Approximation Algorithms for Data Placement in Arbitrary Networks

Ivan D. Baev* Rajmohan Rajaraman†

Abstract

We study approximation algorithms for placing replicated data in arbitrary networks. Consider a network of nodes with individual storage capacities and a metric communication cost function, in which each node periodically issues a request for an object drawn from a collection of uniform-length objects. We consider the problem of placing copies of the objects among the nodes such that the average access cost is minimized. Our main result is a polynomial-time constant-factor approximation algorithm for this placement problem. Our algorithm is based on a careful rounding of a linear programming relaxation of the problem. We also show that the data placement problem is MAXSNP-hard.

We extend our approximation result to a generalization of the data placement problem that models additional costs such as the cost of realizing the placement. We also show that when object lengths are nonuniform, a constant-factor approximation is achievable if the capacity at each node in the approximate solution is allowed to exceed that in the optimal solution by the length of the largest object.

1 Introduction

Consider a set of distributed caches in a large-scale information system such as a digital library, an information repository within an organization, or the World Wide Web. A powerful paradigm to improve cache effectiveness is *cooperation*, whereby caches cooperate in serving one another's requests and in making storage decisions. Such cooperation is particularly attractive in environments where the caches trust one another such as within an Internet service provider, a web hosting service, or a corporate intranet. Cooperative caching helps for two reasons: it prevents excessive replication by having caches access objects from other nearby caches and it allows a busy cache to utilize a nearby idle cache. A number of studies, beginning with the taxonomy developed in [15] have discussed the benefits of cooperative caching in distributed file systems and large-scale information systems. These studies include analytical results (e.g., [5, 21, 24, 35]), simulation experiments (e.g., [8, 16, 18, 33]) and prototypes and products (e.g., Harvest [9, 10], xFS [2, 14]).

This paper considers the *data placement* component of cooperative caching, which determines a static placement of replicated objects among the nodes of a network in accordance with a given access pattern. More precisely, given a network of nodes with a communication cost function, individual storage capacities of the node caches, and a demand function describing the access pattern of each node for each object, we study the problem of determining a placement of objects to the caches such that the average access cost, taken over all nodes and all objects, is minimized.

Overview of the model. The "cost" of communication in wide-area networks is a function of many parameters, including edge delays, edge capacities, buffer space, communication overhead, and patterns of user communication. Ideally, we would like to take all of these factors into account when optimizing performance; such a task, however, may not be feasible in general because the network parameters interact in a complex manner. For this reason, we adopt a simplified model in which the combined effect of the detailed network parameter values is captured by a single function that specifies the cost of communicating a fixed-length message between any given pair of nodes. As in several previous studies [3, 7, 22, 25], we assume that the cost function defines a metric; i.e., it satisfies nonnegativity, symmetry, and the triangle inequality.

We evaluate the quality of a given placement by the average cost of an access request under the placement, the average being taken over all nodes and all objects. Since a placement may place multiple copies of an object in the network, we need to specify the cost of satisfying an access request for a given object at a given node. We assume that any request at a node is satisfied by a copy of the requested object that is *nearest* to the node. This assumption is justified by the existence of

*Dept. of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. Email: `baev@ece.neu.edu`.

†College of Computer Science, Northeastern University, Boston MA 02115. Email: `rraj@ccs.neu.edu`. Supported by NSF CAREER award NSF CCR–9983901.

distributed directory services that direct each request to a nearby, if not the nearest, copy of the requested object [6, 16, 29, 34].

Our problem formulation is most suitable for applications where the objects are rarely written and the global pattern of accesses does not change rapidly. For simplicity, we assume throughout that the objects are read-only. Our results also apply to the case where the objects are written infrequently and there is a separate mechanism to maintain consistency among the replicas.

Overview of our results. The main result of this paper is a polynomial-time constant-factor approximation algorithm for the data placement problem with uniform-length objects. Our algorithm, described and analyzed in Section 4, is based on a careful rounding of a linear programming relaxation of the problem. Our rounding scheme builds on techniques developed recently for the k-median problem [11] and consists of a series of transformations to the problem instance and the fractional LP solution. A major technical challenge in the rounding process is to preserve the individual node capacity constraints in the final integral solution, while only giving up a constant factor in the approximation. The problem transformations that we perform during rounding lead to a fractional solution in which the assignment of individual node demands to fractional object copies forms a flow. This enables the formulation of an appropriate minimum cost flow problem that captures the capacity constraints. We finally invoke the flow integrality theorem to derive an integral solution. We also show that the placement problem is MAXSNP-hard (see Section 3), thus indicating that our approximation result is asymptotically the best possible.

We extend our approximation results in two directions in Section 5. First, we derive a constant-factor approximation for a generalization of the placement problem that also models the cost of realizing the placement. Second, we give a polynomial-time algorithm for the placement problem with non-uniform object lengths. This algorithm yields a solution with cost within a constant factor of the optimal cost, assuming a slight increase in the capacity of each node. It can be easily shown that when object lengths are non-uniform, it is NP-hard to obtain any polynomial-time approximation algorithm without resource augmentation.

Related work. The data placement problem, even though formulated in the specialized context of accessing a distributed data repository, can be viewed as a generalization of the facility location problem with *multiple* types of facilities and constraints on the number of facilities located at a point. Indeed, as mentioned above, our approximation algorithms draw on several techniques developed for the k-median and facility location problems in recent years. For a survey of results related to facility location see [13, 32].

Dowdy and Foster [15] initiated the study of cooperative caching in the context of allocating files in a distributed network. A sequence of results [3, 7, 25] describe improved algorithms for centralized as well as distributed file allocation. These results, however, do not consider cache capacities at the individual nodes. Awerbuch, Bartal, and Fiat [4] provide a polylog(n)-competitive on-line algorithm for the general placement problem under the assumption that the size of each cache in the on-line algorithm is polylog(n) times more than the size in the optimal algorithm. In contrast, we obtain a constant-factor approximation algorithm for the off-line version of the problem on arbitrary networks without any blowup in the cache sizes.

