February 28, 2001 TR CS-01-03

Data Replication on Networks with Bounded Treewidth

Konstantinos Kalpakis[1,2] and Koustuv Dasgupta[1]

Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, Maryland 21250

[1]Email: {kalpakis,dasgupta}@csee.umbc.edu

[2]Supported in part by the National Science Foundation under grant number IRI–9729495, and by NASA under Cooperative Agreement NCC5–315.

Data Replication on Networks with Bounded Treewidth

Konstantinos Kalpakis[1,2] and Koustuv Dasgupta[1]

Abstract

We consider the problem of placing copies of an object in a distributed system, in order to minimize the cost of servicing read and write requests to object, together with the cost of storing those copies, when the number of copies allowed is given. The network is modeled by an undirected weighted graph. The set of nodes that have a copy of the object, called replica nodes, constitute the replica set of the object. Read requests of a node are serviced from the closest replica node. Write requests of a node are forwarded to the closest replica node. Replica nodes propagate write requests to all the other replica nodes using a minimum spanning tree of the distance graph that spans the replica set. The cost associated with a replica set equals the cost of servicing all the read and write requests, plus the storage cost at all the replica nodes. We are interested in finding a replica set with minimum cost, i.e. an optimal replica set.

Given a graph G with n nodes and treewidth bounded by a constant t, and an integer k, we provide a $O(n^{2t+5})$–time algorithm for finding an optimal replica set with $\leq k$ replicas, taking into consideration the read, write, and storage costs. Our algorithm can also find optimal replica sets with $\geq k$ replicas, a requirement that may be imposed due to reliability considerations.

Keywords: data replication, distributed systems, multicasting, facility location, p–medians, file allocation, bounded treewidth.

1 Introduction

Data replication can improve the performance and availability of a distributed information system. The recent growth in the World-Wide-Web (WWW) is rapidly moving us towards highly distributed and wholly interconnected information systems. In such systems, a data object is accessed, i.e. read and written from multiple locations that are geographically

[1]Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: kalpakis@csee.umbc.edu

[2]Supported in part by the National Science Foundation under grant number IRI–9729495, and by NASA under Cooperative Agreement NCC5–315.

distributed world-wide. For example, in electronic publishing a document(e.g. a newspaper, an article or a book) may be co-authored and read by users in a distributed fashion. Time-sensitive documents like weather forecasts and stock market prices are read and updated from all over the world. Raw data (like those on scientific experiments or satellite data) are used and updated by numerous laboratories. Even images (e.g. X-rays, road maps) are read and annotated by users spread world-wide. In the mobile computing environments of the future, an identification will be associated with each user rather than a physical location. The location of the user needs to be updated as a result of the user's mobility, and it will be read on behalf of the callers. The replication of objects has a critical implications for the overall performance and availability of such systems.

The growing popularity of the World-Wide-Web over the last decade has added a whole new dimension to our understanding of a distributed information system. The Web can be easily viewed as the largest distributed computing resource with millions of bytes of repository data. Reports show that the percentage of network traffic generated due to HTTP requests has gone up from 19% in 1994 to a significant 40% in 1996 (see [6]). There is common agreement that with the Web, we are currently experiencing an exponential growth in demand. The proliferation of new information services and the enormous potential for commercial use of the Internet has led to a number of problems - overloaded servers, traffic congestion and increasing client latencies. As pointed about by Seltzer and Manley [16], the answer to the question "Why do users wait on the Web?" still eludes the research community. In an environment like this (where performance and cost factors are of crucial importance) it is only fitting that a great deal of research has been conducted to devise efficient schemes that guarantee faster and better services on the Internet. Specifically, replication (or *mirroring*) of web servers has been commonly used to address these problems of scale (Rabinovich and Aggarwal [15]). Distributing client requests among replicas is an interesting issue, since it deals with the two (often conflicting goals) of increasing client-server proximity and load distribution. Note that caching is often used to address this problem of replication (Cao and Irani [6], Heddaya and Mirdad [10]), being sometimes advantageous and at other times detrimental. Caching creates or deletes replicas at a client, as need dictates. We consider caching to be a particular case of data replication in the sense that it is usually analyzed in a client/server architecture and with respect to a given consistency model, in order to improve performance at a client. Data replication is a more general concept, that can be applied to richer architectures, for multiple consistency models, and for optimizing different cost functions.

In this paper, we consider the problem of placing copies of an object at multiple locations in a distributed system, in order to minimize the total cost of servicing the read/write requests. The interconnection network of the system is modeled by an undirected graph G. Consider an object o. Each node of the system issues a number of read requests and a number of write requests to the object. Let S be a subset of the nodes of the graph at which there exist copies (replicas) of the object. The subset of nodes S is called a replica set. Assume that each node of the graph sends its read/write requests to the closest node that

has a copy of the object, and that each write to a copy of the object must be propagated to all other copies of the object. We assume that only nodes that have a replica of the object can propagate write requests to other replicas. The write propagation protocol we use is the following: each writer node sends its request to its closent replica node; writes are broadcasted to all other replica nodes using a minimum spanning tree of a certain graph, the subgraph of the complete distance graph of G that spans the replica set. The model we use in this paper is based on the model of Wolfson and Milo [19]. The cost of servicing a read request equals the distance of the requesting node from its closest replica. The cost of servicing a write request equals the distance of the requesting node from its closest replica together with the cost of the minimum spanning tree used by the write propagation protocol. Further, we assume that having a copy of the object at a node has an associated storage cost that depends on the object and that node. The cost of a replica set is equal to the total read, write, and storage costs. We are interested in finding a replica set of minimum cost with at most k replicas, for a given integer k. This is the optimal replica set problem with storage costs.

The optimal replica set (file allocation) problem has been studied extensively in the literature. Dowdy and Foster [7] survey a number of mixed linear programming models for the file allocation problem. Wolfson and Milo [19] consider the optimal replica set problem without storage costs for various interconnection networks (completely–connected, tree, and ring networks). They show that the optimal replica set problem without storage costs is NP–hard for general topologies, and they provide efficient algorithms for finding optimal replica sets: a $O(n)$–time algorithm for tree networks, a $O(n^5)$–time algorithm for ring networks, and a $O(1)$–time algorithm for a completely connected network, where n is the number of nodes in the network. Our model is different from that of Wolfson and Milo [19] since we also consider storage costs. Fisher and Hochbaum [8] consider the problem of database location in computer networks (a problem similar to the one of this paper), and describe computational experiments based on mixed integer programming models. Our model is different from that of [8], since they assume what Wolfson and Milo [19] call a "naive–write policy" (i.e. the write cost is equal to the sum of the distances between the writer and each one of the nodes with a copy of the object).

When there are no write–costs, our problem reduces to the (uncapacitated) facility location problem which also has been studied extensively due to its applications (see [11, 9, 17]). When there are only read–costs, our problem reduces to the k–median problem [20, 11, 13]. Hochbaum [11] describes approximation algorithms for finding k–medians on the plane (discrete and continuous) as well as on networks. Arora et al [20] describe randomized polynomial approximation schemes for finding k–medians in Euclidean spaces. Kariv and Hakimi [13] show that the k–median problem is NP–hard for general networks, and they provide an $O(n^2k^2)$–time algorithm for finding a k–median for a tree with n nodes. Megiddo and Supowit [14] show that the k–median problem is NP–hard, when considering the geometric version of the problem, i.e. nodes are points on the plane, with either the Euclidean or the Rectilinear metrics.

Kalpakis et al [12], describe a $O(n^3k^2)$–time dynamic programming algorithm for finding an optimal replica set of size k for an object on a tree with n nodes when there are read, write, and storage costs associated with storing replicas at nodes of the tree. The model used in this paper is the model in [12].

The notion of treewidth plays an important role in recent advances in algorithmic graph theory, because many problems that are otherwise NP–hard become polynomial time solvable when restricted to graphs of bounded treewidth (see Bodlaender [4] and references therein). We use treewidth–based techniques to tackle the optimal replica set problem with storage costs on graphs of bounded treewidth.

In this paper, we provide the first polynomial time algorithm for finding an optimal replica set for graph of bounded treewidth. In particular, we describe a $O(n^{2t+5})$–time algorithm for finding an optimal replica set of size $\leq k$, assuming that the graph G has treewidth bounded by a constant t. Our algorithm can also find optimal replica sets with $\geq k$ replicas, a requirement that may be imposed due to reliability considerations. Our algorithm is the first polynomial time algorithm for the optimal replica set problem with storage costs for non–tree graphs. Our algorithm is based on a technique, introduced by Arnborg and Proskurowski [1] and Bern [2], described by Bodlaender [4]. Table 1 shows some classes of graphs of bounded treewidth, i.e. classes for which we can solve the optimal replica set problem efficiently.

The rest of the paper is organized as follows. In section 2, we provide various preliminaries definitions and notations. Then, we introduce the notion of canonical replica assignments in section 3, and the notion of canonical minimum spanning trees in section 4. We introduce the notion of terminal–restrictions of a tree to a terminal graph, and the notion of canonical minimum spanning forests in section 5. Then, in section 6, we analyze the structure of canonical minimum spannign forests. In section 7, we characterize replica assignments using canonical minimum spanning forests. Finally, we describe and analyze our algorithm for finding optimal replica sets for graphs of bounded treewidth in section 8.

