An Efficient LP Rounding Scheme for Replica Placement

Zhihui Du
Department of Computer Science
New Jersey Institute of Technology, Nework, New Jersey, US
zhihui.du@njit.edu

Sen Zhang
Department of Mathematics, Computer Sciences and Statistics
State University of New York, College at Oneonta
zhangs@oneonta.edu

David A. Bader
Department of Computer Science
New Jersey Institute of Technology, Nework, New Jersey, US
bader@njit.edu

Jingkun Hu
Worldmoney Blockchain Management Limited
Hong Kong
kun.hu@worldmoney.org

***Abstract*—Large fault-tolerant network systems with high Quality of Service (QoS) guarantee are critical in many real world applications and entail diverse replica placement problems. In this paper, the replica placement problem in terms of minimizing the replica placement cost subject to both QoS and fault-tolerant constraints is formulated as a binary integer linear programming problem first and then relaxed as a linear programming problem. Given the optimal fractional linear programming solution, we propose a two-step rounding algorithm to obtain its integer solution. In the first step, a *half rounding* algorithm is used to simplify the problem. In the second step, a *cheapest amortized cost rounding* algorithm uses a novel metric, named *amortized cost*, to make locally optimal rounding decision for the remaining vertices independently. Furthermore, a conflict resolution algorithm is presented to tackle the situations when different vertices make conflicting rounding decisions. Finally, we prove that the proposed two-step rounding algorithm has a 2-approximation ratio when the additional conflict cost meets a given constraint.**

***Index Terms*—Replica Placement, Quality of Service, Fault Tolerance, Rounding Algorithm, Approximation Ratio**

I. Introduction

High Quality of Service (QoS) and fault-tolerance (FT) become much more important in emerging applications. When widely employed replica placement [5] technology is combined with both QoS and FT constraints, the problem becomes NP-hard [6].

Approximation algorithms [10], [11] are important because they can run in polynomial time and provide guarantee for the quality of solution even under the worst case. Considering replica placement technology has been widely used in many practical fields such as Content Delivery/Distribution Network (CDN) [8], cloud [4] and edge computing environment [2], it is necessary to explore approximation algorithms for replica placement problems to limit the cost boundary.

When a combinatorial optimization problem can be modeled as a binary integer linear programming (BILP) problem, the BILP could be relaxed to a linear programming (LP) problem to first get the fractional solution. Proper LP rounding [9] methods then could be used convert the fractional solutions to obtain sub-optimal integer solution to meet constraints of the original problems.

In this paper we present a novel LP rounding scheme to efficiently solve the replica placement problem that aims to minimize the total replica cost subject to both QoS and FT constraints simultaneously. Furthermore, we identify the condition in which the proposed rounding algorithms can achieve provable constant approximation ratio 2.

The major contributions of this paper are as follows.

1) Propose a two-step LP rounding scheme to significantly simplify the problem into different solvable steps.
2) Develop a novel *cheapest amortized cost rounding* and conflict resolution algorithm to generate optimized integer programming solution.
3) Prove the proposed rounding algorithm has a 2-approximation ratio when the conflict amortized cost is no more than the cheapest amortized cost.

II. PROBLEM DESCRIPTION

In this paper we attack the replica placement optimization problem that considers both QoS and FT constraints at the same time [6] by developing a novel rounding scheme. This section serves as a recap of the problem description, to both make this paper self-contained and facilitate the presentation of the proposed LP rounding scheme and approximation ratio proof.

The application concerns a group of geographically distributed servers that can provide service directly (with replica) or indirectly (without replica but it can relay the request to other servers). Furthermore, each server is associated with a potential cost if being placed with a replica. Each link between a pair of servers is associated with a distance weight. All servers are subject to both QoS requirement, which is defined with respect to a desired reachable distance threshold, and fault-tolerance, which is a desired number of replica servers in its close vicinity. The object is to determine a replica placement strategy that minimizes the total cost yet meets both

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

the above QoS and FT constraints which are allowed to be customized with respect to individual servers.

