Computing and Informatics, Vol. 39, 2020, 51–82, doi: 10.31577/cai_2020_1-2_51

OPTIMIZING DATA PLACEMENT FOR COST EFFECTIVE AND HIGH AVAILABLE MULTI-CLOUD STORAGE

Pengwei Wang, Caihui Zhao, Wenqiang Liu
Zhen Chen, Zhaohui Zhang

School of Computer Science and Technology
Donghua University
201620 Shanghai, China
e-mail: wangpengwei@dhu.edu.cn

Abstract. With the advent of big data age, data volume has been changed from trillionbyte to petabyte with incredible speed. Owing to the fact that cloud storage offers the vision of a virtually infinite pool of storage resources, data can be stored and accessed with high scalability and availability. But a single cloud-based data storage has risks like vendor lock-in, privacy leakage, and unavailability. Multi-cloud storage can mitigate these risks with geographically located cloud storage providers. In this storage scheme, one important challenge is how to place a user's data cost-effectively with high availability. In this paper, an architecture for multi-cloud storage is presented. Next, a multi-objective optimization problem is defined to minimize total cost and maximize data availability simultaneously, which can be solved by an approach based on the non-dominated sorting genetic algorithm II (NSGA-II) and obtain a set of non-dominated solutions called the Pareto-optimal set. Then, a method is proposed which is based on the entropy method to determine the most suitable solution for users who cannot choose one from the Pareto-optimal set directly. Finally, the performance of the proposed algorithm is validated by extensive experiments based on real-world multiple cloud storage scenarios.

Keywords: Data hosting, cloud storage, multi-cloud, multi-objective optimization, genetic algorithm

Mathematics Subject Classification 2010: 68-M02

1 INTRODUCTION

With the rapid development of Internet, mobile Internet, IoT and other related technologies [1, 2, 3], the explosive growth of data volume has become an important and challenging issue. From the statistical result, 8×10^5 PB of data were generated and replicated in the world by the year of 2000 and it is expected that this number will increase to 35 ZB by 2020 [4]. Storing such data volume has been an important and challenging issue for enterprises.

In recent year, cloud computing has become a popular computing paradigm for hosting and delivering services over the Internet [5, 6]. Cloud storage, the prominent service in cloud computing, is synonymous with pay-for-use pricing structures. Compared with the traditional storage mode, cloud-based data storage offers high availability, durability, and scalability. However, a single cloud-based data storage comes with the following risks.

Data unavailability. The first obstacle to the growth of Cloud Computing is the availability of a service [7]. The availability of data is also an indicator that users are most concerned about. Although cloud service providers (CSPs) have strict Service Level Agreement (SLA) for their services, some unpredictable events may cause services to be unavailable. These unpredictable events include server downtime, natural disasters, power failures, and so on. On August 8th, 2016, Google Cloud Storage and File Backup Server service terminals crashed, which brought huge economic losses to users. Coincidentally, in the afternoon of December 7$^{\text{th}}$, 2017, Alibaba Cloud's domain name resolution failed due to sudden large-scale traffic attacks. Unavailability of services can bring huge economic losses to users.

Vendor lock-in. Vendor lock-in is a major barrier to the adoption of cloud computing, due to the lack of standardization [8]. The vendor lock-in problem in cloud storage is the situation where users are vulnerable to price hike, availability decrement, or even to provider bankruptcy [7]. The reason why users give up migrating their data to a new provider who provides better service or lower price is the expensive bandwidth cost. Consequently, once a provider adjusts price, users are on the horns of a dilemma [27]. Moreover, the time it takes to migrate large amounts of data from one CSP to another is also huge [9].

Data privacy leakage. As data is stored with a third party, users want to avoid an untrusted CSP. If users put their data into a single cloud provider, their data is completely exposed to CSP, easily causing data privacy leakage. Data privacy leakage mainly includes cases where some untrusted CSP steal data without user permission. Malicious insiders of CSP can steal or corrupt the data and external attacks may also lead to data privacy leakage [10].

Recently, multi-cloud storage can mitigate the abovementioned risks with geographical providers and also provide benefits including adequate responsiveness, better load balance, and quick data recovery [11]. In multi-cloud storage, there

Symbols	Descriptions
C	List of cloud providers
N	Total number of cloud service
m	Number of the data splitting into
n	Number of the data storing
i	$i = 1, 2, \ldots, N$
S	Size of the file
P	Total cost of the data hosting scheme
P_{stor}	Total price of storage in scheme
P_{net}	Total out of out-bandwidth in scheme
P_{op}	Total price of operation in scheme
τ_t	Number of users access to the file
C_T	Total cost of a data file
C_i	Total cost of the i^{th} cloud service
P_{si}	Storage price of the i^{th} cloud service
P_{bi}	Out-bandwidth price of the i^{th} cloud service
P_{oi}	Operation price of the i^{th} cloud service
A_i	Availability of the i^{th} cloud service
A_{req}	Lowest limit of the data availability

Table 1. Symbol table

are many metrics with which users are concerned, especially low cost and high availability. The price of the same service across providers is different, and a provider offers different service with the same functionality while performance is directly proportional to price [11]. If users desire to enhance data availability, the more cost is incurred.

In multi-cloud storage, two main redundant strategies to categorize data distributed storage are replication and erasure coding [27]. For erasure coding, a data object is divided into m equal-size chunks and these chunks are used to generate $(n - m)$ encoded data chunks. Users can retrieve the original data through any m data chunks of these n chunks and tolerate any $0 \sim (n - m)$ cloud providers' shutdown at the same time. This strategy can reduce storage cost compared with data replication.

From a user's perspective, the key issue is to maximize data availability by minimizing data management cost that consists of *storage* cost and *network* cost (i.e., operation cost and out-bandwidth cost) [4]. The goal of optimizing multi-objective functions is to obtain the optimal data placement given cloud storage providers. In other word, the optimization problem is: **How to choose CSPs so as to minimize data management cost and maximize data availability?**

In this paper, an architecture in multi-cloud storage is presented at first. Next, a multi-objective optimization problem is defined to minimize monetary cost and maximize data availability. Then, an approach based on the non-dominated sorting genetic algorithm II (NSGA-II) [31] is given, whose goal is to effectively solve a multi-objective optimization problem and obtain a set of non-dominated solutions (i.e.,

list of cloud storage providers) and erasure coding parameters. Since CSPs with low (resp. high) storage cost may have low (resp. high) availability and high (resp. low) network cost, it is nontrivial to trade off data management cost and data availability. Some users can choose the data placement solution from the Pareto-optimal set directly. However, most users are still confused when they face the Pareto-optimal set. In order to recommend a suitable solution for such users, a method based on the entropy method is proposed. Finally, we demonstrate the performance of the proposed algorithm dealing with the real-world cloud storage providers from *CloudHarmony, cloudharmony* in a simulation.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents a cloud storage scenario to show the risks of reliance on a single cloud and discuss the benefits of multi-cloud storage. Section 4 formulates a data placement problem. The proposed algorithm is presented in Section 5. The performance of our proposed algorithms is shown via extensive experiments by using real-world cloud information in Section 6. Finally, Section 7 concludes the paper.

2 RELATED WORK

More and more users are hosting their data into multi-cloud not only to overcome the risks arising from reliance on a single cloud, but also to obtain higher availability, lower latency, and lower monetary cost [11]. Reducing monetary cost and enhancing data availability are two most important driving forces for users to host their data into the cloud. According to the optimized metrics, the existing studies can be divided into three categories, i.e., monetary cost optimization based on QoS metrics, data availability optimization based on QoS metrics, and cost-availability trade-off. In the following, we review and discuss them, respectively.

2.1 Monetary Cost Optimization Based on QoS Metrics

In this category, minimizing the monetary cost based on some QoS metrics is full or part of the work in the studies. Abu-Libdeh et al. [13] propose Redundant Array of Cloud Storage (RACS), a proxy striping user data across multiple providers to reduce the cost of switching providers. However, it does not propose a method to solve the data placement problem to meet any optimization goal. Papaioannou et al. [14] propose Scalia as inspired by RACS. It is a cloud storage brokerage solution for adaptive data placement, which minimizes the storage cost. Furthermore, Mansouri et al. [9] present an algorithm to find subsets of data centers to store original data and their replicas such that the storage cost is minimized while the expected availability is guaranteed.

But the above two tasks only consider a part of the monetary cost optimization: the cost of switching providers and storage cost. The cost of data storage management in the cloud should consist of residential cost (i.e. storage and data access operations), and network cost resulting from data transfer from CSP [4].

In [15], Hadji proposes a commodity flow solution to minimize the cost of storing data and latency to access data centers. However, the network and operation costs are ignored when users access their data. Ma et al. [16] adopt the ensemble of replication and erasure coding leading to low bandwidth cost, low storage cost, and low latency, but ignore the proper selection of CSPs. One function in CHARM, proposed by Zhang et al. [17], is to select the data placement configuration which contains several suitable clouds and an appropriate redundancy strategy to store their data with minimized data storage management cost and guaranteed availability. But the proposed algorithm to solve the minimization problem is a simple heuristic solution and cannot obtain the global optimal one. The studies in [15] and [17] only minimize the monetary cost at a certain point in the time slot. Mansouri et al. [18] propose the optimal offline algorithm to minimize the residential and migration costs in a time slot where the exact and known future workload is assumed.

