Time Optimized Algorithm for Web Document Presentation Adaptation

Rong Pan and Peter Dolog

IWIS – Intelligent Web and Information Systems,
Department of Computer Science, Aalborg University,
Selma Lagerlöfs Vej 300, DK-9220 Aalborg, Denmark
{rpan,dolog}@cs.aau.dk

Abstract. Currently information on the web is accessed through different devices. Each device has its own properties such as resolution, size, and capabilities to display information in different format and so on. This calls for adaptation of information presentation for such platforms. This paper proposes content-optimized and time-optimized algorithms for information presentation adaptation for different devices based on its hierarchical model. The model is formalized in order to experiment with different algorithms.

Keywords: adaptation, hierarchical model, web document.

1 Introduction

Currently the mobile multimedia network is developing fast, and many more choices are being provided for people to access Internet resources. A large number of resources are available on the web. However, one fixed page size fits all approach does not fit mobile web very well mainly because of their different capabilities to exhibit information comparing with personal computer (PC) monitors, such as the size, resolution and the color quality of a device. For example, displaying a web page on a PC might turn out a mess on the screen of a PDA or a mobile phone: texts are jam-packed due to the limited screen size. Pictures are fragmented and presented with inappropriate resolution, requiring scrolling both horizontally and vertically necessary to read the whole page. With the development of various interconnected devices, the situation is worsening.

An adaptive presentation approach of a web document might relieve such a problem. That is, the mobile device should be able to choose the best way to display the content automatically. Our former work [11] proposed a method of retaining some redundancy on the web documents by repeating the same information in different scales of detail like size and color, to allow the terminal (which means the PDA, mobile phone's screen) to choose the most suitable scale for it. In the previous example, a traditional web page can be repeated in different grains of detail, together with some meta-tags for the terminal to identify. When it is browsed on a PDA, the quality of pictures will be reduced to fit the screen. If it is shown on an ordinary mobile phone,

D. Dicheva and D. Dochev (Eds.): AIMSA 2010, LNAI 6304, pp. 81–90, 2010.
© Springer-Verlag Berlin Heidelberg 2010

only some text-based descriptions will be provided to facilitate the even smaller display. To sum up, the web document will permit the devices to display auto-adaptive according to their abilities, by choosing the content's appropriate form, and hence alleviate the misrepresentation.

This work is a progress of a series of former works [7, 9, 10, 11]. Previously a simpler hierarchical model was introduced as a special case of the one defined in Pan et al [11], in which the model is generalized from binary trees to more practical unranked ones. Building on this ongoing work, this paper's main contributions are as follows:

- For the hierarchical model, it provides an algorithm for time-optimized solutions which is more feasible with its detailed analyses.
- The comparison among other two algorithms for content-optimized solutions and time-optimized solutions; discussion about the advantages and disadvantages of each solution.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. Section 3 will recall the hierarchical model and the formal problem related to it, then Section 4 will propose the algorithms for such problem, analyze it and compare the algorithms. Section 5 introduces the evaluation for three algorithms, and Section 6 presents the conclusion and future works.

2 Related Works

The ontology or knowledge based method [1, 2, 6] is a main approach to creating an adaptive or pervasive computing environment. The knowledge to be presented is structured upon its formal semantics and reasoned about using some reasoning tools, based upon certain axioms and rules. This is suitable to derive deeper facts beyond known knowledge and their exhibition. Compared with their way, the work presented in this paper is more pages implementation-related and fit for simple format-based hypertext to be displayed. In [13], traditional bipartite model of ontology is extended with the social dimension, leading to a tripartite model of actors, concepts and instances, and illustrates ontology emergence by two case studies, an analysis of a large scale folksonomy system and a novel method for the extraction of community-based ontology from web pages.

[3] presents a statemachine based approach to design and implement adaptive presentation and navigation. The work in this paper can be used as a target model for a generator from a state machine.