A. Graph Model

We model the above replicated network topology using a connected undirected weighted (both vertices and edges) graph $G = (V, E)$. Let $V = \{v_0, v_1, \ldots, v_{N-1}\}$ is the set of vertices whose cardinality is N, each representing a site that can be placed replica. $E = \{(u, v)|u, v \in V\}$ is a set of edges built over V. Let $L =< l_0, l_1, \ldots, l_{|E|-1} >$ be the weights associated with each $e \in E$. Let $S =< s_0, s_1, \ldots, s_{N-1} >$ be the cost vector with each element representing the cost associated with each $v \in V$. Let $Q =< q_0, q_1, \ldots, q_{N-1} >$ stand for the QoS requirements on V with q_i corresponding to the QoS requirement of v_i. Let $M =< m_0, m_1, \ldots, m_{N-1} >$ stand for the fault-tolerant requirements on V with m_i corresponding to the fault-tolerant level of vertex v_i.

B. Two Auxiliary Sets

The following two critical auxiliary data structures [6] will be used to facilitate the algorithm presentation in the subsequent section.

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

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

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

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

Here $d(u, v)$ is used to denote the shortest distance among all paths between v and u. Since we assume the graph is connected, $d(u, v)$ always exists and can be found by any well-known all-pairs shortest path algorithms.

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

Note that the general problem configuration is very flexible with respect to both edges and vertexes and the elements in S and Q vector can take different values.

C. Problem Formulation

Given a graph $G = (V, E)$ with each vertex 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 IX whose total cost of replicas is minimized subject to the constraints of QoS and fault tolerance of every vertex at the same time. Here, IX is a vector of 0/1 values, where ones refer to a set of vertices which are placed replicas. Formally, the above problem can be modeled as the following a binary integer linear programming problem, whose objective is to minimize the total cost of the network subject to the Q and M constraints:

$$Min : Cost(IX) = IX \cdot S^T \tag{1}$$

$$\text{subject to} \sum_{j \in IG(v_i)} IX_j \geq m_i, \forall v_i \in V \tag{2}$$

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

III. Rounding Scheme

In this section, we describe a rounding scheme to solve the proposed replica placement problem.

A. Linear Programming Relaxation as a Pre-Process

We relax the $\{0, 1\}$ constraints of IX in Eq.(3) to the range of $[0, 1]$, consequently, convert the original binary integer linear programming optimization problem into a continuous linear programming optimization problem which can be efficiently solved by LP solvers. The optimal result computed from this relaxed linear programming optimization problem then will be subsequently used as the input of our rounding algorithms.

According to Eq.(2), the number of constraints of our problem is a linear function of number of variables, it is thus easy to see that a linear programming solver will be able to run in polynomial time with respect to the numbers of the variables and the constraints to produce fractional coefficients.

B. Two-Step Rounding Scheme

We present an efficient rounding scheme to obtain an integer solution from the optimal fractional linear solution. It includes two rounding algorithms: *half rounding* and *cheapest amortized cost rounding*. The pseudo-codes of the two-step rounding scheme is given in Alg. 1.

Algorithm 1: Two-Step Rounding Algorithm

```
   input : M, S, LX
   output: integer solution IX
1  IX = 0;
2  forall LX_i ∈ LX do
3      if (LX_i ≥ 0.5) then
4          IX_i = 1
5      end
6  end
7  if (IX can meet constraint M) then
8      return IX
9  end
10 else
11     Call the cheapest amortized cost rounding algorithm Alg. 2 to update IX
12     return IX
13 end
```

Let $LX =< LX_0, ..., LX_{N-1} >$ be the fractional coefficients of linear programming solution. Let $IX =< IX_0, ..., IX_{N-1} >$ be initialized to be all zeros and be holding the binary integer values obtained from the rounding algorithms. During the *half rounding* step, for each $LX_i \geq 0.5$, we will round this fractional value to 1 ($IX_i = 1$). At the end of the first step, if the solution vector IX can already meet all the constraints, the algorithm will finish with a feasible

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

solution and it can be proved that the cost of IX is no more than 2 times cost of the optimal linear programming solution LX (see the proof of theorem IV.1).

