Delay Optimizations in Quorum Consensus

Xuemin Lin

School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052, Australia. e-mail: lxue@cse.unsw.edu.au

Abstract. The management of replicated data in distributed database systems is a classic problem with great practical importance. Quorum consensus is one of the popular methods, combining with eager replication, for managing replicated data. In this paper, we will investigate the problems of delay-optimal quorum consensus. Firstly, we will show that the problem of minimizing the total-delay (or mean-delay) is NP-hard. However, we can show that the problem restricted to some specific network topology, such as trees, rings, and meshes, can be solved in polynomial time. We also developed an approximate algorithm for a general case. The algorithm gives an approximate ratio less than 2. Secondly, we will present an efficient algorithm, based on the dynamic programming technique, to solve the problem of minimizing the maximal-delay.

Keywords: Quorum Consensus, Replicated Data Management, and Optimizations.

1 Introduction

The replicated data management in distributed databases is a classic problem with great practical importance. Distributed data warehouses and data marts contain a huge amount of replicated data distributed among a number of sites. Therefore, in recent developments [3, 4, 6, 10] of the area there is always a trade-off among system efficiency, data availability, data freshness, and data consistency.

Two replicated data management methods are available in the literature: *eager* and *lazy*. Eager replication management gives the data *consistency* and the highest data *freshness*. However, it suffers from the system efficiency due to an application of 2-phase commit protocol [18]. On the other hand, lazy replication management provides high system efficiency but does not necessarily provide the data freshness. Moreover, pure lazy replication management does not generally guarantee the data consistency. Recent research results [3] reveal that it is the best if these two methods can be combined.

To achieve a high system efficiency, "quorum consensus" is often adopted in eager replication. In this paper, we will investigate the quorum consensus method. A quorum consensus method is based on the design of a "coterie" (to be formally defined in section 2), such that each data processing (read or write) operation is executed on a subset (*quorum* - an element in the coterie) of the data sites over the network.

Recent developments in quorum consensus are mainly focused on 1) minimizing the total communication costs for processing a given set of transactions, and 2) minimizing the number of remote sites to be communicated while assembling a quorum. A number of quorum consensus protocols [5, 9, 11, 12, 13, 14, 16, 17, 20, 21] have been developed for these purposes.

Note that in quorum consensus, since messages are sent (possibly by the *multicast* mechanism [19]) to the multiple nodes in a quorum in order to ensure consistency of the operations the delays by passing messages through a long distance communication channel in a wide area network can create a bottleneck [18, 8] in the response time. In this paper we will investigate the problems of minimizing the"average" (or total) delay and minimizing the "maximal" delay.

These two problems were first investigated in [8]. It provided several algorithms for the problem of minimizing the maximal delay with respect to special classes of network topologies, such as rings, trees, and clustered graphs, while the complexity in general was left open. For the problem of minimizing the average (or total) delay, the paper [8] provides an approximate algorithm with approximation ratio 1.25 for rings with uniform links and uniform nodes. However, the algorithm in [8] does not generally guarantee a constant approximation ratio.

The first contribution of this paper is that we show that the problem of minimizing the average (or total) delays is NP-hard. Besides this, we provide an approximate algorithm with approximation ratio 2 in general. Note that the average-delay minimization problem, which we study in the paper, is more general than the problem in [8] where each node takes only a unit weight. While the NP-hardness we show in this paper is restricted to the case where each node takes a unit weight, the approximation ratio of our algorithm covers a general case where each node can take an arbitrary weight. Further, we can show that our approximate algorithm guarantees the exact solution for rings with uniform links and uniform nodes in contrast with the approximation result in [8]. Moreover, we can show that our approximate algorithm can guarantee exact solutions for the popular network topologies, such as meshes [18] and trees. The second contribution of the paper is that we present an efficient polynomial time algorithm to solve the problem of minimizing the maximum delay of quorum consensus. Again, this improves the results in [8] which can handle only some special graph topology.