There are also several studies to minimize the monetary cost based on QoS metrics for a geographical distributed cloud storage. Wu et al. [19] present a unified view of storage services in geographically distributed data centers called *SPANStore*. It aims to minimize the monetary cost and compute resources with the constraints of GET/PUT latencies, flexible consistency, and tolerate failures. Liu et al. [20] propose a multi-cloud service to minimize the payment cost while providing Service Level Objective (SLO) guarantee to customers. In order to minimize the payment cost, the authors propose a heuristic solution based on genetic algorithm to maximize the reservation benefit. However, these studies mainly focus on GET/PUT latency, this paper focuses on the optimization of data availability and monetary cost.

2.2 Data Availability Optimization Based on QoS Metrics

In addition to optimizing cost, increasing data availability is also an optimization objective in recent studies. As mentioned above, data replication and erasure coding are two main data redundancy strategies to improve data availability [11]. In [21, 22], the authors compare them. Here, we briefly introduce them.

Data Replication. Wei et al. [23] propose a novel model to capture the relationship between availability and the number of replica. It only calculates the minimal replica count for a given availability requirement instead of improving availability. DEPSKY, proposed by Bessani et al. [24], is a system that stores critical data with high availability through replication of the data on diverse clouds. Another algorithm in [9] is to provide the optimal data placement for chunks of an object across CSPs such that data availability is maximized under a given budget. It also prevents vendor lock-in through splitting a data object into multiple CSPs, but neglects how to determine the replica count. In [15], Hadji discusses the rational the number of chunks to be used to split the original data according to data center failure probabilities, number of replicas of each chunk, and expected data availability.

Erasure Coding. Mu et al. [25] introduce μLibCloud, a client-side library based on Apache libCloud to improve the data availability through erasure coding. But it cannot give any optimization model to provide an optimal data placement. In [13] and [14], the authors also choose erasure coding to enhance data availability and avoid vendor lock-in, but fail to provide the specific mathematical expressions. Zhang et al. [17] use erasure coding to store a data object and calculate data availability. Yet it is only a constraint in the optimization model. Similarly, Wang et al. [27] optimize the data availability through erasure coding.

2.3 Cost-Availability Trade-Off

The studies in the above two categories only minimize the monetary cost or maximize data availability based on some QoS metrics, but fail to optimize both at the same time.

Singh et al. [26] propose a secured cost-effective multi-cloud storage model to minimize the total cost of storing data, while maximizing QoS, but give many unreasonable assumptions and use cost minus QoS as the final optimization goal. Its experimental results are not convincing. Wang et al. [27] propose an ant colony algorithm-based approach to minimize the monetary costs and maximize data availability. However, for simplicity, the authors use the weights to calculate the integrated QoS value, which is the final optimization goal. This is not a true multi-objective optimization. Su et al. [28] propose a systematic model to formally formulate data placement in multi-cloud storage by using erasure coding. It can solve the data placement under complex requirements. However, in order to solve multi-objective optimization problem, the authors adopt Euclidean distance to obtain the best solution, in which the optimization weight for each objective is subjectively determined.

3 MOTIVATION

3.1 Cloud Storage Scenario

There are a large number of cloud service providers now providing storage services, and we select five most popular CSPs to obtain their pricing: Amazon S3, Microsoft Azure Cloud Storage, Alibaba Cloud Object Storage, Google Cloud Storage, and Century Cloud Object Storage, as shown in Table 2. We can see that there is heterogeneousness in the price of the same functional storage service provided by the same CSP in different regions. For example, Eastern Australia Microsoft Azure Cloud Storage has lower storage price but higher out-bandwidth price than that in Eastern USA and Northern Europe. The price models of the same functional storage service across CSPs are different. For instance, Amazon S3 in Oregon, USA, has lower storage price but higher GET request price than CenturyLink Cloud in Eastern USA.

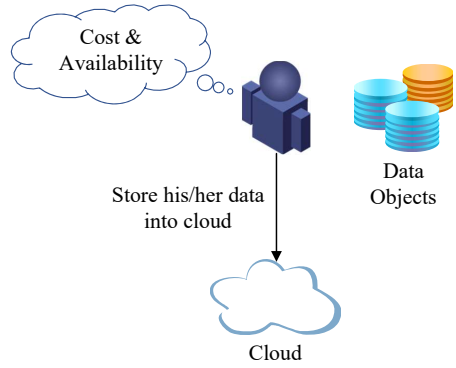


Figure 1. Cloud storage scenario

Today, more and more users are hosting their data into cloud to reduce the maintenance cost and improve data availability. Here, for clarity and conciseness, we abstract and simplify scenarios that users put their data into the cloud, as shown in Figure 1. It depicts that users store their data objects in the cloud with a series of requirement, which includes data access frequency, low monetary cost, high availability, and so on. Since users' data may contain common files, the user demands also include the data access frequency (DAF) to the data, that is, the number of times the data is retrieved within a unit period.

For example, users need to store 200 G files in the cloud, and require data availability not less than 99.99 %, and retrieve their data 0.3 times during a month. However, facing the complex cloud market, this user may choose a CSP with a lower storage cost and the availability greater than 99.99 %, i.e., Amazon S3 Paris. However, this choice is subject to higher out-bandwidth price than other CSPs when users retrieve their data, and also has risks like vendor lock-in, data unavailability, data privacy leakage, and so on.

3.2 Discussions

From the scenario, reliance on a single cloud has the abovementioned risks. Multi-cloud storage can mitigate these risks through distributing user data across multiple CSPs. Then, we discuss in detail the benefits if users in the above scenario put their data into multi-cloud.

Achieving high data availability. In the above scenario, users require data availability not less than 99.99 %. The availability of SLA in many CSPs is much greater than this value. However, it is common to hear that some well-known CSPs have experienced crash down at their data centers. As mentioned in the introduction, erasure coding and replication are two common data dispersion schemes in multi-cloud storage. Although replication can achieve higher availability than erasure coding, it also generates high storage cost. So in our paper,

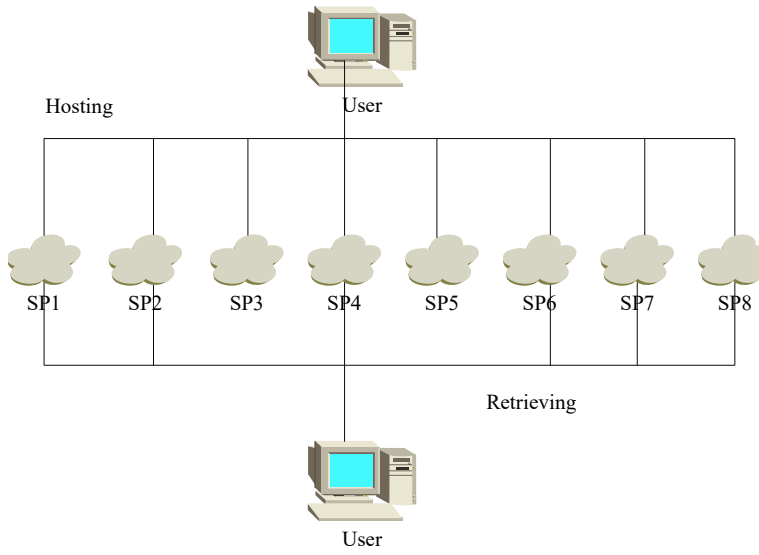


Figure 2. Erasure coding $(6, 8)$

we use erasure coding to improve data availability. As shown in Figure 2, users cannot tolerate more than 2 CSPs crash at the same time. We assume that the availability of CSPs in $(6, 8)$-erasure coding is to 99.99 %. According to the availability calculation method using erasure coding to propose in next section, the overall availability is 99.99997 %, which is larger than 99.99 % that is the availability of a single cloud.

Lowering data retrieving cost and avoiding vendor lock-in. The data retrieving cost consists of GET request cost and out-bandwidth cost. Due to the use of erasure coding, users can retrieve their data through the lowest request price and out-bandwidth price CSPs. For instance, assume that the above user puts their data into Amazon S3 in Oregon because of the low storage price, and out-bandwidth cost is $ 3. If the user puts their data into multi-cloud with a $(2, 3)$-erasure coding and CSPs are at Amazon S3 in Oregon, Azure in Eastern USA and Northern Europe, the data access can be satisfied by Azure in Eastern USA and Northern Europe and the out-bandwidth cost is $ 1.2. Apart from this, the most important phenomenon in the vendor lock-in is the high bandwidth costs brought by data migration when users face the bankruptcy of CSP or the emergence of a CSP with lower price and high availability or price hike of CSP. When such conditions emerge, users only need to pay for part of the entire data migration cost but are no longer subject to vendors.

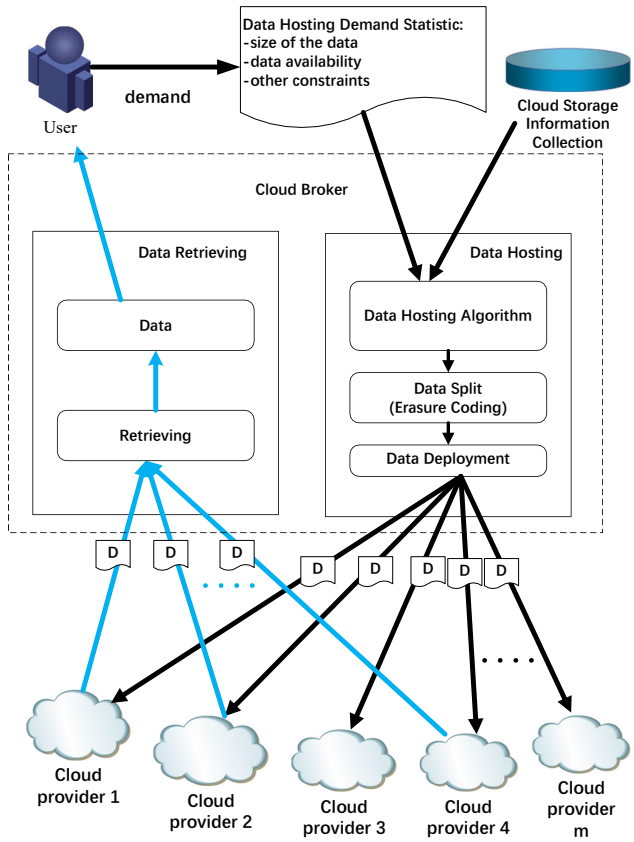


Figure 3. Multi-cloud storage framework

Protecting data privacy. As a result of erasure coding, each CSP only stores a chunk of user's original data object. Even if the cloud provider has malicious insiders or suffers external attacks, the attacker cannot recover user's original data object with a data chunk. To a certain extent, multi-cloud storage with erasure coding guarantees the privacy of user data.