If the rounding result IX from the first step cannot meet all vertices' constraints, we will simplify the problem as follows. For each vertex that has been rounded to 1 due to $LX_i \geq 0.5$, we will remove v_i from all other vertices' influencing set and at the same time reduce the fault-tolerance requirement of that vertex by 1. Consequently, we only need to consider those coefficients in the linear programming solution whose values are less than 0.5 to meet the unsatisfied vertices. The second step will be applied to this simplified problem and employ the *cheapest amortized cost rounding* algorithm to select a subset of the remaining fractional values and round them to 1 to meet the integer programming constraints.

C. Reformulation of the Simplified Problem

The problem after *half rounding* is formulated as follows. Given a linear programming solution $LX, 0 < LX_i < 0.5, 0 \leq i \leq Z-1$, we have $|V_Z| = Z$ vertices with replica cost $S =< S_0, ..., S_{Z-1} >$ that can meet $|V_W| = W$ vertices with fault-tolerance requirements $M =< m_0, ..., m_{W-1} >$. We will define the LXM matrix to describe the solution of the simplified problem as follows.

$$LXM_{W\times Z} = \begin{pmatrix} lx_{0,0} & ... & lx_{0,Z-1} \\ lx_{1,0} & ... & lx_{1,Z-1} \\ & ... & \\ lx_{W-1,0} & ... & lx_{W-1,Z-1} \end{pmatrix} \tag{4}$$

where $lx_{i,j} = LX_j$ if the replica on vertex $j \in V_Z$ can support vertex $i \in V_W$, otherwise $lx_{i,j} = 0$. For the simplified problem, we have

$$IG(v_i) = \{u_j | lx_{i,j} > 0, v_i \in V_W\}$$

and

$$ID(u_j) = \{v_i | lx_{i,j} > 0, u_j \in V_Z\}.$$

In summary, each row of $LXM_{W\times Z}$ describes the reduced influencing set of a vertex to be further satisfied, and each column represents the influenced set of a contributing vertex with a remaining non-zero coefficient.

D. Cheapest Amortized Cost Rounding

We first define the concept of *amortized cost* and then use it in designing our *cheapest amortized cost rounding* algorithm.

Definition III.1. *Amortized Cost: If one replica on vertex $u_j \in V_Z$ has cost $S(u_j)$ and the replica can support $ID(u_j)$ vertices to meet their requirements, then $AC(u_j) = \frac{S(u_j)}{|ID(u_j)|}$ is defined as the amortized cost of the replica on vertex u_j. It is the amortized cost of each supported vertex in $ID(u_j)$ if they equally share and pay the total cost $S(u_j)$ on vertex u_j.*

Based on *amortized cost* for each vertex in V_Z, we can generate the amortized cost vector $AC =< AC_0, ..., AC_{Z-1} >$ using S and the influenced set of each vertex. Then we can define the amortized cost matrix $ACM = (ac_{i,j})_{W\times Z}$, where $ac_{i,j} = AC_j$ if $lx_{i,j} > 0$. Otherwise $ac_{i,j} = 0$. Here i is the i^{th} vertex in V_Z and j is the j^{th} vertex in V_W.

We then sort the vertices in V_Z in their *amortized cost* increasing order and the vertices in V_W also in increasing order based on their cheapest supporting vertex's amortized cost.

Algorithm 2: Cheapest Amortized Cost Rounding Algorithm

input : M, S, LX
output: an integer solution IX

Calculate the amortized cost vector AC.
Sort V_Z in amortized cost increasing order.
Sort V_W based on each v_i's cheapest supporting vertex's amortized cost in increasing order.
$IXM = 0$
forall $v_i \in V_W$ **do**
  Select the cheapest amortized cost vertex in $IG(v_i)$.
  Update $IXM(v_i, :)$ based on the selecting results.
end
if *(no conflict in IXM)* **then**
  Let $IX_j = 1, \forall ix_{i,j} = 1$.
end
else
  Call Alg.3 to resolve the conflict and update IX.
end
return IX

For $\forall v_i \in V_W$, if $m_{v_i} = k > 1$, we can split v_i into k virtual vertex v_{i_0} to $v_{i_{k-1}}$ and each virtual vertex has $m_{i_l} = 1, 0 \leq l \leq k-1$. At the same time, we can set each vertex's fractional solution based on $IG(v_i)$ and make their sum equal to 1. In this way, all the unsatisfied vertices will have fault-tolerance requirement with 1. So we only need to handle the special case when all vertices have the value $m = 1$.

