Data Allocation for Multi-Class Distributed Storage Systems

K. P. Roshandeh, M. Noori, *Member, IEEE,*, M. Ardakani, *Senior Member, IEEE,*
and C. Tellambura, *Fellow, IEEE*
Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada
Email: {pourtahm, moslem, ardakani, ct4}@ualberta.ca

***Abstract*—Distributed storage systems (DSSs) are vastly used for reliably storing large amounts of data generated by current and future wireless networks, e.g. social mobile networks or Internet of things. Depending on the features of the data source, various data files may require different levels of quality of service (QoS), e.g. in terms of the probability of successful recovery or data recovery delay. This means that data files can be divided into different classes in terms of their QoS requirements. To address the requirements of each class of data, efficient data (storage) allocation methods, meaning how data is spread over the storage nodes, should be devised. In this paper, we study the optimal data allocation for maximizing the weighted sum of the probability of successful recovery of the data of different classes. Finding such optimal allocations is intractable in general. Therefore, we focus on finding the optimal minimal spreading allocation (MSA) where the data of each class is spread minimally over the storage nodes. MSA possesses several benefits including minimum expected recovery delay and maximum average service rate. Simulation results show that our proposed MSA is indeed the optimal storage allocation in many cases.**

***Index Terms*—DSS, storage allocation, data recovery, fixed-access.**

I. Introduction

DISTRIBUTED storage systems (DSSs) have been extensively employed to store different types of data, e.g. text, audio and video. Availability and fault-tolerance are the two important characteristics of DSSs [1]. That is, a DSS provides anywhere/anytime access to one's data and also enables data recovery even when some data nodes fail. . Furthermore, DSSs give a scalable and cost-effective solution for storing vast amounts of data [2], where the storage capacity can be increased simply by adding more servers.

A DSS consists of different storage nodes with equal or different storage capacities. To store a data file over these nodes, the file first encoded according to a suitable error correction coding scheme. Then, the encoded blocks of the data are stored over the storage nodes according to a *storage allocation* policy [3], [4], determining which storage nodes store which encoded data blocks.

To recover a file, the server needs to receive enough number of encoded blocks that allows for successful decoding of the file. To this end, the server tries to access those storage nodes containing the encoded data blocks of the requested file. Typically, two access models, i.e., probabilistic access and fixed access are considered and studied [3].

In a probabilistic access model, the server asks all the storage nodes that have the encoded blocks of the requested file to send their stored data to the server. However, some nodes may be unavailable or fail otherwise to send the requested data resulting in server not receiving enough encoded blocks. Hence, a probability of successful data recovery, denoted by $P_{\rm s}$ can be defined [3]. In a fixed-access model, the server sends a request to a random subset of size r of nodes to recover the data file. All the nodes in the random subset send the requested data without any failure. Due to the random selection of the accessed nodes, there is a possibility that the selected nodes do not have the whole file collectively. Hence, a probability of successful recover can be associated with this access model as well.

Storage allocation significantly affects various performance measures of DSSs, notably probability of successful recovery, service rate, and recovery delay [3]–[12]. Therefore, several prior studies have focused on finding efficient storage allocations to improve the aforementioned performance metrics.

The optimal storage allocation for minimizing the recovery delay in a DSS with probabilistic access model has been studied in [10]. The authors in [12] have investigated the optimal storage allocation to maximize the service rate for both fixed and probabilistic access models. The optimal symmetric allocation(i.e., when the encoded data blocks are spread evenly over only a subset of nodes) has been studied for both probabilistic and fixed access models in [3]. Two allocation algorithms that try to maximize the probability of successful recovery when different nodes have different access probability are proposed in [6].

A multi-class DSS refers to a DSS that stores the data of multiple data sources, each with different quality of service (QoS) requirements. Multi-class DSSs have recently gained attention as they can meet different QoS requirements of several classes of data [13], [14]. The optimal storage allocation in terms of maximizing the weighted sum of the probabilities of successful recovery of all data classes in a multi-class DSS with probabilistic access model has been studied in [14].

