Semantics of Links and Document Structure Discovery

John R. Punin
puninj@cs.rpi.edu
http://www.cs.rpi.edu/~ puninj
Department of Computer Science, RPI, Troy, NY 12180.

M. S. Krishnamoorthy
moorthy@cs.rpi.edu
http://www.cs.rpi.edu/~ moorthy
Department of Computer Science, RPI, Troy, NY 12180.

ABSTRACT

This paper presents a novel algorithm to discover the hierarchical document structure by classifying the links between the document pages. This link classification adds metadata to the links that can be expressed using Resource Description Framework Syntax [7]. Several well-known programs automatically generate HTML web pages from different document formats such as LaTeX, Powerpoint, Word, etc. Our interest is in the intertwined HTML web pages generated by the LaTeX2HTML program [6]. We use the web robot of the WWWPal System [11] to save the structure of the web document in a webgraph. Then the web analyzer of the system applies our algorithm to discover the semantics of the links and infer the hierarchical structure of the document.

1. INTRODUCTION

The semantic information of a document is conveyed by its logical structure. Suppose we are given a collection of URL's of a web document, along with the *up*, *previous*, and *next* links, and we are asked to determine the hierarchical structure of the document. This task can easily be accomplished using some standard traversal of the web document. However, if all the links in the web document are classified the same, the problem of finding the document's hierarchical structure is not trivial. Our hierarchical structure finding algorithm solves this nontrivial problem and discovers the logical structure.

Several well-known programs automatically generate HTML web pages from different document formats such as LaTeX, Powerpoint, Word, etc. Our interest is in the HTML web pages generated by the LaTeX2HTML program [6]. Our algorithm is also applicable to HTML documents produced by Powerpoint.

This hierarchical discovery problem is related to finding a Hamiltonian path in a webgraph. Linearization resulting from following the *next* links from the starting web page corresponds to a Hamiltonian path. Of course not all Hamiltonian paths are meaningful linearizations.

In recent years, a number of software systems have been developed for the analysis of web pages, such as Mapuccino [8] [9], Microsoft Site Analyst [12] and WWWPal [11]. WWWPal is different because it can handle large graphs (called webgraphs), has a better display that uses clustering algorithms, and has a skeletal graph browser. One of WWWPal's functions is to visualize the output XGMML file from the web robot into a graph that represents the structure of the website. Several drawing algorithms have been implemented to visualize webgraphs that produce nice graph layouts. The graph visualization for web documents generated using the LaTeX2HTML program is not able to layout the hierarchical structure of the underlying web document. See Figure 1. Our algorithm discovers the structure by classifying the links and then using the semantics of the links for discovering the hierarchical structure of the web document.

In the following sections, we describe the problem statement, our algorithm, examples and the report of link classification using RDF.

2. PROBLEM STATEMENT AND ANALYSIS

2.1 Common Characters in LaTeX2HTML websites

After several LaTeX2HTML websites (web documents generated by the LaTeX2HTML program) are examined and the XGMML files are generated by a web robot, some insights for the LaTeX2HTML websites are obtained. Most of the LaTeX2HTML websites contain the following nodes/pages:

1. Table of Contents Node: Most of the LaTeX2HTML documents have this node at the top, which we can call the root. It lists all the sections the site covers much like a book's table of contents.
2. Index Node: This node functions like the book's index pages, one can go to this node and find the terminologies one is interested in. Then one can go directly to the content related to the term through the link from the Index Node.
3. Bibliography Node: This node contains the information of the author or some related materials and links.
4. Start Page Node: This is not the same as the Table of Contents Node, but it includes the same links as the Table of Contents Node. It is the initial page of the whole document.