Leff, Wolf, and Yu [24] study the placement problem for a network of workstations, which they model as a single-level hierarchy. In addition to providing an optimal centralized algorithm, they give heuristics for a distributed solution. These results have been improved upon in [23], where exact and approximation algorithms are given for the placement problem in hierarchical networks. In a recent experimental study [22], Korupolu and Dahlin evaluate the practical performance of several placement and replacement algorithms including the ones developed in [23] for cooperative caching in hierarchical networks.

By adopting a fixed cost function as our communication model, we endeavor to separate the concerns of caching (a higher-level operation) from routing (a lower-level operation). In contrast, several previous studies have incorporated network topology and routing information into the caching problem. For example, the algorithms developed in [19, 30, 35] tend to cache copies of an object in nodes that are close to the path along which the object is being transferred. Routing information is also used in the placement algorithms developed in [26, 28], where the primary goal is to minimize network congestion that may occur when access requests and objects are routed within the network.

2 Problem definition

Let $\mathcal{N}$ be a network of n nodes and let Ψ be a collection of m objects. For each pair of nodes i and j, let $cost(i,j)$ denote the cost of transmitting a unit-length message between these two nodes. We assume that the cost function defines a metric space; that is, it is nonnegative, symmetric and satisfies the triangle inequality. Each node i, which may act as both a client and a server, has capacity $Size(i)$ of space devoted to the storage of the objects in Ψ. The n nodes periodically issue access requests for the m objects, the rate of which is given by the demand function d. For each node j and object α, $d(j, \alpha)$ represents the frequency of node i accessing object α.

A placement P is a function $\mathcal{N} \to 2^{\Psi}$ that yields for each node the set of objects stored in that node. For a placement to be valid, the sum of the lengths of the objects stored at any node must be at most the capacity of the node. For any node j and object α, the demand-weighted cost $access(j, \alpha)$ of accessing α at j equals $d(j, \alpha) \cdot cost(i, j) \cdot length(\alpha)$, where i is the node nearest to j that has a copy of α in the placement and $length(\alpha)$ is the length of object α. The *cost* of a placement is given by the sum, taken over all nodes j and all objects α, of $access(j, \alpha)$.

The data placement problem can be written as an integer linear program (ILP) as follows. For each object α in Ψ, let binary variable $y(i, \alpha), i \in \mathcal{N}$, indicate if node i is selected to store a copy of object k, and binary variable $x(i, j, \alpha), i, j \in \mathcal{N}$, indicate if node j is assigned to access the copy of object α stored at node i.

$$\min \sum_{\alpha \in O} \sum_{i,j \in N} d(j, \alpha) \cdot cost(i, j) \cdot length(\alpha) \cdot x(i, j, \alpha)$$

subject to

$$\begin{aligned}
(2.1) \qquad \sum_{i \in N} x(i, j, \alpha) &= 1 && j \in \mathcal{N},\ \alpha \in \Psi \\
x(i, j, \alpha) &\leq y(i, \alpha) && i, j \in \mathcal{N},\ \alpha \in \Psi \\
\sum_{\alpha \in \Psi} length(\alpha) \cdot y(i, \alpha) &\leq Size(i) && i \in \mathcal{N} \\
x(i, j, \alpha) &\in \{0, 1\} && i, j \in \mathcal{N},\ \alpha \in \Psi \\
y(i, \alpha) &\in \{0, 1\} && i \in \mathcal{N},\ \alpha \in \Psi.
\end{aligned}$$

3 Hardness of approximability

This section establishes two results on the hardness of approximating the optimal data placement. Our first result concerns the data placement problem with uniform-length objects. By means of an approximation-preserving reduction from a special case of the uncapacitated facility location problem, we show that the problem is MAXSNP-hard.

THEOREM 1. *The data placement problem with uniform-length objects is MAXSNP-hard.*

Proof: We use a reduction from the metric uncapacitated facility location problem, henceforth referred to as UFL. In UFL, we are given a set $N = \{1, \ldots, n\}$ of n locations and a subset $F \subseteq N$ of locations at which we may open a facility. For each location i in F, there is a cost f_i for opening a facility at i. Each location j in N has demand d_j. Given a set S of open facility locations, the demand at any location j in N is served by the location in S nearest to j. For any two locations i and j, we have a cost c_{ij} of shipping a unit of demand from i to j. These costs form a metric. The objective in UFL is to determine a set of open facilities such that the total facility and shipping cost is minimized.

Our reduction is from a special case of UFL in which the facility cost at each node in F is identical. It has been shown in [17] that UFL with uniform facility costs is MAXSNP-hard. Given such an instance of UFL, we construct the following instance of the metric data placement problem. The set of nodes is the set N of n locations in UFL and a special node Γ. The communication cost function among the nodes in N is the same as the shipping cost in UFL, while the communication cost between any node i in N and Γ is given by a large number M which is set to be the maximum among all the distances in the UFL instance. We have $|F| + 1$ objects, an object labeled 0, and an object having a label for every node in F. For each j in $\{1, \ldots, n\}$, if the demand at j in UFL is d_j, then in the metric data placement instance, j has demand d_j for object 0. In addition, each node j in F has demand f/M for object j, where f is the facility cost in the UFL instance. Finally, each node in F has capacity 1, each node in $N - F$ has capacity 0, and node Γ has capacity $|F| + 1$. This completes the construction of the instance of the data placement problem.

For any solution S for the given UFL instance with cost C, we construct the following placement that also has cost C. Each node i in S stores a copy of object 0, each node i in $F - S$ stores a copy of object i, while node Γ stores a copy of every object. The total access cost for object 0, taken over all nodes in N, equals the shipping cost in the UFL instance, while the total access cost for the objects in $F - S$ equals the facility cost. Hence the total cost of the placement is C.

Given any placement P for the data placement instance with cost C, we now show that if S is the set of nodes in F that have a copy of the object 0 in P, then the cost of the solution S for the UFL instance is at most C. Clearly, the shipping cost of S equals the total access cost for object 0 in P. Of the objects in F, only $|F - S|$ of them are stored in F; thus, the total access cost associated with the objects in F is at least $|S| \cdot M \cdot (f/M) = |S|f$, which is the facility cost of S.

