To appear in Distributed Computing 15 (2002), ©Springer-Verlag

The Optimal Location of Replicas in a Network Using a READ-ONE-WRITE-ALL Policy *

Stephen A. Cook[1], **Jan Pachl**[2], **Irwin S. Pressman**[3]

[1] Department of Computer Science, University of Toronto, e-mail sacook@cs.toronto.edu
[2] Fields Institute for Research in Mathematical Sciences, e-mail pachl@acm.org
[3] School of Mathematics and Statistics, Carleton University, e-mail ipress@math.carleton.ca

/ August 7, 2001

Summary. We consider the problem of locating replicas in a network to minimize communications costs. Under the assumption that the READ-ONE-WRITE-ALL policy is used to ensure data consistency, an optimization problem is formulated in which the cost function estimates the total communications costs. The paper concentrates on the study of the optimal communications cost as a function of the ratio between the frequency of the *read* and *write* operations. The problem is reformulated as a zero-one linear programming problem, and its connection to the *p*-median problem is explained. The general problem is proved to be NP-complete. For path graphs a dynamic programming algorithm for the problem is presented.

Key words: Data replication – Optimization – Dynamic programming – NP-completeness

---

Contents

* The initial version of this paper [17] was written while the authors were supported by the Centre for Advanced Studies, IBM Canada.

1 Introduction

When a data object is accessed from multiple locations in a network, it is often advantageous to replicate the object and disperse the replicas through the network. This leads to many design decisions about the *number* of replicas, their *location*, and the *policies* used to keep the replicas consistent.

The key parameter influencing the design decisions is how often the access to the data object changes the state of the object. In other words, how many accesses include *writing* into the object, rather than merely *reading* it. In this paper we use the *update ratio* to measure this parameter. To make informed decisions, one should understand how the costs of replication change with changing update ratio.

Potential benefits of data replication include increased availability of data, decreased access time, and decreased communications traffic or cost. Since replication is used for different reasons in different situations, it is not possible to identify one "correct" criterion of optimality. Moreover, there are many variables that could be included in a general economic model: fixed costs such as office space, hardware, software, and network infrastructure, and variable costs such as telecommunications costs, labor and CPU cycles.

In this paper we do not attempt to deal with such general models. Instead, we select a simple cost function, and investigate in detail how the minimal cost and the optimal replica locations depend on the update ratio. Our cost function describes communications cost (or communications traffic), using a variant of a cost function that was previously defined by other researchers.

The paper is organized as follows: After describing related work in Section 2 and the cost model in Section 3, we give a 0–1 programming formulation of the optimal replica problem in Section 4. The remainder of the paper deals with the minimal cost as a function of the update ratio. The function *mincost* is defined in Section 5. In Section 6 we prove that computing the function is NP-complete. Bounds for the function and for the optimal replica set are derived in Section 7. A partial answer to the monotonicity question from Section 5 is given in Section 8. In Section 9 we describe a (not necessarily efficient) algorithm for computing the function $mincost(u)$ on the interval $(0, 1)$, and in Section 10 we give an efficient algorithm for a class of path graphs.

2 Related work

A good introduction to the general problems of replication is found in the paper of Garcia-Molina [6]. A number of protocols [13] for replication have been studied, but we will focus on *traditional* ones here. Other priorities such as quality of service have been studied [1]. Heuristics for solving a database location problem with replication are investigated in [4], using a mixed integer programming formulation of the problem.

The cost model that we study is found in [2] and is a special case of Model 1 of the list of 12 models proposed for studying the File Assignment Problem *(FAP)*. The importance and generality of the FAP problem is discussed, and the authors note that if there were a polynomial time solution to the FAP, then other classical problems would have an efficient solution.

Wolfson and his coauthors have written extensively about replication. In [18], dynamic replication is studied using a read-one-write-all policy. There is the concern that writes might not execute if there is a failure in the system. The authors consider two-phase-locking to ensure one-copy serializability, and introduce other protocols to treat the difficult problem of replication in an environment where the read-write patterns are changing. Another view of this policy is given by Samy Mahmoud [14]:

> You can have a delayed update to some locations provided that you incorporate a proper audit trail and a synchronization mechanism to ensure that updates are applied in the same sequence order at all sites. It is not necessary to lock all sites before you apply the update. In this case it should be noted that if a read is attempted on a site before the latest updates are applied, the data retrieved may be dated. It will be the responsibility of the read module to ensure that no updates are attempted at that point in time.

A minimal-spanning-tree write policy is introduced in [15], where the cost of the update is the total length of the edges in the minimum spanning tree needed to reach all the replicas. The resulting optimal configurations tend to locate replicas in nodes adjacent to each other; for example, in a ring the replica set lies in a contiguous sequence. This may reflect environments where immediate updates are essential, and it gives a different pattern of optimal locations. The problem of dynamic replication is treated in [19]. Two algorithms are presented: one minimizes communications costs, and the other, communications time. Communications costs are minimized by the Cost-Adaptive-Replication (CAR) algorithm that produces a connected set of contiguous replicas, which, according to the authors, "grows like a variable-sized amoeba."

The related problem of locating supply centers in trees is studied in [5]. The p-center problem is solved for a tree in which supply center and demand points are at the nodes (this restriction can be relaxed), and the arcs have weights (distances, costs). The problem is to locate the supply center so that the maximum distance from each supply center is minimized. A linear-time and linear-space algorithm is provided in [5]. The problem is also known as the MIN-MAX MULTICENTER problem [7], and it has been shown that a polynomial time solution exists if the graph is a tree [11], but the problem on general graphs is NP-complete. The MIN-SUM MULTICENTER problem, or p-median problem, minimizes the sum of distances times the weights (traffic or demand at each node) over all sets of supply centers of size p [12]. The relationship to our problem is discussed in Section 6 below.

For many location problems on a network $G = (V, E)$, there is an optimal solution where the replicas are placed at locations that are a subset of V [16]. The problem of locating facilities on a network in which the locations are not constrained to be at nodes but may be along the edges is considered in [3]. Our work has been facilitated by Hakimi and Schmeichel [9] who proved that for the problems we described in [17], and continue to study here, all the replicas can be located at the nodes of the network. They also discuss the connection to the p-median problem.

3 Notation

3.1 The cost model

We consider a general network connecting a set $V = \{1, 2, \ldots, n\}$ of nodes. For $i, j \in V$, a client in node i may access a data object in node j; the cost of such an access is $d(i, j)$, where $d(i, j) \geq 0$. The costs $d(i, j)$ form an $n \times n$ matrix D.