The basic idea of the *cheapest amortized cost rounding* is given in Alg.2. $\forall v_i \in V_W$, it will select the cheapest amortized cost vertex in its influencing set and set corresponding entries in IXM to 1. So after the rounding, each vertex v_i will have an integer programming solution vector $IXM(v_i, :)$ which means the v_i row of matrix IXM. In this way, we will build an integer programming solution matrix $IXM = (ix_{i,j})_{W\times Z}$ and it has the same shape as LXM, where $ix_{v_i,u_j} = 1$ means that vertex $v_i \in V_W$ selects the replica on vertex $u_j \in V_Z$, otherwise $ix_{v_i,u_j} = 0$.

$\forall u_j \in V_Z$, and $\forall v_{i1}, v_{i2} \in V_W$, if we have $ix_{v_{i1},u_j} = ix_{v_2,u_j}$, it means that all vertices in V_W have made the same decision on selecting or discarding the replica on vertex $u_j \in V_Z$. So we can just let $IX_j = 1, \forall ix_{i,j} = 1$ and get the *cheapest amortized cost* rounding solution. Otherwise it means that at least two vertices $v_{i1}, v_{i2} \in V_W$ and their decisions on at least one vertex $u_j \in V_Z$ conflict, or $ix_{v_{i1}}, u_j \neq ix_{v_2,u_j}$. Let $ConflictSet = \{v_c | v_c \in V_Z \wedge (v_c \text{ has conflict })\}$ be the set of conflict vertices. For each conflict vertex v_c, we will have two kind of costs, *Cheapest Amortized Cost* and *Conflict Amortized Cost*.

$$CheapestAmortizedCost(v_c) = \sum\{ac_{v_i,u_c} | \forall v_i \in V_W, ix_{v_i,u_c} = 1\} \tag{5}$$

It is the amortized cost that should be paid by those vertices that select the replica on v_c. We also have

$$ConflictAmortizedCost(v_c) = \sum\{ac_{v_i,u_c} | \forall v_i \in V_W, ix_{v_i,u_c} = 0\} \tag{6}$$

and it is the amortized cost that would not be paid by those vertices that discard the replica on v_c . So the payment conflict on v_c is generated.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

If there is conflict on any vertex, we will call the conflict resolution algorithm Alg.3 to get an updated integer programming solution IX to remove the conflict. We prove that if the condition inequality (7) holds,

$$ConflictAmortizedCost(v_c) \leq CheapestAmortizedCost(v_c),$$
$$\forall v_c \in ConflictSet \tag{7}$$

our algorithm will have a 2-approximation ratio (see theorem IV.4).

Algorithm 3: Conflict Resolution Algorithm

input : M, S, LXM, ACM, IXM
output: IX

forall $u_j \in V_Z$ **do**
  if *(vertex u_j has conflict)* **then**
    Calculate $CostA(u_j)$ and $CostB(u_j)$
    if $CostA(u_j) \leq CostB(u_j)$ **then**
      Select strategy A and update corresponding entry in IXM
    end
    else
      Select strategy B and update corresponding entry in IXM
    end
  end
end
Let $IX_j = 1, \forall ix_{i,j} = 1$
return IX

In Alg.3, we will check all vertices in V_Z to get the final solution IX without conflict and over-provisioning vertex.

