QoS-Aware and Fault-Tolerant Replica Placement

Jingkun Hu[1], Zhihui Du[2(✉)], Sen Zhang[3], and David A. Bader[2(✉)]

[1] Worldmoney Blockchain Management Limited, Hong Kong, Hong Kong
kun.hu@worldmoney.org
[2] New Jersey Institute of Technology, Newark, NJ, USA
{zhihui.du,bader}@njit.edu
[3] State University of New York, College at Oneonta, Oneonta, USA
zhangs@oneonta.edu

Abstract. As emerging applications become more and more distributed and decentralized, it has become a more challenging problem to design and build fault-tolerant network systems with high Quality of Service (*QoS*) guarantee. In this paper, an optimal replica placement problem is formulated in terms of minimizing the replica placement cost subject to both *QoS* and fault-tolerant constraints. Based on the generalized graph model, the optimal replica placement problem is proved to be NP-hard. The essential properties of the proposed problem are investigated and two criteria, which can quantitatively measure the replica demand of each node and the contribution of one replica to other nodes in the graph, are proposed. The criteria are employed to develop efficient heuristic algorithms. Finally the proposed algorithms are evaluated with extensive network configurations and the experimental results show that the proposed heuristic algorithms can generate solutions very close to the optimal results.

Keywords: Replica placement · Quality of service · Fault tolerance · Heuristic algorithm · Distributed system

1 Introduction

Replica placement [10] is a critical technology that can be used for many different purposes, such as reducing latency, enhancing fault tolerance, optimizing bandwidth use, or improving scalability of network. So it has been employed in extensive applications such as Content Delivery/Distribution Network (CDN) [15], data grid [10], cloud [9] and edge computing environment [1]. The theoretical model of replica placement is named as facility location [4], K-median [2], minimum K-center [13] and so on under different application scenarios. The great variety and constant changes in application requirements are essentially part of the reasons why so many different replica placement policies have been developed and why replica placement problems remain to be an active research field. Emerging applications, such as smart city [11] and autonomous unmanned

M. Qiu (Ed.): ICA3PP 2020, LNCS 12453, pp. 157–172, 2020.
https://doi.org/10.1007/978-3-030-60239-0_11

cars [19], entail critical requirements in both Quality of Service (QoS) and fault tolerance for supporting networks behind these applications. How to provide guaranteed service in real time with optimal cost yet simultaneously meeting constraints with both fault tolerance and QoS is a challenging problem.

In this paper, we investigate a novel optimal replica placement problem where both QoS and fault-tolerant constraints must be met simultaneously, and propose heuristic algorithms to tackle the problem. The major contributions of our work are as follows.

1. A novel optimal replica placement problem with both QoS and fault-tolerant constraints has been formulated and proved to be NP-hard.
2. Two critical criteria (metrics) *Replica Demanding Priority* and *Replica Profit Ratio* have been developed to quantitatively estimate the replica demand degree and the contribution of given replica to other nodes.
3. Efficient heuristic replica placement algorithms have been developed and extensive experimental results show that the proposed algorithms can achieve results very close to optimal solutions.

2 Related Work

Sahoo et al. [15] provides a comprehensive survey of data replica in CDN systems. Aral et al. [1] summarizes the replica placement in even broader environments and applications, which could be centralized or decentralized, with complete or partial information, and static or dynamic. These surveys show that replica placement problems vary from each other to meet the varying underneath constraints. Most replica placement algorithms take QoS as their constraint or optimization object due to the importance of QoS under different scenarios. Tang et al. [16] proposed an early and typical replica placement research that presents their algorithm solution based on general problem formulation with QoS constraint. At the same time, there are many QoS related replica placement researches focus on algorithm efficiency under practical and specific application scenarios [5] instead of analyzing the problem's hardness under general cases.

There are also replica placement problems subject to constraints related to fault tolerance. Khan et al. [12] developed the replica placement techniques to guarantee a fault-tolerant performance under the uncertainties of arbitrary server failures. However, this work did not consider the QoS requirement. There are some further research [14,18] on fault-tolerant requirement.

In principal, the QoS requirement will reshape each node's local topology and fault-tolerant requirement will entail redundant resources. So the QoS and fault-tolerant constrains together will make the problem very different from existing ones. Up to date, little research has been found on optimal replica problem subject to both *QoS* and fault-tolerant constraints with locality freedom. Du et al. [6] attacks the proposed problem by proposing an approximation algorithm. Their basic idea and contribution is that based on a linear programming (LP) relaxation, the replica problem will be constructed and solved by LP solvers to achieve a real number solution in range [0,1] first. According to the LP solution,

the integer programming (IP) solution with a strict upper bound or approximation ratio will be achieved by developing the corresponding rounding algorithm. This work is different from that paper since the major contribution of this paper is developing efficient heuristic algorithms based on the proposed two criteria to obtain good performance on typical data instances but without guaranteeing that a theoretical approximation ratio will stand for any inputs.