Although multi-cloud storage has the abovementioned benefits, the trade-off between storage cost and retrieval cost and data availability brings a considerable challenge. Since high-availability CSPs impose enormous storage cost and retrieval cost on the user, it is a critical problem. Its solution answers how to choose suitable cloud storage providers and erasure coding parameters so as to minimize storage and retrieval cost while maximizing data availability.

CSP	Amazon S3			Microsoft Azure Cloud Storage			Alibaba Cloud Object Storage			CenturyLink Cloud		Google Cloud Storage
	Oregon	Seoul	Paris	USA East	Europe North	Australia East	China	USA West	Australia	USA	USA East	Asia Pacific
Storage price	0.0125	0.018	0.0131	0.0208	0.022	0.02	0.0226	0.02	0.0209	0.04	0.14	0.026
Out-bandwidth price	0.05	0.108	0.05	0.02	0.02	0.12	0.117	0.076	0.13	0.05	0.06	0.2
Get request price	0.004	0.0035	0.0042	0.004	0.0044	0.004	0.001	0.001	0.002	0.0	0.0	0.004

Table 2. Pricing of storage (in $/GB/month), out-bandwidth (in $/GB, and GET requests (in $/10K) of each CSP

4 SYSTEM MODEL AND PROBLEM DEFINITION

In this section, we briefly discuss problem statement, and then based on that, we formulate a data management model. Afterwards, we define a multi-objective optimization problem based on data management formulation.

4.1 Problem Statement

Figure 3 shows a scenario of how users put their data into multi-cloud storage. There are four components: *User Demand Statistic*, *Cloud Storage Information Collection*, *Data Retrieving* and *Hosting*. *Data Demand Statistic* collects user needs, which includes data size, required availability of data, and data access frequency. *Cloud Storage Information Collection* is used to collect the information of cloud providers from *CloudHarmony*, which is a third party website for collecting and monitoring cloud service information including the charges of services, attributes, services status, and so on.

Data Hosting and *Data Retrieving* are two core components in the framework. *Data Hosting* determines clouds in which the data should be deployed. *Data Retrieving* decides the clouds where the data of a user is to be retrieved from. These two components rely on the *erasure coding* which has been widely used in storage systems to provide high availability[17]. With the aid of (m, n)-erasure coding, the data object can be divided into m equal size chunks, and $(n - m)$ chunks can be encoded through m data chunks. The key property of *erasure coding* is that the original data can be recovered from any m data chunks [21]. In Figure 2, data is splitted and stored by $(6, 8)$-erasure coding, where any 6 of the 8 CSPs' data chunks can be used to recover the original data.

The primary objective of the above scenario is the optimization of data placement based on user needs, which is to choose CSPs and erasure coding parameters.

4.2 Problem Definition

To well describe a data management model, we introduce the following definitions. The symbols used in this article are listed in Table 1.

Definition 1 (Cloud Service Provider)**.** The data management model is represented as a set of independent cloud service providers $C = \{SP_1, SP_2, \ldots, SP_N\}$ where

each cloud service provider supplies the storage service. Each CSP has tuple: $CSP = \{P_{si}, P_{bi}, P_{oi}, a_i\}$, where:

1. P_{si} denotes the storage cost per unit size in CSP i;
2. P_{bi} is the out-bandwidth cost per unit size in CSP i;
3. P_{oi} defines the cost for GET requests in CSP i; and
4. a_i represents the probability of CSP i being available (i.e. availability).

Definition 2 (Data File)**.** We assume that a data file is related with a triples: $DF = \{S, \tau, A_{req}\}$, where:

1. S is the size of a data file that user stores;
2. τ denotes user data access frequency, which is equal the data access count during a time period; and
3. A_{req} defines user's required data file availability.

The objective is to choose CSP and erasure coding parameters (m, n) such that the total cost including storage and GET costs for data as well as the network cost is minimized; while the data availability is maximized. For simplicity, we assume that each CSP only stores one data chunk. It is worth noting that the following definition for availability and cost are similar to that in [17, 28], which is a universal way to define them for data hosting in erasure coding mode.

Definition 3 (Erasure Coding Parameters)**.** An (m, n)-erasure coding divides a data file into m equal size chunks, and encodes the m chunks into n ($n \geq m$) chunks which contain the m original equalized chunks and the $(n - m)$ parity chunks. Users can tolerate any $0 \sim (n - m)$ clouds simultaneously shut down.

Definition 4 (Data Availability)**.** Based on erasure coding, data availability is the sum of all cases that k CSPs are simultaneously available, where $k \in [m, n]$. This depends on the fact that outage occurrences are independent among CSPs [29]. We define $C' = \{SP_1 \times \mu_1, SP_2 \times \mu_2, \ldots, SP_N \times \mu_N\}$ ($|C'| = n$) as the service list of the n block choices, where $\{\mu_i \in \{0, 1\} \,|i = 1, 2, \ldots, N\}$, and μ_i is used to mark whether the i^{th} SP is chosen. $\Omega = \binom{|C'|}{k}$ means the number of cases that k cloud service providers are available, S_j^{Ω} denotes the j^{th} cloud services collection in Ω cases. The availability of the data file, denoted as A, can be calculated as follows:

$$A = \sum_{k=m}^{n} \sum_{j=1}^{\Omega} \left[\prod_{i \in S_j^{\Omega}} a_i \prod_{i \in C' \setminus S_j^{\Omega}} (1 - a_i) \right] \tag{1}$$

where $C' \setminus S_j^{\Omega}$ represents the CSPs that are not in S_j^{Ω}.

Definition 5 (Storage Cost)**.** The storage cost of a data file is equal to the storage cost of all data chunks in n CSPs. Since each CSP stores the data chunk of size

S/m, it can be defined as follows:

$$P_{stor} = \sum_{i \in C'} \frac{S}{m} P_{si}. \tag{2}$$

In fact, some CSPs use a tiered pricing scheme for storage. Taking AWS S3 in USA East as an example, the storage price is \$ 0.023 if the data size is less than 50 TB, and when the data size is between 50 TB and 450 TB, the storage price is \$ 0.022 [30]. Since we adopt erasure coding to divide data object in this work, the size of each data chunk is not too large. However, in the case that the data size is very large, we can use the threshold of each tier to calculate the storage cost, which is similar to the piecewise functions.

Definition 6 (Network Cost)**.** Due to erasure coding, users can retrieve the data file through any m data chunks from n clouds. In order to minimize the total network cost, we choose the m-cheapest clouds for data retrieving. It can be solved as follows:

$$P_{net} = \min_{j \in [1,\Omega]} \left(\sum_{i \in S_j^{\Omega}} \frac{S}{m} \tau_t P_{bi} \right). \tag{3}$$

Definition 7 (Operation Cost)**.** The operation cost is the cost of users' GET requests for retrieving the data file from the cheapest m CSPs. Thus, it can be calculated as follows:

$$P_{op} = \min_{j \in [1,\Omega]} \left(\sum_{i \in S_j^{\Omega}} \tau_t P_{oi} \right). \tag{4}$$

It is worth noting that the value of j in Equation (3) is equal with that in Equation (4).

Definition 8 (Total Cost)**.** The total cost of a data file C_T is the sum of *storage cost*, *operation cost*, and *network cost* and is defined as follows:

$$C_T = P_{stor} + P_{net} + P_{op}. \tag{5}$$

4.3 Optimization Problem

Given a data management model, we formalize a data placement optimization problem. It aims to maximize the availability of a data file and minimize the total cost. The overall optimization problem can be defined as follows:

$$\begin{cases} Maximize\ A, \\ Minimize\ C_T. \end{cases} \tag{6}$$

Subject to:

$$A \geq A_{req}.$$

In the above optimization problem, constraint 1) guarantees that the availability of a data file is not less than a user's required availability.

5 SOLUTION

In this section, we present a multi-objective optimization algorithm based on NSGA-II [31] to solve the multi-objective optimization problem defined in the previous section firstly. Then we use the entropy weight method to determine the weights of cost and availability, and find the most suitable data placement solution for users in the Pareto-optimal set.

5.1 Multi-Objective Optimization Algorithm

The proposed algorithm used to solve the optimization problem formulated above is mainly based on NSGA-II [31]. NSGA-II algorithm is one of the most popular multi-objective optimization algorithms. [2]. It has the advantages of fast running speed and good convergence of the solution set. Compared with NSGA, the previous generation algorithm, it uses a fast non-dominated sorting algorithm, which greatly reduces the computational complexity. The introduction of the elite strategy ensures that the individuals of the excellent population are not discarded in the iterative process, which can improve the accuracy of the optimization results. By using congestion degree, we not only overcome the defect of artificially specifying shared parameters, but also can take the congestion degree as the comparison standard among individuals in the population, so that the individuals can be evenly extended to the whole Pareto domain, which will ensure the diversity of the population. As mentioned before, how to place a user's data cost-effectively with high availability in multi-cloud environments is a hot and challenging multi-objective optimization problem. So we propose an NSGA-II-based algorithm NDP to solve this problem, which has been fully defined in Section 4. The NDP algorithm depicted in the pseudocode **Algorithm NDP** includes population initialization, individual fitness calculation, genetic operators (i.e. selection, crossover, and mutation), non-dominated sorting approach, and making new population based on an elitism approach.

