Optimal Methods for Coordinated Enroute Web Caching for Tree Networks

KEQIU LI
Japan Advanced Institute of Science and Technology
HONG SHEN
Japan Advanced Institute of Science and Technology and
University of Science and Technology of China
FRANCIS Y. L. CHIN
University of Hong Kong
and
SI QING ZHENG
University of Texas at Dallas

Web caching is an important technology for improving the scalability of Web services. One of the key problems in coordinated enroute Web caching is to compute the locations for storing copies of an object among the enroute caches so that some specified objectives are achieved. In this article, we address this problem for tree networks, and formulate it as a maximization problem. We consider this problem for both unconstrained and constrained cases. The constrained case includes constraints on the cost gain per node and on the number of object copies to be placed. We present dynamic programming-based solutions to this problem for different cases and theoretically show that the solutions are either optimal or convergent to optimal solutions. We derive efficient algorithms that produce these solutions. Based on our mathematical model, we also present a solution to coordinated enroute Web caching for autonomous systems as a natural extension of the solution for tree networks. We implement our algorithms and evaluate our model on different performance metrics through extensive simulation experiments. The implementation results show

Preliminary versions of the results of this article have partially appeared in Li and Shen [2003a] on unconstrained enroute Web caching and in Li and Shen [2003b] on constrained enroute Web caching for tree networks.
This work was supported by the Japan Society for the Promotion of Science (JSPS) under its General Research Scheme B Grant No. 14380139.
Corresponding author: H. Shen, shen@jaist.ac.jp.
Authors' addresses: K. Li, Graduate School of Information Science, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan; H. Shen, Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, 230026, China; email: shen@jaist.ac.ip; F. Y. L. Chin, Department of Computer Science, University of Hong Kong, Pokfulam Road, Hong Kong; S. Q. Zheng, Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688.

that our methods outperform the existing algorithms of either coordinated enroute Web caching for linear topology or object placement (replacement) at individual nodes only.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.4 [**Computer System Organization**]: Performance of Systems—*Design studies*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms: Algorithms, Design, Management, Performance

Additional Key Words and Phrases: Web caching, dynamic programming, object placement (replacement), performance evaluation, tree network, autonomous system (AS)

1. INTRODUCTION

With the explosive growth in popularity of the World Wide Web, prompt Web content delivery is becoming increasingly important. Web caching is an important technology for improving Web performance since caching objects close to users can save network bandwidth, alleviate server load, and reduce Internet access latency. An overview of Web caching can be found in Davison [2001] and Wang [1999].

An important technology for improving the efficiency of Web content delivery is to cache Web objects in various locations in the network such as servers, proxies, and clients [Davison 2001; Wang 1999]. Although Web caching is similar to memory caching in that they both store objects at different locations for future requests, significant differences exist between them as a result of the nonuniformity of Web object sizes, access frequencies, retrieval costs, and cacheability [Davison 2001; Tang and Chanson 2002]. In addition, the performance of Web caching depends greatly on the network distance from the user to the server since users are geographically distributed over the entire Internet.

To obtain the full benefits of Web caching, different architectures have been employed such as hierarchical caching [Rabinovich and Spatscheck 2002; Rodriguez et al. 2001] and distributed caching [Rodriguez et al. 2001; Tewari et al. 1999]. Enroute caching is a new caching architecture developed recently [Krishnan et al. 2000; Rodriguez and Sibal 2000; Tang and Chanson 2002] in which caches are placed on the access path from the user to the server. Each enroute cache intercepts any request that passes through its associated node and either satisfies the request by sending the requested object to the client or forwards the request upstream along the path to the server until it can be satisfied. Enroute Web caching can be implemented in several ways such as by using lightweight techniques [Rabinovich and Wang 2001; Rodriguez et al. 2000], active network [Tennenhouse et al. 1997], and so on.

Cooperative caching, in which caches cooperate in serving each other's requests and making storage decisions, is a powerful paradigm to improve cache effectiveness [Dahlin et al. 1994; Korupolu and Dahlin 2002; Korupolu et al. 1999]. The performance of enroute caching mainly relies on where the caches are placed and how the cache contents are managed. While the first issue has been extensively studied recently [Krishnan et al. 2000; Li et al. 1999], little attention has been paid to the second issue. In this article, we address one of the key problems in coordinated enroute Web caching, that is, for tree networks

to compute the locations of copies of an object to be placed among the enroute caches so that the overall cost gain is maximized. We consider this problem for both unconstrained and constrained cases where the constrained case includes constraints on the cost gain per node and on the number of copies to be placed. We also extend our solution to autonomous systems.

Our contributions are summarized as follows. (1) We present solutions to this problem for different cases applying a dynamic programming-based mathematical model and theoretically show that the solutions are either optimal or convergent to optimal solutions. (2) We derive algorithms for these solutions and show that their time complexities are very low. (3) We present a solution to autonomous systems as a natural extension of the solution for tree networks. (4) We perform extensive experiments to evaluate our model by several performance metrics. The simulation results show that our methods outperform existing algorithms of either coordinated enroute Web caching for linear topology or object placement at individual nodes only.

The rest of the article is organized as follows. Section 2 discusses related work on enroute Web caching. Section 3 gives a mathematical formulation of our problems. Section 4 and Section 5 present algorithms for unconstrained and constrained coordinated enroute Web caching for tree networks. Section 6 presents an algorithm for coordinated enroute Web caching for autonomous systems. Section 7 describes the simulation model and discusses the experimental results. Section 8 summaries our work and concludes the article.

2. RELATED WORK

There have been many studies on replacement algorithms for a single Web cache [Jin and Bestavros 2001; Scheuermann et al. 1997; Williams et al. 1996]. However, these algorithms store copies of an object at each node through which the object passes, without checking whether it is beneficial to do so. This may cause ineffective use of the limited cache space since there are numerous objects to be distributed in the network. Therefore, it is necessary and important to find methods that can optimally determine the locations to place the copies of an object. In Dahlin et al. [1994] and Leff et al. [1993], the placement and replacement algorithms for local-area networks were studied. However, this problem is relatively unexplored in wide-area networks which are very different from local-area networks in regard to the number of users and objects.

Cache cooperation is an important approach to improve Web performance. Recent studies have focused on the benefits of cooperative caching for distributed systems and large-scale systems [Awerbuch et al. 1998; Chankhunthod et al. 1996; Leff et al. 1993; Tewari et al. 1999]. In Yu and MacNair [1998], wide-area cache cooperation was studied under a simple model in which distances among all nodes in the network are assumed to be the same. In Korupolu et al. [1999], the authors examined three practical cooperative placement algorithms for large-scale distributed caches and showed that cooperative object placement could significantly improve Web performance compared to local replacement algorithms, particularly when the sizes of individual caches were small compared to those of the objects. In Korupolu and Dahlin [2002], the object placement

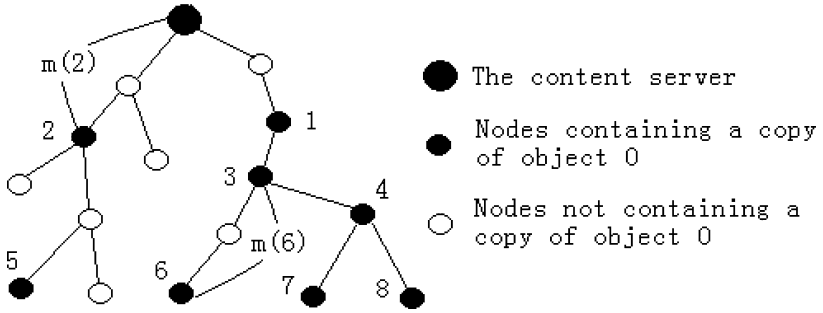

Fig. 1. Enroute Web caching for tree networks.

problem was formulated as an instance of the facility location problem and solved by reducing it to the minimum cost problem. The time complexity of this object placement problem is very high, at least quadratic of the product of the number of nodes and the number of objects in the network. Here, our problems are formulated as maximization problems from a different point of view and our algorithms are low-cost. In Korupolu and Dahlin [2002], two approximation algorithms, a greedy placement algorithm and an amortizing placement algorithm, were proposed. Although the greedy algorithm looks simple and is easy to implement, it has been shown that the performance of its worst case solution can be arbitrarily far from optimal, that is, the approximation ratio is relevant to the number of nodes concerned. The amortizing placement algorithm is a constant-factor approximation algorithm. The problem studied in Tang and Chanson [2002] considered the coordinated enroute Web caching problem for linear topology, deciding the optimal locations for placing copies of an object among the enroute caches. This scheme, which optimizes the placement of objects on the path from the user to the server, has been shown to perform significantly better than other schemes that considered either object placement or replacement in individual caches only. An enroute Web caching algorithm, applying dynamic programming for placing Web files in the tree network, was proposed in Jiang and Bruck [2003]. They established a model which does not consider the impact of caching a copy of a Web file at a node on the requests for this file from all downstream nodes of this node. This model can result in an optimal solution for placing a Web file at only one node on the path from client to server.

3. PROBLEM FORMULATION

Before formulating the problem for coordinated enroute Web caching, we introduce the notations and definitions used in this article. We model the network as a tree $T = (V, E)$ where V is the set of nodes, each of which is associated with an enroute cache, and E is the set of (bidirectional) network links. Without loss of generality, we assume there is only one content server at the root of a tree, where the objects requested by users are maintained. Our analysis can be easily extended to the case in which enroutes are associated with a subset of nodes only if we include in the graph the nodes with enroute caches. Figure 1 shows an example of such a tree topology. In this article, we use T_w to denote a tree whose root is w.

Let $P \subseteq V$ be a subset of nodes, at each of which a copy of an object is cached. For every node $v \in V$, $D(v)$ denotes the set of all nodes that are the descendants of node v, and $C(v)$ denotes the set of all nodes that are the children of node v. For any two nodes $u, v \in V$, we use $E[u \rightarrow v]$ to denote the set of all edges on the path between u and v, and $V[u \rightarrow v]$ to denote the set of all nodes on the path between u and v including u and v. Let O be the given object for caching. For notational tidiness, we omit argument O in all parameters and functions throughout the article. Let $f(v)$ denote the access frequency of object O which is defined by the number of requests to access O that pass through node v during a certain period of time. Obviously, $f(v) \geq \sum_{w \in C(v)} f(w)$. Estimation of $f(v)$ can be tedious and will be discussed in Section 7. Let $c(u, v)$ be a nonnegative cost assigned to edge $(u, v) \in E$ for object O which is defined by network latency, bandwidth consumption, and processing cost at the cache, or some combination of these measures, incurring on (u, v) for accessing O. The cost of a path for object O is the summation of all edge costs on the path. In this article, we assume that $O(\max_{v \in V}\{c(v, w)\}) = O(n)$ for tree T_w,

As we have mentioned, it is necessary and important to find methods to optimally distribute copies of an object among the enroute caches since the size of each cache is limited. Accordingly, when a new object is stored in a cache, one or more objects may need to be removed from the cache to make room for it. Storing an object at a node enables all the requests previously passing it now to be satisfied at it; hence, its access cost, which is defined in this article as cost saving, is decreased. Similarly, removing the copy of an object from a node increases its access cost, which is defined as cost loss. In this article, we consider cost saving and cost loss in a coordinated way.