In this paper, we extend the analysis of [14] to a multi-class DSS with fixed-access model. The goal is to maximize the weighted sum of the probabilities of successful recovery of all data classes where different weights represent different QoS requirements of the data classes. The optimal storage allocation for the considered problem is unknown even for a single-class DSS (where there is only one data class) [3]. Having more than one data class makes this problem even more difficult since the storage allocation of a data class affects the storage allocation,

and consequently the probability of successful recovery of other classes. In view of this, we focus on finding the optimal storage allocation that spreads the data minimally over the storage nodes, i.e. minimal spreading allocation (MSA). It has been shown that MSA is optimal in terms of
1) minimum expected recovery delay [10] and maximum average service rate [12] for a single-class DSS
2) maximum probability of successful recovery in some ranges of access probabilities to the nodes, e.g. for small access probabilities, in a single-class DSS [3]
3) maximizing the weighted-sum of the probabilities of successful recovery in a multi-class DSS with probabilistic access model, for large access probabilities [14].

The success of MSA in other setups motivates us to study its performance for multi-class fixed-access DSSs. More specifically, we propose an algorithm for finding the optimal MSA in a multi-class DSS with heterogeneous nodes in term of their storage capacities. Simulation results show that, when the weighted sum of the probabilities of successful recovery for the proposed optimal MSA achieves the upper bound. In other words, the optimal MSA is indeed *the optimal storage allocation* when r is sufficiently large. In [14], we have solved a similar problem for probabilistic access model. However, the same approach cannot be used here as the nature of the objective function of the optimization problem is completely different from that in [14], i.e. binomial instead of polynomial. Moreover, here we solve the optimization problem for a more general case, where the nodes may not have equal unit storage capacity.

II. System Model

A. Storage Model

We consider K different classes of data for storing over N storage nodes. We assume each data class has k data blocks and a proper minimum distance separable (MDS) code is used to form k_i coded data blocks for class i, $\forall i \in \mathcal{K} = \{1, 2, ..., K\}$, from the original k blocks. The number of encoded blocks k_i is limited by a storage constraint $k_i \leq T_i$, where the storage limit T_i is determined according to the corresponding data class QoS. The vector of storage constraints for all data classes is defined as $\mathcal{T} = (T_1, T_2, \ldots, T_N)$. The encoded data blocks of all data classes are spread over the N storage nodes based on a storage allocation policy. We denote the storage capacity of node n by c_n such that $c_n \geq k$, $\forall n \in \mathcal{N} = \{1, 2, ..., N\}$. Since an MDS coding is used, to recover the k data blocks of class i, it is sufficient to receive k out of k_i encoded data blocks from the N storage nodes. Let us define $x_{i,n}$ as the number of encoded data blocks of class i that is stored over storage node n, where $x_{i,n} \leq c_n$.

We normalize all the parameters by k to simplify the mathematical representation. However, with some abuse of notations, we still use c_n and T_i as the normalized node storage capacities and the normalized storage constraints. Moreover, $x_{i,n}$ and k_i denote the normalized number of encoded blocks of class i stored in node n and the total number of encoded blocks of the i-th class. Hence, the server needs at least one (normalized) unit of encoded blocks of class i to recover the data of this class. The node storage constraint and the class storage constraint imply

$$\sum_{i \in \mathcal{K}} x_{i,n} \leq c_n,$$

and

$$\sum_{n \in \mathcal{N}} x_{i,n} = k_i \leq T_i.$$

A storage allocation policy describes how the encoded data blocks of all data classes are spread over the storage nodes as formally defined in the following:

Definition 1. *A storage allocation is a K-tuple $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K)$ where $\mathcal{A}_i = (x_{i,1}, x_{i,2}, ..., x_{i,N})$ for all $i \in \mathcal{K}$.*

B. Access Model

Here, we employ a fixed-access model where the server randomly accesses a fixed number of nodes when a download request arrives [3]. In other words, the server attempt to acquire the required encoded blocks from a random subset of nodes $\mathbf{r}$ where the cardinality of the subset is fixed, i.e. $|\mathbf{r}| = r$. Data of class i can be recovered if $\sum_{n \in \mathbf{r}} x_{i,n} \geq 1$.

For a fixed-access model, the probability of successful recovery for a given data class i can be expressed as

$$P_{\mathrm{s},i} = \frac{1}{\binom{N}{r}} \sum_{\boldsymbol{r} \subseteq \mathcal{N}, |\mathbf{r}| = r} \mathbf{I}\left[\sum_{n \in \mathbf{r}} x_{i,n} \geq 1\right]. \tag{1}$$

where r denotes the cardinality of the random sets. Also, $\mathbf{I}[Y]$ denotes the indicator function such that $\mathbf{I}[Y] = 1$ if Y holds, and $\mathbf{I}[Y] = 0$ otherwise. It should be note that putting more than one unit of data from a specific data class in a storage node is pointless since only one unit of that class data is needed for recovery.