An important special case is where V is the set of vertices in a connected graph, each edge of the graph is assigned a length, and $d(i, j)$ is the length of the shortest path from i to j. The quantities $d(i, j)$ then model communications costs. If each edge has length 1, $d(i, j)$ is called the *hop distance*.

It is often beneficial to maintain copies (replicas) of the data object in multiple nodes in the network. Various policies may be used to ensure the consistency of the replicas. Most policies distinguish between two types of operations by clients on the data objects—*write* and *read*. One of the simplest policies is called READ-ONE-WRITE-ALL: To read, the client accesses one replica and reads its value; to write, the client accesses all replicas and updates them.

In deriving the cost estimates in this paper, we assume that the cost of multiple accesses in the network is the sum of individual accesses. This independence assumption is a major simplification; some of the studies surveyed in Section 2 deal with more detailed cost models and do not assume independence.

With the assumption, we can now derive the total data access cost with the READ-ONE-WRITE-ALL policy. In view of the independence assumption, it is sufficient to consider the replicas of a single data object. Let the replicas be located at each node of a subset $R \subseteq V$. The cost of a single write operation by client i is

$$\sum_{j \in R} d(i,j)$$

The cost of a single read operation by client i is

$$d(i,R) = \min_{j \in R} d(i,j)$$

Let the total number of accesses to the object by client i in some fixed time interval be τ_i (thus τ_i = traffic originating at client i). For some u_i, $0 \le u_i \le 1$, there are $\tau_i u_i$ *write* operations and $\tau_i(1-u_i)$ *read* operations. The total cost of all operations is

$$cost(R, \vec{u}, \vec{\tau}) = \sum_{j \in R} \sum_{i \in V} \tau_i u_i d(i,j) + \sum_{i \in V} \tau_i (1-u_i) d(i,R) \ . \quad (3.1)$$

In this paper we investigate several variants of the problem to minimize (3.1).

If $[\tau_1 u_1, \ldots, \tau_n u_n]D = [\xi_1, \xi_2, \ldots, \xi_n]$ is the matrix product, then ξ_j is the cost of all writes to node j and the formula may be written as

$$cost(R, \vec{u}, \vec{\tau}) = \sum_{j \in R} \xi_j \ + \sum_{i \in V} \tau_i (1-u_i) d(i,R)$$

As is pointed out in [9], in this form the write costs at j may be considered to be part of the setup costs of node j. Note however that this reformulation does not capture the dependency of the cost on $\tau_i u_i$, which is studied in more detail below.

The second term in (3.1) is the sum of the costs of all read accesses to the set R of size $p = |R|$. Considering only the second term, we have a standard p-median problem where the setup costs are ignored and p is fixed. Together we have in (3.1) a variation of the p-median problem with setup costs.

The cost function (3.1) is not the only possible one. As in any modeling setting, we have to decide which aspects of the problem should be included in the model and which should be omitted. Even when we focus specifically on communications costs, there is no universal agreement on the most appropriate cost function, and in fact the cost depends on the underlying communications technology. Data access typically occurs at a fairly high level in protocol hierarchy, and the traffic generated by a single data access may include multiple interactions at lower levels, such as acknowledgments, retransmissions and transaction management messages. We do not attempt to capture such lower level interactions in our model.

Alternative cost functions have been studied in the publications surveyed in Section 2. (See also [8].) Yet another cost function that may be of interest is

$$cost_2(R,u) = \sum_{i \in V} \left(u \max_{j \in R} d(i,j) \ + (1-u) \min_{j \in R} d(i,j) \right) \quad (3.2)$$

This models the time needed to complete a data access. In a tree network, the replica sets optimizing (3.2) may be chosen to be contiguous, which is reminiscent of the results in [15].

In the rest of this paper we stay with the cost function (3.1).

3.2 Uniform clients

When all the clients accessing the replicated object exhibit the same access pattern, τ_i and u_i are the same for all i. The common value u of u_i is called the *update ratio*:

$$u = \frac{\text{write traffic}}{\text{read traffic} + \text{write traffic}}$$

The update ratio is called θ in [10] where it is used to develop cost models for mobile computers.

Without loss of generality we may assume that the common value of τ_i is 1, and the cost function (3.1) then becomes

$$cost(R,u) = u \sum_{j \in R} \sum_{i \in V} d(i,j) \ + (1-u) \sum_{i \in V} d(i,R). \quad (3.3)$$

This can be written as:

$$cost(R,u) = uA + (1-u)B = (A-B)u + B \quad (3.4)$$

$$A = A(R) = \sum_{j \in R} \sum_{i \in V} d(i,j) \quad (3.5)$$

$$B = B(R) = \sum_{i \in V} d(i,R) \quad (3.6)$$

A is the *total write cost*; uA is the write cost for update ratio u; B is the *total read cost* and $(1-u)B$ is the read cost for update ratio u. This gives a line on the $(Cost, u)$ plane with Y-intercept $b = B$ = the total read cost and slope $m = A - B$ = the difference between the total write and read costs. Since $d(i,j) \ge 0$, it follows that $b \ge 0$ and $m \ge 0$.

Remark. If we remove one replica from R, then the total write cost A does not increase, and the total read cost B does not decrease. The slope of the line $y = (A-B)u+B$ does not increase, and the Y-intercept does not decrease. The maximum slope is attained for $R = V$ (a replica in each node). The minimum slope is attained for one-element replica sets: $A = B$ and $m = 0$.

4 The 0–1 linear programming problem

Consider the problem of minimizing (3.1) over all choices of R, given $\vec{u}, \vec{\tau}, D$. To facilitate comparison with other location problems (see e.g. [2], [4], [16]), we now formulate the problem as a 0-1 linear programming problem. We introduce variables x_j and y_{ij}, where

- $x_j = 1$ if there is a replica at the j-th server; otherwise, $x_j = 0$.
- $y_{ij} = 1$ if the client at node i reads from the replica at node j (i *homes on* j); otherwise, $y_{ij} = 0$. Each client homes on a unique server.