3 Problem Description

The replicated infrastructure considered in this work consists of multiple geographically distributed servers that can be classified into two categories: the servers with a replica placement and the servers serving as proxies only. A proxy server doesn't have a replica on itself, but it can delegate requests to other replica servers in its close vicinity. Each node is associated with a certain cost for hosting a replica.

QoS requirement can be distance, response time, latency or throughput, depending on the concerned environment and application. We allow QoS to be individualized (localized) and the situation where a uniformed QoS constraint across all nodes is a naive case of our problem.

A fault-tolerant network means that it can continue to provide service in the presence of failures. Failures may happen at different levels of a network. In our work we consider failures at the replica level rather than at the node level or any other lower levels. The fault-tolerant requirement is also allowed to vary from node to node and the situation where a uniformed fault tolerance configuration on all nodes is thus also a naive case of our problem.

3.1 System Model

We model the replicated network topology using a connected undirected graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_{N-1}\}$ is the set of nodes whose cardinality is N, which can also be simply represented by the unique values from 0 to $N - 1$. And $E = \{(u, v)|u, v \in V\}$ is a set of edges over V. Each node $v \in V$ is an abstract representation of the site that can be placed with replica. An edge $e = (u, v) \in E$ represents that there is *direct* link between the two sites identified by u and v.

For a pair of nodes v and u belonging to V, we use $l(u, v)$ or $l(v, u)$ to represent the distance of the direct link between them. If there is no direct link from v to u, the value of $l(v, u)$ would be infinite and we define $l(v, u) = \infty$. Let L denote the set of weights of all these direct links. We use $d(u, v)$ to represent the shortest distance among all paths between v and u. Since the graph is connected, $d(u, v)$ always exists.

For each node v, we introduce a parameter $s(v)$ to represent the total cost for placing a replica on the node v.

Now let us consider QoS requirement for each node. We use $q(v)$ to quantitatively describe the QoS requirement of a node v. Without loss of generality,

here $q(v)$ is defined as the largest distance that is allowed for the node v to possibly get replica service from other nodes. If a node v has a replica, it can meet the request immediately; if a node v does not have a replica, it should send requests to a nearby node with a replica placed, denoted by u, that satisfies $d(v, u) \leq q(v)$. We use Q to represent all the QoS requirements of all the nodes.

Because a network is composed of N nodes, the entire replication scheme can be represented by a vector $X = <x_0, x_1, \ldots, x_{N-1}>$ where x_i gets 1 if a replica is stored at node v_i, and 0 otherwise. Let R represent the set of nodes with replicas placed. Then the total cost C of the replicated network is simply calculated by the following objective function: $C(R) = \sum_{i=0}^{N-1} x_i \times s_i$.

Furthermore, we use another parameter $m(v)$ to stand for the fault-tolerant level of node v and all nodes are represented by M. More specifically, let $mrft(v)$ represent the maximum number of replica failures the node v can tolerate, then we define $m(v) = mrft(v) + 1$. It means that in order to tolerate $m(v) - 1$ arbitrary replica failures, at least $m(v)$ replicas should be accessible for the node v. When $M = <1, ..., 1>$ for all the nodes, the network has zero failure tolerance, which can only happen in a replica-fault-free system.

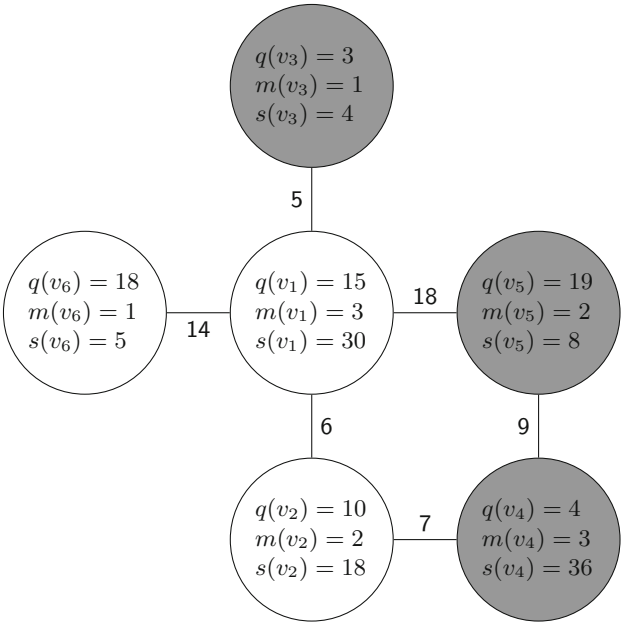


Fig. 1. An example of replica network with QoS and fault-tolerant requirements where each node v has a QoS requirement $q(v)$, fault-tolerant requirement $m(v)$ and replica storing cost $s(v)$. The numbers associated with the edges represent the distances between the incident nodes. The nodes with replica placed are colored in gray and those without white.