Let $m(v)$ be the miss penalty of object O with respect to node v which is given by

$$m(v) = \sum_{(u_1, u_2) \in E[v \rightarrow v']} c(u_1, u_2), \tag{1}$$

where v' is the nearest higher-level node of v that stores a copy of object O (see Figure 1). Therefore, the *cost saving* for node $v \in P$ denoted by $s(v)$ is defined as

$$s(v) = (f(v) - f'(v))\, m(v), \tag{2}$$

where $f'(v)$ is the total access frequency of object O that can still be served by the original caches on the downstream of node v if the copy of object O stored at node v is removed. For instance, in Figure 1, $f'(1) = f(3)$, $f'(2) = f(5)$, $f'(3) = f(4) + f(6)$, $f'(4) = f(7) + f(8)$, $f'(5) = f'(6) = f'(7) = f'(8) = 0$.

Let $l(v)$ be the cost loss for storing a copy of object O at node v caused by removal of some cached objects at v due to limited cache space. Suppose that $O_1, O_2, \ldots, O_l$ are cached at node v. Obviously, the removed objects should introduce the least total cost loss while making enough room to accommodate the object to be cached. We apply the following greedy heuristic to decide replacement candidates. Note that the normalized cost loss (*NCL*, i.e., the cost loss introduced by creating one unit of free space) of ejecting O_i at v is $\frac{f_i(v) m_i(v)}{e_i}$, where $f_i(v)$ is the access frequency for object O_i observed at node v, $m_i(v)$ is the miss penalty of object O_i with respect to node v, and e_i is the size of object

O_i. The objects in the cache are ordered by their *NCL*s and are selected sequentially, starting from the object with the smallest *NCL* until enough space is created. Therefore, $l(v)$ can be calculated by summing the cost losses caused by all the selected objects removed.

Thus, the *cost gain* for (caching object O at) a single node v, denoted by $g(v)$, is defined as

$$g(v) = s(v) - l(v) = (f(v) - f'(v))\, m(v) - l(v). \tag{3}$$

Based on the cost gain for a single node, we formulate the general problem for coordinated enroute Web caching as a maximization problem as follows:

$$\begin{cases} \max\limits_{P \subset V} G(T, P) = \max\limits_{P \subset V} \left\{ \sum\limits_{v \in P} [(f(v) - f'(v))\, m(v) - l(v)] \right\}, \\ s.t.\ \ \mathcal{C} \end{cases} \tag{4}$$

where $\mathcal{C}$ is called the constraint space.

For unconstrained coordinated enroute Web caching (UCERWC), $\mathcal{C}$ is null, that is, there is no constraint.

For constrained coordinated enroute Web caching (CCERWC), we consider the following settings of $\mathcal{C}$:

—Nonnegative cost gain per node
$\mathcal{C} : (f(v) - f'(v))\, m(v) - l(v) \geq \alpha_v \geq 0 \ \ (\forall\, v \in P)$.
This constraint states that an object may be cached at node v if the cost gain for caching it at v is above a predefined threshold. Clearly, this threshold should be nonnegative to avoid any possible cost loss, making the caching practically beneficial.

—Placing exactly k copies
$\mathcal{C} : \ |P| = k$.
This constraint is to restrict the number of copies to be distributed. Since, in practice, caching an object will also generate overheads such as maintaining consistency between caches and the content server, it is necessary to discuss the case of placing a fixed number of copies.

—Placing at most k copies
$\mathcal{C}$: $|P| \leq k$.
This constraint sets an up-bound on the number of copies to be placed and allows freedom of placement within this bound to maximize the total cost gain. It can be viewed as an extension of the previous constraint but requires different technical treatment.

In Equation (4), $G(T, P)$ is the overall cost gain for placing copies of an object at each node in P in the constraint space. Regarding the solution to the constrained cases, we give the following definitions. A placement P is called a feasible solution if and only if P satisfies the relevant constraints. For example, if a placement P is a feasible solution to the constrained problem of nonnegative cost gain per node, then we have $(f(v) - f'(v))\, m(v) - l(v) \geq \alpha_v \ \ (\forall\, v \in P)$. On the contrary, if we have $(f(v) - f'(v))\, m(v) - l(v) \geq \alpha_v \ \ (\forall\, v \in P)$, then the placement P is a feasible solution to the constrained problem of nonnegative cost gain per

cost gain per node. A placement P^* is called an optimal solution if and only if P^* is a feasible solution and satisfies that $G(T, P^*) = \max_{P \subset V}\{G(T, P)\}$.

4. UNCONSTRAINED COORDINATED ENROUTE WEB CACHING FOR TREE NETWORKS

In this section, we focus on solving the unconstrained coordinated enroute Web caching problem for tree network topology.

Based on Equation (4), the unconstrained coordinated enroute Web caching problem for tree T_w is defined as follows:

$$\max_{A_w} G(T_w, A_w) = \max_{A_w} \left\{ \sum_{v \in A_w} [(f(v) - f'(v))\, m(v) - l(v)] \right\}, \tag{5}$$

where $A_w \subseteq D(w)$ and $f(v)$, $f'(v)$, $m(v)$, and $l(v)$ are the same as defined in Section 3. Suppose that A_w^* is an optimal solution to Equation (5) with respect to tree T_w, then we should store a copy of an object at each node in A_w^*, and $G(T_w, A_w^*)$ represents the relevant maximum overall cost gain.

Let $T_{w,x}$ be a subtree of T_w, whose node set is $V[w \rightarrow x] \cup D(x)$, where $x \in D(w)$. Similarly, we define the unconstrained coordinated enroute Web caching problem for tree $T_{w,x}$ as follows:

$$\max_{A_{w,x}} G(T_{w,x}, A_{w,x}) = \max_{A_{w,x}} \left\{ \sum_{v \in A_{w,x}} [(f(v) - f'(v))\, m(v) - l(v)] \right\}, \tag{6}$$

where $A_{w,x} \subseteq D(x) \cup \{x\}$. Suppose that $A_{w,x}^*$ is an optimal solution to Equation (6) with respect to tree $T_{w,x}$, then we should store a copy of an object at each node in $A_{w,x}^*$, and $G(T_{w,x}, A_{w,x}^*)$ represents the relevant maximum overall cost gain.

Before presenting our dynamic programming-based algorithm for solving Equation (5), we give the following lemmas or theorems.

LEMMA 1. *For tree T_w, if $C(w) = \{w_1, w_2, \ldots, w_m\}$, then we have*

$$G\left(T_w, \cup_{i=1}^m A_{w,w_i}\right) = \sum_{i=1}^m G(T_{w,w_i}, A_{w,w_i}), \tag{7}$$

where $A_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$, $i = 1, 2, \ldots, m$.

PROOF. Since $A_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$, we have $\cup_{i=1}^m A_{w,w_i} \subseteq \cup_{i=1}^m (D(w_i) \cup \{w_i\}) = D(w)$. Since $A_{w,w_i} \cap A_{w,w_j} = \phi$ for $i \neq j$, by the definition of $G(T_w, A_w)$, we have

$$\begin{aligned} G\left(T_w, \cup_{i=1}^m A_{w,w_i}\right) &= \sum_{v \in \cup_{i=1}^m A_{w,w_i}} [(f(v) - f'(v))\, m(v) - l(v)] \\ &= \sum_{i=1}^m \sum_{v \in A_{w,w_i}} [(f(v) - f'(v))\, m(v) - l(v)] \\ &= \sum_{i=1}^m G(T_{w,w_i}, A_{w,w_i}). \end{aligned}$$

Hence, the lemma is proven. □

LEMMA 2. *If* $C(w) = \{w_1, w_2, \ldots, w_m\}$ *and* $A'_{w,w_i} = A^*_w \cap (D(w_i) \cup \{w_i\})$, *then we have* $A^*_w = \cup_{i=1}^m A'_{w,w_i}$, *where* $A^*_w \subseteq D(w)$ *is an optimal solution to Equation* (5) *with respect to tree* T_w.

PROOF. Since $C(w) = \{w_1, w_2, \ldots, w_m\}$ and $A'_{w,w_i} = A^*_w \cap (D(w_i) \cup \{w_i\})$, we have

$$\begin{aligned} \cup_{i=1}^k A'_{w,w_i} &= \cup_{i=1}^k (A^*_w \cap (D(w_i) \cup \{w_i\})) \\ &= A^*_w \cap \cup_{i=1}^k (D(w_i) \cup \{w_i\}) \\ &= A^*_w \cap D(w) = A^*_w. \end{aligned}$$

Hence, the lemma is proven. □

THEOREM 1. *For tree* T_w, *if* $C(w) = \{w_1, w_2, \ldots, w_m\}$, *then we have*

$$A^*_w = \cup_{i=1}^m A^*_{w,w_i}, \tag{8}$$

where $A^*_w \subseteq D(w)$ *is an optimal solution to Equation* (5) *with respect to tree* T_w, *and* $A^*_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$ *is an optimal solution to Equation* (6) *with respect to tree* T_{w,w_i}, $i = 1, 2, \ldots, m$.

PROOF. For $A^*_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$, we have $\cup_{i=1}^m A^*_{w,w_i} \subseteq \cup_{i=1}^m (D(w_i) \cup \{w_i\}) = D(w)$. Since A^*_w is an optimal solution to Equation (5) with respect to tree T_w, we have $G(T_w, A^*_w) \geq G(T_w, \cup_{i=1}^m A^*_{w,w_i})$. Let $A'_{w,w_i} = A^*_w \cap (D(w_i) \cup \{w_i\})$ by Lemma 2, then we have $A^*_w = \cup_{i=1}^m A'_{w,w_i}$. Obviously, $A'_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$, so we have $G(T_{w,w_i}, A'_{w,w_i}) \leq G(T_{w,w_i}, A^*_{w,w_i})$ since $A^*_{w,w_i} \subseteq D(w_i) \cup \{w_i\}$ is an optimal solution to Equation (6) with respect to tree T_{w,w_i}. By Lemma 1, we have

$$\begin{aligned} G(T_w, A^*_w) &= G(T_w, \cup_{i=1}^m A'_{w,w_i}) = \sum_{i=1}^m G(T_{w,w_i}, A'_{w,w_i}) \\ &\leq \sum_{i=1}^m G(T_{w,w_i}, A^*_{w,w_i}) = G\big(T_w, \cup_{i=1}^m A^*_{w,w_i}\big). \end{aligned}$$

Therefore, we have $G(T_w, A^*_w) = G(T_w, \cup_{i=1}^m A^*_{w,w_i})$, so we have $A^*_w = \cup_{i=1}^m A^*_{w,w_i}$. Hence, the theorem is proven. □

THEOREM 2. *For tree* $T_{w,x}$, *if* $C(x) = \{x_1, x_2, \ldots, x_k\}$, *then we have*

$$A^*_{w,x} = \begin{cases} \cup_{i=1}^k A^*_{w,x_i} & G(T_{w,x}, \cup_{i=1}^k A^*_{w,x_i}) \geq G(T_{w,x}, A^*_x \cup \{x\}) \\ A^*_x \cup \{x\} & G(T_{w,x}, \cup_{i=1}^k A^*_{w,x_i}) < G(T_{w,x}, A^*_x \cup \{x\}), \end{cases} \tag{9}$$

where $A^*_{w,x} \subseteq D(x) \cup \{x\}$ *is an optimal solution to Equation* (6) *with respect to tree* $T_{w,x}$, $A^*_x \subseteq D(x)$ *is an optimal solution to Equation* (5) *with respect to tree* T_x, *and* $A^*_{w,x_i} \subseteq D(x_i) \cup \{x_i\}$ *is an optimal solution to Equation* (6) *with respect to tree* T_{w,x_i}, $i = 1, 2, \ldots, k$.

