

# Scratchpad: Mechanisms for Better Navigation in Directed Web Searching

Dale Newfield, Bhupinder Singh Sethi\*, Kathy Ryall  
Computer Science Department  
Thornton Hall  
University of Virginia  
Charlottesville, VA 22903-2442 USA  
{DNewfield, bss4k, Ryall}@cs.virginia.edu

## ABSTRACT

Current navigation mechanisms for the World Wide Web promote a depth-first search for information on pages in hyperspace. This search strategy frequently results in the unintentional and often undesirable behavior of “web surfing” — a user starts off in search of some information, but is sidetracked by tangential links. We propose a set of mechanisms based on breadth-first traversal that are better suited for directed searching. We have implemented our ideas as a scratchpad by augmenting an existing browser. Such a system makes web navigation both faster and easier.

**KEYWORDS:** World Wide Web, information navigation, hypertext navigation, search technique, user interface

## INTRODUCTION

As the amount of information on the World Wide Web continues to grow, efficient hypertext navigation is becoming crucial. More and more people are using the web, often for *directed* searches in which the goal is to extract information about a predetermined topic. Subject-based exploration and fact-finding are two examples of directed searches. In a subject-based exploration, the user is looking to gain a basic understanding of a specific subject area for which there are usually many linked pages. Such a search may cover a broad range of related topics. In a fact-finding mission, the user is looking for some specific piece of information, but does not know exactly where to find it.

In contrast, “web surfing” is an example of *undirected* web browsing. While surfing, the user simply selects interesting links as they appear, obtaining new information without any firm goals regarding the target information. Marchionini and Shneiderman [11] note that “goals are not well defined and

change dynamically as new information is encountered by the user.”

While current navigation techniques seem to support undirected web browsing, they are often inadequate for the task of supporting directed searches. The desired information should be obtained quickly and with minimal mental effort on the part of the user. We assert that it is important for web navigation techniques to facilitate efficient hypertext traversal of a fundamentally different sort.

## MOTIVATION AND GOALS

Most current web navigation techniques are based on the paradigm of depth-first traversal [10]; users select a link on the current page which loads a new page, and displays it. This process is repeated until either the needed information is found or the current search path is abandoned. When a search path is abandoned, the user jumps to a previous page using some history mechanism such as a Back button. Catledge and Pitkow [2] refer to this process as the hub-and-spoke strategy, where the user follows a path from the start page or the *hub*, ending at a *spoke*, and then returns to the hub to start another spoke.

Many researchers have focused on efficient history mechanisms to enable the user to get back to the hub with a low number of user actions [13]. Catledge and Pitkow [2] also state that these complex history mechanisms are often not used; 41% of the user actions involve the browser’s Back button. This large number of user actions could be avoided altogether by having a navigation mechanism that eliminates the need to revisit the hub. Such a strategy would eliminate the need for the user to reread the page while also eliminating the need for the browser to redisplay the page.

Current browsing mechanisms have an inherent flaw that limits the effectiveness of any solution built on top of them. There is often more than one hub present in the traversal stack at a given point in time. When the traversal of a secondary hub is completed, the user must be able to resume the traversal at a previous level. Backtracking is often difficult since the context in which that page was being viewed has been forgotten. Frequently, entire sub-trees are accidentally

---

\* Currently at Microsoft Corporation, One Microsoft Way, Redmond WA 98052.

skipped because users do not remember what their intentions had been when an earlier hub was last being viewed [4]. Chang and Chi [3] point out that using a depth-first strategy, “the user loses more overall perspective the further he/she goes down through the layers.” Visiting any more than a few pages before revisiting some hub page may mean that the context for that page has been forgotten. Conklin [5] identifies this *cognitive overhead*, the additional effort and concentration necessary to maintain several tasks or trails at once, as a key problem with hypertext navigation. Kandogan and Shneiderman [8] note that cognitive overhead is related to a user’s browsing strategy.