We now establish the MAXSNP-hardness of the data placement problem. Let S^* be an optimal solution, with cost C^*, for a given instance of UFL. It follows that there exists a placement for the corresponding data placement instance with cost at most C^*. Since any placement for the data placement instance can be transformed into a solution to the UFL instance with no increase in cost, any ρ-approximation to the data placement problem implies a ρ-approximation to UFL, thus yielding a contradiction. ■

A reduction from the PARTITION problem establishes the following theorem. We defer the proof to the full version of the paper.

THEOREM 2. *There is no polynomial-time approximation for the data placement problem with non-uniform object lengths unless* $\mathbf{P} = \mathbf{NP}$. ■

4 A constant-factor approximation algorithm for uniform-length objects

Our approximation algorithm for uniform-length objects is obtained by rounding a linear programming relaxation of ILP (2.1) with $length(\alpha) = 1$ for every object α. We relax the integrality constraints for the binary variables to obtain a linear program (LP) in which $y(i,\alpha) \in [0,1]$ and $x(i,j,\alpha) \in [0,1]$ for all i,j,α. This LP, which can be solved efficiently, forms the basis for an approximation algorithm. Let (x^*, y^*) denote an optimal solution to the LP relaxation and let C^* denote its objective function value. By means of several transformations, we round (x^*, y^*) to a feasible solution to ILP (2.1) of cost at most $20.5C^*$. We begin with a brief outline of our rounding algorithm.

1. We first simplify the given problem instance $\mathcal{I}$ by consolidating demands of nearby nodes (along the lines of [12] for the k-median problem). We show that any integral solution for the new instance $\mathcal{I}_1$ can be transformed into a solution for $\mathcal{I}$ at the expense of at most $4C^*$ in total cost.
2. We next modify the assignment of "demand nodes" (the nodes with positive demand) to object copies so that at least a (1/2)-fraction of any requested object is accessible within a "local neighborhood" of the requesting node. Furthermore, the remaining at most (1/2)-fraction is satisfied by fractional copies located in a local neighborhood around the nearest demand node. This transformation increases the cost by a factor of at most 3.
3. We next change the fractional object locations and the fractional assignment such that for each demand node, either *exactly* a (1/2)-fraction or the entire object is satisfied by the fractional copies in the local neighborhood around the node. This is done without any increase in cost by solving an appropriately formulated minimum-cost flow problem involving all objects. We refer to the resulting solution as a *half-primary* solution.
4. For each object, we construct a directed graph, which we refer to as a *demand graph*, that has a vertex for each demand node and an arc from vertex j to vertex j' if the fractional object copies in the local neighborhood of j' together serve half an object to j. We perform another set of demand consolidations that moves demand from certain nodes to their neighbors in the demand graphs, and splits the demand graphs into a collection of one-level trees. This leads to a new instance $\mathcal{I}_2$ for which the current solution has cost no more than $15C^*/4$. Furthermore, any solution to the new instance can be transformed to a solution for $\mathcal{I}_1$ for an additive cost of $3C^*$.
5. On the basis of the one-level trees derived in Step 4, we perform a final set of demand and assignment consolidations. In the resulting fractional solution each fractional object copy serves *at most one* demand node. This implies that the assignment of demand to fractional object copies in the new solution forms a flow for the new instance $\mathcal{I}_3$. The increase in cost as a result of this transformation is at most $51C^*/4$.
6. Finally, we formulate instance $\mathcal{I}_3$ as a minimum-cost flow problem for which the fractional solution obtained after Step 5 is feasible. This flow problem involves all objects and captures the individual cache capacity constraints. Since the problem instance consists of integral edge capacities only, there is an optimal solution that is integral and can be computed in polynomial-time.

Steps 1 through 6 yield an integral placement for $\mathcal{I}_3$, which can then be converted to an integral placement for $\mathcal{I}$ via the reverse transformations arising out of Steps 5, 4, and 1. We obtain the following theorem.

THEOREM 3. *There exists a polynomial-time* 20.5*-approximation algorithm for the data placement problem with uniform-length objects.*

The following six subsections (Sections 4.1 through 4.6) describe Steps 1 through 6, respectively. In the following, we will adopt the notation that (x_k, y_k) is the solution obtained after Step k. In Section 4.7, we put together the claims of Sections 4.1 through 4.6 to derive Theorem 3.

4.1 Consolidating demands. We simplify the given problem instance $\mathcal{I}$ by consolidating demands of nearby nodes. We do not change the LP solution (x^*, y^*) but modify the demands to obtain a new instance $\mathcal{I}_1$. This transformation is separately applied for the individual objects. The demand consolidation we perform for a given object is identical to Step 1 of [12]. In the following, we briefly describe the consolidation step and state a lemma that bounds the resulting cost increase.

Let α be any object. Let the *average access cost* $C_{j\alpha}$ denote the cost that the optimal LP solution pays for assigning one unit of demand at node j for α. That is, $C_{j\alpha}$ equals $\sum_{i\in\mathcal{N}} cost(i,j)x^*(i,j,\alpha)$. Let $B_{j\alpha}$ denote the ball of radius $2C_{j\alpha}$ around j. We set the new demands such that for all pairs of nodes j, j', both with positive demand, $cost(j,j') > 2\max(C_{j\alpha}, C_{j'\alpha})$. We go through the nodes in nondecreasing order of their average access cost for object α and move the demands at certain nodes to others. Without loss of generality, let us renumber the nodes in nondecreasing order of their average access cost for α. While processing node j we perform the following operation: if there exists a node $j' < j$ with positive demand such that $cost(j,j') < 4C_{j\alpha}$, then we move the demand at j to j', and set the demand at j for α to be 0. It is easy to see that at the end of this step, for all pairs j, j' of nodes with positive demand, the condition $cost(j,j') > 4\max\{C_{j\alpha}, C_{j'\alpha}\}$ implies that the balls $B_{j\alpha}$ and $B_{j'\alpha}$ around j and j', respectively, do not overlap. Since the above transformation does not modify the solution, we have $(x_1, y_1) = (x, y)$. The following lemma establishes an upper bound on the increase in cost due to Step 1.