For a given storage allocation policy $\mathcal{A}$, $\mathbf{p}_\mathrm{s} = [P_{\mathrm{s},i}]_{i \in \mathcal{K}}$ denotes the vector of probabilities of successful recovery of all data classes where class i data is recovered with probability $P_{\mathrm{s},i}$. For a given vector of storage constraints, the union of all vectors $\mathbf{p}_\mathrm{s}$ is denoted by $\boldsymbol{\Theta}$, and is known as the feasible region of probabilities of successful recovery for a multi-class DSS [14].

III. Problem Definition

To account for different classes's QoS requirements, we assign a weight to the probability of successful recovery of each data class. That is, the class with higher importance in terms of QoS is assigned a higher weight compared to other classes. Let w_i denote the weight assigned to class i. Without loss of generality, we assume $\sum_{i=1}^{K} w_i = 1$. Moreover, $\boldsymbol{w} = [w_i]_{i \in \mathcal{K}}$ denotes the vector of weights of all data classes. The goal is to maximize the weighted sum of the probabilities of successful recovery of all data classes. The mathematical definition of the optimization problem can be expressed as

$$\begin{aligned} \underset{\mathcal{A}}{\text{maximize}} \quad & \langle \boldsymbol{w}, \mathbf{p}_\mathrm{s} \rangle \\ \text{subject to} \quad & \mathbf{p}_\mathrm{s} \in \boldsymbol{\Theta}, \end{aligned} \tag{P1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors.

The weighted form of the objective function comes from the fact that the optimization problem has a multi-objective nature, if we aim at maximizing the probability of successful recovery for all or a number of classes. One way to deal with multi-objective optimization problems is to work with a sum or a weighted sum of the objective functions. Here, we use the weighted sum of $P_{\mathrm{s},i}$ as the objective function.

In a general case, the problem (P1) is very difficult so solve, however, the solution can be found in some cases. The following proposition gives the optimal solution for a specific case of system setup.

Proposition 1. *Let m and j denote the node with smallest storage capacity and the class with smallest storage constraint, respectively. Assume $K \leq rc_m$ and $N \leq rT_j$ where r denotes the cardinality of the random subsets accessed by the server. Then, the optimal storage allocation is $x_{i,n} = \frac{1}{r}, \forall i \in \mathcal{K}$ and $\forall n \in \mathcal{N}$.*

Proof. If we set $x_{i,n} = \frac{1}{r}, \forall i \in \mathcal{K}$ and $\forall n \in \mathcal{N}$, then neither the budget limits of the classes are violated nor the capacity limits of the storage nodes. Also, all the data classes have the probability of successful recovery of one which is the highest possible amount. Thus, the weighted sum achieves its highest amount. □

As can be seen, the proposed solution given in Proposition 1 is optimal. However, solving the optimization problem (P1) in a general setup is intractable. This is because finding Θ is burdensome, otherwise, (P1) turns into a linear optimization problem, and one could easily solve the problem with a known Θ. The difficulty of finding Θ lies in the fact that the data allocation of one class is strongly correlated with the way the data of other classes are allocated in the system. The following example better illustrates the complexity of the problem.

Example 1. *Consider a DSS with $N = 3$ nodes where $c_1 = c_2 = c_3 = 1$. We want to store two classes of data over these nodes where the storage constraints are $T_1 = \frac{3}{2}, T_2 = \frac{5}{4}$. Table I shows four possible allocations for this setup. For $r = 1$, Case 1 results in the maximum of $P_{\mathrm{s},1}$, $P_{\mathrm{s},2}$ and $3P_{\mathrm{s},1} + P_{\mathrm{s},2}$. Assuming $r = 2$, if we only focus on maximizing $P_{\mathrm{s},1}$, then Case 2 and 3 maximize $P_{\mathrm{s},1}$. However, Case 1 and 4 result in maximum $P_{\mathrm{s},2}$ for $r = 2$. On the other hand, if the goal is to maximize $3P_{\mathrm{s},1}+P_{\mathrm{s},2}$, then Case 3 gives the optimal allocation.* ■