2 Preliminaries

Consider an edge–weighted undirected graph $G = (V, E)$ with n vertices.

The vertices of G issue read and write requests for an object. Copies of that object can be stored at multiple vertices of G. Let $r, w : V \rightarrow N$ be two vertex–weight functions representing the number of read requests and the number of write requests issued by vertices of G, respectively. Let $\alpha : V \rightarrow N$ be a function representing the number of requests (read and write) issued by the vertices of G, i.e. $\alpha(v) = \mathrm{r}(v) + \mathrm{w}(v)$ for all $v \in V$. Let $s : V \rightarrow R_+$ be a vertex-weight function representing the storage cost of placing a replica of the object at vertices of G.

Table 1: Some classes of graphs of bounded treewidth [18].

Class of graphs	Bound on treewidth
trees/forests	1
almost trees (t)	$t+1$
partial t–trees	t
bandwidth–t	t
cutwidth–t	t
planar with radius t	$3t$
series parallel	2
outerplanar	2
Halin	3
t–outerplanar	$3t-1$
chordal with maximal clique size t	$t-1$
undirected path with maximal clique size t	$t-1$
directed path with maximal cliqeue size t	$t-1$
interval with maximal clique size t	$t-1$
proper interval with maximal clique size t	$t-1$
circular arc with maximal clique size t	$2t-1$
proper circular with maximal clique size t	$t-2$

The distance $\delta(u,v)$ between any two vertices u and v of G is defined as the length of a shortest path from u to v in G. The distance of a vertex $v \in G$ from any subset $S \subseteq V$ is $\delta(v,S) = \min\{\delta(v,u) : u \in S\}$. The distance between two subsets $X, Y \subseteq V$ is $\delta(X,Y) = \min\{\delta(u,v) : u \in X, v \in Y\}$.

The distance graph $\Delta[V'] = (V', V' \times V')$ for $V' \subseteq V$ is an edge–weighted complete undirected graph, such that the weight of each edge (u,v) of $\Delta[V']$ is equal to the distance $\delta(u,v)$ of u from v in G. Let $\mathrm{MST}(V')$ be the cost of a minimum spanning tree of $\Delta[V']$.

A partition of a set $S \subseteq V$ is a collection Π of non-empty disjoint subsets of S, called *blocks*, such that each element of S is contained in one block of the partition.

A partition Π' is a *refinement* of a partition Π if (a) Π' and Π are partitions of the same set, and (b) each block of Π' is a subset of some block of Π. A partition Π' is the *restriction* of a partition Π to a set S' if $\forall \pi' \in \Pi',\ \exists \pi \in \Pi,\ \pi' = \pi \cap S'$. Given two partitions Π_1 and Π_2 of a set S, we define the *closure* $\Pi_1 \otimes \Pi_2$ to be the partition of S that results by (repeatedly) combining (merging) blocks from $\Pi_1 \cup \Pi_2$ that have common elements. Note that both Π_1 and Π_2 are refinements of $\Pi_1 \otimes \Pi_2$.

Every undirected graph G' induces a partition Π for any subset S of its vertices in a natural way: two vertices in S belong to the same connected component of the graph if and

only if they belong to the same block of the partition. We call Π the partition of S *induced* by G'.

With each partition Π of a set $S \subseteq V$ we associate an edge–weighted graph G[Π] as follows. G[Π] is a complete graph that has one vertex for each block of Π. The weight of the edge (π_i, π_j) between two vertices is equal to the distance $\delta(\pi_i, \pi_j)$ between the two sets of vertices π_i and π_j in G. We denote the cost of a minimum spanning tree of G[Π] with MST(Π).

2.1 Replica Assignments

A replica assignment for G is a function $\sigma : V \to V$, that assigns to each vertex in V a vertex in V such that $\forall v \in V$, $\delta(v, \sigma(v)) = \delta(v, \sigma(V))$. For brevity, we denote the range of σ with $\sigma(V)$. The replica set of σ is defined to be $\sigma(V)$. A replica assignment σ is a k–replica assignment if $|\sigma(V)| = k$. We say that u covers (is assigned to) v if $\sigma(v) = u$. We say that v is covered by (is assigned) u if $\sigma(v) = u$. A replica assignment σ is *compatible* with a set $S \subseteq V$ if $\sigma(V) = S$ and $\forall v \in V$, $\delta(v, \sigma(v)) = \delta(v, S)$. Given S it is easy to find a replica assignment σ for G that is compatible with S. We assume, w.l.o.g, that $\sigma(v) = v$ for all $v \in \sigma(V)$. The *restriction* of a replica assignment $\sigma : V \to V$ to $V' \subseteq V$ is a replica assignment $\sigma' : V' \to V$ such that $\sigma'(v) = \sigma(v)$ for all $v \in V'$. A replica assignment σ is an *extension* of σ' if the restriction of σ to the domain of σ' is σ'.

Let $\sigma : V' \to V$ be a replica assignment for a subgraph $G' = (V', E')$ of G. The *access cost* of σ is defined as

$$C_a(\sigma) = \sum_{v \in V'} \alpha(v) \cdot \delta(v, \sigma(v)) = \sum_{v \in V'} (\mathrm{r}(v) + \mathrm{w}(v)) \cdot \delta(v, \sigma(v)). \tag{1}$$

The access cost of σ is the total cost of sending the requests from each vertex in the domain of σ to its assigned vertex. The *update cost* of σ is defined as

$$C_u(\sigma) = W_{\text{total}} \cdot \mathrm{MST}(\sigma(V')), \tag{2}$$

where $W_{\text{total}} = \sum_{v \in V} \mathrm{w}(v)$. The update cost of σ equals the cost of broadcasting all the writes issued by the vertices of G to the vertices in the replica set (range) of σ. The *storage cost* of σ is defined as

$$C_s(\sigma) = \sum_{v \in \sigma(V')} \mathrm{s}(v). \tag{3}$$

The storage cost of σ equals the storage costs of all the vertices in the replica set of σ. The *total cost* of σ is defined as

$$C(\sigma) = C_a(\sigma) + C_u(\sigma) + C_s(\sigma). \tag{4}$$

The total cost of σ equals the sum of its access, update, and storage cost. Observe, that if σ is a replica assignment for G, i.e. it is defined for all the vertices of G, then its total cost is equal to the cost of servicing all the requests issued by the vertices of G plus the cost of storing the object at the vertices in the replica set of σ.

A k–replica assignment for G of minimum cost, among all k–replica assignments for G, is called an *optimal k–replica assignment* and its replica set an *optimal k–replica set*. Similarly, a replica assignment for G of minimum cost, among all replica assignments for G, is called an *optimal replica assignment* for G, and its replica set an *optimal replica set* for G.

We are interested in finding optimal replica assignments for G, as well as in finding optimal k–replica assignments for G, for a given integer $1 \leq k \leq n$.

2.2 Treewidth and Tree–Decompositions

For completeness, we include here essential definitions and results, which we refer to later on, about the treewidth of graphs. For more details, the interested reader is referred to Bodlaender [4] and references therein.

A *tree–decomposition* of a (undirected) graph $G = (V, E)$ is a pair $(\mathcal{X}, T)$ with $T = (I, F)$ a tree, and $\mathcal{X} = \{X_i : i \in I\}$ a family of subsets of V, one for each node of T, such that

1. $\cup_{i \in I} X_i = V$,
2. for all the edges $(u, v) \in E$ there exists an $i \in I$ such that $u, v \in X_i$,
3. and for all $i, j, k \in I$, if j is on a path from i to k in T then $X_i \cap X_k \subseteq X_k$.

The *width* of a tree–decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* of G is the minimum width over all tree–decompositions of G. A *rooted tree–decomposition* of G is a tree–decomposition of G in which T is a rooted tree.

Given a graph G and a constant k, Bodlaender [3] provides a linear time algorithm that determines whether G has a tree–decomposition with width at most k, and if so, finds such a tree–decomposition.

A rooted tree–decomposition $(\{X_i : i \in I\}, T = (I, F))$ is called a *nice tree–decomposition*, if the following are satisfied:

1. every node of T has at most two children.
2. if a node i has two children j and k then $X_i = X_j = X_k$. Node i is called a *join* node.
3. if a node i has one child j then one the following is true

- $X_i = X_j \cup \{v\}$ for some vertex $v \in V$. Node i is called an *introduce* node, and vertex v is called an *introduced* vertex.
- $X_i = X_j - \{v\}$ for some vertex $v \in V$. Node i is called a *forget* node, and vertex v is called a *forgotten* vertex.

4. if a node i has no children, then $|X_i| = 1$. Node i is called a *leaf* node.

Bodlaender and Kloks [5] show that for every graph G with treewidth k, one can find a nice tree–decomposition of width k and $O(n)$ nodes in time $O(n)$. Hereafter, we assume only nice tree–decompositions, unless stated otherwise.