Indeed, the lack of efficient mechanisms for maintaining context may be partially responsible for the phenomenon of “web surfing.” Users tend to create secondary hubs as they traverse links, and successive hubs may drift from the initial subject of the search as attention is diverted by links unrelated to the original search topic. While this drift is acceptable in undirected web browsing, it defeats the purpose of a directed search, which is to find information about a certain predetermined subject.

Our research develops techniques to improve support for directed browsing in a hypertext environment. Our goals can be divided into two categories:

### 1. Making Hypertext Navigation Faster

Reducing the number of necessary user actions can improve the speed of navigation. Since the time between user actions is currently dominated not by the will of the user, but by the time taken to load each new page, navigation speed can be further improved by reducing or eliminating this load time.

### 2. Making Hypertext Navigation Easier

Web navigation can be made easier by maintaining the context around the hypertext being viewed. In some sense, faster navigation makes it easier to maintain context. However, the mental burden on users can be further reduced by changing their thought process. Instead of thinking *back* to what they were viewing at a previous point in order to recall what their intentions had been, users should be thinking *ahead* to where they would still like to go. Rather than users having to constantly force themselves back on target, the navigation mechanism should make it difficult for them to lose track of their prior intentions. It should actively try to pull the user back to the intended subject of the search. Such context information should include a way to easily plan future searches on discovered topics, without requiring that those new searches happen immediately, and without incurring a large management burden.

## OUR APPROACH

Our approach introduces two new mechanisms to support web navigation, and two additions to existing browser technologies that result from the new mechanisms. We begin by introducing the notion of *dogears*, a revisitation mechanism that provides a set of transient bookmarks.<sup>1</sup> We then propose the use of *breadth-first traversal* as an alternative navigation strategy to depth-first traversal. This mechanism is not a revisitation mechanism, but a navigational planning tool, used not to find already visited links, but to plan future page visitations. The information gathered by breadth-first search regarding intended future page visitations opens up avenues for intelligent *prefetching* of these pages, thereby allowing faster navigation. This alternative search technique also permits us to augment existing browser interfaces to use additional *color cues* for hyperlink display, reducing the effort required to view new pages.

### Dogears

*Dogears* are the first new mechanism we propose; they serve as a set of temporary bookmarks. Visiting new links to gather information is a large component of most web browsing. At the same time, however, users may wish to return to certain pages. Revisitation can be performed using the browser’s existing history mechanisms. History lists, however, do not necessarily ensure that every visited page will be stored. Another alternative is for the user to bookmark interesting pages. Bookmarks, however, are often too persistent and place an undesirable management burden on the user.

Our solution is to introduce a new revisitation mechanism; this mechanism falls between history mechanisms and bookmarks. When people read (hard copy) books, they often turn down the corners of pages to which they may wish to return. For the electronic equivalent, if a user finds a page interesting, the page can be marked with a *dogear*. Such pages are placed in a list of dogears maintained by the browser. Dogears do not persist between sessions and are intended to act as lightweight bookmarks.

### Breadth-First Navigation

The second new mechanism we propose is an alternative navigation strategy, namely *breadth-first navigation*. In contrast to depth-first navigation, breadth-first navigation permits users to maintain more context as they search the web, especially that relating to prior intention. The basic idea is as follows:

- An ordered list of *pending* links is maintained at all times in the form of a first-in, first-out queue.
- Each time the user selects a link, it is placed at the end of the pending list. This link is *not* immediately visited.
- The browser advances to the page at the head of the pending queue once the user has finished reading the current

---

<sup>1</sup>We use the term ‘bookmarks’ to denote a large class of revisitation mechanisms. Other common terms for such revisitation lists include ‘hotlists’ or ‘favorites’.

page and selecting any interesting links. The link corresponding to the new page is then removed from the queue.

Breadth-first navigation changes the model with which users interact with hyperspace. Using depth-first traversal, each time a link is selected its corresponding page is displayed. When using breadth-first traversal, however, each time a link is selected, it is appended to the tail of the pending queue. That is, although the page corresponding to the selected link will be viewed at some point in the near future, it is not displayed immediately. New pages are visited based on their ordering in the pending queue — the link at the head of the queue is removed each time a new page is to be visited.<sup>2</sup>

There are many advantages to the breadth-first method. The new interaction paradigm enables users to escape the traditional hub-and-spoke model. While viewing a hub the user now selects any interesting spokes emanating from it. These links are added to the the pending queue, and the user need never revisit that hub again. Other benefits of the breadth-first traversal mechanism are the result of extensions to current browser technologies. Two key byproducts are a better prefetch strategy, and additional types of color cues for hypertext links.

