Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 953–965
Computer Networks
www.elsevier.com/locate/comnet

An optimization model for Web content adaptation ☆

Rong-Hong Jan [a,*], Ching-Peng Lin [a], Maw-Sheng Chern [b]

[a] Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC
[b] Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30043, Taiwan, ROC

Received 18 December 2003; received in revised form 7 June 2005; accepted 16 June 2005
Available online 8 August 2005

Responsible Editor R. Boutaba

Abstract

This paper considers Web content adaptation with a bandwidth constraint for server-based adaptive Web systems. The problem can be stated as follows: Given a Web page P consisting of n component items $d_1, d_2, \ldots, d_n$ and each of the component items d_i having J_i versions $d_{i_1}, d_{i_2}, \ldots, d_{i_{J_i}}$, for each component item d_i select one of its versions to compose the Web page such that the fidelity function is maximized subject to the bandwidth constraint. We formulate this problem as a linear multi-choice knapsack problem (LMCKP). This paper transforms the LMCKP into a knapsack problem (KP) and then presents a dynamic programming method to solve the KP. A numerical example illustrates this method and shows its effectiveness.

Keywords: Content transcoding; Knapsack problem; Dynamic programming

1. Introduction

Over the past decade, Internet use has exploded with people gaining rich information from the World Wide Web (WWW). With traditional wired-line Internet, users can only access the Internet in fixed places. Recently, however, due to the technology explosion in wireless communication and portable communication devices, e.g., cellular phones, personal digital assistants, and pagers, it

☆ This research was supported in part by the Communications Software Technology Project of Institute for Information Industry and in part by the National Science Council, Taiwan, ROC, under grant NSC 93-2219-E-009-002 and NSC 93-2752-E-009-005-PAE.
* Corresponding author. Tel.: +886 3 573 1637; fax: +886 3 572 1490.
E-mail address: rhjan@cis.nctu.edu.tw (R.-H. Jan).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2005.06.006

has become possible for people to connect to the Internet and remain on-line while roaming.

However, these portable communication devices are very different from the typical personal computers (PC). They vary widely in their screen size, resolution, color depth, computing power, and memory. From notebook PCs to cellular phones, the diversity of these devices makes it difficult and expensive to offer contents separately for each type of device. Many generic WWW servers lack the ability to adapt to the greatly varying bandwidths or to the heterogeneity of client devices. Therefore, the technologies that adapt the Web content to diverse portable communication devices will become very important in the future.

Many content adaptation technologies have been proposed for the WWW [1–10]. These adaptation methods can be divided into three categories: client-based, proxy-based and server-based adaptations. In client-based adaptations [7], the client transforms the original Web pages to the proper presentation according to its capability. However, this method does not work well for mobile devices because mobile devices have lower computing power. In proxy-based adaptations [4,8,9], the proxy intercepts the requested Web pages, performs the adaptation, and then sends the transformed content to the client. But this method requires huge calculations when transforming multi-media data. In contract, server-based adaptations [1,5,10] offer key advantages. Specifically, the server constructs Web pages in accordance to the users' device capabilities and network bandwidths. Repositing multi-versions of Web pages on Web servers in advance not only accelerates response time but also reduces network traffic.

In this paper, we consider a server-based adaptive Web system as shown Fig. 1. Clients can access the Internet via local area networks (LAN), wireless LAN, dial up, or GPRS networks. The Web server contains a set of multi-media Web pages. A multi-media Web page is composed of a number of component items. The clients browse Web pages by sending http requests with capability and preference information [11–13] to the Web server. The Web server parses the requests to learn the capabilities of the clients and probes the network to determine the bandwidth of the connection. Based on clients' capabilities and the bandwidth of the connection, the Web server generate an optimal version of the requested Web page and returns it to the clients.

This paper studies how to generate an optimal version of a Web page with a bandwidth constraint for the server-based adaptive Web system. Formally, the problem, denoted as a Web content selection problem, can be stated as follows: Given a Web page P consisting of n component items

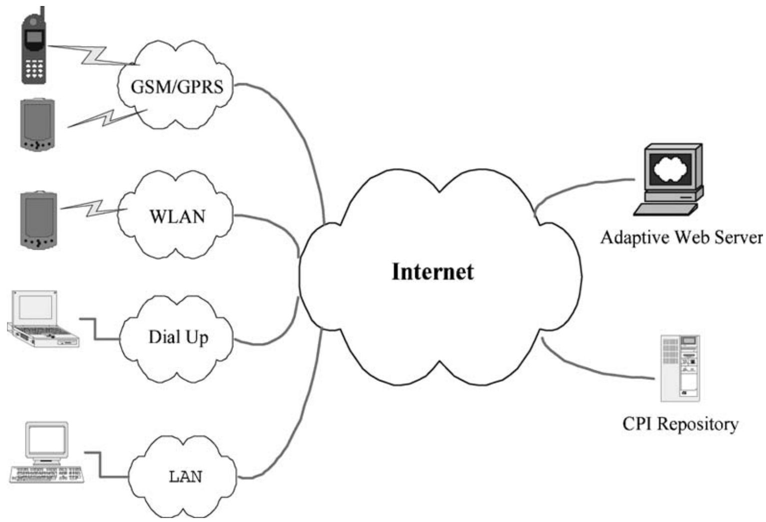


Fig. 1. An adaptive Web system architecture.

$d_1, d_2, \ldots, d_n$ and each of the component items d_i having J_i versions $d_{i_1}, d_{i_2}, \ldots, d_{i_{J_i}}$, for each component item d_i select one of the versions to compose the Web page such that the fidelity function is maximized subject to the bandwidth constraint. We formulate the Web content selection problem as a linear multi-choice knapsack problem (LMCKP) [14]. This paper transforms the LMCKP into a 0/1 knapsack problem (KP)[15,16]. The 0/1 KP problem is a well-known problem in combinatorial optimization. The problem has a large range of applications: capital budgeting, cargo loading, cutting stock, and so on. It can be solved by dynamic programming [17,18], branch and bound [19–21], and greedy methods. This paper presents a dynamic programming method for solving the 0/1 KP because dynamic programming can be easily extended to solve parametric LMCKP problem with different resources. This avoids having to solve the problem anew and slashes the computations needed.