If $\exists u_j \in V_Z$ has decision conflict, we have developed two strategies A and B to resolve the conflict. The basic idea of strategy A is letting all vertices select u_j. If there is over-provisioning vertex u_{jo} after solving the conflict on u_j, we will discard it. So the total cost of solution A to resolve the conflict on vertex u_j is the sum of additional rounding up cost by removing the over-provisioning vertex's cost if such vertex exists. Let

$$UpSet(u_j) = \{< v_i, u_j > |(ix_{v_i,u_j} == 0) \wedge (lx_{v_i,u_j} > 0)\}$$

$$OverUpSet(u_j) = \{< v_i, u_{jo} > |(ix_{v_i,u_{jo}} == 1) \wedge$$

$$u_{jo} \text{ is an over-provisioning vertex}\}$$

then we have

$$CostA(u_j) = \sum_{<v_i,u_j> \in UpSet(u_j)} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in OverUpSet(u_j)} ac_{v_i,u_j}$$

Alternatively, the basic idea of strategy B is letting all vertices discard u_j instead of choosing it. However, if we let v_i discard u_j, we must select another vertex u_{j+r} whose amortized cost is no less than $AC(u_j)$ to meet the replica requirement of v_i. To avoid introducing new conflict, if one vertex selects u_{j+r}, then all the vertices will be forced to select it. So the cost of strategy B to resolve the conflict on vertex u_j is the total new rounded vertex's cost subtracting the original rounding cost on vertex u_j. Let

$$RightSet(u_j) = \{< v_x, u_{j+r} > |(ix_{v_i,u_j} == 1) \wedge$$

$$(lx_{v_x,u_{j+r}} > 0) \wedge (ix_{v_x,u_{j+r}} == 0)\}$$

and

$$DownSet(u_j) = \{< v_i, u_j > |ix_{v_i,u_j} == 1\}$$

Similar to strategy A, if there is an over-provisioning vertex u_{jo} after rounding all u_{j+r} , B solution will also discard the replica on this vertex and add the corresponding tuple into $OverRightSet(u_j)$. So the total cost of strategy B can be defined as follows.

$$CostB(u_j) = \sum_{<u_i,v_j> \in RightSet(u_j)} ac_{v_i,u_j} -$$
$$\sum_{<v_i,u_j> \in DownSet(u_j)} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in OverRightSet(u_j)} ac_{v_i,u_j}$$

We will select the smallest r that can resolve the conflict to reduce the cost of solution B.

Based on the cost of two conflict resolution strategies, if $CostA(u_j) \leq CostB(u_j)$ then we will select A strategy. It means that we will let $ix_{v_i,u_j} = 1$ for $\forall v_i \in V_Z$ and $lx_{v_i,u_j} > 0$ to resolve the conflict. At the same time, over-provisioning vertex u_{jo} will be discarded by letting $ix_{v_i,u_{jo}} = 0$ for $\forall v_i \in V_W$. Otherwise, for the B strategy, we will let $ix_{v_i,u_j} = 0$ for $\forall < v_i, u_j > \in DownSet(u_j)$ and let $ix_{v_i,u_j} = 1$ for $\forall < v_i, u_j > \in RightSet(u_j)$. Similarly, if there is over-provisioning vertex u_{jo}, it will be discarded by letting $ix_{v_i,u_{jo}} = 0$ for $\forall v_i \in V_W$.

After this checking and employing one of the two conflict resolution strategies, we will achieve an integer solution with no conflict and no removable over-provisioning vertex in V_Z. Then we just let $IX_j = 1, \forall ix_{i,j} = 1$ and IX will be the solution of our *cheapest amortized cost rounding* method.

IV. Approximation ratio analysis and proof

It is clear that the time complexity of *half rounding* is $O(n)$ and the *cheapest amortized cost rounding* is $O(n^2)$. So both of our rounding algorithms are polynomial time complexity. Then we will prove that the proposed two-step rounding scheme has a 2-approximation ratio when condition inequality (7) holds. Since our rounding scheme consists of two algorithms: *half rounding* and *cheapest amortized cost rounding*, we will prove that 1). the total cost of *half rounding* vertices will not be more than two times of the corresponding optimal linear programming solution's cost; and 2). the cost due to *cheapest amortized cost rounding* on the rest vertices will also be no more than two times of the corresponding optimal linear programming solution's cost. So the total cost of our rounding scheme has a 2-approximation ratio. By combining the two rounding solutions together, we will generate a feasible solution of the original problem.

