On Designing a Shortest-Path-Based Cache Replacement in a Transcoding Proxy

Hao-Ping Hung and Ming-Syan Chen*
Graduate Institute of Communication Engineering
National Taiwan University
mschen@cc.ee.ntu.edu.tw, hphung@arbor.ee.ntu.edu.tw

Abstract

The technology advance in network has accelerated the development of multimedia applications over the wired and wireless communication. To alleviate network congestion and to reduce latency and workload on multimedia servers, the concept of multimedia proxy has been proposed to cache popular contents. Caching the data objects can relieve the bandwidth demand on the external network, and reduce the average time to load a remote data object to local side. Since the effectiveness of a proxy server depends largely on cache replacement policy, various approaches are proposed in recent years. In this paper, we discuss the cache replacement policy in a multimedia transcoding proxy. Unlike the cache replacement for conventional web objects, to replace some elements with others in the cache of a transcoding proxy, we should further consider the transcoding relationship among the cached items. To maintain the transcoding relationship and to perform cache replacement, we propose in this paper the RESP framework (standing for Replacement with Shortest Path). The RESP framework contains two primary components, i.e., procedure MASP (standing for Minimum Aggregate Cost with Shortest Path) and algorithm EBR (standing for Exchange-Based Replacement). Procedure MASP maintains the transcoding relationship using a shortest path table, whereas algorithm EBR performs cache replacement according to an exchanging strategy. The experimental results show that the RESP framework can approximate the optimal cache replacement with much lower execution time for processing user queries.

Keywords: Transcoding proxy, Cache replacement

*The corresponding author

1 Introduction

In modern communication system, users can access the information from the remote server via various devices such as PCs, cellular phones, PDAs or laptops. Due to this trend, content providers should provide multiple versions of the same content according to each client device's computing power such as display resolution, memory space, CPU power, and network bandwidth as well as software running on it. Such a technique is regarded as *content adaptation* [16]. The transcoding technology, which is defined as converting a multimedia object from one form to another, is a popular way to realize *content adaptation*. A proxy capable of transcoding (referred to as the transcoding proxy) is placed between clients and an information server to combine the mismatch between what the server provides and what the clients prefer.

Compared to the client-based [29] and server-based [18] technologies, the proxy-based transcoding has the following advantageous features. First, unlike the client devices, an intermediate proxy has larger bandwidth to download the web objects and higher computing power to perform encoding/decoding, which enables proxy-based transcoding to effectively reduce the waiting time for clients to access the required version. Moreover, different from the server-based transcoding in which the server should keep all possible versions of an object at its disk, the proxy-based technologies can perform on-the-fly transcoding for dynamic *content adaptation*. Figure 1 shows the architecture of the transcoding proxy. There are two primary components: transcoder and cache manager. After downloading the web objects from the remote servers, the transcoder will perform transcoding according to the clients' requirements. The transcoding results will be kept in the local storage (referred to as cache) of the proxy. Since the cache has limited capacity, the cache manager will perform cache replacement to evict the less valuable objects. Research topics in proxy-based transcoding include cache replacement schemes [4][5][11][15], system architectures [10][12], and proxy collaboration [4].

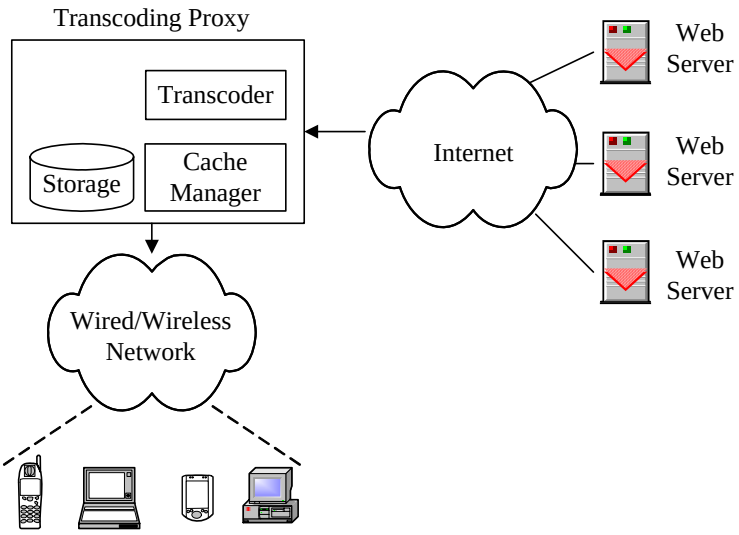

Figure 1: The transcoding proxy architecture

Due to the advances of Internet technologies, the Web has been constantly growing and so has the load on the Internet and Web servers [22]. To alleviate the network bandwidth, user-perceived delays and the loads on the origin server, web caching in a proxy becomes an important technique to scale the Internet. Till now, various caching strategies have been proposed to suit the web proxies for different purposes. In general, the cache replacement policies can be divided into the following categories.

- recency-based strategies [1][25], in which the server tends to cache the most recent web objects.
- frequency-based strategies [2][30], in which the server tends to cache the most frequent web objects.
- hybrid strategies [6][17], which are the combinations of recency-based and frequency-based strategies.
- function-based strategies [3][26], which use a potentially general function to calculate the value of an object. The cache replacement will be performed according to the

value.

- randomized strategies[21][24], which use randomized policies to determine the object for replacement.

In recent years, since various of multimedia data are distributed in the Internet, more factors should be considered in caching strategies. In addition to recency and frequency, the proxy should also take the characteristics of multimedia objects such as QoS requirement into account. Therefore, the function-based strategies which are capable of considering multiple factors simultaneously become the most commonly used approaches in modern Internet computing.

In this paper, we focus on designing an effective and efficient cache replacement algorithm in a transcoding proxy. Different from the conventional schemes, to design the cache replacement in transcoding proxy, we should take the transcoding relationship among the cached elements into consideration. The transcoding relationship indicates the relevant cost of obtaining an transcoded version from each cached element. For example, assume that the transcoding proxy tries to cache some versions of a specific object. If a more detailed version exists in the cache, the proxy can perform transcoding procedure instead of fetching the exact version from the remote server. Moreover, the profit of caching a new element will also depend on the current elements in the cache. If there is a sophisticated version in the cache, it will be less profitable to cache an rough version. Therefore, the conventional cache replacement algorithms [1][2][7][19][28] will not be suitable in the transcoding proxy architecture. In [4], the authors proposed *coverage-based* and *demand-based* scheme to manage the cached elements in a transcoding proxy. Although they are very efficient, some important features of the objects such as reference rate, object size and the interrelationship are neglected in the design of the algorithms. In [5], the authors regarded the transcoding relationship as a *transcoding graph*, and proposed algorithm AE can be viewed as a *greedy* algorithm in guiding the caching policy. As for the work in [11], algorithm MPR

was proposed to reach an optimal solution by using the concept of *dynamic programming*. It is noted that algorithm MPR, which is based on the concept of *dynamic programming*, requires high overhead in memory and computing complexity, whereas the simple strategy adopted by algorithm AE may waste some cache space.