Initialize Population. The first step of **NDP** is to generate an initial population. In our problem, the combination of CSPs is converted to the individual in the population. It is encoded in a binary array $[x_1, x_2, \ldots, x_N]$, where i^{th} CSP is chosen if $x_i = 1$. Specific to this optimization problem, each gene represents a CSP and the number of the genes whose value equals 1 is n (i.e. erasure coding parameter (m, n)). The algorithm **GenInd** presents how to generate an individual of the population. Firstly, it generates an integer array of length n randomly, whose elements are integers between 0–35 (lines 3–13). Then, the

Algorithm 1

Algorithm NDP: Getting the Pareto-optimal set

Require: The set of CSPs, C, the upper limit of n, ξ, and a user request $DF = \{S, \tau, A_{required}\}$

Ensure: A series of Pareto-optimal set, P

Initialize the parameters, N_p (population size), N_g (number of generation), p_c (cross probability), p_m (mutation probability);
Initialize P as empty;
Initialize population through $PopInitEQ$ or $PopInitDC$;
$g = 0$;
$Q_0 = Cpoy(P_0)$;
while $g \leq N_g$ **do**
  Cross the population P_g;
  Mutate the population P_g;
  for $i = 0$ to N_p **do**
    $fitness = \{0, 0\}$
    Calculate total cost (*totalCost*) and data file availability (*availability*) of data placement scheme represented by individual $P_g[i]$;
    $fitness = \{totalCost, availability\}$
    $P_g[i] = fitness$
  end for
  $F = Nondominated(P_g \cup Q_g)$;
  Calculate the crowding distance for each Pareto set in F;
  $P_{g+1} = []$;
  for $i = 0$ to $(|F| - 1)$ **do**
    if $|P_{g+1}| + |F_i| \leq N_p$ **then**
      $P_{g+1} = P_{g+1} \cup F_i$;
    else
      $P_{g+1} = P_{g+1} \cup F_i\,[1 : (N_p - |P_{g+1}|)]$
    end if
  end for
  $Q_{g+1} = Copy(P_{g+1})$
  $g = g + 1$;
end while
$paretoFront = []$;
for $i = 0$ to N_p **do**
  if $Q_g\,[i]\,.rank == 1$ **then**
    $paretoFront = paretoFront \cup Q_g\,[i]$;
  end if
end for
$P = P \cup paretoFront$;
return P;

Algorithm 2

Algorithm GenInd: Generating an individual

Require: The erasure coding parameter, m, n, the number of CSP, *length*, and the population size, N_p

Ensure: An individual, P_0

```
1: Initialize empty arrays index[n], P_0;
2: i = 0, j = 0;
3: for i = 0 to n do
4:    index[i] = Generate an integer (0 − length) randomly;
5:    for j = 0 to i do
6:       if index[i] == index[j] then
7:          break;
8:       end if
9:    end for
10:   if j == i then
11:      i++;
12:   end if
13: end for
14: for i = 0 to n do
15:   P_0[index[i]] = 1;
16: end for
17: return P_0;
```

corresponding position of the array representing the individual is modified to 1 (lines 14–16). In this optimization problem, the optimal data placement solution includes not only a list of CSPs but also erasure coding parameters. Based on the characteristic of our optimization problem, we propose two strategies for initializing population, as follows:

1. The idea of this strategy called **EQ** is to initialize an equal number of individuals for each erasure coding parameter. The procedure **PopInitEQ** is the pseudo code for this strategy. The individuals corresponding to all erasure coding parameters make up the entire population.
2. The second strategy called **DC** is inspired by the "divide-and-conquer" idea. The procedure **PopInitDC** is the pseudo code for this strategy. We run multiple **NDP** for different erasure coding parameter and the population belongs to a parameter in each run. Finally, we choose the best one from the results of all erasure coding parameters as the final data placement solution.

Crossover Operation and Mutation Operation. A crossover operation is used to generate new individuals through single-point or multi-point intersection. In our paper, we use a single-point crossover operator. Firstly, the algorithm pairs individuals in the population randomly. Then, it generates a point randomly and

two individuals exchange part of their genes at the mating point with crossover probability. The mutation operation is to avoid premature convergence of the population during the later iterations of the algorithm. In our paper, due to the binary encoding, the algorithm first generates a point randomly. Then, the individual's value at this point is modified to 1 if it is 0, and vice versa.

Calculate Fitness. In a genetic algorithm, calculating the fitness of individuals in the population is one of the important steps. In terms of our problem, each individual's fitness is a two-dimensional array, which contains cost and data availability of a data placement scheme represented by this individual. We first transform the individual into its corresponding data placement scheme. Then, data file availability and total cost are calculated through Equations (1), (2), (3), (4) and (5).

Make New Population. In order to maintain population distribution and diversity, the algorithm **NDP** first merges two generations of populations and the non-dominated set is constructed through a fast non-dominated sorting approach. An individual is a non-dominated one when no individual in the population is superior to this individual in all objective functions. These non-dominated individuals constitute a non-dominated set. Then, it calculates the crowding distance for each pareto set in a non-dominated set and sorts it in descending order. Finally, the algorithm selects individuals into new populations in turn from a non-dominant set.

When iterations end, **NDP** facilitates all individuals and selects individuals with a pareto rank of 1 to compose the optimal data hosting solutions.

Algorithm 3

Algorithm PopInitEQ: initializing the population according the first strategy

Require: The population size, N_p
Ensure: The Population, *Population*

```
1: Population = [];
2: for n = 2 to ξ do
3:    for m = 1 to n do
4:       for i = 1 to N_p do
5:          Population[i] = GenInd;
6:       end for
7:    end for
8: end for
9: return Population;
```

Algorithm 4

Algorithm PopInitDC: initializing the population according the second strategy

Require: The population size, N_p
Ensure: The Population, *Population*

```
1: Population = [];
2: for i = 1 to N_p do
3:     Population[i] = GenInd;
4: end for
5: return Population;
```

5.2 Approach to Determine the Most Suitable Solution

We get a Pareto-optimal set solution through **NDP**. For users who have very specific preferences and the ability to make choice, they can choose the data placement solution directly from the Pareto-optimal set. For example, one user wants to choose the solution with highest availability from the Pareto-optimal set, and would rather pay more cost. Another user wants to choose the solution with lowest cost, and accept a lower availability. However, most of users are still confused and to choose a solution from the Pareto-optimal set is difficult for them. In fact, regarding cloud storage, most users tend to choose the solutions that are more compromised on each metric, especially for cost and availability. However, there are many extreme solutions in the resulted Pareto-optimal set. For example, there always exist such solutions in the Pareto-optimal set: A [99.9999 %, $ 10] and B [99.1 %, $ 3], respectively. Although A has higher availability, its cost is also more expensive, and solution B is the opposite. Therefore, in order to recommend suitable data placement solutions for the users who cannot make choice from the Pareto-optimal set directly, the entropy based method is proposed. We calculate the QoS of each solution in the resulted Pareto-optimal set by determining the weights of the two metrics of cost and availability, and recommend the solution with the maximum QoS to the user.

There are many ways to determine the weights, which can be classified as subjective and objective weighting methods [32]. The former determine weight methods are based on the subjective value judgment of indices, including Delphi method, Analytic Hierarchy Process (AHP) method, least square method, and binomial coefficient method. The objective methods are based on the objective information (e.g. decision matrix), which includes principal component analysis, entropy method, deviation and mean square method, and multiple objective programming model [32].

Since the subjective weighting methods have strong subjective randomness and poor objectivity, in our paper, we use the entropy method to determine the weights of cost and availability, which is one of the most common objective methods. In the information theory, entropy is a measure of uncertainty [33]. The greater the amount of information, the less uncertainty and the smaller entropy, and vice versa. According to the characteristics of entropy, we can use it to judge the degree of

dispersion of an index. The greater the entropy, the higher the degree of dispersion of the index, and the greater the influence of the index on comprehensive evaluation.

Before introducing the application of an entropy method in our paper, we present the definition of the element in Pareto-optimal set:

Definition 9 (The Element in Pareto-Optimal Set)**.** The Pareto-optimal set is obtained through **NDP**, and it is represented as $P = \{P_1, P_2, \cdots, P_N\}$. Each element of Pareto-optimal set has triples: $P_i = \{P_i^{ep}, P_i^c, P_i^a\}$, where:

1. P_i^{ep} is the erasure parameter of the i^{th} element in Pareto-optimal set;
2. P_i^c denotes the i^{th} data placement solution's cost; and
3. P_i^a defines the i^{th} data placement solution's availability.

There are many studies using entropy to calculate weights [34, 35, 36], and the process of calculating weights based on entropy is very mature. In our paper, we calculate the weights of cost and availability through the following steps:

Step 1. Normalize cost and availability.

Owing to the fact that the availability (resp. cost) index is a positive (resp. negative) index, we use different normalization functions for them:

$$f_1\left(P_i^c\right) = \begin{cases} \frac{P_{max}^c - P_i^c}{P_{max}^c - P_{min}^c}, & \text{if } P_{max}^c \neq P_{min}^c, \\ 1, & \text{if } P_{max}^c = P_{min}^c, \end{cases} \tag{7}$$

where P_{max}^c (resp. P_{min}^c) means the maximum (resp. minimum) cost of all solutions in Pareto-optimal set P.

$$f_2\left(P_i^a\right) = \begin{cases} \frac{P_i^a - P_{min}^a}{P_{max}^a - P_{min}^a}, & \text{if } P_{max}^a \neq P_{min}^a, \\ 1, & \text{if } P_{max}^a = P_{min}^a, \end{cases} \tag{8}$$

where P_{max}^a (resp. P_{min}^a) means the maximum (resp. minimum) availability of all solutions in Pareto-optimal set P.