A *terminal graph* is a triple $H = (V, E, X)$ where (V, E) is a graph with vertex set V and edge set E, and X is an ordered subset of the vertices in V, called the terminals of H. A terminal graph with k terminals is called a k–terminal graph. The (composition) operation $\oplus$ is defined on pairs of k–terminal graphs H and H' as follows: $H \oplus H'$ is obtained by taking the disjoint union of H and H' and then identifying the ith terminal of H with the ith terminal of H' for all $i = 1, 2, \ldots, k$. A terminal graph H is a terminal subgraph of G iff there exists a terminal graph H' such that $G = H \oplus H'$.

We associate with every node i in a nice tree–decomposition $(\{X_i : i \in I\}, T = (I, F))$ the terminal graph $G_i = (V_i, E_i, X_i)$, where

$$V_i = \{v : v \in X_j \text{ and } j = i \text{ or } j \text{ is a descendant of } i \text{ in } T\},$$

$$E_i = \{(u, v) \in E : u, v \in V_i\},$$

i.e. G_i is the subgraph induced by vertices in the sets of the node i and all the nodes below i in T, with X_i as the set of terminals. The ordering of X_i is not important. We call the non-terminal vertices of V_i an *internal* vertices and the vertices in $V - V_i$ *external* vertices.

Some additional observations are in order (see Bodlaender and Kloks [5], and Bodlaender [3]).

Proposition 1 ([3, 5]) *Consider a nice tree–decomposition $(\{X_i : i \in I\}, T = (I, F))$ of G. For every node $i \in I$, the following hold:*

- *$G = G_i \oplus H$ for some terminal graph H.*
- *every path in G from a vertex $v \in V_i - X_i$ to a vertex $u \in V - V_i$ goes through a vertex in X_i.*
- *the subtree of T rooted at i is a nice tree–decomposition for its associated graph G_i (ignoring its terminals).*

- *if i is a join node with children j and k then $V_j \cap V_k = X_i$ and $G_i = G_j \oplus G_k$.*
- *if i is a forget node with child node j then $G_i = G_j$.*
- *if i is an introduce node with child j and introduced vertex v, then $V_i = V_j \cup \{v\}$, $v \notin V_j$, and all the neighbors of v in G_i belong to X_j.*

3 Canonical Replica Assignments

Consider a graph G with constant treewidth t. Fix a nice tree–decomposition $\mathcal{T} = (\{X_p : p \in I\}, T_G)$ of G of width t. We refer to the nodes of T_G as nodes of the tree–decomposition $\mathcal{T}$. [3] Let p be a node of $\mathcal{T}$, and $G_p = (V_p, E_p, X_p)$ be the terminal graph associated with p. Let σ be a replica assignment for G.

We call a replica assignment σ for G *canonical* if for every node p of the tree–decomposition $\mathcal{T}$ of G, $\forall v \in V_p$, it is true that $\sigma(v) \in \sigma(X_p) \cup \sigma(V_p) \cap V_p$. Intuitively, in a canonical replica assignment, for every p, every vertex in the terminal graph G_p is assigned either a vertex also assigned to a terminal vertex of G_p or an internal vertex of G_p.

Lemma 1 *Let σ be a replica assignment for G. There exists a canonical replica assignment σ' for G such that $\delta(v, \sigma(v)) = \delta(v, \sigma'(v))$ for every vertex $v \in V$, $\sigma(V) = \sigma'(V)$, and $C(\sigma') = C(\sigma)$.*

Proof.
Assume that σ is not canonical. Let p be a node in the tree–decomposition for G and v a vertex in V_p such that $\sigma(v) \notin \sigma(X_p) \cup \sigma(V_p) \cap V_p$. Let $A = \sigma(X_p) \cup \sigma(V_p) \cap V_p$. Such v exists, since σ is not canonical. Vertex $v \notin X_p$, since $\sigma(X_p) \subseteq A$. Hence, v is an internal vertex of G_p. Since $\sigma(v) \notin A$, $\sigma(v)$ is an external vertex of G_p. From Proposition 1, there exists a vertex $z \in X_p$ that is on a shortest path from v to $\sigma(v)$ in G. Further, by definition of σ, $\delta(z, \sigma(z)) \leq \delta(z, \sigma(v))$. Then, $\delta(v, \sigma(v)) = \delta(v, z) + \delta(z, \sigma(v)) \geq \delta(v, z) + \delta(z, \sigma(z)) \geq \delta(v, \sigma(z))$. Since $\delta(v, \sigma(v)) = \delta(v, \sigma(V))$, it follows that $\delta(v, \sigma(z)) = \delta(v, \sigma(v))$. By assigning vertex $\sigma(z)$ to v instead of $\sigma(v)$, we obtain a replica assignment with one less vertex for which the canonical property of the assignment may be violated. By repeating this process top–down using the tree–decomposition of G, we obtain a canonical replica assignment σ' for V. Clearly, $\delta(v, \sigma'(v)) = \delta(v, \sigma(v))$ for all $v \in V$. Since $\sigma'(v) = v$ for all $v \in \sigma(V)$, we also have that $\sigma'(V) = \sigma(V)$. By definition of the total cost of a replica assignment follows that the total cost of σ' equals that of σ. ■

Hereafter, we assume only canonical replica assignments for G.

[3] We use "vertex" for the vertices of G and "node" for the vertices of T_G.

Let σ_p denote the restriction of a (canonical) replica assignment σ to G_p. The replica assignment $\sigma_p : V_p \rightarrow \sigma(V_p)$ is such that $\sigma_p(v) = \sigma(v)$ for all $v \in V_p$, i.e. σ_p is a replica assignment for G_p that agrees with σ on all vertices in G_p. Moreover, σ_p is a canonical replica assignment for G_p, since σ is a canonical replica assignment for G.

Lemma 2 *Let σ be a canonical replica assignment for G. Let p, q be two nodes in the tree–decomposition $\mathcal{T}$ of G, such that q is a descendant of p. Let σ_p (σ_q) be the restrictions of σ to G_p (G_q). Then, σ_p (σ_q) is a canonical replica assignment for G_p (G_q), and σ_q is the restriction of σ_p to G_q.*

Proof.
Since σ is a canonical replica assignment for G, and σ_p agrees with σ on all vertices in V_p, σ_p is a canonical replica assignment for G_p. Similarly, σ_q is a canonical replica assignment for G_q.

Observe that by definition of the terminal graph associated with a node of the tree–decomposition of G, and since q is a descendant of p, $V_q \subseteq V_p$. Since σ_p and σ_q agree with σ on all the vertices in V_p and V_q, respectively, it follows that σ_q agrees with σ_p on all vertices in V_q. Hence, σ_q is the restriction of σ_p to G_q. ■

4 Canonical Minimum Spanning Trees

Consider a (canonical) replica assignment σ for G, and let $S = \sigma(V)$ be its replica set. We call a minimum spanning tree T of $\Delta[S]$ a *canonical minimum spanning tree* of $\Delta[S]$ if: for every node p in the tree decomposition $\mathcal{T}$ of G, T does not have an edge between a vertex in $V_p - \sigma(X_p)$ and a vertex in $V - V_p - \sigma(X_p)$. Intuitively, a canonical MST of $\Delta[S]$ does not have edges between replica vertices, which are internal for G_p, and replica vertices, which are external for G_p and are not assigned to any terminal vertex of G_p. A canonical MST for $\Delta[S]$ only has edges between replica vertices that are either internal vertices of G_p or are assigned to terminals of G_p, for every node p of the tree–decomposition $\mathcal{T}$ of G.

Lemma 3 *Let σ be a (canonical) replica assignment for G, and let S be its replica set $\sigma(V)$. Then, there exists a minimum spanning tree of $\Delta[S]$ which is canonical.*

Proof.
Let T be an MST of the distance graph $\Delta[S]$ for S. Suppose that T is not a canonical MST of $\Delta[S]$. Traverse $\mathcal{T}$ in preorder, and examine each node p to see whether T contains an edge (u, v) with $u \in V_p - \sigma(X_p)$ and $v \in V - V_p - \sigma(X_p)$.

Let p be the first node of $\mathcal{T}$ for which $(u, v) \in T$, where $u \in V_p - \sigma(X_p)$ and $v \in V - V_p - \sigma(X_p)$. From Proposition 1 we know that $G = G_p \oplus H$, where H is some terminal

subgraph of G with terminals X_p. Note that vertex u is an internal vertex of G_p and vertex v is an internal vertex of H.

Consider a shortest path from u to v in G. From Proposition 1, there exists a vertex $z \in X_p$ on that shortest path from u to v.

Since $\delta(z, \sigma(z)) \leq \delta(z, v)$ and $\delta(u, v) = \delta(u, z) + \delta(z, v)$, it follows that $\delta(u, \sigma(z)) \leq \delta(u, v)$.

Since $\delta(z, \sigma(z)) \leq \delta(z, u)$ and $\delta(v, u) = \delta(v, z) + \delta(z, u)$, it follows that $\delta(v, \sigma(z)) \leq \delta(v, u) = \delta(u, v)$.

