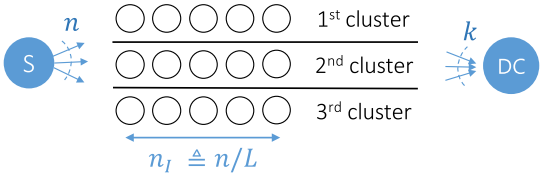


Fig. 2. Clustered distributed storage system ($n = 15, L = 3$).

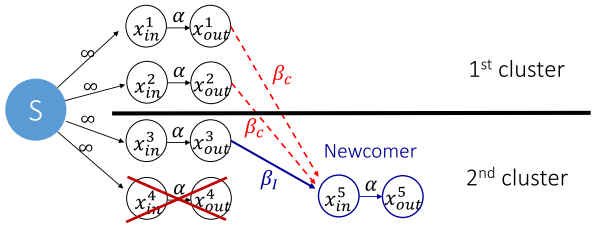


Fig. 3. Repair process in clustered DSS ($n = 4, L = 2, d_I = 1, d_c = 2$).

## III. CAPACITY OF CLUSTERED DSS

### A. Clustered Distributed Storage System

A distributed storage system with multiple clusters is shown in Fig. 2. Data from source S is stored at $n$ nodes which are grouped into $L$ clusters. The number of nodes in each cluster is fixed and denoted as $n_I = n/L$. The storage size of each node is denoted as $\alpha$. When a node fails, a newcomer node is regenerated by contacting $d_I$ helper nodes within the same cluster, and $d_c$ helper nodes from other clusters. This paper considers functional repair [33] in the regeneration process; the newcomer node may store different content from that of the failed node, while maintaining the MDS property of the code. The amount of data a newcomer node receives within the same cluster is $\gamma_I = d_I\beta_I$ (each node equally contributes to $\beta_I$), and that from other clusters is $\gamma_c = d_c\beta_c$ (each node equally contributes to $\beta_c$). Fig. 3 illustrates an example of information flow graph representing the repair process in a clustered DSS.

### B. Assumptions for the System

We assume that $d_c$ and $d_I$ have the maximum possible values ($d_c = n - n_I, d_I = n_I - 1$), since this is the capacity-maximizing choice, as formally stated in the following proposition. The proof of the proposition is in Appendix G-A.

*Proposition 1: Consider a clustered distributed storage system with given $\gamma$ and $\gamma_c$. Then, setting both $d_I$ and $d_c$ to their maximum values maximizes storage capacity.*

Note that Dimakis *et al.* [10] already showed that in the non-clustered scenario with given repair bandwidth $\gamma$, maximizing the number of helper nodes $d$ is the capacity-maximizing choice. Here, we are saying that a similar property also holds for clustered scenario considered in the present paper. Under the setting of the maximum number of helper nodes,

the overall repair bandwidth for a failure event is denoted as

$$\gamma = \gamma_I + \gamma_c = (n_I - 1)\beta_I + (n - n_I)\beta_c \tag{2}$$

Data collector DC contacts any $k$ out of $n$ nodes in the clustered DSS. Given that the typical intra-cluster communication bandwidth is larger than the cross-cluster bandwidth in real systems, we assume

$$\beta_I \geq \beta_c$$

throughout the present paper; this assumption limits our interest to $\epsilon \leq 1$. Moreover, motivated by the security issue, we assume that a file can be retrieved entirely by contacting a sufficiently large number of nodes. To be specific, the number $k$ of nodes contacted by the data collector satisfies

$$k \geq n_I. \tag{3}$$

We also assume

$$L \geq 2 \tag{4}$$

which holds for most real DSSs. Usually, all storage nodes cannot be squeezed in a single cluster, i.e., $L = 1$ rarely happens in practical systems, to prevent losing everything when the cluster is destroyed. Note that many storage systems [7], [8], [16] including those of Facebook uses $L = n$, i.e., every storage node reside in different racks (clusters), to tolerate the rack failure events. Finally, according to [16], nearly 98% of data recoveries in real systems deal with single node recovery. In other words, the portion of simultaneous multiple nodes failure events is small. Therefore, the present paper focuses single node failure events.

### C. The Closed-Form Solution for Capacity

Consider a clustered DSS with fixed $n, k, L$ values. In this model, we want to find the set of *feasible* parameters $(\alpha, \beta_I, \beta_c)$ which enables storing data of size $\mathcal{M}$. In order to find the feasible set, min-cut analysis on the information flow graph is required, similar to [10]. Depending on the failure-repair process and $k$ nodes contacted by DC, various information flow graphs can be obtained.

Let $\mathcal{G}$ be the set of all possible flow graphs. Consider a graph $G^* \in \mathcal{G}$ with minimum min-cut, the construction of which is specified in Appendix A. Based on the max-flow min-cut theorem in [11], the maximum information flow from source to data collector for arbitrary $G \in \mathcal{G}$ is greater than or equal to

$$\mathcal{C}(\alpha, \beta_I, \beta_c) := \text{min-cut of } G^*,$$

which is called the *capacity* of the system. In order to send data $\mathcal{M}$ from the source to the data collector, $\mathcal{C} \geq \mathcal{M}$ should be satisfied. Moreover, if $\mathcal{C} \geq \mathcal{M}$ is satisfied, there exists a linear network coding scheme [11] to store a file with size $\mathcal{M}$. Therefore, the set of $(\alpha, \beta_I, \beta_c)$ points which satisfies $\mathcal{C} \geq \mathcal{M}$ is *feasible* in the sense of reliably storing the original file of size $\mathcal{M}$. Now, we state our main result in the form of a theorem which offers a closed-form solution for the capacity $\mathcal{C}(\alpha, \beta_I, \beta_c)$ of the clustered DSS. Note that setting $\beta_I = \beta_c$ reduces to capacity of the non-clustered DSS obtained in [10].

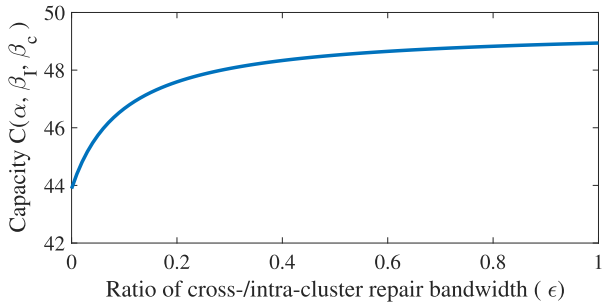


Fig. 4. Capacity as a function of $\epsilon$, under the setting of $n = 100$, $k = 85$, $L = 10$, $\alpha = 1$, $\gamma = 1$.

*Theorem 1: The capacity of the clustered distributed storage system with parameters $(n, k, L, \alpha, \beta_I, \beta_c)$ is*

$$\mathcal{C}(\alpha, \beta_I, \beta_c) = \sum_{i=1}^{n_I} \sum_{j=1}^{g_i} \min\{\alpha, \rho_i \beta_I + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m)\beta_c\}, \tag{5}$$

*where*

$$\rho_i = n_I - i, \tag{6}$$

$$g_m = \begin{cases} \lfloor \frac{k}{n_I} \rfloor + 1, & m \leq (k \bmod n_I) \\ \lfloor \frac{k}{n_I} \rfloor, & otherwise. \end{cases} \tag{7}$$

The proof is in Appendix A. Note that the parameters used in the statement of Theorem 1 have the following property, the proof of which is in Appendix G-B.

*Proposition 2: For every $(i, j)$ with $i \in [n_I]$, $j \in [g_i]$, we have*

$$\rho_i \beta_I + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m)\beta_c \leq \gamma. \tag{8}$$

*Moreover,*

$$\sum_{m=1}^{n_I} g_m = k \tag{9}$$

*holds.*

### D. Relationship Between $\mathcal{C}$ and $\epsilon = \beta_c/\beta_I$

In this subsection, we analyze the capacity of a clustered DSS as a function of an important parameter

$$\epsilon := \beta_c/\beta_I, \tag{10}$$

the cross-cluster repair burden per intra-cluster repair burden. In Fig. 4, capacity is plotted as a function of $\epsilon$. From (2), the total repair bandwidth can be expressed as

$$\begin{aligned} \gamma &= \gamma_I + \gamma_c = (n_I - 1)\beta_I + (n - n_I)\beta_c \\ &= \Big(n_I - 1 + (n - n_I)\epsilon\Big)\beta_I. \end{aligned} \tag{11}$$

Using this expression, the capacity is expressed as

$$\mathcal{C}(\epsilon) = \sum_{i=1}^{n_I} \sum_{j=1}^{g_i} \min\left\{\alpha, \frac{(n - \rho_i - j - \sum_{m=1}^{i-1} g_m)\epsilon + \rho_i}{(n - n_I)\epsilon + n_I - 1}\gamma\right\}. \tag{12}$$

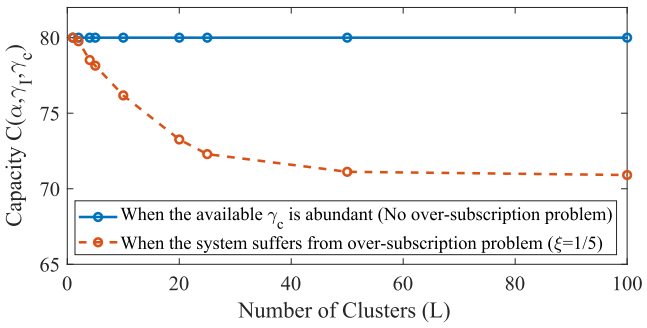


Fig. 5. Capacity as a function of $L$, under the setting of $n = 100$, $k = 80, \alpha = 1, \gamma = 10$.

For fair comparison on various $\epsilon$ values, capacity is calculated for a fixed $(n, k, L, \alpha, \gamma)$ set. The capacity is an increasing function of $\epsilon$ as shown in Fig. 4. This implies that for given resources $\alpha$ and $\gamma$, allowing a larger $\beta_c$ (until it reaches $\beta_I$) is always beneficial, in terms of storing a larger file. For example, under the setting in Fig. 4, allowing $\beta_c = \beta_I$ (i.e., $\epsilon = 1$) can store $\mathcal{M} = 48$, while setting $\beta_c = 0$ (i.e., $\epsilon = 0$) cannot achieve the same level of storage. This result is consistent with the previous work on asymmetric repair in [20], which proved that the symmetric repair maximizes capacity. Therefore, when the total communication amount $\gamma$ is fixed, a loss of storage capacity is the cost we need to pay in order to reduce the communication burden $\beta_c$ across different clusters.

### *E. Relationship Between $\mathcal{C}$ and $L$*

In this subsection, we analyze the capacity of a clustered DSS as a function of $L$, the number of clusters. For fair comparison, $(n, k, \alpha, \gamma)$ values are fixed for calculating capacity. In Fig. 5, capacity curves for two scenarios are plotted over a range of $L$ values. First, the solid line corresponds to the scenario when the system has abundant cross-rack bandwidth resources $\gamma_c$. In this ideal scenario which does not suffer from the over-subscription problem, the system can store $\mathcal{M} = 80$ irrespective of the dispersion of nodes.

However, consider a practical situation where the available cross-rack bandwidth is scarce compared to the intra-rack bandwidth; for example, $\xi = \gamma_c/\gamma = 1/5$. The dashed line in Fig. 5 corresponds to this scenario where the system has not enough cross-rack bandwidth resources. In this practical scenario, reducing $L$ (i.e., gathering the storage nodes into a smaller number of clusters) increases capacity. However, note that sufficient dispersion of data into a fair number of clusters is typically desired, in order to guarantee the reliability of storage in rack-failure events. Finding the optimal number $L^*$ of clusters in this trade-off relationship remains as an important topic for future research.

In Fig. 5, the capacity is a monotonic decreasing function of $L$ when the system suffers from an over-subscription problem. However, in general $(n, k, \alpha, \gamma)$ parameter settings, capacity is not always a monotonic decreasing function of $L$. Theorem 2 illustrates the behavior of capacity as $L$ varies, focusing on the special case of $\gamma = \alpha$. Before formally stating the next main result, we need to define

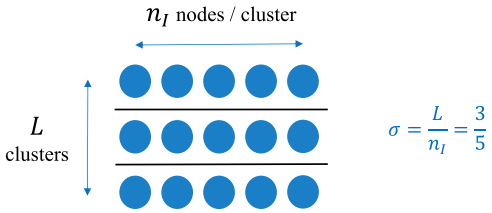


Fig. 6. An example of DSS with dispersion ratio $\sigma = 5/3$, when the parameters are set to $L = 3, n_I = 5, n = n_I L = 15$.

$$\sigma := \frac{L}{n_I} = \frac{L^2}{n}, \tag{13}$$

the *dispersion factor* of a clustered storage system, as illustrated in Fig. 6. In the two-dimensional representation of a clustered distributed storage system, $L$ represents the number of rows (clusters), while $n_I = n/L$ represents the number of columns (nodes in each cluster). The dispersion factor $\sigma$ is the ratio of the number of rows to the number of columns. If $L$ increases for a fixed $n_I$, then $\sigma$ grows and the nodes become more dispersed into multiple clusters.

Now we state our second main result, which is about the behavior of $\mathcal{C}$ versus $L$.

*Theorem 2: Consider the $\gamma = \alpha$ case when $\sigma, \gamma, R$ and $\xi$ are fixed. In the asymptotic regime of large $n$, capacity $\mathcal{C}(\alpha, \beta_I, \beta_c)$ is asymptotically equivalent to*

$$\underline{C} = \frac{k}{2}\left(\gamma + \frac{n-k}{n(1-1/L)}\gamma_c\right), \tag{14}$$

*a monotonically decreasing function of $L$. This can also be stated as*

$$\mathcal{C} \sim \underline{C} \tag{15}$$

*as $n \to \infty$ for a fixed $\sigma$.*

Note that under the setting of Theorem 2, we have

$$\begin{aligned} n_I &= \frac{n}{L} = \frac{n}{\sqrt{n\sigma}} = \Theta(\sqrt{n}), \\ k &= nR = \Theta(n), \quad \alpha = \gamma = \text{constant}, \\ \beta_I &= \frac{\gamma_I}{n_I - 1} = \Theta(\frac{1}{\sqrt{n}}), \quad \beta_c = \frac{\gamma_c}{n - n_I} = \Theta(\frac{1}{n}) \end{aligned}$$

in the asymptotic regime of large $n$. The proof of Theorem 2 is based on the following two lemmas.

*Lemma 1: In the case of $\gamma = \alpha$, capacity $\mathcal{C}(\alpha, \beta_I, \beta_c)$ is upper/lower bounded as*

$$\underline{C} \le \mathcal{C}(\alpha, \beta_I, \beta_c) \le \underline{C} + \delta \tag{16}$$

*where*

$$\delta = n_I^2(\beta_I - \beta_c)/8 \tag{17}$$

*and $\underline{C}$ is defined in (14).*

*Lemma 2: In the case of $\gamma = \alpha$, $\underline{C}$ and $\delta$ defined in (14), (17) satisfy*

$$\begin{aligned} \underline{C} &= \Theta(n), \\ \delta &= O(n_I) \end{aligned}$$

*when $\gamma, R$ and $\xi$ are fixed.*

The proofs of these lemmas are in Appendix H. Here we provide the proof of Theorem 2 by using Lemmas 1 and 2.

*Proof (of Theorem 2):* From Lemma 2,

$$\delta/\underline{C} = O(1/L) = O(\sqrt{1/n\sigma}) \to 0 \tag{18}$$

as $n \to \infty$ for a fixed $\sigma$. Moreover, dividing (16) by $\underline{C}$ results in

$$1 \le \mathcal{C}/\underline{C} \le 1 + \delta/\underline{C}. \tag{19}$$

Putting (18) into (19) completes the proof. □

## IV. Discussion on Feasible $(\alpha, \beta_I, \beta_c)$

In the previous section, we obtained the capacity of the clustered DSS. This section analyzes the feasible $(\alpha, \beta_I, \beta_c)$ points which satisfy $\mathcal{C}(\alpha, \beta_I, \beta_c) \ge \mathcal{M}$ for a given file size $\mathcal{M}$. Using

$$\gamma = \gamma_I + \gamma_c = (n_I - 1)\beta_I + (n - n_I)\beta_c$$

in (2), the behavior of the feasible set of $(\alpha, \gamma)$ points can be observed. Essentially, the feasible points demonstrate a trade-off relationship. Two extreme points – minimum storage regenerating (MSR) point and minimum bandwidth regenerating (MBR) point – of the trade-off has been analyzed. Moreover, a special family of network codes which satisfy $\epsilon = 0$, which we call the *intra-cluster repairable codes*, is compared with the *locally repairable codes* considered in [32]. Finally, the set of feasible $(\alpha, \beta_c)$ points is discussed, when the system allows maximum $\beta_I$.

### A. Set of Feasible $(\alpha, \gamma)$ Points

We provide a closed-form solution for the set of feasible $(\alpha, \gamma)$ points which enable reliable storage of data $\mathcal{M}$. Based on the range of $\epsilon$ defined in (10), the set of feasible points show different behaviors as stated in Corollary 1.

*Corollary 1: Consider a clustered DSS for storing data $\mathcal{M}$, when $\epsilon = \beta_c/\beta_I$ satisfies $0 \le \epsilon \le 1$. For any $\gamma \ge \gamma^*(\alpha)$, the data $\mathcal{M}$ can be reliably stored, i.e., $\mathcal{C} \ge \mathcal{M}$, while it is impossible to reliably store data $\mathcal{M}$ when $\gamma < \gamma^*(\alpha)$. The threshold function $\gamma^*(\alpha)$ can be obtained as:*

1) *if* $\frac{1}{n-k} \le \epsilon \le 1$

$$\gamma^*(\alpha) = \begin{cases} \infty, & \alpha \in (0, \frac{M}{k}) \\ \frac{M - t\alpha}{s_t}, & \alpha \in [\frac{M}{t+1+s_{t+1}y_{t+1}}, \frac{M}{t+s_t y_t}), \\ & \quad (t = k-1, k-2, \cdots, 1) \\ \frac{M}{s_0}, & \alpha \in [\frac{M}{s_0}, \infty) \end{cases} \tag{20}$$

2) *Otherwise (if* $0 \le \epsilon < \frac{1}{n-k}$*)*

$$\gamma^*(\alpha) = \begin{cases} \infty, & \alpha \in (0, \frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i}) \\ \frac{\mathcal{M} - \tau\alpha}{s_\tau}, & \alpha \in [\frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i}, \frac{\mathcal{M}}{\tau + s_\tau y_\tau}) \\ \frac{M - t\alpha}{s_t}, & \alpha \in [\frac{M}{t+1+s_{t+1}y_{t+1}}, \frac{M}{t+s_t y_t}), \\ & \quad (t = \tau - 1, \tau - 2, \cdots, 1) \\ \frac{M}{s_0}, & \alpha \in [\frac{M}{s_0}, \infty) \end{cases} \tag{21}$$

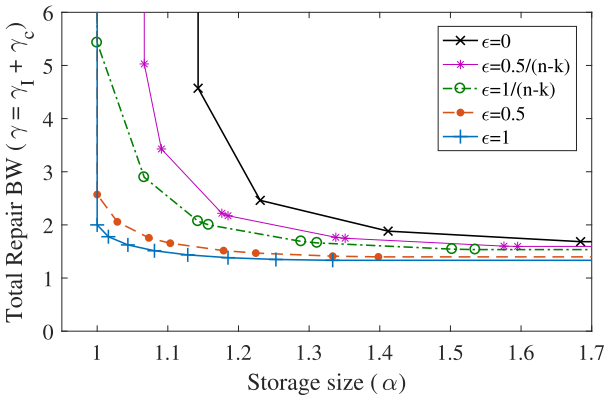


Fig. 7. Optimal tradeoff between node storage size $\alpha$ and total repair bandwidth $\gamma$, under the setting of $n = 15, k = 8, L = 3, \mathcal{M} = 8$.

