Unpackable Treemaps as Web History Graphs

Jennifer Tillett

Department of Computer Science

University of British Columbia

ABSTRACT

Web browser users often find themselves lost in their navigation. They cannot revisit pages they have already seen and have little sense of the context of their browsing. Web history graphs can augment or replace traditional browser history mechanisms.

I have implemented a web history application called Unpackable Treemaps which is coupled with a web browser. Unpackable Treemaps builds a treemap representing a user's browsing history which can be navigated to find previously visited sites. Unpackable Treemaps are organized by browsing session and a user can add on to any session by focusing a previously visited page in Unpackable Treemaps, which will bring it back up in the browser.

Unpackable Treemaps also feature an interactive design which allows the user to "pack up" nodes which do not interest him for the time being. The user can zoom in on interesting features of the treemap and delete subtrees.

CR Categories and Subject Descriptors: H.5.4 [Hypertext/Hypermedia]: Navigation, User Issues, I.3.6 [Computer Graphics]: Interaction Techniques

Additional Keywords: Web History Visualization, Information Visualization, Web Browser Usability

1 INTRODUCTION

Browsing the web, as enjoyable as it is for most, can be quite difficult for the novice or even an experienced user when he has a specific task or search in mind. Even within one session, navigation can quickly lead the user deep into a tangle of hyperlinks -- unsure of how he got there, how to get out, or especially how to return to a previously-browsed web page. The problem compounds between browsing sessions, when the user would like to revisit a useful web page but cannot remember the address or how he got there. Though tools exist to aid the user in this kind of navigation, these are not used very frequently. Moreover the tools available have limitations on their functionality which may inhibit their use.

The phenomenon of being "lost" in hyperspace is all too common for the average user. Surveys have shown that between 13.4% and 16.6% of users report that finding previously visited web pages is a major problem [6] [7]. This number is significant because it is estimated that a majority of a users page visits are actually revisits. In surveys conducted in 1996 and 2000, it was found that between 42% and 80% of page visits are revisits [7].

Most browsers have simple functions to help the user navigate back to previously visited web pages, such as Back and Forward buttons, bookmarks, and history trees. However some of these tools may be underutilized by the average user. In a survey conducted by Catledge and Pitkow, it was found that page visits as a result of pushing the Back button comprise 40.6% of a user's actions, but only 2% of navigation was due to bookmarks, and 0.1% was a result of consulting the history list [4]. Clearly, some of the tools provided with web browsers are not considered useful by the majority of users.

Some problems with traditional history mechanisms include:

1. The Back button is implemented as a stack, so branches, where the user has visited two links from the same page using the Back button to navigate back to the original page, are not maintained. According the Back button, the first link was never visited.
2. Bookmarks require a prescience of which pages will be useful in the future, as well as maintenance by the user. He must explicitly choose to save the page and keep the list organized so that it itself is navigable. This is difficult because bookmark lists grow at a rate of about one page per 5 days [7].
3. History trees a) are text-based, requiring the user to recall whether a page is the one he's seeking based only on a title and a URL, b) are frequently organized only by domain, not by order of visitation and c) require navigation to use, as the user must select the history browser from a menu before he can search it.

Many alternatives to traditional browser histories have been developed to combat these problems. These applications are typically graphical representations of a user's history presented in an interactive manner, tightly coupled with the user's web browser so that navigation in the web history application affects navigation in the browser window and vice versa. The type and form of these web histories varies by the structure used to represent the data and the organization of the data. Often seen are trees or cyclic graphs for the data structure, but the organization is highly varied. Some applications, such as the Domain Tree Browser [6], choose to organize the history by domain, in the same manner as the browser's history. Others seek to preserve the structure and linkages of the user's browsing, such as in the Zooming Web Browser by Bederson et. al. [1]. Some choose to link the data by other similarity measures such as the content of pages, or in the case of Nestor [10], a user's personal classification of a document's meaning. Still others have combined a user's path with domain information, such as in WebMap [5], and Browsing Icons [7], which also uses a cyclic graph.

For my web history application, I have chosen a tree metaphor for the history data. However, I have chosen to use nested treemaps to draw the tree rather than a node-link diagram. There are several reasons for this. First, these treemaps save space over a node-link diagram by eliminating arcs and by overlapping nodes with one another, which presents an opportunity to fill a node with more information about the node such as an image and/or a name. Second, nested treemaps represent hierarchy very clearly even

201-2366 Main Mall
Vancouver BC Canada V6T 1Z4
jtillett@cs.ubc.ca

within the minimum of space. Third, treemaps offer the benefit of preserving some information about a node in its size, such as importance or degree of interest. The problems that trees and treemaps present is that a) there is no handling of cycles and b) treemaps do not offer a method of aggregation for condensing nodes which are not currently of interest.

In this paper I describe the tool called Unpackable Treemaps. It is implemented as a coupled web browser and web history graph, for displaying and interacting with the browser's history as a treemap. It is intended to be a suitable tool for any web browser user with any level of experience. Some familiarity with treemaps would help in using the application, but it is not necessary.

The rest of this paper is as follows: In Section 2, I describe the related work in the area of graphical web history maps. Following this, in Section 3 I describe the key features of Unpackable Tremaps. In Section 4 I present a sample scenario that illustrates how Unpackable Treemaps can be used. Section 5 is an evaluation of the strengths and weaknesses of Unpackable Treemaps, and Section 6 presents future work which would increase the usability and functionality of Unpackable Treemaps. The_final section concludes.

2 RELATED WORK

Many other graphical web history browsing applications have been developed in the last 10 years. Each attempted to improve upon commercial web browser's existing history mechanisms, and discover the best methods for presenting a browser history to the user.

Bederson et. al. developed a "Zooming Web Browser", based on Pad++, a zoomable user interface toolkit which is now available as Jazz[1] [1]. The developers chose to represent the browser history as a node-link tree, where each node is a window in which a hypertext document can be displayed. Clicking links in a document adds a node to the tree as a child of the document containing the link. The interface allows panning and zooming so that the user can smoothly move between focus and context.

Browsing Icons is a visual web history tool created by Mayer and Bederson [7]. Browsing Icons chose a session/task format for saving history, and stores a separate graph for each session. The session is considered to be a unit wherein the user is browsing the web with some task in mind. In experiments with a session-based history map, users have appreciated this "chunking" of work on the web into manageable pieces. Each session is represented as a cyclic graph, with pages represented as small circular nodes which grow as a function of the number of times the page has been viewed. The pages' titles are displayed as labels next to nodes. The authors believe that the unique structure of each session's graph helps the user by serving as landmarks for different task sessions.

WebMap, developed by Peter Doemel, is a browser extension that creates a spanning tree which represents the exact structure of a user's browsing [5]. The tree is represented as a node-link diagram with numbers as labels. Choosing nodes displays the title and URL of the page represented and displays the page in the browser.

The Domain Tree Browser is a web history visualization tool that organizes history by domain [6]. A tree is built for each domain the user has visited with branches representing all of the pages visited within that domain. The user accesses a specific domain tree from a list of all domains visited.