TABLE I
DIFFERENT ALLOCATIONS

	Storage allocation	
Case 1:	$\mathcal{A}_1 = (1, \frac{1}{2}, 0)$	$\mathcal{A}_2 = (0, \frac{1}{4}, 1)$
Case 2:	$\mathcal{A}_1 = (1, \frac{3}{8}, \frac{1}{8})$	$\mathcal{A}_2 = (0, \frac{5}{8}, \frac{5}{8})$
Case 3:	$\mathcal{A}_1 = (\frac{3}{4}, \frac{2}{4}, \frac{1}{4})$	$\mathcal{A}_2 = (\frac{1}{4}, \frac{1}{4}, \frac{3}{4})$
Case 4:	$\mathcal{A}_1 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\mathcal{A}_2 = (\frac{5}{12}, \frac{5}{12}, \frac{5}{12})$

As it can be seen from the above example, for a DSS with given storage constraints and storage capacity of the nodes, the optimal storage allocation depends on the choice of r. In addition, even different feasible allocations vectors express different recovery performances depending on the performance objective function (goal) defined for the DSS. The problem becomes even more challenging when storage constraints and the node capacities can also take arbitrary values as we consider in our problem.

To this end, we focus on one of the most promising storage allocation schemes known as minimal spreading allocation (MSA). In an MSA scheme, the data of each class is spread minimally over the storage nodes. The formal description of an MSA strategy is presented in the following.

Definition 2. *Let $\mathbb{D}_i \subset \mathcal{N}$ denote the set of nodes in which the data of the i-th class is stored. Then, for an MSA strategy, we have*

$$x_{i,n} = \begin{cases} 1 & \forall n \in \mathbb{D}_i \\ 0 & \textit{otherwise.} \end{cases}$$

In other words, in an MSA, we either put a unit amount of data from class i on a node or we do not put any thing at all. Thus, MSA does not require employing any coding scheme, and therefore reduces the computational complexity to a great extent at the server side.

It has been shown that an MSA strategy is optimal for minimizing the expected recovery delay, maximizing the average service rate and maximizing probability of successful recovery for many cases in single-class DSSs [3], [10], [12]. Moreover, assuming a probabilistic access model, it is shown in [14] that MSA scheme is near-optimal for a wide range of nodes' access probabilities in terms of maximizing the weighted sum of the probability of successful recovery of all data classes in a multi-class DSS. Hence, we focus on finding the optimal MSA strategy for our problem.

IV. Optimal MSA Solution

After narrowing our attention to MSA strategies, optimization problem (P1) turns into a non-linear integer optimization problem as follows

$$\begin{aligned} \underset{x'_{i,n}s}{\text{maximize}} \quad & \sum_{i=1}^{K} \sum_{\boldsymbol{r} \subseteq \mathcal{N}, |\boldsymbol{r}|=r} \frac{w_i}{\binom{N}{r}} \mathbf{I}\left[\sum_{n \in \mathbf{r}} x_{i,n} \geq 1\right] \\ \text{subject to} \quad & \sum_{i \in \mathcal{K}} x_{i,n} \leq c_n, \forall n \in \mathcal{N} \\ & \sum_{n \in \mathcal{N}} x_{i,n} \leq T_i, \forall i \in \mathcal{K} \\ & x_{i,n} \in \{0, 1\}. \end{aligned} \tag{P2}$$

There is no systematic solution for the optimization problem (P2), and therefore we propose a recursive algorithm for finding the optimal MSA policy as shown in Algorithm 1.

Algorithm 1 works based on maximizing the reward gained by assigning a storage unit to a data class at each step. The reward for each class is defined as the amount of increase in the weighted sum after assigning the next storage unit to that data class. The reward for the i-th data class is denoted by $\mathcal{R}_i^{z \to z+1}$ where z is the number of storage units assigned to

the i-th data class until the current time. That is, for the i-th data class we have

$$\mathcal{R}_i^{z\to z+1} = \begin{cases} U_i^{(1)}(z) & \text{if } z = N-r \\ U_i^{(2)}(z) & \text{if } z < N-r \\ 0 & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} U_i^{(1)}(z) &= w_i P_{\mathrm{s},i}^{z+1} - w_i P_{\mathrm{s},i}^{z} \\ &= w_i(1) - w_i(1 - \frac{\binom{N-z}{r}}{\binom{N}{r}}) \\ &= \frac{w_i}{\binom{N}{r}} \end{aligned} \tag{2}$$

