Efficient Content Adaptation Based on Dynamic Programming

Truong Cong Thang and Yong Man Ro
Multimedia Group, Information and Communication University (ICU).

ABSTRACT

Content adaptation is an effective solution to support the quality of service for multimedia services over heterogeneous environments. This paper deals with the accuracy and the real-time requirement, two important issues in making decision on content adaptation. From our previous problem formulation, we propose an optimal algorithm and a fast approximation based on the Viterbi algorithm of dynamic programming. Through extensive experiments, we show that the proposed algorithms can enable accurate adaptation decisions, and especially they can support the real-time processing.

1. INTRODUCTION

In a universal multimedia access (UMA) system, content adaptation is an important method to provide the best possible presentation under constraints of various kinds of terminals and network connections available today [1]. Basically, content adaptation includes three major modules: decision engine, modality converter, and content scaler (Fig. 1). The decision engine analyzes the content description, user preferences, resource constraints and then makes optimal decision on modality conversion and content scaling. The modality converter and the content scaler include the specific converting and scaling operations to adapt, either offline or online, the content objects according to instructions from the decision engine. The functionality of the decision engine is essentially the same for both online and offline transcodings. It is important that the decision engine can both provide accurate solutions and support the real-time processing.

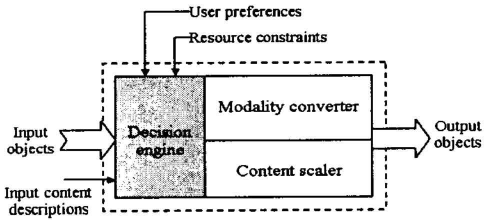

Fig. 1: Architecture of an adaptation engine

However, these two related issues of decision engine so far have been considered very little. In [2], the adaptation process was modeled as the resource allocation and then solved by the Lagrangian method. This method is supposed to have low complexity, however it is notorious for the inaccuracy with the non-concave content value (utility) function. In [3], several searching algorithms were proposed to find the appropriate adapted version of a single content object. These methods are said to be effective, yet no processing time have been reported.

In our previous work [1], we have proposed an approach that effectively supports the content adaptation under different constraints of terminals, networks, and user preferences. In this paper, we show that the Viterbi algorithm of dynamic programming and some approximations can be used in this framework to find the optimal solution and especially it is possible to support the real-time requirement of decision engine.

The paper is organized as follows. Section 2 reviews the problem formulation of content adaptation. In section 3, we propose the Viterbi-based algorithms to accurately solve the problem. Section 4 presents some experiment results, and finally section 5 concludes the paper.

2. PROBLEM FORMULATION

The decision-making process of the decision engine can be represented as the traditional resource allocation problem as follows [1]. Suppose we have a multimedia document consisting of multiple content objects. Let denote R_i and V_i the resource and content value of the content object i in the document. The content value V_i can be represented as a function of resource R_i, modality capability M, user preference P_i:

$$V_i = f_i(R_i, M_i, P_i). \qquad (1)$$

Then the problem of content adaptation for the given document is that:

Given a resource constraint R^c, find the set of $\{R_i\}$ so as

$$\sum_i V_i \text{ is maximum,} \qquad (2a)$$

$$\text{and} \quad \sum_i R_i \le R^c. \qquad (2b)$$

Regarding equation (1), we have proposed the *Overlapped Content Value (OCV) model* to represent the relationship between content value, modalities, and resource [1]. Each content object will be given an overlapped content value model (Fig. 2) representing the content values of different modalities versus the resource. The number of curves in the model is the number of modalities the content object may have. The final content value function will be the upper hull of the model. Each point on the function corresponds to a content version and is called a *selection*. Denote K_i as the number of modalities and VM_{ij} as the content value curve of modality j of the content object i, j=1...K_i. The content value of a content object can be mathematically represented as follows:

$$V_i = max\{w_{ij} \cdot VM_{ij}(R_i) \,|\, j=1...K_i\} \qquad (3)$$

where w_{ij} is the scale factor of modality j of object i and j=1 is the original modality of the object.

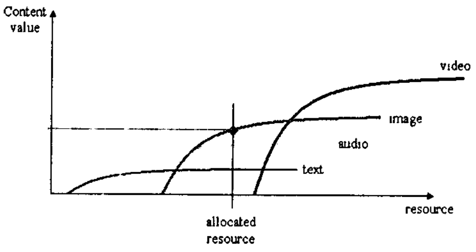

Fig. 2: Overlapped content value model of a content.