PROOF. It is easy to see that $A^*_x \cup \{x\} \subseteq (D(x) \cup \{x\})$, and $\cup_{i=1}^m A^*_{w,x_i} \subseteq (D(x) \cup \{x\})$. For node x, we consider the following two cases. One case is that a copy of an object is stored at node x, that is, $x \in A^*_{w,x}$, and the other case is that no copy of that object is placed there, that is, $x \notin A^*_{w,x}$.

(1) First, we prove $A^*_{w,x} = A^*_x \cup \{x\}$ for $x \in A^*_{w,x}$. Let $A'_{x,x_i} = A^*_{w,x} \cap (D(x_i) \cup \{x_i\}$, by Lemma 2, then we have $A^*_{w,x} = \cup_{i=1}^k A'_{x,x_i} \cup \{x\}$. Therefore, we have

$$\begin{aligned}
G(T_{w,x}, A^*_{w,x}) &= G\big(T_{w,x}, \cup_{i=1}^k A'_{x,x_i} \cup \{x\}\big) \\
&= \sum_{v \in (\cup_{i=1}^k A'_{x,x_i} \cup \{x\})} [(f(v) - f'(v))m(v) - l(v)] \\
&= \sum_{v \in \cup_{i=1}^k A'_{x,x_i}} [(f(v) - f'(v))m(v) - l(v)] + [(f(x) - f'(x))m(x) - l(x)] \\
&= \sum_{i=1}^k \sum_{v \in A'_{x,x_i}} [(f(v) - f'(v))m(v) - l(v)] + [(f(x) - f'(x))m(x) - l(x)] \\
&= \sum_{i=1}^k G(T_{x,x_i}, A'_{x,x_i}) + [(f(x) - f'(x))m(x) - l(x)] \\
&= G\big(T_x, \cup_{i=1}^k A'_{w,x_i}\big) + [(f(x) - f'(x))m(x) - l(x)] \\
&\le G(T_x, A^*_x) + [(f(x) - f'(x))m(x) - l(x)].
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
G(T_{w,x}, A^*_{w,x}) &\ge G(T_{w,x}, A^*_x \cup \{x\}) \\
&= \sum_{v \in A^*_x} [(f(v) - f'(v))m(v) - l(v)] + [(f(x) - f'(x))m(x) - l(x)] \\
&= G(T_x, A^*_x) + [(f(x) - f'(x))m(x) - l(x)].
\end{aligned}$$

Therefore, we have $G(T_{w,x}, A^*_{w,x}) = G(T_{w,x}, A^*_x \cup \{x\})$, so we have $A^*_{w,x} = A^*_x \cup \{x\}$ for $x \in A^*_{w,x}$.

(2) Now, we prove $A^*_{w,x} = \cup_{i=1}^m A^*_{w,x_i}$ for $x \notin A^*_{w,x}$. Let $A'_{w,x_i} = A^*_{w,x} \cap (D(x_i) \cup \{x_i\})$ then by Lemma 2, we have $A^*_{w,x} = \cup_{i=1}^k A'_{w,x_i}$. Therefore, we have

$$\begin{aligned}
G(T_{w,x}, A^*_{w,x}) &= G\big(T_{w,x}, \cup_{i=1}^k A'_{w,x_i}\big) \\
&= \sum_{v \in \cup_{i=1}^k A'_{w,x_i}} [(f(v) - f'(v))m(v) - l(v)] \\
&= \sum_{i=1}^k \sum_{v \in A'_{w,x_i}} [(f(v) - f'(v))m(v) - l(v)] \\
&= \sum_{i=1}^k G(T_{w,x_i}, A'_{w,x_i}) \\
&\le \sum_{i=1}^k G(T_{w,x_i}, A^*_{w,x_i}) \\
&= G\big(T_{w,x}, \cup_{i=1}^k A^*_{w,x_i}\big).
\end{aligned}$$

Since $G(T_{w,x}, A^*_{w,x}) \ge G(T_{w,x}, \cup_{i=1}^k A^*_{w,x_i})$, we have $G(T_{w,x}, A^*_{w,x}) = G(T_{w,x}, \cup_{i=1}^k A^*_{w,x_i})$. So we have $A^*_{w,x} = \cup_{i=1}^k A^*_{w,x_i}$ for $x \notin A^*_{w,x}$.

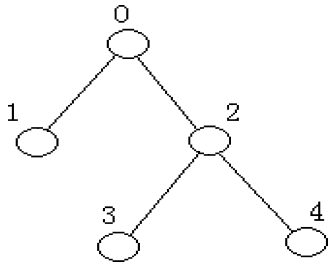

Fig. 2. A simple example.

From (1) and (2), we can know that $G(T_{w,x}, A^*_{w,x}) = \max\{G(T_{w,x}, A^*_x \cup \{x\}), G(T_{w,x}, \cup_{i=1}^{k} A^*_{w,x_i})\}$. Therefore, it is easy to see that the theorem is correct. □

By Theorem 2, we can see that for tree $T_{w,x}$, if $G(T_{w,x}, \cup_{i=1}^{k} A^*_{w,x_i}) \geq G(T_{w,x}, A^*_x \cup \{x\})$, then we do not store a copy of an object at node x and further consider the subtrees $\{T_{w,x_i}, i = 1, 2, \ldots, k\}$, where $C(x) = \{x_1, x_2, \ldots, x_k\}$. Otherwise, we store a copy at node x and further consider the subtree T_x.

Based on Theorem 1 and Theorem 2, we can present the dynamic programming-based algorithm for unconstrained coordinated enroute Web caching for tree networks as follows:

Algorithm 1: Unconstrained Coordinated Enroute Web Caching for Tree Networks
Step 1. Initialization:
$A^*_w = \phi$ and $G(T_w, A^*_w) = 0$;
Step 2. End condition
if $D(w) = \phi$ then return;
Step 3. Recursive procedure
for $v \in C(w)$ do
  if $D(v) = \phi$ then
    if $f(v)c(w, v) - l(v) > 0$ then
      $A^*_{w,v} = \{v\}$
    else
      $A^*_{w,v} = \phi$
  else
    for $x \in C(v)$ do
      if $\sum_{x \in C(v)} G(T_{w,x}, A^*_{w,x}) \geq G(T_v, A^*_v) + (f(v) - f'(v))c(w, v) - l(v)$ then
        $A^*_{w,v} = \cup_{x \in C(v)} A^*_{w,x}$
      else
        $A^*_{w,v} = A^*_v \cup \{v\}$ (According to Theorem 2)
  $A^*_w = \cup_{v \in C(w)} A^*_{w,v}$ (According to Theorem 1)

Now we give in Figure 2 a simple example to show how Algorithm 1 works. First, we decompose tree T_0 into two subtrees, $T_{0,1}$ and $T_{0,2}$. For tree $T_{0,1}$, we can get an optimal placement by calculating $g(1)$ directly. For tree $T_{0,2}$, we can further decompose it into either subtrees, $T_{0,3}$ and $T_{0,4}$, or tree T_2 according to the relationship between $G(T_{0,3}, A^*_{0,3}) + G(T_{0,4}, A^*_{0,4})$ and $G(T_2, A^*_2) + g(2)$. Obviously, $G(T_{0,3}, A^*_{0,3})$ and $G(T_{0,4}, A^*_{0,4})$ can be solved directly, therefore, what we should do is to further decompose tree T_2 until $G(T_2, A^*_2)$ can be solved directly. Accordingly, we can obtain an optimal placement for tree T_0.

From Algorithm 1, we can know that every cache should maintain some information on the objects, including size, access frequency, update frequency, and miss penalty with the associated node. Fortunately, it is not necessary to store the information of an object at all the nodes in the network. The following theorem describes an important property of Algorithm 1.

THEOREM 3. *If A_w^* is an optimal solution to Equation* (5) *with respect to tree T_w, then we have $f(v)c(v,w) - l(v) \geq 0, \ \forall\, v \in A_w^*$.*

PROOF. Suppose there exists $x \in A_w^*$ that satisfies $f(x)c(x,w) - l(x) < 0$, then we have

$$\begin{aligned}
G(T_w, A_w^*) &= \sum_{v \in A_w^*} [(f(v) - f'(v))\, m(v) - l(v)] \\
&= \sum_{v \in (A_w^* - \{x\})} [(f(v) - f'(v))\, m(v) - l(v)] + [(f(x) - f'(x))c(x,x') - l(x)] \\
&< \sum_{v \in (A_w^* - \{x\})} [(f(v) - f'(v))\, m(v) - l(v)] + [f(x)c(x,w) - l(x)] \\
&< \sum_{v \in (A_w^* - \{x\})} [(f(v) - f'(v))\, m(v) - l(v)] \\
&= G(T_w, A_w^* - \{x\}),
\end{aligned}$$

which contradicts the fact that A_w^* is an optimal solution to Equation (5) with respect to tree T_w. Hence, the theorem is proven. □

From Theorem 3, we can easily see that we should consider placing a copy of an object only among the caches where object caching is locally beneficial. Here, locally beneficial means that the cost gain is greater than zero if we put only one copy of an object among the enroute caches.

Regarding the time complexity of Algorithm 1, we have the following theorem.

THEOREM 4. *If all nodes are locally beneficial, then the time complexity of Algorithm* 1 *is $O(n^2)$, where n is the total number of nodes in the network.*

PROOF. From Algorithm 1, we can easily know that the time complexity of it is $O(\sum_{v \in V} |D(v)|)$, where $|D(v)|$ is cardinality of the set $D(v)$. Since $|D(v)| \leq n-1$, we have $O(\sum_{v \in V} |D(v)|) \leq O(\sum_{v \in V}(n-1)) = O(n(n-1)) = O(n^2)$. Hence, the theorem is proven. □

5. CONSTRAINED COORDINATED ENROUTE WEB CACHING FOR TREE NETWORKS

In this section, we concentrate on solving the constrained coordinated enroute Web caching problem for the case in which the network topology is a tree,

determining the locations for placing copies of an object among the enroute caches such that the overall cost gain is maximized under different constraints, including nonnegative cost gain per node, placing exactly k copies, and placing at most k copies of an object among the enroute caches.

5.1 Constraint I: Nonnegative Cost Gain Per Node

Suppose that $B_w \subseteq D(w)$ is a subset of nodes of tree T_w. Based on Equation (4), the constrained coordinated enroute Web caching problem of nonnegative cost gain per node for tree T_w is defined as follows:

$$\begin{cases} \max\limits_{B_w} \; G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))\, m(v) - l(v)] \right\}. \\ s.t. \;\; (f(v) - f'(v))\, m(v) - l(v) \geq \alpha_v \;\; (\forall\, v \in B_w) \end{cases} \tag{10}$$

Suppose that B_w^* is an optimal solution to Equation (10), then we should store a copy of an object at each node in B_w^*, and $G(T_w, B_w^*)$ represents the relevant maximum overall cost gain.