For the cost matrix $D = [d(i,j)] = d_{ij}$, we formulate the problem in terms of constraints that represent the following requirements:

- Each update is sent to all replicas.
- The client reads from a unique replica.
- A client can only home on a server that has a replica.

$$\begin{array}{llll} \textit{minimize} & \sum_{i=1}^{n}\sum_{j=1}^{n} \tau_i d_{ij}(u_i x_j + (1-u_i)y_{ij}) & & \\ \textit{subject to:} & x_j - y_{ij} \geq 0 & 1 \leq i \leq n,\ 1 \leq j \leq n & \text{(i)} \\ & \sum_{j=1}^{n} y_{ij} = 1 & 1 \leq i \leq n & \text{(ii)} \\ & x_j = 0,1 & 1 \leq j \leq n & \\ & y_{ij} = 0,1 & 1 \leq i \leq n,\ 1 \leq j \leq n & \end{array}$$

The n^2 constraints (i) represent the fact that a client may home on a node only if a replica is at the node. The n constraints (ii) ensure that each client homes on exactly one replica.

For uniform clients we apply the same simplification as in Section 3.2 and the objective function becomes

$$cost(R,u) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}(ux_j + (1-u)y_{ij})$$

Given a choice of τ_i and u_i (or u in the uniform case), the $0-1$ programming solution determines an optimal replica set and, for each node i, the replica on which i homes.

We have already noted that the cost function (3.1) is closely related to the p-median problem. The $0-1$ formulation of the p-median problem includes the same set of constraints plus one further constraint: $x_1 + x_2 + \cdots + x_n = p$. Hence the polytope of the replica problem contains all the p-median polytopes on n parallel hyperplanes $x_1 + \cdots + x_n = p$, $1 \leq p \leq n$.

5 Minimal cost as a function of the update ratio

The optimal choice of the replica set R depends in an essential way on the relative frequency of the read and write operations. To investigate this dependency in detail, let us consider the case of uniform clients (Section 3.2), and define

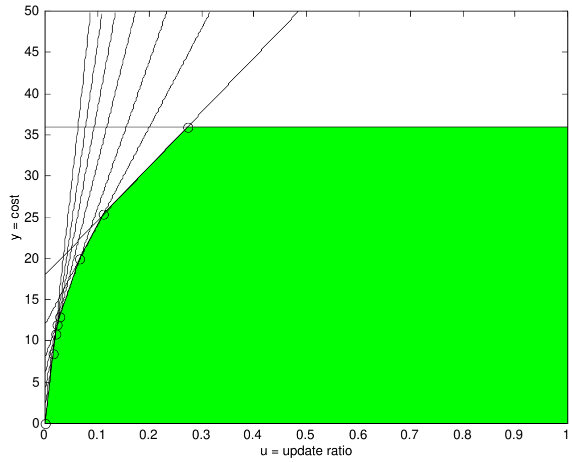

Fig. 1. The *mincost* curve for the 12-node path graph.

$$mincost(u) = \min_{\emptyset \neq R \subseteq V} cost(R,u) \tag{5.7}$$

for $0 \leq u \leq 1$. A set $R \subseteq V$ is *optimal for* u if $mincost(u) = cost(R,u)$.

We wish to investigate the behavior of the minimal cost as a function of the update ratio u. From Equations (3.4) and (5.7) we see that the function $mincost(u)$ is the minimum of $2^n - 1$ linear functions $cost(R,u)$ corresponding to the $2^n - 1$ nonempty subsets $R \subseteq V$. Thus $mincost(u)$ is piecewise linear and concave.

As discussed in the remark at the end of Section 3.2, for $u = 1$ the optimal set R consists of a single replica, and the optimal line has slope zero and Y-intercept $b_1 = mincost(1)$. For $u = 0$ there is a replica at each node and the optimal line has the maximum possible slope m_0 and Y-intercept $mincost(0)$. If the costs $d(i,j)$ are given by a shortest-path distance then $mincost(0) = 0$. The other linear segments $y = mu+b$ forming the function $mincost$ satisfy $0 < b < b_1$ and $0 < m < m_0$.

Example 1. Consider a network with 12 nodes $1, 2, \ldots, 12$ and the cost function $d(i,j) = |i-j|$. This is the hop distance in the 12-node path graph. There are $4095 = 2^{12} - 1$ possible choices of replica sets. The following table describes the linear segments that form the function *mincost* and the corresponding replica sets. The function $mincost(u)$ is plotted in Fig. 1.

In view of symmetry, some of the linear segments belong to more than one replica set; in such cases only one replica set is listed in the table. ● indicates that there is a replica at node i, and ○ indicates there isn't one. Note that in four instances there are three lines coincident at a point on the cost curve; this is shown by $\times$ in the last column of the table. These intermediate lines are omitted from Fig. 1.

	replica set R	$cost(R,u)$	
(1)	●●●●●●●●●●●●	$572u$	
(2)	●○●●●●●●●●●●	$505u+1$	×
(3)	○●●●●●●●●●●○	$438u+2$	
(4)	○●○●●●●●●●●○	$389u+3$	×
(5)	○●○●●●●●●○●○	$340u+4$	
(6)	○●○○●●●●●○●○	$297u+5$	×
(7)	○●○○●●●●○○●○	$254u+6$	
(8)	○●○○●○●●○○●○	$217u+7$	×
(9)	○●○○●○○●○○●○	$180u+8$	
(10)	○○●○○○●○○●○○	$120u+12$	
(11)	○○○●○○○○●○○○	$66u+18$	
(12)	○○○○○○●○○○○○	36	

The $(X,Y)-$ coordinates of the intersection points are

X	Y	line intersections
1/67	572/67	lines (1) (2) (3)
1/49	536/49	lines (3) (4) (5)
1/43	512/43	lines (5) (6) (7)
1/37	476/37	lines (7) (8) (9)
1/15	20	lines (9) (10)
1/9	76/3	lines (10) (11)
3/11	36	lines (11) (12)

5.1 Motivating questions

Answering the following questions would bring us closer to understanding the function *mincost*.

1. $mincost(u)$ is defined as a minimum of 2^n-1 functions. Is there an efficient algorithm for computing the function $mincost(u)$?
2. What is the maximum number of linear pieces making up the function $mincost(u)$?
3. For a given u, how large can an optimal R be?
4. Is the size of an optimal R monotonically non-increasing as a function of u?
5. Is there an efficient algorithm for computing a good approximation to $mincost(u)$?

Partial answers to most of these questions are given in the rest of the paper.

6 NP-completeness

In this section we prove an NP-completeness result, which strongly indicates that the general problem of computing $mincost(u)$ is intractable. The result applies when the input to the problem is an arbitrary, undirected graph G with unit edge length, together with an arbitrary rational update ratio u. Of course this negative result also applies to the more general setting in which the input is an arbitrary cost matrix D and non-uniform parameters τ_i and u_i.