The OCV model is the underlying basis to support user preferences and to make accurate decisions on modality conversion and content scaling. The methods to incorporate user preferences into the content adaptation process have been proposed in [1]. Given the OCV model for each content object, a resource allocation method is then used to distribute the resource among multiple contents. Mapping the allocated resources back to content value models, we can find the appropriate qualities and modalities of adapted contents.

In the next section, we will present the possible solutions to this problem using Viterbi algorithm of dynamic programming.

3. SOLUTIONS BASED ON DYNAMIC PROGRAMMING

3.1. Optimal Solution by Viterbi Algorithm

A content value function can be continuous or discrete. If it is continuous, we may discretize it because the practical transcoding is done in the unit of bits or bytes; from now on we implicitly suppose it is discrete. Then a content value function will have a finite number of *selections*. Meanwhile, function (1) is inherently non-concave, thus the above optimization can be solved optimally by Viterbi algorithm of dynamic programming [4] [5].

The principle of Viterbi algorithm lies in building a trellis to represent all viable allocations at each instant, given all the predefined constraints. The basic terms used in Viterbi algorithm are defined as follows (Fig. 3):

- *Trellis*: The trellis is made of all surviving paths that link the initial node to the nodes in final stage.
- *Stage*: Each stage corresponds to an object to be adapted.
- *Node*: In our problem, each node is represented by a pair (i, a_i), where i=0...N is stage number, a_i is the accumulated resource of all objects until this stage.
- *Branch*: If selection k at stage i has the value-resource pair (V_{ik}, R_{ik}), then node $(i\text{-}1, a_{i-1})$ will be linked by a branch of value V_{ik} to node (i, a_i) with:

$$a_i = a_{i-1} + R_{ik}\ , \qquad (4)$$

satisfying (if not, the branch will not be linked):

$$a_i \le R_c\ . \qquad (5)$$

- *Path*: A path is a concatenation of branches. A path from the first stage to the final stage corresponds to a set of possible selections for all objects.

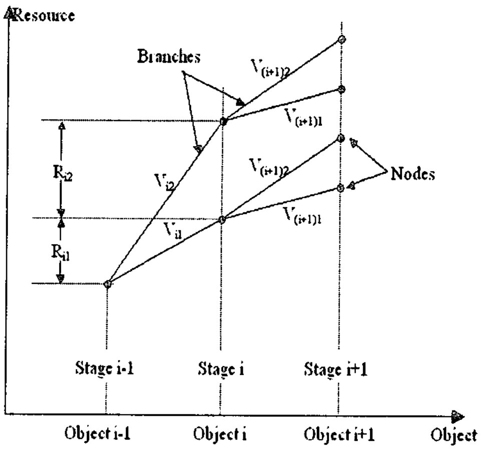

Fig. 3. Trellis diagram grown by Viterbi algorithm.

From the above, we can immediately see that the optimal path, corresponding to the optimal set of selections, is the one having the highest accumulated content value. We now apply Viterbi algorithm to generate the trellis and then to find the optimal path as follows [5] [6]:

Optimal algorithm:

Step 0*: Start from the initial node (0, 0)*

Step 1*: At each stage i, add possible branches to the end nodes of the surviving paths. At each node, a branch is grown for each of the available selections; the branch must satisfy condition (5).*

Step 2*: Of all the paths arriving at a node in stage i+1, the one having the highest accumulated content value is chosen, and the rest are pruned.*

Step 3*: Increase i and go to step 1.*

Step 4*: At the final stage, compare all surviving paths then select the path having the highest accumulated content value. That path corresponds to the optimal set of selections for all objects.*

In fact, the problem of resource allocation represented as a constrained optimization is often solved by two basic methods: Lagrangian method and dynamic programming method [4]. However, the Lagrangian method cannot provide accurate results when content value functions are non-concave [4]. The advantage of dynamic programming is that it can work with the non-concave content value functions, helping to find exactly the needed version. The disadvantage of dynamic programming is the high complexity. Yet, through practical considerations, we will show that the Viterbi algorithm is suitable for real-time computation of the decision engine.