Note that in [1, 19], the *file* allocation has been studied in an *on-line* environment with the assumption that each read is processed by reading one copy and each write has to be propagated to each copy - read one and write all policy. The duplicated data management, discussed in this paper, assumes that a file allocation is given, and investigates read/write policies instead of the policy of read one and write all. Therefore, the results and techniques in [1, 19] are not applicable.

The rest of the paper is organized as follows. In the second section, we provide the background knowledge, and precisely define the problems. In the third section, we present our results for the problem of minimizing the average (or total) delays. The fourth section presents our algorithm for solving the problem of minimizing the maximum delays. This is followed by conclusions and remarks. Due to the space limitation, we will not detail every proof; the interested readers may refer to the full paper [15] for details.

2 Preliminaries

A network is represented by a weighted graph $G = (V, E)$ where for each edge (physical link) $(u, v) \in E$, $d_{u,v}$ denotes the communication delay to send a unit message along the link from node u to v. A set S of subsets of V is *coterie* for G if and only if the following conditions hold [9, 20]:

1. *Intersection*: $\forall Q_1, Q_2 \in S$, $Q_1 \cap Q_2 \neq \emptyset$.
2. *Non-redundancy*: $\forall Q_1, Q_2 \in S$, $Q_1 \not\subset Q_2$.
3. *Connectivity*: $\forall Q \in S$, the subgraph G_Q *induced* [7] by Q from G is connected.

Each element Q in a coterie is called a *quorum*. The intersection property guarantees that any pair of quorums in a coterie have at least one common node. The connectivity of the nodes (vertices) in each quorum of a coterie is an important and practical restriction [11, 20]. If a vertex u is connected to vertex v only through z in a network, then we consider $\{u, z, v\}$ as a quorum rather than $\{u, v\}$; this is because a message between u and v must go through z.

Below is the notation to be used in the paper. We use, $V(G)$ to denote the vertex set of G and $l_{u,v}|_G$ to denote the length of the *shortest path* between u and v in G.

Suppose that S is a coterie for a network G; and that a vertex u uses $Q \in S$ as a quorum to execute a data operation issued from u. If the network adopts a multicast mechanism, then the *delay* of sending a message from u to every vertex (site) in Q is the length of the longest shortest path from u to a vertex of Q in G_Q; and it can be defined as: $\text{delay}(u, Q) = \max_{v \in Q}\{l_{u,v}|_{G_Q}\}$.

Given a coterie S, for each node u we define the smallest value of its delay with respect to all quorums as the *delay* of u in S; that is, $\text{delay}(u, S) = \min_{Q \in S}\{\text{delay}(u, Q)\}$. Further, for a coterie S we define the *max-delay* as:

$$\text{max-delay}(S) = \max_{\forall u \in V(G)} \{\text{delay}(u, S)\}. \tag{1}$$

The problem of minimizing the maximum delay (MMD) is to find a coterie for a given network G such that (1) is minimized.

Note that to apply a quorum consensus method, each data object must have a coterie designed. Different data objects may have different coteries designed. The problems discussed in this paper are about an optimal design of coterie for a given data object. Without loss of generality, we may assume a full replication of each data object, that is, a data object is replicated to every node (vertex) over the network. (For a data object partially replicated over the network, the results in this paper are also applicable; and we will discuss this in the last section.)

To define the problem of minimizing the average delay for a given data object, we assume that each vertex u is also associated with a weight w_u to represent the number of the messages issued at this site for reading/writing the data object, which have to be passed to an entire quorum. The average delay of a coterie S is defined as:

$$\text{avg-delay}(S) = \frac{\sum_{u \in V(G)} w_u \text{delay}(u, S)}{\sum_{u \in V(G)} w_u} \tag{2}$$

The problem of minimizing the average delay for a given data object over a network with weights specified on each vertex is to find a coterie such that (2) is minimized. As $\sum_{u \in V(G)} w_u$ is a constant, the problem of minimizing the average delay is equivalent to the problem of minimizing the total delays:

$$\text{tot-delay}(S) = \sum_{u \in V(G)} w_u \text{delay}(u, S) \tag{3}$$

In this paper we will present our results for the problem of minimizing the total delays (MTD).