The function $mincost(u)$ below is the function defined by (5.7), where $cost(R,u)$ is defined by (3.3) and $d(i,j)$ is the hop distance from i to j in the graph G.

REPLICA PROBLEM

INSTANCE: Undirected connected graph $G=(V,E)$, rational numbers u and k.
QUESTION: Is $mincost(u) \le k$?

We note that the REPLICA PROBLEM is related to the MIN-SUM MULTICENTER ([7], p. 220), which was proven to be NP complete in [12] as the p-median problem. The difference between the REPLICA PROBLEM and the p-median problem is that in the latter problem the INSTANCE includes an integer p instead of a number u, and the question is whether there exists $R \subseteq V$ such that $|R| = p$ and $B(R) \le k$ (*see* (3.6)).

In [19] an NP-completeness proof is sketched for another variation of the REPLICA PROBLEM in which the write cost component $A(R)$ in $cost(R,u)$ is the sum over all nodes i of the total length of the minimum spanning tree for $R \cup \{i\}$.

Here we prove the following:

Theorem 1. *The REPLICA PROBLEM is NP-complete.*

Proof. A proof that a problem X is NP-complete consists first in showing that X is in the class NP and second in showing that every NP problem can be transformed (in polynomial time) to X. See [7] for details. Here, showing that the REPLICA PROBLEM is in NP is straightforward. A nondeterministic polynomial time algorithm consists simply of guessing the replica set $R \subseteq V$ and accepting the input (G,u,k) if and only if $cost(R,u) \le k$.

To show that every NP problem can be transformed to X, it suffices to transform a single known NP-complete problem to X. Here, we choose EXACT COVER BY 3-SETS (X3C) as the known NP-complete problem ([7], p. 53), and show how to transform X3C to the REPLICA PROBLEM.

X3C

INSTANCE: A finite set X with $|X| = 3q$ and a collection C of 3-element subsets of X.
QUESTION: Does C contain an *exact cover* for X; that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

The idea of the transformation is as follows. Given an instance (X,C) of $X3C$, we produce an instance (G,u,k) of the REPLICA PROBLEM. Here, G is like a bipartite graph with one vertex set X and the second vertex set C, with an edge between a 3-element set $Y \in C$ and $x \in X$ if and only if $x \in Y$, except that we add edges between every pair of nodes in C. The parameters u and k are selected so that if $X3C$ has a solution $C' \subseteq C$, then we may choose the replica set R to be C' and $cost(R, u) = k$. If the instance of $X3C$ has no solution, however, then $cost(R, u) > k$ for each choice of R.

More formally, let the instance of $X3C$ be (X, C), with $X = \{a_1, \ldots, a_{3q}\}$ and $C = \{Y_1, \ldots, Y_t\}$. Define the graph $G = (V, E)$ as $V = \{a_1, \ldots, a_{3q}, b_1, \ldots, b_t\}$ for new elements $b_1, \ldots, b_t$, and

$$\begin{aligned} E = \{ (i,j) \mid & \text{ either } i \text{ and } j \text{ are distinct elements} \\ & \quad \text{of } \{b_1, \ldots b_t\} \\ & \text{ or } \{i,j\} = \{a_\ell, b_m\} \text{ for some } \ell, m \\ & \quad \text{such that } a_\ell \in Y_m \ \} \end{aligned}$$

We may assume that every a_i in X is in at least one Y_j in C, so that G is connected. Define the following parameters:

$$d = 6q + t - 4$$
$$u = \frac{2}{d+2}$$
$$A_0 = qd, \ B_0 = 2q + t$$
$$k = uA_0 + (1-u)B_0 = \frac{d}{d+2}(4q + t)$$

Suppose that the instance (X, C) of $X3C$ has a solution $C' = \{Y_{i_1}, \ldots, Y_{i_q}\}$. Select $R = \{b_{i_1}, \ldots, b_{i_q}\}$. Each $j \in R$ contributes an amount $\sum_{i \in V} d(i,j) = (t-1) + 3 + 2(3q-3) = 6q + t - 4 = d$ to the sum $A(R)$. Hence $A(R) = qd = A_0$. Every node of G not in R has the distance 1 from some vertex in R, so $B(R) = |V| - q = 2q + t = B_0$. Thus, $cost(R, u) = k$, so $mincost(u) \leq k$.

Conversely we show that if the instance (X, C) of $X3C$ has no solution then $mincost(u) > k$. Consider any subset $R \subseteq V$, and let $r = |R|$. As demonstrated above, each $j \in R$ that is in $\{b_1, \ldots, b_t\}$ contributes an amount d to the sum $A(R)$. On the other hand, each $j \in R$ that is in $\{a_1, \ldots, a_{3q}\}$ has a distance of at least 2 from every other node in $\{a_1, \ldots, a_{3q}\}$, so every such j contributes an amount at least $t + 2(3q-1) = 6q + t - 2 > d$ to the sum $A(R)$. Therefore:

$$A(R) \geq rd \tag{6.8}$$

Now partition V into three sets of sizes n_0, n_1, and n_2 corresponding to those nodes of distance 0, 1, and 2 or more from R, respectively. Then $n_0 = |R| = r$ and $n_0 + n_1 + n_2 = |V| = 3q + t$, so $n_1 = 3q + t - r - n_2$. Also $B(R) \geq n_1 + 2n_2 = (3q + t - r - n_2) + 2n_2$. Therefore:

$$B(R) \geq 3q + t - r + n_2 \tag{6.9}$$

We consider three cases, and show in each case that $cost(R, u) > k$. Since $R \subseteq V$ is arbitrary, it follows that $mincost(u) > k$.