NESTOR Navigator is a tool that helps a user create a personal network of browsing history [10]. A tree is created using the user's browsing history, which the user can then alter by adding their own documents or altering the graph's shape, by adding notes and keywords, and by specifying conceptual areas which documents can be pinned to. Documents are represented in a node-link graph with documents drawn as small circles and titles displayed as labels next to nodes.

Finally, in related work Catledge and Pitkow surveyed web user's habits, including use of browser tools and typical history session, and categorized 3 types of browsing strategies [4]. The "Serendipitous Browser" makes fairly shallow browsing paths and avoids long sequences. The "Searcher" often performs very long navigational sequences, but far less frequently performs short sequences. The "General Purpose Browser" is the average of the other two types of browser.

3 UNPACKABLE TREEMAPS

Unpackable Treemaps is a tool implemented to help users navigate the history of their web browsing. It consists of a simple web browser and a visualization application which draws the user's web history as an interactive treemap.

The user's history is stored as sessions, where one session is an instance of starting the web browser and following hyperlinks or entering URLs into the text field at the top of the browser in order to open web pages. The session terminates when the web browser is closed. Information is stored for each session about the date and time of browsing and the structure of the paths the user took while browsing, That structure consists of one node for each page that is visited in a session. Within a page's node, information is stored about the page's URL, title, thumbnail image, background color and permissions on whether it can be packed or rolled up. Upon starting a session the user has an option to give her browsing session a name, based on the task she seeks to perform in this session or any other criteria.

Whenever the user chooses, she can open the Unpackable Treemaps visualization using a button at the top of her browser. This opens a new window displaying all of her saved history, including past sessions, as a treemap. Each node in the treemap is either the tree's root, a session root, or a web page that was browsed. The background of a node will be coloured according to criteria discussed in Section 3.3. If space allows, nodes will also display their name, which is the default label for nodes. The session root will display the name that the user chose for that particular session. Also displayed where space allows is the node's image, if a thumbnail screenshot of the web page is defined for that node. Figure 1 is a screenshot of the application in action.

If space is tight, nodes will display only their labels, and children node at the greatest depths may not be displayed. In this case, the user can zoom in on sessions or nodes of interest, and the subtree beginning from that node will be displayed, taking up the whole of the visualization panel. The user can continue to zoom, or return to the global view of all nodes for context. At any time, the user can left-click a node to display a popup menu which offers options for managing the tree, including zoom, showing all nodes of the tree at once and as much of each node as possible, setting the focus to a particular node, showing more or less of a node, packing or unpacking a node, and deleting a node.

Showing all nodes sets the size preference for each node to the minimum size possible so that each node has equal weight for the treemap drawing algorithm. When this option is selected, no node has focus, and only leaf nodes will have the option of showing their images, though all nodes may show their labels if space is available.

Setting the focus to a particular node allocates more space to that node and its ancestors and children by giving them heavier weights for the treemap drawing algorithm. All other nodes have

[1] http://www.cs.umd.edu/hcil/jazz/

the minimum weight. When a node is set in focus, the corresponding URL is brought up in the web browser, and if the user navigates from that site the navigation is added to that particular node, even if it is from another session. In this way, the browser is coupled to Unpackable Treemaps, but only in a one way tight coupling. Navigation in the browser does not automatically update Unpackable Treemaps; the user must press the button to display Unpackable Treemaps again if she wishes to update the tree displayed in the visual panel.

Expanding a node sets the preference for that node so that if possible a greater portion of the node will be displayed, making room for the screenshot. Rolling up a node sets the preference that the node should not be expanded even when there is space to do so.

Packing a node slides a node's children up to use all the space that the node was allocated, hiding the node from view. When this happens the children's background becomes red to indicate that they are occluding their parent. Unpacking forces a node's parent to be unpacked and displayed.

3.1 Navigation/Zooming

When the user opens Unpackable Treemaps, her whole history is visualized, with the current session receiving the focus and thus the largest plot of screen real estate. In this situation, both focus and context are maintained in one visual panel. However, because the focused nodes are given so much space, the user may not be able to discern much detail in the unfocused area. In this case she can do one of four things:

- Choose to show all items with the exact same degree of focus.
- Choose to set the focus on a different node, allowing it the majority of the screen space and minimizing the previously-focused node.
- Zoom in on areas of interest.
- Pack or roll up nodes she is not interest in.

Zooming is not smoothly animated in Unpackable Treemaps. When a user chooses a node to zoom in on, the display jumps to a view where that node and its children, or (if the node is a leaf) the node and its parent, take up the entire visual panel. When the user clicks the right mouse button the display jumps back to the global view of all the history.

3.2 Aggregation and Filtering

Because there is simply not enough screen space to display all of the features of every node, some data must be aggregated in order to display the entire context of the user's history. The most natural choice for a high level of the history, as it is stored in this application, is the session level. Thus in all views, even if nodes are packed tight so that no labels or images are visible, the session node still displays its label. The user can then choose to expand or zoom in on a particular session which interests him.

There are two ways in which nodes can become aggregated, or "packed up": by automatic sizing and by manual packing.

Automatic sizing occurs both when the user chooses to show all nodes and when the user sets the focus on a particular node. In both cases, nodes are assigned a weight which is used by the treemap algorithm to choose a size for the node. When a node is set to be in focus, it is given a weight of 30 (on a scale from minimum 1 to maximum 50), and its ancestors and children are assigned diminishing weights by their degree of interest, which is defined as the difference in depth between the ancestor or child and the focused node. When the user chooses to "show all" nodes, each node is given the minimum weight. In both of these cases, Unpackable Treemaps determines how much of a node and which nodes can be shown based on their size weight and the mode the user has chosen. In "show all" nodes can be packed up to their minimum size (with no label or image showing for parent nodes), if that is required to display all of the children of a branch. In "set focus" mode, showing labels is preferred and so the last children may be packed up and invisible. In this case, there is no visual indication that there are packed children, but when the user zooms in on a thread they will become visible.

Manual packing, also referred to in this paper as a "red" pack, is the situation where the user chooses a specific node to be aggregated with its child. Choosing the menu item "Pack" for a node hides the node behind its child, and allows the child node to occupy the node space that was allocated to the parent. In this case the child node becomes red to indicate that it is hiding at least one parent.

In this application, aggregating nodes which are ancestors of each other is almost guaranteed to be an aggregation of similar nodes. Connectedness in the tree is a good similarity measure for the session history tree because if a link exists between two pages they likely are similar in content, or at least similar in the user's mental model.

Related to aggregation are sorting and filtering functions provided in the control panel. At any time the user can choose to sort the treemap or filter its contents based on any attribute of the nodes, which include depth, degree (number of children), name, URL and date of browsing.

3.3 Colour

The background of each node in Unpackable Treemaps is coloured to convey information about depth, the domain, or whether the node has packed parents. Only three colours are used to distinguish types of domain: black indicates that a web page is the user's home page, blue indicates that the domain is a search engine such as Google, and a greyscale colour indicates all other domains. Red is used as a background to indicate that a node has parents which are packed away behind it. Saturation (the greyscale for backgrounds) is used as a depth cue in addition to the nesting of nodes; the lightness of a greyscale node indicates how deep it is in the hierarchy, i.e. how many ancestors it has. The saturation scale from black (exclusive) to white (inclusive) has 13 steps, due to Catlege and Pitkow's observation that the mean successive document requests within a single site is 10.64 and my observation that when the nesting goes beyond about 12 nested nodes it is preferable to zoom in for more information [4].