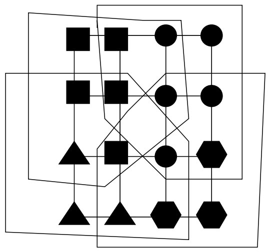
In a coterie S, it is clear that for each vertex u there is a quorum $Q_u \in S$ such that $\text{delay}(u, Q_u) = \text{delay}(u, S)$, and $\{Q_u : u \in V\}$ also form a coterie. This implies that both MMD and MTD problems can be simplified as:

Minimizing Max-Delays (MMD)
INSTANCE: A weighted graph $G = (V, E)$, and $\forall (u, v) \in E$ $d_{u,v}$ is an integer.
QUESTION: Find a set $\Pi = \{G_u : u \in V\}$ of connected subgraphs of G such that

- $S = \{V(G_u) : G_u \in \Pi\}$ is a coterie of G;
- $\forall u \in V$, $delay(u, V(G_u)) = delay(u, S)$;
- $\max_{u \in V}\{\text{delay}(u, V(G_u))\}$ is minimized among all such possible sets Π of connected subgraphs.

Note that in such a set Π as specified above, G_u is not necessarily different with G_v when $u \neq v$. For instance, in Figure 1 the 4 subgraphs (depicted by polylines), right-upper corner, right-bottom corner, left-upper corner, and left-bottom corner form a coterie for the graph in the figure. In the example, if each edge takes the weight 1, each circle node uses the right-upper corner subgraph as a quorum, each hexagon node uses the right-bottom corner subgraph as a quorum, each square node uses the left-upper corner subgraph as a quorum, and each triangle node uses the left-bottom corner subgraph as a quorum, then the coterie is the optimal solution for MMD.

Minimizing Total-Delays (MTD)
INSTANCE: A weighted graph $G = (V, E)$, $\forall (u, v) \in E$ $d_{u,v}$ is an integer, and $\forall u \in V$ w_u is an integer.
QUESTION: Find a set $\Pi = \{G_u : u \in V\}$ of connected subgraphs such that



Max-delay = 3

Fig. 1. Max-delays

- $S = \{V(G_u) : G_u \in \Pi\}$ is a coterie of G;
- $\forall u \in V$, $delay(u, V(G_u)) = delay(u, S)$;
- tot-delay(S) is minimized among all such possible sets Π of connected subgraphs.

Note that the problems of MMD and MTD were studied in [8]. However, our definition of MTD is more general than that in [8] which studied only MTD restricted to the case where each vertex takes a unit weight.

In the next two sections, we will present our results for MTD and MMD problems.

3 Minimizing Total-Delays

The section is organized below. It starts with an investigation of the complexity of the problem. Then we will present an approximate algorithm to solve the problem. It can be shown that the proposed algorithm has approximate ratio 2; and the bound is quite tight. Moreover, we can show that the algorithm can guarantee exact solutions for quite a large class of graphs.

3.1 Complexity of Minimizing Total-Delays

In this subsection, we prove the NP-hardness of MTD problem; this is done by proving the NP-completeness of the corresponding decision problem.

MTD decision (MTDD) problem

INSTANCE: A weighted graph $G = (V, E)$, $\forall (u, v) \in E$ $d_{u,v}$ is an integer, $\forall u \in V$ w_u is an integer, and an integer N.
QUESTION: Are there a set $\Pi = \{G_u : u \in V\}$ of connected subgraphs such that 1) $S = \{V(G_u) : G_u \in \Pi\}$ is a coterie of G; 2) $\forall u \in V$, $delay(u, V(G_u)) = delay(u, S)$; and 3) tot-delay$(S) \leq N$.

Theorem 1. *MTDD is NP-complete.*

To prove Theorem 1, below we transform the vertex cover problem to a special case of MTDD.

Vertex Cover

INSTANCE: A graph $G = (V, E)$, positive integer $K \leq |V|$.
QUESTION: Is there a vertex cover of size K or less for G?