Figure 1 shows an example of a replicated network with QoS and fault-tolerant requirements defined for each node. We can see only node v_3 and node v_5's requirements can be met at the same time.

In summary, we have established the following concepts and notations to describe the network in the rest of the discussion. Let $G = (V, E)$ be a graph representing a replicated network, where $V = \{v_0, v_1, \ldots, v_{N-1}\}$ represents the set of nodes of the graph, and E is the set of the edges of the graph built over V. Let $L = <l_0, l_1, \ldots, l_{|E|-1}>$ be the weights associated with each $e \in E$. Let $S = <s_0, s_1, \ldots, s_{N-1}> = <s(v_0), s(v_1), \ldots, s(v_{N-1})>$ be the cost vector associated with each $v \in V$. Let $Q = <q_0, q_1, \ldots, q_{N-1}> = <q(v_0), q(v_1), \ldots, q(v_{N-1})>$ stand for the QoS requirements on V with q_i corresponding to the QoS requirement of v_i. Let $M = <m_0, m_1, \ldots, m_{N-1}> = <m(v_0), m(v_1), \ldots, m(v_{N-1})>$ stand for the fault-tolerant requirements on V with m_i corresponding to the fault-tolerant level of node v_i. R refers to a set of nodes which are placed replicas. M refers to the whole vector of fault-tolerant levels. The same convention also applies to S, Q, L and other notations. In the following discussion, all these notations will be used whenever they don't cause any ambiguity.

3.2 Optimal QoS-Aware and Fault-Tolerant Replica Placement

Given a graph $G = (V, E)$ with each node annotated by s, q and m values, and each edge annotated by l, if any replica placement solutions exist for the network, then the objective of the problem is to find a replica placement solution R whose total cost of replicas is minimized subject to the constraints of QoS and fault tolerance of every node. Here we name it as OQFRP problem. Alternatively, the proposed problem can also be modeled as an optimization problem whose objective is to minimize the total cost of the network subject to the Q and M constraints:

$$Min : C(R) = \sum_{i=0}^{N-1} x_i \times s_i, \text{subject to} \tag{1}$$

$$\sum_{d(i,j) \le q(i), \forall j \in \{0,\ldots,N-1\}} x_j \ge m_i, \forall i \in \{0, ..., N-1\}, \text{and} \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

The problem turns out to be an integer linear programming problem, which, generally being NP-hard, suggests that our problem is likely also NP-hard. In next subsection, we will provide a proof the NP-completeness of the decision version of the OQFRP problem.

3.3 Intractability of the Problem

The decision form of the OQFRP, denoted by $DOQFRP$, asks whether or not a network has a replica solution that has the cost less or equal to a specified value

k:

$$\{<G' = (V', E'), L, Q, M, k> | \exists R(V') \wedge \sum_{v \in R(V')} s(v) \leq k\}$$

It basically asks that given a k and a graph G that satisfies the preconditions L, Q, and M [1], whether there exists a replica placement solution set $R(V')$ which has a total cost less than or equal to k.

We now show that DOQFRP is NP-complete. First, it is easy to see that DOQFRP is in NP, since given a solution R and k, we know these input can always be encoded in polynomial length and whether or not the total cost of R is less than k can always be verified in polynomial time. Then, we show that DOQFRP can be reduced from the dominating set problem (DS), a well-known NP-complete problem, i.e., $DS \leq_p DOMQRP$.

Dominating Set(DS): For a graph $G = (V, E)$, a Dominating Set $DS(V)$ of G is defined as a subset of V such that $\forall v \in (V - DS(V))$ is adjacent to $\exists v' \in DS(V)$. It can be formally described using the following language.

$$\{< G = (V, E), k > | \exists DS(V) \wedge |DS(V)| \leq k\}$$

First, we show that a polynomial time reduction [7] can be constructed from any instance G of DS into an instance G' of DOMQRP problem using the following straightforward method. For $\forall v \in V$, we create a node $v' \in V'$, and assign $q(v') = 1$, $m(v') = 1$ and $s(v') = 1$ to the node. For every edge in E, we create an edge for E' and associate the edge with $l(u, v) = 1$, Obviously this reduction can be carried out in polynomial-time. We then need to show that a positive solution to one problem implies a positive solution to the other problem under the above described polynomial time transformation. More specifically, we need to show the instance $<G' = (V', E'), L, Q, M, k> \in DOMQRP$ has a replica set of whose total cost is no more than k if and only if the instance $<G = (V, E), k> \in DS$ has a dominating set of which the cardinality is no greater than k.