By removing the edge (u, v) from T we obtain a forest F with two trees T_u and T_v, such that u is contained in T_u and v in T_v. There are two cases to consider. If $\sigma(z)$ is in T_u, then by adding the edge $(v, \sigma(z))$ to F, we get a spanning tree T_1 of $\Delta[S]$. Since the cost of T_1 is no more than that of T, T_1 is also a minimum spanning tree of $\Delta[S]$. If $\sigma(z)$ is in T_v, then by adding the edge $(u, \sigma(z))$ to F, we get a spanning tree T_2 of $\Delta[S]$. Since the cost of T_2 is no more than that of T, T_2 is also a minimum spanning tree of $\Delta[S]$. In both cases, the edge added to F has one end–vertex that is in $\sigma(X_p)$. In either case we obtain a minimum spanning tree of $\Delta[S]$ with one fewer undesired edge. Clearly, by repeating this procedure for each undesired edge, we can obtain a canonical minimum spanning tree of $\Delta[S]$. ■

5 Terminal–Restrictions of Canonical MSTs: Canonical Π–MSFs

Consider a (canonical) replica assignment σ for G, and let $S = \sigma(V)$ be its replica set. Let p be a node of the tree–decomposition $\mathcal{T}$ of G.

We define the *terminal–restriction* of a subgraph G' of $\Delta[V]$ to G_p to be the graph that results by removing from G' all vertices in $\sigma(V) - \sigma(V_p)$, as well as their incident edges, and all edges between vertices in $\sigma(X_p)$.

Lemma 4 *Let σ be a canonical replica assignment for G. Let p, q be two nodes in the tree–decomposition of G, such that q is a descendant of p. Let T be a canonical MST for $\Delta[\sigma(V)]$. Let F_p (F_q) be the terminal–restriction of T to G_p (G_q). Then, F_q is a subgraph of F_p.*

Proof.
Consider the process with which terminal–restrictions are computed.

F_p is obtained from T by removing from it all vertices (and their incident edges) in $\sigma(V) - \sigma(V_p)$, as well as all edges between vertices in $\sigma(X_p)$. Let Q_p be the set of vertices removed from T, and let Z_p be the set of edges removed from T.

Similarly, for F_q; let Q_q be the set of vertices removed from T, and let Z_q be the set of edges removed from T.

Since T is a canonical MST of $\Delta[\sigma(V)]$, it has no edges between a vertex internal for G_p and a vertex external for G_p which is not in $\sigma(X_p)$. Similarly, T has no edges between a vertex internal for G_q and a vertex external for G_q which is not in $\sigma(X_q)$.

Since q is a descendant of p, from Proposition 1, G_q is a subgraph of G_p, and $V_q \subseteq V_p$. From Lemma 2, we have that σ_q is the restriction of σ_p to G_q, which implies that $\sigma(V_q) \subseteq \sigma(V_p)$. Therefore, the set of vertices removed from T to obtain F_p is a subset of the set of vertices removed from T in order to obtain F_q. Similarly, the set of edges removed from T to obtain F_p is a subset of the set of edges removed from T in order to obtain F_q. Thus, F_q is a subgraph of F_p. ■

Consider the restriction σ_p of σ to G_p. Let $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$ be a partition of $\sigma_p(X_p)$. A spanning forest F of $\Delta[S_p]$ *respects* the partition Π if F induces Π and does not contain edges between vertices in $\sigma_p(X_p)$. A spanning forest of $\Delta[S_p]$ that respects Π and has minimum cost among all spanning forests of $\Delta[S_p]$ that respect Π is called a Π–*minimum spanning forest* (Π–MSF) of $\Delta[S_p]$. Let $\mathrm{MSF}(\Pi, p, \sigma_p)$ denote the cost of a Π–minimum spanning forest of $\Delta[S_p]$. A Π–MSF of $\Delta[S_p]$ that does not have an edge between a vertex in $V_q - \sigma_p(X_q)$ and a vertex in $V - V_q - \sigma_p(X_q)$, for any descendant q of p in the tree–decomposition $\mathcal{T}$ of G, is called a *canonical* Π–*MSF* of $\Delta[S_p]$.

Observe that the distance graph $\mathrm{G}[\Pi]$ associated with a partition Π of $\sigma_p(X_p)$ has at most $t+1$ vertices, since $|X_p| \leq t+1$.

Consider a canonical minimum spanning tree T of $\Delta[S]$. The terminal–restriction of T to G_p is a spanning forest F of $\Delta[S_p]$. Let Π be the partition of $\sigma_p(X_p)$ induced by F. Since T is an MST of $\Delta[S]$, F has minimum cost among all spanning forests of S_p that respect Π. Moreover, since T is a canonical MST of $\Delta[S]$, and σ_p agrees with σ on all vertices in V_p, F is a canonical Π–minimum spanning forest of $\Delta[S_p]$.

Lemma 5 *Let σ be a canonical replica assignment for G, and let σ_p its restriction to G_p. Let $S = \sigma(V)$ and $S_p = \sigma_p(V_p)$. Every canonical MST T of $\Delta[S_p]$ can be obtained, in $O(t^2)$ time, from a canonical Π–MSF F of $\Delta[S_p]$, for some partition Π of $\sigma_p(X_p)$. Further, F is a subgraph of T, all the edges in $T - F$ are in $\sigma_p(X_p) \times \sigma_p(X_p)$, and*

$$\mathrm{MST}(S_p) = \mathrm{MSF}(\Pi, p, \sigma_p) + \mathrm{MST}(\Pi).$$

Moreover, every canonical MST of $\Delta[S]$ has a Π'–MSF of $\Delta[S_p]$ as subgraph, for some partition Π' of $\sigma(X_p)$.

Proof.
Let T be an canonical MST of $\Delta[S_p]$. Let F be the terminal–restriction of T to G_p. Since T

is a tree, F is a forest. Since T only spans vertices in $\sigma_p(V_p) = \sigma(V_p)$, T and F contain the same set of vertices. Let $A \subseteq \sigma(X_p) \times \sigma(X_p)$ be the set of edges removed from T in order to get F. Forest F induces a partition Π of $\sigma_p(X_p)$. Since T is an MST of $\Delta[S_p]$, it follows that F is a Π–MSF of $\Delta[S_p]$.

Since we only removed the edges in A from T to obtain F, in order to obtain T given F, we only need to consider edges between vertices in $\sigma_p(X_p)$ in $\Delta[S_p]$. Since $|\sigma_p(X_p)| \leq t+1$, there are $\leq (t+1)^2$ such edges. Forest F can be easily extended to an MST of $\Delta[S_p]$ by using an MST T_0 for G[Π] as follows. For each edge (π_i, π_j) in T_0 connect the corresponding trees in F with an edge of minimum cost among those between vertices in π_i and in π_j. Clearly, this can be done in time $O(t^2)$, and $\mathrm{MST}(S_p) = \mathrm{MSF}(\Pi, p, \sigma_p) + \mathrm{MST}(\Pi)$.

By considering a canonical MST T_0 of $\Delta[S]$ instead of a canonical MST of $\Delta[S_p]$, and using similar arguments as above, we can show that there exists a partition Π' of $\sigma(X_p)$ and a subgraph of T_0 that is a canonical Π'–MSF of $\Delta[S_p]$. ■

6 Structure of Canonical Π–Minimum Spanning Forests

Consider a (canonical) replica assignment σ for G, and a canonical minimum spanning tree T of $\Delta[\sigma(V)]$. Let p be a node of the tree–decomposition $\mathcal{T}$ of G.

Lemma 6 *Let σ be a canonical replica assignment for G. Let p be an introduce node of the tree–decomposition $\mathcal{T}$ of G, x be the introduced vertex for p, and q be the child of p. Let $\Pi_p = (\pi_1, \pi_2, \ldots, \pi_m)$ and Π_q be the partitions of $\sigma(X_p)$ and $\sigma(X_q)$ induced by the terminal–restrictions of a canonical MST for $\Delta[\sigma(V)]$ to G_p and G_q, respectively. Let π_1 be the block of Π_p that contains $\sigma(x)$. Then,* $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$. *Moreover,*

- *if $\sigma(x) \in \sigma(X_q)$, then $\sigma(X_p) = \sigma(X_q)$, $\sigma(V_p) = \sigma(V_q)$, and $\Pi_p = \Pi_q$.*
- *otherwise, $\sigma(X_q) = \sigma(X_p) - \sigma(x)$, $\sigma(V_q) = \sigma(V_p) - \sigma(x)$, $\sigma(x) \notin \sigma(V_q)$, $\pi_1 = \{\sigma(x)\}$, and $\Pi_q = (\pi_2, \ldots, \pi_m)$.*

Proof.
Since p is an introduce node with child q, from Proposition 1, we have that $X_q = X_p - x$ and that the introduced vertex x is not in V_q. Further, $V_q = V_p - x$. Then, $\sigma(V_p) = \sigma(V_q) \cup \{\sigma(x)\}$ and $\sigma(X_p) = \sigma(X_q) \cup \{\sigma(x)\}$.

Let T be a canonical MST for $\Delta[\sigma(V)]$. Let F_p (F_q) be the terminal–restriction of T to G_p (G_q). There are two cases to consider.

Case 1: $\sigma(x) \in \sigma(X_q)$.
Then, $\sigma(V_p) = \sigma(V_q)$ and $\sigma(X_p) = \sigma(X_q)$. Moreover, $F_p = F_q$, since, by the definition

of terminal–restriction, the exact same vertices and edges are removed from T in order to obtain either F_p or F_q. Since $F_p = F_q$, we have that $\Pi_p = \Pi_q$ and that $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$.