*Prefetch* Although not a fundamental limitation of hypertext, user-perceptible network latencies are a reality of today's world wide web. Most current browsers seek to alleviate the problem by maintaining a local cache of pages. While a cache ensures quick display of revisited pages, it does not reduce the download time for new pages. Within bandwidth limitations, it may be theoretically possible to prefetch every link on a page while the user is viewing it; however, such a method is network intensive and may not be effective if there are many links on a page.

The breadth-first traversal mechanism lends itself naturally to an efficient method for prefetching web pages. While using such a navigation strategy, there is a potentially large delay between the selection of a link and the display of the corresponding page. This delay can often be used to prefetch pending pages. If users spend enough time viewing pages, the time to load subsequent pages becomes invisible. The ordering of the entries determines the order in which the pages will be displayed, and provides a precise mechanism for prefetching web pages of interest to the user.

*Color Cues* The original goal behind exploring alternative navigation techniques was to provide a browser in which users may find what they are looking for more quickly and with less wasted effort. Many browsers that support depth-first navigation use different colors for visited and unvisited links to help achieve that goal. The different link styles enable users to focus attention only on unvisited links, or to more quickly identify links they might like to revisit.

---

<sup>2</sup>A pending queue can also be used to implement a depth-first search strategy if the user always keeps the queue empty.

Similarly, altering the display style for two new categories of links that arise in a browser supporting breadth-first traversal can help reduce wasted effort by further focusing user attention. One category of links are those that have been selected and are currently in the queue pending the time at which they will be viewed. These pending links are displayed in a distinct style which indicates to the user that they do not need to be evaluated again, since they are already on the queue and the corresponding page will be displayed sometime in the near future. The other new link category arises because the user evaluates the candidacy of all links on a page before going on to the next page. It can be assumed that links left unselected at that time have been evaluated as uninteresting. When these links are seen again on a new page, they are displayed in another link style. Since these links become easier to distinguish when seen again, redundant link selection effort is avoided. We have found that it is particularly useful while viewing a set of link-intensive pages which consist largely of links to one another, but do not provide much additional content, as new links stand out from ones already seen.

Thus, using a breadth-first search strategy enables us to categorize links into four categories:

1. *never before seen*: links that have not previously appeared on a page already visited (a subset of the never before seen links in traditional browsers)
2. *visited*: links whose pages have already been visited (corresponds directly to visited links in traditional browsers)
3. *pending*: links that have been selected, but whose pages have not yet been visited
4. *uninteresting*: links that have been previously seen, but implicitly identified as uninteresting because the user did not select them before continuing on to the next page

## RELATED WORK

In this section, we present an overview of existing methods which aim to make hypertext navigation easier and faster. Most commercial browsers focus on efficient page revisitation (history lists, bookmarks) while only indirectly addressing the problem of hypertext context management. Some experimental browsers have attempted to use graphical maps to provide a sense of hypertext context; yet others have integrated the problems of context management and page revisitation more closely. Attempts have also been made to extend hypertext capabilities to provide a sense of context (HTML Frames). We now analyze each of these methods in detail.

Virtually all commercial and experimental browsers provide users with some method to revisit pages of interest. We categorize these mechanisms into two categories: history and revisitation mechanisms. History mechanisms are syntactic in nature, resulting from the navigation actions taken by the

user. Bookmarks have a more semantic quality and are explicitly created by users based on their interest level in the page.

History mechanisms provide a passive method to help users maintain some context about a particular web session. The browser stores some subset of the pages visited by the user, typically in a list. However, since most history mechanisms store only the links on the last spoke traversed (i.e., the current path in a depth-first traversal), using this mechanism, the user may only be able to reach a small portion of those pages that have been viewed. If, when deep in a search tree, the user finds an interesting page (i.e., one to which they may wish to return), it may be difficult to preserve that link while continuing with the search.

