Placement Solutions for Multiple Versions of A Multimedia Object *

Keqiu Li, Hong Shen
Graduate School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa, 923-1292, Japan

Francis Y. L. Chin
Department of Computer Science and Information Systems
University of Hong Kong
Pokfulam Road, Hong Kong

Abstract

Transcoding is an important technology which adapts the same multimedia object to diverse mobile appliances; thus, users' requests for a specified version of a multimedia object could be served by a more detailed version cached according to transcoding. Therefore, it is of particularly theoretical and practical necessity to determine the proper versions to be cached at a node such that the specified objective is achieved. In this paper, we address the problem of multimedia object placement. The performance objective is to minimize the total access cost by considering both transmission cost and transcoding cost. We present an optimal dynamic programming-based solution for this problem. The performance of the proposed solutions is evaluated with a set of carefully designed simulation experiments for various performance metrics over a wide range of system parameters. The simulation results show that our solution consistently and significantly outperforms comparison solutions in terms of all the performance metrics considered.

Key words: Web caching, multimedia, object placement, transcoding, transparent data access, optimization.

1 Introduction

The World Wide Web has become the most successful application on the Internet since it provides a simple way to access a wide range of information and services. However, due to the dramatic growth in demand, considerable access latency is often experienced in retrieving web objects from the Internet, and popular web sites are suffering from overload. An efficient way to overcome such deficiencies is web caching, by which multiple copies of the same object are stored in geographically dispersed caches. An overview of web caching can be found in [3, 7]. As many mobile appliances are divergent in size, weight, I/O capabilities, network connectivity, and computing power, differentiated devices should be employed to satisfy their diverse requirements.

*This work was supported by Japan Society for the Promotion of Science (JSPS) under its General Research Scheme **B** Grant No. 14380139).

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

In addition, different presentation preferences from users make this problem more serious. Transcoding, used to transform a multimedia object from one form to another, frequently through trading off object fidelity for size, is a technology that can meet these needs [1, 4–6].

Due to the limited cache capacity, it is impossible to store all the versions of a multimedia object at a node. This, it is significant to determine what versions should be cached if the number versions is given so that the total access cost is minimized. In this paper, we address the problem of multimedia object placement, i.e., determining exactly what versions should be cached at a node such that the total access cost is minimized. The main contributions of this paper are summarized as follows.

- We present a model for the problem of multimedia object placement, formulated as an optimization problem. In our model, multimedia object placement decisions are made based on both transcoding and transmission cost.
- We propose an optimal dynamic programming-based solution to compute the optimal versions to be cached at a node.
- We give an extensive and detailed analysis on the proposed solution and show that our solution is optimal and low-cost.

The rest of this paper is organized as follows. Section 2 introduces web object transcoding. Section 3 presents an optimal solution for the problem of multimedia object placement. Section 4 summarizes our work and concludes the paper.

2 Web Object Transcoding

Transcoding is used to transform a multimedia object from one form to another, frequently trading off object fidelity for size for the prevailing operating environments. The relationship among different versions of a multimedia object can be expressed by a weighted transcoding graph [2]. An example of such a graph is shown in Fig. 1. In Fig. 1, we can

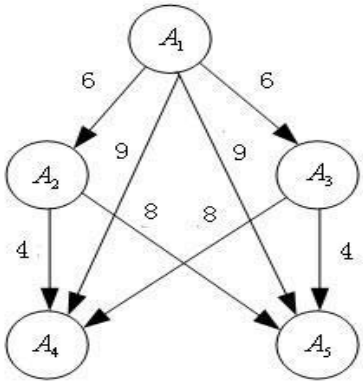

Figure 1: A Weighted Transcoding Graph

see that the original version A_1 can be transcoded to each of the less detailed versions A_2, A_3, A_4, and A_5. It should be noted that not every A_i can be transcoded to A_j since it is possible that A_i does not contain enough content information for the transcoding from A_i to A_j. In the example, transcoding can not be executed between A_4 and A_5 due to insufficient content information. The transcoding cost of a multimedia object from A_i to A_j is denoted by $t(A_i, A_j)$. Obviously, $t(A_i, A_i) = 0$. The number beside each edge in Fig. 1 is the transcoding cost from one version to another. For example, $t(A_1, A_2) = 6$, and $t(A_3, A_4) = 8$. If a version cannot be transcoded from another version, we consider the transcoding cost as infinity. For instance, $t(A_2, A_1) = \infty$, and $t(A_4, A_5) = \infty$. If version A_j can be transcoded from version A_i through version