For each instance $I_{VC} = \{G, K\}$ in the vertex cover, we construct the corresponding instance I_{VC}^{MTDD} of MTDD, say $I_{VC}^{MTDD} = \{G', N'\}$, as follows.

- $V(G) \subset V(G')$, and the vertices in $V(G') - V(G)$ are added along with the new edges created below.
- In G', replace each edge (u, v) in G by 4 edges (u, u'), (u', v), (u, v'), (v', v) such that $d_{u,u'} = 1.5$, $d_{u',v} = 1$, $d_{u,v'} = 1$, and $d_{v',v} = 1.5$.
- For each pair $\{u, v\}$ of vertices in G such that $(u, v) \notin E(G)$, add two edges (u, u') and (u', v) to G' such that $d_{u,u'} = 1$ and $d_{u',v} = 1$.
- For each pair $\{u, v\}$ of vertices in $V(G') - V(G)$, G' has one edge (u, v) connected and $d_{u,v} = 1$. Note that the induced graph on $V(G') - V(G)$ (with respect to G') is thus fully connected.
- In G', for each vertex $u \in V(G)$ $(V(G) \subset V(G'))$ let $w_u = 2n^4$ where n is the number of vertices in G; and for each vertex $u \in V(G') - V(G)$ let $w_u = 1$.
- Let $N' = 2n^5 + Kn^4 + n^2$.

An example of such a transformation is depicted in Figure 2(a) and (b), where Figure 2 (a) gives an instance in the vertex cover, the square vertices in Figure 2(b) represent the newly added vertices, the dotted edges carry the weight 1, and the square vertices are connected to form a complete subgraph.

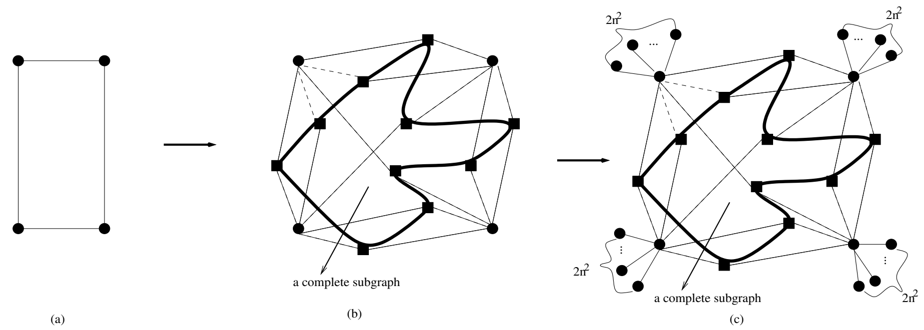


Fig. 2. a transformation from vertex cover to MTDD

Proof Sketch of Theorem 1: Clearly, the above transformation is polynomial with respect to n. It can also be shown in the full paper [15] that for each instance I_{VC} in the vertex cover problem, it has a solution if and only if the corresponding instance I_{VC}^{MTDD} has a solution. □

Further, we can have a stronger version of Theorem 1.

Corollary 2. *MTDD is NP-complete even restricted to the case when each vertex has a unit weight.*

Proof: To prove the corollary, we do the following modification on the instance $I_{VC}^{MTDD} = (G', N')$ constructed as above from I_{VC}.

- *Modification of* G'. For each vertex u in G' with weight $2n^4$, add $2n^4$ new vertices which are connected to u as a star, where those new added edges have weight 1. Figure 2(c) illustrates such a modification from the graph as depicted in Figure 2(b) regarding the instance (Figure 2(a)) of the vertex cover.
- *Modification of* N'. We also modify N' from $2n^5+Kn^4+n^2$ to $4n^5+Kn^4+n^2$.

Similar arguments to those in the proof of Theorem 1 [15] can lead to a proof of this corollary. □

3.2 An Approximate Algorithm

In this subsection, we present a heuristic for MTD. The algorithm consists of three steps as follows. Suppose that a weighted network G is given.