Bookmarks provide an active method for users to mark interesting web pages to which they would like to return. For most people, however, bookmarks create a new complication. Because they persist across different sessions, and because users may find a large number of pages interesting, bookmark lists quickly become long and unwieldy. Although most browsers provide mechanisms for editing and otherwise maintaining bookmark lists, they are not an ideal mechanism for maintaining context within a single browsing session.

Another technique commonly supported by browsers involves spawning a new browser window for each spoke of interest. The new browsers quickly become very difficult to manage, and even more quickly consume the user's most important resource: screen real estate. DeckScape [1], an experimental browser, seeks to avoid this problem by using the concept of *decks*. Every deck is a linear stack of pages that the user can leaf through. As with history mechanisms, if a user starts from page *A* and goes to *B*, *B* is added to that stack or deck. However, unlike history mechanisms, if the user goes back to *A* and then traverses a link to *C*, *B* is not lost; it remains in the deck of pages. The user can choose to spawn a new deck, and start a separate stack of pages on that deck. Spawning decks instead of new browsers consumes less resources. Also, the user can always revisit any page since no page is ever lost. However, users are further burdened with the responsibility of maintaining pages logically in different stacks. Decks are persistent, and hence the user can experience information overload unless decks are pruned regularly.

There are several techniques that web page developers can utilize to help users successfully browse their pages. Frames and graphical overviews both help users identify where they are in hyperspace. For example, an online manual may have a frame which always contains a table of contents, while another frame contains the specific contents of the page that the user wishes to view. Selecting a link in the contents frame brings up the corresponding material in the other frame. Frames actually provide a dynamic extension to the user interface. Well designed pages always display the hub (the table of contents page), so users may progress directly from spoke

to spoke. Unfortunately, creating a good frame set requires quite a bit of hand tuning and domain expertise.

Two types of systems present graphical maps through which a user may navigate. Some are like HTML frames in that they are created once by the content provider. Others are created on the fly by the web browser. WebTOC[12] is an automated system for creating table of contents(TOC) frames for sets of web pages. The TOC frame can be quite useful, and having that frame actually run a Java program allows it to more dynamically present the desired information. Its main drawback is that it occupies a large portion of the screen. WebNet[4] is a browser extension that displays a graphical representation of the hyperspace being explored. It does so dynamically, and independent of the content provider. In fact it can do so across many sites. It is a challenge, however, to present the graph in such a way that the contextual information is highlighted. More generally, it is not clear how much is gained by a graphical representation, or how easy it would be to internalize that information. Fisheye mechanisms[7] are among those techniques used to prevent information overload, but in any case, they are only able to show syntactic information, not semantic information.

Elastic Windows [8] also provides a mechanism that illustrates graphically the hyperspace in which a user is navigating, but it does so more interactively. If the user selects a link using this system, the contents of the corresponding page do not replace the currently displayed page; instead, the new page is displayed alongside its parent. Selecting multiple links from a page results in all the new pages being displayed alongside the parent, but in a smaller size. The same operations may be performed on any window in the browser. Users are provided with functionality to manage the windows by collapsing some sections of the hierarchy, while maximizing the size of others. Since the complete hierarchy is visible at anytime, users can easily move in the hierarchy while not losing their search context. The simultaneous display of multiple pages again places a management burden on the user; screen real estate can easily become cluttered if not managed efficiently. Cyclic links are not dealt with very well; the same page may be displayed multiple times, further wasting screen real estate. Elastic windows enables users to look at many web pages at once, whereas a breadth first navigation technique provides a linear order for web visitations.

Most of the methods discussed above either enable efficient page revisitation or provide a sense of context. While some of them (e.g., Elastic Windows) do manage to address both concerns, such methods place a undesirable management burden on the user. Moreover, none of the existing methods address other characteristics of the underlying medium such as highly scattered information and user-perceptible network latencies.