*where*

$$\tau = \max\{t \in \{0, 1, \cdots, k-1\} : z_t \ge 1\} \tag{22}$$

$$s_t = \begin{cases} \frac{\sum_{i=t+1}^{k} z_i}{(n_I - 1) + \epsilon(n - n_I)}, & t = 0, 1, \cdots, k-1 \\ 0, & t = k \end{cases} \tag{23}$$

$$y_t = \frac{(n_I - 1) + \epsilon\ (n - n_I)}{z_t}, \tag{24}$$

$$z_t = \begin{cases} (n - n_I - t + h_t)\epsilon + (n_I - h_t), & t \in [k] \\ \infty, & t = 0 \end{cases} \tag{25}$$

$$h_t = \min\{s \in [n_I] : \sum_{l=1}^{s} g_l \ge t\}, \tag{26}$$

*and* $\{g_l\}_{l=1}^{n_I}$ *is defined in (7).*

An example for the trade-off results of Corollary 1 is illustrated in Fig. 7, for various $0 \le \epsilon \le 1$ values. Here, the $\epsilon = 1$ (i.e., $\beta_I = \beta_c$) case corresponds to the symmetric repair in the non-clustered scenario [10]. The plot for $\epsilon = 0$ (or $\beta_c = \gamma_c = 0$) shows that the cross-cluster repair bandwidth can be reduced to zero with extra resources ($\alpha$ or $\gamma_I$), where the amount of required resources are specified in Corollary 1. Note that in the case of $\epsilon = 0$, the storage system is *completely localized*, i.e., nodes can be repaired exclusively from their cluster. From Fig. 7, we can confirm that as $\epsilon$ decreases, extra resources ($\gamma$ or $\alpha$) are required to reliably store the given data $\mathcal{M}$. Moreover, Corollary 1 suggests a mathematically interesting result, stated in the following Theorem, the proof of which is in Appendix B.

*Theorem 3: A clustered DSS can reliably store file $\mathcal{M}$ with the minimum storage overhead $\alpha = \mathcal{M}/k$ if and only if*

$$\epsilon \ge \frac{1}{n-k}. \tag{27}$$

Note that $\alpha = \mathcal{M}/k$ is the minimum storage overhead which can satisfy the MDS property, as stated in [10]. The implication of Theorem 3 is shown in Fig. 7. Under the $\mathcal{M} = k = 8$ setting, data $\mathcal{M}$ can be reliably stored with minimum storage overhead $\alpha = \mathcal{M}/k = 1$ for $\frac{1}{n-k} \le \epsilon \le 1$, while it is impossible to achieve minimum storage overhead $\alpha = \mathcal{M}/k = 1$ for $0 \le \epsilon < \frac{1}{n-k}$. Finally, since reducing the

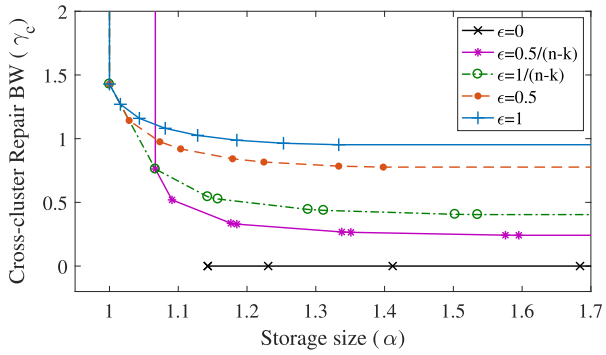


Fig. 8. Optimal tradeoff between node storage size $\alpha$ and cross-rack repair bandwidth $\gamma_c$, under the setting of $n = 15, k = 8, L = 3, \mathcal{M} = 8$.

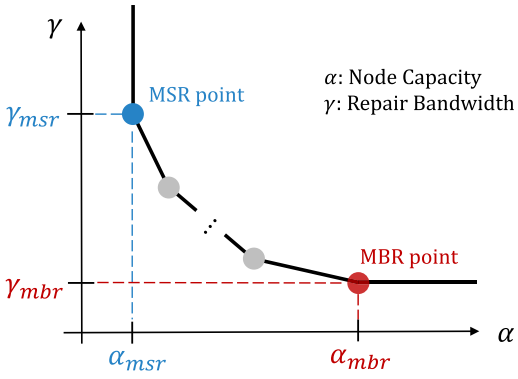


Fig. 9. Set of feasible $(\alpha, \gamma)$ points.

cross-cluster repair burden $\gamma_c$ is regarded as a much harder problem compared to reducing the intra-cluster repair burden $\gamma_I$, we also plotted feasible $(\alpha, \gamma_c)$ pairs for various $\epsilon$ values in Fig. 8. The plot for $\epsilon = 0$ obviously has zero $\gamma_c$, while $\epsilon > 0$ cases show a trade-off relationship. As $\epsilon$ increases, the minimum $\gamma_c$ value increases gradually.

### *B. Minimum-Bandwidth-Regenerating (MBR) Point and Minimum-Storage-Regenerating (MSR) Point*

According to Corollary 1, the set of feasible $(\alpha, \gamma)$ points shows a trade-off curve as in Fig. 9, for arbitrary $n, k, L, \epsilon$ settings. Here we focus on two extremal points: the minimum-bandwidth-regenerating (MBR) point and the minimum-storage-regenerating (MSR) point. As originally defined in [10], we call the point on the trade-off with minimum bandwidth $\gamma$ as MBR. Similarly, we call the point with minimum storage $\alpha$ as MSR.[2] Let $(\alpha^{(\epsilon)}_{msr}, \gamma^{(\epsilon)}_{msr})$ be the $(\alpha, \gamma)$ pair of the MSR point for given $\epsilon$. Similarly, define $(\alpha^{(\epsilon)}_{mbr}, \gamma^{(\epsilon)}_{mbr})$ as the parameter pair for MBR points. According to Corollary 1, the explicit $(\alpha, \gamma)$ expression for the MSR and MBR points are as in the following Corollary, the proof of which is given in Appendix F-B.

[2]Among multiple points with minimum $\gamma$, the point having the smallest $\alpha$ is called the MBR point. Similarly, among points with minimum $\alpha$, the point with the minimum $\gamma$ is called the MSR point.

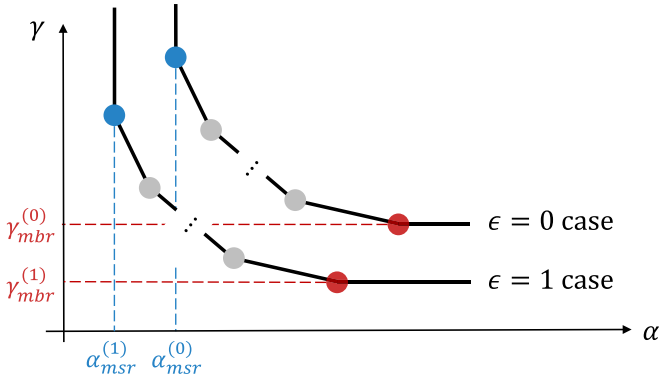


Fig. 10. Optimal trade-off curves for $\epsilon = 0, 1$.

*Corollary 2: For a given $0 \le \epsilon \le 1$, we have*

$$(\alpha^{(\epsilon)}_{msr}, \gamma^{(\epsilon)}_{msr}) = \begin{cases} \left(\frac{\mathcal{M}}{\tau+\sum_{i=\tau+1}^{k} z_i}, \frac{\mathcal{M}}{\tau+\sum_{i=\tau+1}^{k} z_i} \frac{\sum_{i=\tau+1}^{k} z_i}{s_\tau}\right), \\ \quad 0 \le \epsilon < \frac{1}{n-k} \\ \left(\frac{\mathcal{M}}{k}, \frac{\mathcal{M}}{k} \frac{1}{s_{k-1}}\right), \\ \quad \frac{1}{n-k} \le \epsilon \le 1 \end{cases} \tag{28}$$

$$(\alpha^{(\epsilon)}_{mbr}, \gamma^{(\epsilon)}_{mbr}) = \left(\frac{\mathcal{M}}{s_0}, \frac{\mathcal{M}}{s_0}\right). \tag{29}$$

Now we compare the MSR and MBR points for two extreme cases of $\epsilon = 0$ and $\epsilon = 1$. Using $R := k/n$ and the dispersion ratio $\sigma$ defined in (13), the asymptotic behaviors of MBR and MSR points are illustrated in the following theorem, the proof of which is in Appendix C.

*Theorem 4: Consider the MSR point $(\alpha^{(\epsilon)}_{msr}, \gamma^{(\epsilon)}_{msr})$ and the MBR point $(\alpha^{(\epsilon)}_{mbr}, \gamma^{(\epsilon)}_{mbr})$ for $\epsilon = 0, 1$. The minimum node storage for $\epsilon = 0$ is asymptotically equivalent to the minimum node storage for $\epsilon = 1$, i.e.,*

$$\alpha^{(0)}_{msr} \sim \alpha^{(1)}_{msr} = \mathcal{M}/k \tag{30}$$

*as $n \to \infty$ for arbitrary fixed $\sigma$ and $R = k/n$. Moreover, the MBR point for $\epsilon = 0$ approaches the MBR point for $\epsilon = 1$, i.e.,*

$$(\alpha^{(0)}_{mbr}, \gamma^{(0)}_{mbr}) \to (\alpha^{(1)}_{mbr}, \gamma^{(1)}_{mbr}), \tag{31}$$

*as $R = k/n \to 1$. The ratio between $\gamma^{(0)}_{mbr}$ and $\gamma^{(1)}_{mbr}$ is expressed as*

$$\frac{\gamma^{(0)}_{mbr}}{\gamma^{(1)}_{mbr}} \le 2 - \frac{k-1}{n-1} \tag{32}$$

Note that under the setting of Theorem 4, we have

$$\alpha^{(1)}_{msr} = \text{constant}, \quad k = nR = \Theta(n), \quad \mathcal{M} = \Theta(n)$$

in the asymptotic regime of large $n$. According to Theorem 4, the minimum storage for $\epsilon = 0$ can achieve $\mathcal{M}/k$ as $n \to \infty$ with fixed $R = k/n$. This result coincides with the result of Theorem 3. According to Theorem 3, the sufficient and necessary condition for achieving the minimum storage of $\alpha = \mathcal{M}/k$ is

$$\epsilon \ge \frac{1}{n-k} = \frac{1}{n(1-R)}. \tag{33}$$

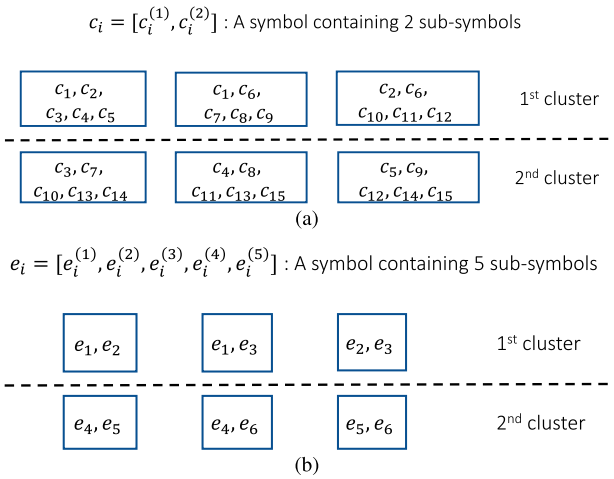


Fig. 11. MBR coding schemes for $n = 6, k = 5, L = 2, \alpha = 10$ [sub-symbol], $\gamma = 10$ [sub-symbol].

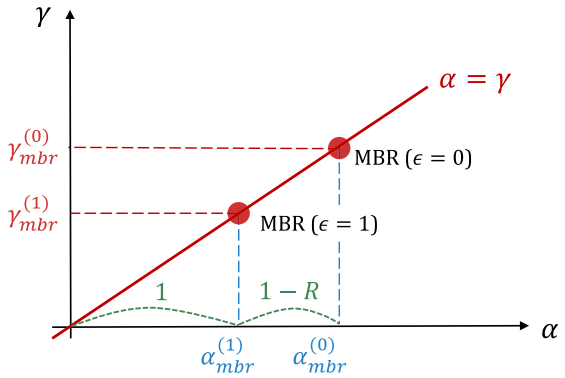


Fig. 12. Relationship between MBR point of $\epsilon = 0$ and that of $\epsilon = 1$.

As $n$ increases with a fixed $R$, the lower bound on $\epsilon$ reduces, so that in the asymptotic regime, $\epsilon = 0$ can achieve $\alpha = \mathcal{M}/k$.

Moreover, Theorem 4 states that the MBR point for $\epsilon = 0$ approaches the MBR point for $\epsilon = 1$ as $R = k/n$ goes to 1. Fig. 11 provides two MBR coding schemes with $(n, k, L) = (6, 5, 2)$, which has different $\epsilon$ values; one coding scheme in Fig. 11a satisfies $\epsilon = 1$, while the other in Fig. 11b satisfies $\epsilon = 0$. The RSKR coding scheme [12] is applied to the six nodes in Fig. 11a. Each node (illustrated as a rectangular box) contains five $c_i$ symbols, where each symbol $c_i$ consists of two sub-symbols, $c_i^{(1)}$ and $c_i^{(2)}$. Note that any symbol $c_i$ is shared by exactly two nodes in Fig. 11a, which is due to the property of RSKR coding. This system can reliably store fifteen symbols $\{c_i\}_{i=1}^{15}$, or $\mathcal{M} = 30$ sub-symbols $\{c_i^{(1)}, c_i^{(2)}\}_{i=1}^{15}$, since it satisfies two properties – the exact repair property and the data recovery property – as illustrated below. First, when a node fails, five other nodes transmit five symbols (one distinct symbol by each node), which exactly regenerates the failed node. Second, we can retrieve data, $\mathcal{M} = 30$ sub-symbols $\{c_i^{(1)}, c_i^{(2)}\}_{i=1}^{15}$, by contacting any $k = 5$ nodes. In Fig. 11b, each node contains two $e_i$ symbols, where each symbol $e_i$ consists of five sub-symbols, $\{e_i^{(j)}\}_{j=1}^{5}$. Note that in Fig. 11b, any symbol $e_i$ is shared by exactly two nodes which reside in the same cluster. This is because we applied RSKR coding at each cluster in the system of Fig. 11b. This system can reliably store six symbols $\{e_i\}_{i=1}^{6}$, or $\mathcal{M} = 30$ sub-symbols $\bigcup_{i\in[6], j\in[5]}\{e_i^{(j)}\}$, since it satisfies the exact repair property and the data recovery property.

Note that both DSSs in Fig. 11 reliably store $\mathcal{M} = 30$ sub-symbols, by using the node capacity of $\alpha = 10$ sub-symbols and the repair bandwidth of $\gamma = 10$ sub-symbols. However, the former system requires $\gamma_c = 6$ cross-cluster repair bandwidth for each node failure event, while the latter system requires $\gamma_c = 0$ cross-cluster repair bandwidth. For example, if the leftmost node of the $1^{st}$ cluster fails in Fig. 11a, then four sub-symbols $\{c_1^{(i)}, c_2^{(i)}\}_{i=1}^{2}$ are transmitted within that cluster, while six sub-symbols $\{c_3^{(i)}, c_4^{(i)}, c_5^{(i)}\}_{i=1}^{2}$ are transmitted from the $2^{nd}$ cluster. In the case of $\epsilon = 0$ in Fig. 11b, ten sub-symbols $\{e_1^{(i)}, e_2^{(i)}\}_{i=1}^{5}$ are transmitted within the $1^{st}$ cluster and no sub-symbols are transmitted across the clusters, when the leftmost node of the $1^{st}$ cluster fails. Thus, transition from the former system (Fig. 11a) to the latter system (Fig. 11b) reduces the cross-cluster repair bandwidth to zero, while maintaining the storage capacity $\mathcal{M}$ and the required resource pair $(\alpha, \gamma)$. Likewise, we can reduce the cross-cluster repair bandwidth to *zero* while maintaining the storage capacity, in the case of $R = k/n \to 1$.

Note that $\gamma_{mbr}^{(\epsilon)} = \alpha_{mbr}^{(\epsilon)}$ for $0 \le \epsilon \le 1$, from (29). Thus, the result of (32) in Theorem 4 can be expressed as Fig. 12 at the asymptotic regime of large $n, k$. According to Fig. 12, intra-cluster only repair ($\epsilon = 0$ or $\beta_c = 0$) is possible by using additional resources ($\alpha$ and $\gamma$) in the $1 - R$ portion, compared to the symmetric repair ($\epsilon = 1$) case.

### C. Intra-Cluster Repairable Codes Versus Locally Repairable Codes [32]

Here, we define a family of network coding schemes for clustered DSS, which we call the *intra-cluster repairable codes*. In Corollary 1, we considered DSSs with $\epsilon = 0$, which can repair any failed node by using intra-cluster communication only. The optimal $(\alpha, \gamma)$ trade-off curve which satisfies $\epsilon = 0$ is illustrated as the solid line with cross markers in Fig. 7. Each point on the curve is achievable (i.e., there exists a network coding scheme), according to the result of [11]. We call the network coding schemes for the points on the curve of $\epsilon = 0$ the *intra-cluster repairable codes*, since these coding schemes can repair any failed node by using intra-cluster communication only. The relationship between the intra-cluster repairable codes and the locally repairable codes (LRC) of [32] are investigated in Theorem 5, the proof of which is given in Appendix D. Note that according to the definition in [32], an $(n, l_0, m_0, \mathcal{M}, \alpha)$-LRC encodes a file of size $\mathcal{M}$ into $n$ coded symbols, where each symbol contains $\alpha$ bits. In addition, any coded symbol of the LRC is regenerated by accessing at most $l_0$ other symbols (i.e., the code has repair locality of $l_0$), while the minimum distance of the code is $m_0$.

*Theorem 5: The intra-cluster repairable codes with storage overhead $\alpha$ are the $(n, l_0, m_0, \mathcal{M}, \alpha)$-LRC codes of [32] where*

$$l_0 = n_I - 1,$$
$$m_0 = n - k + 1.$$

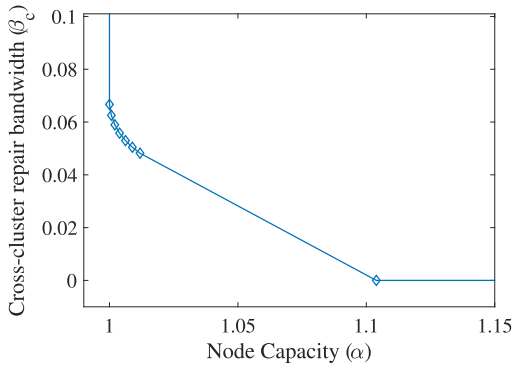


Fig. 13. Optimal tradeoff between cross-cluster repair bandwidth and node capacity, when $n = 100, k = 85, L = 10, \mathcal{M} = 85$ and $\beta_I = \alpha$.

*It is confirmed that Theorem 1 of [32],*

$$m_0 \leq n - \left\lceil \frac{\mathcal{M}}{\alpha} \right\rceil - \left\lceil \frac{\mathcal{M}}{l_0 \alpha} \right\rceil + 2, \tag{34}$$

*holds for every intra-cluster repairable code with storage overhead $\alpha$. Moreover, the equality of (34) holds if*

$$\alpha = \alpha^{(0)}_{msr}, (k \bmod n_I) \neq 0.$$

### D. Required $\beta_c$ for a Given $\alpha$

Here we focus on the following question: when the available intra-cluster repair bandwidth is abundant, how much cross-cluster repair bandwidth is required to reliably store file $\mathcal{M}$? We consider scenarios when the intra-cluster repair bandwidth (per node) has its maximum value, i.e., $\beta_I = \alpha$. Under this setting, Theorem 6 specifies the minimum required $\beta_c$ which satisfies $\mathcal{C}(\alpha, \beta_I, \beta_c) \geq \mathcal{M}$. The proof of Theorem 6 is given in Appendix E.