and

$$\begin{aligned} U_i^{(2)}(z) &= w_i P_{\mathrm{s},i}^{z+1} - w_i P_{\mathrm{s},i}^{z} \\ &= w_i(1 - \frac{\binom{N-z-1}{r}}{\binom{N}{r}}) - w_i(1 - \frac{\binom{N-z}{r}}{\binom{N}{r}}) \\ &= w_i(\frac{\binom{N-z}{r} - \binom{N-z-1}{r}}{\binom{N}{r}}) \\ &= w_i(\frac{\binom{N-z-1}{r-1}}{\binom{N}{r}}) \end{aligned} \tag{3}$$

where the last equality follows from Pascal's formula, and $P_{\mathrm{s},i}^{z}$ denotes the probability of successful recovery of the i-th class when z units of its data blocks have been stored over the nodes.

It can be shown that the reward of the i-th class is a decreasing function of the number of assigned storage units to that data class. It turns out that this is an essential property for proving the optimality of the proposed algorithm. Now, we are ready to explain Algorithm 1 in detail.

We assign one storage unit at each step of the algorithm. Before the assignment happens, the rewards of all classes are calculated. At each step, we choose those classes that meet three conditions

- The class should have a nonzero reward.
- There should be at least a node with available capacity that does not contain the data of the chosen class (the node availability constraint).
- The class storage constraint should not be violated by assigning the next storage unit to the data class.

The first condition indicates that assigning a storage unit to a class with zero reward is pointless. The second condition implies that only one unit of data is needed for data recovery. The algorithm also takes care of not violating the storage constraint of the data class by assigning the next storage unit to this class. As Algorithms 1 proceeds, two cases can happen:

- **Case1**: there is at least one data class satisfying all the three conditions
- **Case2**: there is no such data class

It should be noted that at the first step (assigning the first storage unit) of the algorithm, Case 1 surely happens. Thus, we first consider Case 1. Among all the classes satisfying the three conditions (having nonzero reward, the node availability constraint and the class storage constraint), the algorithm picks the class with the largest reward. This is because the class with the largest reward makes the largest contribution to the objective function while satisfying the conditions. Assume the i-th class is selected. Based on the node availability constraint, there is at least a node with available capacity that does not contain the data of the chosen class. Among the nodes satisfying the condition for class i, the node with largest available capacity is chosen. We put the i-th class data in the node with the largest available capacity to make sure we neither use the total storage capacity of a node (if possible) nor increase the chance of the node being filled sooner. That is, we try to keep a node available to other data classes as long as it is possible. This strategy as we assign the storage units plays a major role in the optimality of the algorithm. The algorithm then updates the number of storage units dedicated to class i, i.e. increases z by one. Next, the algorithm goes back to the first step and updates the reward of the i-th data class and repeats the procedure.

Now, assume Case 2 happens. In this case, the algorithm is terminated and returns the number of storage units assigned to each data class until this step. Since we assign one storage unit at a time, the complexity of the algorithm is $\mathcal{O}(\sum_{n=1}^{N} c_n)$.

Algorithm 1:

```
while can put data do
    compute all the rewards of all classes;
    find all the classes with nonzero reward that satisfy
     both node availability constraint, class storage
     constraint;
    if could not find such data class then
        return;
    else
        pick the class with largest reward among the
         selected classes;
        put the chosen class data in the node with largest
         available capacity;
        update z for the chosen data class;
    end
end
```

V. Case Study: $r = 1$, without class storage constraint

In this section, we consider a special setup where the server has access to only one node each time. Moreover, we do not consider any storage constraint for the data classes, i.e. $T_i \geq N$. The following lemma is used for our discussion later in the section.

Lemma 1. *Assume $r = 1$ and $T_i \geq N, \forall i \in \mathcal{K}$. Then, the following procedure results in the optimal MSA.*

***Procedure 1**: in each step, we pick a class and put its data on the largest number of nodes that have available capacity. We start with the most important class and continue our way towards the least important one.*

Proof. In the first step of the procedure, the data of the class with the largest weight is stored in all nodes. Then, the data of the class with the second largest weight is stored in the largest number of nodes with available capacity. We follow this procedure for all data classes. This procedure ends either when we have gone through all data classes or when all the storage nodes are filled.