For simplicity, we combine the normalized cost and availability into an $N \times 2$ matrix A, and A_{ij} denotes i^{th} element's j^{th} index's value in the Pareto-optimal set, where $i \in \{1, 2, \cdots N\}$ and $j \in \{1, 2\}$.

Step 2. Calculate the proportion of the j^{th} index of the i^{th} element to this index as:

$$p_{ij} = \frac{A_{ij}}{\sum_{i=1}^{N} A_{ij}}. \tag{9}$$

Step 3. Calculate the entropy value of the j^{th} index as follows:

$$e_j = -k \sum_{i=1}^{N} p_{ij} \ln(p_{ij}) \tag{10}$$

where $k = \frac{1}{\ln(N)}$.

Step 4. Calculate the divergence of entropy as follows:

$$d_j = 1 - e_j. \tag{11}$$

Step 5. Calculate the weight of each index through:

$$w_j = \frac{d_j}{\sum_{j=1}^{2} d_j}. \tag{12}$$

Step 6. Calculate the integrated QoS value of each data placement solution according to the weights which are defined by step 5, as follows:

$$q_i = \sum_{j=1}^{2} w_j p_{ij}. \tag{13}$$

5.3 Discussion

In this section, we propose a solution to provide a data placement with low monetary cost and high availability for users. High data availability and low monetary cost are the two most important driving forces for users to host their data into cloud instead of the traditional storage mode. In fact, there are many metrics needed to be considered in cloud storage, such as availability, monetary cost, durability, data lock-in level, latency, and security. The proposed method can be easily extended to consider these metrics. Taking latency as an example, whether it is an optimization objective or a constraint, the algorithm **NDP** based on NSGA-II can well solve the optimization problem. Once the Pareto-optimal set is obtained, the entropy method can determine the most suitable data placement for the user who cannot make choice from the Pareto-optimal set.

6 PERFORMANCE EVALUATION

We implement the proposed data placement algorithms and carry out simulations by using real-world CSP information to evaluate its performance. In this section the goal is threefold. The first is to discuss the experimental setup in terms of CSP information dataset and parameter settings of algorithms. Second, to study the performance of the proposed algorithms through several scenarios. Although multi-cloud has become a research hotspot in recent years, there are not many studies on data storage optimization in multi-cloud environments. CHARM [17] and ACO [27] are two representative ones, which are the closest to this work. Thus, we compare the proposed method to them in this section.

Provider	Location	Specific Location
Amazon S3 (AM), Microsoft Azure (AZ), Google (GO), Alibaba (AL), CenturyLink (CL), and SoftLayer (SL)	USA (US), Europe (EU), Asia Pacific (AP), Australia (AU)	South (S), North (N), West (W), East (E), center (C), Mumbai (M), Seoul (S), Tokyo (T), Frankfurt (F), Ireland (I), Paris (P), London (L), Sydney (Sy), and so on.

Table 3. Element in CSP name

6.1 Experimental Setup

The real-world cloud providers' information is collected from *CloudHarmony* [12], which is a third-party platform to simplify the comparison of cloud services by providing reliable and objective performance analysis, reports, commentary, metrics, and tools. We use 35 CSPs in the experiments and among these, including 12 by Amazon S3 (AM), 4 by Microsoft Azure (AZ), 3 by Google (GO), 7 by Alibaba (AL), 5 by CenturyLink (CL), and 4 SoftLayer (SL). Each CSP has a name that consists of the element in Table 3 [4]. For example, the CSP with name AZ-EUN refers to the cloud provider of Microsoft Azure in the North of Europe. In our dataset, it is noted that Amazon S3 has two data centers in USA-West (i.e. Northern California (N), Oregon (O)) and USA-East (i.e., N. Virginia (N), Ohio (O)) region, respectively. For instance, AWS-USW-N denotes that the CSP of Amazon S3 is in Northern California in Eastern USA. Each CSP is also referred by a specification that consists of storage, out-going bandwidth and operation (i.e., GET requests) prices. Since the availability of SLA for each cloud provider is just what they claim, we also simulate the values of availability of each CSP in the interval of [95.0 %, 99.9 %].

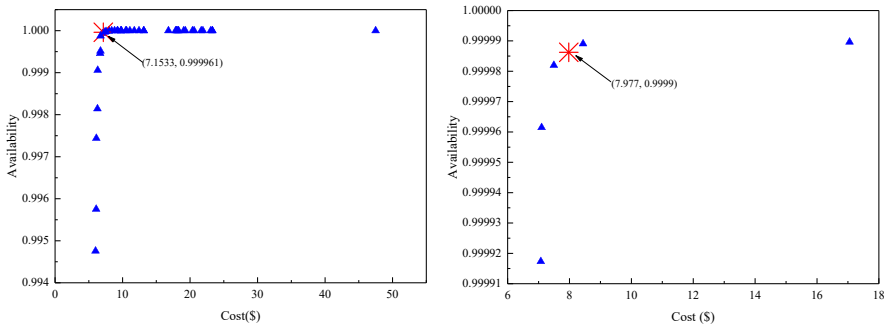
The programs for the proposed algorithm are coded in the Java language and run on an Intel Core^{TM} i7-6700 processor with 3.40 GHz CPU and 16 GB RAM. The settings for various parameters have a direct influence on the algorithm performance. Appropriate parameter values are determined by multiple experiments. Since we have two strategies for initializing the population, there are two final parameter settings, as shown in Table 4. The algorithm based on the first strategy is called **EQ**, the other is **DC**.

Algorithms	EQ	DC
Population Size	1 500	300
Generation Count	3 000	600
Mutation Rate	0.1	
Crossover Rate	0.9	

Table 4. The parameters of algorithms **EQ** and **DC**

Parameters Setting	Default	Range
Data size	200 GB	100–1 000 GB
DAF	0.3	0.0–1.0
Erasure coding (m, n)	$2 < n < 7, 0 < m < n$	

Table 5. Settings of simulation parameters



a) **EQ** b) **DC** with $(3, 5)$-erasure coding

Figure 4. Pareto-optimal set of two algorithms with data size 200 GB and DAF 0.3

6.2 Performance of the Proposed Algorithm

Before describing the performance of the proposed method, we first discuss the correctness of the model in this paper. In multi-cloud storage, providing a cost-effective and high-availability data placement for users is a research hotspot. In this paper, we first define the multi-objective optimization problem, which is to maximize data availability and to minimize the monetary cost, under the erasure coding mode in multi-cloud storage. Erasure coding is used to reduce storage cost and to improve availability, as compared to data replication. The definitions for data availability and cost are similar to that in [17, 28], which has become a common way to define them for data hosting in erasure coding mode. Then, in order to solve the multi-objective optimization problem, we propose a method based on NSGA-II. This algorithm is widely used to solve multi-objective optimization problems and can achieve good results. Since the resulted Pareto-optimal set usually contains many solutions, which makes users still confused and difficult to make choices, we adopt the entropy method to determine the most suitable solution for user from this set. The entropy method is a common method to objectively determine weight of each index based on the characteristics of the solution space. From the final results, the data placement solutions obtained by the entropy method can satisfy the user's requirement for compromise on all objectives. Furthermore, we compare our model with two representative studies. All the results show the effectiveness of our proposed model.

Due to different strategies for initializing the population, the results of **EQ** and **DC** are different. So we discuss the results gained by them in the following aspects.

6.2.1 Pareto Optimal Solution

The data placement problem is a dual-objective optimization problem in our paper. In general, there is no absolute or unique optimal solution in multi-objective problems. In this section, we study the Pareto optimal solutions of the proposed algorithms for our dataset with the default parameters in Table 5, as shown in Figure 4. Figure 4 a) shows the result of **EQ**. Since the algorithm **DC** separately solves Pareto optimal solutions under different erasure coding parameters, there are 15 Pareto optimal solutions. Due to the space limitation, we only depict the Pareto optimal solution under the $(3, 5)$-erasure coding, as shown in Figure 4 b).

An optimal data placement solution can be obtained through the method in Section 5.2. For **EQ**, this method can find the most suitable solution with maximum QoS value in Pareto optimal solution, which is marked in Figure 4 a). The data placement scheme represented by this point contains the chosen CSPs {AZ-USAE, AZ-EUN, AWS-USE-O, AWS-USW-O, AWS-EU-I} and erasure coding $(3, 5)$. The total cost and availability of this placement are \$ 7.1533 and 99.9961 % respectively. Each CSP in the result stores the size of 40 GB data, and 3 CSPs with the cheapest out-bandwidth price for GET requests.

For **DC**, the point marked in Figure 4 b) only represents the best solution under the $(3, 5)$-erasure coding. Intuitively, we have 15 solutions under this. We need to run the method in Section 5.2 again to gain the most suitable data placement scheme and results for all erasure coding parameters. The best data placement consists of:

1. the chosen CSPs {AZ-USAE, AZ-EUN, GO-AS, AWS-USE-O, AWS-EU-I, AL-USE};
2. total cost \$ 7.715 and availability 99.997 %; and
3. the $(4, 6)$-erasure coding.