Although the use of color for representing domain type does not take advantage of the user's preattentive processing of colour, the greyscale depth does. It is very easy to see, without beginning to scan, which nodes are child nodes and where searches began as well as the length of browsing paths.

Fewer colours were used in Unpackable Treemaps because it was thought that when displaying screen shot thumbnails (which may be colourful themselves) too many extra colours would prove distracting. In addition, it was not thought to be useful to group nodes into colour bins because it is not known if the developer's intuition about similarity between web pages would coincide with the user's mental model of page similarity. However, a limited number of other colours could be useful for visualization of bookmarks, cycles or extra domain categories, especially if the user were able to choose the categories for binning.

3.4 Hierarchy

The hierarchy of the tree representing a user's browsing paths is indicated by both nesting of nodes in the treemap and the

saturation of the nodes' backgrounds.

When the user is navigating, two situations occur which must be handled in the tree structure. First, when the user types a URL into the text field at the top of the browser in order to navigate to a specific page, that event could be considered either another link from whatever page is currently displayed or a new branch emanating from the session root. For Unpackable Treemaps, I decided that that constituted a new browsing thread, and thus should be visualized as a new branch in the session tree. Second, when a user navigates to a web page she has already visited, the fact that the node is a multiple could be ignored, or the cycle could be flagged with a visual cue, or the tree could be managed so that multiple nodes don't need to be maintained. I decided on a combination of two of these strategies. When a web page is accessed that has previously been accessed as an ancestor of the current page, instead of adding the identical node as a new child, the original node representing the page becomes current again, and any further links pursued from the page will become new branches emanating from the original node. Figure 6 illustrates this strategy. Note that this cycle handling is different from the case when two branches from the same ancestor each eventually access the same page. In that case, if the child node being created would be a sibling of an identical node (i.e. the multiples are both immediate children of the same parent), the identical node is not added as a new branch. Instead, the original node becomes the current node and any navigation away from the page will be represented as a branch out of the original node. However, when the same page is accessed in two branches of the tree that share a more distant ancestor, both multiples are allowed to remain in the tree and there is no visual indication that the nodes create a cycle.

3.5 Images

The Unpackable Treemap web history graph features screenshots of the web pages that some nodes represent. These images are hypothesized to act as landmarks for the user and to be easier to quickly scan than a web page's title. In order to present the image most closely related to the image the user may remember from the web page, I didn't want to distort the image at all when scaling it. The danger in using undistorted thumbnail screenshots of web pages was that because the children of a node are nested inside the node, a thumbnail which didn't fill the whole of a node's box would interfere with discerning the hierarchy of children within the node. This is because an image which doesn't fill the node but is wider or narrower than its children nodes will create a ragged edge within the parent node and an image which exactly aligns with a child node might appear to be another child [Figure 3]. In an informal user poll of 7 grad students and professionals between the ages of 22 and 37, each of whom use a computer at least 20 hours per week, 6 of 7 preferred images which aligned with their child nodes over images which were wider or narrower, even when asked to try to discern the hierarchy of nodes. One user's rationale was that "the neatness and orderliness makes it easier to scan the window, and once you understand how it's all organized, the aligned pictures don't confuse you into thinking that they're separate boxes."

Unpackable Treemaps does have a function (disabled in the release) to take screenshots of a user's browsing on the fly, instead of just relying on hard copies stored in the user's computer. For reasons discussed in the Implementation section, it was unreasonable to use these during runtime. However, the visualization implications of this function can be discussed. First, a major concern was whether images for each node would create too much visual clutter. Figure 3 is an example with every node displaying a screenshot. Because of the limited use of background color and because of the dynamic rolling up of nodes, the images do not seem to clutter the screen too much. However, as discussed, both the ragged edges caused by image scaling and the alignment of images with a nodes children are potential problems. Visual scanning of the hierarchy of nested nodes may be disrupted because the image borders may be confused with nested node borders. I hypothesize that this disruption is worth the tax on visual scan due to the gains in large landmark memory cues presented by the thumbnails.

3.6 Treemap Algorithm

Unpackable Treemaps uses a squarified treemap algorithm by default, though it is possible to view the treemap with a strip algorithm or even slice-and-dice. I chose squarified treemaps because they offer the lowest aspect ratio, albeit without the promise of ordered nodes [2]. Squarified treemaps offer the benefits that: nodes are easier to select than the long skinny rectangles that often result from other algorithms, and because the nodes have aspect ratios approaching 1, they are more likely to be able to legibly display images [3]. In the same informal user poll of 7 grad students and professionals, all 7 found the squarified algorithm more *readable*, when readability was defined as a measure of ease of visual scanning [2]. I hypothesize that this preference results because the squarified algorithm rarely creates the long skinny rectangles which would form ragged borders between images and child nodes and make labels disappear or get cut off. However, for long-term use or for extended searching it may become important to maintain the order of sessions (by date or otherwise) and to avoid large layout changes when the tree is altered. That way the user wouldn't become disoriented between searches or additions or deletions to the tree. In that case it has been shown that strip algorithms offer more readability and that users prefer them to squarified treemaps for searching tasks [2]. Strip algorithms first divide the tree into strips before dividing those strips into rectangles, only dividing further if the aspect ratios of the resulting rectangles is not worse than the aspect ratio of the strip. This algorithm does not offer the lowest aspect ratios, but the order of nodes is maintained and changes to the layout are minimal even when the tree is altered [2].

3.7 Implementation

Unpackable Treemaps was implemented using Java 5 with Swing and the libraries provided in the InfoVis Toolkit[2], which is a library of visualizations including tables, trees and treemaps based on the Java 2D API. The InfoVis Toolkit provided visualization panels and algorithms for treemaps as well as a treemap control panel.

The web browser was implemented using a *JEditorPane* contained within a *JScrollPane* for the display of HTML and several *JButton*s for navigating backwards, exiting and opening the Unpackable Treemaps applications. The *NewBrowser* class was written to handle input and output of the tree, maintenance of the tree due to browsing, deletion or other options in the Unpackable Treemaps window and dynamic assigning size preferences for the nodes in the tree using a degree-of-interest measure. Several small classes were written to assign colors, labels and stored images to nodes. The history tree was input and output from an XML file using InfoVis Toolkit's *XMLReader* and *XMLWriter* classes. The tree was stored as InfoVis Toolkit's *DefaultTree*. The treemap visualization was a modification of InfoVis Toolkit's *ImageTreemapVisualization* using a border drawing class that extended *LabeledBorderDrawer*, with functionality added for centering and aligning images, dynamically resizing or expanding nodes, and packing and

[2] http://ivtk.sourceforge.net/

unpacking nodes.

The screenshots were generated using Java's *Robot* class. However, due to time differences and long loading times for the browser, accurate screenshots were not guaranteed. Often if a user moved too quickly between pages (where too quickly was approximately 12-15 seconds), the screenshot generated was not the page it was supposed to be; it was often the page's parent, or a screenshot of the Unpackable Treemaps interface, because the user had switched to that view while waiting for her page to load. In addition to those problems, space constraints made it such that images could not be saved between sessions.