*Theorem 6: Suppose the intra-cluster repair bandwidth is set at the maximum value, i.e., $\beta_I = \alpha$. For a given node capacity $\alpha$, the clustered DSS can reliably store data $\mathcal{M}$ if and only if $\beta_c \geq \beta_c^*$ where*

$$\beta_c^* = \begin{cases} \frac{\mathcal{M}-(k-1)\alpha}{n-k}, & \text{if } \alpha \in [\frac{\mathcal{M}}{k}, \frac{\mathcal{M}}{f_{k-1}}) \\ \frac{\mathcal{M}-m\alpha}{\sum_{i=m+1}^{k}(n-i)}, & \text{if } \alpha \in [\frac{\mathcal{M}}{f_{m+1}}, \frac{\mathcal{M}}{f_m}) \\ & (m = k-2, k-3, \cdots, s+1) \\ \frac{\mathcal{M}-k_0\alpha}{\sum_{i=k_0+1}^{k}(n-i)}, & \text{if } \alpha \in [\frac{\mathcal{M}}{f_{k_0+1}}, \frac{\mathcal{M}}{k_0}) \\ 0, & \text{if } \alpha \in [\frac{\mathcal{M}}{k_0}, \infty), \end{cases} \tag{35}$$

$$f_m = m + \frac{\sum_{i=m+1}^{k}(n-i)}{n-m}, \tag{36}$$

$$k_0 = k - \lfloor \frac{k}{n_I} \rfloor. \tag{37}$$

Fig. 13 provides an example of the optimal trade-off relationship between $\beta_c$ and $\alpha$, explained in Theorem 6. For $\alpha \geq \mathcal{M}/k_0 = 1.104$, the cross-cluster burden $\beta_c$ can be reduced to zero. However, as $\alpha$ decreases from $\mathcal{M}/k_0$, the system requires a larger $\beta_c$ value. For example, if $\alpha = \beta_I = 1.05$ in Fig. 13, $\beta_c \geq 0.03$ is required to satisfy $\mathcal{C}(\alpha, \beta_I, \beta_c) \geq \mathcal{M} = 85$. Thus, for each node failure event, a cross-cluster repair bandwidth of $\gamma_c = (n - n_I)\beta_c \geq 2.7$ is required. Theorem 6 provides an explicit equation for the cross-cluster repair bandwidth we need to pay, in order to reduce node capacity $\alpha$.

## V. Further Comments & Future Works

### A. Explicit Coding Schemes for Clustered DSS

According to the part I proof of Theorem 1, there exists an information flow graph $G^*$ which has the min-cut value of $\mathcal{C}$, the capacity of clustered DSS. Thus, according to [11], there exists a linear network coding scheme which achieves capacity $\mathcal{C}$. Although the existence of a coding scheme is verified, explicit network coding schemes which achieve capacity need to be specified for implementing practical systems. Recently, under the setting of clustered DSS modeled in the present paper, MBR codes for all $n, k, L, \epsilon$ are constructed in [36] and MSR codes for limited parameters are designed in [37]. Explicit code construction for general parameters and/or construction of codes that requires small field size are interesting remaining issues.

### B. Optimal Number of Clusters

According to Theorem 2, capacity $\mathcal{C}$ is asymptotically a monotonically decreasing function of $L$, the number of clusters. Thus, reducing the number of clusters (i.e., gathering storage nodes into a smaller number of clusters) increases storage capacity. However, as mentioned in Section III-E, we typically want to have a sufficiently large $L$, to tolerate the failure of a cluster. Then, the remaining question is in finding optimal $L^*$ which not only allows sufficiently large storage capacity, but also a tolerance to cluster failures. We regard this problem as a future research topic, the solution to which will provide a guidance on the strategy for distributing storage nodes into multiple clusters.

### C. Extension to General $d_I, d_c$ Settings

The present paper assumed a maximum helper node setting, $d_I = n_I - 1$ and $d_c = n - n_I$, since it maximizes the capacity as stated in Proposition 1. However, waiting for all helper nodes gives rise to a latency issue. If we reduce the number of helper nodes $d_I$ and $d_c$, low latency repair would be possible, while the achievable storage capacity decreases. Thus, we consider obtaining the capacity expression for general $d_I, d_c$ settings, and discover the trade-off between capacity and latency for various $d_I, d_c$ values.

### D. Scenarios of Aggregating the Helper Data Within Each Cluster

In recent work [28], [38] on multi-rack (multi-cluster) distributed storage, the authors discuss aggregation of repair data leaving a given cluster. The idea is to allow aggregation and compression of all helper data leaving each cluster to aid reconstruction taking place in some other cluster containing a failed node. This type of repair link aggregation has been shown to reduce the cross-cluster repair burden in [28] and [38]. We expect that the same method would also change the tradeoff picture for our distributed cluster model.

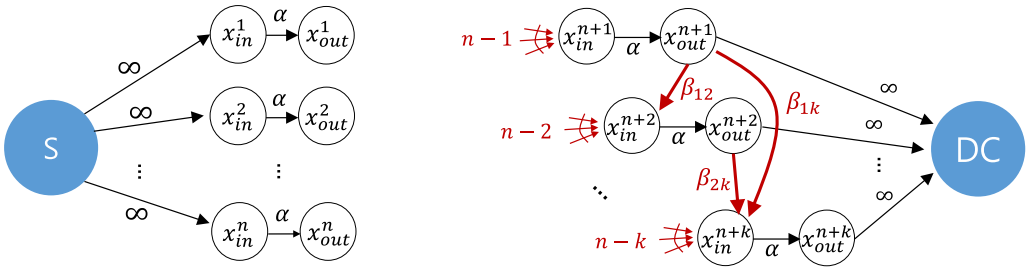


Fig. 14. The information flow graph $G^*$ for the part I proof of Theorem 1.

This is certainly an interesting and important topic to investigate, but careful analysis including the effect of security breach on links will provide a more complete assessment of the merits and potential perils of repair link aggregation. We will leave this as a future endeavor.

## VI. Conclusion

This paper considered a practical distributed storage system where storage nodes are dispersed into several clusters. Noticing that the traffic burdens of intra- and cross-cluster communications are typically different, a new system model for clustered distributed storage systems is suggested. Based on the cut-set bound analysis of information flow graph, the storage capacity $\mathcal{C}(\alpha, \beta_I, \beta_c)$ of the suggested model is obtained in a closed-form, as a function of three main resources: node storage capacity $\alpha$, intra-cluster repair bandwidth $\beta_I$ and cross-cluster repair bandwidth $\beta_c$. It is shown that the asymmetric repair ($\beta_I > \beta_c$) degrades capacity, which is the cost for lifting the cross-cluster repair burden. Moreover, in the asymptotic regime of a large number of storage nodes, capacity is shown to be asymptotically equivalent to a monotonic decreasing function of $L$, the number of clusters. Thus, reducing $L$ (i.e., gathering nodes into less clusters) is beneficial for increasing capacity, although we would typically need to guarantee sufficiently large $L$ to tolerate rack failure events.

Using the capacity expression, we obtained the feasible set of $(\alpha, \beta_I, \beta_c)$ triplet which satisfies $\mathcal{C}(\alpha, \beta_I, \beta_c) \geq \mathcal{M}$, i.e., it is possible to reliably store file $\mathcal{M}$ by using the resource value set $(\alpha, \beta_I, \beta_c)$. The closed-form solution on the feasible set shows a different behavior depending on $\epsilon = \beta_c/\beta_I$, the ratio of cross- to intra-cluster repair bandwidth. It is shown that the minimum storage of $\alpha = \mathcal{M}/k$ is achievable if and only if $\epsilon \geq \frac{1}{n-k}$. Moreover, in the special case of $\epsilon = 0$, we can construct a reliable storage system without using cross-cluster repair bandwidth. A family of network codes which enable $\epsilon = 0$, called the *intra-cluster repairable codes*, has been shown to be a class of the *locally repairable codes* defined in [32].

## Appendix A
## Proof of Theorem 1

Here, we prove Theorem 1. First, denote the right-hand-side (RHS) of (5) as

$$T := \sum_{i=1}^{n_I} \sum_{j=1}^{g_i} \min\{\alpha, \rho_i \beta_I + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m)\beta_c\}. \quad \text{(A.1)}$$

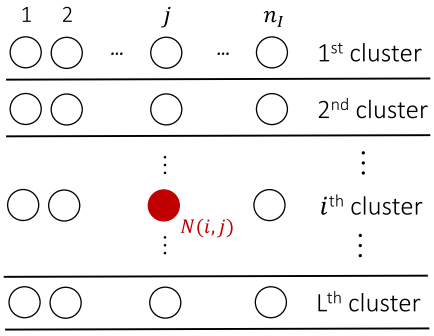


Fig. 15. 2-dim. structure representation.

For other notations used in this proof, refer to subsection II-C. The proof proceeds in two parts.

*Part I:* Show an information flow graph $G^* \in \mathcal{G}$ and a cut-set $c^* \in C(G^*)$ such that $w(G^*, c^*) = T$:

Consider the information flow graph $G^*$ illustrated in Fig. 14, which is obtained by the following procedure. First, data from source node $S$ is distributed into $n$ nodes labeled from $x^1$ to $x^n$. As mentioned in Section II-B, the storage node $x^i = (x_{in}^i, x_{out}^i)$ consists of an input-node $x_{in}^i$ and an output-node $x_{out}^i$. Second, storage node $x^t$ fails and is regenerated at the newcomer node $x^{n+t}$ for $t \in [k]$. The newcomer node $x^{n+t}$ connects to $n-1$ survived nodes $\{x^m\}_{m=t+1}^{n+t-1}$ to regenerate $x^t$. Third, data collector node $DC$ contacts $\{x^{n+t}\}_{t=1}^{k}$ to retrieve data. This whole process is illustrated in the information flow graph $G^*$.

To specify $G^*$, here we determine the 2-dimensional location of the $k$ newcomer nodes $\{x^{n+t}\}_{t=1}^{k}$. First, consider the 2-dimensional structure representation of clustered distributed storage, illustrated in Fig. 15. In this figure, each row represents each cluster, and each node is represented as a 2-dimensional $(i, j)$ point for $i \in [L]$ and $j \in [n_I]$. The symbol $N(i, j)$ denotes the node at $(i, j)$ point. Here we define the set of $n$ nodes,

$$\mathcal{N} := \{N(i, j) : i \in [L], j \in [n_I]\}. \quad \text{(A.2)}$$

For $t \in [k]$, consider selecting the newcomer node $x^{n+t}$ as

$$x^{n+t} = N(i_t, j_t) \quad \text{(A.3)}$$

where

$$i_t = \min\{v \in [n_I] : \sum_{m=1}^{v} g_m \geq t\}, \quad \text{(A.4)}$$

$$j_t = t - \sum_{m=1}^{i_t - 1} g_m, \quad \text{(A.5)}$$

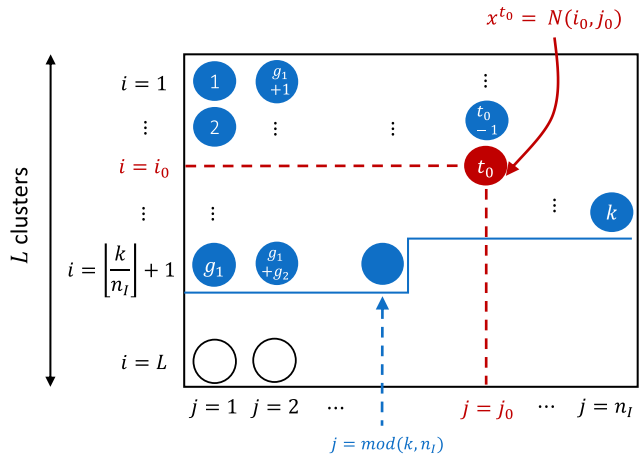


Fig. 16. The location of $k$ newcomer nodes: $x^{n+1}, \cdots, x^{n+k}$.

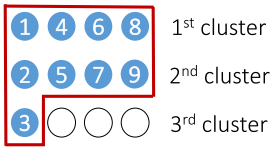


Fig. 17. The location of $k$ newcomer nodes for $n = 12, L = 3, k = 9$ case.

and $g_m$ used in the method is defined in (7). The location of $k$ newcomer nodes selected by this method are illustrated in Fig. 16. Moreover, for the $n = 12, L = 3, k = 9$ case, the newcomer nodes $\{x^{n+t}\}_{t=1}^{k}$ are depicted in Fig. 17. In these figures, the node with number $t$ inside represents the newcomer node labeled as $x^{n+t}$.

For the given graph $G^*$, now we consider a cut-set $c^* \in C(G^*)$ defined as below. The cut-set $c^* = (U, \overline{U})$ can be defined by specifying $U$ and $\overline{U}$ (complement of $U$), which partition the set of vertices in $G^*$. First, let $x_{in}^i, x_{out}^i \in U$ for $i \in [n]$ and $x_{out}^{n+i} \in \overline{U}$ for $i \in [k]$. For $i \in [k]$, the input node $x_{in}^{n+i}$ is included in either $U$ or $\overline{U}$, depending on the condition specified in the next paragraph. Moreover, let $S \in U$ and $DC \in \overline{U}$. See Fig. 18.

Let $U_0 = \bigcup_{i=1}^{n} \{x_{out}^i\}$. For $t \in [k]$, let $\omega_t^*$ be the sum of capacities of edges from $U_0$ to $x_{in}^{n+t}$. If $\alpha \leq \omega_t^*$, then we include $x_{in}^{n+t}$ in $U$. Otherwise, we include $x_{in}^{n+t}$ in $\overline{U}$. Then, the cut-set $c^*$ has the cut-value of

$$w(G^*, c^*) = \sum_{t=1}^{k} \min\{\alpha, \omega_t^*\}. \tag{A.6}$$

All that remains is to show that (A.6) is equal to the expression in (A.1). In other words, we will obtain the expression for $\omega_t^*$.

Recall that in the generation process of $G^*$, any newcomer node $x^{n+t}$ connects to $n-1$ helper nodes $\{x^m\}_{m=t+1}^{n+t-1}$ to regenerate $x^t$. Among the $n-1$ helper nodes, the $n_I - 1$ nodes reside in the same cluster with $x^t$, while the $n - n_I$ nodes are in other clusters. From our system setting in Section III-A, the helper nodes in the same cluster as the failed node help by $\beta_I$, while the helper nodes in other clusters help by $\beta_c$. Therefore, the total repair bandwidth to regenerate any failed

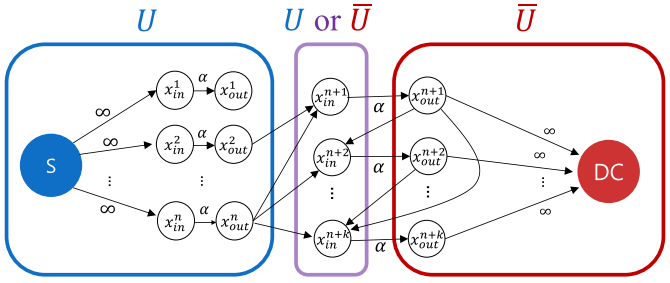


Fig. 18. The cut-set $c^* = (U, \overline{U})$ for the graph $G^*$.

node is

$$\gamma = (n_I - 1)\beta_I + (n - n_I)\beta_c \tag{A.7}$$

as in (2).

The newcomer node $x_{in}^{n+1}$ connects to $\{x_{out}^m\}_{m=2}^{n}$, all of which are included in $U_0$. Therefore, $\omega_1^* = \gamma = (n_I - 1)\beta_I + (n - n_I)\beta_c$ holds. Next, $x_{in}^{n+2}$ connects to $n-2$ nodes $\{x_{out}^m\}_{m=3}^{n}$ from $U_0$ and one node $x_{out}^{n+1}$ from $\overline{U}$. Define variable $\beta_{lm}$ as the repair bandwidth from $x_{out}^{n+l}$ to $x_{in}^{n+m}$. Then, $\omega_2^* = \gamma - \beta_{12}$. From (A.3), we have $x^{n+1} = N(1, 1)$ and $x^{n+2} = N(1, 2)$. Therefore, $x^{n+1}$ and $x^{n+2}$ are in different clusters, which result in $\beta_{12} = \beta_c$. Therefore, $\omega_2^* = \gamma - \beta_{12} = \gamma - \beta_c$.

In general, $x_{in}^{n+t}$ connects to $n - t$ nodes $\{x_{out}^m\}_{m=t+1}^{n}$ from $U_0$, and $t-1$ nodes $\{x_{out}^{n+m}\}_{m=1}^{t-1}$ from $\overline{U}$. Thus, $\omega_t^*$ for $t \in [k]$ can be expressed as $\omega_t^* = \gamma - \sum_{l=1}^{t-1} \beta_{lt}$ where

$$\beta_{lt} = \begin{cases} \beta_I, & \text{if } x^{n+l} \text{ and } x^{n+t} \text{ are in the same cluster} \\ \beta_c, & \text{otherwise.} \end{cases}$$

Recall Fig. 16. For arbitrary newcomer node $x^{n+t} = N(i_t, j_t)$, the set $\{x^{n+m}\}_{m=1}^{t-1}$ contains $i_t - 1$ nodes which reside in the same cluster with $x^{n+t}$, and $t - i_t$ nodes in other clusters. Therefore, $\omega_t^*$ can be expressed as

$$\omega_t^* = \gamma - (i_t - 1)\beta_I - (t - i_t)\beta_c$$

where $i_t$ is defined in (A.4). Combined with (A.7) and (A.5), we get

$$\begin{aligned} \omega_t^* &= (n_I - i_t)\beta_I + (n - n_I - t + i_t)\beta_c \\ &= (n_I - i_t)\beta_I + (n - n_I - j_t - \sum_{m=1}^{i_t - 1} g_m + i_t)\beta_c. \end{aligned}$$

Then, (A.6) can be expressed as

$$\begin{aligned} &w(G^*, c^*) \\ &= \sum_{i=1}^{n_I} \sum_{t \in T_i} \min\{\alpha, (n_I - i)\beta_I + (n - n_I - j_t - \sum_{m=1}^{i-1} g_m + i)\beta_c\} \end{aligned} \tag{A.8}$$

where $T_i = \{t \in [k] : i_t = i\}$. From the definition of $i_t$ in (A.4), we have

$$T_i = \{\sum_{m=1}^{i-1} g_m + 1, \sum_{m=1}^{i-1} g_m + 2, \cdots, \sum_{m=1}^{i} g_m\}.$$

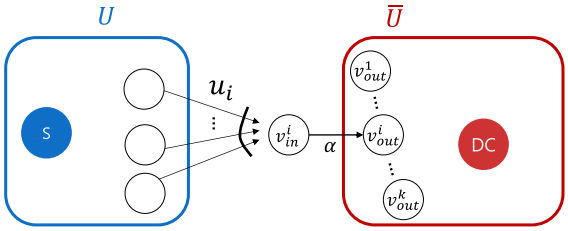


Fig. 19. Arbitrary information flow graph $G \in \mathcal{G}$ and arbitrary cut-set $c \in C(G)$.

Thus, $j_t = 1, 2, \cdots, g_i$ for $t \in T_i$. Therefore, (A.8) can be expressed as

$$w(G^*, c^*) = \sum_{i=1}^{n_I} \sum_{j=1}^{g_i} \min\{\alpha, (n_I - i)\beta_I + (n - n_I - j - \sum_{m=1}^{i-1} g_m + i)\beta_c\},$$

which is identical to $T$ in (A.1), where $\rho_i$ used in this equation is defined in (6). Therefore, the specified information flow graph $G^*$ and the specified cut-set $c^*$ satisfy $\omega(G^*, c^*) = \sum_{t=1}^{k} \min\{\alpha, \omega_t^*\} = T$.

*Part II:* Show that for every information flow graph $G \in \mathcal{G}$ and for every cut-set $c \in C(G)$, the cut-value $w(G, c)$ is greater than or equal to $T$ in (A.1). In other words, $\forall G \in \mathcal{G}, \forall c \in C(G)$, we have $w(G, c) \geq T$.

The proof is divided into 2 sub-parts: Part II-1 and Part II-2.

*Part II-1:* Show that $\forall G \in \mathcal{G}, \forall c \in C(G)$, we have $w(G, c) \geq B(G, c)$ where $B(G, c)$ is in (A.14):

Consider an arbitrary information flow graph $G \in \mathcal{G}$ and an arbitrary cut-set $c \in C(G)$ of the graph $G$. Denote the cut-set as $c = (U, \overline{U})$. Consider an output node $x_{out}^i$ connected to $DC$. If $x_{out}^i \in U$, then the cut-value $w(G, c)$ is infinity, which is a trivial case for proving $w(G, c) \geq B(G, c)$. Therefore, the $k$ output nodes connected to $DC$ are assumed to be in $\overline{U}$. In other words, at least $k$ output nodes exist in $\overline{U}$. Note that every directed acyclic graph can be topologically sorted [34], where vertex $u$ is followed by vertex $v$ if there exists a directed edge from $u$ to $v$. Consider labeling the topologically first $k$ output nodes in $\overline{U}$ as $v_{out}^1, \cdots, v_{out}^k$. Similar to the notation for a storage node $x^i = (x_{in}^i, x_{out}^i)$ in Section II-B, we denote the storage node which contains $v_{out}^i$ as $v^i = (v_{in}^i, v_{out}^i)$. Then, the set of ordered $k-$tuples $(v^i)_{i=1}^k$ can be represented as

$$\mathcal{V}_k = \{(v^1, v^2, \cdots, v^k) : v^t \in \mathcal{N} \text{ for } t \in [k] \quad v^{t_1} \neq v^{t_2} \text{ for } t_1 \neq t_2\}. \quad \text{(A.9)}$$

We also define $u_i$, the sum of capacities of edges from $U$ to $v_{in}^i$. See Fig. 19.