Before developing the dynamic programming-based algorithm for solving Equation (10), we propose the following mathematical proofs.

LEMMA 3. *If $A_w^* \subseteq D(w)$ is an optimal solution to Equation* (5) *with respect to tree T_w, then we have $(f(v) - f'(v))\, m(v) - l(v) \geq 0, \;\; \forall\, v \in A_w^*$.*

PROOF. Suppose that there exists $x \in A_w^*$ that satisfies $(f(x) - f'(x))m(x) - l(x) < 0$, then we have

$$\begin{aligned} G(T_w, A_w^* - \{x\}) &= \sum_{v \in A_w^* - (D(x) \cup \{x\})} [(f(v) - f'(v))\, m(v) - l(v)] \\ &\quad + \sum_{v \in D(x) \cap A_w^*} [(f(v) - f'(v))\, m(v) - l(v)] \\ &> \sum_{v \in A_w^* - (D(x) \cup \{x\})} [(f(v) - f'(v))\, c(v, v') - l(v)] + [(f(x) - f'(x))\, m(x) - l(x)] \\ &\qquad + \sum_{v \in D(x) \cap A_w^*} [(f(v) - f'(v))\, c(v, v') - l(v)] \\ &\geq \sum_{v \in A_w^*} [(f(v) - f'(v))\, c(v, v') - l(v)]^1 \\ &= G(T_w, A_w^*), \end{aligned}$$

which contradicts the fact that A_w^* is an optimal solution to Equation (5) with respect to tree T_w. Hence, the theorem is proven. □

[1]This is because $f'(v)$ becomes larger for the nodes upstream of node x and $c(v, v')$ smaller for the nodes downstream of node x when a copy of an object is cached at node x.

THEOREM 5. *If $A_w^* \subseteq D(w)$ is an optimal solution to Equation (5) with respect to tree T_w, then A_w^* is also an optimal solution to the following equation:*

$$\begin{cases} \max\limits_{B_w} G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))m(v) - l(v)] \right\}. \\ s.t.\ (f(v) - f'(v))m(v) - l(v) \geq 0 \ \ (\forall\, v \in B_w) \end{cases} \tag{11}$$

PROOF. Since A_w^* is an optimal solution to Equation (5) with respect to tree T_w, by Lemma 3, we have $(f(v) - f'(v))m(v) - l(v) \geq 0, \forall\, v \in A_w^*$, therefore, A_w^* is a feasible solution to Equation (11), so we have $G(T_w, A_w^*) \leq G(T_w, B_w^*)$. Now we suppose $G(T_w, A_w^*) < G(T_w, B_w^*)$. It is obvious that B_w^* is a feasible solution to Equation (5), therefore, we have $G(T_w, A_w^*) = G(T_w, A_w^*)$ and $G(T_w, B_w^*) = G(T_w, B_w^*)$, so we have $G(T_w, A_w^*) < G(T_w, B_w^*)$. This contradicts the fact that A_w^* is an optimal solution to Equation (5) with respect to tree T_w. Hence, the theorem is proven. □

By Theorem 5 , we can obtain an optimal solution to Equation (11) by solving Equation (5). It is easy to see that Equation (10) can be transformed into the following equation:

$$\begin{cases} \max\limits_{B_w} G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))m(v) - l(v) - \alpha_v] + \sum\limits_{v \in B_w} \alpha_v \right\}. \\ s.t.\ (f(v) - f'(v))m(v) - l(v) - \alpha_v \geq 0 \ \ (\forall\, v \in B_w) \end{cases} \tag{12}$$

Furthermore, it is obvious that Equation (12) is equivalent to the following equation.

$$\begin{cases} \max\limits_{B_w} G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))m(v) - l(v) - \alpha_v] \right\}. \\ s.t.\ (f(v) - f'(v))m(v) - l(v) - \alpha_v \geq 0 \ \ (\forall\, v \in B_w) \end{cases} \tag{13}$$

By Theorem 5, we can get an optimal solution to Equation (13) by solving the following equation:

$$\max_{A_w} G(T_w, A_w) = \max_{A_w} \left\{ \sum_{v \in A_w} [(f(v) - f'(v))m(v) - l(v) - \alpha_v] \right\}. \tag{14}$$

Therefore, we can obtain an optimal solution to Equation (10) by solving Equation (14).

By now, we have proved that the constrained coordinated enroute Web caching problem as described in Equation (10) can be transformed into an unconstrained coordinated enroute Web caching problem as described in Equation (14). Based on the algorithm proposed in Section 4, we present the following dynamic programming-based algorithm for constrained coordinated enroute Web caching of nonnegative cost gain per node, which is described as follows.

Algorithm 2: UCERWC—Nonnegative Cost Gain Per Node
Step 1. Initialization
$A_w^* = \phi$ and $G(T_w, A_w^*) = 0$;
Step 2. End Condition
if $D(w) = \phi$ then return;
Step 3. Recursive Procedure
for $v \in C(w)$ do
  if $D(v) = \phi$ then
    if $f(v)c(w,v) - l(v) - \alpha_v > 0$ then
      $A_{w,v}^* = \{v\}$
    else
      $A_{w,v}^* = \phi$
  else
    for $x \in C(v)$ do
      if $\sum_{x \in C(v)} G(T_{w,x}, A_{w,x}^*) \geq G(T_v, A_v^*) + [(f(v) - f'(v))c(w,v) - l(v) - \alpha_v]$ then
        $A_{w,v}^* = \cup_{x \in C(v)} A_{w,x}^*$
      else
        $A_{w,v}^* = A_v^* \cup \{v\}$
  $A_w^* = \cup_{v \in C(w)} A_{w,v}^*$

Similar to Algorithm 1, we can know that every cache should maintain some information on the objects, including size, access frequency, update frequency, and miss penalty with the associated node. For Algorithm 2, the constraint for each node should also be maintained. The following corollary describes an important property of Algorithm 2.

COROLLARY 1. *If A_w^* is an optimal solution to Equation* (14)*, then we have* $f(v)c(v,w) - l(v) - \alpha_v \geq 0, \ \forall\, v \in A_w^*$.

The proof of Corollary 1 is similar to that of Theorem 3.

From Corollary 1, we can easily see that we should consider placing a copy of an object only among the caches where object caching is locally beneficial. Here, locally beneficial means that the cost gain for each node is greater than the constraint for that node if we put only one copy of an object among the caches.

Regarding the time complexity of Algorithm 2, we have the following corollary.

COROLLARY 2. *If all nodes are locally beneficial, then the time complexity of Algorithm* 2 *is* $O(n^2)$*, where n is the total number of nodes in the network.*

The proof of Corollary 2 is similar to that of Theorem 4.

5.2 Constraint II: Placing Exactly *k* Copies of An Object

Similarly, the constrained coordinated enroute Web caching problem of placing exactly k copies of an object among the enroute caches for tree T_w is defined as follows:

$$\begin{cases} \max\limits_{B_w} \ G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))m(v) - l(v)] \right\}. \\ s.t. \ \ |B_w| = k \end{cases} \tag{15}$$

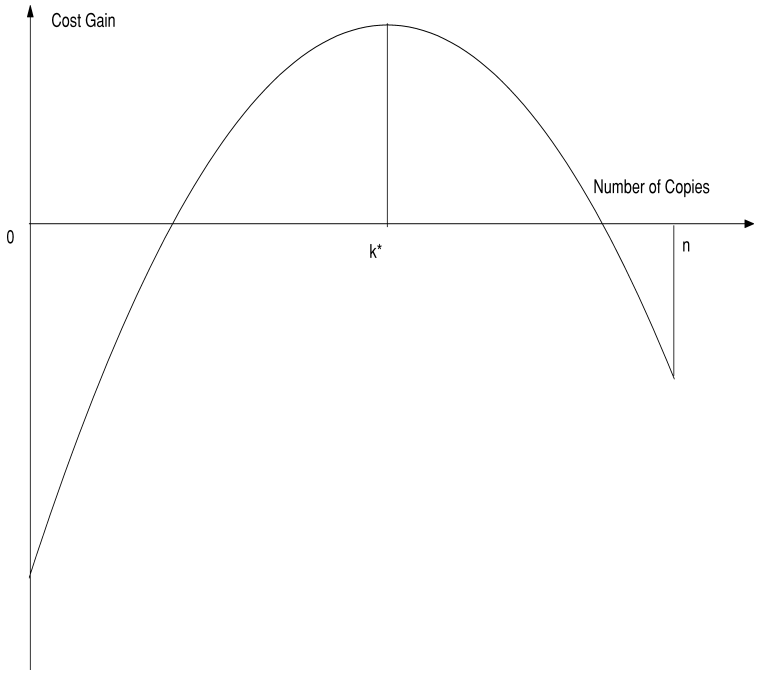

Fig. 3. Relationship between cost gain and number of copies.

Suppose that A_w^* is an optimal solution to Equation (5) with respect to tree T_w, we can easily know that it is not necessary to place more than k^* copies of an object among the enroute caches where $k^* = |A_w^*|$. Otherwise, there must be at least one node whose cost gain is negative. Therefore, k should be less than k^*. So we first compute k^* by Algorithm 2 by setting $\alpha_v = 0$, and the optimal locations are all the nodes in A_w^* when $k \geq k^*$. The relationship between the overall cost gain and the number of copies can be visualized from Figure 3.

Before presenting the algorithm for solving the problem of placing exactly k copies of an object among the enroute caches for tree T_w, we give the following definition. The local cost gain for node v, denoted by $h(v)$, is the cost gain for placing only one copy of an object in the enroute cache at node v, which is given by

$$h(v) = f(v)c(v, w) - l(v). \tag{16}$$

Let $\alpha_{\max} = \max_{v \in V}\{h(v)\}$, then we have the following theorem with respect to a feasible solution to Equation (10).

THEOREM 6. *if $\alpha_v > \alpha_{\max}$, then there is no feasible solution to Equation* (10).