Sufficient Condition $\rightarrow$ If there exists a dominating set $DS(V)$ for G with cardinality less than k for G, for each $v \in DS(V)$, following the above construction mapping scheme, we can always mark the mapped node $v' \in V'$. Since the mapping is a one-to-one relationship and the costs, links, QoS, and fault-tolerant level in G's are all unit values, we can claim that all the marked v's constitute the $R(V')$ that is a replica solution set whose total cost is less than k for G'.

Necessary Condition $\leftarrow$ If $R(V')$ exists for G', following the above one-to-one mapping construction, for each node $v' \in R(V')$, we can mark a node v in V on the graph G. Then we claim all the marked nodes form the $DS(V)$ for G. Since every node $v' \in V' - R(V')$ can be directly linked to at least a node in $R(V')$ due to unit M, QoS, links and costs, a node not in $DS(V)$ must have a neighbor in $DS(V)$. Clearly, $|DS(V)| \leq k$ due to the one-to-one mapping of the

[1] For those graphs that inherently cannot meet the preconditions of Q, L M, it is trivial to conclude that no feasible solutions can be found.

transformation. So we can find a solution $DS(V)$ mapped from $R(V')$ for the reduced dominating set instance.

Hence we have proved that $DOQFRP$ is NP-complete, which immediately follows that $OQFRP$ is NP-hard.

3.4 Influencing Set and Influenced Set

Two critical data structures, which will be used in the subsequent sections to facilitate the analysis of the problem, are defined here.

Influencing Set: $\forall v \in V$, the influencing set of v is the set of nodes u whose distances to v are within the QoS requirement of v.

$$IG(v) = \{u | d(v,u) \leq q(v)\}.$$

Influenced Set: $\forall v \in V$, the influenced set of v is the set of nodes u whose distances to v are within the QoS requirement of u.

$$ID(v) = \{u | d(v,u) \leq q(u)\}.$$

From the perspective of a given node v, $IG(v)$ denotes a set of nodes that can meet the QoS requirement of v, which means if any member in $IG(v)$ is placed replica, it can serve the request of node v. $ID(v)$ denotes a set of nodes whose QoS requirements can be met by v, which means if v is placed a replica, those nodes in $ID(v)$ will be served. From fault tolerance perspective, $IG(v)$ denotes a set of nodes whose failures may influence node v; while $ID(v)$ denotes a set of nodes whose service may be influenced by failure of replica on node v.

3.5 The Essential Properties of the Problem

The following properties of the problem and the interplay between parameters are very important for us to design effective heuristics.

1. *Existence of a solution.* Without considering cost constraints, QoS can always be guaranteed. Given fault-tolerant levels $M = <m_0, m_1, \ldots, m_{N-1}>$, the necessary and sufficient conditions for the existence of a replica placement solution R is as follows:

   $$m(v) \leq |IG(v)|, \forall v \in V.$$

   This property reveals the essential relationship between M and Q.
2. *Relationship among M, Q and R.* Furthermore, if a solution R exists, then for every node v the following inequality holds.

   $$m(v) \leq |IG(v) \cap R|, \forall v \in V.$$

   It means that given node v, $m(v)$ must not exceed the cardinality of the replica placed nodes belonging to the influencing set of v, since the fault-tolerant level of a node v simply refers to the number of replicas being placed

on a subset of $IG(v)$. The size of the replica solution for the whole network should be no smaller than $m(v)$ of any node v and the replicas that meet m requirements for node v should be the intersection of $IG(v)$ and R.

3. *Lower bound and upper bound of M.* Lower bound (or the lowest level) and upper bound (or the highest level) of fault tolerance are the limitations inherent in the graph subject to QoS constraints. The lowest level of fault tolerance occurs when there is no redundant replica. It means that the fault-tolerant value of every node is assigned with 1. Thus, the lower bound of fault-tolerant value is $M = <1, 1, \ldots, 1>$. The highest fault-tolerant value a network can achieve occurs at the most aggressive case where all the nodes of the influencing set of any v are placed replicas. As a result, the upper bound for the fault-tolerant value of the whole network system is $M = <|IG(v_0)|, |IG(v_1)|, \ldots, |IG(v_{N-1})|>$. The two bounds show the range of M values. This analysis also shows that in our problem the QoS and fault-tolerant requirements are not independent, they will interact with each other.

4 Heuristic Algorithms

Based on the two ranking criteria proposed in last section, we will propose two heuristic algorithms for generating efficient solutions to the proposed problem.

4.1 Two-Level Selection Criteria

For any given node, we would like ask the following questions. How necessary is it to put replicas on all nodes of its influencing set to support the node? How much one replica place on one specific node can contribute to other nodes? We propose two criteria named **Replica Demanding Priority** and **Replica Profit Ratio** respectively to quantitatively answer the two questions and then employ the metrics to develop our heuristic algorithms.