3.2 Fast Approximation Algorithm

We see that the complexity of the above algorithm will decrease if the number of selections is reduced. For this purpose, one can intuitively omit or merge some neighboring selections. We note that in the searching process at each stage, if the current selection has a negligible content value change compared to the last un-omitted selection (called the last *considered selection*), this selection can be omitted. So, if we set a minimum *threshold* for the changes of the content values, we can reduce the number of selections considered for each object by omitting the selections having the content value changes within the threshold. Denote k^* the last considered selection, we modify the above step 1 to obtain a fast approximation algorithm as follows:

```
Step 1: At each stage i, add possible branches to the end nodes of surviving paths. At each node, do the following:
    Step 1.1: Add branch for selection 0
    Step 1.2: Check the selection k (k > 0): if |$V_{(i+1)k} - V_{(i+1)k^*}$|>threshold and if (5) is satisfied, add branch for selection k and let k* = k.
    Step 1.3: Increase k and go to step 1.2
```

Meanwhile other steps are unchanged. The modified step 1 constantly checks the content value changes and tries to connect a branch only if the content value difference between the current selection and the last considered one is greater than the threshold.

4. EXPERIMENTS

We have developed a test-bed for providing the multimedia services over heterogeneous networks. The adaptation engine is built on a Windows2000 server with Pentium IV 1.7GHz and 256MB RAM. In our system, the content transcoding is simply done offline. Scaling operations include reducing the spatial size for video and image modalities, reducing bandwidth for audio, and truncating the words for text. The resource constraint is datasize constraint D^c.

For experiments, we employ a multimedia document consisting of six objects: one video, one audio, three images, and one text paragraph. The original datasizes of the objects are respectively 1500KBs, 480KBs, 731KBs, 834KBs, 813KBs, and 8KBs. The modality curves are modeled by the following analytical function:

$$VM_{ij}(R_i) = x_{ij}(R_i - y_{ij})/(R_i - y_{ij} + z_{ij}) \quad (6)$$

where y_{ij} is the starting point, z_{ij} is the slope of the function, and x_{ij} is the saturate value of the function.

The highest possible content value for any objects is 10, yet the actual maximum content value of each object is assigned according to its relative importance, which is also obtained from subjective tests. The objects are transcoded in unit of kilobytes (KBs), so the content value functions can be discretized by the uniform step size of 1KBs. Also, all user preferences are set to be default [1].

To check the response of the adaptation system, we vary the datasize constraint D^c. Table I shows the document versions adapted to different values of D^c. In this Table, the first column is D^c; each object has two columns, one for the datasize and the other for the modality; the last column is the total content value of adapted document. Here, *Mod* means modality and *V, I, A, T* mean video, image, audio, text modalities respectively. We can see that as D^c decreases, the datasizes of the objects are reduced to satisfy the datasize constraint of the whole document. Also, at some points, the modalities of the objects are converted to meet the constraint and to give the highest possible total content value.

Now we check the performance, including the processing time and the optimality, of the decision engine. First we employ optimal algorithm for the decision engine. The continuous lines in Fig. 4(a) and Fig. 4(b) show respectively the total content value of the adapted document and the processing time (to find the optimal solution) versus the different values of D^c. We see that, using optimal algorithm (threshold=0), the processing time of the decision engine is smaller than 0.5s, which is acceptable to the real-time requirement. This result is actually due to the fact that there are only six objects in the document.

Next, to reduce the processing time, we try to apply the fast algorithm with modified step 1. The "good thresholds" are estimated by considering the content value and the processing time versus different thresholds, given some fixed values of the datasize constraint. The dashed lines in Fig. 4(a) and Fig. 4(b) show the performance of the decision engine using fast algorithm with threshold = 0.02. Now we can see that the processing time is much reduced, i.e. below 0.1s compared to 0.4s of optimal algorithm; meanwhile, total content value decreases very little.

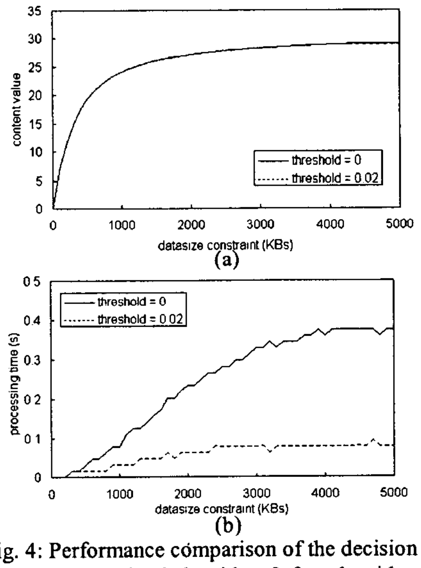

Fig. 4: Performance comparison of the decision engine using optimal algorithm & fast algorithm.