PROOF. Suppose that there exists a node v that satisfies $(f(v) - f'(v))m(v) - l(v) \geq \alpha_v$. On the other hand, we have

$$\begin{aligned}(f(v) - f'(v))m(v) - l(v) &\leq (f(v) - f'(v))c(v, w) - l(v) \\ &\leq f(v)c(v, w) - l(v) \leq h(v) < \alpha_v.\end{aligned}$$

So, the supposition is not correct. Hence, the theorem is proven. □

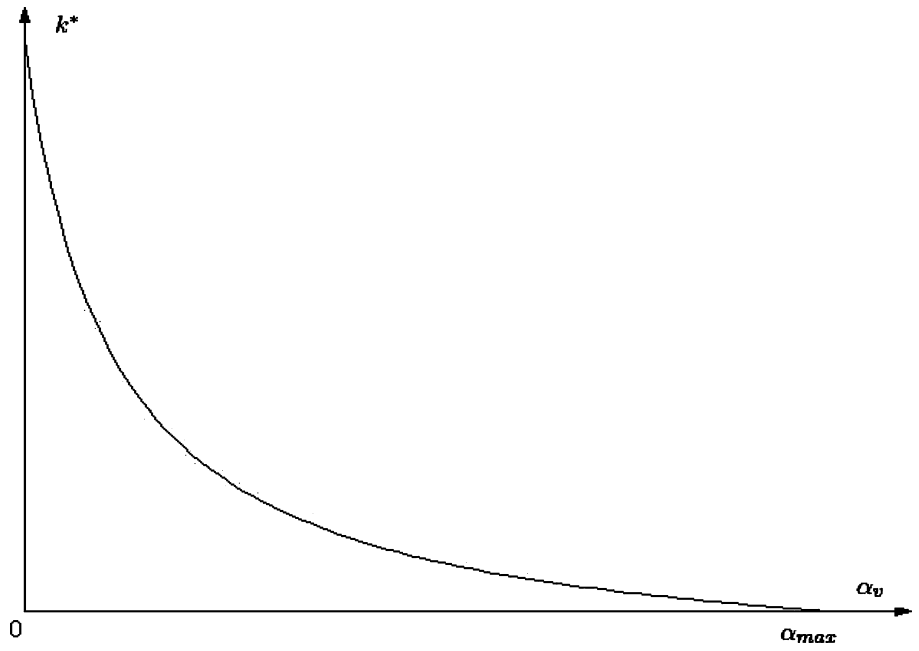

Fig. 4. Relationship between k^* and α_v.

From Theorem 6, we can know that the parameter α_v in Equation (10) should satisfy $0 \leq \alpha_v \leq \alpha_{\max}$. It is obvious that the number of copies of an object to be placed in the network is relevant to the parameter α_v. Hence, the proper selection of α_v determines the number of caching locations. The crucial observation is that the number of caching locations is a monotonically-decreasing function of α_v, that is, as α_v increases, the number of caching locations decreases monotonically. Therefore, we can determine the optimal locations for placing exactly k copies of an object among the enroute caches by tuning the parameter α_v. The relationship between the optimal number of copies k^* and the parameter α_v can be visualized in Figure 4.

The algorithm for placing exactly k copies of an object among the enroute caches is described as follows.

Algorithm 3: UCERWC—Placing Exactly k Copies of An Object
Step 1. Initialization
$\alpha_{\min} = 0$; $\alpha_{\max} = \max_{v \in V}\{h(v)\}$, $k^* = |A_w^*|$.
Step 2. Recursive Procedure
while $k^* \neq k$ do
    $\alpha = (\alpha_{\min} + \alpha_{\max})/2$;
    Call Algorithm 2 by setting $\alpha_v = \alpha$;
    $k^* = |A_w^*|$;
    if $k^* > k$ then
        $\alpha_{\min} = \alpha$
    else
        $\alpha_{\max} = \alpha$

We can see that Algorithm 3 converges to an optimal solution to Equation (15) quickly. The time complexity of Algorithm 3 is given by the following corollary.

COROLLARY 3. *The time complexity of Algorithm* 3 *is* $O(n^2 \log(fn))$*, where* n *is the total number of nodes in the network and* $f = \max_{v \in V}\{f(v)\}$.

PROOF. Since $\alpha_{\max} = \max_{v \in V}\{h(v)\} \leq fn$, from Theorem 2, we can easily know that the time complexity of Algorithm 3 is $O(n^2 \log(fn))$. □

5.3 Constraint III: Placing At Most k Copies of An Object

Based on Equation (4), the constrained coordinated enroute Web caching problem of placing at most k copies of an object among the enroute caches for tree T_w is defined as follows:

$$\begin{cases} \max\limits_{B_w} G(T_w, B_w) = \max\limits_{B_w} \left\{ \sum\limits_{v \in B_w} [(f(v) - f'(v))\, m(v) - l(v)] \right\}. \\ s.t.\ |B_w| \leq k \end{cases} \tag{17}$$

We say that two equations are equivalent if the optimal solution to one equation is also an optimal solution to the other equation. Suppose that $k^* = |A_w^*|$, where A_w^* is an optimal solution to Equation (10) by setting $\alpha_v = 0$, then we have the following theorem on the equivalence between Equation (17) and Equation (5).

THEOREM 7. *if* $k \geq k^*$*, then Equation* (17) *is equivalent to Equation* (5)*.*

PROOF. Suppose that A_w^* is an optimal solution to Equation (5) with respect to tree T_w, and B_w^* is an optimal solution to Equation (17). By the definition of optimal solution, we easily know that A_w^* is an optimal solution to Equation (17) since $k \geq k^*$. Now we prove B_w^* is an optimal solution to Equation (5). We apply reduction to absurdity, and we have $G(T_w, A_w^*) < G(T_w, B_w^*)$. Since A_w^* is a feasible solution to Equation (17), we have $G(T_w, A_w^*) = G(T_w, A_w^*)$ and $G(T_w, B_w^*) = G(T_w, B_w^*)$, so we have $G(T_w, A_w^*) < G(T_w, B_w^*)$. This contradicts the fact that A_w^* is an optimal solution to Equation (5) with respect to tree T_w. Hence, the theorem is proven. □

From Theorem 7, we can see that the problem as described in Equation (5) can be viewed as a special case of the problem being discussed in this section by setting $k = n$.

Based on Algorithm 3 , we can present the following algorithm for placing at most k copies of an object.

Algorithm 4: UCERWC—Placing at Most k Copies of An Object

Step 1. Initialization
gain:=0, placement= ϕ.
Step 2. Recursive Procedure
for $i = 0$ to k do
    Call Algorithm 3 by setting $k = i$;
    gain(i) = $G(T_w, B_w^*)$;
    if gain(i) > gain then
        gain = gain(i)
        placement = B_w^*

We can see that Algorithm 4 converges to an optimal solution to Equation (17) quickly. The time complexity of Algorithm 4 is given by the following corollary.

COROLLARY 4. *The time complexity of Algorithm* 4 *is* $O(kn^2 \log(fn))$*, where* n *is the total number of nodes in the network and* $f = \max_{v \in V}\{f(v)\}$.

PROOF. From Corollary 3, it is obvious that the time complexity of Algorithm 4 is $O(kn^2 \log(fn))$. □

6. COORDINATED ENROUTE CACHING FOR AUTONOMOUS SYSTEMS

In this section, we focus on solving the coordinated enroute Web caching problem for autonomous systems, determining the locations for placing exactly k copies of an object among the enroute caches so that the overall cost gain is maximized. This problem is different from the problem studied in Section 5.2 since the network topology is completely different.

Autonomous systems play an important role in routing objects on the Internet [Bates et al. 1995; Pierre and Steen 2002]. When a client accesses an object, the request is always sent to the local replica server which serves this region. The request is always satisfied by the first node on the path from the client to the local replica sever where a copy of this object is cached. Thus, the routes to all clients served by the replica server form a shortest-path tree rooted at the replica server as assumed in Paxson [1997] and Wolfson and Milo [1991]. Obviously, the routes used by all the clients in the entire network to access an object form a collection of disjoint tress as illustrated in Figure 5.

We denote the whole network by $T_{AS} = (V_{AS}, E_{AS})$ where V_{AS} is the set of the nodes, and E_{AS} is the set of the links. We assume that there are m ASes in the network, each of which is represented by tree $T_i, i = 1, 2, \ldots, m$. Figure 5 shows a simple example of such an autonomous system in which the replica servers have the same contents as the content server. We denote the set of the replica servers and the content server by S_{AS}.

Based on Equation (4), the problem for coordinated enroute Web caching for autonomous systems is defined as follows:

$$\begin{cases} \max\limits_{P_{AS}} \; G(T_{AS}, P_{AS}) = \max\limits_{P_{AS}} \left\{ \sum\limits_{v \in P_{AS}} [(f(v) - f'(v))\, m(v) - l(v)] \right\}, \\ s.t. \;\; |P_{AS}| = k \end{cases} \tag{18}$$

where $P_{AS} \subseteq V_{AS} - S_{AS}$.

From Equation (18), we can see that this problem degenerates to the problem addressed in Section 5.2 when $m = 1$, that is, determining the optimal locations for placing k copies of an object in tree networks. In this section, we also use $\widetilde{G}(T_{AS}, k)$ to denote the overall maximum cost gain for placing k copies of an object in T_{AS} for convenience, that is, $\widetilde{G}(T_{AS}, k) = G(T_{AS}, P^*_{AS})$ where $G(T_{AS}, P^*_{AS})$ is an optimal solution to Equation (18) and $k^* = |P^*_{AS}|$.

Now we apply the following idea to solve this problem, similar to that presented in Jia et al. [2001]. We first divide T_{AS} into two parts: $\cup_{i=1}^{m-1} T_i$ and T_m. Then, we consider the problem of placing k_m copies of an object in the first part and $k - k_m$ copies of an object in the second part where $0 \le k_m \le k$. We divide

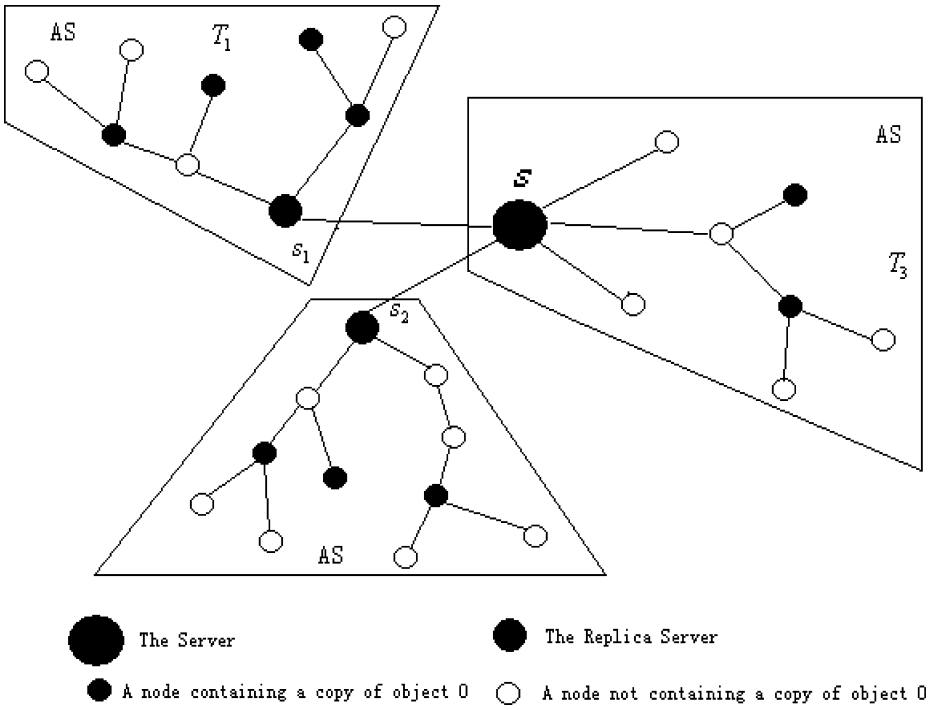

Fig. 5. Enroute Web caching for autonomous systems.

$\cup_{i=1}^{m-1} T_i$ into two parts, $\cup_{i=1}^{m-2} T_i$ and T_{m-1}; thus, we consider the problem of placing k_{m-1} copies of an object in the first part and $k_m - k_{m-1}$ copies of an object in the second part where $0 \le k_{m-1} \le k_m$. We repeat this process until there is only one tree left. Regarding the recursive process, we have the following theorem.