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

A_k with $i < k < j$, then $t(A_i, A_j) \leq t(A_i, A_k) + t(A_k, A_j)$ (triangularity property) because version A_k is always an option if the transcoding cost $t(A_i, A_j)$ is too large. For example, $t(A_1, A_4) < \min\{t(A_1, A_2)+t(A_2, A_4), t(A_1, A_3)+t(A_3, A_4)\}$. $\Phi(A_i)$ is the set of all the versions that can be transcoded from A_i, including A_i. For example, $\Phi(A_1) = \{A_1, A_2, A_3, A_4, A_5\}$, $\Phi(A_2) = \{A_2, A_4, A_5\}$, and $\Phi(A_4) = \{A_4\}$.

3 Dynamic Programming-Based Solution

For a multimedia object O, we assume that it has m versions: $A_1, A_2, \cdots, A_m$. For each version of object O, we associate the link from the client to the server a nonnegative cost L, which is defined as the cost of sending a request for version A_k and the relevant response over this link. Let f_j be access frequency of version A_j from the node.

Before we formulate the problem, we can make the following assumptions.

- *Assumption* 1. The transcoding graph is a linear array and the transcoding cost between any two adjacent versions is constant, i.e., $t(A_i, A_j) = \sum_{k=i}^{j-1} t(A_k, A_{k+1}) = (j-i)^+ T$, where $x^+ = x$ if $x \geq 0$ else $x^+ = \infty$.
- *Assumption* 2. $(\delta - 1)T \leq L$, and $\delta T > L$ for some positive integer δ.

If there does not exist δ such that *Assumption* 3 can be satisfied, i.e., $L \gg T$ or $T \gg L$. Obviously, these are two special cases. If $L \gg T$, then version A_{d_1} should be cached so that no transmission cost is not necessary to incur, where $d_1 = \min\{j | f_j > 0, 1 \leq j \leq m\}$. If $T \gg L$, then version A_{d_2} should be cached, where $d_2 = \max\{j | f_j > 0, 1 \leq j \leq m\}$.

First, we begin by computing the access cost of caching only one version A_k at node v with $1 \leq k \leq m$. Intuitively, all the requests for version A_i with $i < k$ will be handled by server v_0, while some of the requests for A_i with $i \geq k$, depending on the transcoding cost and the transmission cost, will be taken care of by transcoding from version A_k. Thus, the total access cost of caching only version A_k at node v is computed as follows:

$$C(A_k) = \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{m} f_{1,i} \min\{(i-k)T, L\} \quad (1)$$

Since version A_k is cached at node v, we can see (with Assumption 2) that δ is such a parameter that the request for version A_i will be served by the local node if $0 < i - k < \delta$, and the request for version A_i will be served by the server if $i - k \geq \delta$.

Based on Equation (1), $C(A_k)$ can be further defined as follows:

$$C(A_k) = \begin{cases} \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{k+\delta-1} f_{1,i}(i-k)T \\ \qquad + \sum_{i=k+\delta}^{m} f_{1,i} L \quad \text{if } k + \delta \leq m \\ \sum_{i=1}^{k-1} f_{1,i} L + \sum_{i=k}^{m} f_{1,i}(i-k)T \quad \text{if } k + \delta > m \end{cases} \quad (2)$$

It is easy to see that $C(A_1)$ can be calculated in

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

$O(m)$ time. As

$$C(A_{k+1}) = \begin{cases} C(A_k) + f_{1,k}L - \sum_{i=k+1}^{k+\delta-1} f_{1,i}T \\ \quad + f_{1,k+\delta}((\delta-1)T - L) \\ \quad \text{if } k+\delta \le m \\ C(A_k) + f_{1,k}L - \sum_{i=k+1}^{m} f_{1,i}T \\ \quad \text{if } k+\delta > m \end{cases} = C(A_k) + E(k)$$

where

$$E(k) = \begin{cases} f_{1,k}L - \sum_{i=k+1}^{k+\delta-1} f_{1,i}T + f_{1,k+\delta}((\delta-1)T - L) \\ \quad \text{if } k+\delta \le m \\ f_{1,k}L - \sum_{i=k+1}^{m} f_{1,i}T \quad \text{if } k+\delta > m \end{cases}$$