To remedy the drawbacks of prior works, we propose in this paper the RESP framework (standing for REplacement with Shortest Path) to maintain the transcoding graph and to perform cache replacement in a transcoding proxy. The RESP framework contains two primary components, i.e., procedure MASP (standing for Minimum Aggregate cost with Shortest Path) and algorithm EBR (standing for Exchange-Based Replacement). To maintain the transcoding graph, procedure MASP uses a table to record the shortest path information and determines the transcoding sources according to the table. Specifically, procedure MASP can optimally select the most suitable transcoding sources by generating the *minimum cost transcoding graph*, which is a subgraph of the *weighted transcoding graph*. Compared to the prior works, procedure MASP can deal with more complicated transcoding graphs by detecting the shortest path from one vertex to another. On the other hand, algorithm EBR performs cache replacement by exchanging a candidate element with some of the less profitable elements in the cache so as to maximize the profit of cached elements. To quantify the performance of the transcoding proxy, we employ the *profit function* proposed in [5]. The exchanging policy is that the candidate element should have larger $profit$ and smaller size than the summation of $profit$ and the aggregate size of those less profitable elements. Moreover, a heap is also used to maintain the cached elements. The most advantageous feature is that algorithm EBR will utilize the cache space very thoroughly.

To validate the performances of the RESP framework, several experiments are conducted. During the experiments, we first discuss the performances of procedure MASP empirically and show that MASP can perform well in dealing with the complex transcoding relationship. We also analyze the proposed cache replacement algorithm EBR from the viewpoints of cache hit ratio, and observe that under many circumstances, the EBR algorithm can approximate

the optimal solution very effectively. Moreover, in the efficiency analysis, we find that the EBR algorithm costs much less execution time in processing user queries. The experimental study will justify the practicability of the RESP framework over the prior research in cache replacement of the transcoding proxy.

This paper is organized as follows. Preliminaries including related works, notations, and definitions are given in Section 2. In Section 3, we describe the design of the RESP framework. The experimental results are shown in Section 4. Finally, this paper concludes with Section 5.

2 Preliminaries

2.1 Related Works

Unlike the caching schemes in the traditional web proxy, to obtain the queried element in a transcoding proxy, we can either download the element from the remote server or perform transcoding on current cached elements. Due to this transcoding relationship among the cached elements, conventional web cache mechanism may not be suitable in such scenario. To extend the conventional caching algorithms in the transcoding proxy architecture, two strategies are proposed in [4]: the *coverage-based* and the *demand-based* strategy. To process the clients request for multimedia web objects, if there is an exact version stored in the cache, the transcoding proxy will return it directly. Otherwise, the proxy will transcode from the more detailed version. In the coverage-based strategy, the proxy will always choose the original version of the object to cache. When the less detailed versions are requested, they can directly be transcoded from the cached original version. Although such a strategy can effectively save the transmission delay, it is a waste of the cache space since the most detailed version has the largest item size. In the demand-based strategy, the proxy will always choose the exact version to cache. Note that the demand-based strategy can be viewed as the LRU algorithm, which merely results in a fair performance since the transcoding relationship

between the cached elements is neglected.

To improve the performance of the cache replacement policy in a transcoding proxy, Chang et. al. proposed an efficient algorithm AE, standing for Aggregate Effect, in [5]. Algorithm AE performs cache replacement according to a specific profit function of each cached element. Since the value of profit is a combination of reference rate, which reflects the popularity, and the delay saving, which quantifies the transcoding relationship, algorithm AE outperforms the coverage-based strategy and the demand-based strategy. To process the request for a specific version of an object, algorithm AE first caches the requested version and its original version. If the proxy has insufficient cache space, the item with the smallest *generalized profit* will be evicted until the total item size is less then the cache capacity. To achieve the optimal cache replacement, Hung et. al. proposed algorithm MPR [11], standing for Maximum Profit Replacement. The intuition of algorithm MPR is as follows. Given a database D, the subset D_H, which is also regarded as the *caching candidate set*, is selected to represent the elements with the high priority to be cached. The problem of determining D_H can be viewed as a *0-1 knapsack problem* [9]. Since the *0-1 knapsack problem* can be optimally solved by the concept of *dynamic programming*, algorithm MPR, which performs cache replacement according to the content of D_H, can effectively optimize the cache utilization. Compared to algorithm AE which performs like a *greedy algorithm*, MPR requires much more computing power and memory to calculate and maintain the content of D_H. In addition to a simple caching strategies, several advanced topics in the cache replacement over transcoding proxy are also discussed. In [23], the authors discussed transcoding and caching the streaming media distribution over the Internet. Three algorithms TEC-11, TEC-12 and TEC-2 are proposed. The pros and cons for each algorithm are also analyzed. On the other hand, the consistency issue during the cache replacement in a transcoding proxy is also discussed in [13][14][15].

Symbol	Description
G_i	the weighted transcode graph of object i
$V[G_i]$	the set of all vertices in the G_i
$E[G_i]$	the set of all edges in the G_i
$o_{i,j}$	version j of object i
$r_{i,j}$	the mean reference rate of $o_{i,j}$
d_i	the delay of fetching object i from server to proxy
$s_{i,j}$	the size of $o_{i,j}$

Table 1: Description of the symbols

2.2 Notation and Definitions

The symbols used throughout this paper are listed in Table 1. Without loss of generality, we adopt the same analytical model used in [5][11]. Assume that an object can be represented in n versions. The original version of object i is denoted as $o_{i,1}$, whereas the least detailed version which cannot be transcoded any more is denoted as $o_{i,n}$.

Definition 1: The *weighted transcoding graph*, G_i, is a directed graph with weight function ω_i. G_i depicts the transcoding relationship among transcodable versions of object i. Each vertex $v \in V[G_i]$ represents a transcodable version of object i. Version u of object i, i.e., $o_{i,u}$ is transcodable to version v, i.e., $o_{i,v}$ iff there is a directed edge $(u, v) \in E[G_i]$. The transcoding cost from version u to v is given by $\omega_i(u, v)$, regarded as the weight of the edge from u to v. Figure 2(a) illustrates an example of a weighted transcoding graph.

Definition 2: $PF(o_{i,j})$ is defined as the *singular profit* of caching $o_{i,j}$ while no other version of object i is cached. The singular profit can be formulated as

$$PF(o_{i,j}) = \sum_{(j,x)\in E[G_i]} r_{i,x} * (d_i + \omega_i(1, x) - \omega_i(j, x)).$$

It is noted that there are two primary components in the profit function. The reference rate $r_{i,x}$ reflects the popularity of $o_{i,j}$. On the other hand, the physical meaning of $d_i + \omega_i(1, x) - \omega_i(j, x)$ is the amount of delay saving of caching $o_{i,j}$ compared to the situation

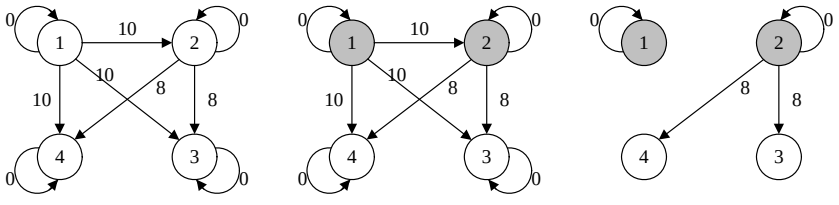

Figure 2: Illustration of minimum cost transcoding graph

at which no elements are cached. Therefore, the higher $PF(o_{i,j})$ indicates that $o_{i,j}$ is more profitable to be cached in the transcoding proxy.

Definition 3: The *minimum cost transcoding graph*, G'_i, is a subgraph in which the aggregate transcoding cost is minimized when the proxy caches multiple versions of an object simultaneously. For example, considering that $o_{i,1}$ and $o_{i,2}$ are cached, as shown in Figure 2(b), we can obtain the *minimum cost transcoding graph* (i.e., Figure 2(c)) by retaining the edges with the *cheapest* weight from the cached versions to all the other versions.