<u>**Algorithm MTD**</u>

Step 1 Compute all pair of shortest paths $\{L_{u,v} : \forall u, v \in V(G)\}$.
Step 2 Choose a vertex u^* such that $\sum_{v \in G} w_v \times l_{u^*,v}$ is minimized (note that $l_{u^*,v}$ denotes the length of the shortest path $L_{u^*,v}$).
Step 3 $\forall v \in V(G)$, let $G_v = \{u^*\}$.

Note that the algorithm MTD gives a coterie consisting of only one vertex $\{u^*\}$. The algorithm can be easily implemented in time $O(n^3)$. This is because that the step 1 can be implemented in $O(n^3)$ time by the algorithm Floyd-Warshall [7], while step 2 and step 3 can be trivially implemented in $O(n^2)$ time.

Next we show the approximation behaviour of the algorithm.

Theorem 3. *The approximation ratio of the algorithm MTD is not greater than* 2.

Sketch of the Proof: For each pair of vertices v_1 and v_2, let l_1 and l_2 respectively denote the delays of v_1 and v_2 for a given coterie S. Then, one can immediately verify $l_{u_1,u_2} \leq l_1 + l_2 \leq \max\{l_1, l_2\}$, where l_{u_1,u_2} denotes the shortest path between u_1 and u_2. This leads to a proof of the theorem; refer to the full paper [15] for the details. □

We can show that the ratio, obtained for our algorithm, is fairly tight by the following example. As depicted in Figure 3, a class of graphs $G = (V, E)$ have the following structure:

- $V = V_1 \cup V_2$ where V_1 has n vertices (circle vertices in V_1) and V_2 (square vertices in V_2) has $\frac{n(n-1)}{2}$ vertices;

- each pair of vertices in V_1 correspond to a vertex in V_2, and are connected by two edges via the vertex in V_2;
- the induced graph on V_2 is complete;
- each edge has a unit weight;
- the weight of a vertex in V_1 is $N = n^k$ (k is sufficiently large) and the weight of a vertex in V_2 is 1.

It can be immediately shown that the Algorithm MTD will randomly choose a square vertex as the coterie; and it produces the total delay is $T_{MTD} = 2(n-1)N + \frac{n(n-1)}{2} - 1$. On the other hand, for this class of graphs the minimal total delay will be the coterie where the delay of a quorum for each vertex is 1; and thus $T_{OPT} = Nn + \frac{n(n-1)}{2}$. Therefore,

$$\lim_{n \to \infty} \frac{T_{MTD}}{T_{OPT}} = 2 \qquad (4)$$

This means that the approximation bound in Theorem 3 is asymptotically tight. Figure 3 illustrates such a graph when $n = 4$. Moreover, the asymptotically lower bound illustrated in (4) also holds even restricted to the case when each vertex has a unit weight; this can be shown by adding N vertices "star-like" attached to each vertex with weight N in the way as depicted in Figure 2(c) and the new edges all take a very small weight ϵ.

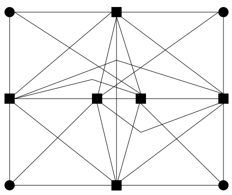


Fig. 3. to shown the tightness of the ratio

Furthermore, we can obtain a tighter expression of the approximation ratio restricted to the case when all vertices have a unit weight. The proof Theorem 4 adopts a similar idea to that of Theorem 3; and we also leave it to the full paper [15].

Theorem 4. *Algorithm MTD has approximation ratio* $2 - \frac{4}{n}$ *if each vertex is equally weighted and* $n \geq 4$. *Here,* n *is the number of vertices in the network.*

Next we show that the algorithm MTD can guarantee the optimal solutions for quite a large class of graphs, including rings, meshes, and trees.