The remainder of this paper is organized as follows. In Section 2, we formulate the Web content selection problem as an optimization problem. Section 3 discusses the solution method, and experimental results are given in Section 4.

2. Statement of the problem

Consider an adaptive Web server having three major modules: content analysis and transcoding, capability and preference information (CPI) filter, and content selection. The architecture of the adaptive server is based on [1]. Fig. 2 illustrates the content adapting process in the adaptive server. In the content analysis and transcoding module, the Web contents are analyzed and transformed into different versions. They are then organized into a content pyramid. The content is prepared in XML, which is converted to HTML prior to delivery. If the server receives an http

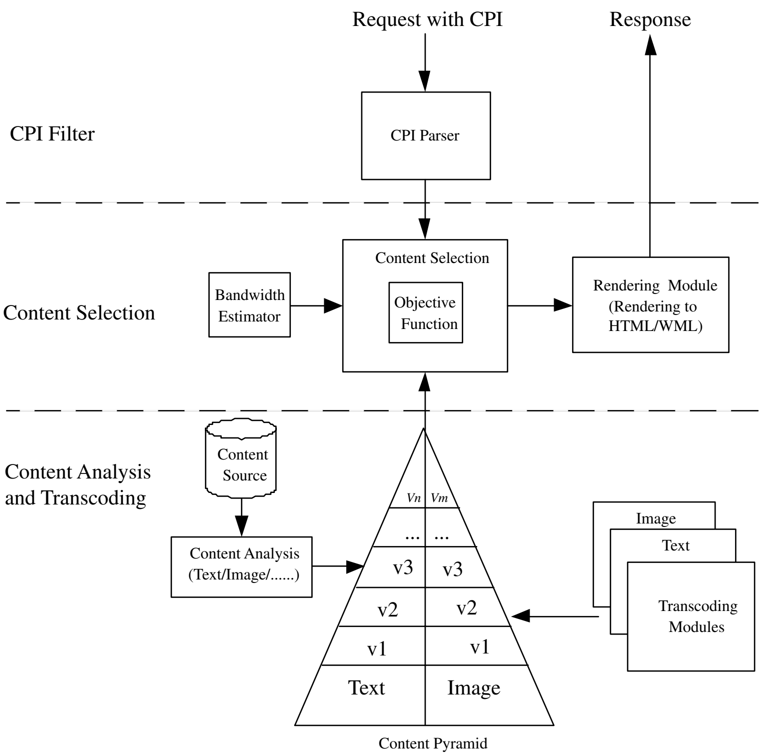


Fig. 2. Server-based content adaptation system architecture.

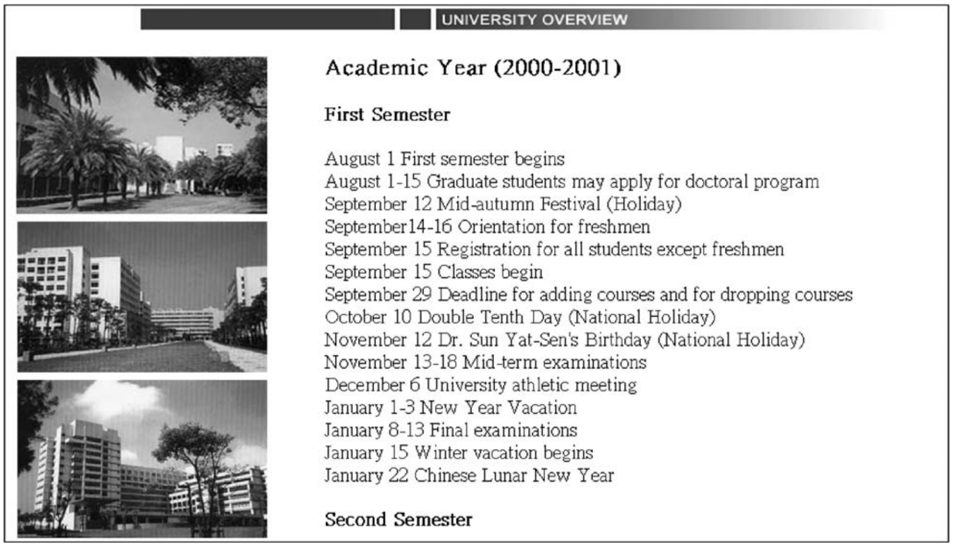


Fig. 3. An example of multi-media Web page.

request from a client, the CPI filter module processes the capabilities of the request and forwards the results to the content selection module. The content selection module selects a set of feasible versions from the content pyramid and calls on the bandwidth probing engine [22,23] to find the bottleneck bandwidth between the client and server. With the client's capabilities and bandwidth information, the content selection module determines an appropriate version for each component item. Based on the appropriate versions, the rendering module tailors a style sheet represented by XML style-sheet language (SML), generates an adaptive content and replies to the client.

Note that a multi-media Web page is composed of a number of component items. For example, the document shown in Fig. 3 consists of five component items. These include four image component items and one text component item. Usually, the image component item can be described at multiple resolutions, called versions. The versions can be transformed from raw data at different resolutions. The different version of the component item has a different data size. Suppose a multi-media Web page, P, consists of a number of component items d_i where $P = d_1, d_2, \ldots, d_n$. A component item d_i can be computed by transcoding into versions, $d_{i1}, d_{i2}, \ldots, d_{iJ_i}$ with different resolutions and modalities. Let w_{ij} be the data size of version d_{ij}.

For each version d_{ij}, we can assign a measure of fidelity, called value v_{ij}. Value v_{ij} can be defined as follows:

$$v_{ij} = \frac{\text{perceived value of transcoded version } d_{ij}}{\text{perceived value of original } d_{i1}},$$

where $0 \leqslant v_{ij} \leqslant 1$.