Definition 4: $PF(o_{i,j1}, o_{i,j2}, ..., o_{i,jk})$ is defined as the *aggregate profit* of caching multiple objects $o_{i,j1}, o_{i,j2}, ..., o_{i,jk}$ simultaneously.

$$\begin{aligned} & PF(o_{i,j1}, o_{i,j2}, ..., o_{i,jk}) \\ = & \sum_{v \in V[G'_i]} \sum_{(v,x) \in E[G'_i]} r_{i,x} * (d_i + \omega_i(1,x) - \omega_i(v,x)). \end{aligned}$$

Definition 5: $PF(o_{i,j}|o_{i,j1}, o_{i,j2}, ..., o_{i,jk})$ is defined as the *marginal profit* of caching $o_{i,j}$, given that $o_{i,j1}, o_{i,j2}, ..., o_{i,jk}$ are already cached where $j \neq j1, j2, ..., jk$.

$$\begin{aligned} & PF(o_{i,j}|o_{i,j1,...}, o_{i,jk}) \\ = \; & PF(o_{i,j}, o_{i,j1,...}, o_{i,jk}) - PF(o_{i,j1,...}, o_{i,jk}). \end{aligned} \quad (1)$$

When some versions already exist in the proxy. If a new version is cached, the amount of additional profit can be derived according to Eq. (1).

Given the database D which represents the collection of all possible queried data objects and the corresponding versions, the element $o_{i,j}$ contains two attributes: *profit* $p_{i,j}$ and

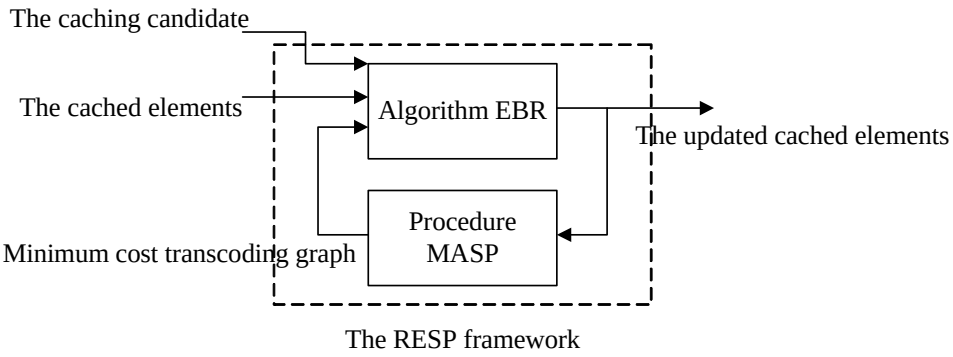

Figure 3: The overview of the RESP framework

item size $s_{i,j}$. Note that $p_{i,j} = PF(o_{i,j})$ if no other version of object i is cached, and $p_{i,j} = PF(o_{i,j}|o_{i,j1,\ldots}, o_{i,jk})$ if there are some other versions $o_{i,j1,\ldots}, o_{i,jk}$ cached.

Definition 6: The *generalized profit*, $g_{i,j}$, of the element $o_{i,j}$ is defined as

$$g_{i,j} = \frac{p_{i,j}}{s_{i,j}}.$$

3 Design of the RESP Framework

3.1 Overview

The RESP framework (standing for REplacement with Shortest Path) is designed for a transcoding proxy to perform cache replacement. Two primary components are contained, i.e., procedure MASP (standing for Minimum Aggregate cost with Shortest Path) and algorithm EBR (standing for Exchange-Based Replacement). Figure 3 shows the architecture of the RESP framework. A user query for a specific element identified by the object number and version number corresponds to a *caching candidate*. With the caching candidate and the current cache state as the input, algorithm EBR will determine which elements in the cache should be replaced with the caching candidate. On the other hand, during the execution of algorithm EBR, procedure MASP will update the minimum cost transcoding graph dynamically so as to maintain the transcoding relationship among the cached elements.

3.2 Generating Transcoding Graph with Shortest Path

When a transcoding proxy processes the queries sent by the users, one important issue is to determine the transcoding source from the cached elements so as to minimize the transcoding delay. As mentioned in Section 2, the minimum cost transcoding graph, which is a subset of the transcoding graph, will indicate the transcoding source. In this section, we design an effective procedure MASP to generate the minimum cost transcoding graph for cache replacement. Unlike the conventional approach MATC [5] (standing for Minimum Aggregate Transcoding Cost) which selects the transcoding sources from adjacent vertices in the transcoding graph and only performs well on simple transcoding relationship, procedure MASP, which is based on the concept of the shortest path, can achieve the optimal solution in maintaining the complex transcoding relationship. The algorithmic form of procedure MASP is outlined as follows.

Procedure MASP
Input: The weighted transcoding graph G, the set of cached versions C
Output: The minimum transcoding graph G'
Begin
01: Create a shortest path table T with $|C| \times |V[G]|$ entries
02: Initialize the value of each $T(i,j)$ to be ∞ if $j < i$
03: **for** each vertex $v(i)$ belonging to C
04: **for** each vertex $v(j)$ with $j \geq i$
05: Find out the shortest path $p(i,j)$ from $v(i)$ to $v(j)$
06: Store the value of $p(i,j)$ in $T(i,j)$
07: **end for**
08: **end for**
09: Create an array Src with $V[G\prime]$ elements
10: **for** each column j in T
11: $Src[j] = \arg\min\{T(i,j)|$ for each $i \leq j\}$
12: **end for**
13: Generate $G\prime$ according to Src
14: **return** $G\prime$
End

Given the transcoding graph G and the cached versions C, the goal of procedure MASP

is to obtain the minimum-cost transcoding graph in such a way that the cost of getting each transcoded version can be minimized. The basic idea behind the MASP procedure is that the transcoding source with the minimum cost must contribute to the minimum aggregated weight among all the shortest paths from the cached elements to each version. Initially, procedure creates a table T with $|C|$ by $|V[G]|$ entries. After that, each entry $T(i,j)$ with $i > j$ will be filled with ∞ because a less detailed version cannot be transcoded to a detailed version. In the remaining entries, we will store the aggregate cost along shortest path from each cached version to all the other transcodable versions. To calculate the shortest path, we employ Dijkatra's algorithm [8] in this paper. According to table T, procedure MASP will obtain the minimum-cost transcoding graph by selecting the cached version with minimum cost along the shortest path as the transcoding source.

Lemma 1: Let $|V|$ and $|E|$ denote the number of vertices and the number of edges in the transcoding graph G, respectively. Also let $|C|$ be the number of cached versions. The complexity of procedure MASP is $O(|C||E| + |C||V|\log|V|)$.