CASE I: $r = |R| < q$

Every node in $\{a_1, \ldots, a_{3q}\}$ has a distance of 2 or more from R, except those nodes in R or in one of the 3-element subsets corresponding to nodes in R. Thus $n_2 \geq 3q - 3r$, so by (6.9):

$$B(R) \geq 3q + t - r + 3q - 3r = 6q + t - 4r$$

By (6.8):

$$\begin{aligned} cost(R,u) = uA + (1-u)B &\geq \frac{2rd}{d+2} + \frac{d(6q+t-4r)}{d+2} \\ &= \frac{d}{d+2}(6q - 2r + t) > \frac{d}{d+2}(4q+t) = k \end{aligned}$$

CASE II: $r > q$

By (6.9), $B(R) \geq 3q + t - r$, so:

$$\begin{aligned} cost(R,u) = uA + (1-u)B &\geq \frac{2rd}{d+2} + \frac{d(3q+t-r)}{d+2} \\ &= \frac{d(3q+r+t)}{d+2} > \frac{d(4q+t)}{d+2} = k \end{aligned}$$

CASE III: $r = q$

By assumption the instance of $X3C$ has no solution, so at least one node of $\{a_1, \ldots, a_{3q}\}$ has a distance of 2 or more from R, so $n_2 \geq 1$. By (6.9),

$$B(R) \geq 3q + t - r + 1 = 2q + t + 1 \ .$$

Thus by (6.8):

$$\begin{aligned} cost(R,u) = uA + (1-u)B &\geq \frac{2qd}{d+2} + \frac{d(2q+t+1)}{d+2} \\ &= \frac{d(4q+t+1)}{d+2} > k. \end{aligned}$$

□

7 Bounds for the function *mincost*

7.1 Number of linear segments

For each replica set $R \subseteq V$, the graph of $cost(R, u)$ is a straight line whose Y-intercept $b = B(R)$ and whose slope $m = A(R) - B(R)$ are non-negative, as in equations (3.4), (3.5), (3.6). See Fig. 1.

For each Y-intercept b, a finite number of lines pass through the point $(0, b)$. There will be a line of least slope, which corresponds to some particular choice or choices of replica positions, and only this one can be part of the function $mincost(u)$. There will also be a lowest horizontal line (of slope zero), which is the result of only having a single replica. The Y-intercept $b = b_1$ of this line is the minimum of all the column sums of D. When the matrix D consists of non-negative integers only, since $m \geq 0$ there can be at most $1 + b_1$ lines that are part of the lower bound of the solution set, with $b = 0, 1, 2, \ldots, b_1$. Thus we have proved:

Theorem 2. *If all the elements of the cost matrix D are non-negative integers, there are at most $1 + b_1$ linear components in the piecewise linear function $mincost(u)$, where b_1 is the minimum of all the column sums of D.* □

Corollary 1. *If the cost matrix is the hop distance matrix D of a graph with n vertices, there are at most $1 + n^2/4$ linear components in the piecewise linear function mincost(u).*

Proof. For a path graph with n nodes, $b_1 = \frac{n^2-1}{4}$ when n is odd, and $b_1 = \frac{n^2}{4}$ when n is even. The column sum is minimized for the column corresponding to the midpoint (or one of the two midpoints) of the line. To see that the path graph is the worst case, first note that b_1 for any connected graph is at most b_1 for any of its spanning trees. Given tree T with n nodes, let P be a longest path in T (say P has p nodes), and let c be one of the midpoints of P. Note that c has distance at most $p/2$ from any node in T. We define a bijection from the nodes of T to the nodes of an n-node path graph L by first embedding the line P into the line L such that c is mapped to one of the midpoints c' of L, and the remaining nodes of T are mapped arbitrarily to the remaining nodes of L. The bijection has the property that the distance of any node x in T from c is at most the distance of the image of x in L from c'. Hence the sum of the distances of all nodes in T from c is at most the corresponding sum in L for c'. □

The bound is achieved on a graph with two vertices and one edge, with distance $d(1, 2) = 1$ between the two vertices. The components are $y = 2u$ for $0 \le u \le 0.5$, and $y = 1$ for $0.5 \le u \le 1$.

7.2 Size of the optimal replica set

The following bound is mentioned by Dowdy and Foster [2]. For the reader's convenience we include a simple proof here.

Theorem 3. *If $u > 0$ and R is optimal for u, then $|R| \le 1/u$.*

Proof. Consider the minimal cost of one-element replica sets:

$$\nu \;=\; \min_{j \in V}\ cost(\{j\}, u) \;=\; \min_{j \in V} \sum_{i \in V} d(i, j)$$

There exists $j_0 \in V$ such that $cost(\{j_0\}, u) = \nu$. Thus if $R \subseteq V$ and u $(0 \le u \le 1)$ are such that R is optimal for u then:

$$\nu \ge cost(R, u) \ge u \sum_{j \in R} \sum_{i \in V} d(i, j) \ge u\ |R|\ \nu$$

Therefore $|R| \le 1/u$. □

For example, in an environment in which the update traffic is one-tenth of the total volume, $u = .1$ and therefore $|R| \le 10$.

8 A counterexample for monotonicity

Question 4 in Section 5 asks whether the size of an optimal R is monotonically non-increasing as a function of u. We now show that the answer is no if one allows arbitrary cost matrices. Consider the following 4×4 matrix D:

$$D = \begin{bmatrix} 10 & 10 & 60 & 60 \\ 200 & 30 & 5 & 30 \\ 200 & 30 & 5 & 30 \\ 2 & 60 & 60 & 10 \end{bmatrix}$$

For $0 \le u < \frac{1}{20}$, the optimal solution is $mincost(u) = 520u + 22$ and the corresponding optimal replica set is $\{1, 3\}$. For $\frac{1}{20} < u < \frac{5}{18}$, $mincost(u) = 360u + 30$, with the replica set $\{2, 3, 4\}$. This is a counterexample to the monotonicity question because at $u = \frac{1}{20}$ the size of the optimal replica set increases from 2 to 3.

The monotonicity question remains open in case the cost matrix D represents shortest-path distances in a graph.

9 An algorithm for computing the *mincost*(u) function

The piecewise linear function $mincost(u)$, $0 \le u \le 1$, defined in (5.7) can be constructed by the following algorithm.

