Slicing*-Tree Based Web Page Transformation for Small Displays

Xiangye Xiao, Qiong Luo, Dan Hong, Hongbo Fu
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{xiaoxy, luo, csdhong, fuhb}@cs.ust.hk

ABSTRACT

We propose a new Web page transformation method for browsing on mobile devices with small displays. In our approach, an original web page that does not fit into the screen is transformed into a set of pages, each of which fits into the screen. This transformation is done through slicing the original page. The resulting set of transformed pages form a multi-level tree structure, called a slicing*-tree, in which an internal node consists of a thumbnail image with hyperlinks and a leaf node is a block from the original web page. Our slicing*-tree based Web page transformation eases Web browsing on small displays by providing screen-fitting visual context and reducing page scrolling effort.

Categories and Subject Descriptors: H.4.3 [Information Systems Applications]: Communications Applications–Information Browsers; H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia–Navigation;

General Terms: Design, Algorithms, Human Factors.

Keywords: Web browsing, small displays, proxy, thumbnails, Web page adaptation, slicing tree, VIPS algorithm.

1. INTRODUCTION

The majority of current Web sites are designed for desktop displays, and only a handful of browsers on PDAs (e.g., PalmScape, ProxiNet, HandWeb, and Power Browser [1]) support limited Web page adaptation for small displays. As a result, PDA users have to scroll constantly when viewing a Web page on a palm-sized screen. In this paper, we explore a new approach to automatic page transformation for small displays to remove two-dimensional scrolling.

Our approach is based on the following key observation: PDA users draw heavily on their browsing experience on desktops when browsing on PDAs. In particular, page layout and visual context information are crucial for users to identify their interests in a page. Therefore, for an original Web page that does not fit the screen, our transformation method first displays its thumbnail image with multiple embedded hyperlinks (Figure 1 (a)). When a part of the page is pen-tapped, its corresponding screen-fitting sub-page is then displayed, and this tap-and-display process may continue a few more times until the target of interest is found (Figures 1 (b)-(d)). During the entire browsing process, the original page layout and context information is preserved and scrolling is seldom needed.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.
ACM 1-59593-140-6/05/0010.

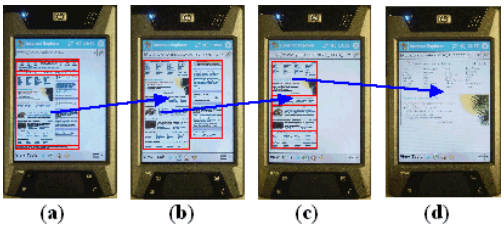

Figure 1: Displaying transformed pages on a PDA.

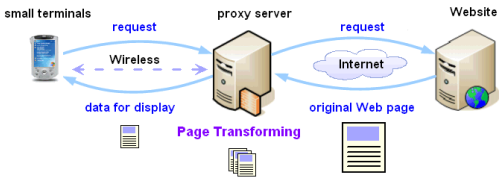

Figure 2: Page transformation proxy architecture.

Figure 2 illustrates the architecture of our page transformation proxy server. When a PDA sends a request to a Web site through our proxy, the proxy forwards the request and transforms the received Web page into a tree hierarchy of thumbnail index pages as internal nodes and leaf sub-pages as leaf nodes. It returns the root index page to the PDA for display and stores the other sub-pages locally. When the user selects a region in the index page, it then serves the user the corresponding sub-page that has been stored.

The page transformation system at the proxy consists of four modules: page splitting, and the generation of thumbnail images, index pages, and leaf pages. The page splitting module builds the tree organization of the set of transformed pages with several factors considered, including the size of the screen, the size of each page block, the number of blocks in each transformed page, as well as the semantic coherence between blocks. We adopt a variation of the binary slicing tree [3], which we call a slicing*-tree, to represent the organization of the set of transformed pages. Each internal node in the tree is a thumbnail of the original Web page (the root) or that of an intermediate sub-page, and each leaf node is a leaf page (a sufficiently small block in the original Web page).

2. SLICING*-TREE FOR A WEB PAGE

In this section, we present our slicing*-tree based page transformation, in particular, the slicing*-tree construction.

A slicing floorplan is a decomposition of a rectangle with horizontal and vertical cuts. It can be represented as a binary tree, called a binary *slicing tree*. An internal node in the slicing tree represents a cut, either in the horizontal dimension or in the vertical dimension. A leaf node in the tree represents an atomic rectangle that has no cut through.

In comparison with the binary slicing tree, our slicing*-tree does not require each internal node to have exactly two children. Instead, it requires each internal node has at least two children and has no more than a specific number of children, which we denote as the threshold T. We add this threshold based on the observation that it is easy for users to identify in a region the sub-region of interest as long as there are only a few, not necessarily two, sub-regions present.