3. ALGORITHM FOR HIERARCHICAL STRUCTURE DISCOVERY

Edges in a webgraph represent the links between web pages. These links can have a type such as: *start*, *next*, *prev*, *chapter*, etc. A group of these standard types has been recommended for HTML 4.0 [10]. When a webgraph has a hierarchical structure, we would like to visualize the webgraph as a radial tree drawing so the hierarchical structure of the document can be clearly seen. For example, the webgraph of Figure 1 and Figure 2 have the same structure. We can see that the hierarchical structure of the webgraph is not evident in Figure 1, but it is easily visible in Figure 2. The only difference between these two drawings is that in Figure 1 the type of the edges are not considered for visualization, and in Figure 2 the links of type *chapter* and *section* are visualized as tree edges. In Figure 2 we can notice that node 5 is a *chapter* of Node 1, and nodes 16 to 20 are *sections* of node 5. We can also notice that each of the children of node 1 have subtrees, and they are linked together through *next* and *prev* links. Figure 3 shows how the subtrees of node 1 are linked with the links of type *next* and *prev*.

There are many webgraph instances where the type of the links is missing. So we have to apply an algorithm to assign a type to each link of the webgraph. In this subsection we will explain a novel algorithm to classify the links of a webgraph that represents the structure of a web document. Our algorithm will only assign these four types of links: *chapter*, *section*, *next* and *prev* The following rules assign types to the edges of the webgraph:

- The children of the root node that also have a back link are considered node chapters. Therefore the edges from the root node to those children are classified as *chapter* edges.
- Our algorithm finds the subtree of the *chapter* nodes. The links between these subtrees are classified as *next* and *prev* links.
- Our algorithm orders these subtrees so there is a linear order of all nodes of the webgraph. This linear order can be considered the Hamiltonian path of the webgraph. This order can be used, for example, to make a linear print out of the web document.
- The subtrees of the webgraph are considered *chapters* of the document and each of the *chapters* have *sections*. Hence the previous rules can be applied to these *chapters* in order to find the subtree *sections*.

These rules will be applied until all edges are classified so the *chapters* and *sections* are fully resolved. For the purpose of visualization the *chapter* and *section* edges are mapped to tree edges and a radial tree drawing is applied (Figure 2).

The classification of the edges (or links) algorithm has two steps: first, the ordering of the children nodes of the root tree, and second, the traversal of the tree by retrieving the *next* node. The first task helps us to find the first child node and find the *chapter* and *section* edges; the second task linearizes the webgraph and finds the *next* and *prev* edges.

The first task is implemented in the function `get_first_child`. This function receives the root node (Start Page Node) of the subtree and returns the first of its children nodes. We

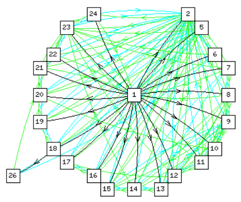


Figure 1: Radial Tree Drawing of a Web Document/webgraph

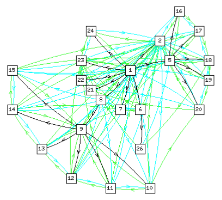


Figure 2: Typed Links Drawing of a Web Document/webgraph

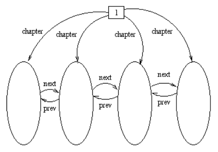


Figure 3: Subtrees of Node 1

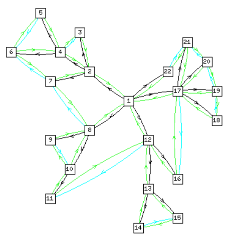


Figure 4: Example of the Typed Links Drawing

Node	Neighbor	Count	Count	Neighbor
2	8	1	—	—
8	2	3	12	1
12	17	1	8	2
17	12	2	22	1
22	17	4	—	—

Table 1: Neighbor nodes of the children of the root node and the counts

found that if we apply Breadth First Search (BFS) starting at any of the children nodes of the root node, the next child node is visited at most one time and the previous node is visited at least one time. For example, in Figure 4, the children of the root node 1 are nodes 2, 8, 12, 17 and 22. If we apply BFS starting in child node 12, node 8 is visited two times and node 17 is visited once. We can conclude that node 8 is the previous node and node 17 is the next node of node 12. With this information a table is constructed (Table 1) with the number of times that BFS visited the next and previous nodes. Using this table we can conclude what neighbors are previous and next to each children node of the root node 1. Previous nodes are the ones whose counts are greater than one and next nodes are the ones whose counts are exactly one. Table 2 shows the next and previous node for the children of the root node 1 (Figure 4). We can easily see that the first child of node 1 is node 2. Notice that the first node 2 does not have a previous node. However, we have to verify that there is a linear ordering between all the children of the root node 1 to conclude that node 2 is indeed the first node. If a first node is not found, the function `get_first_child` returns a null node and a failure error code.