With value v_{ij}, we can then compare different component items that are in different versions. The perceived value may either be assigned by the author for each version, or determined by a function of data size. In this paper, we assume $v_{ij} = f(w_{ij})$ that captures the general trend of fidelity in value. $f(w_{ij})$ may be a concave, convex/non-concave or discrete function of w_{ij}. In this paper, we define[1]

$$f(w_{ij}) = \sqrt{\frac{w_{ij}}{w_{i1}}},$$

where w_{i1} is the data size of item d_i with the original version (see Fig. 4). However, the Web content

[1] This paper is not to suggest that there actually exists a simple function for assigning values to v_{ij}. This is because measuring perceived quality of an image is not easy. Our optimization model allows one to assign arbitrary value to v_{ij} for Web content adaptation problem, by assuming $f(w_{ij})$.

Image							
Version	V1	V2	V3	V4	V5	V6	V7
Data Size (Kbits)	21.3	14.1	11.4	9.0	6.9	4.7	3.1
Value Sqrt(Vn/V1)	1.00	0.81	0.73	0.65	0.57	0.47	0.38

Fig. 4. An example of versions for an image item.

creator can define his own $f(w_{ij})$, say $f(w_{ij}) = w_{ij}/w_{i1}$ or $f(w_{ij}) = \ln w_{ij}/\ln w_{i1}$.

Thus, the Web server can be designed to select the best versions of content items from the Web document sets to meet the client resources while delivering the largest total value of fidelity. Usually, clients do not have the patience to wait for a long time for a Web page. One may expect to receive a Web page in a reasonable waiting time T_{total}, say 15 s. The next problem for the Web server is to determine the data size W (maximum) for transmission so as to fall within the expected waiting time.

Fig. 5 illustrates the browsing procedure. The total waiting time for the user is

$$T_{\text{total}} = T_{\text{prop}} + T_{\text{probe}} + T_{\text{proc}} + T_{\text{trans}} + T_{\text{prop}},$$

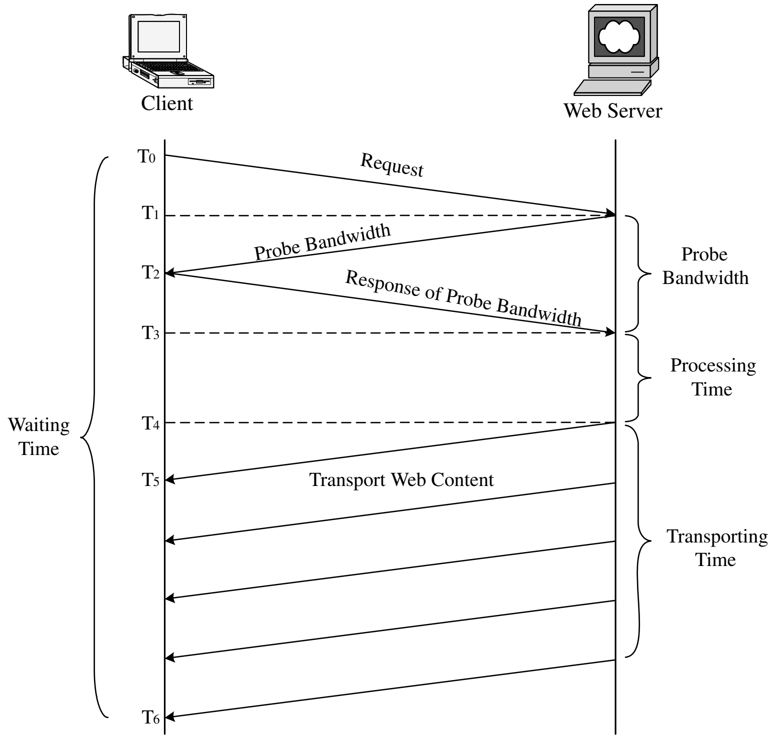


Fig. 5. Event timing for browsing an adaptive Web page.

where T_{total} = total time to wait for each Web page; T_{prop} = propagation time = $T_1 - T_0 = T_5 - T_4$; T_{probe} = time to probe bandwidth = $T_3 - T_1$; T_{proc} = time to process Web content selection = $T_4 - T_3$; T_{trans} = time to transmit Web content = $T_6 - T_5$.

Here we assume for simplicity that T_{prop}, T_{probe}, and T_{proc} are constants. Then data size $W = b \times t = b \times (T_{\text{total}} - 2T_{\text{prop}} - T_{\text{probe}} - T_{\text{proc}})$ where b is the bottleneck bandwidth and $t = T_{\text{trans}}$. For example, if $T_{\text{total}} = 15$, $2T_{\text{prop}} + T_{\text{probe}} + T_{\text{proc}} = 4$, and $b = 10$ Kbps, then the Web server will send a Web page with size not greater than $W = (15 - 4) \times 10 = 110$ KB.

Therefore, the Web content adaptation can be mathematically stated as follows.

Problem LMCKP:

$$\text{Maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{Ji} v_{ij} x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} \sum_{j=1}^{Ji} w_{ij} x_{ij} \leqslant W, \tag{2}$$

$$\sum_{j=1}^{Ji} x_{ij} = 1, \quad 1 \leqslant i \leqslant n, \tag{3}$$

$$x_{ij} = 0 \text{ or } 1, \quad \text{for all } i, j,$$

where v_{ij} and w_{ij} are the measures of fidelity and data size of version d_{ij}, respectively. W is the maximum payload. x_{ij} is the decision variable where $x_{ij} = 1$ indicates version j is selected for item i; otherwise, $x_{ij} = 0$. Constraint (2) ensures that the size of the Web page is not greater than Web page $W = b \times t$. Constraint (3) limits our choice for each item to be one of its versions.

Note that this problem is known as the linear multiple choice knapsack problem (LMCKP). We can apply the dynamic programming method to find the optimal solution for problem LMCKP. An appropriate content can be determined by solving the LMCKP problem.

3. The solution method

The LMCKP is a well-known problem. Many solution methods that have been presented for solving it. This section transforms the LMCKP into a 0/1 KP and apply the dynamic programming method to solve the 0/1 KP.

3.1. Transformation of the problem

At first, we define a knapsack problem, which is equivalent to LMCKP, as follows.