Phanouriou [14] and Ku et al [5] also proposed markup languages to interconnect different devices. The former one proposed a comprehensive solution to the problem of building device-independent (or multi-channel) user interfaces promoting the separation of the interface from the application logic. It introduced an interface model to separate the user interface from the application logic and the presentation device. It also presented User Interface Markup Language 2 to realize the model. The latter proposed the device-independent markup language that generated automatically and thus unified the interfaces of home appliances, while interfaces generated by the proposed transformation engine would also take into account interfaces previously

generated for the user and create single combined interfaces for multiple connected appliances. These two mainly focus on both user and application interfaces.

Er-Jongmanee [4] took XML and XSLT to create user interfaces which is also a possible way of implementation, but her work concentrates on design time adaptation while the work presented in this paper focus runtime adaptation.

Wang and Sajeev [15] provided a good conclusion for the state-of-the-art in such field. They studied abstract interface specification languages as the approach for device-independent application interface design. Then they classified their design into three groups, discussed their features and analyzed their similarities and differences.

3 Hierarchical Model

This section aims at a brief recollection of the hierarchical model defined in Pan et al [11] to make this paper more self-contained. Detailed descriptions of this model can be found in [11].

Informally, our approach of web document presentation adaptation is to provide different abstraction levels of the content, by the following cutting-and-condensing method. First, divide a web document into N_0 segments. Then condense neighboring segments into a briefer one by abstracting words and/or reducing picture quality, viz. losing some type of content details. This is performed recursively to form different levels of more general segments. Each segment corresponds to several adjacent ones in a lower level, forming a complete tree structure with all leaves on the same level. In the achieved tree structure, each segment serves as a node, and the relationship of abstraction forms parents and children. At last, the whole document is condensed as one segment, that is, the root of the tree structure (the title of the web page, for example). Suppose abstracted segments never need more capability of the terminal (in the light that they never take more area on the terminal) than their children, then our problem can be formalized as finding proper nodes on the tree to display, whose capability consumption suits the terminal's limit. Let us now describe formally the model.

First is a mapping to assign a serial number for each segment

$$f : Infoparts \to \mathbb{N}$$

in which we make a sequence of all the segments of all levels, from the n-th level (most condensed; root level) to the 0-th level (most detailed; leaf level). We assign 0 as the root's mapping and denote the number of nodes on the i-th level as N_i. Hence each segment's mapping can be deducted according to its position in the tree.

Definition 1. For $m, n \in \mathbb{N}$, denote m as n's *parent* if $f^{-1}(m)$ is a directly compressed part of $f^{-1}(n)$ and some of its neighbors; if m is n's parent, then n is m's *child*.

From Definition 1 we obtain two mappings *ch* and *pa*:

$ch : \mathbb{N} \to 2^{\mathbb{N}}$, mapping a number to all its children;
$pa : \mathbb{N} \to \mathbb{N}$, mapping a number to its parent.

And for the convenience of statement, we indicate a lifting of *ch* as

$$CH : 2^{\mathbb{N}} \to 2^{\mathbb{N}}, S \mapsto \{ch(n) \mid n \in S\}.$$

Definition 2. For $m, n \in \mathbb{N}$, denote m as n's *ancestor* if a sequence $m, m_1, m_2, \ldots, m_k, n$ exists, such that in each adjacent pair, the left is the parent of the right; if m is n's ancestor, then n is m's *descendant*.

Based on Definition 2, another two mappings *de* and *an* are defined:

$de : \mathbb{N} \to 2^{\mathbb{N}}$, mapping a number to all its descendants;

$an : \mathbb{N} \to 2^{\mathbb{N}}$, mapping a number to all its ancestors.

We can define the *weight* of each node as its exhibited size decided by the layout manager of the browser. When the layout manager generates the nodes, each node has its own size, which can be calculated as the *weight* of the consumed resources.

Definition 3. Denote w_i as i's *weight*, where w_i is defined by an existing mapping $we : \mathbb{N} \to \mathbb{N}$, and $w_i = we(i)$.