If $v_{in}^1 \in U$, then the cut-set $c = (U, \overline{U})$ should include the edge from $v_{in}^1$ to $v_{out}^1$, which has the edge capacity $\alpha$. Otherwise (i.e., $v_{in}^1 \in \overline{U}$), the cut-set $c = (U, \overline{U})$ should include the edges from $U$ to $v_{in}^1$. If $v_{in}^1$ node is directly connected to the source node $S$, the cut-value $w(G, c)$ is infinity (trivial case for proving $w(G, c) \geq B(G, c)$). Therefore, $v_{in}^1$ node is assumed to be a newcomer node helped by $n-1$ helper nodes. Note that all helper nodes of $v_{in}^1$ are in $U$, since $v_{out}^1$ is the topologically first output node in $\overline{U}$. Thus, the cut-set $c$ should include the edges from $U$ to $v_{in}^1$, where the sum of capacities of these edges are

$$u_1 = \gamma = (n_I - 1)\beta_I + (n - n_I)\beta_c.$$

If $v_{in}^2 \in U$, then the cut-set $c = (U, \overline{U})$ should include the edge from $v_{in}^2$ to $v_{out}^2$, which has the edge capacity $\alpha$. Otherwise (i.e., $v_{in}^2 \in \overline{U}$), the cut-set $c = (U, \overline{U})$ should include the edges from $U$ to $v_{in}^2$. As we discussed in the case of $v_{in}^1$, we can assume $v_{in}^2$ is a newcomer node helped by $n-1$ helper nodes. Since $v_{in}^2$ is the topologically second node in $\overline{U}$, it may have one helper node $v_{out}^1 \in \overline{U}$; at least $n-2$ helper nodes in $U$ help to generate $v_{in}^2$. Note that the total amount of data coming into $v_{in}^2$ is $\gamma = (n_I - 1)\beta_I + (n - n_I)\beta_c$, while the amount of information coming from $v_{out}^1$ to $v_{in}^2$, denoted $\beta_{12}$, is as follows: if $v^1$ and $v^2$ are in the same cluster, $\beta_{12} = \beta_I$, otherwise $\beta_{12} = \beta_c$. Recall that the cut-set should include the edges from $U$ to $v_{in}^2$. The sum of capacities of these edges are

$$u_2 \geq \gamma - \beta_{12},$$

while the equality holds if and only if $v_{out}^1$ helps $v_{in}^2$. In a similar way, for $i \in [k]$, $u_i$ can be bounded as

$$u_i \geq \omega_i \quad \text{(A.10)}$$

where

$$\omega_i = \gamma - \sum_{j=1}^{i-1} \beta_{ji}, \quad \text{(A.11)}$$

$$\beta_{ji} = \begin{cases} \beta_I, & v^j \text{ and } v^i \text{ are in the same cluster} \\ \beta_c, & \text{otherwise.} \end{cases} \quad \text{(A.12)}$$

The equality in (A.10) holds if and only if $v_{out}^j$ helps $v_{in}^i$ for $j \in [i-1]$.

Thus, $v_{out}^i$ contributes at least $\min\{\alpha, \omega_i\}$ to the cut value, for $i \in [k]$. In summary, for arbitrary graph $G \in \mathcal{G}$, an arbitrary cut-set $c$ has cut-value $w(G, c)$ of at least $\sum_{i=1}^{k} \min\{\alpha, \omega_i\}$:

$$w(G, c) \geq \sum_{i=1}^{k} \min\{\alpha, \omega_i\}, \quad \forall G \in \mathcal{G}, \ \forall c \in C(G). \quad \text{(A.13)}$$

Note that $\{\omega_i\}$ depends on the relative position of $\{v_{out}^i\}_{i=1}^k$, which is determined when an arbitrary information flow graph $G \in \mathcal{G}$ and arbitrary cut-set $c \in C(G)$ are specified. This relationship is illustrated in Fig. 20. Therefore, we define

$$B(G, c) = \sum_{i=1}^{k} \min\{\alpha, \omega_i\} \quad \text{(A.14)}$$

for arbitrary $G \in \mathcal{G}$ and arbitrary $c \in C(G)$. Combining (A.13) and (A.14) completes the proof *part II-1:*

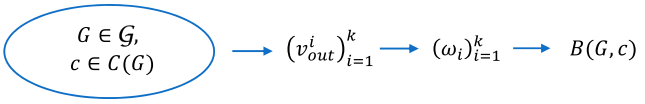


Fig. 20. Dependency graph for variables in the proof of Part II.

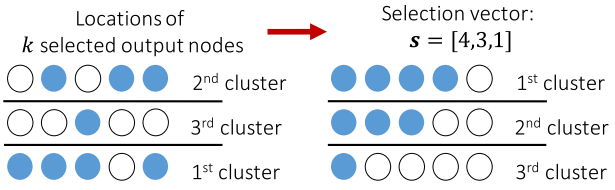


Fig. 21. Obtaining the selection vector for given $k$ output nodes $\{v_{out}^i\}_{i=1}^k$ ($n = 15, k = 8, L = 3$).

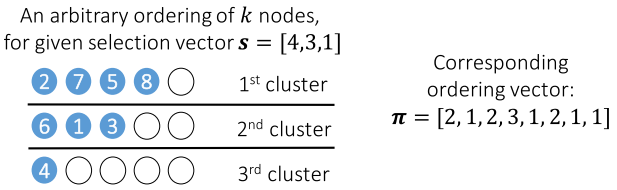


Fig. 22. Obtaining the ordering vector for given an arbitrary order of $k$ output nodes ($n = 15, k = 8, L = 3$).

*Part II-2:* $\min_{G\in\mathcal{G}} \min_{c\in C(G)} B(G,c) = R$:

Assume that $\alpha$ and $k$ are fixed. See Fig. 20. Note that for a given graph $G \in \mathcal{G}$ and a cut-set $c = (U, \overline{U})$ with $c \in C(G)$, the sequence of topologically first $k$ output nodes $(v_{out}^i)_{i=1}^k$ in $\overline{U}$ is determined. Moreover, for a given sequence $(v_{out}^i)_{i=1}^k$, we have a fixed $(\omega_i)_{i=1}^k$, which determines $B(G,c)$ in (A.14). Thus, $\min_{G\in\mathcal{G}} \min_{c\in C(G)} B(G,c)$ can be obtained by finding the optimal $(v_{out}^i)_{i=1}^k$ sequence which minimizes $B(G,c)$. It is identical to finding the optimal $(v^i)_{i=1}^k$, the sequence of $k$ different nodes out of $n$ existing nodes in the system. Therefore, based on (A.14) and (A.11), we have

$$\min_{G\in\mathcal{G}} \min_{c\in C(G)} B(G,c) = \min_{(v^i)_{i=1}^k \in \mathcal{V}_k} \left( \sum_{i=1}^{k} \min\{\alpha, \gamma - \sum_{j=1}^{i-1} \beta_{ji}\} \right) \tag{A.15}$$

where

$$\beta_{ji} = \begin{cases} \beta_I, & v^j \text{ and } v^i \text{ are in the same cluster} \\ \beta_c, & \text{otherwise.} \end{cases} \tag{A.16}$$

holds as defined in (A.12), and $\mathcal{V}_k$ is defined in (A.9). In order to obtain the solution for RHS of (A.15), all we need to do is to find the optimal sequence $(v^i)_{i=1}^k$ of $k$ different nodes, which can be divided into two sub-problems: $i$) finding the optimal way of selecting $k$ nodes $\{v^i\}_{i=1}^k$ out of $n$ nodes, and $ii$) finding the optimal order of selected $k$ nodes. Note that there are $\binom{n}{k}$ selection methods and $k!$ ordering methods. Each selection method can be assigned to a *selection vector* $\boldsymbol{s}$ defined in Definition 1, and each ordering method can be assigned to an *ordering vector* $\boldsymbol{\pi}$ defined in Definition 2.

First, we define a selection vector $\boldsymbol{s}$ for a given $\{v^i\}_{i=1}^k$.

*Definition 1: Assume arbitrary $k$ nodes are selected as $\{v^i\}_{i=1}^k$. Label each cluster by the number of selected nodes in a descending order. In other words, the $1^{st}$ cluster contains a maximum number of selected nodes, and the $L^{th}$ cluster contains a minimum number of selected nodes. Under this setting, define the selection vector $\boldsymbol{s} = [s_1, s_2, \cdots, s_L]$ where $s_i$ is the number of selected nodes in the $i^{th}$ cluster.*

Fig. 21 shows an example of selection vector $\boldsymbol{s}$ corresponding to the selected $k$ nodes $\{v^i\}_{i=1}^k$. From the definition of the selection vector, the set of possible selection vectors can be specified as follows.

$$\mathcal{S} = \Big\{ \boldsymbol{s} = [s_1, \cdots, s_L] : 0 \le s_i \le n_I, s_{i+1} \le s_i, \sum_{i=1}^{L} s_i = k \Big\}.$$

Note that even though $\binom{n}{k}$ different selections exist, the $\{\omega_i\}$ values in (A.11) are only determined by the corresponding selection vector $\boldsymbol{s}$. This is because $\{\omega_i\}$ depends only on the relative positions of $\{v^i\}_{i=1}^k$, whether they are in the same cluster or in different clusters. Therefore, comparing the $\{\omega_i\}$ values of all $|\mathcal{S}|$ possible selection vectors $\boldsymbol{s}$ is enough; it is not necessary to compare the $\{\omega_i\}$ values of $\binom{n}{k}$ selection methods. Now, we define the ordering vector $\boldsymbol{\pi}$ for a given selection vector $\boldsymbol{s}$.

*Definition 2: Let the locations of $k$ nodes $\{v^i\}_{i=1}^k$ be fixed with a corresponding selection vector $\boldsymbol{s} = [s_1, \cdots, s_L]$. Then, for arbitrary ordering of the selected $k$ nodes, define the ordering vector $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_k]$ where $\pi_i$ is the index of the cluster which contains $v^i$.*

For a given $\boldsymbol{s}$, the ordering vector $\boldsymbol{\pi}$ corresponding to an arbitrary ordering of $k$ nodes is illustrated in Fig. 22. In this figure (and the following figures in this paper), the number $i$ written inside each node means that the node is $v^i$. From the definition, an ordering vector $\boldsymbol{\pi}$ has $s_l$ components with value $l$, for all $l \in [L]$. The set of possible ordering vectors can be specified as

$$\Pi(\boldsymbol{s}) = \Big\{ \boldsymbol{\pi} = [\pi_1, \cdots, \pi_k] : \sum_{i=1}^{k} \mathbb{1}_{\pi_i = l} = s_l, \quad \forall l \in [L] \Big\} \tag{A.17}$$

Note that for given $k$ selected nodes, there exists $k!$ different ordering methods. However, the $\{\omega_i\}$ values in (A.11) are only determined by the corresponding ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$, by similar reasoning for compressing $\binom{n}{k}$ selection methods to $|\mathcal{S}|$ selection vectors. Therefore, comparing the $\{\omega_i\}$ values of all possible ordering vectors $\boldsymbol{\pi}$ is enough; it is not necessary to compare the $\{\omega_i\}$ values of all $k!$ ordering methods.

Thus, finding the optimal sequence $(v^i)_{i=1}^k$ is identical to specifying the optimal $(\boldsymbol{s}, \boldsymbol{\pi})$ pair, which is obtained as follows. Recall that from the definition of $\boldsymbol{\pi} = [\pi_1, \pi_2, \cdots, \pi_k]$ in Definition 2, $\pi_i = \pi_j$ holds if and only if $v^i$ and $v^j$ are in the same cluster. Therefore, $\beta_{ji}$ in (A.16) can be expressed by using $\boldsymbol{\pi}$ notation:

$$\beta_{ji}(\boldsymbol{\pi}) = \begin{cases} \beta_I & \text{if } \pi_i = \pi_j \\ \beta_c & \text{otherwise} \end{cases} \tag{A.18}$$

Thus, the $\sum_{j=1}^{i-1} \beta_{ji}$ term in (A.15) can be expressed as

$$\sum_{j=1}^{i-1} \beta_{ji}(\boldsymbol{\pi}) = \Big(\sum_{j=1}^{i-1} \mathbb{1}_{\pi_j = \pi_i}\Big)\beta_I + \Big(i - 1 - \sum_{j=1}^{i-1} \mathbb{1}_{\pi_j = \pi_i}\Big)\beta_c. \tag{A.19}$$

Combining (2), (A.15) and (A.19), we have

$$\min_{G\in\mathcal{G}} \min_{c\in C(G)} B(G,c) = \min_{\boldsymbol{s}\in S} \min_{\boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(\boldsymbol{s},\boldsymbol{\pi})$$

where

$$L(\boldsymbol{s},\boldsymbol{\pi}) = \sum_{i=1}^{k} \min\{\alpha, \omega_i(\boldsymbol{\pi})\}, \tag{A.20}$$

$$\omega_i(\boldsymbol{\pi}) = \gamma - \sum_{j=1}^{i-1} \beta_{ji}(\boldsymbol{\pi}) = a_i(\boldsymbol{\pi})\beta_I + (n-i-a_i(\boldsymbol{\pi}))\beta_c, \tag{A.21}$$

$$a_i(\boldsymbol{\pi}) = n_I - 1 - \sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i}. \tag{A.22}$$

Therefore, the rest of the proof in *part II-2* shows that

$$\min_{\boldsymbol{s}\in S} \min_{\boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(\boldsymbol{s},\boldsymbol{\pi}) = R \tag{A.23}$$

holds. We begin by stating a property of $\omega_i(\boldsymbol{\pi})$ seen in (A.21).

*Proposition 3: Consider a fixed selection vector $\boldsymbol{s}$. We claim that $\sum_{i=1}^{k} \omega_i(\boldsymbol{\pi})$ is constant irrespective of the ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$.*

*Proof:* Let $\boldsymbol{s} = [s_1, \cdots, s_L]$. For an arbitrary ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$, let $b_i(\boldsymbol{\pi}) = n - i - a_i(\boldsymbol{\pi})$ where $a_i(\boldsymbol{\pi})$ is as given in (A.22). For simplicity, we denote $a_i(\boldsymbol{\pi})$, $b_i(\boldsymbol{\pi})$ and $\omega_i(\boldsymbol{\pi})$ as $a_i$, $b_i$ and $\omega_i$, respectively. Then,

$$\sum_{i=1}^{k}(a_i + b_i) = \sum_{i=1}^{k}(n-i) = \textit{constant (const.)} \tag{A.24}$$

for fixed $n, k$. Note that

$$\sum_{i=1}^{k} a_i = k(n_I - 1) - \sum_{i=1}^{k}\sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i} \tag{A.25}$$

from (A.22). Also, from the definition of $\Pi(\boldsymbol{s})$ in (A.17), an arbitrary ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ has $s_l$ components with value $l$, for all $l \in [L]$. If we define

$$I_l(\boldsymbol{\pi}) = \{i \in [k] : \pi_i = l\}, \tag{A.26}$$

then $|I_l(\boldsymbol{\pi})| = s_l$ holds for $l \in [L]$. Then,

$$\sum_{i\in I_l(\boldsymbol{\pi})} \sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i} = 0 + 1 + \cdots + (s_l - 1) = \sum_{t=0}^{s_l-1} t.$$

Therefore,

$$\sum_{i=1}^{k}\sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i} = \sum_{l=1}^{L}\sum_{i\in I_l(\boldsymbol{\pi})}\sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i} = \sum_{l=1}^{L}\sum_{t=0}^{s_l-1} t = const.$$

for fixed $L, \boldsymbol{s}$. Combining with (A.25),

$$\sum_{i=1}^{k} a_i = k(n_I - 1) - \sum_{l=1}^{L}\sum_{t=0}^{s_l-1} t = const. \tag{A.27}$$

for fixed $n, k, L, \boldsymbol{s}$. From (A.24) and (A.27) we get

$$\sum_{i=1}^{k} b_i = const.$$

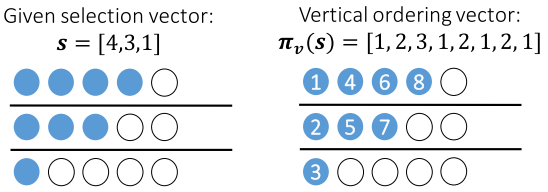


Fig. 23. The vertical ordering vector $\boldsymbol{\pi}_v$ for the given selection vector $\boldsymbol{s} = [4, 3, 1]$ (for $n = 15, k = 8, L = 3$ case).

Therefore, from (A.21),

$$\sum_{i=1}^{k} \omega_i = (\sum_{i=1}^{k} a_i)\beta_I + (\sum_{i=1}^{k} b_i)\beta_c = const.$$

for every ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$, if $n, k, L, \beta_I, \beta_c$ and $\boldsymbol{s}$ are fixed. □

Now, we define a special ordering vector $\boldsymbol{\pi}_v$ called *vertical ordering vector*, which is shown to be the optimal ordering vector $\boldsymbol{\pi}$ which minimizes $L(\boldsymbol{s},\boldsymbol{\pi})$ for an arbitrary selection vector $\boldsymbol{s}$.

*Definition 3: For a given selection vector $\boldsymbol{s} \in \mathcal{S}$, the corresponding vertical ordering vector $\boldsymbol{\pi}_v$ $(\boldsymbol{s})$, or simply denoted as $\boldsymbol{\pi}_v$, is defined as the output of Algorithm 1.*

**Algorithm 1** Generate Vertical Ordering $\boldsymbol{\pi}_v$

**Input:** $\boldsymbol{s} = [s_1, \cdots, s_L]$
**Output:** $\boldsymbol{\pi}_v = [\pi_1, \cdots, \pi_k]$
Initialization: $l \leftarrow 1$
**for** $i = 1$ to $k$ **do**
  **if** $s_l = 0$ **then**
    $l \leftarrow 1$
  **end if**
  $\pi_i \leftarrow l$ (Store the index of cluster)
  $s_{\pi_i} \leftarrow s_{\pi_i} - 1$ (Update the remaining node info.)
  $l \leftarrow (l \bmod L) + 1$ (Go to the next cluster)
**end for**

The vertical ordering vector $\boldsymbol{\pi}_v$ is illustrated in Fig. 23, for a given selection vector $\boldsymbol{s}$ as an example. For $\boldsymbol{s} = [4, 3, 1]$, Algorithm 1 produces the corresponding vertical ordering vector $\boldsymbol{\pi}_v = [1, 2, 3, 1, 2, 1, 2, 1]$. Note that the order of $k = 8$ output nodes is illustrated in Fig. 23, as the numbers inside each node. Although the vertical ordering vector $\boldsymbol{\pi}_v$ depends on the selection vector $\boldsymbol{s}$, we use simplified notation $\boldsymbol{\pi}_v$ instead of $\boldsymbol{\pi}_v(\boldsymbol{s})$. From Fig. 23, obtaining $\boldsymbol{\pi}_v$ using Algorithm 1 can be analyzed as follows. Moving from the leftmost column to the rightmost column, the algorithm selects one node per cluster. After selecting all $k$ nodes, $\pi_i$ stores the index of the cluster which contains the $i^{th}$ selected node. Now, the following Lemma shows that the vertical ordering vector $\boldsymbol{\pi}_v$ is optimal in the sense of minimizing $L(\boldsymbol{s},\boldsymbol{\pi})$ for an arbitrary selection vector $\boldsymbol{s}$.