and

$$E(k+1) = \begin{cases} E(k) - f_{1,k}L + f_{1,k+1}(L+T) \\ -f_{1,k+\delta}(\delta T - L) + f_{1,k+\delta+1}(\delta-1)T \\ -f_{1,k+\delta+1}L \quad \text{if } k+\delta < m \\ E(k) - f_{1,k}L + f_{1,k+1}(L+T) \\ -f_{1,k+\delta}(\delta T - L) \quad \text{if } k+\delta = m \\ E(k) - f_{1,k}L + f_{1,k+1}(L+T) \\ \quad \text{if } k+\delta > m \end{cases}$$

Thus, each $C(A_2), C(A_3), \cdots, C(A_m)$ can be done in constant time; Therefore, the MOP problem can be solved in $O(m)$ time. Regarding to the time complexity of solving the MOP problem, we have the following theorem.

Theorem 1 *Based on the cost function as given in Equation (1), the MOP problem for $\{A_1, A_2, \cdots, A_m\}$ by caching only one version (i.e., $n = 1$) can be solved in $O(m)$ time.*

Proof Since the cost function as given in Equation (2) is equivalent to the cost function as given in Equation (1) and the MOP problem based on the cost function as given in Equation (2) can be solved in $O(m)$ time, the MOP problem based on the cost function as given in Equation (1) can also be solved in $O(m)$ time. Hence, the theorem is proven. □

The second step is to extend the above solution to compute the optimal solution for caching two versions, A_{k_1} and A_{k_2}, at the same time at node v.

Suppose that A_{k_1} and A_{k_2} are the two optimal versions to be cached. The key observation here is that A_{k_1} is also an optimal solution for the problem with $\{A_1, A_2, \cdots, A_{k_2-1}\}$ if $k_1 < k_2$, because the requests for $\{A_{k_2}, A_{k_2+1}, \cdots, A_m\}$ can only be served by A_{k_2}. Regarding to this observation, we have the following lemma.

Lemma 1 *Assume that A_{b_p} and A_{b_q} are the optimal solutions for the problem of caching only one version from the set of $\{A_1, A_2, \cdots, A_{p-1}\}$ and $\{A_1, A_2, \cdots, A_{q-1}\}$ respectively, then we have $b_p \le b_q$ if $p < q$.*

Proof Without loss of generality, it is sufficient for us to prove that $b_p \le b_{p+1}$ where $1 \le b_p \le p-1$ and $1 \le b_{p+1} \le p$. The proof is by contradiction. Assume that we have $b_p > b_{p+1}$. As A_{b_p} is the optimal version to be cached, we have $C_{1,p}(A_{b_p}) < C_{1,p}(A_{b_{p+1}})$. Let $C_{1,p}(A_i)$ denote the access cost of caching A_i for the MOP problem with $\{A_1, A_2, \cdots, A_{p-1}\}$. From the definition of the access cost function $C_{1,p}$ as given in Equation (1), adding A_p to the set $\{A_1, A_2, \cdots, A_{p-1}\}$ will increase both $C_{1,p}(A_{b_p})$ and $C_{1,p}(A_{b_{p+1}})$ by $f_{1,p}\min\{(p-b_p)T, L\}$ and $f_{1,p}\min\{(p-b_{p+1})T, L\}$ respectively. The increase to $C_{1,p}(A_{b_{p+1}})$ is no less than that to $C_{1,p}(A_{b_p})$ because $b_p > b_{p+1}$. So we have