With these definitions, the problem can be changed to a new form: with a given sequence of weights $w_0, w_1, \ldots w_N$, where $w_i \le \sum_{j \in ch(i)} w_j$, construct a set $S \subseteq \{0, 1, \ldots, N\}$, satisfying that for any a, $N - N_0 \le a \le N$, $\exists b \in S$, $b = a$ or $b \in an(a)$, and $\sum_{i \in S} w_i \le w$, where w is given.

Definition 4. Denote a tree T as a *hierarchical model*, if each node of T has been assigned a natural number as an identifier (the assignment is in breadth-first order and begins at zero), and each node i has unique weight w_i, where any parent node i and its children set $ch(i)$ have relationship $w_i \le \sum_{j \in ch(i)} w_j$ for their weights.

Definition 5. Set S is defined to be a *cover set* of a hierarchical model T, if $\forall i \in S, i \in T$ holds for any node i, and for any leaf node $i (N - N_0 \le i \le N)$ in T, on the path from it to the root node 0, there exists one and only one node s, such that $s \in S$. (For any empty hierarchical model, its cover set is defined as empty set.)

The problem of auto-adaptation of web document can be formalized as:

Definition 6. For certain hierarchical model T and constant w, their *hierarchical problem* is to find a cover set S of T, such that provided the sum of weights of the nodes in S does not exceed w, $|S|$ should be maximized.

For any given hierarchical model T and constant w, a natural means to find a best (or largest) cover set S is to try every cover set of T, and choose the one with the most nodes and not exceeding the weight limit w.

The cover set containing only the root, namely, $\{0\}$, is a cover set. Besides this one, there are still many others, which all follow the rule that each of them is built up by some smaller cover sets of the root's children trees. It is not difficult to prove that only these two types of cover sets exist. So an equivalent definition may be stated as follows.

Definition 7. Set S is defined to be a *cover set* of a hierarchical model T, if $S = \{0\}$, or $S = \bigcup_{i=1}^{N_{n-1}} S_i$, where $S_i \cdots S_{N_{n-1}}$ are cover sets of 0's child trees, respectively. (For any empty hierarchical model, its cover set is defined as empty set.)

This recursive definition reveals the essence of cover sets. It also shows a direct way to find all the cover sets of a hierarchical model: first find all the cover sets of all root's children trees, then make a combination with each cover set of each child tree, plus the one containing only the root.

4 Algorithms for the Hierarchical Model

4.1 Two Related Algorithms for Content-Optimized Solutions

In Pan et al [11] we discussed two algorithms for the hierarchical model; and the content-optimized solutions are to reveal as much content as possible (which might be infeasible in computation) while time-optimized plays the other role. One of the algorithms is based on direct search which has both a local set to catch the current cover set and a global repository to store all of them. It first tries to add each whole level of nodes into the current cover set, then saves the cover set into the repository, and attempts to replace some of the nodes with the children next.

Another algorithm is based on dynamic plan. The goal is to find and store the number of nodes in the best cover set of each node as the root of a small hierarchical model, which is part of the original one, and combine these cover sets with the numbers stored to form the solution to the original problem.

The first algorithm's time complexity is $O(m^N)$. The second one's time complexity is $O(\binom{w-1}{m-1}wN)$, with N as the number of nodes in the hierarchical model, for each node has exactly m children, each of which has a weight at most w.

4.2 An Algorithm for Time-Optimized Solutions

For the hierarchical problem of a model, there is no formal definition for what a time-optimized solution is; hence it can be achieved in many different ways. Here the greedy principle is adopted to conduct a local search for a possible solution to the problem. It restricts the search within two adjacent levels of the model, namely, for given weight limit w and hierarchical model T, denoting CH^i as the i-th composition of CH, if

$$\sum_{i \in CH^{n-k-1}(\{0\})} w_i \le w \le \sum_{i \in CH^{n-k}(\{0\})} w_i$$

holds, say, the k-th level of T has a sum of weight no less than w and the (k+1)-th level has a weight sum no more than w, then the search will base on all the nodes on the k-th level, whose total weight just exceeds the limit, and some of these nodes will be replaced with their parents on the (k+1)-th level to reduce the total weight to a value below the limit.