Case 2: $\sigma(x) \notin \sigma(X_q)$.
Since x is an external vertex for G_q, and since σ is a canonical replica assignment for G, it follows that $\sigma(x) \notin \sigma(V_q)$. Then, since $\sigma(V_q) \subseteq \sigma(V_p)$, it follows that $\sigma(V_p) \cap \sigma(V_q) = \sigma(V_q)$. Furthermore, $\sigma(V_p) = \sigma(V_q) \cup \{\sigma(x)\}$.

Since T is a canonical MST for $\Delta[\sigma(V)]$, and $\sigma(x)$ is an external vertex for G_q, there are no edges between $\sigma(x)$ and any internal vertex of G_q in T. By definition of terminal–restriction then follows that the tree of F_p that contains $\sigma(x)$ contains no other vertices of $\sigma(V_p)$. Therefore, $\pi_1 = (\sigma(x))$. Moreover, again by definition of terminal–restriction, it follows that F_q is equal to F_p with the singleton tree containing $\sigma(x)$ removed. Thus, $\Pi_q = (\pi_2, \ldots, \pi_m)$. In addition, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_p, p, \sigma_q)$. ■

Lemma 7 *Let σ be a canonical replica assignment for G. Let p be a forget node of the tree–decomposition $\mathcal{T}$ of G, x be the forgotten vertex for p, and q be the child of p. Let Π_p and Π_q be the partitions of $\sigma(X_p)$ and $\sigma(X_q)$ induced by a canonical MST T of $\Delta[\sigma(V)]$, respectively. Then the following hold:*

- *if every neighbor of $\sigma(x)$ in T is an internal vertex of G_p, then Π_p agrees with Π_q on all blocks except one block, which in Π_q also contains $\sigma(x)$. Further, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$.*
- *otherwise, every block of Π_p, except one block π, is also a block of Π_q. Let Π be the set of blocks of Π_q that are not blocks of Π_p. Then, π contains all the vertices that are in blocks in Π, except vertex $\sigma(x)$; Π is a partition of $\pi \cup \{\sigma(x)\}$; and $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi)$.*

Proof.
From Proposition 1, we have that $X_p = X_q - x$, $V_p = V_q$, and $G_p = G_q$. Since $X_p \subset X_q$, we have that $\sigma(X_p) \subseteq \sigma(X_q)$. Further, $\sigma(V_p) = \sigma(V_q)$. Let F_p (F_q) be the terminal–restriction of T to G_p (G_q). Note that $\sigma(x) \in \sigma(V_q) = \sigma(V_p)$.

Since T is a canonical MST of $\Delta[\sigma(V)]$, and x is an internal vertex of G_p, every neighbor of $\sigma(x)$ in T is either an internal vertex of G_p or a vertex in $\sigma(X_p)$. There are two cases to consider:

Case 1: every neighbor of $\sigma(x)$ in T is an internal vertex of G_p.

Since every internal vertex of G_p different than x is also an internal vertex of G_q, it follows that $F_p = F_q$, since the exact same set of vertices and edges are removed from T in

order to obtain F_p and F_q. Therefore, $\Pi_p = (\pi_1 - \sigma(x), \pi_2, \ldots, \pi_m)$ and $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$. In other words, Π_p agrees with Π_q on all blocks except one block, which in Π_q also contains $\sigma(x)$.

Case 2: $\sigma(x)$ has neighbors in T that are in $\sigma(X_p)$.

Let T' be the tree in F_p that contains $\sigma(x)$, and let π_1 be the block in Π_p that contains all the vertices of T' that are in $\sigma(X_p)$. Vertex $\sigma(x)$ has at least one neighbor in F_p that is in $\sigma(X_p)$. By removing from T' all the edges between $\sigma(x)$ and vertices in $\sigma(X_p)$ we obtain a forest F'. Let A be the set of edges removed from T' in order to get F'. Let V' be the set of vertices of T'. Since T' belongs to F_p, and F_p is a Π_p–MSF of $\Delta[\sigma(V_p)]$, it follows that T' is an MST of $\Delta[V']$. Forest F' induces a partition Π' of the set of vertices $\pi_1 \cup \{\sigma(x)\}$. Since T' is an MST of $\Delta[V']$, F' is a Π'–MSF of $\Delta[V']$. Furthermore, the cost of all the edges in A is equal to $\mathrm{MST}(\Pi')$. Hence, the cost of T' is equal to the cost of F' plus $\mathrm{MST}(\Pi')$.

Observe that F_q can be obtained from F_p by simply removing all edges in A from F_p. In addition, F_q can also be obtained from F_p by replacing T' with F' in F_p. Thus, the partition of $\sigma(X_q)$ induced by F_q consists of all the blocks in Π_p except the block π_1, and all the blocks in Π'. Moreover, since the costs of all the edges in A is $\mathrm{MST}(\Pi')$, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi')$. Note that Π' is a partition of the only block of Π_p that is not a block of Π_q, together with the vertex $\sigma(x)$, and that Π' consists of all the blocks of Π_q that are not in Π_p. ■

Lemma 8 *Let σ be a canonical replica assignment for G. Let p be a join node of the tree–decomposition $\mathcal{T}$ of G, and q and r be the two children of p. Let Π_p, Π_q, Π_r be the partitions of $\sigma(X_p)$, $\sigma(X_q)$, $\sigma(X_r)$ induced by a canonical MST T of $\Delta[\sigma(V)]$, respectively. Then, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MSF}(\Pi_r, r, \sigma_r)$, Π_q and Π_r are refinements of Π_p, and $\Pi_p = \Pi_q \otimes \Pi_r$.*

Proof.
Recall that, from Proposition 1, $X_p = X_q = X_r$, $V_q \cap V_r = X_p$, and $G_p = G_q \oplus G_r$.

Let F_i be the terminal–restriction of T to G_i, for $i = p, q, r$. Since T is a canonical MST of $\Delta[\sigma(V)]$, T does not have any edges (u, v), such that u is an internal vertex of G_q and v is an internal vertex of G_r. Since F_p is a subgraph of T, F_p does not have any such edges either. Then, F_p is the union of F_q and F_r. Moreover, since none of the F_p, F_q, F_r has any edges between vertices in $\sigma(X_p)$, and since F_q and F_r have no common edges and have only the vertices in $\sigma(X_p)$ in common (because σ is canonical), it follows that

$$\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MSF}(\Pi_r, r, \sigma_r). \tag{5}$$

Moreover, since both F_q and F_r can be thought of as terminal graphs with $\sigma(X_q) = \sigma(X_r) = \sigma(X_p)$ as their set of terminals, we also have that $F_p = F_q \oplus F_r$.

Observe that whenever any two trees $T_1 \in F_q$ and $T_2 \in F_r$, have any common vertex, which must be a vertex in $\sigma(X_p)$, they both become subtrees of a tree T_3 in F_p. Moreover, if π_j is the set of vertices of T_j that are in $\sigma(X_p)$, for $j = 1, 2, 3$, then π_3 contains the all the vertices in $\pi_1 \cup \pi_2$. Therefore, by the definition of the $\otimes$ operator for partitions, we have that $\Pi_p = \Pi_q \otimes \Pi_r$. ■

Lemma 9 *Let σ be a canonical replica assignment for G. Let p be a leaf node of the tree–decomposition $\mathcal{T}$ of G, and let x be the single vertex in G_p. Let Π_p be the partition of $\sigma(V_p)$ induced by a canonical MST of $\Delta[\sigma(V)]$. Then, $\sigma(V_p) = \sigma(X_p) = \{\sigma(x)\}$, $\Pi_p = (\{\sigma(x)\})$, and* $\mathrm{MSF}(\Pi_p, p, \sigma_p) = 0$.

Proof.
Follows directly from the definitions and Proposition 1. ■

7 Characteristics of Replica Assignments

Let σ be a replica assignment for G, and let T be a canonical MST of $\Delta[\sigma(V)]$. Let p be a node of the tree–decomposition $\mathcal{T}$ of G, and Π_p be the partition of $\sigma(X_p)$ induced by the terminal–restriction of T to G_p.

Consider the restriction σ_p of σ to G_p. Let $\hat{\sigma}_p : X_p \to \sigma_p(X_p)$ be the restriction of σ_p to X_p. We define the *characteristic* of σ_p for G_p to be the tuple

$$\mathrm{char}(\sigma_p) = (\hat{\sigma}_p, \Pi_p, |\sigma_p(V_p)|, C(\sigma_p)). \tag{6}$$

The characteristic of σ_p provides us with all the information we need to know about σ_p in order to see whether it can be extended to a replica assignment for G of minimum cost.

We associate with p, a set $\mathrm{FS}(p)$ of characteristics as follows. For each canonical replica assignment σ, add the characteristic $\mathrm{char}(\sigma_p)$ to $\mathrm{FS}(p)$, unless $\mathrm{FS}(p)$ already contains a tuple $(\hat{\sigma}_p, \Pi_p, |\sigma_p(V_p)|, \chi)$ with $\chi \leq C(\sigma_p)$. We call the set $\mathrm{FS}(p)$ the *full set of characteristics* of p.