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

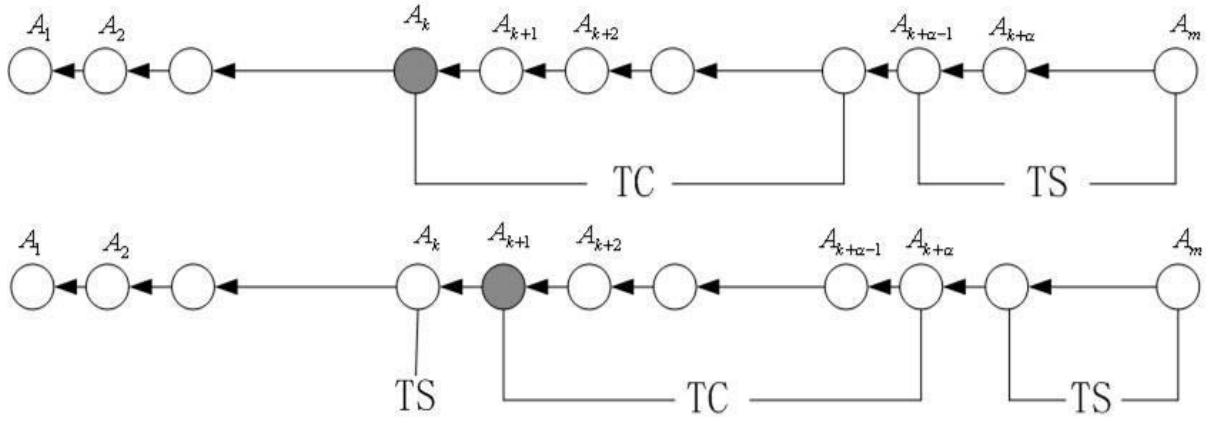

Figure 2: Relationship between $C(A_k)$ and $C(A_{k+1})$

$C_{1,p+1}(A_{b_p}) < C_{1,p+1}(A_{b_{p+1}})$, which contradicts the fact that $C_{1,p+1}(A_{b_{p+1}})$ is the minimum access cost of caching $A_{b_{p+1}}$ for the problem with $\{A_1, A_2, \cdots, A_{p-1}, A_p\}$. Hence the lemma is proven. □

Based on Lemma 1, we can see that the feasible range of the optimal solution for the problem with $\{A_1, A_2, \cdots, A_q\}$ can be reduced if the optimal version for the problem with $\{A_1, A_2, \cdots, A_p\}$ has been obtained. So is the other case when the optimal solution for the problem with $\{A_1, A_2, \cdots, A_q\}$ is known, the feasible range of the optimal solution for the problem with $\{A_1, A_2, \cdots, A_p\}$ is also reduced. Therefore, we can find A_{b_p} and compute $C_{1,p}(A_p)$ by divide and conquer.

Let $D^{(k)}_{p,q}$ denote the minimum access cost of caching k versions for the MOP problem with $q-p$ versions, i.e., $A_p, A_{p+1}, \cdots, A_{q-1}$, where $1 \le p < q \le m$. Thus, $D^{(1)}_{1,p} = C_{1,p}(A_{b_p})$ and $D^{(1)}_{1,m+1} = \min_{1 \le k \le m} \{C_{1,m+1}(A_k)\}$. Based on Lemma 1, we have the following theorem on the time complexity of computing $D^{(1)}_{1,p}$ for $1 < p \le m$.

Theorem 2 *All the p MOP problems for $\{A_1, A_2, \cdots, A_p\}$ where $1 \le p \le m$, i.e., $D^{(1)}_{1,p}$ for $1 < p \le m$, can be computed in $O(m \log m)$ time.*

Proof Assume that there exists an integer θ such that $m = 2^\theta$. Based on Theorem 1, we can compute $D^{(1)}_{1,\frac{1}{2}m}$ in $O(m)$ time. Assume that $A_{b_{\frac{m}{2}}}$ is the optimal solution for the problem of caching only one version with $\{A_1, A_2, \cdots, A_{\frac{m}{2}-1}\}$, then we can find the optimal solution for the problem of caching only one version for $\{A_1, A_2, \cdots, A_{\frac{m}{4}}\}$ in $O(m)$ time. Similarly, $D^{(1)}_{1,\frac{3m}{4}}$ can also be computed by solving the problem of caching only one version with $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$. As we have already computed $C_{1,\frac{m}{2}}(A_y)$ where $y = \min(b_{\frac{m}{2}}, \frac{m}{2} - 1)$, we can base on this result to compute $C_{1,\frac{3m}{4}}(A_y)$ for $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$ (by adding at most $\frac{m}{4}$ terms to $C_{1,\frac{m}{2}}(A_{\frac{m}{2}-1})$. We then compute $C_{1,\frac{3m}{4}}(A_y), C_{1,\frac{3m}{4}}(A_{y+1}), \cdots, C_{1,\frac{3m}{4}}(A_{\frac{3m}{4}-1})$ in at most $O(\frac{3m}{4} - y)$ time. So it takes at most $O(m)$ time to compute $D^{(1)}_{1,\frac{m}{4}}$ and $D^{(1)}_{1,\frac{3m}{4}}$. Ac-