The procedure of our algorithm can be divided into two steps. The first is to find a proper level as the search base, and the second is to replace some nodes on the base level with their parents. In the second step another greedy strategy is used, that on the search base level, the weight which is the sum of children's weight subtracting their parents', and divided by the number of the children minus one, is sorted descendently. The children with larger quotients above are replaced to enforce a faster decrease of

total weight. The greedy strategy used here is to prevent the selection of replacing order from drowning in a KNAPSACK-like problem. At last, also it can be seen that the solutions are restricted in the two adjacent levels.

procedure TIME-OPTIMIZED (*h_model*, *w*) **returns** the time-optimized cover set of *h_model*
  inputs: *h_model*, the hierarchical model
    w, the weight limit
  $i \leftarrow n$; *result* $\leftarrow \{n_0\}$
  while $0 \leq i$
    weight $\leftarrow$ the total weight of the nodes on $level_i$
    if *weight* $\leq w$ **then**
      result $\leftarrow$ {all nodes on $level_i$}
    else if $i=n$ **then**
      result $\leftarrow$ **empty**; **exit while**
    else
      DESC_SORT($SA = \{(\sum_{k \in ch(j)} w_k - w_j) / (|ch(k)| - 1) \mid j \in CH^{n-i}(\{0\})\}$)
      $j \leftarrow 0$
      for $j \leftarrow 0$ **to** $N_{i-1} - 1$ **step** 1
        REPLACE(*result*, *ch* (*j*-th node of *SA*), *j*-th node of *SA*)
        RENEW(*weight*)
        if *weight* $\leq w$ **then**
          exit for and **while**
        end if
      end for
    end if
    $i \leftarrow i - 1$
  end while
  return *result*
end TIME-OPTIMIZED

The greedy idea restricts the search within the nodes: each level is just scanned once at most, and even each node is also considered no more than once. So the complexity of the algorithm is highly improved, that is, $O(N \log N)$ (where N is the number of nodes). This is feasible enough for most applications.

However, the solution gained by this means is generally a possible one. If we take the number of nodes in the cover set as the only standard to judge the quality of a solution, then sometimes this solution can have a considerable gap from the best one, due to the imbalance of the weights on different branches of trees. Below is a somewhat extreme example to illustrate this.

Example 1. Suppose the nodes of a hierarchical model have their weights listed as below, and the weight limit is 7. The time-optimized solution algorithm for this will find t result $S' = \{n_1, n_2, n_8, n_9\}$, shown in Figure 1 (with the numbers beside nodes to represent their weights).

This algorithm's time complexity perfectly fits the needs of web document presentation in runtime, especially in the mobile multimedia network environment. Meanwhile, the nodes that this algorithm returns are placed on the two adjacent levels of

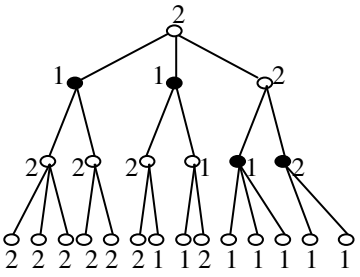

Fig. 1. An Example of the Time-optimized Solutions

the model, so they are similar in abstraction scale of the web document, and are thus more convenient for display, in support of this practical algorithm.

5 Evaluation

For this time-optimized algorithm and the two for content-optimized solutions in Pan et al [9], we have performed experiments over many different sorts of web documents and done statistics with their models.

5.1 Motivation

As far as evaluation is concerned, we focus on two aspects, viz. time consumption of the algorithms and quality of nodes in the cover sets they provide, respectively. Here "quality of nodes" consists of two criteria, i.e. the sum of nodes' weight and the similarity of levels the nodes are in. Hence we have three perspectives of observation for our experiments: time consumption, sum of nodes' weight, and level variance. We will compare and explain these three aspects in the following sections.