THEOREM 8.

$$\widetilde{G}(T_{+i}, k) = \begin{cases} \widetilde{G}(T_1, k) & \text{if } T_{+i} = T_1 \\ \max\limits_{0 \le k' \le k} \{\widetilde{G}(T_{+(i-1)}, k') + \widetilde{G}(T_i, k - k')\} & \text{if } T_{+i} \ne T_1 \end{cases}, \tag{19}$$

where $T_{+i} = \cup_{j=1}^{i} T_j$.

PROOF. When $T_{+i} = T_1$, it becomes the constrained coordinated enroute Web caching problem of placing k copies of an object among the enroute caches for tree topology, therefore, it is obviously correct.

Now we consider $T_{+i} \ne T_1$. Let $\widetilde{G}'(T_{+i}, k) = \max_{0 \le k' \le k} \{\widetilde{G}(T_{+(i-1)}, k') + \widetilde{G}(T_i, k - k')\}$. We first prove $\widetilde{G}'(T_{+i}, k) \ge \widetilde{G}(T_{+i}, k)$. Suppose that P^* is an optimal solution for placing k copies of an object in T_{+i}, then we have $\widetilde{G}(T_{+i}, k) = G(T_{+i}, P^*)$. Suppose that $P^* \cap T_{+(i-1)} = l$, then we have $P^* \cap T_i = k - l$, therefore we have

$$G(T_{+i}, P^*) = G(T_{+(i-1)}, P^* \cap T_{+(i-1)}) + G(T_i, P^* \cap T_i). \tag{20}$$

It is easy to see that $P^* \cap T_{+(i-1)}$ and $P^* \cap T_i$ are optimal placements for placing l copies and $k - l$ copies in $T_{+(i-1)}$ and T_i, respectively. Otherwise, there should

be a better placement than $P^* \cap T_{+(i-1)}$ or $P^* \cap T_i$ which would contradict that P^* is an optimal placement. Therefore, we have

$$\begin{aligned}\widetilde{G}(T_{+i}, k) &= G(T_{+i}, P^*) \\ &= G\big(T_{+(i-1)}, P^* \cap T_{+(i-1)}\big) + G(T_i, P^* \cap T_i) \\ &= \widetilde{G}\big(T_{+(i-1)}, l\big) + \widetilde{G}(T_i, k-l) \\ &\le \max_{0 \le l' \le k} \big\{\widetilde{G}\big(T_{+(i-1)}, l'\big) + \widetilde{G}(T_i, k-l')\big\} \\ &= \widetilde{G}'(T_{+i}, k).\end{aligned}$$

Now we prove $\widetilde{G}'(T_{+i}, k) \le \widetilde{G}(T_{+i}, k)$. Suppose that $\widetilde{G}(T_{+(i-1)}, l) + \widetilde{G}(T_i, k - l)\} = \max_{0 \le l' \le k}\{\widetilde{G}(T_{+(i-1)}, l') + \widetilde{G}(T_i, k - l')\}$. Let $P^*_{+(i-1)}$ be an optimal placement for placing l copies of an object in $T_{+(i-1)}$ and P_i^* an optimal placement for placing $k - l$ copies of an object in T_i, then for any l, we have

$$\begin{aligned}\widetilde{G}(T_{+i}, k) &\ge G\big(T_{+i}, P^*_{+(i-1)} \cup P_i^*\big) \\ &= G\big(T_{+(i-1)}, P^*_{+(i-1)}\big) + G(T_i, \cup P_i^*) \\ &= \widetilde{G}\big(T_{+(i-1)}, l\big) + \widetilde{G}(T_i, k-l),\end{aligned}$$

therefore, we have $\widetilde{G}(T_{+i}, k) \ge \max_{0 \le l' \le k}\{\widetilde{G}(T_{+(i-1)}, l') + \widetilde{G}(T_i, k - l')\} = \widetilde{G}'(T_{+i}, k)$. Hence, the theorem is proven. □

Our algorithm for coordinated enroute Web caching for autonomous systems is described as follows.

Algorithm 5: Algorithm for Coordinated Enroute Web Caching for Autonomous Systems

Main Procedure
for $i = 1$ to m do (Initialization)
  for $j = 0$ to $k - m + i$ do
    $\widetilde{G}(T_{+i}, j) = -1$; (The cost gain for placing j copies in T_{+i})
    $P(T_i, j) = \phi$ (The optimal solution for placing j copies in T_i)
Call *Placement*(T_{+m}, k); (Calling procedure *Placement* recursively)
Procedure Placement(T_{+i}, l)
if $\widetilde{G}(T_{+i}, l) \ge 0$ then
  Return $\widetilde{G}(T_{+i}, l)$; ($\widetilde{G}(T_{+i}, l)$ computed)
if $i = 1$ then
  Return $\widetilde{G}(T_1, l)$; (Call Algorithm 3 since $\widetilde{G}(T_1, l) = G(T_1, A_1^*)$)
$TG = 0$;
for $l' = (i-1)$ to $l - 1$ do (Finding the optimal number of copies to be placed in T_i)
  TG = *Placement*$(T_{+(i-1)}, l') + \widetilde{G}(T_i, l - l')$; (According to Theorem 8)
    = *Placement*$(T_{+(i-1)}, l') + G(T_i, A_i^*)$;
  if $\widetilde{G}(T_{+i}, l) > TG$ then
    $\widetilde{G}(T_{+i}, l) = TG$
    $P(T_i, l - l') = A_i^*$

The time complexity of Algorithm 5 is given by the following corollary.

COROLLARY 5. *The time complexity of Algorithm* 5 *is* $O(kn^2 \log(fn))$*, where* n *is the total number of nodes in the network and* $f = \max_{v \in V}\{f(v)\}$.

PROOF. According to Corollary 3, the time for running *Placement*(T_{+i}, l) is $O(n_i^2 \log(f n_i))$, where n_i is the number of nodes of tree T_i and $1 \le i \le m$. Since

Algorithm 5 calls *Placement*(T_{+i}, l) to compute all the elements in $\widetilde{G}(T_{+i}, l)$, its time complexity is

$$O\left(\sum_{i=1}^{m}\sum_{l=1}^{k}\left(n_i^2\log(fn_i)\right)\right) = O\left(\sum_{i=1}^{m}\left(kn_i^2\log(fn_i)\right)\right)$$
$$\leq O\left(\log(fn)\sum_{i=1}^{m}\left(n_i^2\right)\right) \leq O(kn^2\log(fn)). \quad (21)$$

Hence, the corollary is proven. □

7. SIMULATION AND RESULTS

In this section, we introduce the parameter estimation in Section 7.1, describe the simulation model in Section 7.2, and discuss the experimental results in Section 7.3.

7.1 Parameter Estimation

In the actual implementation, the access frequency and the miss penalty of an object with respect to a node are not usually constant. We have to estimate them accurately so that the characteristics of data access can be well captured.

We apply the methods described in Tang and Chanson [2002] for estimating parameters used in our model. The access frequency $f(v)$ is estimated by recent request data. We apply a sliding window technique to estimate the access frequency to make our model less sensitive to transient workload [Shim et al. 1999]. Specifically, $f(v)$ is calculated by $K/(t - t_K)$, where K is the number of accesses recorded, t is the current time, and t_K is the Kth most recently referenced time (the time of the oldest reference in the sliding window). It is shown by in Shim et al. [1999] that K can be as small as 2 or 3 to achieve the best performance. In our simulation k is set to 2. If knowledge of access frequency is imprecise, another method can be applied to estimate the average access frequency for object O observed at node v based on the size of this object. Specifically, $f(v)$ is calculated by p/s^b, where p and b are constants for object O at node v, and s is the size of object O [Shim et al. 1999]. This is shown by Cunha et al. [1995] and Glassman [1994] based on studies that show that Web clients exhibit a strong preference for accessing small objects. The miss penalty is updated by the response messages. Specifically, a variable with an initial value of zero is attached to each object. At each intermediate node along the way, the variable is increased by the cost of the last link the object has just traversed. The value is then used to update the miss penalty of the object maintained by the associated cache. If the object is inserted into the cache, the node resets the value to zero before forwarding the object downstream. In this way, the updated cost loss is disseminated to all the caches along the way.

7.2 Simulation Model

We have performed extensive simulation experiments for comparing the results of our model with those of the existing models. The network in our simulation

Table I. Parameters of Our Experiments

Parameter	Value
Total Number of Nodes	300
Number of WAN Nodes	150
Number of MAN Nodes	150
Delay of WAN Links	0.45 second
Delay of MAN Links	0.06 second
Number of Objects	1000
Average Object Size	$30KB$
Average Request Rate Per Node	$U(1, 9)$ requests per second

consists of numerous nodes. Here, we assume that there is only one server. As far as we know, it is difficult to find true trace data in the open literature to simulate our model. Similar to the simulation model proposed in Tang and Chanson [2002], we generated the simulation model from empirical results presented in Barford and Crovella [1998], Breslau et al. [1999], and Calvert et al. [1997].

The network topology is randomly generated by the Tier program [Calvert et al. 1997]. We have conducted experiments for many topologies with different parameters and found that the performance of our model was insensitive to topology changes. Here we list only the experimental results for one topology because of space limitations. Table I shows the parameters and their values used in our experiments where $U(x, y)$ denotes the uniform distribution between x and y.

The WAN (Wide Area Network) is viewed as the backbone network to which no servers or clients are attached. Each MAN (Metropolitan Area Network) node is assumed to connect to the content server. Each MAN and WAN node is associated with an enroute cache. Similar to the studies in Breslau et al. [1999], Cao and Irani [1997], Jin and Bestavros [2001], and Shim et al. [1999], we describe cache size as the total relative size of all objects available in the content server. We assume for our experiments that the object sizes follow the distribution as described in Barford and Crovella [1998] and that the average object size is $30KB$. In our experiments, the client at each MAN node randomly generates the requests and the average request rate of each node follows the distribution of $U(1, 9)$. The cost for each link is calculated by the access latency. For simplicity, the delay caused by sending the request and the relevant response for that request is proportional to the size of the requested object. We consider the average object sizes for calculating all delays, including the propagation delay, the transmission delay, and the searching delay. The cost function is taken to be the delay of the link which means that the cost in our model is interpreted as the access latency in our simulation.