For each i, let

$$\begin{aligned} y_{i1} &= x_{i1}, \\ y_{i2} &= x_{i1} + x_{i2}, \\ &\cdots \\ y_{iJ_i-1} &= x_{i1} + x_{i2} + \cdots + x_{iJ_i-1}, \\ y_{iJ_i} &= x_{i1} + x_{i2} + \cdots + x_{iJ_i} = 1. \end{aligned}$$

Then, we can rewrite the objective function (1) as in the following:

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{Ji} v_{ij} x_{ij} &= \sum_{i=1}^{n} v_{i1} y_{i1} + v_{i2}(y_{i2} - y_{i1}) + \cdots \\ &\quad + v_{iJ_i-1}(y_{iJ_i-1} - y_{iJ_i-2}) + v_{iJ_i}(y_{iJ_i} - y_{iJ_i-1}) \\ &= \sum_{i=1}^{n} (v_{i1} - v_{i2}) y_{i1} + (v_{i2} - v_{i3}) y_{i2} + \cdots \\ &\quad + (v_{iJ_i-1} - v_{iJ_i}) y_{iJ_i-1} + v_{iJ_i} y_{iJ_i} \\ &= \sum_{i=1}^{n} \left[\sum_{j=1}^{J_i-1} (v_{ij} - v_{ij+1}) y_{ij} + v_{iJ_i} y_{iJ_i} \right] \\ &= \sum_{i=1}^{n} \sum_{j=1}^{J_i-1} (v_{ij} - v_{ij+1}) y_{ij} + \sum_{i=1}^{n} v_{iJ_i}. \end{aligned}$$

Similarly, the constraint (2) can be rewritten as

$$\sum_{i=1}^{n} \sum_{j=1}^{J_i-1} (w_{ij} - w_{ij+1}) y_{ij} + \sum_{i=1}^{n} w_{iJ_i} \leqslant W.$$

Note that $\sum_{i=1}^{n} v_{iJ_i}$ and $\sum_{i=1}^{n} w_{iJ_i}$ are constants. Let $e_{ij} = v_{ij} - v_{ij+1}$, $d_{ij} = w_{ij} - w_{ij+1}$ and $W' = W - \sum_{i=1}^{n} w_{iJ_i}$. Then, the problem LMCKP (Eqs. (1)–(3)) is equivalent to the following KP:

$$\text{Maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{J_i-1} e_{ij} y_{ij} \tag{4}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} \sum_{j=1}^{J_i-1} d_{ij} y_{ij} \leqslant W', \tag{5}$$

$$y_{ij} = 0 \text{ or } 1, \quad y_{i1} \leqslant \cdots \leqslant y_{iJ_i}, \ 1 \leqslant i \leqslant n, \quad 1 \leqslant j \leqslant J_i - 1. \tag{6}$$

Note that the above problem can be rewritten as a precedence constraint 0/1 KP [24,25] as follows.

Problem KP:

$$\text{Maximize} \quad \sum_{i=1}^{m} p_i z_i \tag{7}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} d_i z_i \leqslant M, \tag{8}$$

$$z_i = 0 \text{ or } 1, \quad z_h \leqslant z_k, \ (h,k) \in A_i,$$
$$1 \leqslant i \leqslant m,$$

where $m = \sum_{i=1}^{n} \sum_{j=1}^{J_i - 1} 1; M = W'$ and A_i is the precedence constraint as described in (6).

3.2. Dynamic programming method

The precedence constraint 0/1 knapsack problem can be solved by dynamic programming method as that for the ordinary 0/1 knapsack problem with slight modification. We may make a decision on z_1 first, then on z_2, then on z_3, etc. The solution to the 0/1 KP problem can be viewed as the result of a sequence of decisions. An optimal sequence of $z_1, z_2, \ldots, z_k$ will maximize the objective function and satisfy the constraint. Moreover, we can apply dynamic programming to solve the *parametric precedence constraint* 0/1 KP problem with right-hand side $M \in [a, b]$.

Let KP(j, S) denote the problem

$$\text{Maximize} \quad \sum_{i=1}^{j} p_i z_i$$

$$\text{Subject to} \quad \sum_{i=1}^{j} d_i z_i \leqslant S,$$

$$z_i = 0 \text{ or } 1, \quad z_h \leqslant z_k, \ (h,k) \in A_i,$$
$$1 \leqslant i \leqslant j,$$

where $1 \leqslant j \leqslant n$ and $0 \leqslant S \leqslant M$. Note that KP($j$, S) is a sub-problem of Problem KP with variables $z_1, z_2, \ldots, z_j$ and right-hand side S. Problem KP is KP(n, M). Let $f_k(s)$ be the value of an optimal solution to KP(k, s). From the principle of optimality it follows that:

$$f_k(s) = \max\{f_{k-1}(s), f_{k-1}(s - d_k) + p_k, \text{ subject to precedence constraints}\}. \tag{9}$$

Clearly, $f_n(M)$ is the value of an optimal solution to KP(n, M). $f_n(M)$ can be solved by beginning with $f_0(s) = 0$ for all $s > 0$ and $f_0(s) = -\infty$, $s < 0$. Then $f_1, f_2, \ldots, f_n$ can be successively computed using Eq. (9). Notice that $f_k(s)$ is an ascending step function; i.e., there are a finite number of s, $s_1 < s_2 < \cdots < s_t$, such that $f_k(s_1) \leqslant f_k(s_2) \leqslant \cdots \leqslant f_k(s_t)$. For the parametric precedence constraint 0/1 KP problem, we solved the problem KP(n, M) for each $M \in [a, b]$ at the last stage.