According to (2) and (3), the reward for the i-th class when $r=1$ is $\mathcal{R}_i^{z\to z+1}=\frac{w_i}{N}$. That is, the data of the class with a larger weight should be stored over the maximum number of nodes before putting the data of a class with a lower weight. This proves the optimality of the explained procedure. □

The next theorem gives a closed-form expression for the weighted sum achieved by the optimal MSA for the considered setup.

Theorem 1. *Assume $r=1$ and $T_i > N, \forall i \in \mathcal{K}$. Also let $w_1 \geq w_2 \geq \ldots \geq w_N$ and $c_1 \geq c_2 \geq \ldots \geq c_N$. Assume there are M unique values among the capacities of storage nodes. Let $u_1, u_2, \ldots, u_M$ denote these unique storage capacities where $u_1 > u_2 > \ldots > u_M$ and $M \leq N$. Also, assume there are n_i nodes with storage capacity u_i. Then, the weighted-sum of probabilities of successful recovery for all data classes achieved by the optimal MSA is*

$$\langle \boldsymbol{w}, \mathbf{p}_{\mathrm{s}} \rangle = \begin{cases} \sum_{i=1}^{K} w_i & K \leq u_N \\ F_1 & K > u_1 \\ F_2 & K = u_t,\ t \in \mathcal{N} \\ F_3 & \textit{otherwise} \end{cases}$$

where

$$F_1 = \sum_{p=2}^{M+1} \sum_{i=u_p+1}^{u_{p-1}} \frac{w_i (N - \sum_{j=p}^{M+1} n_j)}{N}$$

$$F_2 = \sum_{p=t+1}^{M+1} \sum_{i=u_p+1}^{u_{p-1}} \frac{w_i (N - \sum_{j=p}^{M+1} n_j)}{N}$$

$$F_3 = \sum_{p=b+1}^{M+1} \sum_{i=u_p+1}^{u_{p-1}} \frac{w_i (N - \sum_{j=p}^{M+1} n_j)}{N} + \sum_{i=u_b+1}^{K} \frac{w_i (N - \sum_{j=b}^{M+1} n_j)}{N}$$

and u_b is the largest unique capacity of nodes smaller than K. Also, $u_{M+1} = n_{M+1} = 0$.

Proof: As shown in Lemma 1, Procedure 1 results in the optimal MSA in this setup. Following the explained procedure, one can easily show that the number of classes that their data is stored over N nodes is u_M, and their probability of successful recovery is 1. There are $u_{M-1} - u_M$ number of classes that $N - n_M$ data blocks of each one of them have been stored over the storage nodes with probability of successful recovery of $P_{\mathrm{s},i} = \frac{N-n_M}{N}$. Also, the number of classes that their data is stored over $N - n_M - n_{M-1}$ nodes is $u_{M-2} - u_{M-1}$ each with $P_{\mathrm{s},i} = \frac{N-n_M-n_{M-1}}{N}$ and so on. By continuing this procedure and adding all the probabilities of successful recovery, one can achieve the weighted sum shown in Theorem 1 based on the value of K.

■

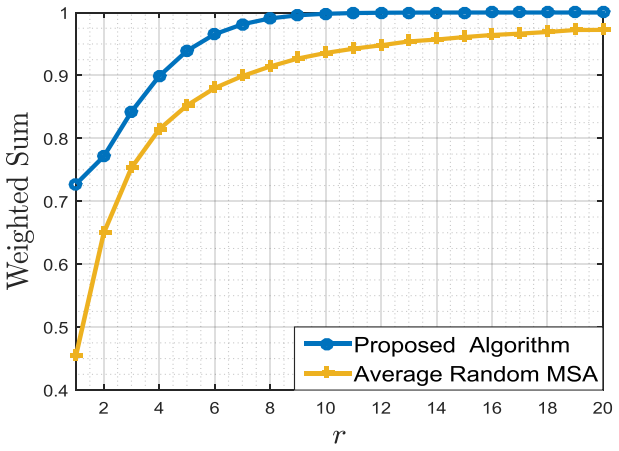

Fig. 1. The performance of optimal MSA compared to the averaged random MSA.