7.3 Experimental Results

In our experiments, we compare the performance results of different models across a wide range of cache sizes, from 0.04 percent to 12 percent, and the performance metrics include the average access latency, the response ratio of a request (the ratio of its access latency to the size of the target object), the object

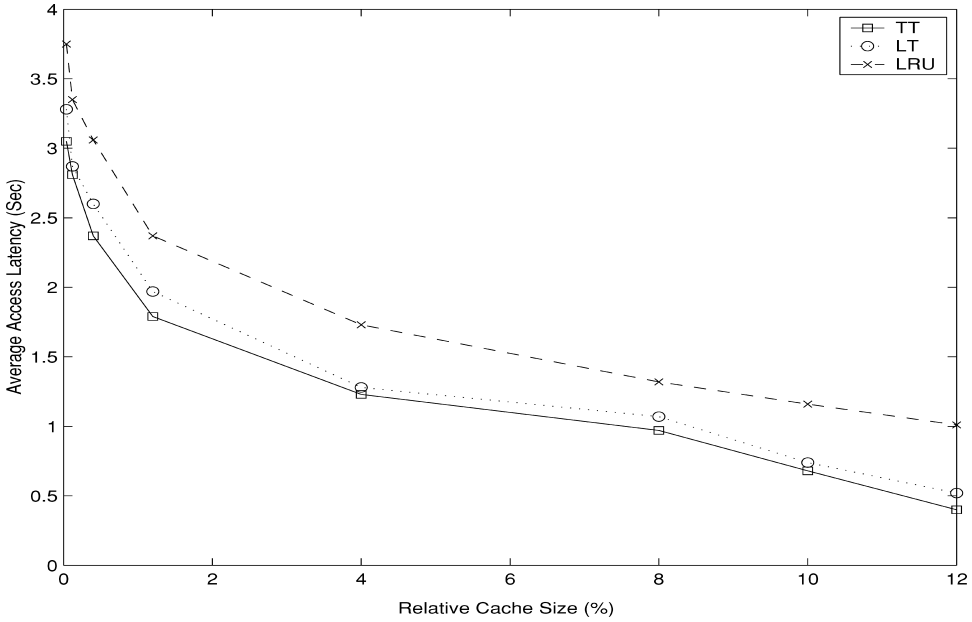

Fig. 6. Experiment for average access latency.

hit ratio (the ratio of the number of requests satisfied by the caches as a whole to the total number of requests.), the byte hit ratio (the ratio of the bytes of requests satisfied by the caches as a whole to the total bytes of requests), and the average server load (the average number of bytes served by the server per second). In our experiments, we denote the results for the *LRU* model [Williams et al. 1996] by *LRU*, the results for coordinated enroute Web caching for linear topology [Tang and Chanson 2002] by *LT*, and the results for unconstrained coordinated enroute Web caching for tree topology by *TT*.

Figure 6 shows the results of the average latency as a function of the relative cache size at each node. We also describe the results of the response ratio of a request as a function of the relative cache size at each node in Figure 7. As we knew, the lower the average access latency or the average response ratio, the better the performance. We can easily see that the performance results for the three models improve as the relative cache size increases. We can also see that *TT* can improve both the average access latency and the average response ratio compared to *LRU* and *LT* since our model determines the optimal locations in the whole tree, while *LRU* places the copies of an object at each enroute cache and *LT* optimally places the copies of an object on the path from the client to the server.

Figure 8 and Figure 9 show the results of the object hit ratio and the byte hit ratio as functions of the relative cache size for different models, respectively. By computing the optimal locations for the tree topology, we can see that the results for our model can greatly outperform those of the other two models, especially for smaller cache sizes. The object hit ratio and the byte hit ratio steadily improve as the relative cache size increases which conforms to the fact

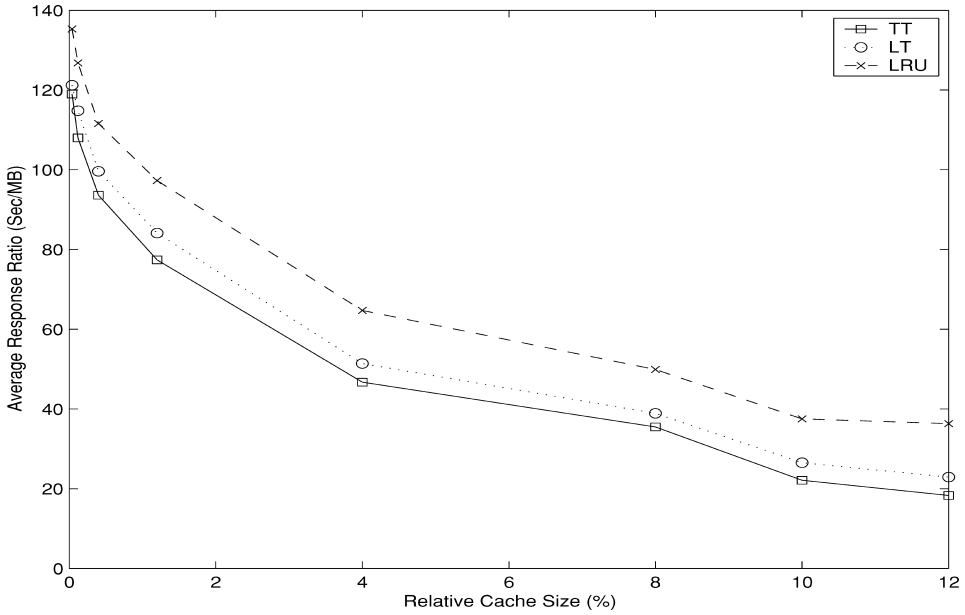

Fig. 7. Experiment for average response ratio.

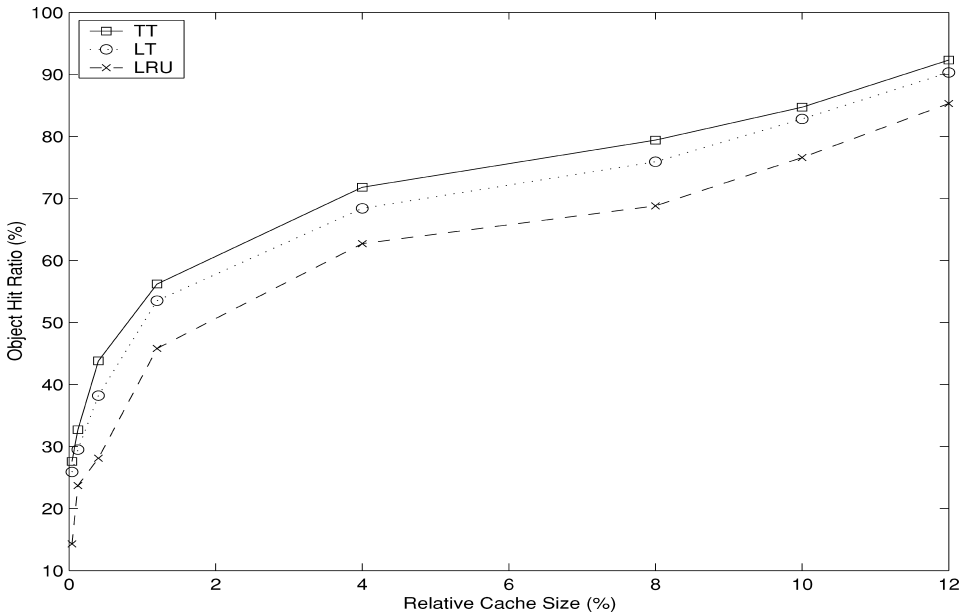

Fig. 8. Experiment for object hit ratio.

that more requests will be satisfied by the caches, as the cache size becomes larger.

Figure 10 shows the results of the server load as a function of the relative cache size. It can be seen that the average server load for our model is lower

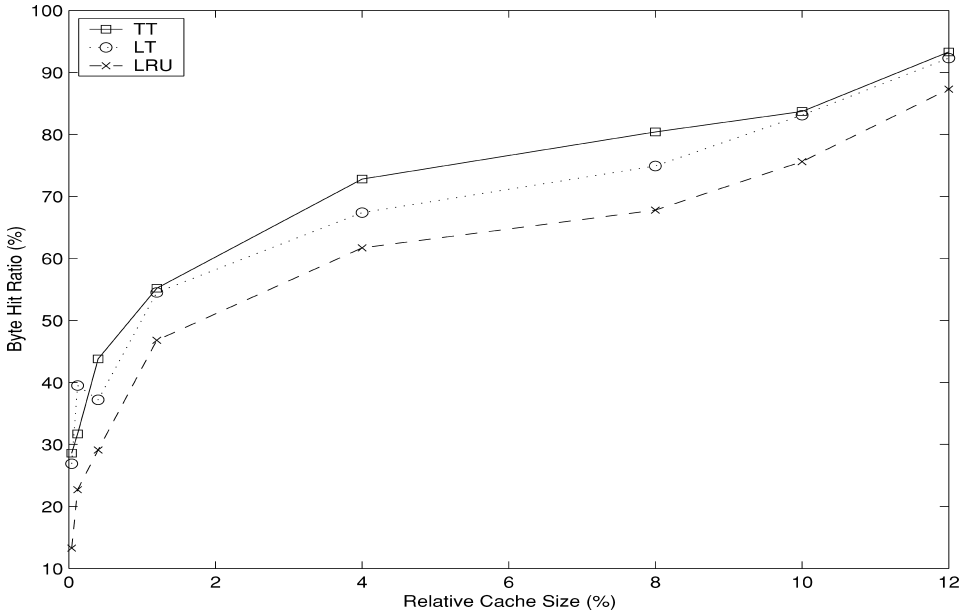

Fig. 9. Experiment for byte hit ratio.

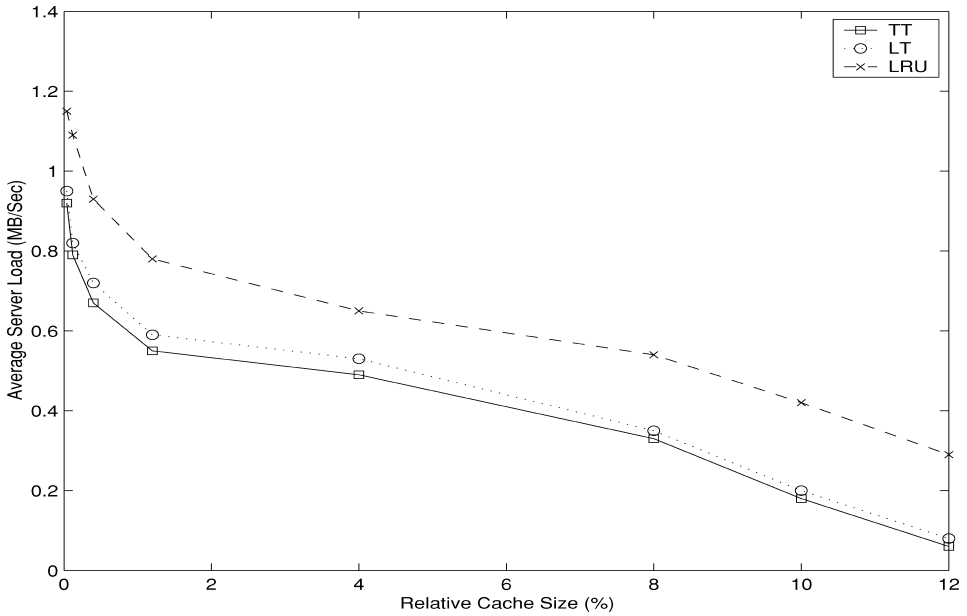

Fig. 10. Experiment for average server load.

than that of the other models. We can also see that the average server load decreases as the relatice cache size increases.