4 Scenario of Use

Suzie is browsing the web. She would like to record an interactive history of her browsing, but finds the "Back" and "Forward" lists to be too narrow in their scope, because they only record linear progress. Her "History" tree simply groups pages by their domains, which doesn't preserve either linearity or any other progression, other than a loose chronological clustering by the last day a page was viewed. She decides to try an interactive history graphing application, namely Unpackable Treemaps.

To begin, she starts the Unpackable Treemap browser and application suite. When she begins browsing, she is prompted for a session name, which may later help her remember this session by the task she seeks to perform in addition to the date. Suzie can browse for as long as she likes, or not at all, before displaying the Unpackable Treemap history graph. If she immediately opens the history graph, she will see any other sessions saved in her history file as well as her current session, which will have the focus in the treemap, indicated by its size. The first page she has loaded is displayed as the first node in the current session. Because there are no other nodes, this page can take up the whole of the session's allotted node space. As she follows hyperlinks in her browser, when she redisplays the Treemap each link will be represented by a new node nested within its parent node. If Suzie visits more pages than can be legibly displayed, older nodes will be "rolled up", sliding under newer nodes above them which will expand [Figures 1 & 2]. Suzie can click any of the rolled-up nodes remaining to choose a menu item that will re-expand the hidden nodes. Suzie can also manually choose to pack a certain node, using the popup menu that displays when she left-clicks on a node. This results in a "red" pack, where the child node of the packed node is allowed to use the space that the packed node previously occupied, and is coloured red to indicate that it is covering its parent. Suzie can use the popup menu and choose "Unpack Parent" to display the red node's parent. At that time, the formerly red child node is no longer red, but the parent may be red if it is covering a packed parent. Children nodes may be packed away as well without manually choosing so. In that case, there is no visualization cue to indicate that the last node to be displayed in a nest is a parent node. However, zooming in on leaf nodes will expand any packed children.

If Suzie ever backtracks to a page she has visited before, and then follows a different hyperlink to the next page she'll browse, the treemap will be updated with another series of nodes coming from the revisited page [Figure 5]. Cycles in Suzie's browsing will be handled by returning to the first instance of the revisited page. For example, if Suzie visits a web page that has the same URL as one of its ancestors, instead of nesting the "new" node inside the parent, the identical ancestor will become the current node, and any links Suzie follows from that node will become another series of nodes starting from the ancestor. Similarly, if Suzie ever navigates to a node which has the same URL as an immediate sibling (another node which is linked directly from the current node's parent), instead of adding a copy of the same page as a new sibling node, the first instance of the page becomes the current node and any links Suzie follows will become children of that node. When Suzie visits a URL by entering it into the text field at the top of the browser window, that page will become a new node that is a child of the session, not of any other URL.

When Suzie wants to see more of a node in her treemap, she can click the "expand" or "unpack" menu item for that node, depending on whether the node is visible. "Expand" rolls a node out more from under its children so that more of the node can be seen, which is important when the node displays a thumbnail of the web page it represents. "Unpack Parent" forces at least one hidden node to be displayed. Suzie can also zoom in on any node for more detail by choosing the menu item "Zoom". After zooming she can return to the global view by clicking her right mouse button anywhere in the visualization window.

To allow Suzie the same navigation benefits as a standard web history widget, several features are included. When Suzie wishes to view a page in her web browser, she can choose to "Set Focus" on that node, which will bring the node into focus in the history map and will also display the page in her browser. If she begins browsing again her navigation will start from that node. Suzie can also select a node to populate an information box with more information about the web page in the control panel adjacent to the visualization. Finally, Suzie can manage her history easily by choosing to delete any subtrees or sessions which she no longer needs.

Table 1. Menu functions for Unpackable Treemaps

Menu Item	**Function**
Zoom	Forces this node and its children (or this node and its parent if the node is a leaf) to fill the visualization window.
Set Focus	Make this node the focus, which makes it and its ancestors and siblings larger than all other nodes, and brings up the selected node's URL in the web browser
Roll Up	Allow the node to show the minimum content.
Expand	Force this node to show more of its content.
Unpack Parent	Applied to a red node, this slides out the node's hidden parent so it is visible.
Pack	Pack this node with a "red" pack, sliding it under its child so it is invisible, while the child becomes red.
Set Packable	Toggle the permission for whether this node can be packed. Fails when permission should not be granted, such as for tree root and session roots.
Show All	Show every node in the tree, assigning to each node the minimum size
Delete	Delete the subtree that begins with the selected node.

5 Evaluation

I used Unpackable Treemaps throughout its development and tested the finished product with my own typical browsing sessions as well as a few with even heavier loads. I determined that Unpackable Treemaps successfully scales to approximately 200 nodes when it uses a full screen and nodes are forced to remain

large enough to display labels. Bearing in mind that the session structure is still legible in this mode, Unpackable Treemaps seems more successful than a web browser's typical history mechanism which can show 30-40 nodes in full screen mode, with none of the browsing path structure retained. When nodes are allowed to overlap such that the borders are at their minimum size, more than a thousand nodes and the names of the sessions that contain them can be seen. However, if the screen is completely stuffed such that there are no labels visible (except session labels), only the structure of sessions can be discerned. In this case it would be helpful to have colours which can be assigned to the nodes to convey information about their domain.

Beyond scalability, I will evaluate Unpackable Treemaps in terms of the tasks that I planned for the tool to perform. At the start of this project, I had 4 main goals: 1) to legibly display web page information in the nodes of a treemap 2) to offer an interactive display of the treemaps coupled with a browser 3) to handle cycles in a user's browsing and 4) to add packing and unpacking functionality to treemaps (dynamically if possible).

5.1.1 Legible Display of Web Page Information

Unpackable Treemaps displays the title of a web page and a thumbnail image of the top of the page for each node which possesses these attributes, as well as a background colour which can denote information about the domain, packing, or depth of the node. In informal sample sessions with 7 grad students and professionals who have ample web-browsing experience, the node information was sufficient for a user to pick out a previously visited node from the history display with confidence. The test users' opinion was that the addition of the screenshot made it easier than searching only on a title to remember whether a node was the page for which they were searching.

5.1.2 Interactivity

The treemap display is quite interactive, allowing the user to re-open previously visited pages in the browser, prune the tree, and sort and filter the tree. Tight coupling whereby navigation in the browser window automatically updated the Unpackable Treemaps display would be more preferable than the one way coupling currently offered.

5.1.3 Cycle Handling

My original intent in handling cycles was to visually flag that a cycle had occurred by marking nodes which were multiples. However, I found this distracting; using a background colour to indicate multiples made the multiples far too prominent in the display and using the same colour for all multiples, i.e. multiples of different pages, was confusing. A glyph added too much clutter when labels and images were being displayed. Moreover, handling the cycles as described in Section 3.4, such that if a user navigates to a copy of an ancestor page the links he follows from the page become new branches from the ancestor, depicts actual browsing more accurately. For example, if I begin browsing a web site from its main page, and then at some point use the site's "Home" link to redisplay the main page, my mental model of the browsing session is such that I have returned to the original page. If the tree didn't record this, or recorded it only by flagging that the page was a multiple, while continuing along the same branch, it would conflict with my mental model and prove confusing when I scanned the tree. Another potential solution was to add some visual flag to the leaf node which was the last node visited before returning to the original node, as shown in Figure 6, but this was not deemed helpful by me or early testers and, as mentioned above, only added clutter.