Proof: Initially, the operation in Line 2 costs $O(|C||V|)$. Next, for each vertex belonging to C, we should find the shortest path to other transcodable versions. It is noted that the complexity of Dijkatra's algorithm is $O(|E| + |V|\log|V|)$. Finally, the operations from Line 10 to Line 12 cost $O(|C||V|)$. Therefore, the complexity of MASP will be $O(|C||V|+|C||E|+|C||V|\log|V| + |C||V|) = O(|C||E| + |C||V|\log|V|)$. ■

Procedure MASP can be illustrated according to the running scenario as follows. Consider the weighted transcoding graph G shown in Figure 4(a) with three versions cached. The cached versions are marked in shadows. Procedure MASP will first create a 3×6 shortest path table T. Each entry $T(i,j)$ records the minimum transcoding cost from $v(i)$ to $v(j)$. Initially, as shown in Figure 5(a), each $T(i,j)$ with $i > j$ will be filled with ∞ because a less detailed version cannot be transcoded to a detailed version. Next, in Figure 5(b), MASP will check each vertex $v(i)$. If $v(i)$ belongs to the set of cached versions C, each entry $T(i,j)$ will be filled with the corresponding cost to the shortest path from $v(i)$ to $v(j)$. To determine

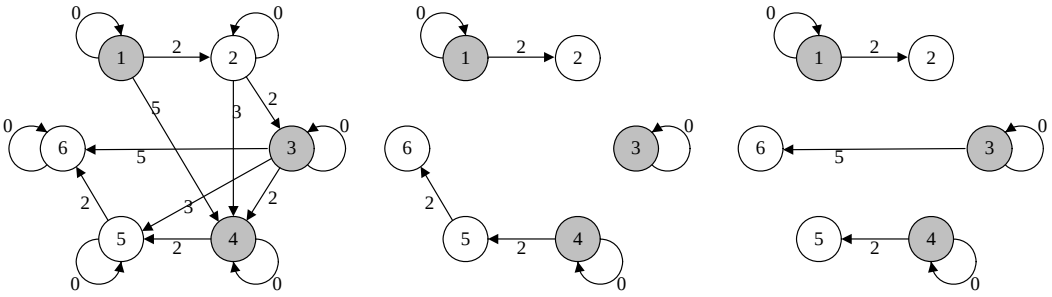

Figure 4: Comparison between MASP and MATC

the transcoding source of each version, the entry with the minimum value will be marked for each column j in T. The row number of the marked entry corresponds to the transcoding source of $v(j)$. Finally, in Figure 5(c), the minimum cost transcoding graph will be generated according to the transcoding sources of each version. Unlike procedure MATC [5] in which each vertex is only aware of the transcoding cost of the adjacent vertices, procedure MASP collects complete path information. According to Figure 4(b) and Figure 4(c) which show the minimum cost transcoding graph generated by MASP and MATC, respectively, we can find that version 6 should be transcoded from version 4 instead of version 3.

3.3 Exchange-Based Cache Replacement Algorithm

As described in [11], the problem of cache replacement in the transcoding proxy can be formulated as a *0-1 knapsack problem*. In [11], although algorithm MPR (standing for Maximum Profit Replacement) can achieve an optimal solution, it requires high memory and CPU overhead. On the other hand, algorithm AE (standing for Aggregate Effect) proposed in [5] can be executed very efficiently. However, the simple strategy adopted by AE (i.e., caching the requested element and its original version simultaneously and evicting the elements with smaller *generalized profit*) may result in the waste of cache space. To remedy the drawbacks of AE and MPR, we propose algorithm EBR, which can be executed

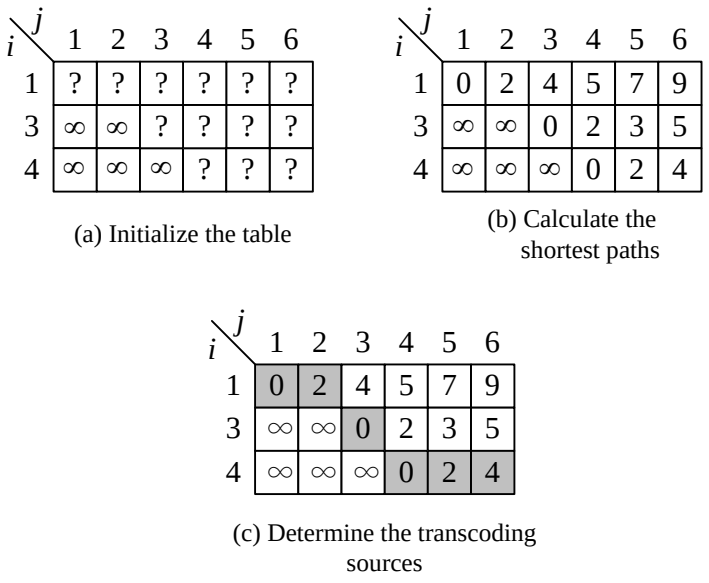

Figure 5: Execution of procedure MASP

efficiently to approximate the optimal cache replacement. The algorithmic form of EBR is outlined as follows.

Algorithm EBR
Input: Heap H, element $o_{i,j}$, cache capacity Z
Output: Heap after cache replacement
(Note: the elements in H are sorted in an ascending order according to the generalized profit $g_{i,j}$)
Begin
01: Z_H = the aggregate size of the cached elements
02: **if** $Z_H + s_{i,j} \leq Z$
03: Insert $o_{i,j}$ into H
04: Update H using procedure MASP
05: **else**
06: pos = the position of the first element in H
07: $S = 0$
08: $P = 0$
09: **while** true
10: $S = S+$ the size of the object at pos
11: $P = P+$ the profit of the object at pos
12: $pos++$

13: **if** $P < p_{i,j}$ and $S > s_{i,j}$
14: Remove the elements before *pos* from H
15: Insert $o_{i,j}$ into H
16: Update H using procedure MASP
17: **break**
18: **else if** $P \geq p_{i,j}$
19: **break**
20: **end if**
21: **end while**
22: **end if**
23: **return** H
End

To design algorithm EBR, we are inspired from the concept proposed in [27], where the optimal cache replacement for web objects is approximated. In addition, we further take the characteristics of the transcoding proxy system into consideration. The execution of algorithm EBR is described as follows. We first regard the cache as a *heap* H, in which the elements are sorted in ascending order according to the *generalized profit*. Next, given the caching candidate element $o_{i,j}$, if the elements before the running pointer *pos* have larger aggregate size and smaller aggregate profit, algorithm EBR will replace these elements with $o_{i,j}$. Note that algorithm EBR is only executed when the cache has insufficient space to retain $o_{i,j}$. Otherwise, the transcoding proxy will cache $o_{i,j}$ without removing any elements. Moreover, since there is inter-relationship among the elements in the transcoding proxy, procedure MASP is employed to dynamically update the profit information in H. Unlike algorithm AE [5] which may waste some cache space, algorithm EBR utilizes the cache space very thoroughly. The insight of algorithm EBR is that we use a more profitable element to *exchange* some less profitable elements in the cache and guarantee that the total amount of profit of the cached elements are increased monotonically. Therefore, algorithm EBR can effectively maximize the profit of cached elements with much lower computation cost compared to algorithm MPR [11]. Figure 6 shows the flowchart of the detail transcoding and caching procedure of the EBR algorithm.