Lemma 10 *Let p be an introduce node in the tree–decomposition $\mathcal{T}$ of G and let q be the child of p. Let x be the introduced vertex for p. Let σ_p (σ_q) be a canonical replica assignment for G_p (G_q). Then,* $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ *if and only if* $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$, *where σ_q is the restriction of σ_p to G_q. Moreover, $\hat{\sigma}_q$ is the restriction of $\hat{\sigma}_p$ to X_q, and*

- *if $\hat{\sigma}_p(x) \in \hat{\sigma}_p(X_q)$, then $\Pi_p = \Pi_q$, $|\sigma_p(V_p)| = |\sigma_q(V_q)|$, and*

$$C(\sigma_p) = C(\sigma_q) + \mathrm{r}(x) \cdot \delta(x, \hat{\sigma}_p(x)) + W_{\mathrm{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q)), \tag{7}$$

- *otherwise,* Π_p *consists of all the blocks of* Π_q *plus the block* $\{\hat{\sigma}_p(x)\}$, $|\sigma_p(V_p)| = |\sigma_q(V_q)| + 1$, *and*

$$C(\sigma_p) = C(\sigma_q) + \mathrm{r}(x) \cdot \delta(x, \hat{\sigma}_p(x)) + W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q)) + \mathrm{s}(\hat{\sigma}_p(x)). \quad (8)$$

Proof.
From Lemma 2 we know that σ_p can be extended to a canonical replica assignment σ of G. Then, the restriction of σ to G_q is equal to the restriction σ_q of σ_p to G_q. Let T be a canonical MST for $\Delta[\sigma(V)]$.

Let $\mathrm{char}(\sigma_p) = (\sigma_p(X_p), \Pi_p, |\sigma_p(V_p)|, C(\sigma_p))$, and $\mathrm{char}(\sigma_q) = (\sigma_q(X_q), \Pi_q, |\sigma_q(V_q)|, C(\sigma_q))$.

Observe that

$$C_a(\sigma_p) - C_a(\sigma_q) = \mathrm{r}(x) \cdot \delta(x, \sigma_p(x)), \quad (9)$$

since $V_q = V_p - x$.

From Lemma 5, we have that $\mathrm{MST}(\sigma_p(V_p)) = \mathrm{MSF}(\Pi_p, p, \sigma_p) + \mathrm{MST}(\Pi_p)$ and that $\mathrm{MST}(\sigma_q(V_q)) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi_q)$. From Lemma 6, we have that $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$. Then,

$$C_u(\sigma_p) - C_u(\sigma_q) = W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q)). \quad (10)$$

From Lemma 6, we have that if $\sigma_p(x) \in \sigma_p(X_q)$ then $\sigma_p(V_p) = \sigma_q(V_q)$, which implies that $C_s(\sigma_p) - C_s(\sigma_q) = 0$. Further, if $\sigma_p(x) \notin \sigma_p(X_q)$ then $\sigma_p(V_p) - \sigma_q(V_q) = \{\sigma_p(x)\}$, which implies that $C_s(\sigma_p) - C_s(\sigma_q) = \mathrm{s}(\sigma_p(x))$.

Therefore,

$$C(\sigma_p) - C(\sigma_q) = \mathrm{r}(x) \cdot \delta(x, \sigma_p(x)) + W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q)) \quad (11)$$

if $\sigma_p(x) \in \sigma_p(X_q)$, and

$$C(\sigma_p) - C(\sigma_q) = \mathrm{r}(x) \cdot \delta(x, \sigma_p(x)) + W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q)) + \mathrm{s}(\sigma_p(x)) \quad (12)$$

otherwise. Recall that $\mathrm{char}(\sigma_p)$ is in $\mathrm{FS}(p)$ iff $C(\sigma_p)$ has minimum among all characteristics that that agree with $\mathrm{char}(\sigma_p)$ on the first three components. Similarly, for σ_q. Note that $\mathrm{FS}(p)$ has a tuple $(\hat{\sigma}_p, \Pi_p, |\sigma_p(V_p)|, \chi)$ iff $\mathrm{FS}(q)$ has a tuple $(\hat{\sigma}_q, \Pi_q, |\sigma_q(V_q)|, \chi')$. The difference $C(\sigma_p) - C(\sigma_q)$ depends only on x, $\sigma_p(X_p)$, and Π_p. Therefore, $C(\sigma_p)$ is minimum if and only if $C(\sigma_q)$ is minimum. Consequently, $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ if and only if $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$. ■

Lemma 11 *Let p be a forget node in the tree–decomposition $\mathcal{T}$ of G and let q be the child of p. Let x be the forgotten vertex for p. Let σ_p (σ_q) be a canonical replica assignment for G_p (G_q). Then,* $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ *if and only if* $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$*, where σ_q is the restriction of σ_p to G_q. Moreover, every block of Π_p, except one block π, is also a block of Π_q, and the set Π, consisting of blocks of Π_q that are not blocks of Π_p, forms a partition of $\pi \cup \{\hat{\sigma}_q(x)\}$. Further, $\hat{\sigma}_p$ is the restriction of $\hat{\sigma}_q$ to X_p, $|\sigma_p(V_p)| = |\sigma_q(V_q)|$, and*

$$C(\sigma_p) = C(\sigma_q) + W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) + \mathrm{MST}(\Pi)). \tag{13}$$

Proof.
From Lemma 2 we know that σ_p can be extended to a canonical replica assignment σ of G. Then, the restriction of σ to G_q is equal to the restriction σ_q of σ_p to G_q. Let T be a canonical MST for $\Delta[\sigma(V)]$.

Let $\mathrm{char}(\sigma_p) = (\sigma_p(X_p), \Pi_p, |\sigma_p(V_p)|, C(\sigma_p))$, and $\mathrm{char}(\sigma_q) = (\sigma_q(X_q), \Pi_q, |\sigma_q(V_q)|, C(\sigma_q))$.

Observe that $C_a(\sigma_p) = C_a(\sigma_q)$, since $V_p = V_q$.

From Lemma 5, we have that $\mathrm{MST}(\sigma_p(V_p)) = \mathrm{MSF}(\Pi_p, p, \sigma_p) + \mathrm{MST}(\Pi_p)$ and that $\mathrm{MST}(\sigma_q(V_q)) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi_q)$.

From Lemma 7 we know that all blocks of Π_p except one are also blocks of Π_q. Let Π be the set of blocks of Π_q that are not blocks of Π_p. If Π consists of a single block then, again from Lemma 7, we know that Π_p agrees with Π_q on all blocks except one block, which in Π_q also contains $\hat{\sigma}_q(x)$. Moreover, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q)$. If Π consists of more than one blocks, then, from Lemma 7, we know that: Π is a partition of $\pi \cup \{\hat{\sigma}_q(x)\}$, where π is the single block of Π_p that is not a block of Π_q. Moreover, $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi)$. Since when Π consists of one block only, we have that $\mathrm{MST}(\Pi) = 0$, we conclude that $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi)$, which implies that

$$C_u(\sigma_p) - C_u(\sigma_q) = W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) + \mathrm{MST}(\Pi)). \tag{14}$$

Since $V_p = V_q$ and σ_q is the restriction of σ_p to G_q, we have that $\sigma_p(V_p) = \sigma_q(V_q)$, which implies that $C_s(\sigma_p) - C_s(\sigma_q) = 0$.

Therefore,

$$C(\sigma_p) - C(\sigma_q) = W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) + \mathrm{MST}(\Pi)) \tag{15}$$

where Π consists of all the blocks of Π_q that are not blocks of Π_p.

Recall that $\mathrm{char}(\sigma_p)$ is in $\mathrm{FS}(p)$ iff $C(\sigma_p)$ has minimum among all characteristics that that agree with $\mathrm{char}(\sigma_p)$ on the first three components. Similarly, for σ_q. Note that $\mathrm{FS}(p)$ has a tuple $(\hat{\sigma}_p, \Pi_p, |\sigma_p(V_p)|, \chi)$ iff $\mathrm{FS}(q)$ has a tuple $(\hat{\sigma}_q, \Pi_q, |\sigma_q(V_q)|, \chi')$. The difference

$C(\sigma_p) - C(\sigma_q)$ depends only on x, $\sigma_p(X_p)$, and Π_p. Therefore, $C(\sigma_p)$ is minimum if and only if $C(\sigma_q)$ is minimum. Consequently, $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ if and only if $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$. ■

Lemma 12 *Let p be a join node in the tree–decomposition $\mathcal{T}$ of G and let q and r be the two children of p. Let σ_p, σ_q, and σ_r be canonical replica assignments for G_p, G_q, and G_r respectively. Then,* $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ *if and only if* $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$ *and* $\mathrm{char}(\sigma_r) \in \mathrm{FS}(r)$*, where σ_q and σ_r are the restrictions of σ_p to G_q and G_r respectively. Moreover, $\Pi_p = \Pi_q \otimes \Pi_r$, $\hat{\sigma}_p = \hat{\sigma}_q = \hat{\sigma}_r$, $|\sigma_p(V_p)| = |\sigma_q(V_q)| + |\sigma_r(V_r)| - |\hat{\sigma}_p(X_p)|$, and*

$$\begin{aligned} C(\sigma_p) &= C(\sigma_q) + C(\sigma_r) + W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) + \mathrm{MST}(\Pi)) - \\ &\quad - \sum_{v \in X_p} \mathrm{r}(v) \cdot \delta(v, \hat{\sigma}_p(v)) - \sum_{v \in \hat{\sigma}_p(X_p)} \mathrm{s}(v). \end{aligned} \tag{16}$$

Proof.
From Lemma 2 we know that σ_p can be extended to a canonical replica assignment σ of G. Then, the restriction of σ to G_q (G_r) is equal to the restriction σ_q (σ_r) of σ_p to G_q (G_r). Let T be a canonical MST for $\Delta[\sigma(V)]$.