THE *mincost* **ALGORITHM**

1. Form a list of all straight lines $L(u) = cost(R, u)$ corresponding to all possible choices R of replica sets. Sort the lines in increasing order of Y-intercept.
2. For each set of lines with the same Y-intercept, select the one with the least slope and delete the others.
3. Successively compute points $0 = u_0 < u_1 < \cdots < u_t = 1$ and select lines $L_0, L_1, \ldots, L_{t-1}$ from the list such that $mincost(u) = L_i(u)$ for $u_i \le u \le u_{i+1}$, $0 \le i < t$, according to the following procedure:
   3a) Let $u_0 = 0$ and let L_0 be the first line (with smallest Y-intercept) on the list. Remove L_0 from the list under consideration.
   3b) In general, assume $u_0, \ldots, u_i$ are correctly computed and $L_0, \ldots, L_i$ are correctly selected, where $L_i(u) = m_i u + b_i$. Further assume that each remaining line $L(u) = mu + b$ satisfies $b > b_i$ and either $m \ge m_i$ (in which case delete L from the list because its slope is too large) or $m < m_i$ and L intersects the last line selected, L_i, at a point with u-value $u_L > u_i$. Select L so that u_L is as small as possible. Among lines L with the same value u_L, select the one with least slope and delete the others. Set $u_{i+1} = u_L$ and $L_{i+1} = L$. Delete all lines with a smaller Y-intercept than L_{i+1} from the list, and also remove L_{i+1}. If the slope of L_{i+1} is 0, set $t = i + 2$, $u_t = 1$, and stop. Otherwise repeat 3b.

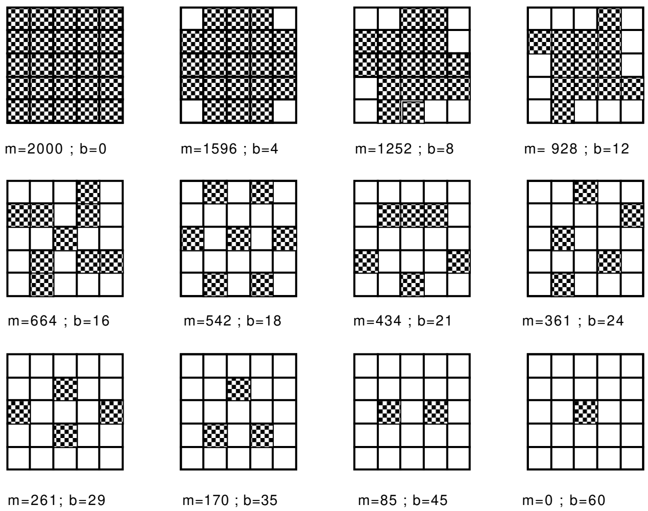

Fig. 2. Optimal placements of replicas in a 5×5 array.

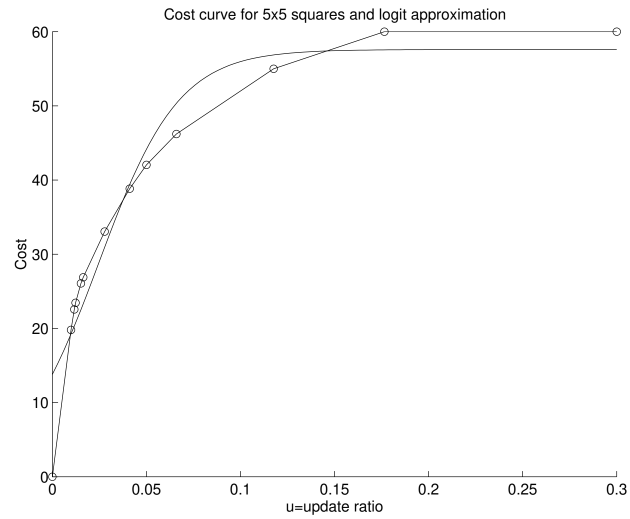

Fig. 3. *mincost*(u) for the 5×5 array.

Step 1 in the algorithm requires time of order 2^n. The time required for steps 2 and 3 is polynomial in the number of distinct lines (for step 3, the number of distinct lines remaining after step 2). Note that if the network is a graph with cost matrix D determined by the hop distance, the number of distinct lines is bounded by a polynomial in n, as follows from Section 7.1.

In view of the theorem in section 6, the problem of computing the function *mincost* on the interval $(0, 1)$ is NP-complete for general graphs. However, it follows from Theorem 3 that for each fixed u an optimal replica set R can be found and *mincost*(u) computed in polynomial time. More precisely, let u_0 be fixed, $u_0 > 0$. The function *mincost*(u) can be computed on the interval $(u_0, 1)$ by considering only the replica sets R of size at most $1/u_0$ in the *mincost* algorithm above. Since there are $O(n^{(1/u_0)})$ such sets R, the running time of the algorithm is $O(n^{1+(1/u_0)})$.

An alternative to step 3 is to apply the simplex algorithm to the following linear program:

$$\begin{array}{ll} \textit{maximize} & y \\ \textit{subject to:} & y - mx - b \leq 0 \\ & \quad \text{for each line } L(x) = mx + b \text{ on the list} \\ & y \geq 0 \\ & x \geq 0 \end{array}$$

The starting solution is $x = 0$, $y = 0$. The constraint $y = b_1$ corresponding to the 1-median solution gives the optimal solution b_1 after a finite number of iterations of the simplex algorithm. We have a sequence of pivots and constraints that comprise the piecewise linear function *mincost*(u). This usage of linear programming is unusual, but it was easy to implement in simulation studies.

Example 2. The algorithm described above was implemented in APL on a RS/6000 system[1] to find the optimal configurations of replicas in a 5 by 5 array. The nodes of the graph are the pairs (i_1, i_2) where $i_1, i_2 \in \{1, 2, 3, 4, 5\}$ and the distance is $d((i_1, i_2), (i'_1, i'_2)) = |i_1 - i'_1| + |i_2 - i'_2|$. This is the L^1 metric where two horizontally or vertically adjacent squares are deemed to be at distance one apart. The $2^{25} - 1$ possible replica configurations R were exhaustively examined to find the cost line for each possible configuration. The set of lines retained was periodically filtered (for example, as suggested in Step 2), because of storage considerations. The effort was distributed among parallel copies of the workspace which interacted periodically with a common file where the current "best" set was kept. Since the Y-intercept b had a maximal value of 60 and all values of m and b were integral, there were at most 61 elements in this file for the values $0 \leq b \leq 60$.

After all possible choices had been examined, the simplex algorithm described was used to reduce the final best set (of 42 configurations) to the final optimal set of twelve configurations in Fig. 2. The shaded squares represent the location of the replicas, m is the slope of the cost line for this configuration, and b is the Y-intercept. It should be noted that there were often two or more possible configurations for each final value of m and b because of the possibility of rotating and reflecting the diagrams, and because there were multiple choices which were in fact essentially different. The final set of configurations are presented in order of their occurrence in the piecewise linear *mincost*(u) curve. Note that the steep slopes indicate that the intersections mostly occur at quite small values of u. For example, the 8th and 9th lines intersect at $u = 0.05$.