As a Web page typically consists of box-shaped sections, we can naturally use a slicing*-tree to represent a Web page decomposition scheme. A slicing*-tree represents the transformation of a Web page as follows: (1) each leaf node represents a leaf page transformed directly from a section in the original page. The leaf page fits into the screen; (2) each internal node represents a thumbnail image page of a large section of the original page. This thumbnail image fits into the screen. Furthermore, for each child node, there is an embedded hyperlink pointing to its corresponding sub-section in the thumbnail image of the parent node.

The construction of slicing*-tree representation of a Web page goes through three steps in sequence: VIPS tree construction, leaf extraction, and internal node adaptation.

The VIsion-based Page Segmentation (VIPS) [2] algorithm extracts the semantic tree structure of a web page based on its visual presentation. Each node corresponds to a box-shaped section (i.e., a block). The VIPS tree of a Web page has most of the properties of a slicing*-tree except that (1) a leaf page may be too small or too large with respect to the screen size, and that (2) the number of children of an internal node may be larger than the threshold. Thus, the problem of slicing*-tree construction becomes to first generate the VIPS tree and then to transform it into the slicing*-tree by addressing these two differences.

To be conservative about the leaf page size, we generate the VIPS tree so that its leaf pages are as small as possible. After the VIPS tree with small leaf pages is constructed, we go through the leaf extraction step to make each leaf block as large as possible within the screen size limit. This is done by merging small-sized neighboring sibling nodes in the tree. We traverse the VIPS tree through Depth-First Search (DFS). For each internal node, if its size is smaller than the screen size, we remove all of its children because there is no need to further decompose it. Otherwise, we examine its children and see if these children can be re-partitioned to decrease the number of children and to increase the size of each new child node within the screen size limit.

The following step, internal node adaptation, adjusts the tree to satisfy fanout requirement and to reduce the height of the tree as much as possible. It is done through a DFS traversal on the tree. If a node has more than T children, the algorithm combines some children by adding more levels of internal nodes between the node and these children so that it has exactly T children. If a node has fewer than T children, the algorithm attempts to increase its degree up to T.

Our slicing*-tree based transformation achieves a balance between two extremes in the previous work. One extreme, the binary slicing tree organization, generates a deep hierarchy of transformed pages consisting of two sub-pages only. With this organization, users need to pen-tap many times to navigate down to the target sub-pages. The other extreme uses a two-level hierarchy, with the top level page consisting of the hyperlinks to all of the bottom level sub-pages. Its downside is that there may be too many sections in the top level page so that it is hard for users to choose from.

3. EXPERIMENTAL EVALUATION

We have conducted initial experiments using an HP iPAQ hx4700 PDA. We asked users to perform ten tasks, including focused search tasks and reading tasks, on PDAs with and without the page transformation proxy, and compared the task completion time, bandwidth consumption, and input effort. Our system achieved a shorter task completion time in all of the ten tasks, with 47% improvement on the focused search tasks and 22.7% improvement on the reading tasks. In nine out of the ten tasks, it reduced the input effort and achieved a smaller number of pen moves. The improvement was 50% for the focused search tasks and 28.4% for the reading tasks. It saved bandwidth consumption in seven out of the ten tasks, and the average saving was 18.9%.

To study the performance of our system in more detail, we further divide the task completion time into three parts: user interaction time on the client device, processing time at the proxy, and data transmission time on the network. Our results indicate that the processing time spent on page transformation at the proxy was only a small portion (less than 10%) of the overall task completion time. In addition, we see that page transformation time drops as T increases. The improvement slows down as T becomes even larger. The total size of the transformed pages is usually a few times larger than that of the original page. However, because only the pages chosen by the user will be transferred to the client device, our system still reduced bandwidth consumption.

4. CONCLUSION

We have proposed a slicing*-tree based page transformation method for improving Web browsing on small terminals. In our approach, a Web page is transformed into a set of thumbnail index pages and leaf pages that form a multi-level tree structure with bounded node degree. We have demonstrated by experiments that our approach significantly eases Web browsing on PDAs. An extended version of this paper is available as a technical report [4].

5. REFERENCES

[1] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: efficient web browsing for pdas. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 430–437, 2000.

[2] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Vips: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2003.

[3] M. Lai and D. Wong. Slicing tree is a complete floorplan representation. In *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, pages 228–232, Piscataway, NJ, USA, 2001. IEEE Press.

[4] X. Xiao, Q. Luo, D. Hong, and H. Fu. Slicing*-tree based web page transformation for small displays. *Technical Report HKUST-CS05-13*, August 2005.