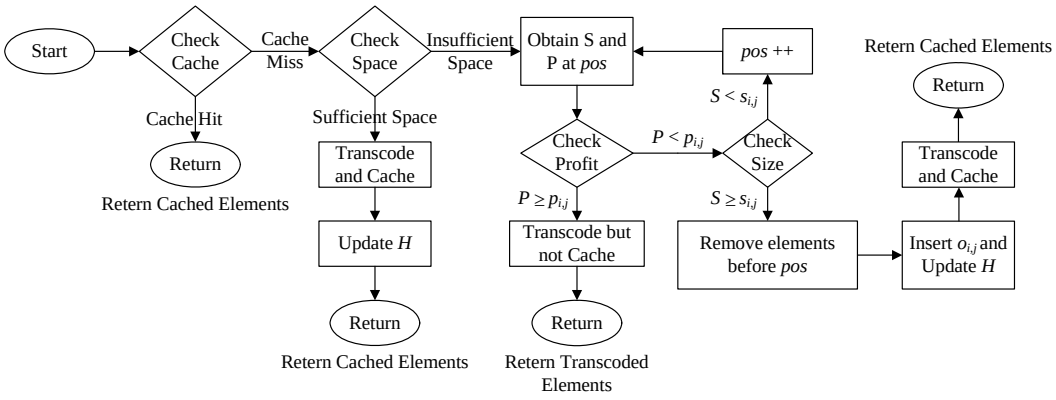

Figure 6: The flowchart of transcoding and caching procedure of the EBR algorithm

Parameters	Default Values
Number of objects (N)	1500
Skewness parameter of (θ)	1
Diversity parameter (Φ)	200
Cache capacity ratio (R)	0.05
Number of queries (N_q)	100000

Table 2: Parameters during the simulation

4 Experimental Study

To inspect the performance of the RESP framework, several experiments are conducted. We evaluate the transcoding delay, the cache hit ratio and the efficiency of the system by varying important parameters. Several algorithms [5][11] in prior works are also investigated empirically. We will describe the simulation environment in Section 4.1. In Section 4.2, we will discuss the impact of the procedure MASP in saving the transcoding delay. In Section 4.3, we will analyze the cache hit ratio of the proposed algorithm EBR. The efficiency of the EBR algorithm will be discussed in Section 4.4, and finally a brief remark will be given in Section 4.5.

$N_{version}$	Distribution of the access probability of the versions	Size of the versions
4	<15%,35%,35%,15%>	<100%, 75%,50%,25%>
5	<15%,20%,30%,20%,15%>	<100%,80%,60%,40%,20%>
6	<10%,15%,25%,25%,15%,10%>	<100%,83%,67%,50%,33%,17%>
7	<8%,12%,18%,24%,18%,12%,8%>	<100%,86%,71%,57%,43%,29%,14%>

Table 3: Distribution of access probability and size of the versions.

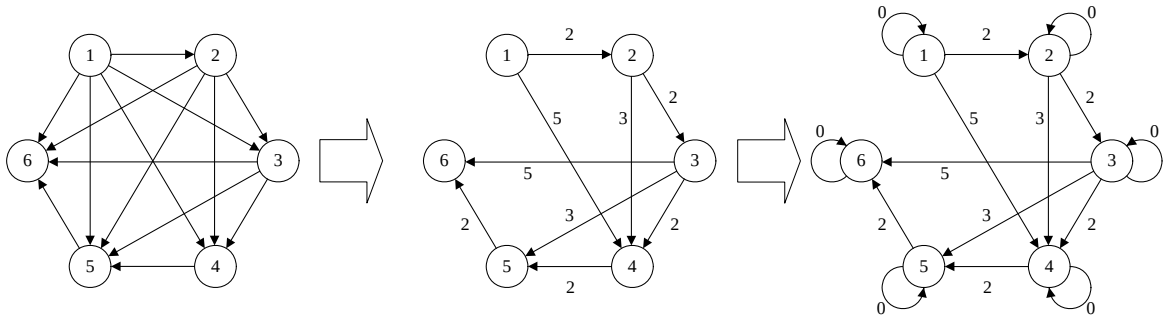

Figure 7: Procedure of generating the weighted transcoding graph

4.1 Simulating Environment

The simulation model is designed to reflect the system environment of the transcoding proxy in Section 2. Table 2 summaries the default settings of some primary simulation parameters. Let the database contains N multimedia objects which are transcodable to $N_{version}^{(i)}$ versions. $N_{version}^{(i)}$ is randomly selected from 4 to 7. Without loss of generality, we assume that the popularity of the versions of an object is Gaussian distributed. Moreover, the size of the versions decreases linearly as the increase of the version identification number. Table 3 shows the size of each version compared to the original version and the distribution of the access probability for different $N_{version}^{(i)}$. The procedure of generating the transcoding relationship is as follows. Given the number of versions $N_{version}^{(i)}$ for each object, we first create a *complete* graph which contains $N_{version}^{(i)}$ vertices. Each vertex is identified from 1 to $N_{version}^{(i)}$. The edge is directed from vertex v to vertex u if $v < u$. Next, some of the edges are randomly pruned. The pruning procedure should follow the constraint that there is at least one path

from the original version to each transcoded version. The weight of each remaining edge is also a Gaussian random variable. Its mean value is calculated according to the transcoding rate and the size of the versions. Finally, the weighted transcoding graph is finished after we add the edge (u, u) to each vertex u. The procedure of creating the weighted transcoding graph can be illustrated in Figure 7.

In the client model, user devices can be partitioned into seven classes. That is, each object can be transcoded to seven different versions by the transcoding proxy to satisfy the users' requirements. We generate the weighted transcoding graph of each object by removing several edges randomly from a complete graph. As for the transcoding proxy model, we assume that there are 1500 different objects by default (i.e., $N = 1500$), and choose the cache capacity to be $R * (\sum \text{object size})$, where R is regarded as the *capacity ratio*. The reference rate of each object is generated by Zipf distribution $r_i = (\frac{1}{i})^\theta \sum_{j=1}^{N} (\frac{1}{j})$, where θ is viewed as a *skewness parameter*. Note that large θ indicates the highly-skewed access patterns in the mobile computing environment. When $\theta = 0$, the access probabilities of the queries are uniformly distributed. By default, the value of θ is set to be 1 since it is observed in [20] that θ is usually larger than 1 for busy web sites. The size of each object is represented by the uniform distribution between $(0, \Phi]$ units, where we define Φ as the *diversity parameter*. At higher Φ, there will be higher variance of object sizes. As for the setting of other minor parameters, interested readers are referred to [5].

4.2 Delay Saving Analysis

In the first experiment, we evaluate the impact of the MASP procedure in generating the minimum-cost transcoding graph. We compare MASP with the conventional MATC approach [5]. Two procedures MASP and MATC are combined with the same cache replacement algorithm EBR. It is noted that the transcoding sources determined by the MASP/MATC procedure will affect the cost of transcoding when the RESP framework is executed. Usually, the cost will be reflected by the transcoding delay. Here we use the

delay saving ratio, denoted by DSR, as the performance metric. The delay saving ratio is defined as the ratio of total delay saved from cache hits to the total delay incurred under the circumstance without caching. Considering N_q queries, the delay saving ratio can be formulated as

$$DSR = \frac{\sum_{i=1}^{N_q} d_{i\_saving}}{\sum_{i=1}^{N_q} d_{i\_original}},$$

where d_{i_saving} denotes the delay saving for each query i if the a cache is employed, and $d_{i\_original}$ stands for the delay of each query i without the cache.

Figure 8 shows the delay saving ratio with the skewness parameter θ varied. We observe that at higher skewness parameter, higher delay saving ratio will be achieved for both approaches. The reason is that under the circumstances of high skewness, most of the clients tend to access a small part of elements. Therefore, the server can easily save the delay of transcoding by caching these elements with high access probabilities. In Figure 9, we evaluate the impact of the diversity parameter Φ. At higher diversity, since the average size of each element becomes larger, there are fewer cached elements, which also affect the performance of the delay saving. As for Figure 10, the delay saving ratio at different capacity ratios is depicted. From these three figures, since the MASP is capable of generating the optimal minimum-cost transcoding graph, in this figure, the RESP framework (i.e., MASP+EBR) outperforms the combination of MATC and EBR.