6.2.2 Storage Mode Change

In fact, the DAF of a data object is time-varying in the cloud. In this section, we study the impact on data placement scheme with DAF varying from 0.0 to 1.0 with 0.05 interval. For clarity, Table 6 in Appendix summarizes the erasure coding parameters (i.e., storage mode) change with varying DAF. Whether **EQ** or **DC**, the data storage mode becomes the special erasure coding when DAF is greater than a certain value (i.e., the value of n is an integer multiple of m). For **EQ**, the storage mode is $(2, 4)$-erasure coding when DAF is greater than 0.55. For **DC**, the storage mode is $(1, 2)$-erasure coding (i.e., replication) when DAF is beyond 0.60.

The reason for this phenomenon is that high DAF requires expensive operation cost, especially network cost, and it accounts for a large proportion of the total

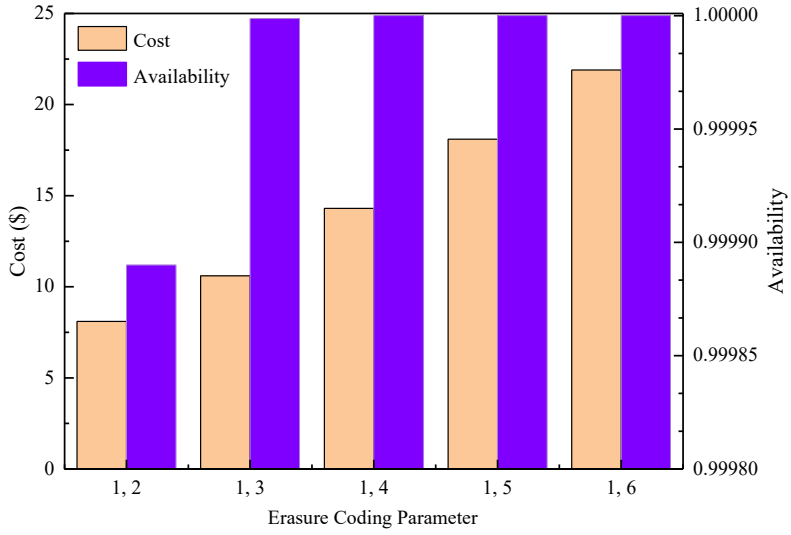
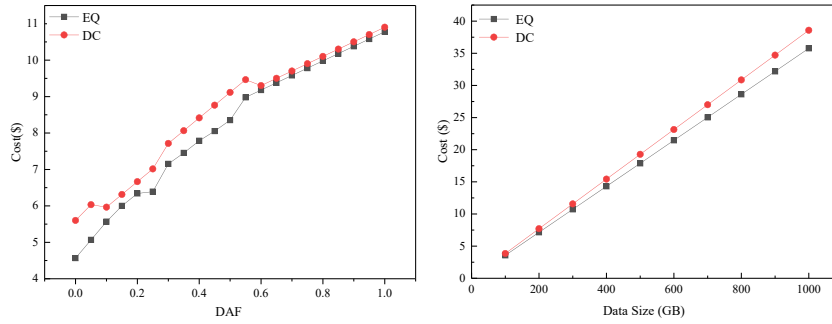


Figure 5. Cost and availability vs. erasure coding

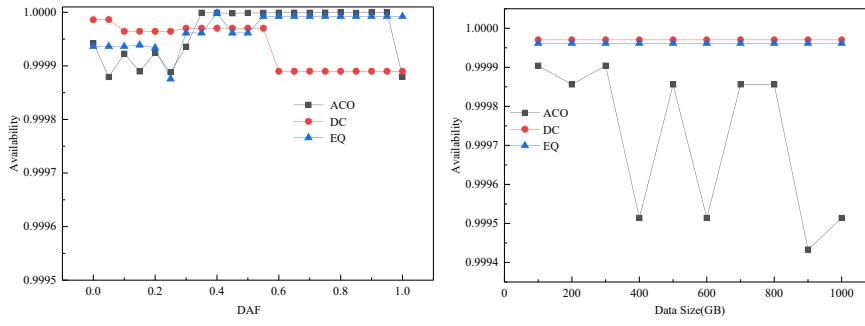
DAF	**EQ**	**DC**
0.0	$(3, 5)$	$(2, 4)$
0.05	$(3, 5)$	$(4, 5)$
0.10	$(3, 5)$	$(4, 5)$
0.15	$(3, 5)$	$(4, 5)$
0.20	$(4, 6)$	$(4, 6)$
0.25	$(4, 6)$	$(4, 6)$
0.30	$(3, 5)$	$(4, 6)$
0.35	$(3, 5)$	$(4, 6)$
0.40	$(3, 5)$	$(4, 6)$
0.45	$(3, 5)$	$(4, 6)$
0.50	$(3, 5)$	$(4, 6)$
0.55	$(2, 4)$	$(4, 6)$
0.60	$(2, 4)$	$(1, 2)$
0.65	$(2, 4)$	$(1, 2)$
0.70	$(2, 4)$	$(1, 2)$
0.75	$(2, 4)$	$(1, 2)$
0.80	$(2, 4)$	$(1, 2)$
0.85	$(2, 4)$	$(1, 2)$
0.90	$(2, 4)$	$(1, 2)$
0.95	$(2, 4)$	$(1, 2)$
1.0	$(2, 4)$	$(1, 2)$

Table 6. Erasure coding parameter (m, n) changing with varying DAF

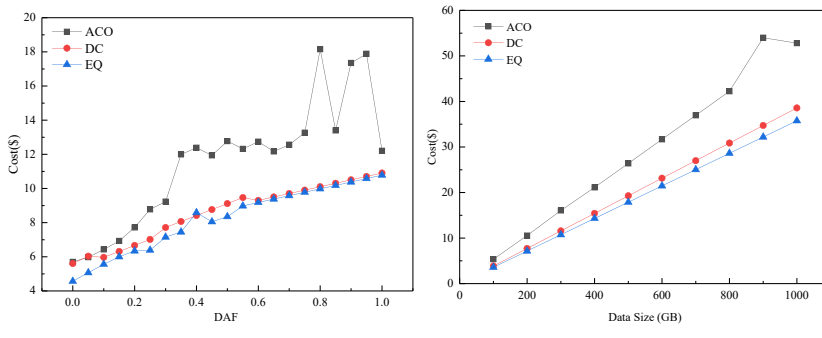


a) Cost vs. varying DAF b) Cost vs. varying data size

Figure 6. Total cost of data placement scheme of **EQ** and **DC** when the DAF and data size are varied



a) Availability vs. DAF b) Availability vs. data size



c) Total cost vs. DAF d) Total cost vs. data size

Figure 7. The total cost and availability comparison of data placement schemes from our proposed algorithms and **ACO**

cost. For example, in **EQ**, the network cost is \$ 1.8 which is 25.16 % of the total when DAF is 0.3, and the ratio increases by 6.9 % compared to the former when DAF equals 0.8. When DAF is high, the proposed algorithms explore CSPs with the cheaper out-going bandwidth price to handle high DAF. Therefore, it is really necessary to timely adjust the data placement scheme according to varying DAF. Data migration of varying DAF is beyond the scope of our paper, and we leave it as the future work.

6.2.3 Cost and Availability Performance

In this section, we evaluate the cost and availability performance of the proposed algorithms. Since each erasure coding has an optimal result for **DC**, we study the impact of erasure coding on cost and availability by varying it from $(1, 2)$ to $(1, 6)$ with data size 200 GB and DAF 0.3. As shown in Figure 5, as n increases as an erasure coding parameter, the availability gradually approaches 1. The reason is that the overall availability of a data object is equal to the probability that not more than $(n - m)$ CSPs crash at the same time. When n becomes larger, the data placement scheme can tolerate the simultaneous failure of more CSPs, and so the availability can be enhanced. At the same time, the total cost is higher with n. This is because of the storage cost of a data object growing with the number of replicas.

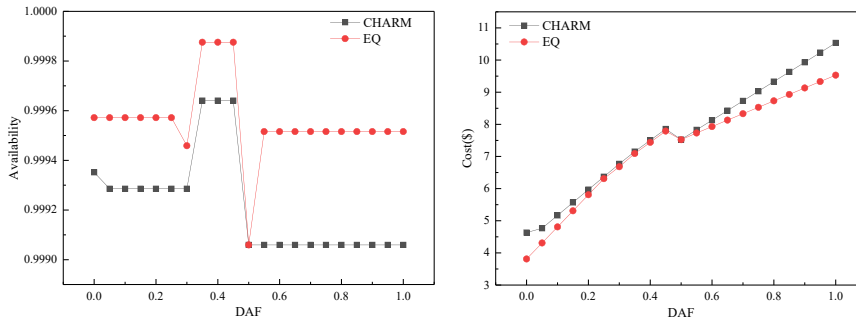
Figure 6 shows the impact of DAF and data size on the total cost of the optimal data placement scheme of the proposed algorithms. In Figure 6 a), the total cost only contains the storage cost and **EQ** can save approximately 18.6 % than **DC** when DAF is 0. It is worth noting that the polyline in Figure 6 a) is composed of several straight lines. It is because of the data placement scheme varying with DAF. For instance, the black polyline consists of four parts and the change points are 0.15, 0.30, and 0.55. This result can correspond to the change of the storage mode in Table 6 in Appendix.

We also explore the impact of data size on the total cost by varying it from 100 GB to 1 000 GB with the step size of 100 GB. As shown in Figure 6 b), the results of the proposed algorithms show the positive correlation between cost and data size. The reason why the result is a straight line is that the data placement scheme does not change as data size increases. The total cost of a resultant data placement scheme through **EQ** can save about \$ 2.8 comparing with that of **DC**.