5.2 Method

The idea of the experiment is as follows. 1. Select a group of HTML pages as a basic test page. 2. Process the various elements in the page as the nodes in the hierarchical model. 3. Assign the weight of each node based on the consumed resources, where the terminal maximum weight limit is based on the issuance capacity. 4. Use the above algorithm for the cover sets. 5. Finally, the nodes in the model obtained are organized as a new re-mix, and make comparison among the three algorithms according to the returned nodes set and the time consumption. Thus the original web page is completed for adaptive information issuance.

5.3 Dataset

As for experiment dataset, we have built respective 1,000 and 10,000 hierarchical models from two sets of web documents. When we build those models, the web documents' layout is removed and we only consider the number of characters to calculate the weight of each node. We want to show the differences of results when choosing

the various numbers of hierarchical models. For both sets of 1,000 and 10,000 test cases we still have two dimensions as scales of the data: level numbers of models and weight limits of each node. The level numbers are 3, 4 and 5 and the weight limits are 3, 4 and 6, dividing each dataset as 9 groups. We also want to show the stability of the algorithm in time consuming and the quality of cover sets by choosing the different number of levels and weight limits. The observed time consumption is based on the average time of executing the same experiment ten times over the same dataset (to obtain more accurate result).

5.4 Experimental Result

Of the entire 10,000 hierarchical models constructed, in the condition of the maximal weight limit is 6 and for the 5-levels, the two algorithms for content-optimized solution both return 57,981 nodes, while the one for time-optimized gave 53,277; the ratio of the two types of solutions is about 1.09:1. It is found that, on a very large hierarchical model (which is the only one), the content-optimized solution has 10 more nodes than the time-optimized one; in other situations the difference in nodes of two types of solutions has not exceeded 5, and mostly there are only one in difference.

With a little gap in nodes quantity, these three algorithms have distinct time performance. For all the 10,000 cases, the direct search algorithm took 399.487 seconds, the dynamic plan version is done in 0.257 seconds, and it is just 0.018 for the time-optimized one. The first is over 22,194 times more than the last one. The comparison on time consumption is shown in Table 1.

Another observation on our experiments is the other evaluation of cover set quality: the level variance of nodes, which is the sum of all cases' square roots of variance concerning the level difference of nodes in each cover set. The smaller the value is, the nearer levels are the nodes in the cover sets on. Since the time-optimized solution algorithm always finds solutions in adjacent levels, its level variance is just about half of the former two's. On this aspect the time-optimized algorithm performs better than the other two due to its greedy essence. On average, the time-optimized algorithm finds cover sets with slightly more than half level variance of that of the results from the other two methods, say 2043.37 compared with 3571.19 (57.22% in proportion). This trend is more evident for hierarchical models with more levels, as this data is 626.01 vs. 988.74 (63.31%) for 3-level hierarchical models whereas 731.13 vs. 1428.43 (51.18%) for 5-level ones. This might be attributed to the fact that the nodes' levels tend to vary more in models with more levels (and hence the other two algorithms tend to find worse results albeit they have more total nodes). We conjecture this trend would be more obvious for more complicated hierarchical models where time-optimized algorithm would yield more practical results for issuance.

Through the comparison, the direct search algorithm should be eliminated from practical use. The dynamic plan can find better cover sets within not long time, but the abstraction degree of nodes is not as good as the one for time-optimized solutions, whose execution time is also the least. To conclude, in a practical environment, these two algorithms should be evaluated more precisely according to the application's real needs.

Table 1. Comparison of the three algorithms On Time Consumption

	1000 Cases				10000 Cases			
Algorithm	**Weight Limit**	**3-Levels**	**4-Levels**	**5-Levels**	**Weight Limit**	**3-Levels**	**4-Levels**	**5-Levels**
Direct Search	3	0.637	3.473	64.893	3	2.339	9.690	381.250
	4	0.781	3.561	67.815	4	2.351	10.310	390.120
	6	0.820	3.667	69.375	6	2.397	11.375	399.487
Dynamic Plan	3	0.000	0.001	0.016	3	0.031	0.072	0.156
	4	0.001	0.002	0.032	4	0.046	0.087	0.188
	6	0.016	0.002	0.038	6	0.049	0.110	0.257
Time-Optimized	3	0.001	0.001	0.001	3	0.006	0.007	0.009
	4	0.001	0.001	0.001	4	0.006	0.008	0.010
	6	0.001	0.001	0.002	6	0.006	0.016	0.018