*Lemma 3: Let $\boldsymbol{s} \in \mathcal{S}$ be an arbitrary selection vector. Then, a vertical ordering vector $\boldsymbol{\pi}_v$ minimizes $L(\boldsymbol{s},\boldsymbol{\pi})$. In other words, $L(\boldsymbol{s},\boldsymbol{\pi}_v) \le L(\boldsymbol{s},\boldsymbol{\pi})$ holds for arbitrary $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$.*

*Proof:* In the case of $\beta_c = 0$, we show that $L(\boldsymbol{s}, \boldsymbol{\pi})$ is constant for every $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$. From (A.20),

$$L(\boldsymbol{s}, \boldsymbol{\pi}) = \sum_{i=1}^{k} \min\{\alpha, a_i(\boldsymbol{\pi})\beta_I\} \tag{A.28}$$

holds for $\beta_c = 0$. Using $I_l(\boldsymbol{\pi})$ in (A.26), note that $[k]$ can be partitioned into $L$ disjoint subsets as $[k] = \bigcup_{l=1}^{L} I_l(\boldsymbol{\pi})$. Therefore, (A.28) can be written as

$$L(\boldsymbol{s}, \boldsymbol{\pi}) = \sum_{l=1}^{L} \sum_{i \in I_l(\boldsymbol{\pi})} \min\{\alpha, a_i(\boldsymbol{\pi})\beta_I\}.$$

Recall that $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ contains $s_l$ components with value $l$ for $l \in [L]$. Thus, from (A.22),

$$\bigcup_{i \in I_l\ (\boldsymbol{\pi})} \{a_i(\boldsymbol{\pi})\} = \{n_I - 1, n_I - 2, \cdots, n_I - s_l\}$$

for $l \in [L]$. Therefore, $\sum_{i \in I_l(\boldsymbol{\pi})} \min\{\alpha, a_i(\boldsymbol{\pi})\beta_I\}$ is constant $\forall \boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ for arbitrary $l \in [L]$. In conclusion, $L(\boldsymbol{s}, \boldsymbol{\pi})$ in (A.28) is constant irrespective of $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ for $\beta_c = 0$.

The rest of the proof deals with the $\beta_c \neq 0$ case. For a given arbitrary $\boldsymbol{s} \in \mathcal{S}$, define two subsets of $\Pi(\boldsymbol{s})$ as

$$\Pi_r = \{\boldsymbol{\pi}^* \in \Pi(\boldsymbol{s}) : \ S_t(\boldsymbol{\pi}) \leq S_t(\boldsymbol{\pi}^*), \quad \forall t \in [k], \ \forall \boldsymbol{\pi} \in \Pi(\boldsymbol{s})\} \tag{A.29}$$

$$\Pi_m = \{\boldsymbol{\pi}^* \in \Pi(\boldsymbol{s}) : L(\boldsymbol{s}, \boldsymbol{\pi}) \geq L(\boldsymbol{s}, \boldsymbol{\pi}^*), \quad \forall \boldsymbol{\pi} \in \Pi(\boldsymbol{s})\} \tag{A.30}$$

where $S_t(\boldsymbol{\pi}) = \sum_{i=1}^{t} w_i\ (\boldsymbol{\pi})$ is the running sum of $w_i(\boldsymbol{\pi})$. Here, we call $\Pi_r$ the *running sum maximizer* and $\Pi_m$ the *min-cut minimizer*. Now the proof proceeds in two steps. The first step proves that the *running sum maximizer* minimizes min-cut, i.e., $\Pi_r \subseteq \Pi_m$. The second step proves that the vertical ordering vector is a running sum maximizer, i.e., $\boldsymbol{\pi}_v \in \Pi_r$.

*Step 1:* Prove $\Pi_r \subseteq \Pi_m$:

Define two index sets for a given ordering vector $\boldsymbol{\pi}$:

$$\begin{aligned} \Omega_L(\boldsymbol{\pi}) &= \{i \in [k] : w_i(\boldsymbol{\pi}) \geq \alpha\} \\ \Omega_s(\boldsymbol{\pi}) &= \{i \in [k] : w_i(\boldsymbol{\pi}) < \alpha\} \end{aligned} \tag{A.31}$$

Now define a set of ordering vectors as

$$\Pi_p = \{\boldsymbol{\pi} \in \Pi(\boldsymbol{s}) : i \leq j \quad \forall i \in \Omega_L(\boldsymbol{\pi}), \ \forall j \in \Omega_s(\boldsymbol{\pi})\}. \tag{A.32}$$

The rest of the proof is divided into 2 sub-steps.

*Step 1-1:* Prove $\Pi_m \subseteq \Pi_p$ and $\Pi_r \subseteq \Pi_p$ by transposition:

Consider arbitrary $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_k] \in \Pi_p^c$. Use a short-hand notation $\omega_i$ to represent $\omega_i(\boldsymbol{\pi})$ for $i \in [k]$. From (A.32), there exists $i > j$ such that $i \in \Omega_L(\boldsymbol{\pi})$ and $j \in \Omega_s(\boldsymbol{\pi})$. Therefore, there exists $t \in [k-1]$ such that $t+1 \in \Omega_L(\boldsymbol{\pi})$ and $t \in \Omega_s(\boldsymbol{\pi})$ hold. By (A.31),

$$\omega_{t+1} \geq \alpha > \omega_t \tag{A.33}$$

for some $t \in [k-1]$. Note that from (A.21), $\pi_t = \pi_{t+1}$ implies $\omega_{t+1} = \omega_t - \beta_I < \omega_t$. Therefore,

$$\pi_t \neq \pi_{t+1} \tag{A.34}$$

should hold to satisfy (A.33). Define an ordering vector $\boldsymbol{\pi'} = [\pi'_1, \cdots, \pi'_k]$ as

$$\begin{cases} \pi'_i = \pi_i & i \neq t, t+1 \\ \pi'_t = \pi_{t+1} \\ \pi'_{t+1} = \pi_t. \end{cases} \tag{A.35}$$

Use a short-hand notation $\omega'_i$ to represent $\omega_i(\boldsymbol{\pi'})$ for $i \in [k]$. Note that $\{\omega'_i\}$ satisfies

$$\begin{cases} \omega'_i = \omega_i & i \neq t, t+1 \\ \omega'_t = \omega_{t+1} + \beta_c \\ \omega'_{t+1} = \omega_t - \beta_c \end{cases} \tag{A.36}$$

for the following reason. First, use simplified notations $a_i$ and $a'_i$ to mean $a_i(\boldsymbol{\pi})$ and $a_i(\boldsymbol{\pi'})$, respectively. Then, using (A.22), (A.34) and (A.35), we have

$$\begin{aligned} a'_t &= n_I - 1 - \sum_{j=1}^{t-1} \mathbb{1}_{\pi'_j = \pi'_t} = n_I - 1 - \sum_{j=1}^{t-1} \mathbb{1}_{\pi_j = \pi_{t+1}} \\ &= n_I - 1 - \sum_{j=1}^{t} \mathbb{1}_{\pi_j = \pi_{t+1}} = a_{t+1}. \end{aligned} \tag{A.37}$$

Similarly,

$$\begin{aligned} a'_{t+1} &= n_I - 1 - \sum_{j=1}^{t} \mathbb{1}_{\pi'_j = \pi'_{t+1}} = n_I - 1 - \sum_{j=1}^{t-1} \mathbb{1}_{\pi'_j = \pi'_{t+1}} \\ &= n_I - 1 - \sum_{j=1}^{t-1} \mathbb{1}_{\pi_j = \pi_t} = a_t. \end{aligned} \tag{A.38}$$

Therefore, from (A.11), (A.37) and (A.38), we have

$$\begin{aligned} \omega'_t &= a'_t \beta_I + (n - t - a'_t)\beta_c \\ &= a_{t+1}\beta_i + (n - t - a_{t+1})\beta_c = \omega_{t+1} + \beta_c. \end{aligned}$$

Similarly, $\omega'_{t+1} = \omega_t - \beta_c$ holds. This proves (A.36). Thus, from (A.33) and (A.36),

$$\begin{aligned} L(\boldsymbol{s}, \boldsymbol{\pi}) &= \sum_{i=1}^{k} \min\{\alpha, \omega_i\} = \sum_{i \notin \{t, t+1\}} \min\{\alpha, \omega_i\} + \omega_t + \alpha, \\ L(\boldsymbol{s}, \boldsymbol{\pi'}) &= \sum_{i=1}^{k} \min\{\alpha, \omega'_i\} \\ &= \sum_{i \notin \{t, t+1\}} \min\{\alpha, \omega'_i\} + \alpha + (\omega_t - \beta_c) \\ &= \sum_{i \notin \{t, t+1\}} \min\{\alpha, \omega_i\} + \alpha + (\omega_t - \beta_c). \end{aligned}$$

Therefore, $L(\boldsymbol{s}, \boldsymbol{\pi}) > L(\boldsymbol{s}, \boldsymbol{\pi'})$ holds for $\beta_c > 0$. In other words, if $\pi \in \Pi_p^c$, then $\pi \in \Pi_m^c$. This proves that $\Pi_p^c \subseteq \Pi_m^c$ holds for $\beta_c \neq 0$.

Similarly, $\Pi_p^c \subseteq \Pi_r^c$ can be proved as follows. For the pre-defined ordering vectors $\boldsymbol{\pi}$ and $\boldsymbol{\pi'}$, we have $S_t(\boldsymbol{\pi}) = \sum_{i=1}^{t-1} \omega_i + \omega_t$ and $S_t(\boldsymbol{\pi'}) = \sum_{i=1}^{t-1} \omega_i + \omega_{t+1} + \beta_c$. Using (A.33), we have $S_t(\boldsymbol{\pi}) < S_t(\boldsymbol{\pi'})$, so that $\boldsymbol{\pi}$ cannot be a running-sum maximizer. Therefore, $\Pi_p^c \subseteq \Pi_r^c$ holds.

*Step 1-2:* Prove that $L(\boldsymbol{s}, \boldsymbol{\pi}^*) \leq L(\boldsymbol{s}, \boldsymbol{\pi})$, $\forall \boldsymbol{\pi}^* \in \Pi_r, \forall \boldsymbol{\pi} \in \Pi_p \cap \Pi_r^c$ :

Consider arbitrary $\boldsymbol{\pi}^* \in \Pi_r$ and $\boldsymbol{\pi} \in \Pi_p \cap \Pi_r^c$. For $i \in [k]$, let $\omega_i^*$ and $\omega_i$ be short-hand notations for $\omega_i(\boldsymbol{\pi}^*)$ and $\omega_i(\boldsymbol{\pi})$, respectively. Note that from Proposition 3,

$$\sum_{i=1}^{k} \omega_i = \sum_{i=1}^{k} \omega_i^*. \tag{A.39}$$

Let

$$\begin{aligned} t &= \max\{i \in [k] : w_i \geq \alpha\}, \\ t^* &= \max\{i \in [k] : w_i^* \geq \alpha\} \end{aligned} \tag{A.40}$$

Then, from (A.29),

$$\sum_{i=1}^{t} \omega_i \leq \sum_{i=1}^{t} \omega_i^*. \tag{A.41}$$

Combining with (A.39), we obtain

$$\sum_{i=t+1}^{k} (\omega_i - \omega_i^*) \geq 0. \tag{A.42}$$

Note that from the result of Step 1-1, both $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^*$ are in $\Pi_p$. Therefore, $\omega_i \geq \alpha$ for $i \in [t]$. Similarly, $\omega_i^* \geq \alpha$ for $i \in [t^*]$. Therefore, (A.20) can be expressed as

$$L(\boldsymbol{s}, \boldsymbol{\pi}) = \sum_{i=1}^{k} \min\{\alpha, \omega_i\} = \sum_{i=1}^{t} \alpha + \sum_{i=t+1}^{k} \omega_i, \tag{A.43}$$

$$L(\boldsymbol{s}, \boldsymbol{\pi}^*) = \sum_{i=1}^{k} \min\{\alpha, \omega_i^*\} = \sum_{i=1}^{t^*} \alpha + \sum_{i=t^*+1}^{k} \omega_i^*. \tag{A.44}$$

If $t = t^*$, then we have

$$L(\boldsymbol{s}, \boldsymbol{\pi}) - L(\boldsymbol{s}, \boldsymbol{\pi}^*) = \sum_{i=t+1}^{k} (\omega_i - \omega_i^*) \geq 0$$

from (A.42). If $t > t^*$, we get

$$L(\boldsymbol{s}, \boldsymbol{\pi}) - L(\boldsymbol{s}, \boldsymbol{\pi}^*) = \sum_{i=t^*+1}^{t} (\alpha - \omega_i^*) + \sum_{i=t+1}^{k} (\omega_i - \omega_i^*).$$

From (A.40), $\omega_i^* < \alpha$ holds for $i > t^*$. Therefore,

$$L(\boldsymbol{s}, \boldsymbol{\pi}) - L(\boldsymbol{s}, \boldsymbol{\pi}^*) > \sum_{i=t+1}^{k} (\omega_i - \omega_i^*) \geq 0$$

from (A.42). In the case of $t < t^*$, define $\Delta = \sum_{i=1}^{t} (\omega_i - \alpha)$ and $\Delta^* = \sum_{i=1}^{t} (\omega_i^* - \alpha)$. From (A.43),

$$L(\boldsymbol{s}, \boldsymbol{\pi}) = \sum_{i=1}^{t} \alpha + \sum_{i=t+1}^{k} \omega_i = (\sum_{i=1}^{k} \omega_i) - \Delta.$$

Similarly, from (A.44),

$$\begin{aligned} L(\boldsymbol{s}, \boldsymbol{\pi}^*) &= \sum_{i=1}^{k} \omega_i^* - \sum_{i=1}^{t^*} (\omega_i^* - \alpha) \\ &= \sum_{i=1}^{k} \omega_i^* - \Delta^* - \sum_{i=t+1}^{t^*} (\omega_i^* - \alpha) \leq \sum_{i=1}^{k} \omega_i^* - \Delta^* \end{aligned}$$

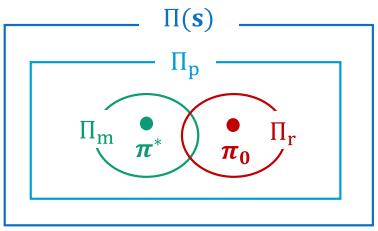


Fig. 24. Relationship between sets.

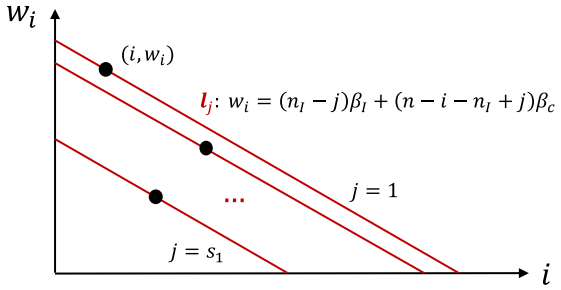


Fig. 25. Set of $s_1$ lines where $(i, \omega_i)$ points can position.

where the last inequality is from (A.40). Combined with (A.39) and (A.41), we obtain

$$L(\boldsymbol{s}, \boldsymbol{\pi}) - L(\boldsymbol{s}, \boldsymbol{\pi}^*) \geq \Delta^* - \Delta \geq 0.$$

In summary, $L(\boldsymbol{s}, \boldsymbol{\pi}^*) \leq L(\boldsymbol{s}, \boldsymbol{\pi})$ irrespective of the $t, t^*$ values, which completes the proof for Step 1-2. From the results of Step 1-1 and Step 1-2, the relationship between the sets can be depicted as in Fig. 24.

Consider $\boldsymbol{\pi}_0 \in \Pi_r$ and $\boldsymbol{\pi}^* \in \Pi_m \cap \Pi_r^c$. Then, $L(\boldsymbol{s}, \boldsymbol{\pi}_0) \leq L(\boldsymbol{s}, \boldsymbol{\pi}^*)$ from the result of Step 1-2. Based on the definition of $\Pi_m$ in (A.30), we can write $L(\boldsymbol{s}, \boldsymbol{\pi_0}) \leq L(\boldsymbol{s}, \boldsymbol{\pi})$ for every $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$. In other words, $\pi_0 \in \Pi_m$ holds for arbitrary $\pi_0 \in \Pi_r$. Therefore, $\Pi_r \subseteq \Pi_m$ holds.

*Step 2:* Prove $\boldsymbol{\pi}_v \in \Pi_r$:

For a given selection vector $\boldsymbol{s} = [s_1, \cdots, s_L]$, consider an arbitrary ordering vector $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_k] \in \Pi(\boldsymbol{s})$. The corresponding $\omega_i(\boldsymbol{\pi})$ defined in (A.21) is written as

$$w_i = (n_I - j)\beta_I + (n - i - n_I + j)\beta_c \tag{A.45}$$

where $j = \sum_{t=1}^{i} \mathbb{1}_{\pi_t = \pi_i}$.

Consider a set of lines $\{l_j\}_{j=1}^{n_I}$, where line $l_j$ represents an equation: $w_i = (n_I - j)\beta_I + (n - i - n_I + j)\beta_c$. Since we assume $\beta_I \geq \beta_c$, these lines can be illustrated as in Fig. 25. For a given $\boldsymbol{\pi}$, consider marking a $(i, \omega_i)$ point for $i \in [k]$. Note that the $(i, \omega_i)$ point is on line $l_j$ if and only if

$$j = \sum_{t=1}^{i} \mathbb{1}_{\pi_t = \pi_i} \tag{A.46}$$

where the summation term in (A.46) represents the number of occurrence of $\pi_i$ value in $\{\pi_t\}_{t=1}^{i}$. For the example in Fig. 23, when $\boldsymbol{s} = [4, 3, 1]$ and $\boldsymbol{\pi} = [1, 2, 3, 1, 2, 1, 2, 1] \in \Pi(\boldsymbol{s})$, line $l_3$ contains the point $(i, \omega_i) = (6, \omega_6)$ since $3 = \sum_{t=1}^{6} \mathbb{1}_{\pi_t = \pi_6}$.

Recall that $I_l(\boldsymbol{\pi}) = \{i \in [k] : \pi_i = l\}$, as defined in (A.26), where $|I_l(\boldsymbol{\pi})| = s_l$ holds $\forall l \in [L]$. For $j \in [n_I]$, consider

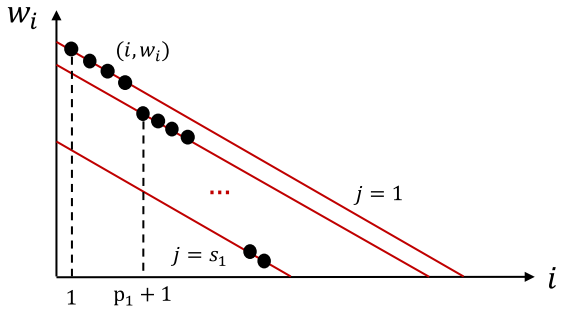


Fig. 26. Optimal packing of $k$ points.

$l \in [L]$ with $s_l \geq j$. Let $i_0$ be the $j^{th}$ smallest element in $I_l(\boldsymbol{\pi})$. Then, $j = \sum_{t=1}^{i_0} \mathbb{1}_{\pi_t = l}$ and $\pi_{i_0} = l$ hold. Thus, the $(i_0, \omega_{i_0})$ point is on line $l_j$. Similarly, we can find

$$p_j = \sum_{l=1}^{L} \mathbb{1}_{s_l \geq j} \tag{A.47}$$

points on line $l_j$, irrespective of the ordering vector $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$. Note that

$$\sum_{j=1}^{n_I} p_j = \sum_{l=1}^{L} \sum_{j=1}^{n_I} \mathbb{1}_{s_l \geq j} = \sum_{l=1}^{L} s_l = k, \tag{A.48}$$

which confirms that Fig. 25 contains $k$ points. Moreover,

$$\forall j \in [n_I - 1], \quad p_j \geq p_{j+1} \tag{A.49}$$

holds from the definition in (A.47).

In order to maximize the running sum $S_t(\boldsymbol{\pi}) = \sum_{i=1}^{t} w_i(\boldsymbol{\pi})$ for every $t$, the optimal ordering vector packs $p_1$ points in the leftmost area ($i = 1, \cdots, p_1$), pack $p_2$ points in the leftmost remaining area ($i = p_1 + 1, \cdots, p_1 + p_2$), and so on. This packing method corresponds to Fig. 26.

Note that from the definition of $p_j$ in (A.47) and Fig. 23, vertical ordering $\boldsymbol{\pi}_v$ in Definition 3 first chooses $p_1$ points on line $l_1$, then chooses $p_2$ points on line $l_2$, and so on. Thus, $\boldsymbol{\pi}_v$ achieves optimal packing in Fig. 26, which maximizes the running sum $S_t(\boldsymbol{\pi})$. Therefore, vertical ordering is a running sum maximizer, i.e., $\boldsymbol{\pi}_v \in \Pi_r$. Combining Steps 1 and 2, we conclude that $\boldsymbol{\pi}_v$ minimizes $L(\boldsymbol{s}, \boldsymbol{\pi})$ among $\boldsymbol{\pi} \in \Pi(\boldsymbol{s})$ for arbitrary $\boldsymbol{s} \in S$. □

Now, we define a special selection vector called the *horizontal selection vector*, which is shown to be the optimal selection vector which minimizes $L(\boldsymbol{s}, \boldsymbol{\pi}_v)$.