3.3. A numerical example

Consider an example of a Web page with three image items (i.e., $P = d_1, d_2, d_3$). Each item has three versions. The right-hand side $W \in [6.7, 30.8]$. The data sizes w_{ij} and the values v_{ij}, $i, j = 1, 2, 3$, are

$$[w_{ij}] = \begin{bmatrix} 9.0 & 4.4 & 1.3 \\ 10.8 & 4.6 & 1.0 \\ 11.0 & 5.4 & 1.3 \end{bmatrix},$$

$$[v_{ij}] = \left[\sqrt{\frac{w_{ij}}{w_{i1}}}\right] = \begin{bmatrix} 1.0 & 0.7 & 0.4 \\ 1.0 & 0.7 & 0.3 \\ 1.0 & 0.7 & 0.3 \end{bmatrix}.$$

The content selection problem can be formulated as follows:

$$\text{Maximize} \quad v_0 = \sum_{i=1}^{3} \sum_{j=1}^{3} v_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij} x_{ij} \leqslant W, \quad W \in [6.7, 30.8],$$

$$\sum_{j=1}^{3} x_{ij} = 1, \quad 1 \leqslant i \leqslant 3,$$

$$x_{ij} = 0 \text{ or } 1, \quad 1 \leqslant i \leqslant 3, \ 1 \leqslant j \leqslant 3.$$

3.3.1. Transformation of the problem

For $i = 1, 2, 3$, let $y_{i1} = x_{i1}$, $y_{i2} = x_{i1} + x_{i2}$, and $y_{i3} = x_{i1} + x_{i2} + x_{i3} = 1$. Then, the problem can be transformed as follows:

$$\begin{aligned}\text{Maximize}\quad & 0.3y_{11}+0.3y_{12}+0.3y_{21}+0.4y_{22}\\ & +0.3y_{31}+0.4y_{32}+1\\ \text{Subject to}\quad & 4.6y_{11}+3.1y_{12}+6.2y_{21}+3.6y_{22}\\ & +5.6y_{31}+4.1y_{32}+3.6\leqslant W,\\ & W\in[6.7,30.8],\\ & y_{11}\leqslant y_{12},\ y_{21}\leqslant y_{22},\ y_{31}\leqslant y_{32};\\ & y_{11},y_{12},y_{21},y_{22},y_{31},y_{32}=0 \text{ or } 1.\end{aligned}$$

Let $z_1 = y_{11}$, $z_2 = y_{12}$, $z_3 = y_{21}$, $z_4 = y_{22}$, $z_5 = y_{31}$, $z_6 = y_{32}$. Clearly, the above problem is equivalent to the following problem, KP(6, M), $M \in [3.1, 27.2]$:

$$\begin{aligned}\text{Maximize}\quad & 0.3z_1+0.3z_2+0.3z_3+0.4z_4\\ & +0.3z_5+0.4z_6\\ \text{Subject to}\quad & 4.6z_1+3.1z_2+6.2z_3+3.6z_4+5.6z_5\\ & +4.1z_6\leqslant M,\quad M\in[3.1,27.2],\\ & z_1\leqslant z_2,\ z_3\leqslant z_4,\ z_5\leqslant z_6;\\ & z_i=0 \text{ or } 1,\ i=1,\ldots,6.\end{aligned}$$

3.3.2. Dynamic programming method

Let $f_k(s)$ be the value of an optimal solution to KP(k, s) where $s \in [0, 27.2]$ and $i = 1,\ldots,6$. Clearly, $f_6(M)$ is the value of an optimal solution to KP(6, M). Applying Eq. (9), $f_6(M)$, $M \in [a, b]$ can be solved by starting with $f_0(s) = 0$ for all $s > 0$. Then $f_1(s), f_2(s),\ldots,f_6(s)$, $s \in [0, 27.2]$ can be successively found. For example

$$f_4(s)=\max_{z_4=0,1}\begin{cases}0.4+f_3(s-3.6), & z_4=1,\quad s\geqslant 3.6,\\ f_2(s-3.6), & z_4=0.\end{cases}$$

Fig. 6 graphically shows $f_1(s), f_2(s),\ldots,f_5(s)$, $s \in [0, 27.2]$, and $f_6(s)$, $s \in [3.1, 27.2]$. Thus, the optimal values and the optimal solutions for $M \in [3.1, 27.2]$ is summarized in Table 1.

For example, the optimal solution to KP(6, 27.2) is $f_6(27.2) = 2.0$. The optimal solution of KP(6, 27.2) is: $(z_1, z_2, z_3, z_4, z_5, z_6) = (y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}) = (1, 1, 1, 1, 1, 1)$. Thus, the

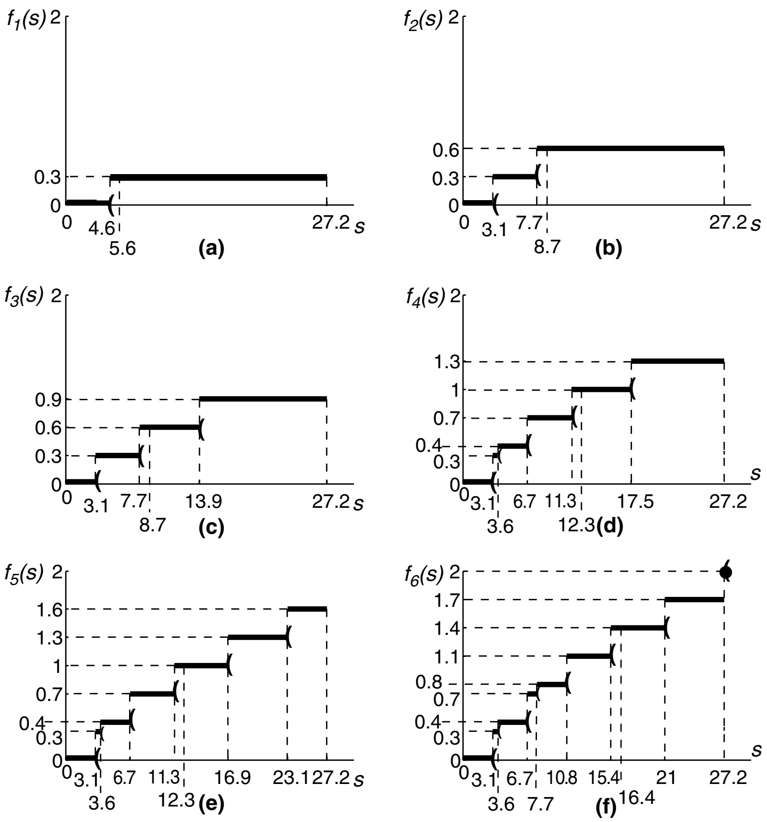


Fig. 6. Functions of $f_1(s), f_2(s), f_3(s), f_4(s), f_5(s)$, and $f_6(s)$.