Replica Demanding Priority(*RDP*): For each node v, $RDP(v)$ is defined as follows.

$$RDP(v) = \frac{1}{C^{m(v)-|IG\_R(v)|}_{|IG(v)-IG\_R(v)|}} \tag{4}$$

where $IG\_R(v)$ represents the set of nodes that are already placed replicas inside $IG(v)$. The higher demand priority is, the less flexibility or possibility for a node with various configurations is. For example, if $RDP(v) = 1$, it means that there is only one replica placement configuration can meet the requirements of v, so we can safely put replicas in $IG(v)$ to meet v without considering other potential better solutions. This measurement therefore enables us to rank all the nodes so that our algorithms can systematically choose candidate nodes to place replicas

in an ordered way. According to this criterion, we would like to select the nodes to place replicas inside their influencing sets according to the decreasing order of their RDP values.

Replica Profit Ratio (RPR): For each node v, $RPR(v)$ is defined as follows.

$$RPR(v) = \frac{\sum_{u \in (ID(v) - ID\_R(v)\} - \{v\}} AvgCost(u)}{s(v)} \tag{5}$$

where $AvgCost(u) = \frac{\sum_{p \in (IG(u) - IG\_R(u))} s(p)}{|IG(u) - IG\_R(u)|}$ is the average cost to meet one replica failure of given node u. Here, $ID\_R(u)$ means the nodes belonging to the influenced set of u have been placed replicas. So $RPR(v)$ means the profit ratio can be obtained if put one replica on node v. As some nodes in its influenced set may have been placed replicas, we exclude them from the cost calculation. We would like to pick those nodes that can bring the most profit to their individual influenced set first, because the more profit to influenced nodes, the less total cost may be needed to meet the overall M requirement. For example, to achieve the same effect as we put a replica on v, $RPR(v) = 10$ means that the total average cost is 10 times of $s(v)$. We would also like pick those nodes with the most profit ratio node first to place replicas for the objective of minimizing the cost.

These two selection criteria are used in tandem in our algorithm design because we would like to know which node we need to meet its fault tolerance first, then choose where the replicas should be placed within its influencing set based on their contribution to other nodes.

4.2 Framework of the Two Heuristics

The general ideas of the two heuristics are outlined in the following steps, and their more detailed pseudo-codes are given in Algorithms 1 and 2.

1. Calculate the shortest distance of all pairs.
2. Build the influencing set and the influenced set for every node.
3. Calculate the initial value of RDP for every node (the value will be iteratively updated as our algorithms increasingly place replicas).
4. For node v with the largest RDP, select nodes without replica from its influencing set and generate the candidate node set CNS.
   (a) Sort the nodes in CNS according to their RPR in decreasing order.
   (b) **Algorithm 1**: Select $m(v) - |IG\_R(v)|$ nodes in descending RPR order from CNS. Update the corresponding $IG\_R(v)$ and $ID\_R(v)$;
   Algorithm 2: Select *one* node with the largest RPR value from CNS. Update the corresponding $IG\_R(v)$ and $ID\_R(v)$.
5. If there is unsatisfied node left, update their RDP, RPR value and repeat step 4.

Algorithm 1: The pseudo code of batched replica placement

```
/* Procedure: Main Routine                                          */
input  : G = (V, E), Q, M
output: a replica placement solution X
1  Find all-pairs shortest paths distance;
2  Build influencing sets and influenced sets;
3  for ∀v ∈ V do
4  |  IG_R(v) = ID_R(v) = φ,r(v) = 0;
5  end
6  mark all nodes as unsatisfied;
7  calculate RDP and RPR for each node;
8  while there exists unsatisfied node do
9  |  Find one unsatisfied node p with the highest RDP value;
10 |  Select L = m(p) − |IG_R(p)| nodes from IG(p) − IG_R(p) according to the
   |  descending order of their RPR values as u_1, ..., u_L with r(u_i) == 0;
11 |  for j ← 1 to L do
12 |  |  Let r(u_j) = 1  and update all IG_R and ID_R sets related with u_j;
13 |  |  for ∀v ∈ V&&v is unsatisfied do
14 |  |  |  if |IG_R(v)| ≥ m(v) then
15 |  |  |  |  mark node v as satisfied;
16 |  |  |  end
17 |  |  end
18 |  end
19 end
20 return X =< r(v_0), r(v_1), ..., r(v_{|V|−1}) >;
```

5 Experimental Results

5.1 Experiment Setup

We have implemented two heuristics in C++ and conducted extensive experiments to evaluate the performance of the algorithms using widely different network configurations. The network used in our experiments has been generated using the Georgia Tech Internetwork Topology Models tool (GT-ITM)[2]. The network topology was randomly generated using the pure random model, where N nodes are randomly placed on a $s \times s$ square, and for each pair of nodes a link is generated with a probability σ, $(0 \leq \sigma \leq 1)$. Specifically, we choose the scale to be 1000, which means that the nodes of graph will be generated in a $1000 * 1000$ logical grid. We set N, the number of nodes, to be 100 and σ to be 0.8. The undirected graph such generated and used in our experiments contains 3997 links. The weight of a link is the Euclidean distance between the two nodes. The tool also provides another parameter β, which can be used to adjust the

[2] Interested readers are referred to [8,17] for details of the topology model and usage of the tool.