*Definition 4: The horizontal selection vector* $\boldsymbol{s}_h = [s_1, \cdots, s_L] \in \mathcal{S}$ *is defined as:*

$$s_i = \begin{cases} n_I, & i \leq \lfloor \frac{k}{n_I} \rfloor \\ (k \bmod n_I), & i = \lfloor \frac{k}{n_I} \rfloor + 1 \\ 0 & i > \lfloor \frac{k}{n_I} \rfloor + 1. \end{cases}$$

The graphical illustration of the horizontal selection vector is on the left side of Fig. 27, in the case of $n = 15$, $k = 8$, $L = 3$. The following Lemma states that the horizontal selection vector minimizes $L(\boldsymbol{s}, \boldsymbol{\pi}_v)$.

*Lemma 4: Consider applying the vertical ordering vector* $\boldsymbol{\pi}_v$. *Then, the horizontal selection vector* $\boldsymbol{s}_h$ *minimizes the lower bound* $L(\boldsymbol{s}, \boldsymbol{\pi})$ *on the min-cut. In other words,* $L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) \leq L(\boldsymbol{s}, \boldsymbol{\pi}_v)$ $\forall \boldsymbol{s} \in \mathcal{S}$.

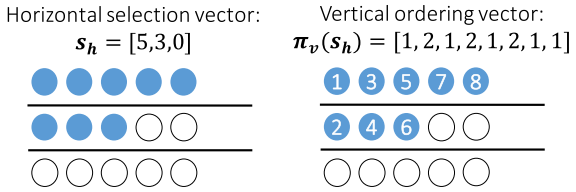


Fig. 27. The optimal selection vector $\boldsymbol{s}_h$ and the optimal ordering vector $\boldsymbol{\pi}_v$ (for $n = 15$, $k = 8$, $L = 3$ case).

*Proof:* From the proof of Lemma 3, the optimal ordering vector turns out to be the vertical ordering vector, where the corresponding $\omega_i(\boldsymbol{\pi}_v)$ sequence is illustrated in Fig. 26. Depending on the selection vector $\boldsymbol{s} = [s_1, \cdots, s_L]$, the number $p_j$ of points on each line $l_j$ changes.

Consider an arbitrary selection vector $\boldsymbol{s}$. Define a point vector $\boldsymbol{p}(\boldsymbol{s}) = [p_1, \cdots, p_{n_I}]$ where $p_j$ is the number of points on $l_j$, as defined in (A.47). Similarly, define $\boldsymbol{p}(\boldsymbol{s}_h) = [p_1^*, \cdots, p_{n_I}^*]$. Using Definition 4 and (A.47), we have

$$p_j^* = \begin{cases} \lfloor \frac{k}{n_I} \rfloor + 1, & j \leq (k \bmod n_I) \\ \lfloor \frac{k}{n_I} \rfloor & otherwise \end{cases} \tag{A.50}$$

Now, we prove

$$\forall t \in [n_I], \quad \sum_{j=1}^{t} p_j^* \leq \sum_{j=1}^{t} p_j. \tag{A.51}$$

The proof is divided into two steps: *base case* and *inductive step*.

*Base Case:* We wish to prove that $p_1^* \leq p_1$. Suppose $p_1^* > p_1$ (i.e., $p_1 \leq p_1^* - 1$). Then,

$$\sum_{l=1}^{n_I} p_l \leq \sum_{l=1}^{n_I} p_1 \leq \sum_{l=1}^{n_I} (p_1^* - 1) \tag{A.52}$$

where the first inequality is from (A.49). Note that if $(k \bmod n_I) = 0$, then

$$\sum_{l=1}^{n_I} p_l^* = \sum_{l=1}^{n_I} p_1^* > \sum_{l=1}^{n_I} (p_1^* - 1). \tag{A.53}$$

Otherwise,

$$\begin{aligned} \sum_{l=1}^{n_I} p_l^* &= \sum_{l=1}^{(k \bmod n_I)} p_l^* + \sum_{l=(k \bmod n_I)+1}^{n_I} p_l^* \\ &= \sum_{l=1}^{(k \bmod n_I)} (\lfloor \frac{k}{n_I} \rfloor + 1) + \sum_{l=(k \bmod n_I)+1}^{n_I} \lfloor \frac{k}{n_I} \rfloor \\ &> \sum_{l=1}^{n_I} \lfloor \frac{k}{n_I} \rfloor = \sum_{l=1}^{n_I} (p_1^* - 1). \end{aligned} \tag{A.54}$$

Therefore, combining (A.52), (A.53) and (A.54) results in $\sum_{l=1}^{n_I} p_l < \sum_{l=1}^{n_I} p_l^* = k$, which contradicts (A.48). Therefore, $p_1^* \leq p_1$ holds.

*Inductive Step:* Assume that $\sum_{l=1}^{l_0} p_l^* \le \sum_{l=1}^{l_0} p_l$ for arbitrary $l_o \in [n_I - 1]$. Now we prove that $\sum_{l=1}^{l_0+1} p_l^* \le \sum_{l=1}^{l_0+1} p_l$ holds. Suppose not. Then,

$$p_{l_0+1}^* - \Theta - 1 \ge p_{l_0+1} \tag{A.55}$$

holds where

$$\Theta = \sum_{l=1}^{l_0} (p_l - p_l^*). \tag{A.56}$$

Using (A.49) and (A.55), we have

$$\begin{aligned}\sum_{l=l_0+1}^{n_I} p_l &\le \sum_{l=l_0+1}^{n_I} p_{l_0+1} \le \sum_{l=l_0+1}^{n_I} (p_{l_0+1}^* - 1 - \Theta) \\ &\le \sum_{l=l_0+1}^{n_I} (p_{l_0+1}^* - 1) - \Theta \end{aligned} \tag{A.57}$$

where equality holds for the last inequality iff $l_0 = n_I - 1$. Using analysis similar to (A.53) and (A.54) for the *base case*, we can find that $\sum_{l=l_0+1}^{n_I} (p_{l_0+1}^* - 1) < \sum_{l=l_0+1}^{n_I} p_l^*$. Combining with (A.57), we get

$$\sum_{l=l_0+1}^{n_I} p_l < \sum_{l=l_0+1}^{n_I} p_l^* - \Theta. \tag{A.58}$$

Equations (A.56) and (A.58) imply $\sum_{l=1}^{n_I} p_l < \sum_{l=1}^{n_I} p_l^* = k$, which contradicts (A.48). Therefore, (A.51) holds.

Now define

$$f_i = \min\{s \in [n_I] : \sum_{l=1}^{s} p_l \ge i\} \tag{A.59}$$

$$h_i = \min\{s \in [n_I] : \sum_{l=1}^{s} p_l^* \ge i\} \tag{A.60}$$

for $i \in [k]$. Then,

$$\forall i \in [k], \quad h_i \ge f_i \tag{A.61}$$

holds directly from (A.51). Note that since $p_i^*$ in (A.50) is identical to $g_i$ in (7), $h_i$ can be written as

$$h_i = \min\{s \in [n_I] : \sum_{l=1}^{s} g_l \ge i\} \tag{A.62}$$

Consider $\boldsymbol{\pi}_v(\boldsymbol{s})$, the vertical ordering vector for a given selection vector $\boldsymbol{s}$. Recall that as in Fig. 26, vertical ordering packs the leftmost $p_1$ points on line $l_1$, packs the next $p_2$ points on line $l_2$, and so on. Using (A.45), we can write

$$\omega_i = \begin{cases} (n_I - 1)\beta_I + (n - i - n_I + 1)\beta_c, & \text{if } i \in [p_1] \\ (n_I - 2)\beta_I + (n - i - n_I + 2)\beta_c, & \text{if } i - p_1 \in [p_2] \\ \quad \vdots & \\ 0 \cdot \beta_I + (n - i)\beta_c, & \text{if } i - \sum_{t=1}^{n_I - 1} p_t \\ & \quad \in [p_{n_I}] \end{cases}$$

Therefore, using (A.59), we further write

$$\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s})) = (n_I - f_i)\beta_I + (n - i - n_I + f_i)\beta_c. \tag{A.63}$$

Similarly,

$$\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h)) = (n_I - h_i)\beta_I + (n - i - n_I + h_i)\beta_c. \tag{A.64}$$

Combining (A.61), (A.63) and (A.64), we have

$$\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h)) \le \omega_i(\boldsymbol{\pi}_v(\boldsymbol{s})) \quad \forall i = 1, \cdots, k, \ \forall \boldsymbol{s} \in \mathcal{S},$$

since we assume $\beta_I \ge \beta_c$. Therefore, combining with (A.20), we conclude that $L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) \le L(\boldsymbol{s}, \boldsymbol{\pi}_v)$ for arbitrary $\boldsymbol{s} \in \mathcal{S}$, which completes the proof of Lemma 4. □

From Lemmas 3 and 4, we have

$$\forall \boldsymbol{s} \in \mathcal{S}, \quad \forall \boldsymbol{\pi} \in \Pi(\boldsymbol{s}), \ L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) \le L(\boldsymbol{s}, \boldsymbol{\pi}).$$

All that remains is to compute $L(\boldsymbol{s}_h, \boldsymbol{\pi}_v)$ and check that it is identical to (A.1).

From (A.20), $L(\boldsymbol{s}_h, \boldsymbol{\pi}_v)$ can be written as

$$L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) = \sum_{i=1}^{k} \min\{\alpha, \omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h))\} \tag{A.65}$$

where $\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h))$ is defined in (A.64). From $h_i$ in (A.62), we have

$$h_i = \begin{cases} 1, & i \in [g_1] \\ 2, & i - g_1 \in [g_2] \\ \quad \vdots & \\ n_I, & i - \sum_{t=1}^{n_I - 1} g_t \in [g_{n_I}] \end{cases} \tag{A.66}$$

If we define

$$I_m^* = \{i \in [k] : h_i = m\},$$

then $L(\boldsymbol{s}_h, \boldsymbol{\pi}_v)$ in (A.65) can be expressed as

$$\begin{aligned} L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) &= \sum_{i=1}^{k} \min\{\alpha, (n_I - h_i)\beta_I + (n - n_I - i + h_i)\beta_c\} \\ &= \sum_{m=1}^{n_I} \sum_{i \in I_m^*} \min\{\alpha, (n_I - m)\beta_I + (n - n_I - i + m)\beta_c\} \\ &= \sum_{m=1}^{n_I} \sum_{i \in I_m^*} \min\{\alpha, \rho_m \beta_I + (n - \rho_m - i)\beta_c\} \end{aligned} \tag{A.67}$$

where $\rho_m$ is defined in (6).

Using (A.66), we have

$$I_m^* = \{\sum_{t=1}^{m-1} g_t + 1, \cdots, \sum_{t=1}^{m-1} g_t + g_m\}.$$

for $m \in [n_I]$. Therefore, $i \in I_m^*$ can be represented as

$$i = \sum_{t=1}^{m-1} g_t + l$$

for $l \in [g_m]$. Using this notation, (A.67) can be written as

$$L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) = \sum_{m=1}^{n_I} \sum_{l=1}^{g_m} \min\{\alpha, \rho_m \beta_I + (n - \rho_m - s_m^{(l)})\beta_c\} \tag{A.68}$$

where $s_m^{(l)} = \sum_{t=1}^{m-1} g_t + l$. This expression reduces to (A.1). This completes the proof of Part II-2. Therefore, the storage capacity of clustered DSS is as stated in Theorem 1.

## APPENDIX B
## PROOF OF THEOREM 3

We begin with introducing properties of the parameters $z_t$ and $h_t$ defined in (25) and (26).

*Proposition 4:*

$$h_k = n_I, \tag{B.1}$$

$$z_k = (n-k)\epsilon. \tag{B.2}$$

*Proof:* Since we consider $k > n_I$ case as stated in (3), we have

$$g_i \geq 1 \quad \forall i \in [n_I]$$

for $\{g_i\}_{i=1}^{n_I}$ defined in (7). Combining with (9) and (26), we can conclude that $h_k = n_I$. Finally, $z_k = (n-k)\epsilon$ is from (25) and (B.1). □

First, consider the $\epsilon \geq \frac{1}{n-k}$ case. From (20), data $\mathcal{M}$ can be reliably stored with node storage $\alpha = \mathcal{M}/k$ if the repair bandwidth satisfies $\gamma \geq \gamma^*$, where

$$\gamma^* = \frac{\mathcal{M} - (k-1)\mathcal{M}/k}{s_{k-1}} = \frac{\mathcal{M}}{k}\frac{1}{s_{k-1}} = \frac{\mathcal{M}}{k}\frac{(n-k)\epsilon}{(n_I-1)+(n-n_I)\epsilon}$$

where the last equality is from (23) and (B.2). Thus, $\alpha = \mathcal{M}/k$ is achievable with finite $\gamma$, when $\epsilon \geq \frac{1}{n-k}$.

Second, we prove that it is impossible to achieve $\alpha = \mathcal{M}/k$ for $0 \leq \epsilon < \frac{1}{n-k}$, in order to reliably store file $\mathcal{M}$. Recall that the minimum storage for $0 \leq \epsilon < \frac{1}{n-k}$ is

$$\alpha = \frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i} \tag{B.3}$$

from (28). From (22) and (B.2), we have $z_i < 1$ for $i = \tau+1, \tau+2, \cdots, k$. Therefore,

$$\tau + \sum_{i=\tau+1}^{k} z_i < \tau + (k-(\tau+1)+1) = k$$

holds, which result in

$$\frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i} > \frac{\mathcal{M}}{k}. \tag{B.4}$$

Thus, the $0 \leq \epsilon \leq \frac{1}{n-k}$ case has the minimum storage $\alpha$ greater than $\mathcal{M}/k$, which completes the proof of Theorem 3.

## APPENDIX C
## PROOF OF THEOREM 4

First, we prove (30). To begin with, we obtain the expression of $\alpha_{msr}^{(\epsilon)}$, for $\epsilon = 0, 1$. From (28), we obtain

$$\alpha_{msr}^{(0)} = \frac{\mathcal{M}}{\tau + \sum_{i=\tau+1}^{k} z_i},$$
$$\alpha_{msr}^{(1)} = \frac{\mathcal{M}}{k}. \tag{C.1}$$

We further simplify the expression for $\alpha_{msr}^{(0)}$ as follows. Recall

$$z_t = n_I - h_t \tag{C.2}$$

for $t \in [k]$ from (25), when $\epsilon = 0$ holds. Note that we have

$$z_t = \begin{cases} 0, & t \geq k - \lfloor \frac{k}{n_I} \rfloor + 1 \\ 1, & t = k - \lfloor \frac{k}{n_I} \rfloor \end{cases} \tag{C.3}$$

from the following reason. First, from (26) and (C.2), $z_t = 0$ holds for

$$t \geq \sum_{l=1}^{n_I-1} g_l + 1 = \sum_{l=1}^{n_I} g_l - g_{n_I} + 1 = k - \lfloor \frac{k}{n_I} \rfloor + 1$$

where the last equality is from (9) and (7). Similarly, we can prove that $z_t = 1$ holds for $t = k - \lfloor \frac{k}{n_I} \rfloor$. From (C.3) and (22), we obtain

$$\tau = k - \lfloor \frac{k}{n_I} \rfloor \tag{C.4}$$

when $\epsilon = 0$. Combining (C.1), (C.3) and (C.4), we have

$$\alpha_{msr}^{(0)} = \frac{\mathcal{M}}{k - \lfloor \frac{k}{n_I} \rfloor}. \tag{C.5}$$

Then, using $R = k/n$ and $\sigma = L^2/n$,

$$\frac{\alpha_{msr}^{(0)}}{\alpha_{msr}^{(1)}} = \frac{k}{k - \lfloor \frac{k}{n_I} \rfloor} = \frac{nR}{nR - \lfloor RL \rfloor} = \frac{nR}{nR - \lfloor R\sqrt{n\sigma} \rfloor} = \frac{R}{R - \lfloor R\sqrt{n\sigma} \rfloor / n}.$$

Thus, for arbitrary fixed $R$ and $\sigma$,

$$\lim_{n\to\infty} \frac{\alpha_{msr}^{(0)}}{\alpha_{msr}^{(1)}} = 1.$$

Therefore, $\alpha_{msr}^{(0)}$ is asymptotically equivalent to $\alpha_{msr}^{(1)}$.

Second, we prove (31). Note that from (29), $\alpha_{mbr}^{(\epsilon)} = \gamma_{mbr}^{(\epsilon)}$ holds for arbitrary $0 \leq \epsilon \leq 1$. Therefore, all we need to prove is

$$\gamma_{mbr}^{(0)} \to \gamma_{mbr}^{(1)}.$$

To begin with, we obtain the expression for $\gamma_{mbr}^{(\epsilon)}$, when $\epsilon = 0, 1$. For $\epsilon = 1$, $z_t$ in (25) is

$$z_t = n - t \tag{C.6}$$

for $t \in [k]$. Moreover, from (23),

$$s_0 = \frac{\sum_{i=1}^{k} z_i}{n-1} \tag{C.7}$$

for $\epsilon = 1$. Therefore, from (29), (C.6) and (C.7),

$$\gamma_{mbr}^{(1)} = \frac{\mathcal{M}}{s_0} = \frac{(n-1)\mathcal{M}}{\sum_{i=1}^{k}(n-i)} = \frac{\mathcal{M}}{k}\frac{2(n-1)}{2n-k-1}. \tag{C.8}$$

Now we focus on the case of $\epsilon = 0$. First, let $q$ and $r$ be

$$q := \lfloor \frac{k}{n_I} \rfloor, \tag{C.9}$$

$$r := (k \bmod n_I), \tag{C.10}$$

which represent the quotient and remainder of $k/n_I$. Note that

$$qn_I + r = k. \tag{C.11}$$

Then, we have

$$\sum_{t=1}^{n_I} tg_t = \sum_{t=1}^{r}(q+1)t + \sum_{t=r+1}^{n_I} qt = q\sum_{t=1}^{n_I} t + \sum_{t=1}^{r} t$$
$$= q\frac{n_I(n_I+1)}{2} + \frac{r(r+1)}{2} = \frac{1}{2}(qn_I^2+r^2+k) \quad \text{(C.12)}$$

where the last equality is from (C.11). From (C.2) and (A.66), we have

$$\sum_{i=1}^{k} z_i = \sum_{t=1}^{n_I}(n_I - t)g_t = n_I k - \frac{1}{2}(qn_I^2 + r^2 + k)$$
$$= (n_I - 1)k - \frac{1}{2}(qn_I^2 + r^2 - k) \quad \text{(C.13)}$$

where the second last equality is from (9) and (C.12). Moreover, using (C.11), we have

$$qn_I^2 + r^2 - k = qn_I^2 + r^2 - qn_I - r$$
$$= (qn_I + r)(n_I - 1) - r(n_I - r)$$
$$\le (qn_I + r)(n_I - 1) = k(n_I - 1) \quad \text{(C.14)}$$

where the equality holds if and only if $r = 0$. Furthermore,

$$s_0 = \frac{\sum_{i=1}^{k} z_i}{n_I - 1} \quad \text{(C.15)}$$

for $\epsilon = 0$ from (23). Combining (29), (C.13), (C.14) and (C.15) result in

$$\gamma_{mbr}^{(0)} = \frac{\mathcal{M}}{s_0} \le \frac{2\mathcal{M}}{k}. \quad \text{(C.16)}$$

From (C.8) and (C.16), we have

$$\gamma_{mbr}^{(0)} - \gamma_{mbr}^{(1)} \le \frac{\mathcal{M}}{k}\left(2 - \frac{2(n-1)}{2n-k-1}\right) = \frac{\mathcal{M}}{k}\frac{2(n-k)}{2n-k-1}$$
$$= \frac{\mathcal{M}}{nR}\frac{2n(1-R)}{n(2-R)-1}$$

where $R = k/n$. Thus, for arbitrary $n$,

$$\gamma_{mbr}^{(0)} \to \gamma_{mbr}^{(1)}$$

as $R \to 1$. This completes the proof of (31). Finally, (C.8) and (C.16) provides

$$\frac{\gamma_{mbr}^{(0)}}{\gamma_{mbr}^{(1)}} \le \frac{2n-k-1}{n-1} = 2 - \frac{k-1}{n-1},$$

which completes the proof of (32).