First, we will prove the *half rounding* theorem.

Theorem IV.1 (2 approximation ratio of *half rounding*)**.** *The total cost of* half rounding *vertices is no more than twice cost of the corresponding vertices in optimal linear programming solution.*

Proof. Let

$$LX =< LX_0, ..., LX_{N-1} >,$$

$0 \leq LX_i \leq 1, 0 \leq i \leq N-1$ be the solution of optimal linear programming and integer solution

$$IX =< IX_0, ..., IX_{N-1} >,$$

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

if $0.5 \leq LX_i$ then let $IX_i = 1$, otherwise $IX_i = 0, 0 \leq i \leq N-1$ be the result of *half rounding*.

So the total cost of solution IX is

$$\sum_{IX_i \neq 0} IX_i \times S_i \leq \sum_{IX_i \neq 0} 2 \times LX_i \times S_i \leq \sum_{IX_i \neq 0} LX_i \times S_i + \sum LX_i \times S_i$$
$$\leq 2 \times Cost(LP).$$

Here $Cost(LP)$ means the cost of corresponding vertex in linear programming solution. □

The *cheapest amortized cost rounding* algorithm's proof includes two parts. In the first part we will prove this method can generate a feasible integer programming solution. In the second part we will prove that the 2-approximation ratio holds.

Lemma IV.2 (Feasibility of *cheapest amortized cost rounding*)**.** *The* cheapest amortized cost rounding *mechanism can generate a feasible integer solution IX based on the optimal linear programming solution LX.*

Proof. Based on the procedure of Alg.2, if the first cheapest amortized cost vertex can meet all given requirements without any conflict, it will be a feasible solution. If there is any conflict, both A and B strategies can solve the conflict without violating the constraints. So the proposed *cheapest amortized cost rounding* can generate a feasible solution. □

To prove the proposed *cheapest amortized cost rounding* has 2-approximate ratio, based on the analysis in subsec III-D, we only need to prove the case when $M = 1$. Here we will first give a basic inequality about the cost for each vertex $v_i \in V_W$.

Lemma IV.3 (Basic inequality based on amortized cost)**.** *$\forall v_i \in V_W$ and $IG(v_i)$ is sorted in amortized cost increasing order, we have the following inequality.*

$$ac_{v_i,u_0} \leq$$
$$\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times ac_{v_i,u_j} - \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j} \quad (8)$$

where $\Delta_{v_i,u_0}^{v_i,u_j} = ac_{v_i,u_j} - ac_{v_i,u_0} \geq 0$.

Proof. Since LX is an optimal solution, $\forall v_i \in V_W$, it must meet the following constraint.

$$\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} LX_{u_j} \geq M(v_i) = m = 1.$$

If we let $ac_{max} = max\{ac_{v_i,u_0}, ..., ac_{v_i,u_{m-1}}\} = ac_{v_i,u_{m-1}} = ac_{v_i,u_0}$, then we will have the following results.

$$ac_{vi,u_0} = ac_{max} \leq (\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j}) \times ac_{max} \leq$$
$$\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times ac_{v_i,u_j} - \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j}$$

where $\Delta_{v_i,u_0}^{v_i,u_j} = ac_{v_i,u_j} - ac_{v_i,u_0} \geq 0$ □

Theorem IV.4 (The *cheapest amortized cost rounding* has a 2-approximation ratio for $M = 1$)**.** *The cost of* cheapest amortized cost rounding *mechanism has no more than twice cost of the optimal linear programming solution for a special case $M = 1$.*

Proof. If $M = 1, \forall v_i \in V_W$, we have the basic inequality (8). After each vertex selects its first cheapest amortized cost vertex and set the corresponding entry of IXM, let $HeadSet = \{u_j | \forall v_i \in V_W, ix_{v_i,u_j} = 1\}$. It is the set of vertices which have the cheapest amortized cost in each $IG(v_i)$. If there is no conflict in IXM, then we can sum the inequality (8) for $\forall v_i \in V_W$.