Algorithm 2: The pseudo code of one by one replica placement

```
/* Procedure: Main Routine                                          */
input  : G = (V, E), Q, M
output: a replica placement solution X
1  begin
2      Find all-pairs shortest paths distance;
3      Build influencing sets and influenced sets;
4      for ∀v ∈ V do
5          IG_R(v) = ID_R(v) = φ, r(v) = 0;
6      end
7      mark all nodes as unsatisfied;
8      calculate RDP and RPR for each node;
9      while there exists unsatisfied node do
10         Select u, an unsatisfied node which has the highest RDP value;
11         Select the node p, r(p) == 0, with the highest RPR value from IG(u) ;
12         r(p) = 1;
13         and update all IG_R and ID_R sets related with p;
14         for ∀v ∈ V&&v is unsatisfied do
15             if |IG_R(v)| ≥ m(v) then
16                 mark node v as satisfied;
17             end
18         end
19     end
20 end
21 return X =< r(v_0), r(v_1), ..., r(v_{|V|-1}) >;
```

ratio of long connections relative to short ones. In the current network generation configuration, we simply adopted the default value for β.

In the following reported experiments, different values of other parameters will be chosen for different evaluation purposes. Once the graph topology is generated, it poses constraints to the choices of values of M and Q. Therefore, we must first check for each node v if its m value is no greater than the cardinality of its influencing set $IG(v)$. To ensure the intrinsic constraints agree with each other, we can adjust them through trial and error. For example, we can reduce values of M. We can also test out larger $q(v)$ values. The larger $q(v)$ is, the larger the $IG(v)$ tends to be. Alternatively, we can even go back to the topology generation phase to assign a higher value to σ. This is because, for a node that demands m fault-tolerant requirement, the cardinality of its influenced set is at least m. If σ is too small, the chance for the node to get an influencing set with a cardinality larger than m is likely low. That is why σ was chosen as large as 0.8 for our network generation. If both M and Q for V have to be fixed but the graph does not meet the sufficient and necessary condition, then we must add link(s). Another way is to write validating routine to spot out cases where the value of m of a given node v is larger than the cardinality of the influencing set of v, then we can investigate the configuration of those m in practical environments

for further action. For the simulation purpose of our concern, we can always reset m to a value smaller than the cardinality of the influencing set of v so that our algorithms will have a set of legitimate input to proceed.

5.2 Calculation the Super Lower Bound SLB

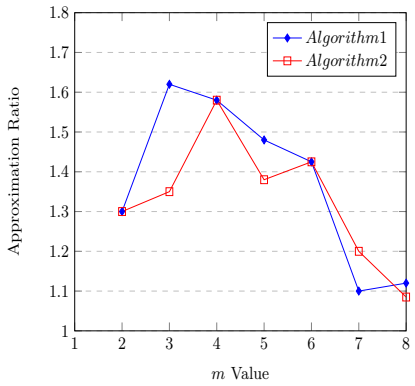
In order to quantitatively evaluate the effectiveness of the algorithms for different parameter configurations, ideally we should compare the replica cost results found by our heuristics with the cost provided by the optimal solutions. However, as our problem is NP-hard, we cannot have a polynomial time algorithm to achieve the optimal solution. Therefore, we relax the $\{0, 1\}$ constraints of X to the range of $[0, 1]$, consequently, we convert the original integer linear programming optimization problem into a linear programming optimization problem which can be efficiently solved. The optimal result computed from this relaxed linear programming optimization problem then will be used as the "super" lower bound for the solutions of the original optimization problem. This super lower bound serves as a comparison basis to evaluate the effectiveness of our solutions. Specifically, after we have calculated the cost due to the "super" lower bound (SLB) and the cost due to the feasible solution obtained from our heuristics (HSC), we use a cost approximation ratio ($\frac{HSC}{SLB}$) to measure the effectiveness of the solutions found by our heuristics. The smaller this ratio is, the closer the feasible solution found by our heuristics approximates the super lower bound of the relaxed optimal solution and the better the solution is supposed to be. So our approximation ratio is much strict and this evaluation approach has been commonly adopted by the research community [3,16].