VI. Simulation Results

Simulation results for a multi-class DSS with three data classes and 20 nodes is depicted in Figure 1. The capacity of the nodes and weights of classes are $c_i = 1, \forall i \in \{1, 2, ..., 20\}$, $w_1 = \frac{8}{11}, w_2 = \frac{2}{11}$ and $w_3 = \frac{1}{11}$, respectively. Also, the storage constraints for data classes are $T_1 = 20, T_2 = 14$ and $T_3 = 5$.

The optimal MSA achieved using Algorithm 1 for maximizing the weighted sum of the probabilities of successful recovery of all data classes is given in Figure 1. Also, the average random MSA strategy where a random number of storage units assigned to each data class while taking care of the feasibility conditions of the system (e.g. the class storage constraint) is also depicted for comparison. The term *average* comes from the fact that the curve is obtained by averaging over 10^4 weighted sums of different random MSAs. As can be seen, the optimal MSA outperforms the average random MSA.

Moreover, the weighted sum achieved by the proposed algorithm and *weighted storage assignment method* for a DSS with arbitrary node capacities have been given in Figure 2. More specifically, we consider 10 storage nodes where the storage capacity of nodes are given as $c_1 = 5$, $c_2 = 4$ $c_3 = 6$, $c_4 = 5$, $c_5 = 2$, $c_6 = 2$, $c_7 = 3$, $c_8 = 2$, $c_9 = 1$ and $c_{10} = 3$. We also consider seven data classes with storage constraints $T_1 = 10$, $T_2 = 10$, $T_3 = 9$, $T_4 = 8$, $T_5 = 6$, $T_6 = 5$ and $T_7 = 4$. The weights for the classes are $w_1 = \frac{18}{51}$, $w_2 = \frac{12}{51}$, $w_3 = \frac{10}{51}$, $w_4 = \frac{6}{51}$, $w_5 = \frac{3}{51}$, $w_6 = \frac{1}{51}$ and $w_7 = \frac{1}{51}$. In the *weighted storage assignment method*, the number of storage units of class i stored over a DSS is determined by $\sum_{n\in\mathcal{N}} x_{i,n} = \min\{\lceil \frac{w_i \sum_{n\in\mathcal{N}} c_n}{\sum_{i=1}^{K} w_i} \rceil, N\}$. Note that this is possible if $\min\{\lceil \frac{w_i \sum_{n\in\mathcal{N}} c_n}{\sum_{i=1}^{K} w_i} \rceil, N\} \leq T_i,\ \forall i \in \mathcal{K}$ which is the case in the considered setup.

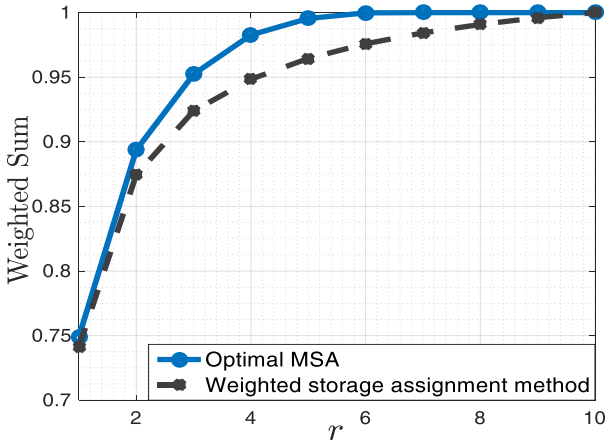

Fig. 2. The performance of optimal MSA.

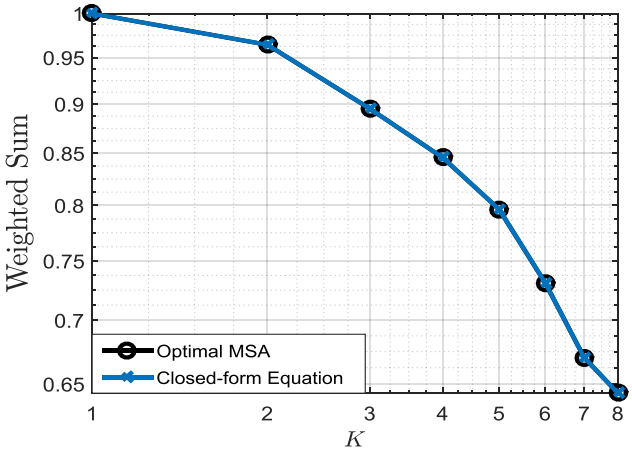

Fig. 3. Closed-form equation result for the considered DSS.