4.3 Discussion of Cache Hit Ratio

In the next experiment, we discuss the performances of the proposed cache replacement algorithm EBR. The effects of θ, Φ and $CRatio$ are also examined. Unlike the various metrics used in [5], we choose the tightest one, the *exact hit ratio*, denoted by EHR as the performance metric. The exact hit ratio is defined as the fraction of requests which are

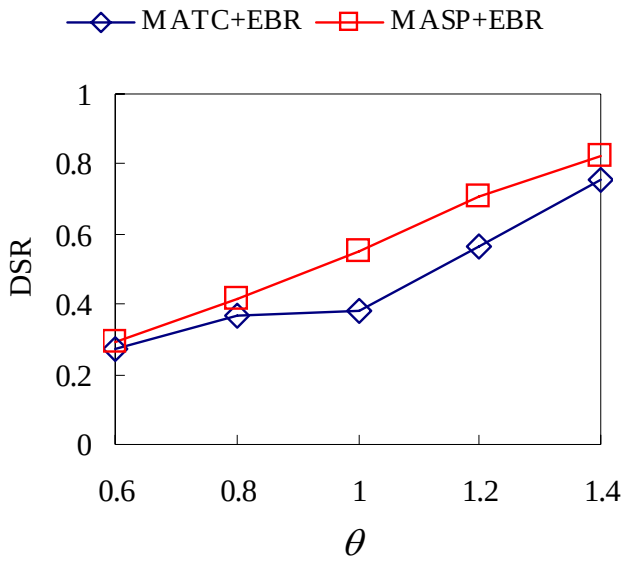

Figure 8: The delay saving ratio with the skewness parameter varied

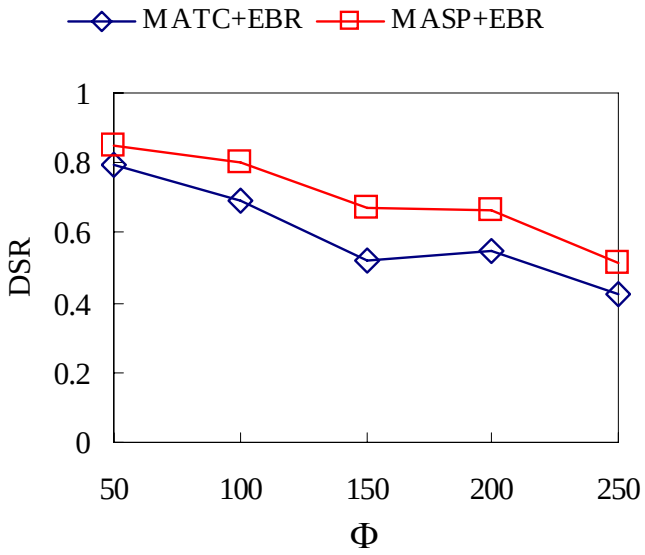

Figure 9: The delay saving ratio with the diversity parameter varied

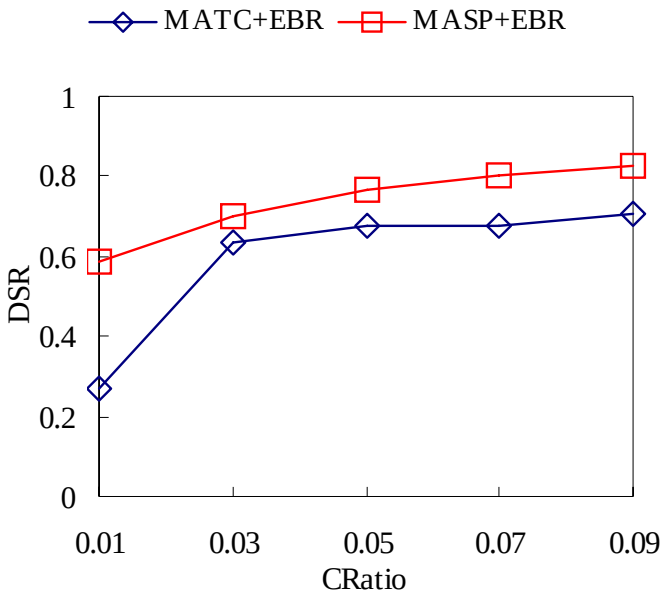

Figure 10: The delay saving ratio with the capacity ratio varied

satisfied by the exact versions of the cached objects. This metric is also motivated by the fact that we usually intend to provide an exact version to users (rather than an overqualified one) for effective bandwidth use. Formally, the exact hit ratio can be formulated as

$$EHR = \frac{\text{Number of exact cache hits}}{N_q}.$$

During this experiment, we compare algorithm EBR with the following competitive cache replacement approaches.

- MPR [11], a cache replacement algorithm which is capable of achieving optimal solutions in the transcoding proxy.
- AE [5], a cache replacement algorithm based on the concept of *aggregate effect*.

Moreover, we also implement a greedy algorithm, denoted by *Greedy*, for comparison purposes. Algorithm Greedy is very similar to algorithm AE since the transcoding proxy

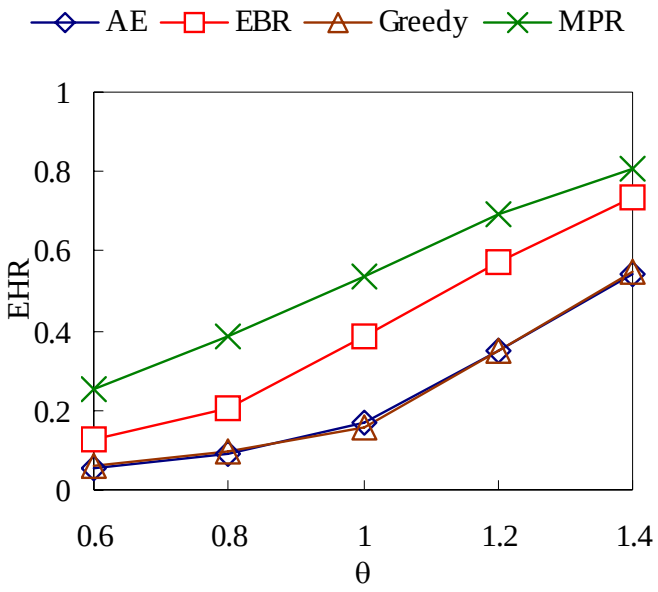

Figure 11: The exact cache hit ratio at different skewness parameters

always tries to cache the elements with larger generalized profit. The only difference is that algorithm Greedy does not cache the original version when a certain less detailed version is requested.

According to the measure of EHR at different values of the skewness parameter, in Figure 11, we observe that algorithm MPR has the most outstanding performance, whereas algorithm AE and Greedy only performs fairly. As for algorithm EBR, it results in a satisfactory EHR which is better than AE and Greedy but poorer than MPR. When θ is getting larger, the EHR of algorithm EBR is close to algorithm MPR. The reason is that at a highly skewed situation, the value of $profit$ will be dominated by the reference rate. Since algorithm EBR enhance the profit by increasing the *exact hit ratio*, which reflects the reference rate, it has a good performance when θ is larger. Therefore, algorithm EBR is very suitable for cache replacement in a highly skewed situation.