## Appendix D
## Proof of Theorem 5

Recall the definition of an $(n, l_0, m_0, \mathcal{M}, \alpha)$-LRC which appear right before Theorem 5. Moreover, recall that the *repair locality* of a code is defined as the number of nodes to be contacted in the node repair process [32]. Since each cluster contains $n_I$ nodes, every node in a DSS with $\epsilon = 0$ has the repair locality of

$$l_0 = n_I - 1. \quad \text{(D.1)}$$

Moreover, note that for any code with minimum distance $m$, the original file $\mathcal{M}$ can be retrieved by contacting $n - m + 1$ coded symbols [38]. Since the present paper considers DSSs such that contacting any $k$ nodes can retrieve the original file $\mathcal{M}$, we have the minimum distance of

$$m_0 = n - k + 1. \quad \text{(D.2)}$$

Thus, the intra-cluster repairable code defined in Section IV-C is a $(n, l_0, m_0, \mathcal{M}, \alpha)$-LRC.

Now we show that (34) holds. Note that from Fig. 10, we obtain $\alpha \ge \alpha_{msr}^{(0)}$ for $\epsilon = 0$, where

$$\alpha_{msr}^{(0)} = \frac{\mathcal{M}}{k - \lfloor \frac{k}{n_I} \rfloor} \quad \text{(D.3)}$$

holds according to (C.1). Thus, (34) is proven by showing

$$m_0 \le n - \left\lceil \frac{\mathcal{M}}{\alpha_{msr}^{(0)}} \right\rceil - \left\lceil \frac{\mathcal{M}}{l_0 \alpha_{msr}^{(0)}} \right\rceil + 2. \quad \text{(D.4)}$$

By plugging (D.1), (D.2) and (D.3) into (D.4), we have

$$n - k + 1 \le n - \left\lceil k - \left\lfloor \frac{k}{n_I} \right\rfloor \right\rceil - \left\lceil \frac{k - \lfloor \frac{k}{n_I} \rfloor}{n_I - 1} \right\rceil + 2$$
$$= n - k + \left\lfloor \frac{k}{n_I} \right\rfloor - \left\lceil \frac{k - \lfloor \frac{k}{n_I} \rfloor}{n_I - 1} \right\rceil + 2.$$

Therefore, all we need to prove is

$$0 \le \left\lfloor \frac{k}{n_I} \right\rfloor - \left\lceil \frac{k - \lfloor \frac{k}{n_I} \rfloor}{n_I - 1} \right\rceil + 1, \quad \text{(D.5)}$$

which is proved as follows. Using $q$ and $r$ defined in (C.9) and (C.10), the right-hand-side (RHS) of (D.5) is

$$RHS = q - \left\lceil \frac{qn_I + r - q}{n_I - 1} \right\rceil + 1$$
$$= \begin{cases} q - (q+1) + 1 = 0, & r \ne 0 \\ q - q + 1 = 1, & r = 0 \end{cases}$$

Thus, (D.5) holds, where the equality condition is $r \ne 0$, or equivalently $(k \bmod n_I) \ne 0$. Therefore, (34) holds, where the equality condition is $\alpha = \alpha_{msr}^{(0)}$ and $(k \bmod n_I) \ne 0$. This completes the proof of Theorem 5.

## Appendix E
## Proof of Theorem 6

According to (A.20) and (A.68) in Appendix B, the capacity can be expressed as

$$\mathcal{C} = \sum_{i=1}^{k} \min\{\alpha, \omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h))\}. \quad \text{(E.1)}$$

Using (A.64) and (A.66), $\omega_i(\boldsymbol{\pi}_v\ (\boldsymbol{s}_h))$ in (E.1), or simply $\omega_i$ has the following property:

$$\omega_{i+1} = \begin{cases} \omega_i - \beta_I, & i \in I_G \\ \omega_i - \beta_c, & i \in [k] \setminus I_G \end{cases} \quad \text{(E.2)}$$

where $I_G = \{g_m\}_{m=1}^{n_I - 1}$. Note that $g_{n_I} = \lfloor \frac{k}{n_I} \rfloor$ from (7). Therefore, $k_0$ in (37) can be expressed as

$$k_0 = k - \lfloor \frac{k}{n_I} \rfloor = k - g_{n_I} \le k - 1 \quad \text{(E.3)}$$

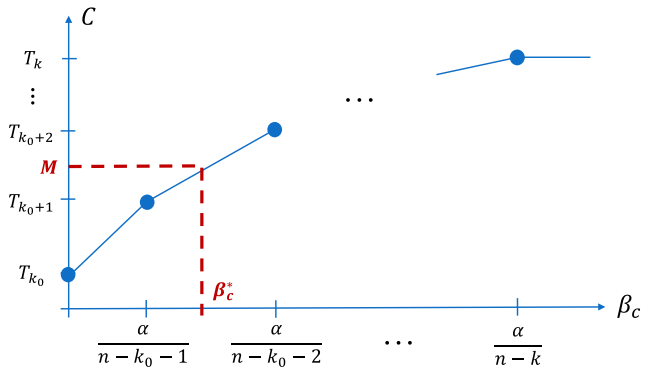


Fig. 28. Capacity as a function of $\beta_c$.

where the last inequality is from (3). Combining (26) and (E.3) result in

$$h_{k_0} = n_I - 1. \tag{E.4}$$

From (A.64), (E.3) and (E.4), we have

$$\begin{aligned}\omega_{k_0} &= (n_I - h_{k_0})\beta_I + (n - k_0 - n_I + h_{k_0})\beta_c \\ &= \beta_I + (n-k_0-1)\beta_c \geq \beta_I + (n-k)\beta_c \geq \beta_I = \alpha. \end{aligned} \tag{E.5}$$

where the last equality holds due to the assumption of $\beta_I = \alpha$ in the setting of Theorem 6. Since $(\omega_i)_{i=1}^k$ is a decreasing sequence from (E.2), the result of (E.5) implies that

$$\omega_i \geq \alpha, \quad 1 \leq i \leq k_0. \tag{E.6}$$

Thus, the capacity expression in (E.1) can be expressed as

$$\mathcal{C} = \begin{cases} k_0\alpha + \sum_{i=k_0+1}^{k} \omega_i, & \omega_{k_0+1} \leq \alpha \\ m\alpha + \sum_{i=m+1}^{k} \omega_i, & \omega_{m+1} \leq \alpha < \omega_m \\ & (k_0 + 1 \leq m \leq k-1) \\ k\alpha, & \alpha < \omega_k \end{cases} \tag{E.7}$$

Note that from (A.64) and (A.66), we have $\omega_i = (n-i)\beta_c$ for $i = k_0+1, k_0+2, \cdots, k$. Therefore, $\mathcal{C}$ in (E.7) is

$$\mathcal{C} = \begin{cases} k_0\alpha + \sum_{i=k_0+1}^{k}(n-i)\beta_c, & 0 \leq \beta_c \leq \frac{\alpha}{n-k_0-1} \\ m\alpha + \sum_{i=m+1}^{k}(n-i)\beta_c, & \frac{\alpha}{n-m} < \beta_c \leq \frac{\alpha}{n-m-1} \\ & (k_0 + 1 \leq m \leq k-1) \\ k\alpha, & \frac{\alpha}{n-k} < \beta_c, \end{cases} \tag{E.8}$$

which is illustrated as a piecewise linear function of $\beta_c$ in Fig. 28. Based on (E.8), the sequence $(T_m)_{m=k_0}^k$ in this figure has the following expression:

$$T_m = \begin{cases} k_0\alpha, & m = k_0 \\ (m + \frac{\sum_{i=m+1}^{k}(n-i)}{n-m})\alpha, & k_0 + 1 \leq m \leq k-1 \\ k\alpha, & m = k \end{cases} \tag{E.9}$$

From Fig. 28, we can conclude that $\mathcal{C} \geq \mathcal{M}$ holds if and only if $\beta_c \geq \beta_c^*$ where

$$\beta_c^* = \begin{cases} 0, & \mathcal{M} \in [0, T_{k_0}] \\ \frac{\mathcal{M}-m\alpha}{\sum_{i=m+1}^{k}(n-i)}, & \mathcal{M} \in (T_m, T_{m+1}] \\ & (m = k_0, k_0+1, \cdots, k-1) \\ \infty, & \mathcal{M} \in (T_k, \infty). \end{cases} \tag{E.10}$$

Using $T_m$ in (E.9) and $f_m$ in (36), (E.10) reduces to (35), which completes the proof.

## Appendix F
## Proofs of Corollaries

### A. Proof of Corollary 1

From the proof of Theorem 1, the capacity expression is equal to (A.67), which is

$$\mathcal{C} = \sum_{i=1}^{k} \min\{\alpha, (n_I - h_i)\beta_I + (n - n_I - i + h_i)\beta_c\}.$$

where $h_i$ is defined in (A.60). Using $\epsilon = \beta_c/\beta_I$ and (2), this can be rewritten as

$$\mathcal{C} = \sum_{i=1}^{k} \min\{\alpha, \frac{(n - n_I - i + h_i)\epsilon + (n_I - h_i)}{(n-n_I)\epsilon + (n_I - 1)}\gamma\}. \tag{F.1}$$

Using $\{z_t\}$ and $\{y_t\}$ defined in (25) and (24), the capacity expression reduces to

$$\mathcal{C}(\alpha, \gamma) = \sum_{t=1}^{k} \min\{\alpha, \frac{\gamma}{y_t}\}, \tag{F.2}$$

which is a continuous function of $\gamma$.

*Remark 1:* $\{z_t\}$ in (25) is a decreasing sequence of $t$. Moreover, $\{y_t\}$ in (24) is an increasing sequence.

*Proof:* Note that from (A.62),

$$h_{t+1} = \begin{cases} h_t + 1, & t \in T \\ h_t, & t \in [k-1] \setminus T \end{cases}$$

where $T = \{g_1, g_1 + g_2, \cdots, \sum_{m=1}^{n_I - 1} g_m\}$. Therefore, $\{z_t\}$ in (25) is a decreasing function of $t$, which implies that $\{y_t\}$ is an increasing sequence. □

Moreover, note that $\beta_I \leq \alpha$ holds from the definition of $\beta_I$ and $\alpha$ in Table I. Thus, combined with $\epsilon = \beta_I/\beta_c$, it is shown that $\gamma$ in (2) is lower-bounded as

$$\begin{aligned}\gamma &= (n_I - 1)\beta_I + (n - n_I)\beta_c \\ &= \{n_I - 1 + (n - n_I)\epsilon\}\beta_I \leq \{n_I - 1 + (n - n_I)\epsilon\}\alpha. \end{aligned}$$

Here, we define

$$\overline{\gamma} = \{n_I - 1 + (n - n_I)\epsilon\}\alpha.$$

Then, the valid region of $\gamma$ is expressed as $\gamma \leq \overline{\gamma}$, as illustrated in Figs. 29 and 30. The rest of the proof depends on the range of $\epsilon$ values; we first consider the $\frac{1}{n-k} \leq \epsilon \leq 1$ case, and then consider the $0 \leq \epsilon < \frac{1}{n-k}$ case.

*1) If $\frac{1}{n-k} \leq \epsilon \leq 1$:* Using (B.2), $z_k = (n-k)\epsilon \geq 1$ holds. Combining with (24), we have $y_k \leq n_I - 1 + \epsilon\ (n - n_I)$, or equivalently, $y_k\alpha \leq \overline{\gamma}$. If $y_t\alpha < \gamma \leq y_{t+1}\alpha$ for some $t \in [k-1]$, then (F.2) can be expressed as

$$\begin{aligned}\mathcal{C}(\alpha, \gamma) &= t\alpha + \sum_{m=t+1}^{k} \frac{\gamma}{y_m} \\ &= t\alpha + \frac{\gamma\ (\sum_{m=t+1}^{k} z_m)}{(n_I - 1) + \epsilon\ (n - n_I)} = t\alpha + s_t\gamma \end{aligned}$$

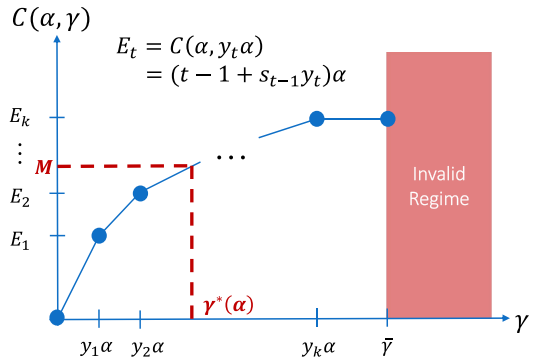


Fig. 29. Capacity as a function of $\gamma$, for $\frac{1}{n-k} \le \epsilon \le 1$.

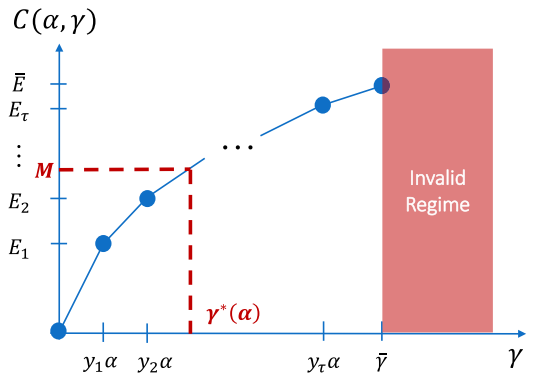


Fig. 30. Capacity as a function of $\gamma$, for $0 \le \epsilon < \frac{1}{n-k}$ case.

where $\{s_t\}$ is defined in (23). If $0 \le \gamma \le y_1\alpha$, then

$$\mathcal{C}(\alpha,\gamma) = \sum_{m=1}^{k} \frac{\gamma}{y_m} = \frac{\gamma\ (\sum_{m=1}^{k} z_m)}{(n_I - 1) + \epsilon\ (n - n_I)} = s_0\gamma.$$

If $y_k\alpha < \gamma \le \bar{\gamma}$, then $\mathcal{C}(\alpha,\gamma) = \sum_{m=1}^{k} \alpha = k\alpha$. In summary, capacity is

$$\mathcal{C}(\alpha,\gamma) = \begin{cases} s_0\gamma, & 0 \le \gamma \le y_1\alpha \\ t\alpha + s_t\gamma, & y_t\alpha < \gamma \le y_{t+1}\alpha \\ & (t = 1, 2, \cdots, k-1) \\ k\alpha, & y_k\alpha < \gamma \le \bar{\gamma} \end{cases} \tag{F.3}$$

which is illustrated in Fig. 29. Since $\{z_t\}$ is a decreasing sequence from Remark 1, we have $z_t \ge z_k = (n-k)\epsilon > 0$ for $t \in [k]$. Thus, $\{s_t\}_{t=1}^{k}$ defined in (23) is a monotonically decreasing, non-negative sequence. This implies that the curve in Fig. 29 is a monotonic increasing function of $\gamma$.

From Fig. 29, it is shown that $\mathcal{C}(\alpha,\gamma) \ge \mathcal{M}$ holds if and only if $\gamma \ge \gamma^*(\alpha)$. From (F.3), the threshold value $\gamma^*(\alpha)$ can be expressed as

$$\gamma^*(\alpha) = \begin{cases} \frac{\mathcal{M}}{s_0}, & \mathcal{M} \in [0, E_1] \\ \frac{\mathcal{M} - t\alpha}{s_t}, & \mathcal{M} \in (E_t, E_{t+1}] \\ & (t = 1, 2, \cdots, k-1) \\ \infty, & \mathcal{M} \in (E_k, \infty). \end{cases} \tag{F.4}$$

where

$$E_t = \mathcal{C}(\alpha, y_t\alpha) = (t - 1 + s_{t-1}y_t)\alpha \tag{F.5}$$

for $t \in [k]$. The threshold value $\gamma^*(\alpha)$ in (F.4) can be expressed as (20), which completes the proof.

*2) Otherwise (if $0 \le \epsilon < \frac{I}{n-k}$):* Using (B.2),

$$z_k = (n-k)\epsilon < 1 \tag{F.6}$$

holds. Since $\{z_t\}$ is a decreasing sequence from Remark 1, there exists $\tau \in \{0, 1, \cdots, k-1\}$ such that $z_{\tau+1} < 1 \le z_\tau$ holds, or equivalently, $y_\tau\alpha \le \bar{\gamma} < y_{\tau+1}\alpha$.

Using the analysis similar to the $\frac{1}{n-k} \le \epsilon \le 1$ case, we obtain

$$\mathcal{C}(\alpha,\gamma) = \begin{cases} s_0\gamma, & 0 \le \gamma \le y_1\alpha \\ t\alpha + s_t\gamma, & y_t\alpha < \gamma \le y_{t+1}\alpha \\ & (t = 1, 2, \cdots, \tau-1) \\ \tau\alpha + s_\tau\gamma, & y_\tau\alpha < \gamma \le \bar{\gamma} \end{cases} \tag{F.7}$$

which is illustrated in Fig. 30.

From Fig. 30, it is shown that $\mathcal{C}(\alpha,\gamma) \ge \mathcal{M}$ holds if and only if $\gamma \ge \gamma^*(\alpha)$. From (F.7), the threshold value $\gamma^*(\alpha)$ can be expressed as

$$\gamma^*(\alpha) = \begin{cases} \frac{\mathcal{M}}{s_0}, & \mathcal{M} \in [0, E_1] \\ \frac{\mathcal{M} - t\alpha}{s_t}, & \mathcal{M} \in (E_t, E_{t+1}] \\ & (t = 1, 2, \cdots, \tau-1) \\ \frac{\mathcal{M} - \tau\alpha}{s_\tau}, & \mathcal{M} \in (E_\tau, \bar{E}] \\ \infty, & \mathcal{M} \in (\bar{E}, \infty). \end{cases} \tag{F.8}$$

where $\{E_t\}$ is defined in (F.5), and

$$\bar{E} = \mathcal{C}(\alpha, \bar{\gamma}) = \tau\alpha + s_\tau\alpha\{n_I - 1 + (n - n_I)\epsilon\}.$$

The threshold value $\gamma^*(\alpha)$ in (F.8) can be expressed as (21), which completes the proof.

## *B. Proof of Corollary 2*

First, we focus on the MSR point illustrated in Fig. 9. From (20), the MSR point for $\frac{1}{n-k} \le \epsilon \le 1$ is

$$\begin{aligned}(\alpha_{msr}^{(\epsilon)}, \gamma_{msr}^{(\epsilon)}) &= (\frac{\mathcal{M}}{k + s_k y_k}, \frac{\mathcal{M} - (k-1)\alpha_{msr}^{(\epsilon)}}{s_{k-1}}) \\ &= (\frac{\mathcal{M}}{k}, \frac{\mathcal{M}}{k}\frac{1}{s_{k-1}})\end{aligned} \tag{F.9}$$

where the last equality is from $s_k = 0$ in (23). Moreover, from (21), the MSR point for $0 \le \epsilon < \frac{1}{n-k}$ is

$$\begin{aligned}(\alpha_{msr}^{(\epsilon)}, \gamma_{msr}^{(\epsilon)}) &= (\frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i}, \frac{\mathcal{M} - \tau\alpha_{msr}^{(\epsilon)}}{s_\tau}) \\ &= (\frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i}, \frac{M}{\tau + \sum_{i=\tau+1}^{k} z_i}\frac{\sum_{i=\tau+1}^{k} z_i}{s_\tau}).\end{aligned} \tag{F.10}$$

Equations (F.9) and (F.10) proves (28). The expression (29) for the MBR point is directly obtained from Corollary 1 and Fig. 9.

## APPENDIX G
## PROOFS OF PROPOSITIONS

### A. Proof of Proposition 1

As in (A.23), capacity $\mathcal{C}$ for maximum $d_I, d_c$ setting $(d_I = n_I - 1, d_c = n - n_I)$ is expressed as

$$\mathcal{C} = \min_{\boldsymbol{s}\in S, \boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(\boldsymbol{s}, \boldsymbol{\pi}) \tag{G.1}$$

where

$$\begin{aligned} L(\boldsymbol{s}, \boldsymbol{\pi}) &= \sum_{i=1}^{k} \min\{\alpha, \omega_i(\boldsymbol{\pi})\}, \\ \omega_i(\boldsymbol{\pi}) &= \gamma - e_i(\boldsymbol{\pi})\beta_I - (i - 1 - e_i(\boldsymbol{\pi}))\beta_c, \\ e_i(\boldsymbol{\pi}) &= \sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i}, \end{aligned} \tag{G.2}$$

as in (A.20), (A.21) and (A.19).