5.1.4 Packing and Unpacking

I added functionality to existing treemaps which allows nested nodes to be dynamically aggregated such that the borders of nested nodes vary according the the current view, and manually aggregated in a manner similar to regular treemaps, which hide all parent nodes. The difference is that in Unpackable Treemaps instead of being presented with a static view of either nested treemaps or regular treemaps the user is given the power to decide how she would like to view the treemap and even to view different portions of the treemap with different formats. Moreover, manual and automatic aggregation allow for views which range from regular treemap views (with all parents hidden) to regular nested treemaps, to a blown-up nested treemap with large portions of the parent nodes revealed, as well as values in between these views.

5.2 Strengths

Unpackable Treemaps is a neat, organized way to view web history. Its 3 main strengths are: it is space-saving while preserving context, it preserves the structure of session histories, and allows the user to return to previous sessions and continue browsing, and finally, it eases the cognitive load of searching for previously-visited web pages.

5.2.1 Space-Saving

Nested treemaps are themselves space-saving formats. They allow large trees to be visualized with a minimum of screen space, thus preserving more of the context within the visual panel. Because of their compactness, they are easy to quickly visually scan. They seem to be ideal for visualizing web history as they can show the whole of a browsing session in a tight space, and give more focus to leaf nodes, which are likely to be more important to the user than nodes from early in a thread, which may be home pages or search pages which are familiar to the user and only serve as springboards to new information.

5.2.2 Preserves Session History

Just like many other graphical web history visualizations, Unpackable Treemaps provides what traditional web browser history mechanisms lack: history preserved by session. This allows the user to scan her history in a way that aligns with her mental model of browsing. It also solves the problem that with traditional history a user loses any record of multiple visits to the same page and must try to remember visited pages by their domain.

5.2.3 Eases Cognitive Load

Screenshot images ease cognitive load because the user does not have to try to recall pages solely based on their title or URL. In a release of Unpackable Treemaps coupled with a commercial website, the thumbnail images could be provided by a source such as thumbshots.org[3], an online provider of free web page screenshots, which would guarantee accuracy of images as well as higher resolution.

5.3 Weaknesses

Many of this application's weaknesses spring from the short time frame in which it was constructed. Given more time, I would add several tweaks to existing features. First, I would dynamically alter the saturation of the red colour on "red" packs to indicate how many ancestors are packed behind a child. Second, I would do more extensive testing to be sure that the user can't "break" the tree with packing, unpacking, setting focus, etc.

[3] http://www.thumbshots.org

Probably because I added too many features too fast, the suite is not as robust as it should be.

Other than these quickly-fixable problems, the application has 3 main weaknesses: First, it should be coupled with a much more functional browser. Second, it needs smooth animation to preserve the user's orientation. Third, packing is not automatic, which makes visualization of more than 250 nodes problematic.

5.3.1 Limited Browser Functionality

The web browser provided with Unpackable Treemaps is not fully functional. Many types of web page, including those with JavaScript, are not able to be visited. That was a hindrance to testing the module and it leaves the application suite unusable except as a research tool. In order for real users to try Unpackable Treemaps, at the very least some code must be written to capture real session histories from a fully functioning web browser. Even more work would be required to provide tight coupling between the browser and Unpackable Treemaps.

5.3.2 Animation

Even when the strip treemap algorithm is chosen in the control panel, changes to the tree can result in fairly large changes to the treemap layout. If these changes were animated it would be easier for the user to maintain focus on the node she is currently interested as well as the context of the rest of the history.

5.3.3 Lack of Automatic Packing

When the number of nodes visualized becomes larger than about 250, if the user chooses to "show all" nodes with equal weight given to each node, most nodes will be packed enough that their labels are not visible. In this case, the treemap doesn't actually provide either focus or context, other than an idea of the structure of each session's browsing paths.

However, if some of the nodes were packed with a "red" pack, which would occlude parent nodes and gain space for children, then the remaining nodes would be able to expand and show more of their information. An algorithm should be devised to "red" pack older nodes or nodes with are closer to the session roots or, in the case where a focus has been set, nodes which are far away from the focus by some distance measure such as the date of browsing. As has been discussed in Section 3.2, packing nodes is an aggregation that is almost always going to group similar nodes. The major difficulty, of course, is deciding how the algorithm will choose which nodes to pack.

5.4 Lessons Learned

Over the course of this project I learned several things:

- It is just as easy to err on the side of too little colour as too much. Colour could have been added to distinguish more types of domains as well as favorites.
- Packing is a bigger problem than simply the graphics end. Deciding on degree of interest measures and choosing when packing was allowed was hard.
- Users are willing to do some work if they believe a tool to be useful, but it's very easy to cross the line into too much work.
- Using a toolkit to build from is useful when the time-frame is short, but the existing framework may hinder innovative efforts and stifle creativity.
- Before settling on an existing toolkit to utilize, one should check and make sure that it is well-documented.
- Milestones with dates attached are very important. Otherwise one can become mired in deliberations over a minor problem, leaving little time for more important problems ahead.

6 Future Work

There are many features of Unpackable Treemaps which could be improved or added to make it a more effective tool for web history visualization. Several have already been addressed in Section 5.3. The most important of these is integration into a mainstream web browser. In addition to the obvious benefits that Unpackable Treemaps would be available to real users and have access to real data, the additional gain would be less infrastructure required in Unpackable Treemaps for dealing with web navigation.

One further improvement which would add valuable functionality to the tool is searching. Currently the user can sort or filter the displayed treemap by features of the nodes such as date, name, and depth, but there is no method that allows the user to search by any of these attributes. That feature would prove helpful when the user remembers something about the name or domain of the web page he's looking for, because instead of searching all of the World Wide Web he could search only those sites he knows he's visited. Google has recently introduced a new product called Google Desktop Search[4] which provides this functionality, in addition to allowing the user to search his personal documents and emails.

It may also be useful to devote a background colour to identifying the user's favorites. Because these are already landmarks which the user often uses to orient himself, like the home page, making it a landmark in the treemap could help him quickly identify useful sections of the graph.

Finally, it can be argued that the nesting used in Unpackable Treemaps wastes screen space because the bottom and right borders are small enough that no additional information about the node can be conveyed there, and showing only top and left borders would be adequate for the user to discern the hierarchy of the nodes. However, testers rejected this format in early prototypes as hard to read. Figure 5 is an example of this format. It remains as a possible testing area to determine whether users would accept this form of nesting and whether it makes visual scan more difficult.

7 Conclusion

I have developed the Unpackable Treemaps web history visualization tool. This tool is a dynamic visualization of a web browser's history, maintained by session. I have presented a scenario of its use which demonstrates is usefulness. I believe that with future work it could be a very effective aid to web browsers who wish to explore their browsing history.