LEMMA 4.1. *The cost of (x_1, y_1) on instance $\mathcal{I}_1$ is at most C^*. Furthermore, any integral solution for instance $\mathcal{I}_1$ can be transformed into an integral solution for instance $\mathcal{I}$ at an additional cost of at most $4C^*$.* ■

In the remainder of Section 4, we use the term "demand node" for an object α to refer to any node that has positive demand for α.

4.2 Consolidating assignments. The second step transforms the assignment x_1 to a new assignment x_2. Fix object α and consider any demand node j for α. We transform the fractional assignment $x_1(\cdot, j, \alpha)$ to $x_2(\cdot, j, \alpha)$ as follows. We differentiate between three kinds of nodes that serve a fractional copy of α to node j. We refer to a node i in $B_{j\alpha}$ serving object α to node j as a *primary server* of α for j. We refer to a node $i \notin B_{j\alpha}$ serving α to j as a *secondary server* of α to j if j is the demand node for α that is nearest to i (breaking ties arbitrarily). We refer to any other node serving α to j as a *tertiary server* of α for j. For each primary and secondary server i, we keep the assignment $x_1(i,j,\alpha)$; that is, we set $x_2(i,j,\alpha) = x_1(i,j,\alpha)$. For each tertiary server i, we set $x_2(i,j,\alpha)$ to 0 and distribute the fractional assignment $x_1(i,j,\alpha)$ among the nodes in a ball $B_{j'\alpha}$ around j', where $j' \neq j$ is the demand node for α that is nearest to j. The preceding transformation leads to a valid fractional solution since the sum of fractional object copies in the primary servers for any node is at least $1/2$, while the fractional demand of j served by tertiary servers is at most $1/2$.

We now place an upper bound on the increase in cost incurred as a result of the transformation in Step 2. Let P, S, and T denote the total cost of accessing the objects from primary, secondary, and tertiary servers, respectively, in the solution (x_1, y_1) for instance $\mathcal{I}_1$. Since the assignment to primary and secondary servers is unchanged, the cost of accessing the objects from the primary and secondary servers in (x_2, y_2) is P and S, respectively. We note that $P + S + T$ equals C^*.

We now consider the cost of accessing objects from the tertiary servers. For any demand node j, the per-unit cost of accessing object α from a tertiary server is at most $cost(j,j') + 2C_{j'\alpha}$, where j' is the demand node for α that is nearest to j. Since $cost(j,j') \geq 4C_{j'\alpha}$, it follows that the per-unit cost of accessing α from a tertiary server is at most $3cost(j,j')/2$. While considering the cost of assigning to tertiary servers, we charge $3cost(j,j')/2$ per unit demand, and refer to the resulting cost as the *auxiliary tertiary cost*. We denote the auxiliary tertiary cost for α at node j by $t(j,\alpha)$. Throughout the remainder of our analysis, we maintain the *invariant* that the auxiliary tertiary cost is an *upper bound* on the total tertiary cost. Consider a tertiary server i of j for object α in x_1. We now show that the per-unit auxiliary tertiary cost for serving α to j in x_2 is at most $3cost(i,j)$. Let $j_1 \neq j$ be the demand node for α that is nearest to i. Since $cost(i,j) \geq cost(i,j_1)$, it follows from the triangle inequality that $cost(j,j') \leq cost(j,j_1) \leq cost(j,i) + cost(i,j_1) \leq 2cost(i,j)$. Therefore, the auxiliary tertiary cost of (x_2, y_2) is at most thrice the tertiary cost of (x_1, y_1). We thus have the following lemma.

LEMMA 4.2. *The primary and secondary cost of (x_2, y_2) for $\mathcal{I}_1$ equal P and S, respectively, while the auxiliary tertiary cost of (x_2, y_2) is at most $3T$.* ■

4.3 A half-primary solution. The solution (x_2, y_2) satisfies the following properties for each object α: (i) for each demand node j for object α, at least a $(1/2)$-fraction of the demand at j is satisfied by primary servers; (ii) the remaining at most $(1/2)$-fraction is served by secondary and tertiary servers. In Step 3, we construct a solution (x_3, y_3) in which for any demand node j of object α, the primary servers together serve either exactly a $(1/2)$-fraction of α or the whole object α to j. We refer to the solution thus obtained as a *half-primary* solution.

To obtain a half-primary solution, we formulate a minimum cost flow problem on a network N that is an extension of a bipartite graph $G = (A, B, E)$, where A and B form the bipartition of G and E is the set of edges between them. The set A contains two copies of each pair $\langle j, \alpha \rangle$ such that α is an object and j is a demand node for α. We refer to the two copies as *primary* and *nonprimary*. The set B consists of all nodes, together with a special node labeled Γ. We have an edge between the primary copy of $\langle j, \alpha \rangle$ in A to every node i in B that is a primary server of object α for j in the solution (x_2, y_2). This edge has a *lower capacity* of 0 and an *upper capacity* of 2. (For an edge in a minimum-cost flow problem, we use the terms lower capacity and upper capacity to refer to the lower and upper bounds, respectively, on the amount of flow that can pass through the edge.) The cost of the edge is $d_1(j, \alpha) cost(i, j)$, where d_1 is the demand function for instance $\mathcal{I}_1$. Similarly, we have an edge between the nonprimary copy of $\langle j, \alpha \rangle$ in A to every node i in B that is a secondary server of object α to j in the solution (x_2, y_2). This edge too has a lower capacity of 0 and an upper capacity of 2. The cost of the edge is $d_1(j, \alpha) cost(i, j)$. We also have an edge from the nonprimary copy of j to the special node Γ with lower capacity 0, upper capacity 2, and cost $t(j, \alpha)$. (Recall that $t(j, \alpha)$ is the auxiliary tertiary cost of accessing object α at node j.)