As shown in Figure 2, the optimal MSA obtained using the proposed algorithm almost achieves the upperbound (the line where the weighted sum is equal to one) for some ranges of r (cardinality of the random subsets), i.e. $r \geq 6$, and therefore is the optimal storage allocation. Also, the optimal MSA outperforms the weighted storage assignment method.

Finally, the weighted sum achieved by the optimal MSA and the closed-form equation given in Theorem 1 have been depicted in Figure 3. The capacity of nodes are $c_1 = 5$, $c_2 = 5$ $c_3 = 7$, $c_4 = 5$, $c_5 = 6$, $c_6 = 2$, $c_7 = 3$, $c_8 = 2$, $c_9 = 1$ and $c_{10} = 3$. The initial weights (as we remove one class and its correspondences each time to reduce K) of the classes are $w_1 = \frac{18}{74}$, $w_2 = \frac{10}{74}$, $w_3 = \frac{14}{74}$, $w_4 = \frac{6}{74}$, $w_5 = \frac{8}{74}$, $w_6 = \frac{8}{74}$, $w_7 = \frac{7}{74}$ and $w_8 = \frac{3}{74}$, and $r = 1$. Figure 3 verifies the mathematical analysis given in Theorem 1.

VII. Conclusion

In this paper, we considered a multi-class DSS with arbitrary storage node capacities. The term multi-class refers to different classes of data, each requiring different levels of QoS which are to be stored over the storage nodes. To account for different QoS requirements, we aimed at maximizing a weighted-sum of the probabilities of successful recovery of different classes. A fixed-access model has been considered for the DSS. An algorithm for finding the optimal MSA has been proposed. The simulation results confirmed an improvement gap between the optimal MSA and average random MSA. Moreover, simulation results showed that MSA achieves the upperbound.

VIII. Acknowledgement

This work was supported by Alberta Innovates Technology Futures (AITF), Natural Sciences and Engineering Research Council of Canada (NSERC), and TELUS Corporation.

References

[1] 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, 2010.

[2] M. Placek and R. Buyya, "A taxonomy of distributed storage systems," *Reporte técnico, Universidad de Melbourne, Laboratorio de sistemas distribuidos y cómputo grid*, 2006.

[3] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4733–4752, 2012.

[4] M. Sardari, R. Restrepo, F. Fekri, and E. Soljanin, "Memory allocation in distributed storage networks," in *IEEE Intl. Symp. on Information Theory (ISIT)*, June 2010, pp. 1958–1962.

[5] B. Hong and W. Choi, "Optimal storage allocation for wireless cloud caching systems with a limited sum storage capacity," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6010–6021, Sept 2016.

[6] V. Ntranos, G. Caire, and A. G. Dimakis, "Allocations for heterogenous distributed storage," in *IEEE Intl. Symp. on Information Theory (ISIT)*, 2012, pp. 2761–2765.

[7] Z. Li, T. Ho, D. Leong, and H. Yao, "Distributed storage allocation for heterogeneous systems," in *Allerton Conf. on Communication, Control, and Computing*, Oct 2013, pp. 320–326.

[8] M. Noori and M. Ardakani, "Allocation for heterogeneous storage nodes," *IEEE Commun. Lett.*, vol. 19, no. 12, pp. 2102–2105, Dec 2015.

[9] I. Andriyanova and P. M. Olmos, "On distributed storage allocations for memory-limited systems," in *IEEE Global Communications Conf. (GLOBECOM)*, 2015, pp. 1–6.

[10] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocations for optimal delay," in *IEEE Intl. Symp. on Information Theory (ISIT)*, July 2011, pp. 1447–1451.

[11] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, 2014.

[12] M. Noori, E. Soljanin, and M. Ardakani, "On storage allocation for maximum service rate in distributed storage systems," in *IEEE Intl. Symp. on Information Theory (ISIT)*, July 2016, pp. 240–244.

[13] A. Kumar, R. Tandon, and T. C. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 221–233, 2017.

[14] K. Roshandeh, M. Noori, M. Ardakani, and C. Tellambura, "Distributed storage allocation for multi-class data," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 2223–2227.