## A PROTOTYPE SYSTEM

In this section we describe our prototype system. We have incorporated the breadth-first mechanism and dogears into Chimera-1.70 [9]. The browser is written in C and runs under X-Windows using the Athena widget set. We also demonstrate the use of the system by describing a sample web browsing session.

### Prototype

Many of our additions to Chimera are encapsulated in a separate window that we call a *scratchpad* (Figure 1). The scratchpad maintains the context of the session by storing the list of pending links and the dogears. As the name suggests, the scratchpad is not meant to store information with a long lifespan; its contents exist only as long as the current session. Currently the scratchpad is implemented as a separate window in which both the pending links and the dogears are displayed as text lists. The scratchpad concept, however, does not preclude integration into the browser window, or the use of an alternate display technique, such as a site map display, or some other graphical form.

Selecting a link in the browser with the left mouse button causes the link to be added to the bottom of the pending queue in the scratchpad. Selecting a link with the middle mouse button causes the browser to traverse that link immediately using a depth-first mechanism. The motivation for still including this mechanism is discussed in the following section. A `Next` button in the browser window has been added: selecting it causes the top link on the pending queue to be popped off the queue and its page to be displayed in the browser.

Users may dogear a page by pressing the appropriate button at the top of the browser. Ideally dogearing would be done by *folding down* the upper right corner of the displayed page with the mouse. Other manipulations of the scratchpad and its contents (e.g., clearing the queue and visiting links out of order) are supported by appropriate mechanisms in the scratchpad interface.

The web page display has also been modified to include additional color cues. Links in the pending queue and links determined to be uninteresting are marked in separate colors, chosen so as not to be eye-catching. The purple color used in today's commercial browsers to indicate visited links draws less attention than the blue color used for new links. In the same manner, muted colors were selected for the new link categories. Links in the pending queue are colored forest green. Links previously indicated as uninteresting are drawn in brown. For an illustration, please refer to Figure 2.

Since the web browser now has knowledge of which pages will be viewed in the near future, it can efficiently prefetch pages prior to the time at which the user is ready to view them, removing the network delay from the interaction loop. The current implementation, however, does not yet include

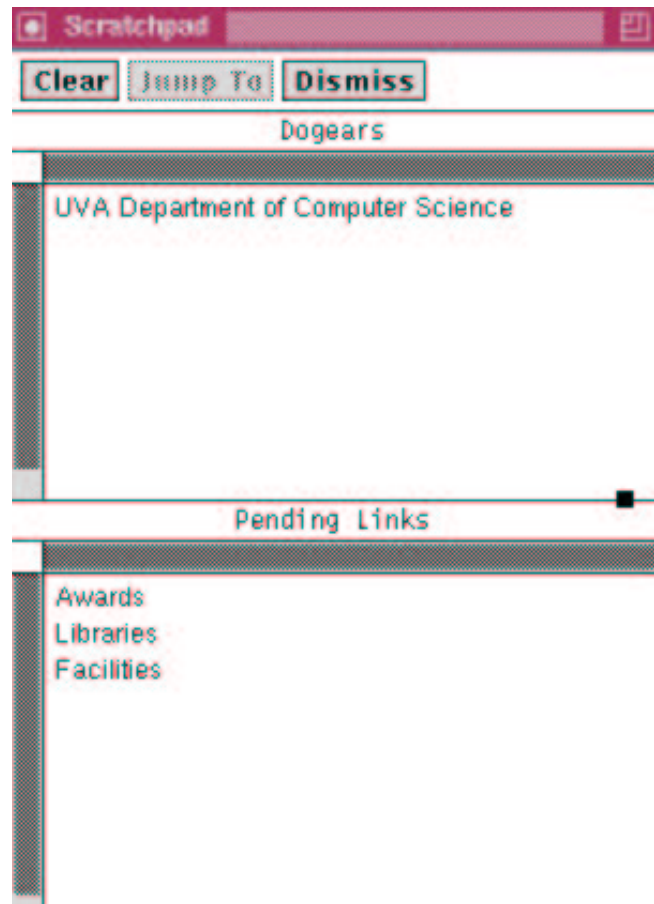


Figure 1: The scratchpad contains both pending and dogeared links.

this prefetching mechanism.