The algorithm of the second step uses the `get_first_child` function to retrieve the first node of the current visited node. If first node is not found, it means that the current node is a leaf node. Hence, the next node is the neighbor node that has not been visited yet. This algorithm produces a linear ordering of the nodes of the webgraph. The edges between the current node and the not-visited neighbors are classified as *next* and *prev* edges. The edges between the current node and its children are classified as *chapter* and *section* edges. Table 3 shows the algorithm of the second step of the webgraph traversal. Once all edges of the webgraph have been assigned a type, the *chapter* and *section* nodes are mapped to tree edges (black color), the *next* edges are mapped to forward edges (magenta color), and the *prev* edges are mapped to backward edges (green color). Afterwards, a tree or radial tree drawing can be applied to visualize the hierarchical structure the webgraph. (Figures 2 and 4).

Node	Previous	Next
2	—	8
8	2	12
12	8	17
17	12	22
22	17	—

Table 2: Table of previous and next nodes for the children of the root node

```
// Linear traversal of the webgraph
int linear_traversal(PNODE root)
{
int error = 0;
PNODE next = root;
// chapter edges
classify_chapter_edges(root);
// Linear traversal
while(next && error) {
next = get_next_vertex(next, &error);
}
return error;
}
```

```
// Get next node in linear traversal
PNODE get_next_vertex(PNODE v, int *error)
{
PNODE next;
mark(v);
// get first child of children nodes
next = get_first_child(v, error);
// section edges
if(next && *error)
classify_section_edges(v);
// leaf nodes
if(!next && *error) {
next = get_unmarked_neighbor(v,error);
// next and previous edges
if(next && *error)
classify_next_prev_edges(v,next);
}
return next;
}
```

Table 3: Algorithm for classification of the edges of a webgraph

Our algorithm fails when all the counts of the table of neighbor nodes (Table 1) are one since we will not be able to construct the table of previous and next nodes (Table 2). Without that table we cannot conclude which node is the first node. For example, Figure 5 shows a simple hierarchical webgraph where we cannot infer what node is the first node; it is either node 2 or 8. When we construct the table of neighbor nodes all counts are one as shown in Table 4.

Our algorithm is of linear time complexity as each node gets visited a constant number of times.

4. REPORT OF LINK CLASSIFICATION USING RDF

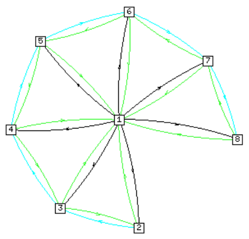


Figure 5: An example of a Web Document/webraph where the Algorithm fails.

Node	Neighbor	Count	Count	Neighbor
2	3	1	—	—
3	4	1	2	1
4	3	1	5	1
5	4	1	6	1
6	5	1	7	1
7	8	1	6	1
8	7	1	—	—

Table 4: Neighbor nodes of the children of the root node and the counts - All being equal to 1 makes the algorithm fail

The algorithm described in the previous section attaches semantics to the links and hence the web pages. One may question the usefulness of the algorithm if the semantics of hyperlinks, such as *up*, *next* and *prev* is known apriori. We envisage that explicitly specifying the links in a common semantic vocabulary will be useful to both the users and the agents.