Table 1
Summary of the optimal solutions

Right-hand side	$f_6(M)$	$(y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32})$	v_0	$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33})$
$M \in [3.1, 3.6)$	0.3	(0, 1, 0, 0, 0, 0)	1.3	(0, 1, 0, 0, 0, 1, 0, 0, 1)
$M \in [3.6, 6.7)$	0.4	(0, 0, 0, 1, 0, 0)	1.4	(0, 0, 1, 0, 1, 0, 0, 0, 1)
$M \in [6.7, 7.7)$	0.7	(0, 1, 0, 1, 0, 0)	1.7	(0, 1, 0, 0, 1, 0, 0, 0, 1)
$M \in [7.7, 10.8)$	0.8	(0, 0, 0, 1, 0, 1)	1.8	(0, 0, 1, 0, 1, 0, 0, 0, 1)
$M \in [10.8, 15.4)$	1.1	(0, 1, 0, 1, 0, 1)	2.1	(0, 1, 0, 0, 1, 0, 0, 1, 0)
$M \in [15.4, 21.0)$	1.4	(1, 1, 0, 1, 0, 1)	2.4	(1, 0, 0, 0, 1, 0, 0, 1, 0)
$M \in [21.0, 27.2)$	1.7	(1, 1, 0, 1, 1, 1)	2.7	(1, 0, 0, 0, 1, 0, 1, 0, 0)
$M = 27.2$	2	(1, 1, 1, 1, 1, 1)	3	(1, 0, 0, 1, 0, 0, 1, 0, 0)

optimal solution for the content selection problem is

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (1, 0, 0, 1, 0, 0, 1, 0, 0)$$

and the optimal value is 3.0. That is, version 1 is selected for each item.

If another request for this page arrives, the adaptive server finds $b = 16$ Kbps and $t = 5$ s for this connection. Then, the total data size W that the adaptive server may return to the client is

$$W = \frac{16 \times 10}{8} = 20 \text{ Kbps}.$$

Note that the adaptive server does not need to solve the problem KP(6, 20 − 3.6) anew. The optimal solution for KP(6, 16.4) can be found in Table 1. Since $M = 16.4$, we look in Table 1 down the $M \in [15.4, 21.0)$ row. We find that the optimal solution for KP(6, 16.4) is $(y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}) = (1, 1, 0, 1, 0, 1)$, and the optimal solution for the content selection problem is

$$(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}) = (1, 0, 0, 0, 1, 0, 0, 1, 0).$$

That is, the returned Web page is compose of version 1 for item 1 and version 2 for items 2 and 3.

4. Experimental results

In order to test our optimization model for Web content adaptation, we built three Web servers: two adaptive and one non-adaptive servers. Both adaptive Web servers consisted of three major modules (content analysis and transcoding, CPI filter, and content selection) as shown in Fig. 2. The difference between them is in the content selection module. One, denoted as Sever 1, found optimal solutions of parametric LMCKP by dynamic programming method in advance after the Web page was created. When the request arrives, it just looks up the optimal solutions table. The other, denoted as Sever 2, selects contents by using greedy algorithm [14] to solve LMCKP whenever the request arrives. The non-adaptive server is denoted as Sever 3. A Linux operating system and an Apache server were selected as developing platform for three servers. Apache is a well-known, open source Web server that performs well. The machines for the three servers are the desktop computers with AMD K7-850 and 256 MB memory. The test Web page, a sub-page of the University's Web pages, consists of seven component items with 140 KB data size. These include six image component items and one text item. Each image item has six versions.

The Servers 1, 2 and 3 were tested by two clients. The clients' browser was modified from Internet Explorer (IE) so that the users can specify the expected time, CPI data, and where the CPI profiles are by URLs (see Fig. 7). Client 1 was a notebook PC with Intel P3-650 and 256MB memory, using PPP-dialup 56 Kbps (campus dialup service) connecting to the campus Internet. The expected waiting time was set to 15 s when browsing the Servers 1 and 2. For the measurement, the system clock of the three servers and two clients was synchronized using the Network Time Protocol (NTP). Client 1 browsed the Web page 10 times. We use the t distribution with 9 degrees of

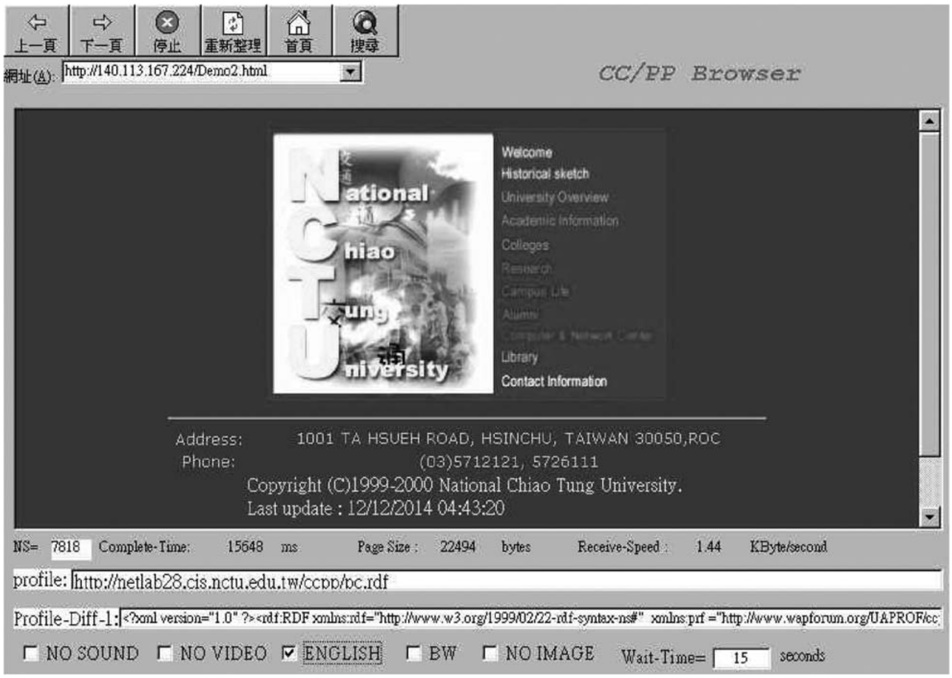


Fig. 7. An example of CC/PP browser.