In addition to the nodes in G, the network N includes a set D of nodes and a distinguished source s and a distinguished sink t. The set D is another copy of the set of all node-object pairs. For every node-object pair $\langle j, \alpha \rangle$, there are two edges outgoing from its copy in D: (i) one to the primary copy of the pair in A with lower and upper capacities 1 and 2, respectively; and (ii) the other to the nonprimary copy of the pair in A with lower and upper capacities 0 and 1, respectively. The cost of both of these edges is 0. The source s has an edge to every node $\langle j, \alpha \rangle$ in D with a lower capacity of 2, upper capacity of 2, and cost 0. Finally, we have a sink t which has an edge incoming from every vertex i in B with a lower capacity of 0, upper capacity of $2 \mathit{Size}(i)$, and cost 0, and an edge incoming from the special vertex Γ with a lower capacity of 0, an upper capacity of ∞, and cost 0. The minimum-cost flow formulation is illustrated in Figure 1.

Lemmas 4.3 and 4.4 establish the correspondence between $\mathcal{I}_1$ and the minimum-cost flow problem.

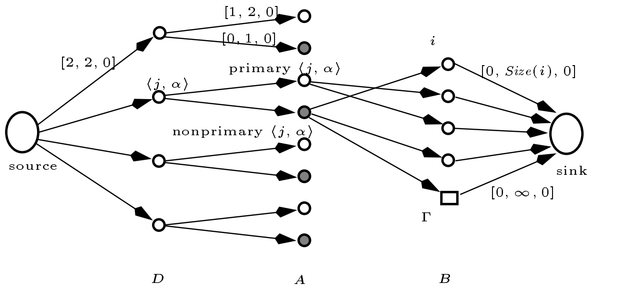

Figure 1: The minimum-cost flow problem formulated in Step 3. Each edge is labeled with a triple $[\ell, u, c]$, where ℓ, u, and c represent the lower capacity, upper capacity, and the cost of the edge, respectively. Labels have been shown for the edges from the source to vertices in D, from vertices in D to vertices in A, and from vertices in B to the sink. An edge from the vertex "primary $\langle j, \alpha \rangle$" in A to the vertex i in B has the label $[1, 2, d_1(j, \alpha) cost(i, j)]$, while an edge from the vertex "nonprimary $\langle j, \alpha \rangle$" to the vertex i in B has the label $[0, 1, d_1(j, \alpha) cost(i, j)]$. An edge from vertex "nonprimary $\langle j, \alpha \rangle$" to Γ has label $[0, 1, t(j, \alpha)]$.

LEMMA 4.3. *For any feasible flow f in N, there is a fractional solution for instance $\mathcal{I}_1$ with cost at most half of the cost of f.*

Proof: We construct a fractional solution (x^f, y^f) for instance $\mathcal{I}_1$ as follows. We set $y^f(i, \alpha)$ equal to half the total flow coming into i from any primary or nonprimary node labeled $\langle j, \alpha \rangle$ for any j. Clearly, y^f satisfies the capacity constraints. We now define the assignment x^f. For each pair of nodes i and j and object α, we set $x^f(i, j, \alpha)$ equal to half the flow from a primary or nonprimary node $\langle j, \alpha \rangle$ to i. (Note that there is an edge from at most one of the primary or nonprimary copies of $\langle j, \alpha \rangle$ to any other node i.) As a result of the assignment, the fraction of the demand at any node for object α that is satisfied by the primary servers is at least $1/2$. Furthermore, the cost incurred by the assignment thus far is exactly half the cost incurred along the edges to the nodes in the set B.

It still remains to assign a fraction of the demand for object α at node j that corresponds to half the flow from the nonprimary copy of $\langle j, \alpha \rangle$ in A to the special node Γ. This can be distributed among the primary servers of α for a node $j' \neq j$ which is the demand node nearest to j. Since there is at least half a copy of α located among the primary servers, the assignment is a valid one. Since $t(j, \alpha)$ is an upper bound on the per-unit cost of the assignment to tertiary servers, the upper bound on the cost follows. ■

LEMMA 4.4. *There is a flow in N with cost at most twice the cost of the solution (x_2, y_2) for instance $\mathcal{I}_1$.*

Proof: Consider the following flow along the edges from A to B. For an edge from a copy of $\langle j, \alpha \rangle$ to i, we have a flow of $2x_2(i, j, \alpha)$. Since $x_2(i, j, \alpha) \leq 1$, the capacity constraints at these edges are satisfied. The flow from nonprimary vertices $\langle j, \alpha \rangle$ to the special vertex Γ equals twice the fraction of the demand for object α at node j satisfied by tertiary servers.

We now define the flows from the vertices in D, the source, and the sink. The flow from a vertex $\langle j, \alpha \rangle$ in D

to a copy of $\langle j, \alpha \rangle$ in A is the sum of the flows going out of the copy in A. Since the fraction of the demand satisfied by the primary (resp., secondary) servers is at least $1/2$ (resp., 0) and at most 1 (resp., $1/2$), the capacity constraints on these edges are satisfied. The flow from the source s to $\langle j, \alpha \rangle$ in D is the sum of the flows going out of $\langle j, \alpha \rangle$ and the flow from vertex i in B to sink t is the sum of the flows coming into i. The capacity constraints at the edges coming out of s are also satisfied since the total fraction of the demand satisfied in solution (x_2, y_2) is 1. Finally, the capacity constraints at the edges coming into t are satisfied since the capacity constraints at the nodes are satisfied in (x_2, y_2).

Since the flows are twice the assignment values, the upper bound on cost immediately follows from the flow definitions. ■

Lemma 4.3 and 4.4 lead to the following claim.

LEMMA 4.5. *There exists a half-primary solution* (x_3, y_3) *for* $\mathcal{I}_1$ *with cost at most the cost of* (x_2, y_2).

Proof: By the integrality theorem, we can determine in polynomial time an integral flow f^* in N of minimum cost that satisfies all of the capacity constraints. From the integral flow f^*, we derive a new solution (x_3, y_3) using the correspondence defined in Lemma 4.3. Since f^* is integral, we see that the fraction of demand served by the primary servers of each node is either exactly $1/2$, or exactly 1. Moreover, the cost of (x_3, y_3) is at most the cost of (x_2, y_2). ■

At the end of Step 3, we have a half-primary solution (x_3, y_3) with total cost at most $P + S + 3T$.