As an example, the RDF description for node 9 (`http://cbl.leeds.ac.uk/ nikos/doc/www94/subsection3_4_1.html`) of the webgraph of the Figure 3 is as follows:

```
<rdf:RDF xmlns:rdf=''http://www.w3.org/1999/02/22-rdf-syntax-ns#''
         xmlns:rs=''http://purl.org/net/rdf/papers/sitemap#''
         xmlns:dc=''http://purl.org/dc/elements/1.1/''>
   <rdf:Description rdf:about=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html''>
      <dc:title>Common Objections to Automatic Conversion</dc:title>
      <rs:prev rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/section3_4.html''/>
      <rs:next rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_2.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_3.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_4.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_5.html''/>
      <rs:section rdf:resource=''http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_6.html''/>
      <rs:contents rdf:resource =''http://cbl.leeds.ac.uk/~nikos/doc/www94/tableofcontents3_1.html''/>
    </rdf:Description>
</rdf:RDF>
```

The RDF triples in N-triples format [2] are:

```
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/dc/elements/1.1/title>
"Common Objections to Automatic Conversion" .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#prev>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/section3_4.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#next>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_1.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_2.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_3.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_4.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_5.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#section>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsubsection3_4_1_6.html> .
<http://cbl.leeds.ac.uk/~nikos/doc/www94/subsection3_4_1.html>
<http://purl.org/net/rdf/papers/sitemap#contents>
<http://cbl.leeds.ac.uk/~nikos/doc/www94/tableofcontents3_1.html> .
```

Notice that we use the Dublin core [DC] and the RDF Sitemap vocabulary [RS].

5. EXAMPLE

We illustrate the output of our algorithm on a webgraph, obtained from visiting the LaTeX2HTML website "State of the Art Review on Hypermedia Issues And Applications" at `http://cbl.leeds.ac.uk/nikos/tmp/hypemedia/hypemedia.html`.

Figure 6 shows the radial tree drawing of the web document, and Figure 7 shows the hierarchical drawing of the same web document, where the *chapter* nodes and *section* nodes can be observed.

6. CONCLUSION

This paper is yet another important contribution to the Semantic Web [1][7][3][4] as we provide semantic classification of links and the web pages of a LaTeX2HTML website. This paper also provides a visualization tool based on our algorithm. This visualization not only draws a pretty graph, but also semantic content is imparted. Our algorithm can be extended for other types of documents.

Printing a web document, consisting of a number of web pages, is a laborious task [1], as each of the web pages has to be traversed in a linear order. This linearization is obtained automatically from the hierarchical structure order discovered by our algorithm. WWWPal uses this linearization to simplify the printing of web documents.

7. REFERENCES

[1] T. Berners-Lee, Semantic Web Area for Play: Closed World Machine. `http://www.w3.org/2000/10/swap/Overview.html`, February 2001.

[2] T. Berners-Lee, Notation 3. `http://www.w3.org/DesignIssues/Notation3.html`, April 2001.

[3] D. Brickley, RDF sitemaps and Dublin Core site summaries, `http://purl.org/net/rdf/papers/sitemap/02`), June 1999.

[4] D. Brickley, Biz/ed RDF Metadata Testbed, `http://ilrt.org/discovery/2000/08/bized-meta`, July 2001.

[5] O. De Troyer and T. Decruyenaere, "Conceptual Modelling of Web Sites for End Users," WWW Journal, Vol.3 , Issue 1, Baltzer Science Publishers, 2000.

[6] N. Drakos, "From text to hypertext: A post-hoc rationalisation of LaTeX2HTML," In Proceedings of the First World Wide Web Conference, Geneva, Switzerland, 1991.

[7] O. Lassila and R. Swick, W3C, Resource Description Framework (RDF) Model and Syntax Specification. `http://wwww.w3.org/TR/REC-rdf-syntax`, 1999.