In above experiments, there are only six objects in the document. Now we add some more image objects to see how the processing time depends on the number of objects. The datasize of each added image objects is 900KBs. The datasize constraint is now set to be rather high, D^c=5000KBs. Fig. 5 shows the relationship of the processing time versus the number of objects for various cases. We have three cases: threshold = 0 (optimal algorithm), threshold = 0.02, and threshold = 0.09 (fast algorithm). We see that when the number of objects is more than 20, the processing time of the original algorithm is not quite good (more than 3s), but the result of the fast approximation algorithm is very interesting. With threshold=0.09, the processing time for the document of as many as 30 objects is still below 0.5s.

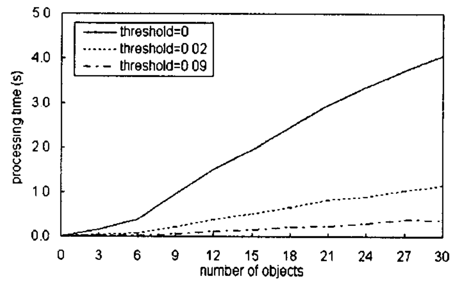

Fig. 5. Processing time vs. number of objects.

A special point is that the number of objects to be transcoded in a practical multimedia document is not many, normally not more than several tens. This is actually the main reason for the real time support of the decision engine. Moreover, there are some simple techniques to reduce the number of objects in a document, e.g. by analyzing the document or using the content description, we can discard some less important objects. In essence, the threshold technique aims at reducing the number of considered selections by omitting the "negligible" ones. In practice, the thresholds can be empirically estimated in advance and stored as metadata. The number of selections can be further reduced by various techniques. For example, the threshold may be adaptive to the slope of content value function, so as to remove more selections when possible.

5. CONCLUSION

In this paper, we have presented a systematic approach to tackle the content adaptation problem. The mechanism of the decision engine was formulated as a resource allocation problem. Then Viterbi algorithm of dynamic programming and an approximation were employed to optimally allocate resource to different content objects. The algorithms were shown to be efficient for accurately adapting multimedia contents to different constraint values. Furthermore, our experiments suggested that the Viterbi algorithm and its approximation could be well applicable for the real-time requirement of a decision engine. Our future works will focus on the automation of online transcoding, in terms of both content scaling and modality conversion. Also, metadata adaptation will be explored and deployed.

REFERENCE

[1]. T. C. Thang, Y. M. Ro, "Presentation Priority and Modality Conversion in MPEG-21 DIA", *Journal of Broadcasting Engineering*, Vol. 8, No. 4, pp.339-350, Dec. 2003.

[2]. R. Mohan, J. R. Smith, C.-S. Li, "Adapting Multimedia Internet Content for Universal Access", *IEEE Trans. Multimedia*, Vol. 1, No. 1, pp. 104-114, Mar. 1999.

[3]. W. Y. Lum and F. C. M. Lau, "A QoS-sensitive content adaptation system for mobile computing", *Computer Software and Application Conference*, pp. 680–685, 2002.

[4]. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, pp. 23-50, Nov. 1998.

[5]. G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[6]. A. Ortega, K. Ramchandran and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. Image Processing*, vol. 3, pp. 26-40, Jan. 1994.

Table I: Results of the adapted documents with different values of constraint

	Object 1		Object 2		Object 3		Object 4		Object 5		Object 6		Content
D^c (KBs)	D_1 (KBs)	Mod	D_2 (KBs)	Mod	D_3 (KBs)	Mod	D_4 (KBs)	Mod	D_5 (KBs)	Mod	D_6 (KBs)	Mod	value
3000	1041	V	260	A	544	I	571	I	576	I	8	T	28.30
1000	343	V	90	A	179	I	189	I	191	I	8	T	24.08
300	100	V	30	A	52	I	56	I	57	I	5	T	14.88
200	85	V	26	A	20	A	47	I	21	A	1	T	11.38
130	45	I	25	A	19	A	20	A	20	A	1	T	8.47
110	36	I	21	A	17	A	17	A	18	A	1	T	7.59
90	34	I	5	T	16	A	17	A	17	A	1	T	6.61
70	14	A	5	T	16	A	17	A	17	A	1	T	5.61
10	1	T	3	T	1	T	2	T	2	T	1	T	1.27