$$\sum_{v_i \in V_W} ac_{v_i,u_0} \leq$$
$$\sum_{v_i \in V_W} (\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times ac_{v_i,u_j} - \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j})$$

The result can be expressed as follows.

$$\sum_{u_j \in HeadSet} S(u_j) \leq \sum_{v_i \in V_W} \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times ac_{v_i,u_j}$$
$$- \sum_{v_i \in V_W} \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j}$$
$$= Cost(LP) - \sum_{v_i \in V_W} \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j} \leq Cost(LP)$$

The left hand side is the total cost of all the selected vertices and the right hand side is the original linear programming solution's cost. This is very excellent result and no any additional cost increase compared with the linear programming solution. So the 2-approximation boundary can definitely be met.

If there are conflicts, we let $ConflictSet$ be the set of vertexes that have conflicts. Let rounding up set be

$$RU = \bigcup_{u_j \in ConflictSet} (UpSet(u_j) \cup RightSet(u_j))$$

and rounding down set be

$$RD = \bigcup_{u_j \in ConflictSet} (OverUpSet(u_j) \cup DownSet(u_j) \cup OverRightSet(u_j)$$

If $RU \cap RD \neq \phi$, then we will let $NeutralizedSet = RU \cap RD$ and update both $RU = RU - NeutralizedSet$ and $RD = RD - NeutralizedSet$. Then by employing the conflict resolution strategy, the inequality (8) can be rewritten as the following expression.

$$ac_{v_i,u_0} + \sum_{<v_i,u_j> \in RU} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j} \leq$$
$$\sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times ac_{v_i,u_j} - \sum_{v_i \in V_W} \sum_{u_j \in IG(v_i),j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j}$$
$$+ \sum_{<v_i,u_j> \in RU} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j} \quad (9)$$

For all the $v_i \in V_W$, we can also sum the left hand side and right hand side of inequality (9) together. Let $SelectSet$

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

be the final vertices set whose replicas are selected. Then we will have the following inequality.

$$\sum_{u_j \in SelectSet} S(u_j) \leq$$

$$Cost(LP) - \sum_{v_i \in V_W} \sum_{u_j \in IG(v_i), j=0}^{|IG(v_i)|-1} lx_{v_i,u_j} \times \Delta_{v_i,u_0}^{v_i,u_j}$$

$$+ \sum_{<v_i,u_j> \in RU} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j}$$

We can prove the following result holds.

$$\sum_{<v_i,u_j> \in RU} ac_{v_i,u_j} - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j} =$$

$$\sum_{u_j \in V_Z} (\frac{\sum_{<v_i,u_j> \in RU} ix_{v_i,u_j}}{ID(u_j)} S(u_j)) - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j}$$

Based on inequality (7), we have $\frac{\sum_{<v_i,u_j> \in RU} ix_{v_i,u_j}}{ID(u_j)} \leq \frac{1}{2}$, then

$$\sum_{u_j \in V_Z} (\frac{\sum_{<v_i,u_j> \in RU} ix_{v_i,u_j}}{ID(u_j)} S(u_j)) - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j} \leq$$

$$Cost(LP) - \sum_{<v_i,u_j> \in RD} ac_{v_i,u_j} \leq Cost(LP)$$

Combining them together, we have

$$\sum_{u_j \in SelectSet} S(u_j) \leq 2 \times Cost(LP)$$

□

V. Related Work

Aral et al. [2] and Sahoo et al. [8] surveyed the replica placement problems in a broad range of environments and applications with varying objectives and constraints. Benoit et al. studied the problems of replica placement in tree networks subject to server capacity and distance constraints [1]. They proved that under single server policy, the general problem is NP-hard and provided a polynomial time optimal algorithm to solve the problem in a restricted case of binary trees.

The targeted replica placement is a special binary integer linear programming (BILP) problem. A BILP could be relaxed as an LP problem, and sometimes certain clever rounding algorithms [3] can be used to post-process fractional coefficients discovered by LP solvers to obtain sub-optimal integer solutions. Furthermore, some LP-based rounding algorithms have also been shown to actually be approximation algorithms to solve many NP-hard combinatorial problems such as vertex cover, set packing, and multiway-cut [10], [11].