freedom and a 95% confidence interval to estimate the delays. The average delays for Client 1 are shown in Table 2a. Table 2b shows the percentage of measured delays out of the total delay. The other client, Client 2 was a Compaq pocket PC using IEEE 802.11b wireless LAN connecting to campus Internet. The results are summarized in Tables 3a and 3b. Our experiments use campus Internet as network testbed. There are many factors, such as irrelevant traffic in the network, buffer sizes, etc., that may influence (or pollute) the results. Tables 2 and 3 are only intended to offer the reader some realistic feeling about the respond time and how the system works. From the theoretical point of view, the time complexity for Server 1 to pick up an optimal solution from the optimal

Table 2a
Results for notebook PC with 56K dialup

	$2 \times T_{prop}$ (ms)	T_{probe} (ms)	T_{proc} (ms)	T_{trans} (ms)	T_{total} (ms)	W (KB)
Server 1	179.5 ± 2.0	6000.1 ± 2.0	<0.001	9614.3 ± 20.2	15794 ± 20	28.9
Server 2	173.5 ± 2.8	6123.3 ± 1.8	4.8 ± 0.3	10141.5 ± 23.4	16443 ± 22	30.0
Server 3	180.2 ± 2.0	–	–	46205.4 ± 116.9	46386 ± 117	140

Table 2b
Percentage of the measured time out of T_{total} for Table 2a

	$2 \times T_{prop}/T_{total}$ (%)	T_{probe}/T_{total} (%)	T_{proc}/T_{total} (%)	T_{trans}/T_{total} (%)
Server 1	1.14	37.99	0.00	60.87
Server 2	1.05	37.24	0.03	61.68
Server 3	0.39	0.00	0.00	99.61

Table 3a
Results for pocket PC with 802.11b

	$2 \times T_{prop}$ (ms)	T_{probe} (ms)	T_{proc} (ms)	T_{trans} (ms)	T_{total} (ms)	W (KB)
Server 1	5.8 ± 0.5	2028.3 ± 23.6	<0.001	118.8 ± 1.9	2153 ± 24	28.9
Server 2	6.5 ± 0.4	2212.6 ± 16.7	4.4 ± 0.4	109.7 ± 1.8	2333 ± 17	28.9
Server 3	6.4 ± 0.4	–	–	641.9 ± 6.8	648 ± 7	140

Table 3b
Percentage of the measured time out of T_{total} for Table 3a

	$2 \times T_{prop}/T_{total}$ (%)	T_{probe}/T_{total} (%)	T_{proc}/T_{total} (%)	T_{trans}/T_{total} (%)
Server 1	0.27	94.21	0.00	5.52
Server 2	0.28	94.83	0.19	4.70
Server 3	0.99	0.00	0.00	99.06

solutions table is O(1) while the time complexity of the greedy method for Server 2 is $O(n \log n)$ where n is the number of variables.

From Table 2a, note that the total delays for browsing the Servers 1 and 2 were 15794 ± 20 ms and 16443 ± 22 ms, while for browsing the Server 3 it was 46386 ± 117 ms. The adaptive servers show their benefits. The total delays of Servers 1 and 2 are controlled and close to the expected waiting time 15 s. For processing time T_{proc}, Server 1 performs better than Server 2 because Server 1 finds optimal solutions by dynamic programming method in advance and looks up the optimal solutions table when the request arrives. The advantage of dynamic programming method is that it can be easily applied to solve the parametric LMCKP with a set of right-hand sides.

In Tables 2a and 3a, the values of probing delay T_{probe} are very large. This is because the probing method ("pathchar" algorithm [23]) sends a few dozen packets with varying sizes, measures their round trip times (RTTs), and then finds the available bandwidth by correlating the RTTs with packet sizes. The RTT depends on traffic load. Thus, the probing delay T_{probe} varies as traffic load varies. The traffic load in our campus Internet varies dynamically from one instant to another. Consequently, the mean probing time may be different for the same mobile device in the case of Server 1 and Server 2.

Note that if the access network has a higher data rate, the value of T_{probe} dominates the value of T_{trans} (see Table 3b). That is, the overhead of bandwidth estimation is too large, thus negating the advantage of shorter transmission times. In fact, instead of measuring bandwidth, we can just use predefined classes of data rate r, (say dialup 54 Kbps, T1 1.544 Mbps, or WiFi 11 Mbps) and set available bandwidth $b = \alpha \times r$, $0 < \alpha < 1$. Note that data rate r can be obtained from the CPI profile. By this way, the overhead of bandwidth estimation is eliminated and thus the adaptive Web server can give a smaller latency.

5. Conclusion

This paper formulates a Web content adaptation problem as a linear multi-choice knapsack problem and presents a dynamic programming method to solve it. We think that the dynamic programming is very suitable for solving this kind of problems because dynamic programming can be easily extended to solve parametric LMCKP problem with different resources. This avoids having to solve the problem anew and slashes the computations needed.

In practical implementation, we can analyze and transform the component items of Web page into different versions when the Web page is created. Then, dynamic programming is applied to solve a parametric LMCKP problem and a binding table which binds the optimal solutions to different resources can be created. If a request for this page arrives, we just look up the binding table to find the optimal versions of the component items for the request.

In this paper, we assumed that the content items are independent of each other. However, this assumption may not hold in some cases. Consider a news story page. If the story has to be discarded due to space limitations, then the pictures for the story has also to be discarded. For such cases, the content creator has to define the dependencies among the items of the Web page. Then, our optimization model can be extended by adding the constraint $\sum_{j=1}^{Ji} x_{ij} \leqslant \sum_{j=1}^{Jk} x_{kj}$ to Problem LMCKP if item d_i is dependent on item d_k.