Figure 11 shows the results of the average response ratio and the object hit ratio as functions of the average number of copies of the objects placed among

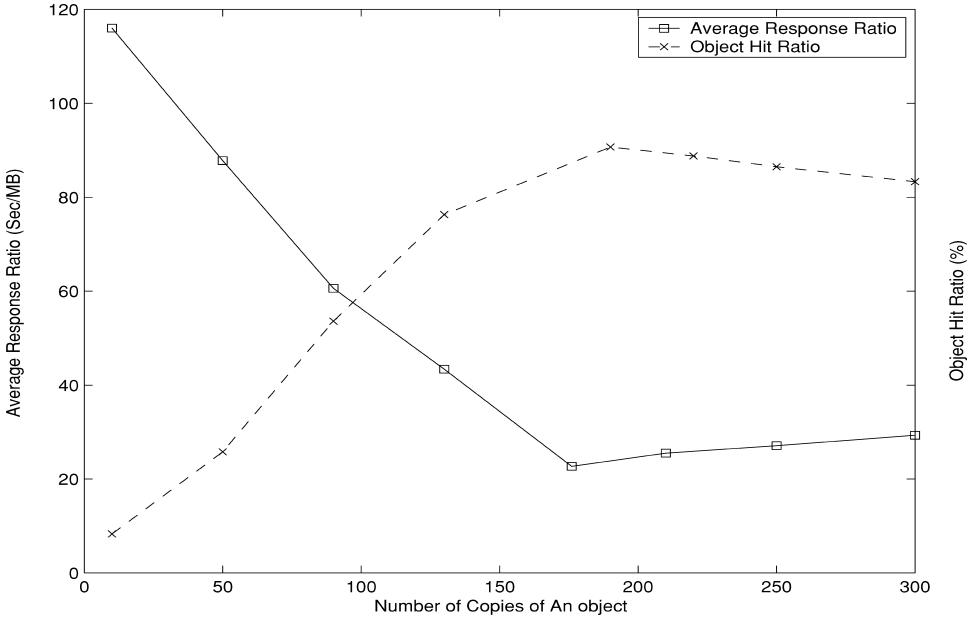

Fig. 11. Experiment for different number of copies.

Table II. Different Cases Addressed In This Article

Case	Network Topology	Description	Time Complexity
UCERWC	Tree Network	No constraints	$O(n^2)$
CCERWC	Tree Network	Non-negative cost gain per node	$O(n^2)$
	Tree Network	Placing exactly k copies	$O(n^2 \log(fn))$
	Tree Network	Placing at most k copies	$O(kn^2 \log(fn))$
	Autonomous System	Placing exactly k copies	$O(kn^2 \log(fn))$

the enroute cache, respectively. We can see that the average response ratio decreases as the number of copies of an object placed among the enroute caches increases. When the number arrives at about 175, the average response ratio begins to decrease slowly. This is true because the optimal number of copies of an object to be placed in such a network topology is approximately 175. We also can see that the object hit ratio decreases with the number of copies of an object placed among the enroute caches increasing. When the number reaches about 190, the object hit ratio starts to decrease. This is true because the optimal number of copies of an object to be placed in such a network topology is approximately 190.

8. CONCLUSION

The performance of enroute Web caching depends mainly on the locations of caches and on the management of cache contents. In this article, we studied the coordinated enroute Web caching problem on tree networks and autonomous systems for both unconstrained and constrained cases (see Table II).

We presented a mathematical model that integrates both cost loss and cost saving for storing an object at a node. According to the information stored at each node, we can optimally decide where copies of the requested object should be placed and what should be removed, if necessary, to make room for them. We also proposed low-cost dynamic programming-based algorithms for the different cases and theoretically showed that the algorithms were either optimal or convergent to optimal solutions. We have performed extensive experiments to compare the proposed methods with the existing algorithms. The simulation results show that our methods significantly outperform the existing algorithms which consider either coordinated enroute Web caching for linear topology or object placement (replacement) at individual nodes only. Our methods have made significant contributions to enroute Web caching since the locations for placing copies of an object among the enroute caches can be optimally obtained for different networks under different constraints.

A challenging task for our future research is to solve the coordinated enroute object caching problem in networks of arbitrary topology. The techniques of applying dynamic programming shown in this article may serve as useful tools for deriving such solutions in the general case.

In this article we treated each object as a separate problem and assumed that costs for different objects are independent of each other, as done by others in the literature [Tang and Chanson 2002; Xu et al. 2002]. Taking into consideration interobject dependency to cache individual objects also remains an important and challenging problem for our future research.

REFERENCES

AWERBUCH, B., BARTAL, Y., AND FIAT, A. 1998. Distributed paging for general networks. *J. Algorithms 28*, 67–104.

BARFORD, P. AND CROVELLA, M. 1998. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of ACM SIGMETRICS'98*. 151–160.

BATES, T., GERICH E., JONCHERAY, L., JOUANIGOT, J. M., KARRENBERG, D., TERPSTRA, M. AND YU, J. 1995. Representation of IP routing policies in a routing registry. *RFC 1786*. IETF.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. 1999. Web caching and zip-like distributions: evidence and implications. In *Proceedings of IEEE INFOCOM'99*. 126–134.

CALVERT, K. L., DOAR, M. B., AND ZEGURA, E. W. 1997. Modelling internet topology. *IEEE Comm. Magazine 35*, 6, 160–163.

CAO, P. AND IRANI, S. 1997. Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems (USITS)*. 193–206.

CHANKHUNTHOD, A., DANZIG, P., NEERDAELS, C., SCHWARTZ, M., AND WORRELL, K. 1996. A hierarchical Internet object cache. In *Proceedings of the USENIX Technical Conference*. 22–26.

CUNHA, C., BESTAVROS, A., AND CROVELLA, M. 1995. Characteriatics of WWWW Client-Based Traces. Tech. rep. TR-95-010, Boston University, Boston, MA.

DAHLIN, M. D., WANG, R. Y., ANDERSON, T. E., AND PATTERSON, D. A. 1994. Cooperative caching: using remote client memory to improve file system performance. In *Proceedings of the 1st Symposium on Operating Systems Design and Implementations*. 267–280.

DAVISON, B. D. 2001. A Web caching primer. *IEEE Internet Comput. 5*, 4, 38–45.

GLASSMAN, S. 1994. A caching relay for the World Wide Web. *Comput. Netw. ISDN Syst. 27*, 2, 165–173.

JIA, X., LI, D., HU, X., AND DU, D. 2001. Optimal placement of Web proxies for replicated Web servers in the Internet. *Comput. J. 44*, 5, 329–339.

JIANG, A. AND BRUCK, J. 2003. Optimal content placement for enroute Web caching. In *Proceedings of the 2nd International Symposium on Network Computing and Applications (NCA'03)*. 9–16.

JIN, S. AND BESTAVROS, A. 2001. Greeddual* Web caching algorithm exploiting the two sources of temporal locality in Web request streams. *Comput. Comm. 4*, 2, 174–183.

KORUPOLU, M. R. AND DAHLIN, M. 2002. Coordinated placement and replacement for large-scale distributed caches. *IEEE Tran. Knowl. Data Eng. 14*, 6, 1317–1329.

KORUPOLU, M. R., PLAXTON, C. G., AND RAJARAMAN, R. 1999. Placement algorithms for hierarchical coorperative caching. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*. 586–595.

KRISHNAN, P., RAZ, D., AND SHAVITT, Y. 2000. The cache location problem. *IEEE/ACM Trans. Netw. 8*, 5, 568–582.

LEFF, A., WOLF, J. L., AND YU, P. S. 1993. Replication algorithms in a remote caching architecture. *IEEE Trans. Paral. Distrib. Syst. 4*, 11, 1185–1204.

LI, B., GOLIN, M. J., ITALIANO, G. F., DENG, X., AND SOHRABY, K. 1999. On the optimal placement of Web proxies in the Internet. In *Proceedings of IEEE INFOCOM'99*. 1282–1290.

LI, K. AND SHEN, H. 2003. Constrained coordinated enroute Web caching in tree networks. In *Proceedings of the 3rd International Conference on Hybrid Intelligent Systems (HIS'03)*. 1054–1063.

LI, K. AND SHEN, H. 2003. An optimal method for coordinated enroute Web object caching. In *Proceedings of the 5th International Symposium on High Performance Computing (ISHPC-V)*. 368–375.

PAXSON, V. 1997. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. Netw. 5*, 5, 601–615.

PIERRE, G. AND STEEN, M. 2002. Dynamically selecting optimal distribution strategies for Web documents. *IEEE Trans. Comput. 51*, 6, 637–651.

RABINOVICH, M. AND SPATSCHECK, O. 2002. *Web Caching and Replication*. Addison-Wesley.

RABINOVICH, P. AND WANG, H. 2001. Dhttp: an efficient and cache-friendly transfer protocol for Web traffic. In *Proceedings of IEEE INFOCOM'01*. 1597–1606.

RODRIGUEZ, P. AND SIBAL, S. 2000. Spread: Scalable platform for reliable and efficient automated distribution. *Comput. Netw. 33*, 33–49.

RODRIGUEZ, P., SIBAL, S., AND SPATSCHECK, O. 2000. Tpot: translucent proxying of TCP. In *Proceedings of the 5th International Web Caching and Content Delivery Workshop (WCW'00)*.

RODRIGUEZ, P., SPANNER, C., AND BIERSACK, E. W. 2001. Analysis of Web caching architectures: Hierarchical and distributed caching. *IEEE/ACM Trans. Netw. 9*, 4, 404–418.

SCHEUERMANN, P., SHIM, J., AND VINGRALEK, R. 1997. A case for delay-conscious caching of Web documents. *Comput. Netw. ISDN Syst*. 2, 8–13, 997–1005.

SHIM, J., SCHEUERMANN, P., AND VINGRALEK, R. 1999. Proxy cache algorithms: design, implementation, and performance. *IEEE Trans. Knowl. Data Eng. 11*, 4, 549–562.

TANG, X. AND CHANSON, S. T. 2002. Coordinated enroute Web caching. *IEEE Trans. Comput. 51*, 6, 595–607.

TENNENHOUSE, D. L., SMITH, J. M., SINCOSKIE, W. D., WETHERALL, D. J., AND MINDEN, G. J. 1997. A survey of active network research. *IEEE Comm. Magazine 35*, 1, 80–86.

TEWARI, X., DAHLIN, M., VIN, H. M., AND KAY, J. S. 1999. Design considerations for distributed caching on the Internet. In *Proceedings of the 19th International Confernece on Distributed Computing Systems (ICDCS'99)*. 273–284.

WANG, J. 1999. A survey of Web caching schemes for the internet. *ACM SIGCOMM Comput. Comm. Rev. 29*, 5, 36–46.

WILLIAMS, S., ABRAMS, M., STANDBRIDGE, C. R., ABDULLA, G., AND FOX, E. A. 1996. Removal policies in network caches for World Wide Web documents. In *Proceedings of ACM SIGCOMM'96*. 293–305.

WOLFSON, O. AND MILO, A. 1991. The multicast policy and its relationship to replicated data placement. *ACM Trans. Datab. Syst. 16*, 1, 181–205.

XU, J., LI, B., AND LI, D. L. 2002. Placement problems for transparent data replication proxy services. *IEEE J. Select. Areas Comm. 20*, 7, 1383–1398.

YU, P. S. AND MACNAIR, E. A. 1998. Performance study of a collaborative method for hierarchical caching in proxy servers. *Computer Netw. ISDN Syst. 30*, 215–224.

Received September 2003; revised February 2004; accepted June 2004