Theorem 5. *Suppose that* G *is a weighted graph with* n *vertices, where*

1. *Each vertex in* G *is equally weighted.*

2. The vertex set V can be divided into m ($n/2$ or $(n-1)/2$) disjoint pairs of vertices:
   - *if $n = |V|$ is even, $V = \{u_i, v_i : 1 \leq i \leq \frac{n}{2}\}$, and*
   - *if $n = |V|$ is odd, $V = \{u_i, v_i : 1 \leq i \leq \frac{n-1}{2}\} \cup \{z_0\}$,*

such that a shortest path between each pair of v_i and u_i shares the common vertex; and the common vertex is z_0 if n is odd. Then the algorithm MTD will produce the optimal solution for G.

Proof: Let m denote $\frac{n}{2}$ when n is even, and denote $\frac{n-1}{2}$ when n is odd. Let l_i denote the length of the shortest path between v_i and u_i for $1 \leq i \leq m$.

Suppose that $S = \{V_v : v \in V\}$ is a coterie that leads to the optimal solution for MTD, where for each $v \in V$, $delay(v, V_v) = delay(v, S)$. We have that for $1 \leq i \leq m$ $delay(u_i, V_{u_i}) + delay(v_i, V_{v_i}) \geq l_i$; and thus $T_{OPT} \geq \sum_{i=1}^{m} l_i$.

On the other hand, if we choose this common vertex v (it is z_0 if n is odd) of G as a coterie, the total delay is $\sum_{i=1}^{m}(l_{v_i,v} + l_{u_i,v})$. Therefore, $T_{MTD} \leq \sum_{i=1}^{m}(l_{v_i,v} + l_{u_i,v})$.

Note that $l_{v_i,v} + l_{u_i,v} = l_i$. This implies that $T_{MTD} \leq \sum_{i=1}^{m} l_i$. Consequently, $T_{MTD} \leq T_{OPT}$. Thus, $T_{MTD} = T_{OPT}$. $\Box$

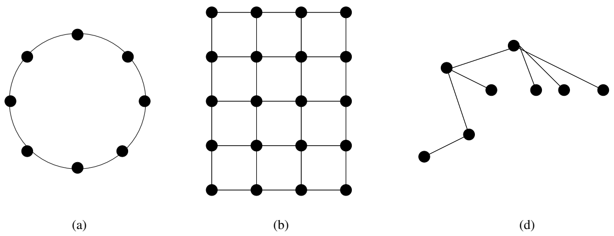


Fig. 4. rings, meshes, and trees

Theorem 5 implies that for those popular network topologies [18], such as *rings* (see Figure 4(a) for an example), *meshes* (Figure 4(b)), and *trees* (Figure 4(c)), the algorithm MTD can produce the optimal solution for MTD when each vertex takes the same weight and each edge also takes the same weight. Note that for rings where each edge and vertex take a unit weight, the algorithm in [8] can guarantee only the approximation ratio 1.25.

4 Minimizing Maximal-Delays

In this section, we investigate the problem of minimizing maximum-delays. Particularly, we present an efficient algorithm based on a mixture of dynamic programming and greedy paradigms. The algorithm conceptually consists of three steps as follows. Suppose that $G = (V, E)$ is the network.

Algorithm MMD

Step 1 In G, compute all pair shortest paths $\{L_{u,v} : u, v \in V\}$ and their lengths $\{l_{u,v} : u, v \in V\}$.
Step 2 Sort $\{l_{u,v} : u, v \in V\}$ in an increasing order and store them in D.
Step 3 Scan D according to the increasing ordering of $l_{u,v}$, iteratively construct subgraphs by an expansion till each pair of subgraphs have an intersection. Then check non-redundancy and remove all redundant quorums if exist.

As mentioned in last section, applying the algorithm Floyd-Warshall in [7], the step 1 can be implemented in $O(n^3)$ time where n is the number of vertices. Clearly, Step 2 can be implemented in $O(n^2 \log n)$ time as there are n^2 pairs of vertices. In fact, step 3 can also be implemented in $O(n^3)$ time; and it is described below.

In our implementation of Step 3, for each vertex v_k in G we use a linear data structure (for instance, a linked list) A_k to store the vertices w such that $l_{v_k,w}$ is less than the current element $l_{u,v}$ of D. Clearly, each induced [7] subgraph G_k by the vertices in A_k is connected, and $delay(v_k, A_k) \leq l_{u,v}$.