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

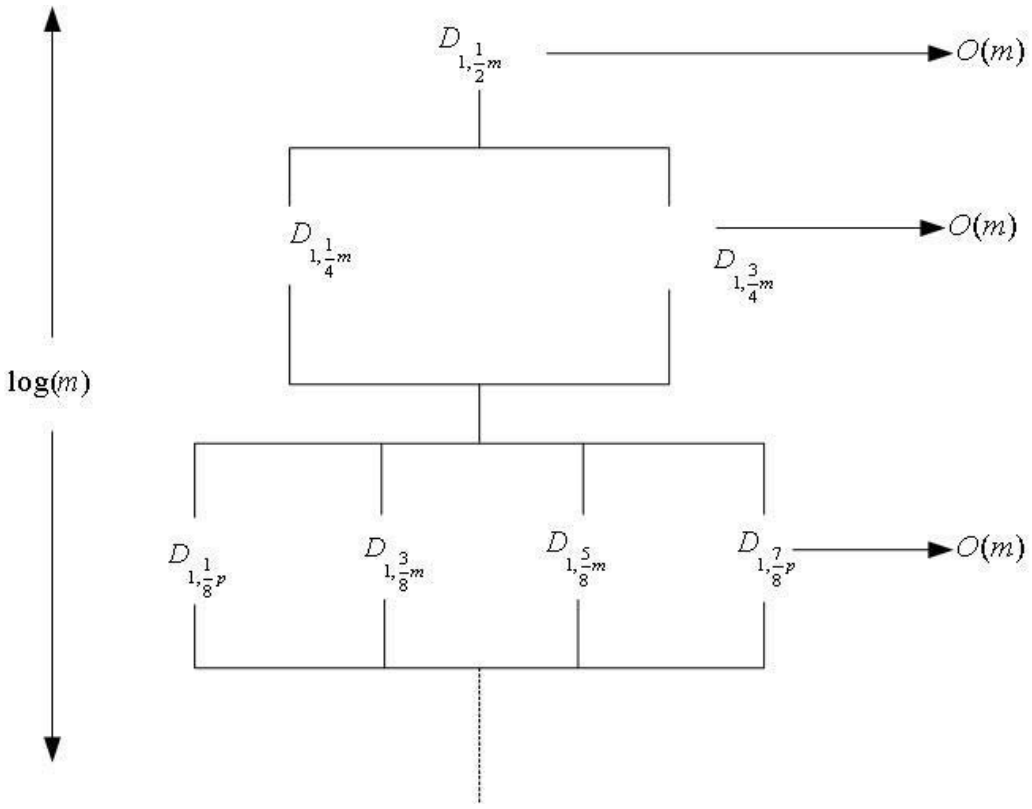

Figure 3: Decomposition Process of Theorem

cording to the similar decomposition, $D^{(1)}_{1,\frac{m}{8}}$, $D^{(1)}_{1,\frac{3m}{8}}$, $D^{(1)}_{1,\frac{5m}{8}}$, and $D^{(1)}_{1,\frac{7m}{8}}$ can all be solved in $O(m)$ time. To be precise, let $A_{z_1}, A_{z_2}, A_{z_3}$ be the optimal versions for $\{A_1, A_2, \cdots, A_{\frac{m}{4}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{m}{2}-1}\}$, and $\{A_1, A_2, \cdots, A_{\frac{3m}{4}-1}\}$ respectively. The first step is to compute $C_{1,\frac{m}{8}}(A_1)$, and then $C_{1,\frac{3m}{8}}(A_{z_1})$, $C_{1,\frac{5m}{8}}(A_{z_2})$, and $C_{1,\frac{7m}{8}}(A_{z_3})$ from $C_{1,\frac{m}{4}}(A_{z_1})$, $C_{1,\frac{m}{2}}(A_{z_2})$, and $C_{1,\frac{3m}{4}}(A_{z_3})$ respectively. As the computation of each item takes $O(\frac{m}{8})$ time, this step takes $O(m)$ time in total. Then we can search the optimal solutions for $\{A_1, A_2, \cdots, A_{\frac{m}{8}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{3m}{8}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{5m}{8}-1}\}$, $\{A_1, A_2, \cdots, A_{\frac{7m}{8}-1}\}$ in the ranges $(1, \min\{z_1, \frac{m}{8}-1\})$, $(z_1, \min\{z_2, \frac{3m}{8}-1\})$, $(z_2, \min\{z_3, \frac{5m}{8}-1\})$, and $(z_3, \frac{7m}{8}-1)$ respectively. Since each step takes constant time, all these searches take no more than $O(m)$ time in total. After repeating this process $\log m$ times, we can finish computing $D^{(1)}_{1,p}$ for $1 < p \leq m$. This process can be viewed from Figure 3. Hence, the theorem is proven. □