Figure 12 shows the exact hit ratio for relevant approaches with the diversity parameter Φ varied. From the viewpoints of the metric EHR, the discrepancy between algorithm MPR

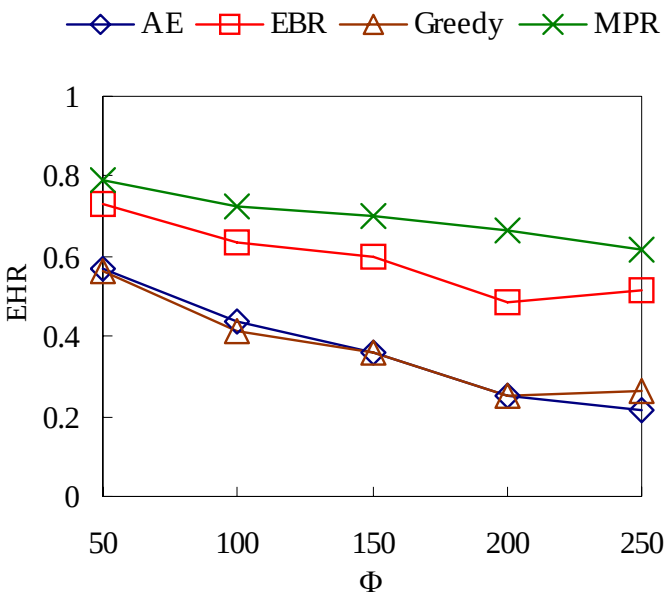

Figure 12: The exact cache hit ratio at different diversity parameters

and algorithm AE/Greedy will be noticeable in a highly diverse situation. It is because algorithms AE and Greedy will start to exclude the cached elements with the smaller generalized profit once the aggregate size exceeds the capacity. It is very likely that the last excluded element before the balance state is reached has large size. After algorithm AE and Greedy terminates the cache replacement procedure, there is still sufficient cache space to store some other elements. The waste of cache space will degrade the performance of algorithm AE and Greedy. Note that the increase of Φ will granulate the size of the cached elements. Therefore, algorithm AE and Greedy have poorer performance in EHR in a situation of higher diversity. As for algorithm EBR, the cache replacement is performed so as to exchange the caching candidate element with some cached elements which have the similar aggregate size but lower aggregate profit. According to this advantageous feature, algorithm EBR will not induce the waste of cache space. The discrepancy between algorithm EBR and algorithm MPR still remains subtle as the diversity parameter varies.

The effect of cache capacity is discussed in Figure 13. It is obvious and intuitive that the

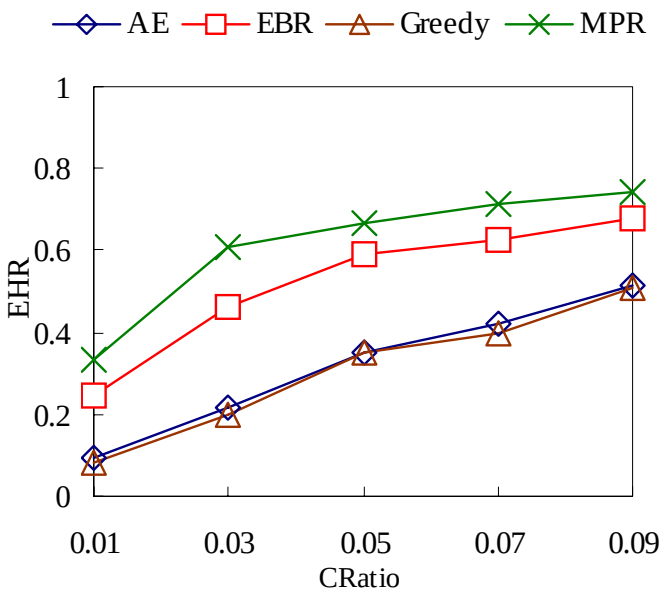

Figure 13: The exact cache hit ratio with $CRatio$ varied

relevant algorithms have better performances when there is more cache space in a transcoding proxy since more elements can be cached to reduce the delay and enhance the occurrence of cache hit. There is an interesting observation. For algorithms MPR and EBR, although the performances are improved gradually as we enlarge the capacity, they are saturated in the situation of higher capacity ratio. It implies that the marginal effect of the cache capacity diminishes as the increase of $CRatio$. The reason of the phenomenon is that for all elements including all objects and the corresponding versions, only part of them are worth caching. If we increase the capacity, the transcoding proxy will start to cache the elements with smaller profit.

4.4 Efficiency Analysis

In this experiment, we discuss the efficiency of the related algorithms. To examine the efficiency of each algorithm, we measure the execution time for simulators to process 100000 client requests. Since we implement the algorithms by Java language and execute the pro-

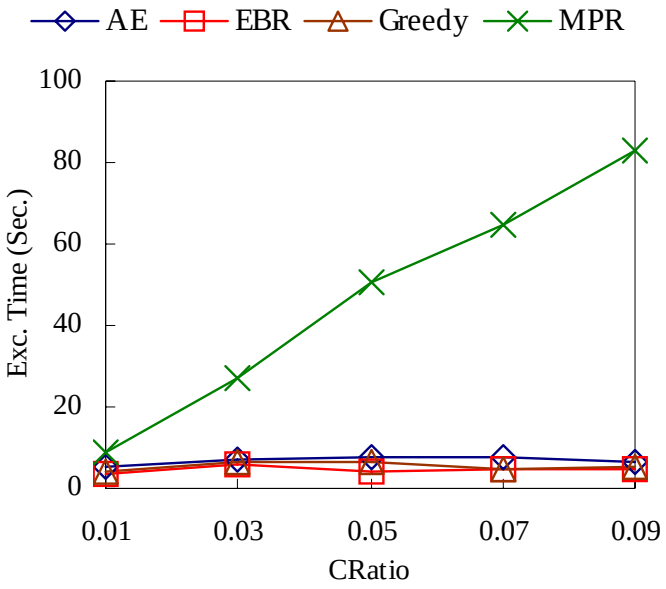

Figure 14: The execution time with $CRatio$ varied

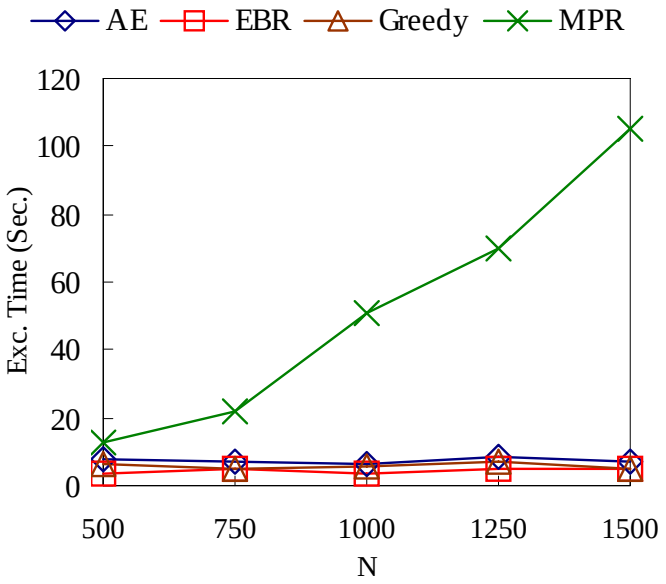

Figure 15: The execution time with the number of objects varied

grams on the same system platform, the execution time can reflect the efficiency. First, we investigate the execution time of the simulator under different capacity ratio $CRatio$. In Figure 14, we can observe that algorithm MPR requires much more execution time than the other three algorithms. Moreover, the execution time of algorithm MPR increases as the increase of $CRatio$. The reason is that the *caching candidate set* D_H is derived according to the concept of *dynamic programming*. In the situation of higher $CRatio$, algorithm MPR requires more time and memory space to derive D_H. Next, we vary the number of the objects N and investigate the execution time under the same capacity ratio and other parameters. As shown in Figure 15, the experimental result is very similar to that in Figure 14. The only difference is that the curve of algorithm MPR raises more drastically compared to the variance of $CRatio$. It is because like the cache capacity, the increase of N_{object} will also enlarge the required memory during the procedure of *dynamic programming*. Moreover, to maintain the service quality of the transcoding proxy, more cache space is required when the clients request for more objects. If we consider the efficiency of algorithm MPR, the effect of number of the objects is more noticeable than that of the capacity ratio. On the other hand, as for the proposed algorithm EBR, like algorithm AE and Greedy, the efficiency will not be affected by either the number of objects or the cache capacity. Therefore, algorithm EBR will be more efficient compared to algorithm MPR.