6.3 Performance Comparison with Other Algorithms

There are many previous studies on data storage in multi-cloud environments. In this section, we compare the cost and availability performance of the proposed algorithms with two recently solutions **ACO** and **CHARM**.

The optimization objective of **ACO** contains the total cost and availability, which is the same as ours. We evaluate their performance, as shown in Figure 7. Figures 7 a) and 7 b) respectively depict the impact of DAF and data size on the availability of obtained data placement schemes. In Figure 7 a), it is obvious that



a) Availability b) Total cost

Figure 8. The availability and total cost of data placement scheme from our proposed algorithms and **CHARM** under the varying DAF

the result of **DC** is better than its two peers when DAF is less than 0.3. **EQ** can achieve higher availability than **DC** when DAF is grater than 0.55. The reason for this result is that data placement can tolerate the crash of more CSP at the same time than **DC**, which is shown in Table 6. **EQ** can tolerate the simultaneous crash of 2 CSPs, while **DC** can tolerate only one CSP's crash when DAF is greater than 0.55. Figure 7 b) shows the availability change under varying data size. It is evident that the performance of our proposed algorithms is superior to that of **ACO**. The results of the proposed algorithms are relatively more stable than those of **ACO**.

Data Size	**Availability**		**Cost**	
	CHARM	EQ	CHARM	EQ
100	0.9992856	0.9994588	3.38693	3.34
200	0.9992856	0.9994588	6.77027	6.68
300	0.9992856	0.9994588	10.1536	10.02
400	0.9992856	0.9994588	13.53693	13.36
500	0.9992856	0.9994588	16.92027	16.7
600	0.9992856	0.9994588	20.3036	20.04
700	0.9992856	0.9994588	23.68693	23.38
800	0.9992856	0.9994588	27.07027	26.72
900	0.9992856	0.9994588	30.4536	30.06
1 000	0.9992856	0.9994588	33.83693	33.4

Table 7. Availability and total cost ($) comparison of **CHARM** and **EQ** with the varying data size (GB)

We also explore the comparison of total cost by the varying DAF and data size. As shown in Figure 7 c), the proposed algorithms **EQ** and **DC** can approximately save $ 3.56 and $ 3.44, respectively, comparing to **ACO** when DAF equals 0.6. Figure 7 d) presents the total cost change by varying the data size from 100 GB to 1 000 GB with the interval of 100 GB. It is clear that the total cost of the data

placement schemes, obtained with the proposed algorithms is lower than **ACO**. Our proposed algorithms can save more than $ 10 than **ACO** when data size is 800 GB.

Finally, we compare the proposed algorithm **EQ** with **CHARM** through two scenarios including the varying DAF and data size. Since the optimization objective of **CHARM** is to minimize the total cost under the guaranteed availability, we select the data placement scheme from the Pareto solution of **EQ**, whose availability is not lesser than that of **CHARM**. Figure 8 a) depicts the availability of these two algorithms. It is obvious that the availability of **EQ** is larger than **CHARM**. Figure 8 b) presents the comparison of the total cost between them by varying DAF from 0.0 to 1.0. Our algorithm can clearly obtain the lower total cost than **CHARM**. Another scenario is to compare two algorithms through the varying data size, whose result is shown in Table 7. Our algorithm can save 1.29 % when data size is 1 000 GB. Although the advantages of our proposed algorithm are not obvious in total cost, its resulting availability of our algorithm is better than **CHARM**'s.

7 CONCLUSION

There are some risks such as vendor lock-in, low data availability and data privacy leakage if users put their data into a single cloud. Data hosting based on multi-cloud is becoming a new development trend. How to strike a trade-off between various factors and realize multi-objective optimization becomes one of the most important concerns in multi-cloud environments. So, in this paper, an architecture in multi-cloud storage is presented at first. Next, a multi-objective optimization problem is defined to minimize total cost and maximize data availability. Then, an approach based on NSGA-II is given with its goal to effectively solve a multi-objective optimization problem and obtain a set of non-dominated solutions (i.e., a list of cloud storage providers) and erasure coding parameters. Then, we use a method based on the entropy to recommend the most suitable solution for users who cannot choose one from the resulted Pareto-optimal set directly. Finally, the performance of this algorithm is examined through extensive experiments which are driven by real-world multiple cloud storage providers' information.

In the future, we intend to improve this work in two directions:

1. The SLA is important for users, and it directly affects the user experience. Thus, we will consider more SLAs in exploring a data hosting scheme, such as the overall security, latency and durability of cloud services [37].
2. Since the data placement varies with the change of user's DAF, it is necessary to propose solutions for dynamic data placement based on the varying DAF. Especially, for the absence of user's future DAF, we will predict it according to the historical data [38, 39, 40].
3. We will consider to optimize with the choice of cloud instance type [41, 42, 43].

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61602109, the DHU Distinguished Young Professor Program under Grant No. LZB2019003, the Shanghai Science and Technology Innovation Action Plan under Grant No. 19511101802, the Natural Science Foundation of Shanghai under Grant No. 19ZR1401900, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Lv, Y.—Chen, Y.—Zhang, X.—Duan, Y.—Li, N.: Social Media Based Transportation Research: The State of the Work and the Networking. IEEE/CAA Journal of Automatica Sinica, Vol. 4, 2017, No. 1, pp. 19–26, doi: 10.1109/JAS.2017.7510316.

[2] Wu, N. Q.—Li, Z. W.—Barkaoui, K.—Li, X. O.—Murata, T.—Zhou, M. C.: IoT-Based Smart and Complex Systems: A Guest Editorial Report. IEEE/CAA Journal of Automatica Sinica, Vol. 5, 2018, No. 1, pp. 69–73, doi: 10.1109/JAS.2017.7510748.

[3] Zhang, P.—Zhou, M.—Fortino, G.: Security and Trust Issues in Fog Computing: A Survey. Future Generation Computer Systems, Vol. 88, 2018, pp. 16–27, doi: 10.1016/j.future.2018.05.008.

[4] Mansouri, Y.—Buyya, R.: To Move or Not to Move: Cost Optimization in a Dual Cloud-Based Storage Architecture. Journal of Network and Computer Applications, Vol. 75, 2016, pp. 223–235, doi: 10.1016/j.jnca.2016.08.029.

[5] Ghahramani, M. H.—Zhou, M. C.—Hon, C. T.: Toward Cloud Computing QoS Architecture: Analysis of Cloud Systems and Cloud Services. IEEE/CAA Journal of Automatica Sinica, Vol. 4, 2017, No. 1, pp. 6–18, doi: 10.1109/JAS.2017.7510313.

[6] Gao, Y.—Guan, H.—Qi, Z.—Hou, Y.—Liu, L.: A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing. Journal of Computer and System Sciences, Vol. 79, 2013, No. 8, pp. 1230–1242, doi: 10.1016/j.jcss.2013.02.004.

[7] Armbrust, M.—Fox, A.—Griffith, R.—Joseph, A. D.—Katz, R.—Konwinski, A.—Lee, G.—Patterson, D.—Rabkin, A.—Stoica, I.—Zaharia, M.: A View of Cloud Computing. Communications of the ACM, Vol. 53, 2010, No. 4, pp. 50–58, doi: 10.1145/1721654.1721672.

[8] Opara-Martins, J.—Sahandi, R.—Tian, F.: Critical Analysis of Vendor Lock-In and Its Impact on Cloud Computing Migration: A Business Perspective. Journal of Cloud Computing, Vol. 5, 2016, No. 1, pp. 4–22, doi: 10.1186/s13677-016-0054-z.

[9] Mansouri, Y.—Toosi, A. N.—Buyya, R.: Brokering Algorithms for Optimizing the Availability and Cost of Cloud Storage Services. Proceedings of 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2013, pp. 581–589, doi: 10.1109/CloudCom.2013.83.

[10] Aldossary, S.—Allen, W.: Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions. International Journal of Ad-

vanced Computer Science and Applications, Vol. 7, 2016, No. 4, pp. 485–498, doi: 10.14569/IJACSA.2016.070464.

[11] Mansouri, Y.—Toosi, A. N.—Buyya, R.: Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions. ACM Computing Surveys (CSUR), Vol. 50, 2017, No. 6, Art. No. 91, 51 pp., doi: 10.1145/3136623.

[12] Cloudharmony, 2017. [Online] Available at: `http://www.cloudharmony.com`.

[13] Abu-Libdeh, H.—Princehouse, L.—Weatherspoon, H.: RACS: A Case for Cloud Storage Diversity. Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10), New York, NY, USA, 2010, pp. 229–240, doi: 10.1145/1807128.1807165.

[14] Papaioannou, T. G.—Bonvin, N.—Aberer, K.: Scalia: An Adaptive Scheme for Efficient Multi-Cloud Storage. Proceedings of the 2012 International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12), Utah, USA, 2012, 10 pp., doi: 10.1109/SC.2012.101.

[15] Hadji, M.: Scalable and Cost-Efficient Algorithms for Reliable and Distributed Cloud Storage. In: Helfert, M., Méndez Muñoz, V., Ferguson, D. (Eds.): Cloud Computing and Services Science (CLOSER 2015). Springer, Cham, Communications in Computer and Information Science, Vol. 581, 2015, pp. 15–37, doi: 10.1007/978-3-319-29582-4_2.

[16] Ma, Y.—Nandagopal, T.—Puttaswamy, K. P.—Banerjee, S.: An Ensemble of Replication and Erasure Codes for Cloud File Systems. Proceedings of 2013 INFOCOM, Turin, Italy, 2013, pp. 1276–1284, doi: 10.1109/INFCOM.2013.6566920.