The piecewise linear cost curve for this problem is obtained by finding the successive intersections of the lines with the slopes and y-intercepts as given in Fig. 2. We examined various models to fit this data and found that logistic models gave better results than others that we considered (e.g., arctangent and hyperbolic tangent). The line segments and their intersections are plotted in Fig. 3 together with the smooth approximation us-

[1] RS/6000 is a registered trademark of International Business Machines Corporation.

ing the curve $y = 1/(A + B * exp(C * x))$ with values $A = 0.017363, B = 0.055056, C = -46.898914$. We determined these coefficients by least squares regression using the MATLAB® function *fminunc*.

10 An efficient algorithm for path graphs

The previous section describes an algorithm for computing the function $mincost(u)$. Since step 1 deals with 2^n lines, the running time is exponential in the size of input. Note that the algorithm does more than merely compute the value of $mincost(u)$ for a given value u. The algorithm constructs the complete description of $mincost(u)$ on the interval $(0, 1)$, as a collection of linear segments.

Section 6 shows that the problem of computing $mincost(u)$ for an arbitrary given value of u is NP-complete in the class of all undirected graphs with costs given by the hop distance. It follows that the problem of constructing the description of $mincost(u)$ on $(0, 1)$ is also NP-complete in the same class of graphs. We now show that this problem is tractable in the class of path graphs with integer distances.

Consider the undirected graph with the set $V = \{1, 2, \ldots, n\}$ of vertices arranged in order on a line and edges $(i, i + 1)$, $i = 1, 2, \ldots, n - 1$. Let each edge have a positive integer length, and let $d(i, j)$ be the length of the shortest path from i to j. For $R \subseteq V$ define the mapping $\alpha : V \longrightarrow R$ so that, for all i, $\alpha(i)$ is a vertex in R nearest to i. We use the α notation to restate the total read cost of (3.6)

$$B(R) = \sum_{i\in V} \min_{j\in R} d(i, j) = \sum_{i\in V} d(i, \alpha(i))$$

It will be technically convenient to consider more general mappings α: For $k = 1, \ldots, n$, denote by $F(k)$ the set of all mappings $\alpha : \{1, \ldots, k\} \longrightarrow \{1, \ldots, k\}$ that are monotonically non-decreasing and idempotent. That is,

$$1 \le i \le j \le k \Longrightarrow \alpha(i) \le \alpha(j)$$
$$1 \le i \le k \Longrightarrow \alpha(\alpha(i)) = \alpha(i)$$

An example of such a mapping is $(2, 2, 4, 4, 4, 6, 6)$ for $k = 7$. This means that vertices 1 and 2 are mapped to 2; 3, 4, and 5 are mapped to 4; and 6 and 7 are mapped to 6.

For $\alpha \in F(k)$, let $R(\alpha)$ be the range of α. Each α divides the interval $\{1, \ldots, k\}$ into subintervals on which α is constant; namely, the subintervals $\alpha^{-1}(j)$, $j \in R(\alpha)$. Moreover, $j \in \alpha^{-1}(j)$ (because α is idempotent). For $\alpha \in F(n)$, define

$$cost(\alpha, u) = u \sum_{j\in R(\alpha)} \sum_{i\in V} d(i, j) + (1 - u) \sum_{i\in V} d(i, \alpha(i)). \qquad (10.10)$$

This generalizes the definition (3.3) of the cost function $cost(R, u)$ in the following sense: if $\emptyset \ne R \subseteq V$ and the mapping $\alpha_R : V \longrightarrow R$ is defined so that $\alpha_R(i)$ is a vertex in R nearest to i for all i, then $cost(R, u) = cost(\alpha_R, u)$.

Although not every $\alpha \in F(n)$ is of the form α_R, in computing the minimum of $cost(\alpha, u)$ over $\alpha \in F(k)$ it is sufficient to consider only α_R. Formally:

Lemma 1.

$$\min_{\emptyset\ne R\subseteq V} cost(R, u) = \min_{\alpha\in F(n)} cost(\alpha, u)$$

Proof. As noted above, for every $R \subseteq V$, $R \ne \emptyset$, we have $cost(\alpha_R, u) = cost(R, u)$. In addition, for every $\alpha \in F(n)$ and $i \in V$ we have $d(i, R(\alpha)) \le d(i, \alpha(i))$, so $cost(R(\alpha), u) \le cost(\alpha, u)$. □

For $k = 1, 2, \ldots, n$, define the following set of ordered pairs, whose components correspond to the coefficients of u and $1 - u$ in (10.10):

$$W_k = \left\{ \left(\sum_{j\in R(\alpha)} \sum_{i=1}^{n} d(i, j) , \sum_{i=1}^{k} d(i, \alpha(i)) \right) \middle| \ \alpha \in F(k) \right\}$$

Let d be the diameter of the graph. Because $d(i, j) \le d$ and all the pairs in W_k have integer coordinates, the size of W_k is less than nk^2d^2.

Once we determine the set W_n, we can modify the algorithm of section 9 to compute the function

$$\min_{\alpha\in F(n)} cost(\alpha, u)$$

on the $0 \le u \le 1$ interval. By Lemma 1 this gives the function $mincost(u)$. The modification changes step 1 of the algorithm in section 9 to form a list of the lines $L(u) = cost(\alpha, u)$ for $\alpha \in F(n)$ (instead of $L(u) = cost(R, u)$ for all R). Since there are at most $|W_n| \le n^3d^2$ such lines, our entire procedure runs in time polynomial in d.

We now give a recursive definition of W_k for $k \ge 2$, which can be converted to an algorithm for computing W_k. To see how $F(k)$ can be constructed from the sets $F(s)$, $s < k$, consider any $\alpha \in F(k)$ and let t be the largest element in the range $R(\alpha)$ of α.

If t is not the only element in $R(\alpha)$, then there is some s, $1 \le s < t \le k$, such that

$$\alpha(i) \le s \quad \text{for} \quad i = 1, \ldots, s$$
$$\alpha(i) = t \quad \text{for} \quad i = s + 1, \ldots, k$$

The restriction α' of α to the interval $\{1, \ldots, s\}$ belongs to $F(s)$. Thus in this case α is completely determined by s, t and α' such that $1 \le s < t \le k$ and $\alpha' \in F(s)$. On the other hand, if t is the only element in $R(\alpha)$ then α is completely determined by t.