Let $\mathrm{char}(\sigma_p) = (\sigma_p(X_p), \Pi_p, |\sigma_p(V_p)|, C(\sigma_p))$, $\mathrm{char}(\sigma_q) = (\sigma_q(X_q), \Pi_q, |\sigma_q(V_q)|, C(\sigma_q))$, and $\mathrm{char}(\sigma_r) = (\sigma_r(X_r), \Pi_r, |\sigma_r(V_r)|, C(\sigma_r))$.

From Proposition 1, we know that $V_q \cap V_r = X_p = X_q = X_r$. Thus, $\hat{\sigma}_p = \hat{\sigma}_q = \hat{\sigma}_r$, and $|\sigma_p(V_p)| = |\sigma_q(V_q)| + |\sigma_r(V_r)| - |\hat{\sigma}_p(X_p)|$.

Observe that

$$C_a(\sigma_p) - C_a(\sigma_q) - C_a(\sigma_r) = - \sum_{v \in X_p} \mathrm{r}(v) \cdot \delta(v, \hat{\sigma}_p(v)) \tag{17}$$

since the access cost for the vertices in X_p is counted twice in $C_a(\sigma_q) + C_a(\sigma_r)$.

From Lemma 5, we have that $\mathrm{MST}(\sigma_p(V_p)) = \mathrm{MSF}(\Pi_p, p, \sigma_p) + \mathrm{MST}(\Pi_p)$, $\mathrm{MST}(\sigma_q(V_q)) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MST}(\Pi_q)$, and $\mathrm{MST}(\sigma_r(V_r)) = \mathrm{MSF}(\Pi_r, r, \sigma_r) + \mathrm{MST}(\Pi_r)$.

From Lemma 8 we have that $\mathrm{MSF}(\Pi_p, p, \sigma_p) = \mathrm{MSF}(\Pi_q, q, \sigma_q) + \mathrm{MSF}(\Pi_r, r, \sigma_r)$, and that $\Pi_p = \Pi_q \otimes \Pi_r$. Thus,

$$\mathrm{MST}(\sigma_p(V_p)) - \mathrm{MST}(\sigma_q(V_q)) - \mathrm{MST}(\sigma_r(V_r)) = \mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) - \mathrm{MST}(\Pi_r), \tag{18}$$

Therefore,

$$C_u(\sigma_p) - C_u(\sigma_q) - C_u(\sigma_r) = W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) - \mathrm{MST}(\Pi_r)). \tag{19}$$

Since both σ_q, σ_r are restrictions of σ_p, it follows that $\sigma_q(V_q) \cap \sigma_r(V_r) = \sigma_p(X_p) = \hat{\sigma}_p(X_p)$. Consequently,

$$C_s(\sigma_p) - C_s(\sigma_q) - C_s(\sigma_r) == - \sum_{v \in \hat{\sigma}_p(X_p)} \mathrm{s}(v), \tag{20}$$

since the storage cost for the vertices in $\hat{\sigma}_p(X_p)$ is subtracted twice.

Therefore,

$$\begin{aligned} C(\sigma_p) \quad - \quad & C(\sigma_q) - C(\sigma_r) = W_{\text{total}} \cdot (\mathrm{MST}(\Pi_p) - \mathrm{MST}(\Pi_q) + \mathrm{MST}(\Pi)) - \\ & - \sum_{v \in X_p} \mathrm{r}(v) \cdot \delta(v, \hat{\sigma}_p(v)) - \sum_{v \in \hat{\sigma}_p(X_p)} \mathrm{s}(v), \end{aligned} \tag{21}$$

where $\Pi_p = \Pi_q \otimes \Pi_r$.

Recall that $\mathrm{char}(\sigma_p)$ is in $\mathrm{FS}(p)$ iff $C(\sigma_p)$ has minimum among all characteristics that that agree with $\mathrm{char}(\sigma_p)$ on the first three components. Similarly, for σ_q and σ_r. Note that $\mathrm{FS}(p)$ has a tuple $(\hat{\sigma}_p, \Pi_p, |\sigma_p(V_p)|, \chi_p)$ iff $\mathrm{FS}(q)$ and $\mathrm{FS}(r)$ have a tuple $(\hat{\sigma}_p, \Pi_q, |\sigma_q(V_q)|, \chi')$ and $(\hat{\sigma}_p, \Pi_r, |\sigma_r(V_r)|, \chi'')$ respectively. The difference $C(\sigma_p) - C(\sigma_q) - C(\sigma_r)$ depends only on $\hat{\sigma}_p(X_p)$, Π_q and Π_r ($\Pi_p = \Pi_q \otimes \Pi_r$). Therefore, $C(\sigma_p)$ is minimum if and only if both $C(\sigma_q)$ and $C(\sigma_r)$ are. Consequently, $\mathrm{char}(\sigma_p) \in \mathrm{FS}(p)$ if and only if $\mathrm{char}(\sigma_q) \in \mathrm{FS}(q)$ and $\mathrm{char}(\sigma_r) \in \mathrm{FS}(r)$. ■

Lemma 13 *Let p be a leaf node in the tree–decomposition of G and let x be the single node in $X_p = V_p$. The full set of characteristics* $\mathrm{FS}(p)$ *consists of the characteristics of all the $|V|$ replica assignments $\sigma_p : \{x\} \to V$.*

Proof.
Follows from Lemma 9 and the definition of characteristics of a replica assignment. ■

8 Computing Optimal Replica Assignments

8.1 Algorithm Description

Our algorithm is based on a technique, introduced by Arnborg and Proskurowski [1] and Bern [2], described by Bodlaender [4].

The cost of an optimal replica assignment for G can be computed from the full set of characteristics of the root of $\mathcal{T}$. The full set of characteristics can be computed in polynomial time since the terminal graphs of leaf nodes have one vertex. Moreover, Lemmas 10, 11,

and 12, allow us to compute the full set of characteristics for all other nodes of the tree–decomposition $\mathcal{T}$ of G in a bottom–up from FS()'s of their children only. We provide pseudocode for accomplishing that in Figures 1, 2, 3, 4, and 5.

```
OptimalReplication(G, k)
1     compute the distance graph Δ[V] of G;
2     compute a nice tree–decomposition T of G;
3     foreach node p of T, in preorder, do
4         if p is an introduce node then
5             Call ComputeFSofIntroduceNode(p);
6         else if p is a forget node then
7             Call ComputeFSofForgetNode(p);
8         else if p is a join node then
9             Call ComputeFSofJoinNode(p);
10        else
11            Call ComputeFSofLeafNode(p);
12        end;
13    end;
14    let r be the root node of T;
15    find a tuple S_k = (σ̂_r, Π_r, i, χ) ∈ FS(r) with minimum cost χ and i ≤ k;
16    return S_k
```

Figure 1: Pseudocode for computing an optimal replica set and an optimal k–replica set for a graph G for a given k.

8.2 Analysis of Running Time

Lemma 14 *Let p be a node in the tree–decomposition $\mathcal{T}$ of G. The number of elements in the full set of characteristics* $\mathrm{FS}(p)$ *is* $\leq (t+1)!2^{t+1}n^{t+2} = O(n^{t+2})$. *Moreover,* $\mathrm{FS}(p)$ *can be computed from the full sets of characteristics of the children of p in time* $O((t+1)^3[(t+1)!2^{t+1}n^{t+2}]^2) = O(n^{2t+4})$.

Proof.

First, we find an upper bound on the size of $\mathrm{FS}(p)$. There are no more than $n^{|X_p|} \leq n^{t+1}$ replica assignments $\hat{\sigma}_p$. There are no more than $m!2^m \leq (t+1)!2^{t+1}$ partitions of $\hat{\sigma}_p(X_p)$, where $m = |\hat{\sigma}_p(X_p)| \leq t+1$. Further, $|\sigma_p(V_p)| \leq |V_p| \leq n$. Therefore,

$$|\mathrm{FS}(p)| \leq n^{|X_p|}(|\hat{\sigma}_p(X_p)|)!2^{|\hat{\sigma}_p(X_p)|}|V_p| \leq n^{t+1}(t+1)!2^{t+1}n = (t+1)!2^{t+1}n^{t+2}. \qquad (22)$$

ComputeFSofIntroduceNode(p)

```
1   let FS(p) = ∅;
2   let x be p's introduced vertex;
3   let q be the child node of p;
4   foreach tuple (σ̂_q, Π_q, k_q, χ_q) ∈ FS(q) do
5       foreach vertex v ∈ V do
6           let σ̂_p be a replica assignment compatible with σ̂_q(X_q) ∪ {v};
7           if σ̂_p is not an extension of σ̂_q then continue;
8           if σ̂_p(x) ∈ σ̂_q(V_q) then
9               let χ_p = χ_q + r(x) · δ(x, σ̂_p(x));
10              add the tuple (σ̂_p, Π_q, k_q, χ_p) to FS(p);
11          else
12              let Π_p consist of all the blocks of Π_q plus the block σ̂_p(x);
13              let χ_p = χ_q + r(x) · δ(x, σ̂_p(x)) + W_total · (MST(Pi_p) − MST(Π_q)) + s(σ̂_x);
14              add the tuple (σ̂_p, Π_p, k_q + 1, χ_p) to FS(p);
15          end;
16      end;
17  end;
18  return;
```

Figure 2: Pseudocode, based on Lemma 10, for computing the full set of characteristics of an introduce node given the full set of characteristics of its child node. Note that when adding a tuple z to FS(p), if FS(p) already has a tuple y that agrees with z on the first three components, we simple set y's cost to the minimum cost of z and y.