5.3 The Effectiveness of the Algorithms

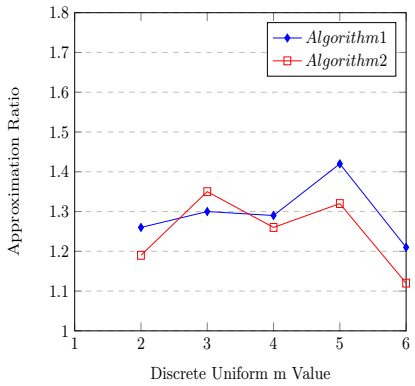
Performance with Different Fault-Tolerant Requirements. In the first group of experiments, we fixed the storage cost and the QoS of every node respectively. Then we evaluate the effectiveness of the two algorithms by changing M in two different methods. In the first method the values of M of all nodes are identical; while in the second method M is taken from a discrete uniform distribution for different nodes.

The first method is relatively straightforward to experiment. For the second method, we generate random numbers for discrete uniform distribution of fault-tolerant level $m(v)$. When there are more than one combination for a value, we average them. For example, if we observe $m = 4$, the uniform distributions in three different ranges $[1, 7]$, $[2, 6]$, $[3; 5]$ will be selected. Both the replication results and the super lower bounds are averaged to get the final experimental results.

The experimental results are shown in sub-figures (a) and (b) on Fig. 2. From the figures, we observe that both algorithms can find very effective solutions. The relative cost (approximation ratio) of the feasible solution to the super optimal cost is below 1.7. The experimental results also show that when the value of m is assigned with the discrete uniform distribution, the corresponding performance



Approximation ratio under different m values.



Approximation ratio under different discrete uniform m values.

Fig. 2. Approximation ratio of two proposed heuristic algorithms with different fault-tolerant value (M) setups. (QoS = 300, storage cost = 100)

is better than that when m is assigned with a constant value. Since the super lower bound is lower than the integer lower bound of the original optimization problem, the performance of our heuristics are actually even more promising.

Performance with Different QoS Requirements. In the second group of experiments, we investigate the effects of different QoS requirements on the performance of the two heuristics. So we fix fault-tolerant requirement (m) and the storage cost (s) of each node for different experiments.

The results are charted in Fig. 3. From the figure, we observe that the total number of needed replica goes down when the QoS requirement is not so strict (value of QoS goes up). The reason is when QoS requirement goes down (the value of QoS increases under our setup), the chance for all the nodes to be satisfied by less replicas increases too and this will lead to overall sparser replica placement. The figure also indicates that Algorithm 2 is slightly more effective

Table 1. Number of replicas on Cartesian product of two sets: Q and M. Here, X represents that the graph cannot meet the requirements.

Value of m	Value of QoS		
	100	*300*	*500*
1	43	5	4
2	X	13	10
3	X	25	15
4	X	34	19
5	X	40	22

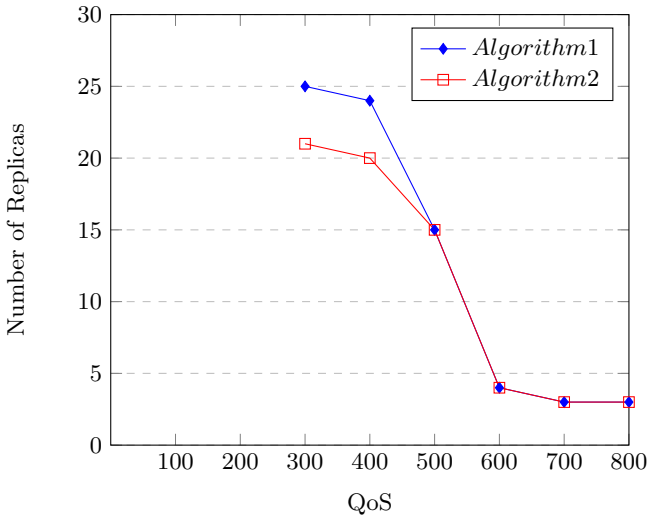


Fig. 3. Total number of replicas with different QoS requirements when fault-tolerant requirement and storage cost are fixed (m = 3, storage cost = 100).

than Algorithm 1 in terms of finding smaller replica numbers and thus less cost under the uniform storage cost setup.