4.5 Remarks

According to the experimental results, to determine the suitable transcoding sources, since procedure MASP is based on the concept of shortest path, it is capable of achieving the optimal solution. Therefore, we suggest that MASP be employed to generate the minimum-cost transcoding graph from the complex transcoding relationship. Moreover, from the viewpoints of exact cache hit ratio, we observe that although algorithm MPR has the most outstanding performances in EHR, it requires more execution time and memory space to process user queries. When the number of the objects increases, algorithm MPR will suffer

from the efficiency issues. On the other hand, algorithms AE and Greedy are very scalable and insensitive to the variation of the efficiency parameters. However, they only result in fair quality of the effectiveness. As for the proposed EBR algorithm, it is as scalable as algorithm AE. Moreover, from the viewpoints of effectiveness, EBR outperforms AE prominently. Under certain circumstances such as high skewness, low diversity and low cache capacity, the quality of the EBR algorithm is very close to that of algorithm MPR. Therefore, the proposed RESP framework, which combines the MASP procedure and algorithm EBR, will be suitable and practical for a transcoding proxy to perform the cache replacement.

5 Conclusion

In this paper, we propose the RESP framework to perform cache replacement in a multimedia transcoding proxy. In RESP, two primary components, i.e., procedure MASP and algorithm EBR, are designed to maintain the inter-relationship and to perform cache replacement, respectively. The MASP procedure, which generates the minimum-cost transcoding graph by selecting the most suitable transcoding source in the shortest path table. On the other hand, algorithm MPR can approximate the optimal cache replacement very efficiently. The experimental results show that the RESP framework is effective in approximating the optimal solution with much lower complexity. Therefore, compared to the algorithms proposed in prior works, the proposed RESP framework will be suitable and practical for a transcoding proxy to perform the cache replacement.

References

[1] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox. Caching Proxies: Limitations and Potentials. In *Proceedings of 4th International World-Wide-Web Conference*, 1995.

[2] M. F. Arlitt, L. Cherkasova, J. Dilley, R. J. Friedrich, and T. Y. Jin. Evaluating Content Management Techniques for Web Proxy Caches. *ACM Sigmetrics Perform. Eval. Rev.*, 27(4), 2000.

[3] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[4] V. Cardellini, P.-S. Yu, and Y.-W. Huang. Collaborative Proxy System for Distributed Web Content Transcoding. In *ACM CIKM*, 2000.

[5] C.-Y. Chang and M.-S. Chen. On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies. *IEEE Transaction on Parallel and Distributed Systems*, 14(6), 2003.

[6] K. Cheng and Y. Kambayashi. A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Caching. In *Proceedings of the 24th International Computer Software and Applications Conference (COMPSAC'00)*, 2000.

[7] H. Chou and D. DeWitt. An Evaluation of buffer Management Strategies for Relational Database Systems. In *Proceedings of the 11th VLDB Conf.*, 1985.

[8] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[9] T. C. et al. Introduction to Algorithms. *McGraw Hill*.

[10] 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 Communication*, 5(6), 1998.

[11] H.-P. Hung and M.-S. Chen. Maximizing the profit for Cache Replacement in a Transcoding Proxy. In *Proceedings of IEEE ICME*, 2005.

[12] R. M. J. R. Smith and C.-S. Li. Content-Based Transcoding on Images in the Internet. In *Proceedings of IEEE Int'l Conf. on Image Processing*, 1998.

[13] K. Li, H. Shen, and K. Tajima. Cache Design for Transcoding Proxy Caching. In *IFIP International Conference on Network and Parallel Computing*, 2004.

[14] K. Li, H. Shen, K. Tajima, and L. Huang. An Effective Cache Replacement Algorithm in Transcoding-Enabled Proxies. *The Journal of Supercomputing*, 35(2), 2006.

[15] K. Li, K. Tajima, and H. Shen. Cache Replacement for Transcoding Proxy Caching. In *Proceedings of WI'05*, 2005.

[16] W. Y. Lum and F. C. M. Lau. A Context-Aware Decision Engine for Content Adaptation. *IEEE Pervasive Computing*, 1(3), 2002.

[17] J. M. Menaud, V. Issarny, and M. Banatre. Improving Effectiveness of Web Caching. *Recent Advances in Distributed Systems*, 1752, 2000.

[18] R. Mohan, J. R. Smith, and C. S. Li. Adapting Multimedia Internet Content for Universal Accesses. *IEEE Trans. on Multimedia*, 1(1), 1999.

[19] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. In *Proceedings of the Int. Conf. Management of Data*, 1993.

[20] V. Padmanabhan and L. Qiu. The Content and Access Dynamics of a Busy Web Site: Findings and Implications. In *Proceedings of ACM SIGCOMM*, 2000.

[21] K. Psounis and B. Prabhakar. A Randomized Web-Cache Replacement Scheme. In *Proceedings of the IEEE INFOCOM*, 2001.

[22] S. Podlipnig and L. Böszörenyi. A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys*, 35(4), 2003.

[23] B. Shen, S.-J. Lee, and S. Basu. Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks. *IEEE Trans. on Multimedia*, 6(2), 2004.

[24] D. Starobinski and D. Tse. Probabilistic Methods for Web Caching. *Perform. Eval.*, 46(2-3), 2001.

[25] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox. Removal Policies in Network Caches for World-Wide Web Documents. In *Proceedings of ACM SIGCOMM*, 1996.

[26] Q. Yang, H. H. Zhang, and H. Zhang. Taylor Series Prediction: A Cache Replacement Policy Based on Second-Order Trend Analysis. In *Proceedings of the 34th Hawaii International Conference on Systems Sciences*, 2001.

[27] K. Yeung and K. Ng. An Optimal Cache Replacement Algorithm for Internet Systems. In *22nd Annual Conference on Local Computer Networks*, 1997.

[28] J. Yoon, S. L. Min, and Y. Cho. Buffer Cache Management: Predicting the Future from the Past. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, 2002. I-SPAN '02. Proceedings*, 2002.

[29] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler. Using Smart Clients to Build Scalable Services. In *Proceedings of Winder 1997 USENIX Technical Conference*, 1997.

[30] J. Zhang, R. Izmailov, D. Reininger, and M. Ott. Web Caching Framework: Analytical Models and Beyound. In *Proceedings of IEEE Workshop on Internet Applications*, 1999.