Hence $F(k)$ is constructed from the sets $F(s)$, $s < k$, as follows:

$$F(k) = \bigcup_{s=1}^{k-1} \bigcup_{t=s+1}^{k} \{ \alpha : \{1,2,\ldots,k\} \longrightarrow \{1,2,\ldots,k\} \mid (\exists \alpha' \in F(s) : \alpha(i) = \alpha'(i) \text{ for } 1 \le i \le s) \text{ and } (\alpha(i) = t \text{ for } s < i \le k) \} \cup \bigcup_{t=1}^{k} \{ \alpha \mid \alpha(i) = t \text{ for } 1 \le i \le k \}$$

Putting this together with the definition of W_k, we obtain:

$$W_k = \bigcup_{s=1}^{k-1} \bigcup_{t=s+1}^{k} \left\{ \left(x + \sum_{i=1}^{n} d(i,t)\,,\ y + \sum_{i=s+1}^{k} d(i,t) \right) \,\middle|\, (x,y) \in W_s \right\} \cup \bigcup_{t=1}^{k} \left\{ \left(\sum_{i=1}^{n} d(i,t)\,, \sum_{i=1}^{k} d(i,t) \right) \right\}$$

To simplify, define $W_0 = \{(0,0)\}$ and rewrite W_k as:

$$W_k = \bigcup_{s=0}^{k-1} \bigcup_{t=s+1}^{k} \left\{ \left(x + \sum_{i=1}^{n} d(i,t)\,,\ y + \sum_{i=s+1}^{k} d(i,t) \right) \,\middle|\, (x,y) \in W_s \right\}$$

We have proved the following theorem.

Theorem 4. *For the path graph with integer distances and the cost $d(i,j)$ given by the length of the shortest path, the function mincost(u) on (0,1) can be constructed in time polynomial in the diameter of the graph.*$\Box$

The example in Section 5 shows the function *mincost* and the optimal replica sets for a path graph with 12 nodes and hop distances.

As we explained above, our focus is the behavior of the function *mincost*(u) on the interval $(0,1)$ and thus we wish to construct the function on the whole interval. If the objective is simply to compute *mincost*(u) for one input value of u then more efficient algorithms are known, even for tree graphs [9]. We do not know whether any of those algorithms can be generalized to yield an analogue of Theorem 4 for tree graphs.

11 Conclusions

In this paper we have addressed the five questions that were raised in Section 5. A bound for the size of the optimal replica set is given in Theorem 3, which provides a partial answer to question 3. Question 4 is still open regarding whether or not the size of an optimal replica set decreases monotonically with increasing u for a graph with a distance metric, but some examples demonstrate that it is false for arbitrary cost matrices as shown in Section 8. A partial answer to question 2, regarding the number of linear pieces making up the function *mincost*(u), is given in Section 7.1, in which the cost matrix is integral. The problem raised in question 1 is proven to be NP-complete in Section 6, even when the cost matrix is constrained to represent the hop distance in a connected graph. However, Theorem 4 provides an efficient algorithm when the graph is a straight line. Question 5, whether there is an efficient algorithm for approximating *mincost*(u), remains open.

References

1. Chuang JC, Sirbu MA: Distributed network storage with quality-of-service guarantees. Journal of Network and Computer Applications 23: 163-185 (2000)
2. Dowdy LW, Foster DV: Comparative models of the file assignment problem. ACM Computing Surveys 14(2): 287-313 (1982)
3. Erkut E, Francis RL, Lowe TJ, Tamir A: Equivalent mathematical programming formulations of monotonic tree network location problems. Oper. Res. (USA) 37(3): 447-461 (1989)
4. Fisher ML, Hochbaum DS: Database location in computer networks. Journal of ACM 27(4): 718-735 (1980)
5. Frederickson GN: Parametric search and locating supply centers in trees. Proc. WADS'91, Workshop on Algorithms and Data Structures (August 1991, Ottawa, ON), pp 299-319. Lecture Notes in Computer Science 519, Berlin Heidelberg New York: Springer 1991
6. Garcia-Molina H: The future of data replication. Proc. IEEE Symposium on Reliability in Distributed Software and Database Systems (January 1986, Los Angeles, CA), pp 13-19
7. Garey MR, Johnson DS: Computers and intractability. New York: W.H. Freeman and Company 1979
8. Gifford DK: Weighted voting for replicated data. Proc. ACM SOSP'79, Symposium on Operating Systems Principles (November 1979, Pacific Grove, CA), pp 150-162
9. Hakimi SL, Schmeichel EF: Locating replicas of a database on a network. Networks 30(1): 31-36 (1997)
10. Huang Y, Sistla P, Wolfson O: Data replication for mobile computers. Proc. ACM SIGMOD'94, International Conference on Management of Data (May 1994, Minneapolis, MN), pp 13-24
11. Kariv O, Hakimi SL: An algorithmic approach to network location problems. I: The p-centers. SIAM J. Appl. Math. 37(3): 513-538 (1979)
12. Kariv O, Hakimi SL: An algorithmic approach to network location problems. II: The p-medians. SIAM J. Appl. Math. 37(3): 539-560 (1979)
13. Kemme B, Alonso G: A suite of database replication protocols based on group communication primitives. Proc. IEEE ICDCS'98, International Conference on Distributed Computing Systems (May 1998, Amsterdam, The Netherlands), pp 156-163
14. Mahmoud S: Private communication. *(January 2001)*

15. Milo A, Wolfson O: Placement of replicated items in distributed databases. Proc. EDBT'88, International Conference on Extending Database Technology (March 1988, Venice, Italy), Lecture Notes in Computer Science 303, pp 414-427. Berlin Heidelberg New York: Springer 1988
16. Mirchandani PB, Francis RL (Eds): Discrete location theory. New York: John Wiley and Sons 1990
17. Pressman IS, Cook SA, Pachl J: The optimal placement of replicas in a network using a read any - write all policy. Proc. IBM CASCON'92 (November 1992, Toronto, ON), vol II, pp 189-201
18. Wolfson O, Jajodia S, Huang Y: An adaptive data replication algorithm. ACM Transactions on Database Systems 22(4):255-314 (1997)
19. Wolfson O, Jajodia S: Distributed algorithms for dynamic replication of data. Proc. ACM PODS'92, Symposium on Principles of Database Systems (June 1992, San Diego, CA), pp 149-163
20. Wolfson O, Milo A: The multicast policy and its relationship to replicated data placement. ACM Transactions on Database Systems 16(1):181-205 (1991)