Future work will include additional functionality. Ideally, there should be a separate scratchpad for each web browsing window allowing users to maintain multiple concurrent contexts. If a page, or links on it, really represent a different logical search, a new browser-scratchpad pair should be launched to manage that new search. In addition, since the state of an ongoing search is encapsulated by the contents of the scratchpad, it should be able to be explicitly saved and restored, allowing a single conceptual web search to be continued across many interactive sessions. At present the prototype system only supports a single scratchpad, and does not implement a save and restore mechanism.

### Example Interaction

We illustrate the use of these new navigation techniques by walking through a sample web session — a directed search of a web site. Bob Bitblit, a fictitious user, is interested in finding out about the Computer Science Department at the University of Virginia. He wishes to learn about the various research projects currently in progress, as well as get a

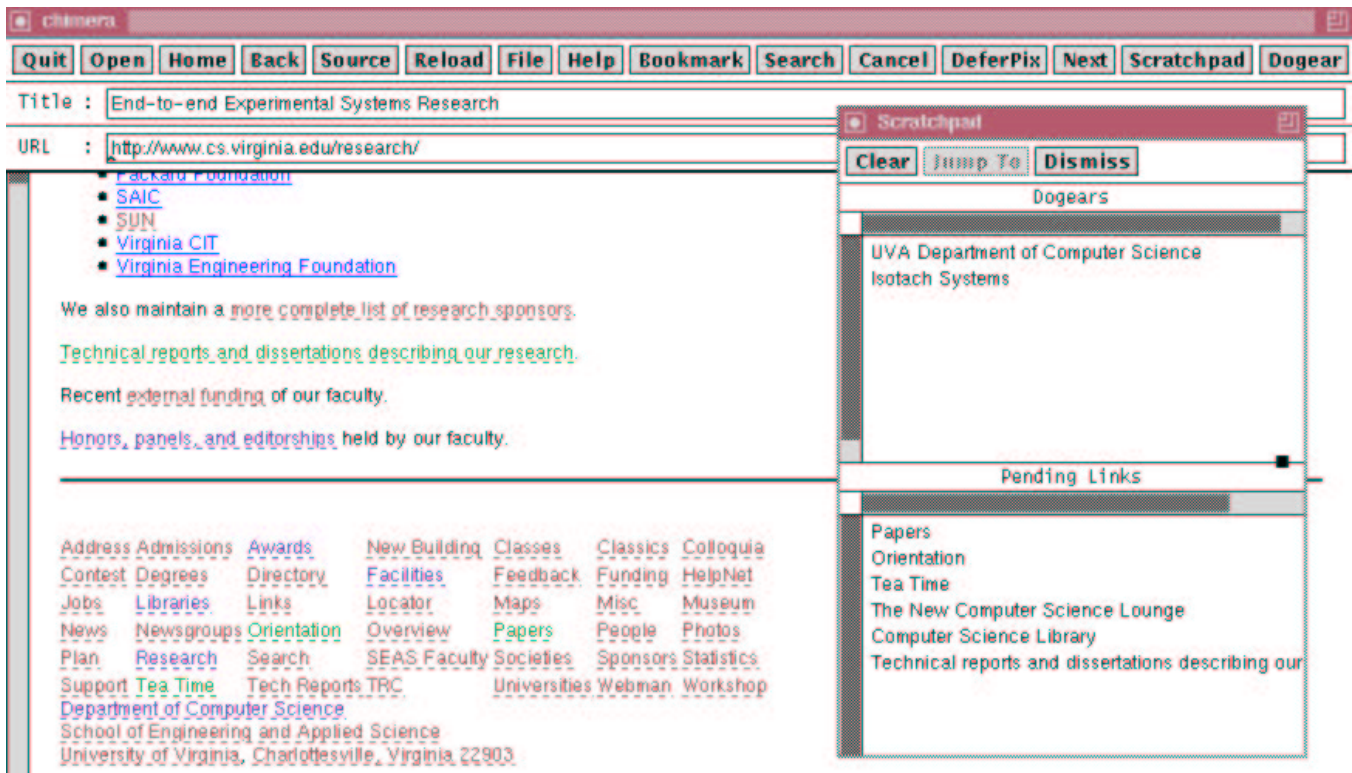


Figure 2: A full screen shot of the prototype system. Pending links (colored in forest green on the page being displayed) appear in the scratchpad window. Uninteresting links are shown in brown, previously visited links in purple, and never before seen links in blue.