Reference

Benjamin B. Bederson, James D. Hollan, Jason Stewart, David Rogers, David Vick, Laura Ring, Eric Grose, and Chris Forsythe, "A Zooming Web Browser", Human Factors and Web Development, Eds.: C. Forsythe, J. Ratner, and E. Grose, Lawrence Earlbaum, New Jersey, 1998.

[4] http://desktop.google.com/

Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg, "Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies", ACM Transactions on Graphics (TOG), 21(4), pp. 833-854, October 2002.

Mark Bruls, Kees Huizing, and Jarke J. van Wijk, "Squarified Treemaps", Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000), IEEE Press, pp. 33-42, 2000.

Lara D. Catledge and James E. Pitkow, "Characterizing Browsing Strategies in the World Wide Web", Computer Networks and ISDN Systems 27, Elsevier Science, 1995.

Peter Doemel, “WebMap - A Graphical Hypertext Navigation Tool”, *2nd International Conference on the World-Wide Web*, Chicago, IL, pp. 785-789, 1994.

R. Gandhi, G. Kumar, B. Bederson, and B. Shneiderman, "Domain Name Based Visualization of Web Histories in a Zoomable User Interface", Proceedings of the Second International Workshop on Web-based Information Visualization (WebVis'00), pp. 591-598, Sep. 2000.

Matthias Mayer and Benjamin B. Bederson, "Browsing Icons: A Task-Based Approach for a Visual Web History", HCIL-200119, CS-TR-4308, UMIACS-TR-2001-85, HCI Lab, University of Maryland, Maryland, USA, 2001.

Ben Shneiderman, "Tree visualization with tree-maps: A 2-d space-filling approach", ACM Transactions on Graphics 11, pp. 92-99, 1992.

Frederic Vernier and Laurence Nigay, "Modifiable Treemaps Containing Variable Shaped Units", Extended Abstracts of the IEEE Information Visualization, 2000.

Romain Zeiliger, Claire Belisle, and Teresa Cerratto, "Implementing a Constructivist Approach to Web Navigation Support", Proceedings of the ED-MEDIA'99 Conference, Eds. B. Collis and R. Oliver, AACE, Seattle, Wa., USA, 1999.

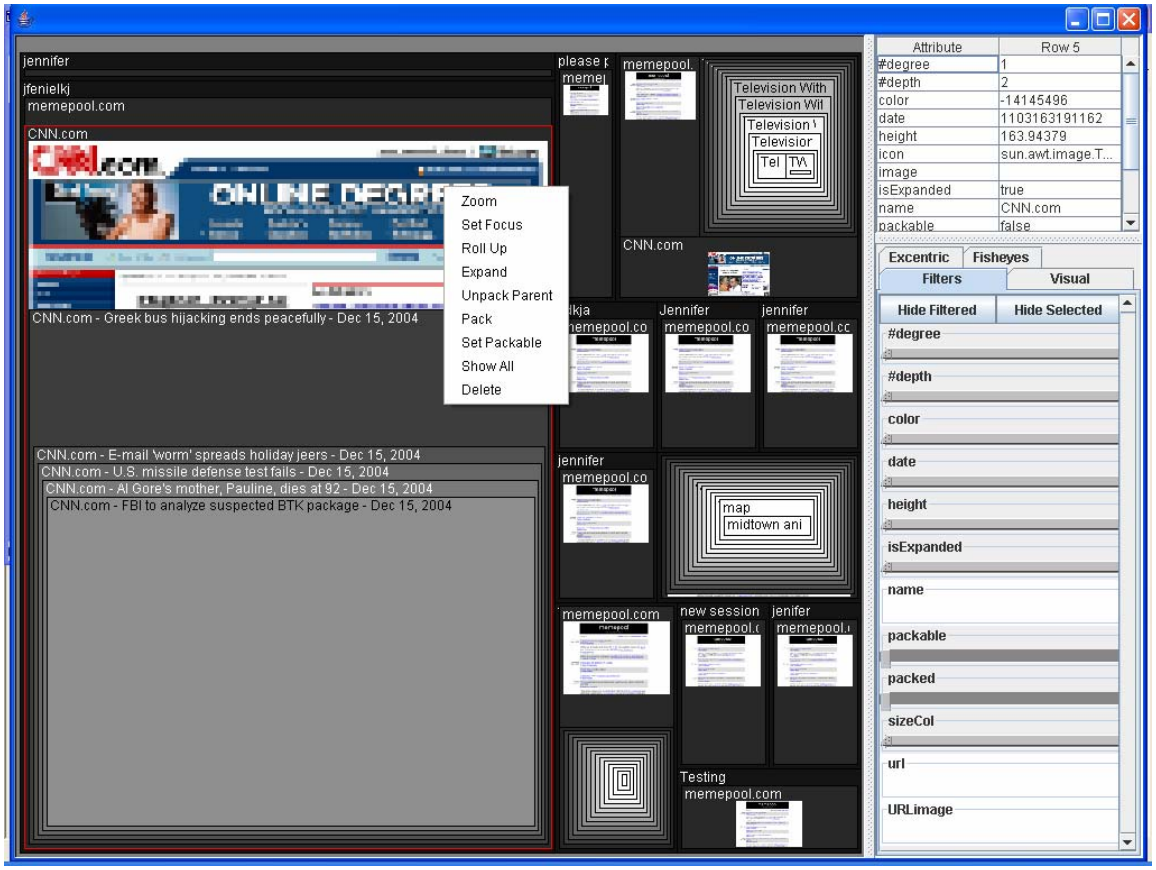

Figure 1. The Unpackable Treemaps application is displayed with a thread emanating from www.cnn.com as the focus. The popup menu is displayed with the options the user has for manipulating this node.