Two widely used rounding approaches for designing and analyzing such approximation algorithms are deterministic rounding and randomized rounding. The most frequently used deterministic rounding algorithms is to round a fractional solution to a binary one by picking some threshold, e.g., 1/2, and rounding up or down depending on if a fractional co-efficient is greater than or less than the threshold. The basic idea of randomized rounding [7] is to use probabilistic methods to convert an optimal fractional solution to a relaxation of the problem into an approximately optimal integer solution to the original problem. A variant of randomized rounding, called oblivious rounding, has also been introduced in [12] to avoid the bottleneck of solving the linear program.

The optimal replica placement problem that is subject to both QoS and fault-tolerant constraints was initially formalized in research [6]. That paper solves the replica placement problem by developing two heuristic graph algorithms based on two novel heuristic metrics and the algorithms' efficiencies are shown through extensive experimental results. In this paper our work contributes in terms of designing an efficient LP rounding scheme and providing a proof that the proposed rounding scheme can not only provide a feasible integer solution, but also achieve a strict upper cost 2-approximation boundary when the conflict amortized cost is no more than the cheapest amortized cost.

VI. Conclusion

As more and more contemporary applications have become increasingly distributed and decentralized, the need for more sophisticated replica placement technologies that consider both QoS and fault tolerance simultaneously is on the rise. So developing efficient approximation algorithm for such problem is critical under many practical scenarios.

In this work, we propose a novel two-step rounding scheme to find sub-optimal solutions to the problem. The first step uses a *half rounding* algorithm, which provides a clear approximation ratio of 2 and is conflict free. The second step uses a *cheapest amortized cost rounding* algorithm, where *amortized cost* distributes the cost of a candidate replica over each influenced vertex of the replica. Since *cheapest amortized cost rounding* algorithm applies to each vertex individually, this step potentially could generate rounding conflicts among multiple vertexes. To resolve the conflicts, we further propose two adjustment strategies for each vertex with conflicts and the one with less adjustment cost will be chosen. Finally, we prove that in the situation when the conflict amortized cost is no more than the cheapest amortized cost, our rounding scheme will have a 2-approximation ratio.

References

[1] Hubert Larchevêque Anne Benoit and Paul Renaud-Goud. Optimal algorithms and approximation algorithms for replica placement with distance constraints in tree networks. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pages 1022–1033, 2012.

[2] Atakan Aral and Tolga Ovatman. A decentralized replica placement algorithm for edge computing. *IEEE transactions on network and service management*, 15(2):516–529, 2018.

[3] Dimitris Bertsimas and Rakesh Vohra. Rounding algorithms for covering problems. *Mathematical Programming*, 80:63–89, 1998.

[4] Hamoun Ghanbari, Marin Litoiu, Przemyslaw Pawluk, and Cornel Barna. Replica placement in cloud through simple stochastic model predictive control. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 80–87. IEEE, 2014.

[5] Kingsy Grace and Manimegalai Rajkumar. Dynamic replica placement and selection strategies in data grids—a comprehensive survey. *Journal of Parallel and Distributed Computing*, 74(2):2099–2108, 2014.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE

[6] Jingkun Hu, Zhihui Du, Sen Zhang, and David A. Bader. Qos-aware and fault-tolerant replica placement. In *20th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2020)*. Springer Nature, 2020.

[7] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[8] Jagruti Sahoo, Mohammad A Salahuddin, Roch Glitho, Halima Elbiaze, and Wessam Ajib. A survey on replica server placement algorithms for content delivery networks. *IEEE Communications Surveys & Tutorials*, 19(2):1002–1026, 2016.

[9] Srikrishna Sridhar, Ji Liu, Ce Zhang, Christopher Ré, and Stephen J Wright. An approximate, efficient solver for lp rounding. In *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2, page 2895–2903, 2013.

[10] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[12] Neal E. Young. Randomized rounding without solving the linear program. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, page 170–178, 1995.

978-1-7281-9219-2/20/$31.00 ©2020 IEEE