general feel for the department as a whole. He begins by going to the department home page.<sup>3</sup> Upon finding the department's home page, Bob realizes that he may wish to later revisit it. He dogears this page. Next Bob scans the home page, selecting links of interest. The first few (Awards, Libraries, and Facilities), are selected just to get a feel for the size of the department. As each link is selected, it is added to the pending queue in the Scratchpad, as shown in Figure 1. Still on the department home page, Bob finds several links (Legion and Isotach) purporting to describe ongoing research topics. He selects these, as well as links titled Research and Papers. Again, as each link is selected, the corresponding page is *not* immediately visited — rather the links are added to the Scratchpad. Since Bob is also interested in learning generally about the people that work there, and the environment in which they work, he also selects links labeled Orientation, Tea Time, and The New Computer Science Lounge.

Bob then decides that he has selected enough links to satisfy his curiosity. He hits the Next button in the browser to continue his search. Pages are visited in the order they appear in the pending list portion of the scratchpad. The pages describing honors received by faculty, available libraries, and computing facilities are quickly skimmed over, except that from

<sup>3</sup>We assume that he can either guess the URL for the page, or locate it through the use of a web search engine.

the libraries page, another link to information about the departmental library is selected. Meanwhile, all the pages that were selected from the departmental home page have already been fetched by the browser. As a result, all the remaining pages will appear almost instantaneously when Bob is ready to view them. He then views the home page for the Legion Project, and after reading the introduction, decides to go on. Bob hits the Next button once again. The Isotach web page is displayed, and after reading the introduction Bob decides that he'd like to come back and investigate this group further. Since this exploration would be a conceptually different search, it should wait until the current search is complete. Bob dogears the page and goes on to the next page.

At this point, we digress to describe the effect the link coloring has on the information displayed by the browser. As shown in Figure 2, some links are bright blue, some purple, some green, and even more brown. The result is that new links (those colored blue) stand out from the rest. As with purple, the two new colors both indicate that the interest level of the associated links has already been determined. They show that Bob has already evaluated whether or not the associated information is important. Reevaluation would be wasteful; the new link types and corresponding colors make it easy to skip over these links.

Bob continues his web browsing session. He visits in turn each of the remaining pages in the queue. At each page, he

takes in whatever information he finds relevant by answering one simple question: "Which links look interesting?" He does not have to remember which other pages he had still intended to visit. Those intentions are indicated to the browser as soon as they are formed. When the scratchpad is empty, Bob has completed his initial search. He does not have a nagging feeling that he missed something — he was able to see every page that he decided would be of interest, and did not have a chance to forget to go back to one of them.

At the end of his search, any further exploration Bob had intended to pursue is contained in the dogears. Each dogear may either be revisited, and a new exploration begun, or transferred to long term bookmarks if he decides he would always like to be able to get back to those pages.

## DISCUSSION

Pure breadth-first traversal can be too restrictive in many cases. For example, consider a search engine query that results in a page with twenty hits on it, and a link to the next twenty hits. One might want to put ten of the first twenty hits into the scratchpad, and five from the next twenty hits. A breadth-first mechanism makes it difficult to perform this operation. The user would easily be able to add ten links from the first page to the scratchpad, but not all the interesting links from both pages before beginning the process of searching through the pages from the first page of hits. The search engine, therefore, would have to be revisited during the search. For any linear information that is presented in an artificially hierarchical or sequential manner, the breadth-first technique breaks down. A depth-first traversal mechanism actually works better in these cases. Therefore, we believe browsers should still support depth-first navigation as a secondary mechanism so that it may be used where needed. For this reason, our implementation provides both mechanisms. As in many situations, using a hybrid solution to avoid undesired tradeoffs is preferred.