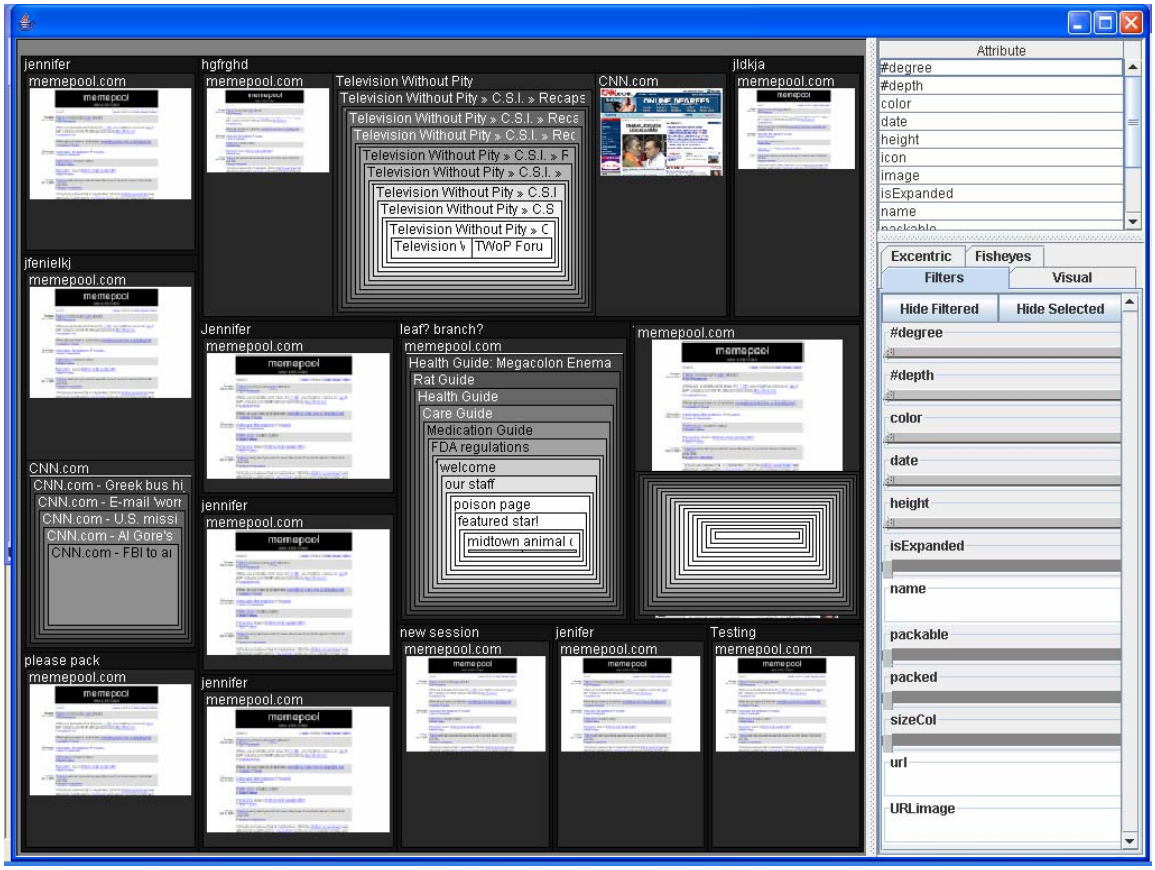

Figure 2. The treemap visual panel is displayed with the “Show All” function chosen. All nodes have equal weight. As shown, the amount of space taken up by nodes which are completely rolled up (with no label showing) and when partially rolled up (only the labels showing), is such that about 200 nodes can be shown partially rolled up when the panel is in full-screen mode, and something on the order of one thousand nodes can be shown in the fully rolled-up mode.

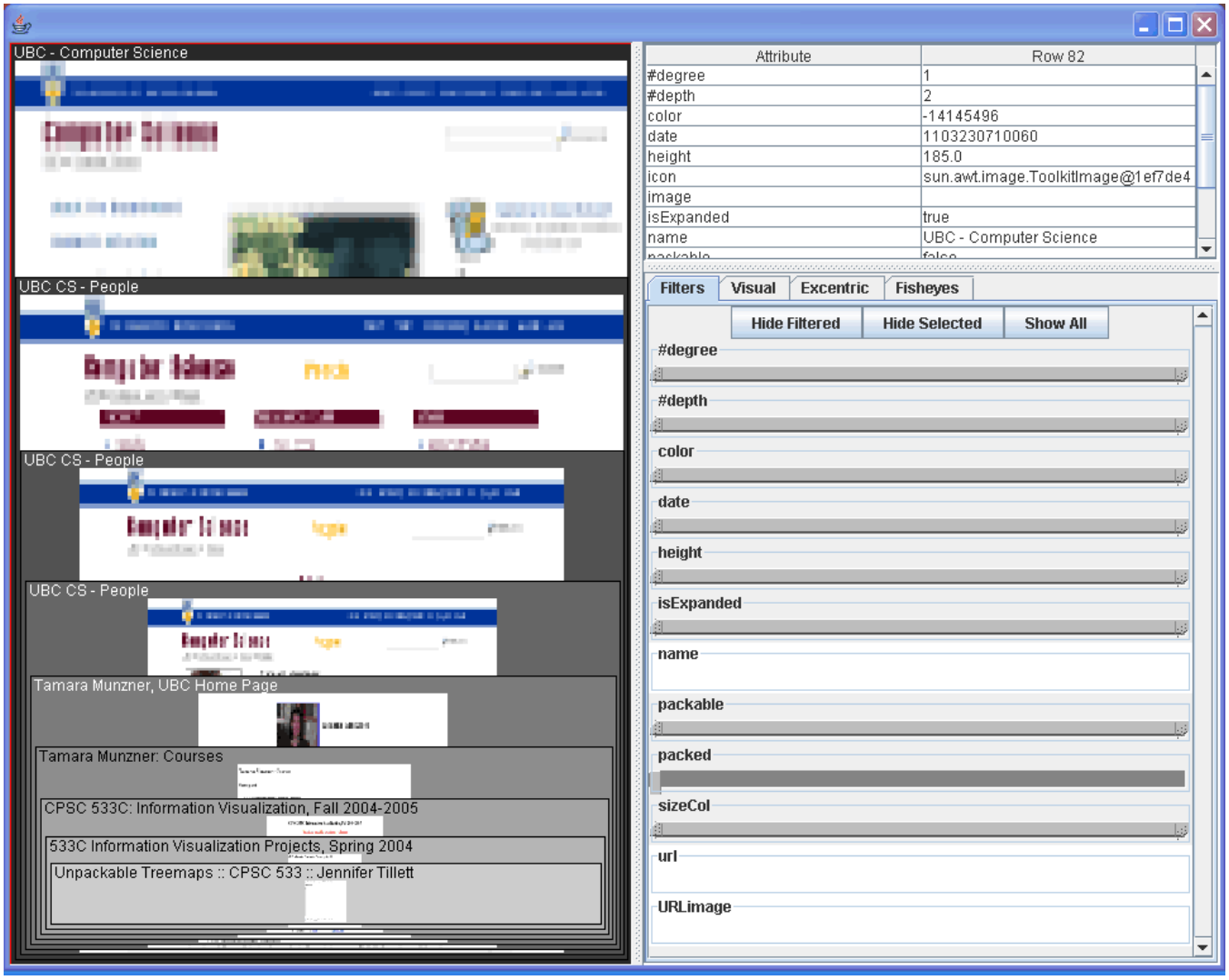

Figure 3. A zoomed-in view of nodes displaying screen shots.

Figure 4. The same images from figure 3 with one node packed away using a “red” pack.