Now we can accomplish the problem of caching two versions in the following three steps.

- *Step* 1: Evaluate $D^{(1)}_{1,p}$ for $1 < p \leq m$,

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

where $D^{(1)}_{1,p}$ denotes the minimum access cost of caching only one version for the MOP problem with $p-1$ versions, i.e., $A_1, A_2, \cdots, A_{p-1}$. In particular, $D^{(1)}_{1,m+1} = \min_{1 \le k \le m}\{C_{1,m+1}(A_k)\}$.

- *Step* 2: Evaluate $\overline{D}_p$ for $2 \le p \le m$, where $\overline{D}_p$ is the access cost for versions $A_p, A_{p+1}, \cdots, A_m$ if A_p is cached at node v. $\overline{D}_p$ is defined as follows:

$$\overline{D}_p = \begin{cases} \sum_{i=p}^{p+\delta-1} f_{1,i}(i-p)T + \sum_{i=p+\delta}^{m} f_{1,i}L \\ \qquad\qquad \text{if } p+\delta \le m \\ \sum_{i=p}^{m} f_{1,i}(i-p)T \quad \text{if } p+\delta > m \end{cases}$$

- *Step* 3: Compute $D^{(2)}_{1,m}$, where $D^{(2)}_{1,m}$ is the minimum access cost of caching two versions for the problem with $\{A_1, A_2, \cdots, A_m\}$. $D^{(2)}_{1,m}$ is calculated as follows:

$$D^{(2)}_{1,m} = \min_{2 \le p \le m}\{D^{(1)}_{1,p} + \overline{D}_p)\}$$

The following theorem shows that $D^{(2)}_{1,m}$ is the minimum access cost of caching two versions the MOP problem.

Theorem 3 *$D^{(2)}_{1,m}$ is the minimum access cost of caching two versions for the MOP problem.*

Proof Assume that $D^{(2)}_{1,m} = D^{(1)}_{1,p^*} + \overline{D}_{p^*} = \min_{2 \le p \le m}\{D^{(1)}_{1,p} + \overline{D}_p)\}$. It is obvious from the computation of $D^{(2)}_{1,m}$ that b_{p^*} and A_{p^*} are the two versions which achieve the minimum access cost of caching two versions, where $D^{(1)}_{1,p^*} = C_{1,p^*}(b_{p^*})$. Hence, the theorem is proven. □

The following theorem shows the time complexity of computing $D^{(2)}_{1,m}$.

Theorem 4 *$D^{(2)}_{1,m}$ can be computed in $O(m \log m)$ time.*

Proof Since *Step* 1 can be computed in $O(m \log m)$ time (Theorem 2) and *Steps* 2 *and 3* both take $O(m)$ time, the total time for computing $D^{(2)}_{1,m}$ is $O(m \log m)$. Hence, the theorem is proven. □