Consider a general $d_I, d_c$ setting, where each newcomer node is helped by $d_I$ nodes in the same cluster, receiving $\beta_I$ information from each node, and $d_c$ nodes in other clusters, receiving $\beta_c$ information from each node. Under this setting, the coefficient of $\beta_I$ in (G.2) cannot exceed $d_I$. Similarly, the coefficient of $\beta_c$ in (G.2) cannot exceed $d_c$. Thus, the capacity for general $d_I, d_c$ is expressed as

$$\mathcal{C}(d_I, d_c) = \min_{\boldsymbol{s}\in S, \boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) \tag{G.3}$$

where

$$\begin{aligned} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) &= \sum_{i=1}^{k} \min\{\alpha, \omega_i(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi})\}, \\ \omega_i(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) &= \gamma - \min\{d_I, e_i(\boldsymbol{\pi})\}\beta_I \\ &\quad - \min\{d_c, i - 1 - e_i(\boldsymbol{\pi})\}\beta_c, \\ e_i(\boldsymbol{\pi}) &= \sum_{j=1}^{i-1} \mathbb{1}_{\pi_j=\pi_i}. \end{aligned} \tag{G.4}$$

Consider arbitrary fixed $\boldsymbol{s}, \boldsymbol{\pi}$ and $d_c$. Since $\gamma$ and $\gamma_c = d_c\beta_c$ are fixed in the basic setting of Proposition 1, only $d_I$ and $\beta_I$ are variables in (G.4), while other parameters are constants. Then, (G.4) can be expressed as

$$\begin{aligned} \omega_i(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) &= C_1 - \min\{d_I, e_i(\boldsymbol{\pi})\}\beta_I \\ &= C_1 - \frac{\min\{d_I, e_i(\boldsymbol{\pi})\}}{d_I} C_2 \end{aligned} \tag{G.5}$$

where $C_1 = \gamma - \min\{d_c, i-1-e_i(\boldsymbol{\pi})\}\beta_c$ and $C_2 = \gamma_I = \gamma - \gamma_c$ are constants. Note that

$$\frac{\min\{d_I, e_i(\boldsymbol{\pi})\}}{d_I} = \begin{cases} 1, & \text{if } d_I \le e_i(\boldsymbol{\pi}), \\ \frac{e_i(\boldsymbol{\pi})}{d_I}, & \text{otherwise} \end{cases} \tag{G.6}$$

is a non-increasing function of $d_I$. Thus, $\omega_i(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi})$ in (G.5) is a non-decreasing function of $d_I$ for arbitrary fixed $\boldsymbol{s}, \boldsymbol{\pi}, d_c$ and $i \in [k]$. Since the maximum $d_I$ value is $n_I - 1$, we have

$$\begin{aligned} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) &= \sum_{i=1}^{k} \min\{\alpha, \omega_i(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi})\} \\ &\le \sum_{i=1}^{k} \min\{\alpha, \omega_i(n_I - 1, d_c, \boldsymbol{s}, \boldsymbol{\pi})\} \\ &= L(n_I - 1, d_c, \boldsymbol{s}, \boldsymbol{\pi}) \end{aligned} \tag{G.7}$$

for $d_I \in [n_I - 1]$. In other words, for all $\boldsymbol{s}, \boldsymbol{\pi}, d_c$, we have

$$\arg\max_{d_I\in[n_I-1]} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) = n_I - 1.$$

Similarly, for all $\boldsymbol{s}, \boldsymbol{\pi}, d_I$,

$$\arg\max_{d_c\in[n-n_I]} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) = n - n_I$$

holds. Therefore, for all $\boldsymbol{s}, \boldsymbol{\pi}$,

$$\arg\max_{[d_I, d_c]} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) = [n_I - 1, n - n_I]. \tag{G.8}$$

Let

$$[\boldsymbol{s}^*, \boldsymbol{\pi}^*] = \arg\min_{\boldsymbol{s}\in S, \boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(n_I - 1, n - n_I, \boldsymbol{s}, \boldsymbol{\pi}). \tag{G.9}$$

Then, from (G.3), (G.8) and (G.9),

$$\begin{aligned} \mathcal{C}(d_I, d_c) &= \min_{\boldsymbol{s}\in S, \boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(d_I, d_c, \boldsymbol{s}, \boldsymbol{\pi}) \\ &\le L(d_I, d_c, \boldsymbol{s}^*, \boldsymbol{\pi}^*) \le L(n_I - 1, n - n_I, \boldsymbol{s}^*, \boldsymbol{\pi}^*) \\ &= \min_{\boldsymbol{s}\in S, \boldsymbol{\pi}\in\Pi(\boldsymbol{s})} L(n_I - 1, n - n_I, \boldsymbol{s}, \boldsymbol{\pi}) \\ &= \mathcal{C}(n_I - 1, n - n_I) \end{aligned} \tag{G.10}$$

for all $d_I \in [n_I - 1]$ and $d_c \in [n - n_I]$. Therefore, choosing $d_I = n_I - 1$ and $d_c = n - n_I$ maximizes storage capacity when the available resources, $\gamma$ and $\gamma_c$, are given.

### B. Proof of Proposition 2

First, we prove (8). Recall $\rho_i$ and $g_m$ defined in (6) and (7). Consider the *support set* $S$, which is defined as

$$S = \{i \in [n_I] : g_i \ge 1\}. \tag{G.11}$$

Then, we have

$$\rho_i = n_I - i \le n_I - 1, \tag{G.12}$$

$$\sum_{m=1}^{i-1} g_m \ge i - 1 \tag{G.13}$$

for every $i \in S$. Therefore, by combining (G.13) and (6),

$$\begin{aligned} n - \rho_i - j - \sum_{m=1}^{i-1} g_m &\le n - (i - 1) - j - \rho_i \\ &= n - n_I - (j - 1) \le n - n_I \end{aligned} \tag{G.14}$$

holds for every $i \in S, j \in [g_i]$. Combining (2), (G.12) and (G.14) results in

$$\rho_i\beta_I + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m)\beta_c \le (n_I - 1)\beta_I + (n - n_I)\beta_c = \gamma \tag{G.15}$$

for arbitrary $i \in S, j \in [g_i]$. Since $[g_i] = \emptyset$ holds for $i \in [n_I] \setminus S$, we conclude that (G.15) holds for $(i, j)$ with $i \in [n_I]$, $j \in [g_i]$.

Second, we prove (9). Using $q$ and $r$ in (C.9) and (C.10),

$$g_i = \begin{cases} q+1, & i \le r \\ q, & \text{otherwise} \end{cases} \tag{G.16}$$

Therefore, $\sum_{i=1}^{n_I} g_i = (q+1)r + q(n_I - r) = r + qn_I = k$, where the last equality is from (C.11).

## Appendix H
## Proofs of Lemmas

### A. Proof of Lemma 1

Using (2), $\underline{C}$ in (14) can be expressed as

$$\begin{aligned} \underline{C} &= \frac{k}{2}\left((n_I-1)\beta_I + (n-n_I)(1+\frac{n-k}{n(1-1/L)})\beta_c\right) \\ &= \frac{k}{2}\left((n_I-1)\beta_I + (n-n_I)(1+\frac{n-k}{n-n_I})\beta_c\right) \\ &= \frac{k}{2}\left\{(n_I-1)\beta_I + (2n-n_I-k)\beta_c\right\}. \end{aligned} \tag{H.1}$$

According to (A.20) and (A.68) in Appendix B, the capacity can be expressed as

$$\mathcal{C} = L(\boldsymbol{s}_h, \boldsymbol{\pi}_v) = \sum_{i=1}^{k} \min\{\alpha, \omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h))\}. \tag{H.2}$$

From (A.21), we have

$$\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h)) \le \gamma \tag{H.3}$$

for $i \in [k]$. Therefore, when $\alpha = \gamma$, the capacity expression in (H.2) reduces to

$$\mathcal{C} = \sum_{i=1}^{k} \omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h)). \tag{H.4}$$

Recall (A.64):

$$\omega_i(\boldsymbol{\pi}_v(\boldsymbol{s}_h)) = (n_I - h_i)\beta_I + (n - i - n_I + h_i)\beta_c. \tag{H.5}$$

From the expression of $(h_i)_{i=1}^k$ in (A.66), we have

$$\sum_{i=1}^{k}(n_I - h_i) = \sum_{s=1}^{n_I} g_s(n_I - s) \tag{H.6}$$

since $\sum_{m=1}^{k} g_m = k$ from (9),

Using (H.5) and (H.6), the capacity expression in (H.4) is expressed as

$$\begin{aligned} \mathcal{C} &= \left(\sum_{i=1}^{k}(n_I - h_i)\right)\beta_I + \left(\sum_{i=1}^{k}(n - i - (n_I - h_i))\right)\beta_c \\ &= A_0\beta_I + B_0\beta_c \end{aligned} \tag{H.7}$$

where $A_0 = \sum_{s=1}^{n_I} g_s\ (n_I - s)$ and

$$B_0 = \sum_{i=1}^{k}(n-i) - A_0. \tag{H.8}$$

Similarly, $\underline{C}$ in (H.1) can be expressed as

$$\underline{C} = A_0'\beta_I + B_0'\beta_c \tag{H.9}$$

where $A_0' = (n_I - 1)\frac{k}{2}$ and

$$B_0' = (2n - n_I - k)\frac{k}{2} = \sum_{i=1}^{k}(n-i) - A_0'. \tag{H.10}$$

First, we show that $\mathcal{C} \ge \underline{C}$ holds. From (H.7), (H.8), (H.9) and (H.10),

$$\begin{aligned} \mathcal{C} - \underline{C} &= (A_0 - A_0')\beta_I + (B_0 - B_0')\beta_c \\ &= (A_0 - A_0')\beta_I - (A_0 - A_0')\beta_c = (A_0 - A_0')(\beta_I - \beta_c). \end{aligned} \tag{H.11}$$

Since we consider the $\beta_I \ge \beta_c$ case, all we need to prove is

$$A_0 - A_0' \ge 0.$$

Using $(g_i)_{i=1}^k$ expression in (G.16), $A_0$ can be rewritten as

$$\begin{aligned} A_0 &= \sum_{s=1}^{n_I} g_s(n_I - s) = q\sum_{s=1}^{n_I}(n_I - s) + \sum_{s=1}^{r}(n_I - s) \\ &= q\frac{n_I(n_I-1)}{2} + \sum_{s=1}^{r}(n_I - s) \\ &= q\frac{n_I(n_I-1)}{2} + \frac{r((n_I-1)+(n_I-r))}{2} \\ &= (qn_I + r)\frac{n_I - 1}{2} + \frac{r(n_I - r)}{2} \\ &= \frac{k(n_I-1)}{2} + \frac{r(n_I-r)}{2} = A_0' + \frac{r(n_I-r)}{2}. \end{aligned}$$

Since $0 \le r < n_I$, we have

$$A_0 - A_0' = \frac{r(n_I - r)}{2} \ge 0. \tag{H.12}$$

Therefore, $\mathcal{C} \ge \underline{C}$ holds.

Second, we prove $\mathcal{C} \le \underline{C} + n_I^2(\beta_I - \beta_c)/8$. Note that $A_0 - A_0'$ in (H.12) is maximized when $r = \lfloor n_I/2 \rfloor$ holds. Thus, $A_0 - A_0' \le \frac{\lfloor n_I/2 \rfloor\ (n_I - \lfloor n_I/2 \rfloor)}{2} \le n_I^2/8$. Combining with (H.11),

$$\mathcal{C} - \underline{C} = (A_0 - A_0')(\beta_I - \beta_c) \le n_I^2(\beta_I - \beta_c)/8.$$

### B. Proof of Lemma 2

Recall that $\xi = \gamma_c/\gamma$, $\gamma$ and $R = k/n$ value are all fixed. The expression for $\underline{C}$ in (14) can be expressed as

$$\begin{aligned} \underline{C} &= \frac{k}{2}(\gamma + \frac{n-k}{n(1-\frac{1}{L})}\gamma_c) = \frac{nR}{2}(\gamma + \frac{1-R}{1-\frac{1}{L}}\gamma_c) \\ &= \gamma\frac{nR}{2}(1 + \frac{1-R}{(1-\frac{1}{L})}\xi) \end{aligned} \tag{H.13}$$

Note that (H.13) is a monotonic decreasing function of $L$. Moreover, we consider the $L \ge 2$ case, as mentioned in (4). Thus, $\underline{C}$ is upper/lower bounded by expressions for $L = 2$ and $L = \infty$, respectively:

$$\gamma\frac{nR}{2}(1 + (1-R)\xi) < \underline{C} \le \gamma\frac{nR}{2}(1 + 2(1-R)\xi). \tag{H.14}$$

Therefore, $\underline{C} = \Theta(n)$ holds. Moreover, the expression for $\delta$ in (17) is

$$\begin{aligned}\delta &= \frac{n_I^2(\beta_I - \beta_c)}{8} = \frac{n_I^2}{8}\left(\frac{\gamma_I}{n_I - 1} - \frac{\gamma_c}{n - n_I}\right) \\ &= \frac{n_I^2}{8}\left(\frac{\gamma(1-\xi)}{n_I - 1} - \frac{\gamma\xi}{n - n_I}\right) \\ &= \frac{n_I^2\gamma}{8}\frac{(1-\xi)(n-n_I) - \xi(n_I - 1)}{(n_I - 1)(n - n_I)}. \end{aligned} \tag{H.15}$$

Putting $n = n_I L$ into (H.15), we get

$$\delta = \frac{n_I\gamma}{8}\frac{(1-\xi)n_I(L-1) - \xi(n_I - 1)}{(n_I - 1)(L-1)} = O(n_I).$$

## REFERENCES


[1] J.-Y. Sohn, B. Choi, S. W. Yoon, and J. Moon, "Capacity of clustered distributed storage," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.

[2] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[3] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. NSDI*, vol. 4, 2004, p. 25.

[4] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *Proc. NSDI*, vol. 4, 2004, pp. 85–98.

[5] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: The oceanstore prototype," in *Proc. FAST*, vol. 3, 2003, pp. 1–14.

[6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10.

[7] C. Huang *et al.*, "Erasure coding in windows azure storage," in *Proc. Annu. Techn. Conf. (USENIX)*, 2012, pp. 15–26.

[8] S. Muralidhar *et al.*, "f4: Facebook's warm BLOB storage system," in *Proc. 11th USENIX Conf. Oper. Syst. Des. Implement. (OSDI)*, 2014, pp. 383–398.

[9] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling, "HDFS RAID," in *Proc. Hadoop User Group Meeting*, 2010.

[10] 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.

[11] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[12] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. 47th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2009, pp. 1243–1249.

[13] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[14] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2134–2158, Apr. 2012.

[15] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. OSDI*, 2010, pp. 1–7.

[16] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in *Proc. HotStorage*, 2013.

[17] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "Shufflewatcher: Shuffle-aware scheduling in multi-tenant mapreduce clusters," in *Proc. Annu. Techn. Conf. (USENIX ATC)*, 2014, pp. 1–12.

[18] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[19] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-out networking in the data center," *IEEE Micro*, vol. 30, no. 4, pp. 29–41, Jul./Aug. 2010.

[20] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. V. Poor, "Capacity and security of heterogeneous distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2701–2709, Dec. 2013.

[21] Q. Yu, K. W. Shum, and C. W. Sung, "Tradeoff between storage cost and repair cost in heterogeneous distributed storage systems," *Trans. Emerg. Telecommun. Technol.*, vol. 26, no. 10, pp. 1201–1211, Oct. 2015.

[22] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "A fundamental trade-off between the download cost and repair bandwidth in distributed storage systems," in *Proc. IEEE Int. Symp. Netw. Coding (NetCod)*, Jun. 2010, pp. 1–6.

[23] N. B. Shah, K. V. Rashmi, and P. V. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 1943–1947.

[24] B. Gastón, J. Pujol, and M. Villanueva. (2013). "A realistic distributed storage system that minimizes data storage and repair bandwidth." [Online]. Available: https://arxiv.org/abs/1301.1549

[25] N. Prakash, V. Abdrashitov, and M. Médard, "A generalization of regenerating codes for clustered storage systems," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2016.

[26] N. Prakash, V. Abdrashitov, and M. Médard. (2017). "The storage vs repair-bandwidth trade-off for clustered storage systems." [Online]. Available: https://arxiv.org/abs/1701.04909

[27] B. Choi, J.-Y. Sohn, S. W. Yoon, and J. Moon. (2017). "Secure clustered distributed storage against eavesdroppers." [Online]. Available: https://arxiv.org/abs/1702.07498

[28] Y. Hu *et al.*, "Optimal repair layering for erasure-coded data centers: From theory to practice," *ACM Trans. Storage*, vol. 13, no. 4, p. 33, 2017.

[29] G. Calis and O. O. Koyluoglu. (2016). "Architecture-aware coding for distributed storage: Repairable block failure resilient codes." [Online]. Available: https://arxiv.org/abs/1605.04989

[30] M. Ye and A. Barg, "Explicit constructions of optimal-access MDS codes with nearly optimal sub-packetization," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6307–6317, Oct. 2017.

[31] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 6661–6671, Dec. 2016.

[32] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843–5855, Oct. 2014.

[33] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, Mar. 2011.

[34] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, Algorithms and Applications*. Springer, 2008.

[35] A. Erdélyi, *Asymptotic Expansions* (Dover Books on Mathematics). New York, NY, USA: Dover Publications, 1956. [Online]. Available: https://books.google.co.kr/books?id=aedk-OHdmNYC

[36] J.-Y. Sohn and J. Moon. (2018). "Explicit construction of MBR codes for clustered distributed storage." [Online]. Available: https://arxiv.org/abs/1801.02287

[37] J.-Y. Sohn, B. Choi, and J. Moon. (2018). "A class of MSR codes for clustered distributed storage." [Online]. Available: https://arxiv.org/abs/1801.02014?context=cs

[38] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2005.



**Jy-Yong Sohn** (S'15) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2014 and 2016. He is currently pursuing the Ph.D. degree in KAIST. His current research interests include distributed storage, distributed computing and information theory. He received the IEEE International Conference on Communications (ICC) Best Paper Award in 2017, and the Qualcomm Innovation Award in 2015.

**Beongjun Choi** (S'17) received the B.S. and M.S. degrees in mathematics and electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2014 and 2017. He is currently pursuing the electrical engineering Ph.D degree in KAIST. His research interests include error-correcting codes, distributed storage system and information theory. He is a co-recipient of the IEEE International Conference on Communications (ICC) Best Paper Award in 2017.

**Sung Whan Yoon** (M'17) received the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2013 and 2017 respectively. He is currently a postdoctoral researcher in KAIST from 2017. His research interests are in the area of coding theory, distributed system and artificial intelligence, with focusing on polar codes, distributed storage system and meta-learning algorithm of neural network. Especially for the area of artificial intelligence, his primary interests include information theoretic analysis and algorithmic development of meta-learning. He was a co-recipient of the IEEE International Conference on Communications Best Paper Award in 2017.

**Jaekyun Moon** (F'05) received the Ph.D degree in electrical and computer engineering at Carnegie Mellon University, Pittsburgh, Pa, USA. He is currently a Professor of electrical engineering at KAIST. From 1990 through early 2009, he was with the faculty of the Department of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. He consulted as Chief Scientist for DSPG, Inc. from 2004 to 2007. He also worked as Chief Technology Officer at Link-A-Media Devices Corporation. His research interests are in the area of channel characterization, signal processing and coding for data storage and digital communication. Prof. Moon received the McKnight Land-Grant Professorship from the University of Minnesota. He received the IBM Faculty Development Awards as well as the IBM Partnership Awards. He was awarded the National Storage Industry Consortium (NSIC) Technical Achievement Award for the invention of the maximum transition run (MTR) code, a widely used error-control/modulation code in commercial storage systems. He served as Program Chair for the 1997 IEEE Magnetic Recording Conference. He is also Past Chair of the Signal Processing for Storage Technical Committee of the IEEE Communications Society. He served as a guest editor for the 2001 IEEE JSAC issue on Signal Processing for High Density Recording. He also served as an Editor for the IEEE TRANSACTIONS ON MAGNETICS in the area of signal processing and coding for 2001-2006. He is an IEEE Fellow.