4.4 Constructing demand graphs and consolidating demands. For each object α, we construct *demand graphs*, the edges of which indicate the assignment of demand at the demand nodes for α to tertiary servers. We construct a directed graph $D_\alpha(N', A')$ in which N' is the set of demand nodes for object α. We have an arc from node $j \in N'$ to node $j' \in N'$ if the primary servers in the ball $B_{j'\alpha}$ around j' are tertiary servers for j with respect to object α in the solution (x_3, y_3). Since each node has out-degree at most 1 and there is an edge from a demand node j to another demand node j' only if j' is the closest demand node to j, it follows that any cycle in each connected component of the graph D_α is of length at most 2. Furthermore, there is at most one such cycle in any component of D_α.

For each component of D_α, we select a root. If the component is a tree, then the root is simply the root of the tree. Otherwise, we break the unique 2-cycle in the component as follows. We move the demand from the node in the cycle with smaller demand to the node with larger demand. We then select the node with zero demand as the root. We define the *level* of each node as the distance to the root, in terms of the number of edges in D_α. By definition, each node j that is not a root has exactly $(1/2)$-fraction of its demand for α served by its primary and secondary servers, and exactly $(1/2)$-fraction served by the servers located in the ball $B_{j'\alpha}$, where j' is the parent of j in D_α.

We now consolidate the demand to obtain a new instance $\mathcal{I}_2$. If the total auxiliary tertiary cost of odd-level nodes is at most half the total auxiliary tertiary cost of all nodes, then we move the demand at every odd-level node of the demand graph (except the root) to its parent; otherwise, we move the demand at every even-level node (except the root) to its parent. Note that the solution (x_3, y_3) is a valid solution for $\mathcal{I}_2$.

We now place upper bounds on the cost of (x_3, y_3) and the increase in cost when a solution for $\mathcal{I}_2$ is transformed to a solution for $\mathcal{I}_1$ by simply moving the relevant demands back. Let M denote the set of nodes the demands of which are moved in this consolidation.

LEMMA 4.6. *The total primary and secondary cost of* (x_3, y_3) *for* $\mathcal{I}_2$ *is at most* $P + S + 3T/4$. *The total auxiliary tertiary cost of* (x_3, y_3) *for* $\mathcal{I}_2$ *is at most* $3T$.

Proof: The cost of (x_3, y_3) for $\mathcal{I}_2$ can be expressed in terms of the cost of the same solution for $\mathcal{I}_1$ as follows. For any node j in M, the per-unit auxiliary tertiary cost in $\mathcal{I}_2$ is at most the per-unit auxiliary tertiary cost in $\mathcal{I}_1$ since the distances along the arcs are nondecreasing with increasing level. Therefore, the total auxiliary tertiary cost of (x_3, y_3), when applied to $\mathcal{I}_2$, is at most $3T$. Furthermore, the per-unit primary and secondary cost at j for $\mathcal{I}_2$ is at most half the per-unit auxiliary tertiary cost at j for $\mathcal{I}_1$. Since the total auxiliary tertiary cost associated with the nodes in M is at most $3T/2$, it follows that the total primary and secondary cost of the solution (x_3, y_3), when applied to instance $\mathcal{I}_2$, is at most $P + S + 3T/4$. ■

LEMMA 4.7. *Any integral solution for* $\mathcal{I}_2$ *can be transformed to an integral solution for* $\mathcal{I}_1$ *at the cost of an additional* $3T$ *in total cost.*

Proof: We transform any solution for $\mathcal{I}_2$ into a solution for $\mathcal{I}_1$ as follows. We maintain the object copy locations as in the given solution. For a demand node j in instance $\mathcal{I}_1$ that is also a demand node in $\mathcal{I}_2$, we maintain the same assignment. Finally, for a demand node j of $\mathcal{I}_1$ that is not a demand node in $\mathcal{I}_2$, the assignment is the same as for that node j' to which the demand at j is moved as a result of the transformation. The per-unit cost increase at j equals the per-unit auxiliary tertiary cost at j of solution (x_3, y_3) for $\mathcal{I}_1$. Since half of the demand at j is satisfied by tertiary servers in the solution (x_3, y_3), and the total auxiliary tertiary cost of the nodes in M is at most $3T/2$, the increase in total cost is at most $3T$. ■

We note that Step 4 transforms instance $\mathcal{I}_1$ to $\mathcal{I}_2$ but does not modify the fractional solution. Thus, (x_4, y_4) equals (x_3, y_3).

4.5 A final consolidation of demands and assignments. Step 4 effectively decomposes each demand graph into several two-level trees, that is, a star configuration with a root at level 0 and leaves at level 1. In each such tree, the root has no demand. We make two transformations in Step 5. We first transform the solution (x_4, y_4) to a new solution (x_5, y_5) as follows. In each star, we arrange the leaves in left-to-right order according to decreasing demand. Consider any non-leaf node j with root r. In the assignment x_4, j is served half a copy of α by tertiary servers located in the ball $B_{r\alpha}$. In the new assignment x_5, we set the tertiary servers of α for j to be the servers located in the ball $B_{j'\alpha}$, where j' is the leaf node immediately to the left of j. The object copy locations remain as in y_4; therefore, $y_5 = y_4$. By triangle inequality, the auxiliary tertiary cost of the new solution is at most twice that of the old one, while the total primary and secondary cost remains unchanged. Therefore, the total primary and secondary cost is at most $P + S + 3T/4$, while the total auxiliary tertiary cost is at most $6T$.

We now compute a new instance $\mathcal{I}_3$ by consolidating assignments. Let the leaves of each star be numbered left to right, starting from 1. We move the demand from every even leaf node to its immediate left neighbor. We now place an upper bound on the cost of the solution (x_5, y_5) for the new instance $\mathcal{I}_3$.