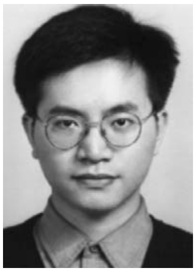
There also exists coarse-grained approaches for content adaptation. The coarse-grained approaches format the Web content for several well-known kinds of clients to suit everyone. However, we think delivering a customized content is worth for content providers. This paper presents a fine-grained adaptation that selects the best content representation to match the resources and capabilities of individual clients.

Looking ahead, integrating both adaptive Web server and transcoding proxy server for wireless Internet access might be interesting future work.

References

[1] R. Mohan, J.R. Smith, C.S. Li, Adapting multimedia Internet content for universal access, IEEE Transactions on Multimedia (March) (1999) 104–114.
[2] V. Cardellini, P.S. Yu, Y.W. Huang, Collaborative proxy system for distributed Web content transcoding, in: Proceedings of the 9th International ACM Conference on Information and Knowledge Management, November 2000, pp. 520–527.
[3] R. Han, V. Perret, M. Naghshineh, WebSplitter: a unified XML framework for multi-device collaborative Web browsing, in: ACM Conference on Computer Supported Cooperative Work (CSCW), December 2000.
[4] P.A. Singh, A. Trivedi, K. Ramamritham, PTC: proxies that transcode and cache in heterogeneous Web client environments, Web Information Systems Engineering (2002) 11–20.
[5] B. Knutsson, H. Lu, J. Mogul, Architecture and pragmatics of server-directed transcoding, in: Proceedings of the 7th International Web Content Caching and Distribution Workshop, August 2002, pp. 229–242.
[6] A. Fox, S.D. Grebble, Y. Chwathe, E.A. Brewer, Adapting to network and client variation using infrastructural proxies: lessons and perspectives, IEEE Personal Communications (August) (1998) 10–19.
[7] B. Noble, System support for mobile adaptive applications, IEEE Personal Communications (February) (2000) 44–49.
[8] H. Bharadvaj, A. Joshi, S. Auephanwiriyakul, An active transcoding proxy to support mobile Web access, in: Proceedings of IEEE Symposium on Reliable Distributed System, October 1998, pp. 118–123.
[9] S. Acharya, H.F. Korth, V. Poosala, Systematic multiresolution and its application to the World Wide Web, IEEE Data Engineering (1999) 40–49.
[10] F. Kitayama, S. Hirose, G. Kondoh, Design of a framework for dynamic content adaptation to Web-enabled terminals and enterprise applications, in: IEEE Software Engineering Conference, 1999, pp. 72–79.
[11] F. Reynolds, J. Hjelm, S. Dawkins, S. Singhal, Composite Capability/Preference Profiles (CC/PP): A User Side Framework for Content Negotiation, W3C note, 27 July 1999.
[12] H. Ohto, J. Hjelm, CC/PP Exchange Protocol Based on HTTP Extension Framework, W3C note, 24 June 1999.
[13] W3C, The Resource Description Framework. Available from: <http://www.w3.org/RDF/>.
[14] E. Zemel, The linear multiple choice knapsack problem, Operations Research 28 (November) (1980) 1412–1423.
[15] E. Balas, E. Zemel, An algorithm for large zero–one knapsack problems, Operation Research 28 (1980) 1130–1154.
[16] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, Wiley, Chichester, UK, 1990.
[17] R.E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
[18] P. Toth, Dynamic programming algorithms for the zero–one knapsack problems, Computing 25 (1980) 29–45.
[19] P.J. Kolesar, A branch and bound algorithm for the knapsack problem, Management Science 13 (1967) 723–735.
[20] E. Horowitz, S. Sahni, Computing partitions with applications to the knapsack problem, Journal of ACM 21 (1974) 277–292.
[21] S. Martello, P. Toth, An upper bound for zero–one knapsack problem and a branch and bound algorithm, European Journal of Operation Research 1 (1977) 169–175.
[22] K. Lai, M. Baker, Measuring bandwidth, in: Proceedings of INFOCOM'99, vol. 1, IEEE Computer and Communications Societies, 1999, pp. 235–245.
[23] V. Jacobson, Pathchar, 1997. Available from: <ftp://ftp.ee.lbl.gov/pathchar/>.
[24] D.S. Johnson, K.A. Niemi, On knapsacks, partitions, and a new dynamic programming technique for tree, Mathematics of Operations Research 8 (1983) 1–14.
[25] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, Berlin, 2004.

Rong-Hong Jan received the B.S. and M.S. degrees in Industrial Engineering, and the Ph.D. degree in Computer Science from National Tsing Hua University, Taiwan, in 1979, 1983, and 1987, respectively. He joined the Department of Computer and Information Science, National Chiao Tung University, in 1987, where he is currently a Professor. During 1991–1992, he was a Visiting Associate Professor in the Department of Computer Science, University of Maryland, College Park, MD. His research interests include wireless networks, mobile computing, distributed systems, network reliability, and operations research.



Ching-Peng Lin received the B.S. degree in Computer Science from Fu Jen Catholic University in 2000 and M.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan, in 2002. His research interests include wireless networks, mobile computing and wireless Internet.

Maw-Sheng Chern received a B.S. degree and an M.S. degree in Mathematics from National Taiwan Normal University and National Tsing Hua University respectively, an M. Math. in Combinatorics and Optimization and a Ph.D. in Management Sciences from the University of Waterloo. He joined the Department of Industrial Engineering and Engineering Management, National Tsing Hua University in 1980 and was the department chair from 1991 to 1994. Dr. Chern served as the program director for Industrial Engineering and Management Division, National Science Council, ROC from 1995 to 1998. He also served on the editorial board of IIE Transactions on Logistics and Scheduling (1997–200), International Journal of Industrial Engineering—Theory, Applications and Practice (1994–1996), Chiao-Ta Management Review, and Journal of Management & Systems. His current research interests include combinatorial optimization, production scheduling, and network programming.