After we have calculated $D^{(1)}_{1,p}$ for $1 \le p \le m$ in *Step* 1, we can obtain $D^{(2)}_{1,p}$ for all $2 \le p \le m$ in another $O(m \log m)$ time by divide and conquer, where $D^{(2)}_{1,p}$ is the minimum access cost of caching only two versions for the problem with $p-1$ versions, i.e., $A_1, A_2, \cdots, A_{p-1}$. The main idea is similar to Lemma 1 in the finding of $D^{(1)}_{1,p}$. Assume that $A_{b_{p_1}}$ and $A_{b_{p_2}}$ with $1 \le b_{p_1} < b_{p_2} < p$ are the two optimal versions cached in node v for $A_1, A_2, \cdots, A_{p-1}$ to achieve the optimal access cost $D^{(2)}_{1,p}$. Similarly, $A_{b_{q_1}}$ and $A_{b_{q_2}}$ with $1 \le b_{q_1} < b_{q_2} < q$ are the two optimal versions cached in node v for $A_1, A_2, \cdots, A_{q-1}$ to achieve the optimal access cost $D^{(2)}_{1,q}$. We can show with a similar argument with Lemma 1 that $b_{p_2} \le b_{q_2}$ if $p < q$ and this property limits the range of searching for the optimal solutions. As in Theorem 2, the two optimal solutions in $D^{(2)}_{1,\frac{m}{2}}$ can be found in $O(m)$ time after knowing the optimal versions of $D^{(1)}_{1,p}$ for $1 < p \le m$; then $D^{(2)}_{1,\frac{m}{4}}$ and $D^{(2)}_{1,\frac{3m}{4}}$ in another $O(m)$ time; then $D^{(2)}_{2,\frac{m}{8}}$, $D^{(2)}_{1,\frac{3m}{8}}$,$D^{(2)}_{1,\frac{5m}{8}}$, and $D^{(2)}_{1,\frac{7m}{8}}$ in another $O(m)$ time until $D^{(2)}_{1,p}$ for $2 < p \le m$ are found after $\log m$ times. Therefore, the minimum access cost of caching three versions, denoted by $D^{(3)}_{1,m}$, can be computed similarly, i.e., $D^{(3)}_{1,m} = \min_{3 \le p \le m}\{D^{(2)}_{1,p} + \overline{D}_p)\}$, with at most $O(m \log m)$ time (similar to Theorem 5). Using the same idea, we can solve the problem of caching K versions in $O(Km \log m)$ time.

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**

Let $D_{1,m}^{(K)}$ denote the minimum access cost of caching K versions from m versions, i.e., $A_1, A_2, \cdots, A_m$, then we have the following theorem on the time complexity of computing $D_{1,m}^{(K)}$.

Theorem 5 *$D_{1,m}^{(K)}$ can be computed in $O(Km \log m)$ time.*

Proof Based on the above analysis, we have $D_{1,m}^{(K)} = \min_{K \le p \le m} \{D_{1,p}^{(K-1)} + \overline{D}_p)\}$. Since $\overline{D}_p$ can all be computed in $O(m)$ time and we have showed that $D_{1,p}^{(1)}$ can be computed in $O(m \log m)$ time, we can easily prove that $D_{1,m}^{(K)}$ can be computed in $O(Km \log m)$ time by induction. Note that in the induction step, $D_{1,p}^{(K-1)}$ for $K - 1 < p \le m$ is computed in $O((K-1)m \log m)$ time. Hence, the theorem is proven. □

4 Conclusion

Transcoding is attracting increasing research interest in the environment of mobile appliances. In this paper, we addressed the problem of multimedia object placement. We studied this problem with the objective of minimizing total access cost by combining both transmission cost and transcoding cost.

References

[1] S. Chandra, C. Ellis, and A. Vahdat. *Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding*. IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, pp. 2544-2565, December 2000.

[2] C. Chang and M. Chen. *On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies*. IEEE Transactions on Parallel and Distributed Systems, Vol. 14, No. 6, pp. 611-624, June 2003.

[3] B. D. Davison. *A Web Caching Primer*. IEEE Internet Computing, Vol. 5, No. 4, pp. 38-45, 2001.

[4] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. *Dynamic Adaptation in An Image Transcoding Proxy for Mobile Web Browsing*. IEEE Personal Communications, Vol. 5, No. 6, pp. 8-17, December 1998.

[5] B. Shen, S.-J. Lee, and S. Basu. *Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks*. IEEE Transactions on Multemidia, Vol. 6, No. 2, pp. 375-386, April 2004.

[6] A. Vetro, C. Christopoulos, and H. Sun. *Video Transcoding Architectures and Techniques: An Overview*. IEEE Signal Processing Magazine, Vol. 20, No. 2, pp. 18-29, March 2003.

[7] J. Wang. *A Survey of Web Caching Schemes for the Internet*. ACM Computer Communication Review, Vol. 29, No. 5, pp. 36-46, 1999.

Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)
0-7695-2356-0/05 $ 20.00 **IEEE**