[17] Zhang, Q.—Li, S.—Li, Z.—Xing, Y.—Yang, Z.—Dai, Y.: CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability. IEEE Transactions on Cloud Computing, Vol. 3, 2015, No. 3, pp. 372–386, doi: 10.1109/TCC.2015.2417534.

[18] Mansouri, Y.—Toosi, A. N.—Buyya, R.: Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers. IEEE Transactions on Cloud Computing, Vol. 7, 2019, No. 3, pp. 705–718, doi: 10.1109/TCC.2017.2659728.

[19] Wu, Z.—Butkiewicz, M.—Perkins, D.—Katz-Bassett, E.—Madhyastha, H. V.: SPANStore: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services. Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13), 2013, pp. 292–308, doi: 10.1145/2517349.2522730.

[20] Liu, G.—Shen, H.—Wang, H.: An Economical and SLO-Guaranteed Cloud Storage Service Across Multiple Cloud Service Providers. IEEE Transactions on Parallel and Distributed Systems, Vol. 28, 2017, No. 9, pp. 2440–2453, doi: 10.1109/TPDS.2017.2675422.

[21] Weatherspoon, H.—Kubiatowicz, J.: Erasure Coding vs. Replication: A Quantitative Comparison. In: Druschel, P., Kaashoek, F., Rowstron, A. (Eds.): Peer-to-Peer Systems (IPTPS 2002). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2429, 2002, pp. 328–337, doi: 10.1007/3-540-45748-8_31.

[22] Rodrigues, R.—Liskov, B.: High Availability in DHTs: Erasure Coding vs. Replication. In: Castro, M., van Renesse, R. (Eds.): Peer-to-Peer Systems IV (IPTPS 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3640, 2005, pp. 226–239, doi: 10.1007/11558989_21.

[23] Wei, Q.—Veeravalli, B.—Gong, B.—Zeng, L.—Feng, D.: CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster. Proceedings of the 2010 IEEE International Conference on Cluster Computing (CLUSTER 2010), Heraklion, Greece, 2010, pp. 188–196, doi: 10.1109/CLUSTER.2010.24.

[24] Bessani, A.—Correia, M.—Quaresma, B.—André, F.—Sousa, P.: DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. ACM Transactions on Storage (TOS), Vol. 9, 2013, No. 4, Art. No. 12, 33 pp., doi: 10.1145/2535929.

[25] Mu, S.—Chen, K.—Gao, P.—Ye, F.—Wu, Y.—Zheng, W.: μLibCloud: Providing High Available and Uniform Accessing to Multiple Cloud Storages. Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, Beijing, China, 2012, pp. 201–208, doi: 10.1109/Grid.2012.28.

[26] Singh, Y.—Kandah, F.—Zhang, W.: A Secured Cost-Effective Multi-Cloud Storage in Cloud Computing. Proceedings of the 2011 IEEE Conference on Computer Communications Workshops, Shanghai, China, 2011, pp. 619–624, doi: 10.1109/INFCOMW.2011.5928887.

[27] Wang, P.—Zhao, C.—Zhang, Z.: An Ant Colony Algorithm-Based Approach for Cost-Effective Data Hosting with High Availability in Multi-Cloud Environments. Proceedings of 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), Zhuhai, China, 2018, 6 pp., doi: 10.1109/ICNSC.2018.8361288.

[28] Su, M.—Zhang, L.—Wu, Y.—Chen, K.—Li, K.: Systematic Data Placement Optimization in Multi-Cloud Storage for Complex Requirements. IEEE Transactions on Computers, Vol. 65, 2016, No. 6, pp. 1964–1977, doi: 10.1109/TC.2015.2462821.

[29] Ford, D.—Labelle, F.—Popovici, F. I.—Stokely, M.—Truong, V.-A.—Barroso, L.—Grimes, C.—Quinlan, S.: Availability in Globally Distributed Storage Systems. Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI '10), Vancouver, BC, Canada, 2010, pp. 61–74.

[30] Amazon S3, 2018. [Online] Available at: `https://aws.amazon.com/cn/s3/pricing/?nc=sn&loc=4`.

[31] Deb, K.—Pratap, A.—Agarwal, S.—Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, Vol. 6, 2002, No. 2, pp. 182–197, doi: 10.1109/4235.996017.

[32] Ma, J.—Fan, Z. P.—Huang, L. H.: A Subjective and Objective Integrated Approach to Determine Attribute Weights. European Journal of Operational Research, Vol. 112, 1999, No. 2, pp. 397–404, doi: 10.1016/S0377-2217(98)00141-6.

[33] Shannon, C. E.: A Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, 1948, No. 3, pp. 379–423, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[34] Ma, L. H.—Zhang, Y. P.—Zhao, Z. W.: Improved VIKOR Algorithm Based on AHP and Shannon Entropy in the Selection of Thermal Power Enterprise's Coal Suppliers. 2008 International Conference on Information Management, Innovation

Management and Industrial Engineering, Taipei, Taiwan, 2008, Vol. 2, pp. 129–133, doi: 10.1109/ICIII.2008.29.

[35] Wang, T.-C.—Lee, H.-D.: Developing a Fuzzy TOPSIS Approach Based on Subjective Weights and Objective Weights. Expert Systems with Applications, Vol. 36, 2009, No. 5, pp. 8980–8985, doi: 10.1016/j.eswa.2008.11.035.

[36] Shemshadi, A.—Shirazi, H.—Toreihi, M.—Tarokh, M. J.: A Fuzzy VIKOR Method for Supplier Selection Based on Entropy Measure for Objective Weighting. Expert Systems with Applications, Vol. 38, 2011, No. 10, pp. 12160–12167, doi: 10.1016/j.eswa.2011.03.027.

[37] Xia, Y.—Zhou, M.—Luo, X.—Zhu, Q.—Li, J.—Huang, Y.: Stochastic Modeling and Quality Evaluation of Infrastructure-as-a-Service Clouds. IEEE Transactions on Automation Science and Engineering, Vol. 12, 2015, No. 1, pp. 160–172, doi: 10.1109/TASE.2013.2276477.

[38] Gao, S.—Zhou, M.—Wang, Y.—Cheng, J.—Yachi, H.—Wang, J.: Dendritic Neuron Model with Effective Learning Algorithms for Classification, Approximation and Prediction. IEEE Transactions on Neural Networks and Learning Systems, Vol. 30, 2019, No. 2, pp. 601–614, doi: 10.1109/TNNLS.2018.2846646.

[39] Li, W.—Xia, Y.—Zhou, M.—Sun, X.—Zhu, Q.: Fluctuation-Aware and Predictive Workflow Scheduling in Cost-Effective Infrastructure-as-a-Service Clouds. IEEE Access, Vol. 6, 2018, pp. 61488–61502, doi: 10.1109/ACCESS.2018.2869827.

[40] Bi, J.—Zhang, L.—Yuan, H.—Zhou, M.: Hybrid Task Prediction Based on Wavelet Decomposition and ARIMA Model in Cloud Data Center. Proceedings of 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), Zhuhai, China, 2018, 6 pp., doi: 10.1109/ICNSC.2018.8361342.

[41] Wang, P.—Zhou, W.—Zhao, C.—Lei, Y.—Zhang, Z.: A Dynamic Programming-Based Approach for Cloud Instance Types Selection and Optimization. International Journal of Information Technology and Management, Vol. 19, 2020, No. 4, doi: 10.1504/IJITM.2020.10028804.

[42] Liu, W.—Wang, P.—Meng, Y.—Zou, G.—Zhang, Z.: A Novel Algorithm for Optimizing Selection of Cloud Instance Types in Multi-Cloud Environment. 25th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 2019, pp. 167–170, doi: 10.1109/ICPADS47876.2019.00033.

[43] Liu, W.—Wang, P.—Meng, Y.—Zhao, Q.—Zhao, C.—Zhang, Z.: A Novel Model for Optimizing Selection of Cloud Instance Types. IEEE Access, Vol. 7, 2019, pp. 120508–120521, doi: 10.1109/ACCESS.2019.2937511.

Pengwei Wang received his B.Sc. and M.Sc. degrees from the Shandong University of Science and Technology, Qingdao, China, in 2005 and 2008, respectively, and his Ph.D. degree from the Tongji University, Shanghai, China, in 2013, all in computer science. He finished his postdoctoral research work at the Department of Computer Science, University of Pisa, Italy, in 2015. He is currently serving as Associate Professor in the School of Computer Science and Technology, Donghua University, Shanghai. His research interests include cloud computing, data mining, and service computing.

Caihui Zhao received his B.Sc. degree in software engineering from the Shandong University of Science and Technology, Qingdao, China, in 2016. He is currently a student studying for his Master's degree in software engineering at the Donghua University in Shanghai, China. His research interests include cloud computing, and multi-cloud storage.

Wenqiang Liu received his B.Sc. degree in computer science and technology from the Shandong University of Science and Technology, Qingdao, China, in 2017. He is currently a student studying for his Master's degree in computer science and technology at the Donghua University in Shanghai, China. His research interests include cloud computing, operations research and machine learning.

Zhen Chen received his B.Sc. degree in software engineering from the Donghua University, Shanghai, China, in 2018, where he is currently pursuing his Master's degree in computer science and technology. His current research interests include cloud computing, crowd intelligence, and machine learning.

Zhaohui Zhang received his B.Sc. degree in computer science from the Anhui Normal University, Wuhu, China, in 1994, his Master's degree from the University of Science and Technology of China, in 2000, and his Ph.D. degree in computer science from the Tongji University, Shanghai, China, in 2007. He became a teacher at the Anhui Normal University. He was Professor with the Anhui Normal University, in July 2015. He currently serves as Professor in the School of Computer Science and Technology, Donghua University, Shanghai. His research interests include network information services, service computing, and cloud computing.