LEMMA 4.8. *There exists a real number C such that the cost of (x_5, y_5) for instance $\mathcal{I}_3$ is at most $2P + 2S + 3T/2 + C$ and any integral solution for $\mathcal{I}_3$ can be transformed into an integral solution for $\mathcal{I}_2$ with an increase in cost of at most $12T - C$.*

Proof: Since the leaves are arranged left to right in increasing order of demand, the increase in the total primary and secondary cost due to the movement of the demand is at most the total primary and secondary cost of (x_5, y_5) in $\mathcal{I}_2$, which is at most $P+S+3T/4$. For the same reason, the new auxiliary tertiary cost can be upper bounded by twice the total auxiliary tertiary cost of the demand nodes in $\mathcal{I}_2$ with odd labels. Therefore, if $C/2$ is the total auxiliary tertiary cost of the demand nodes in $\mathcal{I}_2$ that have odd labels, the total primary and secondary cost (resp., auxiliary tertiary cost) of (x_4, y_4) for instance $\mathcal{I}_3$ is at most $2P + 2S + 3T/2$ (resp., C). We thus obtain that the cost of (x_4, y_4) for instance $\mathcal{I}_3$ is at most $2P + 2S + 3T/2 + C$.

What remains to be calculated is an upper bound on the increase in cost, when an integral solution for $\mathcal{I}_3$ is transformed into an integral solution for $\mathcal{I}_2$. Since the fraction of demand satisfied by tertiary servers is $1/2$, this transformation can be done at an additional cost of at most twice the auxiliary tertiary cost of (x_4, y_4) incurred by the evenly labeled demand nodes in instance $\mathcal{I}_2$, which is at most $2(6T - C/2) = 12T - C$. ■

4.6 Combining solutions for all objects. After Steps 1 to 5, we obtain a solution (x_5, y_5) in which each fractional copy, whether primary, secondary, or secondary, serves at most one demand node. The last step of our algorithm finds an integral solution to the problem. We define a minimum cost flow problem in which we have a bipartite graph $G = (A, B, E)$. The set A is the set of all node-object pairs with positive demand, while the set B is the set of all nodes. The flow network also includes a source s and sink t. There is an edge between each node-object pair $\langle j, \alpha \rangle$ and node i with lower capacity 0, upper capacity 1, and cost $d_3(j, \alpha) cost(i, j)$, where $d_3(j, \alpha)$ is the demand for α at j in $\mathcal{I}_3$. There is an edge from s to each node-object pair in A with lower capacity 1, upper capacity 1, and cost 0. Finally, there is an edge between each node j in B to t with lower capacity 0, upper capacity $Size(j)$ and cost 0. Clearly, the assignment x_5 defines a (fractional) flow through the network with cost equal to the total cost of the solution (x_5, y_5). Note that we crucially use the fact that for each object α, each node in B receives flow from at most one node-object pair of the type $\langle \cdot, \alpha \rangle$. Since all of the capacities in this minimum-cost flow problem are integral, we obtain an optimal integral solution (x_6, y_6) to the flow problem [1] in polynomial time. This solution yields the integral location of the copies as well as the assignment, thus giving an integral solution for $\mathcal{I}_3$.

4.7 Putting it all together. The solution (x_6, y_6) is an integral solution for instance $\mathcal{I}_3$ with cost $2P + 2S + 3T/2 + 2C$. We now obtain an integral solution to the original instance $\mathcal{I}$ by performing the reverse transformations mentioned in Steps 5, 4, and 1. By Lemmas 4.8, 4.7, and 4.1, the additional cost equals $12T - 2C + 3T + 4P + 4S + 4T = 4P + 4S + 19T - 2C$. Thus, the total cost is at most $6P+6S+41T/2$, which is at most $20.5C^*$. This completes the proof of Theorem 3.

5 Generalizations

This section extends the approximation result of Section 4 in two directions, one modeling additional costs, and the other involving non-uniform object lengths.

5.1 Modeling additional costs. In the data placement problem studied thus far, we have assumed that the access pattern is static; that is, the demand for each object at each node remains fixed. In practice, however, the demand function is likely to change with time. If the demands are not volatile and change at a moderate rate over time, then the data placement can be periodically recomputed to address the changes in the demand function. Realizing a new placement from an existing placement, however, may incur new costs since the objects need to be copied over to new locations.

Additional costs, such as transmissions required to obtain a new placement from an existing placement, can be easily modeled by adding a new cost component to the objective function. For each node i and object α, let $p(i, \alpha)$ denote the cost of placing object α at node i. (This cost is analogous to the facility cost in the facility location problem.) The cost of any placement equals the total demand-weighted access cost of all objects and nodes and the total cost of placing the object copies at the nodes according to the placement. The objective function in ILP (2.1) is redefined as

$$\min \sum_{\alpha \in \Psi} \sum_{i,j \in \mathcal{N}} d(j,\alpha) \cdot \mathit{cost}(i,j) \cdot x(i,j,\alpha) \cdot \mathit{length}(\alpha) + \sum_{\alpha \in \Psi} \sum_{i \in \mathcal{N}} p(i,\alpha) y(i,\alpha).$$

The additional cost of placing objects only affects those steps of our rounding scheme where the placement is modified. The transformation of the placement obtained after solving the linear program only occurs in the two minimum-cost flow calculations in Steps 3 and 6. In the full paper, we show that the flow problems can be reformulated such that the same approximation factor (20.5) is achieved for the new problem.

5.2 Non-uniform object lengths. This section considers the data placement problem with non-uniform object lengths. Theorem 2 of Section 3 states that no nontrivial approximation can be obtained for the problem in polynomial time, unless $\mathbf{P} = \mathbf{NP}$. We now show that a constant-factor approximation can be obtained in polynomial-time with "resource augmentation". More precisely, we show how to modify the algorithm of Section 4 so as to achieve the same approximation for the case of non-uniform object lengths, under the assumption that the capacity for each node in the approximate solution exceeds that in the optimal solution by the length of the largest object.

Steps 1, 2, 4, and 5 of the rounding algorithm in Section 4 change the demands and fractional assignments only. These steps continue to hold the same effect in the case where objects have non-uniform lengths. In Steps 3 and 6, minimum-cost flow computations are performed that capture the capacity constraints. In the scenario where objects have non-uniform lengths, we instead formulate appropriate instances of the *generalized assignment problem* [31].