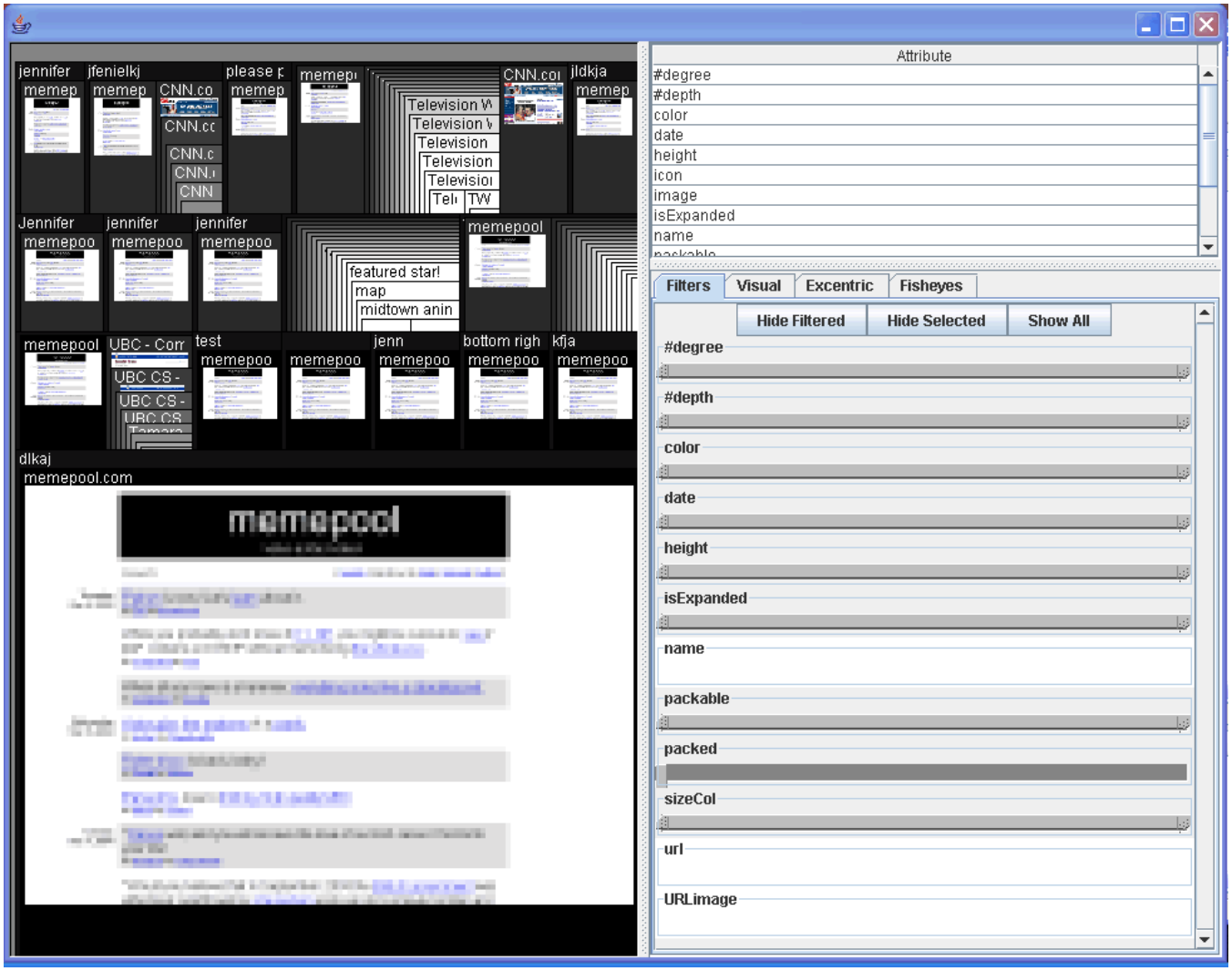

Figure 5. An example of Unpackable Treemaps when only the top and left sides of nodes are indented to indicate hierarchy.

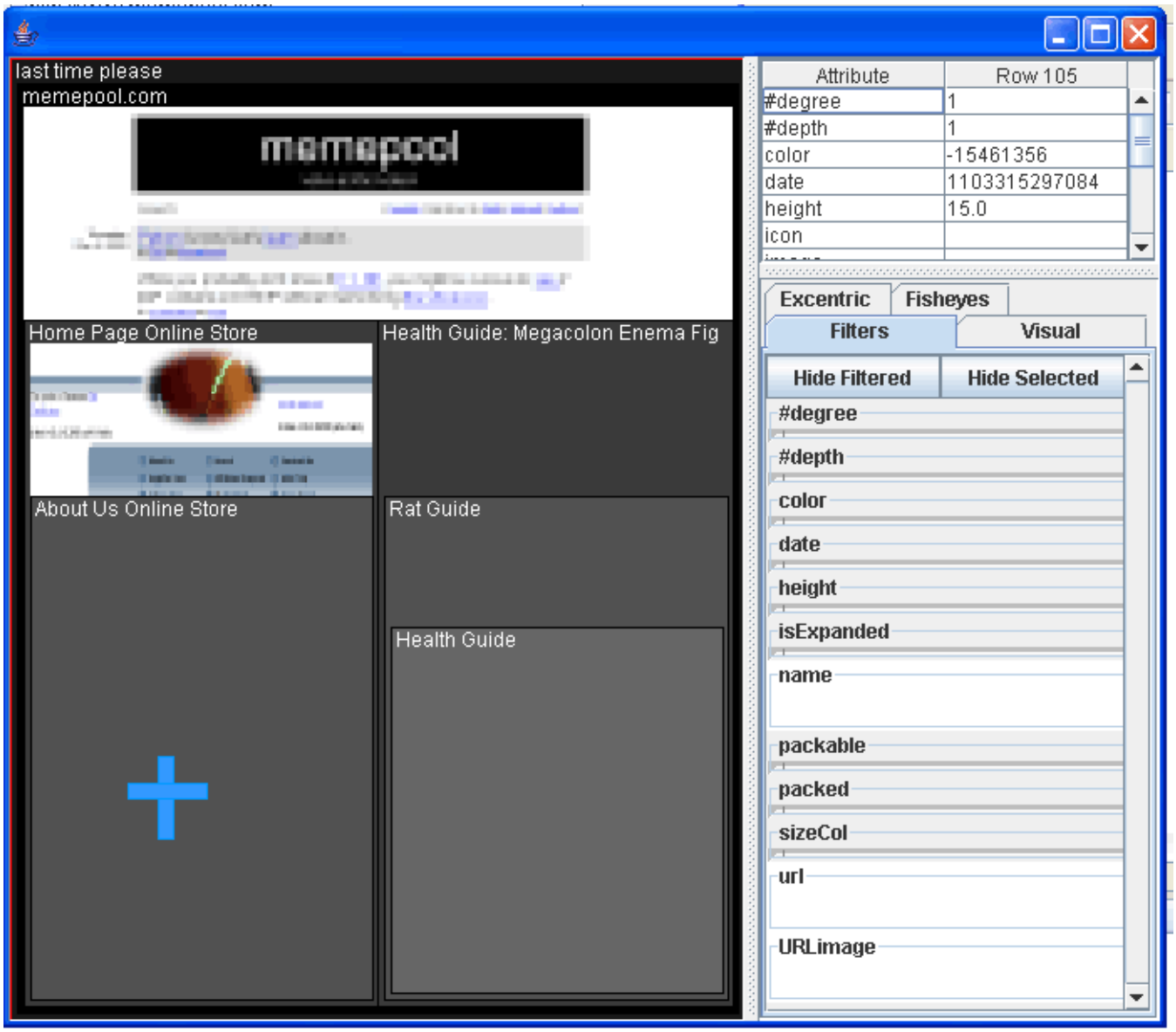

Figure 6. An example where the user has pursued two different links from the same ancestor, creating two branches. The blue cross glyph was a prototype for indicating cycles.