Next, we consider the time it takes to compute FS(p) from those of its children. There are four cases to consider.

- p is an introduce node. From the pseudocode in Figure 2, we have FS(p) can be computed in time $O(|\text{FS}(q)|((t+1)!2^{t+1}+t^2)n) = O([(t+1)!2^{t+1}]^2 n^{t+3})$.
- p is a forget node. From the pseudocode in Figure 3, we have that FS(p) can be computed in time $O(|\text{FS}(q)|(2^{t+1}+(t+1)^2)) = O((t+1)!2^{2t+2}n^{t+2})$.
- p is a join node. Evaluating $\Pi_q \otimes \Pi_r$ can be done in time $O(t^3)$. Then, from the pseudocode in Figure 4, we have that FS(p) can be computed in time $O(|\text{FS}(q)||\text{FS}(r)|((t+1)^3+(t+1)^2)) = O([(t+1)!2^{t+1}n^{t+2}]^2(t+1)^3)$.
- p is a leaf node. From the pseudocode in Figure 5, we have that FS(p) can be computed in time $O(n)$.

Therefore, FS(p) can be computed in time $O((t+1)^3[(t+1)!2^{t+1}n^{t+2}]^2) = O(n^{2t+4})$. ■

ComputeFSofForgetNode(p)

let FS(p) $= \emptyset$;
let x be p's forgotten vertex;
let q be the child node of p;
foreach tuple $(\hat{\sigma}_q, \Pi_q, k_q, \chi_q) \in$ FS(q) do
    foreach each subset Π of blocks of Π_q do
        let π be the set of all the vertices $\neq \hat{\sigma}_q(x)$ of the blocks of Π;
        let Π_p consist of all the blocks in $\Pi_q - \Pi$ plus the block π;
        let $\chi_p = \chi_q + W_{\text{total}} \cdot (\text{MST}(\Pi_p) - \text{MST}(\Pi_q) + \text{MST}(\Pi))$;
        let $\hat{\sigma}_p$ be the restriction of $\hat{\sigma}_q$ to X_p;
        add the tuple $(\hat{\sigma}_p, \Pi_p, k_q, \chi_p)$ to FS(p);
    end;
end;
return;

Figure 3: Pseudocode, based on Lemma 11, for computing the full set of characteristics of a forget node given the full set of characteristics of its child node.

Theorem 1 *Let G be a graph with n nodes and treewidth bounded by a constant t. Let k be an integer $1 \leq k \leq n$. We can find an optimal replica set for G, with $\leq k$ replicas, in time $O((t+1)^3[(t+1)!2^{t+1}n^{t+2}]^2 n) = O(n^{2t+5})$.*

Proof.
Consider the pseudocode in Figure 1. The time to find a nice tree–decomposition $\mathcal{T}$ of G with $O(n)$ nodes is $O(n)$. Let root($\mathcal{T}$) be the root node of $\mathcal{T}$. From Lemma 14, we can compute the full set of characteristics of all the nodes in the tree–decomposition in time

$$O((t+1)^3[(t+1)!2^{t+1}n^{t+2}]^2 n) = O(n^{2t+5}). \tag{23}$$

An optimal replica set, with $\leq k$ replicas, for G can be found by inspecting FS(root($\mathcal{T}$)), which can be done in time

$$O(n^{t+1}(t+1)!2^{t+1}n) = O((t+1)!2^{t+1}n^{t+2}). \tag{24}$$

The procedures given in Figures 1– 5 find just the cost of an optimal replica set for G with at most k replicas. It is easy to modify the pseudocode to compute a corresponding replica assignment, without an increase in the asymptotic running–time of the algorithm (e.g. by recording during the bottom–up pass the elements from FS(q) and FS(r) that lead to each element in FS(p), and then using them, once a minimum cost elemt in the FS(root($\mathcal{T}$)) is selected, to construct a replica assignment). ■

ComputeFSofJoinNode(p)

```
1     let FS(p) = ∅;
2     let q and r be the two children nodes of p;
3     foreach tuple (σ̂_q, Π_q, k_q, χ_q) ∈ FS(q) do
4         foreach tuple (σ̂_r, Π_r, k_r, χ_r) ∈ FS(r) do
5             if σ̂_q = σ̂_r then
6                 let Π_p = Π_q ⊗ Π_r;
7                 let k_p = k_q + k_r − |σ̂_q(X_q)|;
8                 let y = Σ_{v∈X_q} r(v)δ(v, σ̂_q(v)) + Σ_{u∈σ̂_q(X_q)} s(u);
9                 let χ_p = χ_q + χ_r + W_total(MST(Π_p) − MST(Π_q) − MST(Π_r)) − y;
10                add the tuple (σ̂_q, Π_p, k_p, χ) to FS(p);
11            end;
12        end;
13    end;
14    return;
```

Figure 4: Pseudocode, based on Lemma 12, for computing the full set of characteristics of a join node given the full sets of characteristics of its childlen nodes.

ComputeFSofLeafNode(p)

```
1     let FS(p) = ∅;
2     let x be the single vertex in X_p;
3     foreach vertex v ∈ V do
4         let σ̂_p : {x} → {v};
5         let χ_p = C(σ̂_p);
6         let Π_p = ({v});
7         add the tuple (σ̂_p, Π_p, 1, χ_p) to FS(p);
8     end;
9     return;
```

Figure 5: Pseudocode, based on Lemma 13, for computing the full set of characteristics of a leaf node.

References

[1] S. Arnborg and A. Proskurowski. Linear time algorithms for NP–hard problems restricted to partial k–trees. *Discrete Applied Mathematics*, 23:11–24, 1989.

[2] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8:216–235, 1987.

[3] Hans L. Bodlaender. A liner–time algorithm for finding tree–decompositions of small treewidth. *SIAM Journal of Computing*, 25(6):1305–1317, 1996.

[4] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97; Lecture Notes in Computer Science*, volume 1295, pages 29–36. Springer-Verlag, Berlin, 1997.

[5] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21:358–402, 1996.

[6] Pei Cao and Sandy Irani. Cost-aware proxy caching algorithms. In *In Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December, 1997.

[7] Lawrence W. Dowdy and Derrell V. Foster. Comparative models of the file assignment problem. *ACM Computing Surveys*, 14(2):287–313, 1982.

[8] Marshall L. Fisher and Dorit S. Hochbaum. Database location in computer networks. *Journal of the ACM*, 27(4):718–735, 1982.

[9] S. Guha and Shamir Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th ACM–SIAM Symposium on Discrete Algorithms*, 1998.

[10] A. Heddaya and S. Mirdad. Webwave: Globally load balanced fully distributed caching of hot documents. In *In Proceedings of the 17th IEEE Intl. Conference on Distributed Computing Systems*, May, 1997.

[11] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[12] Konstantinos Kalpakis, Koustuv Dasgupta, and Ouri Wolfson. Optimal placement of replicas in trees with read–write–storage costs and capacity constraints. *IEEE Transactions on Parallel and Distributed Systems (under review)*, page 22, 1999.

[13] O. Kariv and S. L. Hakimi. An algorithmic approach to location problems. ii: The p–medians. *SIAM Journal of Applied Mathematics*, 37(3):539–560, 1979.

[14] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal of Computing*, 13(1):182–196, 1984.

[15] I.Rabinovich M.Rabinovich and R.Rajaraman. Dynamic replication on the internet. Technical report, AT&T Bell Labs, Tecnical Report HA6177000-98305-01-TM, 1998.

[16] M. Seltzer S. Manley, M. Courage. A self-scaling and self-configuring benchmark for web servers. Technical report, Technical Report TR-17-97, November, 1997.

[17] D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th ACM STOC*, pages 265–274, 1997.

[18] Jan van Leeuwen. Graph algorithms. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 525–631. MIT Press, 1990.

[19] O. Wolfson and A. Milo. The multicast policy and its relationship to replicated data placement. *ACM Transactions on Database Systems*, 16(1):181–205, 1991.

[20] A. Fiat Y. Bartal and Y. Rabani. Competitive algorithms for distributed data management. In *Proceedings of the 24th ACM STOC*, pages 39–50, 1992.

Contents