We first consider Step 3. Let (x_2, y_2) denote the fractional solution obtained after Step 2, and let C be the total cost of the solution. As before, (x_2, y_2) satisfies the following properties for each object α: (i) for each demand node j for α, at least a $(1/2)$-fraction of α is satisfied by the primary servers of j; (ii) the remaining at most $(1/2)$-fraction is served by secondary and tertiary fractional servers. We formulate an instance Π_1 of the generalized assignment problem in which we have two clients, *primary* and *nonprimary*, for each demand node of the data placement problem, and we have a server for each node in the data placement problem. The primary and nonprimary clients corresponding to demand node j and object α each have a demand of $\mathit{length}(\alpha)/2$. Each server node has a capacity equal to its cache capacity. There is also a special node Γ with infinite capacity. The cost of an edge between a client and a server is $\mathit{length}(\alpha)$ times the cost of the edge in the original data placement problem. The cost of an edge between the vertex Γ and a client associated with the demand node j and object α equals $\mathit{length}(\alpha)$ times the per-unit auxiliary tertiary cost $t(j, \alpha)$. The given solution (x_2, y_2) yields a feasible fractional solution to Π_1 with total cost C. By [31], there is an integral solution to Π_1 with total cost C, which, though infeasible, requires at each server an additional capacity of at most the length of the maximum demand. Since the maximum demand equals half the length of the largest object, this implies that the capacity used up at any node i in the integral solution is at most $S/2$ more than $\mathit{Size}(i)$, where S equals the length of the largest object.

A similar formulation for Step 6 incurs another additive penalty of $S/2$ in the capacity of each node, without incurring any penalty in the cost of the solution. Thus, we have the following theorem.

THEOREM 4. *For the metric data placement problem with non-uniform object lengths, there exists a polynomial-time computable placement with cost at most* 20.5 *times the optimal cost, in which the capacity of each node exceeds the capacity of the node in the optimal solution by at most the length of the largest object.* ■

6 Concluding remarks

The approximation algorithms of Sections 4 and 5 rely on solving large linear programs and minimum-cost flow problems, which are computationally intensive and may not be amenable to efficient distributed implementations. An interesting direction for future research is to identify simple combinatorial algorithms that use greedy or local search approaches. Recent results in facility location using greedy and primal-dual techniques [20, 27] may offer some insight in this regard. Also of practical interest is a capacitated version of the data placement problem, in which the number of requests that any node can serve is bounded.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] T. E. Anderson, M. D. Dahlin, J. N. Neefe, D. A. Patterson, D. S. Rosselli, and R. Y. Wang. Serverless network file systems. In *Proceedings of the 15th Symposium on Operating Systems Principles*, pages 109–126, 1995.

[3] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 164–173, May 1993.

[4] B. Awerbuch, Y. Bartal, and A. Fiat. Heat & Dump: Competitive distributed paging. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 22–31, November 1993.

[5] B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28:67–104, 1998.

[6] B. Awerbuch and D. Peleg. Online tracking of mobile users. *Journal of the ACM*, 37:1021–1058, 1995.

[7] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and Systems Sciences*, 51:341–358, 1995.

[8] M. A. Blaze. Caching in large-scale distributed file systems. Technical Report TR-397-92, Department of Computer Science, Princeton University, January 1993. PhD Thesis.

[9] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28:119–125, 1995.

[10] A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell. A hierarchical Internet object cache. In *Proceedings of the USENIX 1996 Technical Conference*, pages 22–26, January 1996.

[11] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, October 1999.

[12] M. Charikar, S. Guha, D. Shmoys, and É. Tardos. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, May 1999.

[13] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In *Discrete Location Theory*, pages 119–171. Wiley, New York, 1990.

[14] M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pages 267–280, November 1994.

[15] D. Dowdy and D. Foster. Comparative models of the file assignment problem. *ACM Computing Surveys*, 14:287–313, 1982.

[16] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of the 1998 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 254–265, August 1998.

[17] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, January 1998.

[18] J. S. Gwertzman and M. Seltzer. The case for geographical push-caching. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems*, pages 51–57, May 1995.

[19] A. Heddaya and S. Mirdad. WebWave: Globally load balanced fully distributed caching of hot published documents. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 160–168, May 1997.

[20] K. Jain and V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, October 1999.

[21] D. Karger, E. Lehman, F. T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[22] M. Korupolu and M. Dahlin. Coordinated placement and replacement for large-scale distributed caches. In *Proceedings of the IEEE Workshop on Internet Applications*, pages 62–71, July 1999.

[23] M. Korupolu, C. G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 586–595, January 1999.

[24] A. Leff, J. L. Wolf, and P. S. Yu. Replication algorithms in a remote caching architecture. *IEEE Transactions on Parallel and Distributed Systems*, 4:1185–1204, 1993.

[25] C. Lund, N. Reingold, J. Westbrook, and D. Yan. On-line distributed data management. In J. van Leeuwen, editor, *Proceedings of the 2nd Annual European Symposium on Algorithms,* Lecture Notes in Computer Science, volume 855, pages 202–214. Springer-Verlag, 1994.

[26] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 284–293, October 1997.

[27] R. Mettu and C. G. Plaxton. The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, November 2000. To appear.

[28] Meyer auf der Heide, F. and Vöcking, B. and Westermann, M. Caching in networks. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 430–439, January 2000.

[29] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32:241–280, 1999.

[30] M. Rabinovich, I. Rabinovich, R. Rajaraman, and A. Aggarwal. A dynamic object replication and migration protocol for an Internet hosting service. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 101–113, May 1999.

[31] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[32] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, May 1997.

[33] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. Design considerations for distributed caching on the Internet. In *Proceedings of the 19th International Conference on Distributed Computing Systems*, pages 273–284, May 1999.

[34] M. van Steen, F. J. Hauck, and A. S. Tanenbaum. A model for worldwide tracking of distributed objects. In *Proceedings of the 1996 Conference on Telecommunications Information Networking Architecture (TINA 96)*, pages 203–212, September 1996.

[35] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM Transactions on Database Systems*, 22:255–314, 1997.