While most current browsers do not include any prefetching mechanism at all, and the inclusion of any such mechanism would be beneficial, no prefetching technique can perfectly predict the actions of users. As such, the mechanism we have proposed is not perfect. Its effectiveness is dependent on the nature of the search. The first page visited in any search is never prefetched. Subsequently, the amount of prefetching possible depends on the size of the pending list and the time that the user spends on each page. Depending on these factors, it may not always be possible to finish fetching all pages before the time at which they are to be viewed. This problem may also be exacerbated by latency and bandwidth aspects due to the network links over which the data is to be transmitted; our research does not address such problems. Finally, using breadth-first navigation to gain a prefetch queue may not work perfectly since the target information may be found before the queue is empty, in which case pages may have been fetched that will never be viewed.

As previously mentioned, prefetching has not yet been implemented in our browser. In fact there are several problems with the Chimera user interface that prevent it from being suitable for extended use. We plan to reimplement our modifications by building a new prototype based on the Netscape Open Source browser [6]. In this system the scratchpad and the browser will be more tightly coupled. The resulting system should be robust enough to be tested by users, and will provide a better environment in which to evaluate our techniques.

## CONCLUSION

We have argued that current browsing techniques do not adequately support directed searching. The proposed navigation techniques simply add to existing technologies, and promote a breadth-first traversal mechanism rather than the depth-first mechanism that prevails today. This search strategy makes navigation faster by reducing the number of user actions required, and reducing the time between user actions by hiding the time taken to load the next page. In addition, dogears provide users with a set of transient bookmarks. We have described a prototype system that incorporates these ideas in a browser called Chimera, and shown an example interaction which highlights some of the benefits of our techniques.

An important contribution of this work is that it proposes a technique to remove the burden of remembering context, especially that relating to prior intention, from the user. Previous work which maintains context (e.g., DeckScape or Elastic Windows) has placed a significant burden of managing that context on the user. Our system manages context while placing very little load on the user, and therefore provides users with an easier hypertext navigation experience.

## REFERENCES

1. Marc H. Brown and Robert A. Shillner. Deckscape: An experimental web browser. In *The Third International World-Wide Web Conference: Technology, Tools and Applications*, Darmstadt, Germany, April 1995.
2. Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world-wide web. In *The Third International World-Wide Web Conference: Technology, Tools and Applications*, Darmstadt, Germany, April 1995.
3. Yee-Hsiang Chang and Ellis Chi. Htgraph: A new method for information access over the world wide web. In *DAGS'95: Electronic Publishing and the Information Superhighway*, Boston, Massachusetts, May 1995.
4. Andy Cockburn and Steve Jones. Which way now? analysing and easing inadequacies in www navigation. *International Journal of Human-Computer Studies*, 45(1):105–129, July 1996.

5. J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41, 1987.
6. Netscape Communications Corporation. Netscape open source. On-line at <http://www.mozilla.org/>.
7. G. W. Furnas. Generalized fisheye views. In *Human Factors in Computing Systems CHI '86*, pages 16–23, Boston, April 1986.
8. Eser Kandogan and Ben Shneiderman. Elastic windows: A hierarchical multi-window world-wide web browser. In *UIST 97*, pages 169–177, Banff, November 1997.
9. John Kilburg, Gerhard Niklasch, and Winston Edmond. Chimera-1.70, x11/athena world-wide web client. On-line at <http://www.unlv.edu/chimera/>.
10. Henry Lieberman. Autonomous interface agents. In *CHI '97*, pages 67–74, Atlanta, March 1997.
11. G. Marchionini and B. Shneiderman. Finding facts vs. browsing knowledge in hypertext systems. *IEEE Computer*, 21(1):70–80, 1988.
12. David Nation, Catherine Plaisant, Gary Marchionini, and Anita Komlodi. Visualizing websites using a hierarchical table of contents browser: Webtoc. In *3rd Conference on Human Factors and the Web*, Denver, June 1997.
13. L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies*, 1997.