Moreover, suppose that A_k is constructed up to $l_{u,v}$ in D. Then A_k also gives us the information that for any pair $\{v_i, v_j\}$ of vertices and any two quorums Q_i and Q_j such that $delay(v_i, Q_i) \leq d_{u,v}$ and $delay(v_j, Q_j) \leq l_{u,v}$, Q_i intersects with Q_j at v_k if and only if v_i and v_j are both in A_k. Further, for each pair $\{v_i, v_j\}$ we use $B_{i,j}$ $(i < j)$ to indicate whether or not the current A_i and A_j already intersect. As mentioned earlier, Step 3 consists of two sub-steps:

Step 3a: construct a vertex set A_i for each vertex v_i, and then
Step 3b: check non-redundancy: if $A_i \subset A_j$ then replace A_j by A_i.

Below is the detailed implementation of Step 3a.

Step 3a

Set each A_k $(1 \leq k \leq n)$ to empty; and set each $B_{i,j}$ $(1 \leq i < j \leq n)$ to zero; $m \leftarrow 0$;
for each l_{v_i,v_j} **do**: (Scan D according to the increasing ordering of its elements)
  { $A_i \leftarrow A_i \cup \{v_j\}$; $A_j \leftarrow A_j \cup \{v_i\}$;
    for each vertex v_x in $A_i - \{v_j\}$ **do**
    if $B_{\min\{x,j\},\max\{x,j\}} = 0$ **then**
      { $B_{\min\{x,j\},\max\{x,j\}} \leftarrow 1$; $m \leftarrow m + 1$;
        if $m = \frac{n(n-1)}{2}$ **then** terminate the algorithm; }
    for each vertex v_x in $A_j - \{v_i\}$ **do**
    if $B_{\min\{x,i\},\max\{x,i\}} = 0$ **then**
      { $B_{\min\{x,i\},\max\{x,i\}} \leftarrow 1$; $m \leftarrow m + 1$;
        if $m = \frac{n(n-1)}{2}$ **then** terminate the algorithm; } □

It can be immediately shown that the algorithm runs in time $O(n^3)$. To implement Step 3b efficiently, we first sort all A_i in an increasing order according to the cardinalities of each A_i. Then we test the set inclusion property for A_i and A_j according to the ordering where $|A_i| < |A_j|$. In fact, the test of the set inclusion property for a pair A_i and A_j can be done by $O(\min\{|A_i|, |A_j|\})$ if all vertices A_i and A_j are stored in a global ordering applied to each A_k for $1 \leq k \leq n$. Therefore, Step 3b can also be implemented in $O(n^3)$. These imply that the algorithm MMD can be implemented in $O(n^3)$ time.

We can also immediately show that the algorithm MMD guarantees providing an exact solution for MMD problem; this is because we use an iteration procedure and the algorithm immediately terminates once each pair of quorums have an intersection.

5 Conclusion and Remarks

In this paper, we investigated the complexity issues of minimizing the max-delay (MMD) and minimizing the total-delay (MTD) for designing coteries (quorum consensus methods). Firstly, we showed that MTD is NP-hard even restricted to the case where each vertex takes a unit weight. Then we proposed a cubic approximate algorithm for solving MTD. The approximate algorithm has an approximation ratio 2 in general, and has an approximation ratio $2 - \frac{n}{4}$ for the case where each vertex takes a unit weight. We also showed that our approximation ratios are asymptotically tight. Moreover, we showed that the approximate algorithm can guarantee exact solutions for a quite large class of graphs such as "uniform" rings, uniform trees, and uniform meshes. Secondly, in the paper we presented a cubic algorithm to solve the problem MMD.

Note that for presentation simplicity, the results presented in the paper follow the assumption, made in [8], that each data object is fully replicated over the network. In case if a data object is partially replicated in the network, the design of coteries must be restricted to the vertices that hold the replicas. In this case, our results still hold if we add this restriction to our algorithms when generate quorums. However, we have to drop the connectivity property from a coterie; this is because that those sites holding replicas may not necessarily be connected.