6 Conclusion and Future Works

In this paper we proposed a time-optimized algorithm based on the HTML hierarchical model for adaptive web document presentation, and compared it with our former works. The comparison indicates that our previous content-optimized solutions can find more comprehensive cover sets; however, the time-optimized solution algorithm finds solutions in more adjacent levels and has great advantage in time complexity, which is therefore regarded better.

Our future works include designing an adaptive extension of XHTML recommendation finding both other strategies for solutions to the problem, and more potential facets of the device's capabilities to help to extend the XHTML. Meanwhile, as the algorithms rely on a weighting of the nodes in the hierarchical model, we will study different weighting algorithms for document modeling in the future work.

References

1. Cui, G., Sun, D., et al.: WebUnify: An Ontology-based Web Site Organization and Publication Platform for Device Adaptation. In: Proceedings of SNPD 2004 International Conference (July 2004)
2. Dolog, P., Nejdl, W.: Semantic Web Technologies for the Adaptive Web. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web: Methods and Strategies for Web Personalization, pp. 697–719. Springer, Heidelberg (2007)

3. Dolog, P., Nejdl, W.: Using UML and XMI for Generating Adaptive Navigation Sequences in Web Based Systems. In: Stevens, P., Whittle, J., Booch, G. (eds.) UML 2003. LNCS, vol. 2863, pp. 205–219. Springer, Heidelberg (2003)
4. Er-Jongmanee, T.: XML-Driven Device Independent User Interface–build Rich Client Applications Using XML. Master thesis, Lehrstuhl für Informatik V (2005)
5. Ku, T., Park, D., Moon, K.: Device Independent Authoring Language. In: Fourth Annual ACIS International Conference on Computer and Information Science, pp. 508–512 (2005)
6. Lewis, D., Conlan, O., et al.: Managing Adaptive Pervasive Computing using Knowledge-based Service Integration and Rule-based Behavior. In: Proceedings of IFIP/IEEE Network Operations and Management Systems, Seoul, Korea (April 2004)
7. Luo, C., Pan, R., Wang, S.: A Hierarchical Model for Auto-adjusting of Information Issuance. International Journal of Computer Science and Network Security, IJCSNS (December 2006)
8. Ma, W., Bedner, I., Chang, G., et al.: Framework for adaptive content delivery in heterogeneous network environments. In: Proceedings of Multimedia Computing and Networking, San Jose (January 2000)
9. Pan, R.: The PLCH Binary Tree Model of the Auto-adaptation of Web Information Issuance. Journal of Computer Applications (May 2005)
10. Pan, R.: A Mathematical Model for the Hierarchization of Web Information and Its Second-Best Algorithm. Computer Engineering and Science (October 2005)
11. Pan, R., Wei, H., Wang, S., Luo, C.: Auto-adaptation of Web Content: Model and Algorithm. In: ICWMMN 2008, Beijing, China, October 12-15, 2008, pp. 507–511 (2008)
12. Pemberton, S., et al. (eds.): XHTML 1.0 Specification, W3C Recommendation (August 2002), http://www.w3.org/TR/xhtml1/
13. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semantics: Science, Services and Agents on the World Wide Web 5(1), 5–15 (2007)
14. Phanouriou, C.: UIML: A Device-Independent User Interface Markup Language, PhD thesis, Virginia Polytechnic Institute and State University (2000)
15. Wang, L., Sajeev, A.S.M.: Abstract Interface Specification Languages for Device Independent Interface Design: Classification, Analysis and Challenges. In: Proceedings of First International Symposium on Pervasive Computing and Applications, Xinjiang, China, August, pp. 241–246. IEEE Press, Los Alamitos