Effect of Both QoS and Fault-Tolerant Requirements on Replica Placement. Furthermore, we conducted the third group of experiments to evaluate the conjunct influence of both QoS and m on the solution of replica placement using Algorithm 1. Table 1 indicates when both QoS and fault-tolerant requirements are very strict, there will be no replica placement solution for the given m and QoS due to that the requirements can't be met by the graph. From the experimental results we have the following observations. When QoS is fixed, our algorithms tend to return solutions that require more replicas, as M increases (higher fault tolerance requirement). The reason is that the larger M is, the more redundant replicas should be placed. On the other hand, if M is fixed, the larger QoS is (the QoS requirement is lower), the less replicas will be needed to meet the overall M constraints. This is because as QoS goes up, one replica could contribute to more nodes, which leads to an overall sparser solution. Table 1 shows how the number of replicas in our solutions changes with different QoS and fault-tolerant requirements.

6 Conclusion

Existing replica problem research only focuses on either QoS or fault tolerance. However, as more and more contemporary applications have become increasingly

distributed and decentralized, the need for more sophisticated replica problems to consider both QoS and fault tolerance simultaneously is on the rise.

In this work, we propose a unique graph model to capture multiple attributes of replica cost, QoS and fault tolerance at each individual node and link level. Based on the proposed model, we formulate a novel optimal replica placement problem and prove that the problem is NP-hard. To find optimized solutions to the problem efficiently, we introduce two core concepts, influencing set and influenced set, to capture the relationship between each individual node and the nearby nodes that meet the individualized QoS requirements. We then present two criteria, Replica Demanding Priority and Replica Profit Ratio, which are metrics integrating the cost, QoS, fault-tolerant, and topology features, to measure how necessary some special replica configuration is needed by a node and given replica's contribution to other nodes. Based on the proposed criteria, we further design two specific heuristic algorithms under one unified algorithmic framework. Our preliminary experimental results show that the solutions found by our algorithms are very close to the optimal results.

References

1. Aral, A., Ovatman, T.: A decentralized replica placement algorithm for edge computing. IEEE Trans. Netw. Serv. Manage. **15**(2), 516–529 (2018)
2. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. J. Comput. Syst. Sci. **65**(1), 129–149 (2002)
3. Cheng, C., Wu, J., Liu, P.: QoS-aware, access-efficient, and storage-efficient replica placement in grid environments. J. Supercomput. **49**(1), 42–63 (2009)
4. Cornuéjols, G., Nemhauser, G., Wolsey, L.: The uncapicitated facility location problem. Cornell University Operations Research and Industrial Engineering, Technical report (1983)
5. Du, Z., Hu, J., Chen, Y., Cheng, Z., Wang, X.: Optimized QoS-aware replica placement heuristics and applications in astronomy data grid. J. Syst. Softw. **84**(7), 1224–1232 (2011)
6. Du, Z., Zhang, S., Bader, D.A., Hu, J.: A 2-approximation algorithm for QoS-aware and fault-tolerant replica placement. In: 2020 IEEE High Performance Extreme Computing Conference (HPEC). IEEE (2020)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
8. Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/projects/gtitm/. Accessed 4 Apr 2012
9. Ghanbari, H., Litoiu, M., Pawluk, P., Barna, C.: Replica placement in cloud through simple stochastic model predictive control. In: 2014 IEEE 7th International Conference on Cloud Computing, pp. 80–87. IEEE (2014)
10. Grace, R.K., Manimegalai, R.: Dynamic replica placement and selection strategies in data grids-a comprehensive survey. J. Parallel Distrib. Comput. **74**(2), 2099–2108 (2014)
11. Hashem, I.A.T., et al.: The role of big data in smart city. Int. J. Inf. Manage. **36**(5), 748–758 (2016)
12. Khan, S.U., Maciejewski, A.A., Siegel, H.J.: Robust CDN replica placement techniques. In: the 23rd IEEE International Parallel Distributed Processing Symposium (IPDPS), pp. 1–8. Rome, Italy (2009)

13. Lim, A., Rodrigues, B., Wang, F., Xu, Z.: k-center problems with minimum coverage. Theoret. Comput. Sci. **332**(1–3), 1–17 (2005)
14. Mills, K.A.: Algorithms for Optimal Replica Placement in Data Centers. Ph.D. thesis, The University of Texas at Dallas (2017)
15. Sahoo, J., Salahuddin, M.A., Glitho, R., Elbiaze, H., Ajib, W.: A survey on replica server placement algorithms for content delivery networks. IEEE Commun. Surv. Tutor. **19**(2), 1002–1026 (2016)
16. Tang, X., Xu, J.: QoS-aware replica placement for content distribution. IEEE Trans. Parallel Distrib. Syst. **16**(10), 921–932 (2005)
17. Waxman, B.M.: Routing of multipoint connections. IEEE J. Sel. Areas Commun. **6**(9), 1617–1622 (1988)
18. Yousafzai, A., Gani, A., Noor, R.M.: Availability aware continuous replica placement problem. arXiv preprint arXiv:1605.04069 (2016)
19. Yu, S.S., Yu, S.: Autonomous unmanned road vehicle for making deliveries (2015), US Patent App. 14/318,690