As a possible future study, we are interested in MTD for non-uniform rings and meshes, that is, links and vertices do not necessarily take unit weight. Further, we are interested in MTD for design of "wr"-coteries [8, 11, 20].

Acknowledgement

Part of the work was done when the author took study leave in SEEM Department at the Chinese University of Hong Kong (CUHK). The author would also like to thank Prof. S. Zhang from CUHK for useful discussions.

References

1. B. Awerbuch, Y. Bartal, and A. Fiat, Distributed paging for general networks, *Journal of Algorithms*, 28:67-104, 1998.

2. P. Bernstein, V. Hadzilocs and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, Reading, Mass., 1987.
3. Y. Breitbart, R. Komondoor, R. Rastogi, and S. Seshadri, Update Propagation Protocols for Replicated Databases, *Proceedings of the 1999 ACM SIGMOD*, 97 - 108, 1999.
4. Y. Breitbart and H. F. Korth, Replication and Consistency: Being lazy helps sometimes, *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, Arizona, 1997.
5. H. Chen and J. Tang, An Efficient Method for Mutual Exclusion in Truly Distributed Systems, *Int. Conf. on Distributed Computing Systems*, 97-104, 1994.
6. P. Chundi, D.J. Rosenkratz, and S.S. Ravi, Deferred Updates and Data Placement in Distributed Databases, *Proceedings of the Twelveth International Conference on Data Engineering*, New Orleans, Louisiana, 1996.
7. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT press, 1990.
8. A. W. Fu, Delay-Optimal Quorum Consensus for Distributed Systems, *IEEE Transactions on Parallel and Distributed Systems*, 8(1), 59-69, 1997.
9. H. Garcia-Molina and D. Barbara, How to Assign Votes in a Distributed Systems, *J. ACM*, 32(4), 841-860, 1985.
10. J. Gray, P. Helland, P. O'Neil, and Dennis Shasha, The Dangers of Replication and a Solution, *Proceedings of the ACM SIGMOD Conference*, Montreal, Quebec, Canada, 1996.
11. T. Ibaraki, H. Nagamochi, and T. Kameda, Optimal Coteries for Rings and Related Networks, *Distributed Computing*, 8, 191-201, 1995.
12. A. Kumar, Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data, *IEEE Transactions on Computers*, 40(9), 996-1004, 1991.
13. A. Kumar and A. Segev, Cost and Availability Tradeoffs in Replicated Data Concurrency Control, *ACM Transactions on Database Systems*, 18(1), 102-131, 1993.
14. X. Lin, A Fully Distributed Quorum Consensus Method with High Fault-Tolerance and Low Communication Overhead, *Theoretical Computer Science*, 185(2), 259-275, 1997.
15. X. Lin, Delay Optimizations in Quorum Consensus, *full paper*, 2001. (http://www.cse.unsw.edu.au/~lxue.)
16. X. Lin and M. E. Orlowska, An Efficient Optimal Algorithm for Minimizing the Overall Communication Cost in Replicated Data Management, *J. Parallel and Distributed Comput.* 35, 97-103, 1996.
17. M. Maekawa, A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems, *ACM Transactions on Computer Systems*, 3(2), 145-159, 1985.
18. M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1991.
19. B. Maggs, F. Mayer auf der Heide, B. Vocking, and M. Westermann, Exploiting Locality for Data Management in Systems of Limited Bandwidth, *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, 284-293, 1997.
20. C. H. Papadimitriou and M. Sideri, Optimal Coteries, *Proceedings of ACM 10th Symposium of Distributed Computing*, 75-80, 1991.
21. D. Saha, S. Rangarajan, and S. K. Tripathi, An Analysis of the Average Message Overhead in Replica Control Protocols, *IEEE Transactions on Parallel and Distributed Systems*, 7(10), 1026 - 1034, 1995.