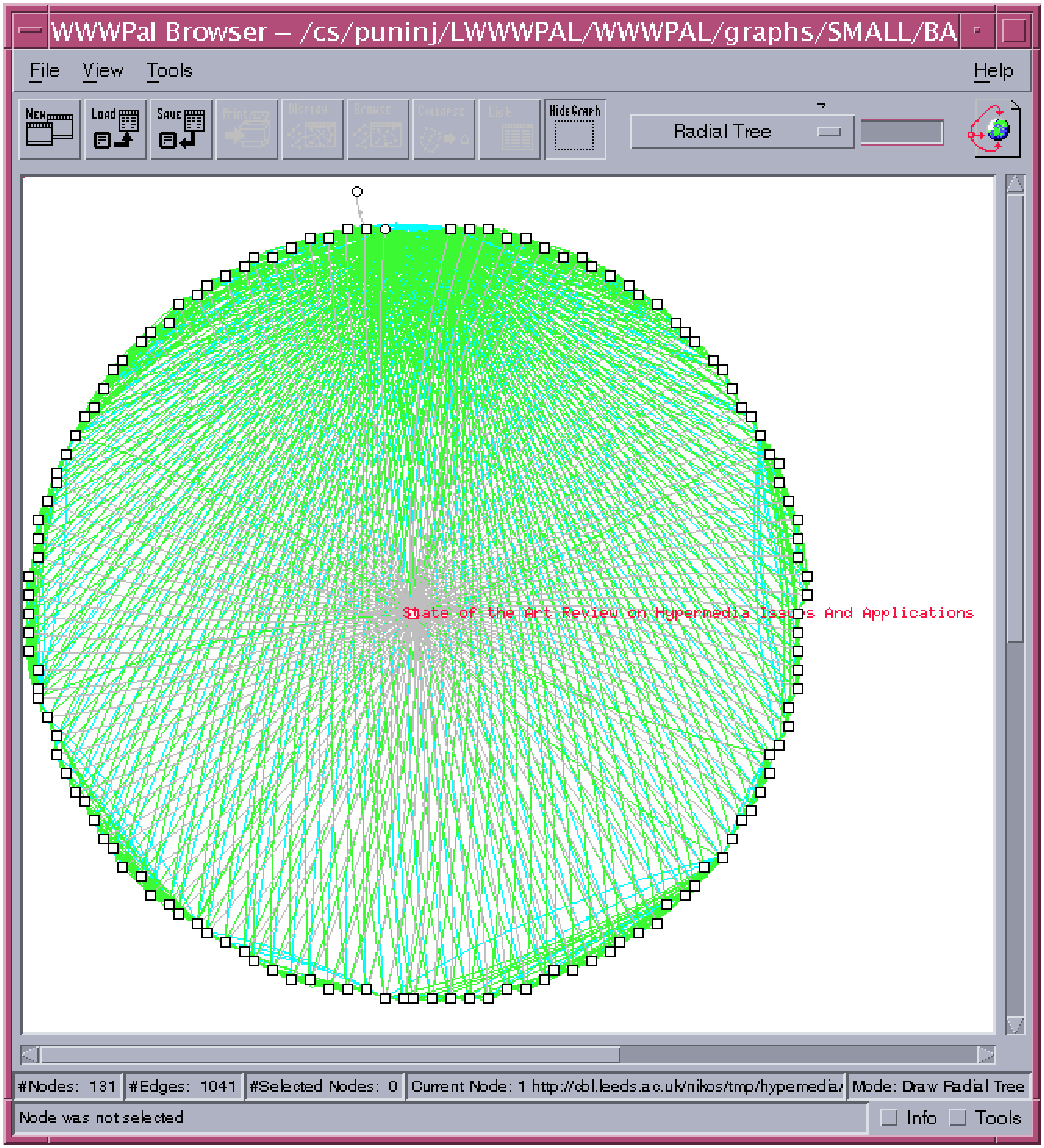
[8] Mapuccino `http://www.ibm.com/java/mapuccino/`

[9] Y. Maarek and I. Shaul, "WebCutter: A System for Dynamic and TailorableSite Mapping," In Proceedings of WWW 6 Conference, Santa Clara, USA, 1997.

[10] D. Raggett et al, HTML 4.01 Specification, `http://www.w3.org/TR/html401/` , 1999.

[11] J. Punin and M. Krishnamoorthy, "WWWPal System - A System for Analysis and Synthesis of Web Pages", In Proceedings of the WebNet 98 Conference, Orlando, November, 1988.

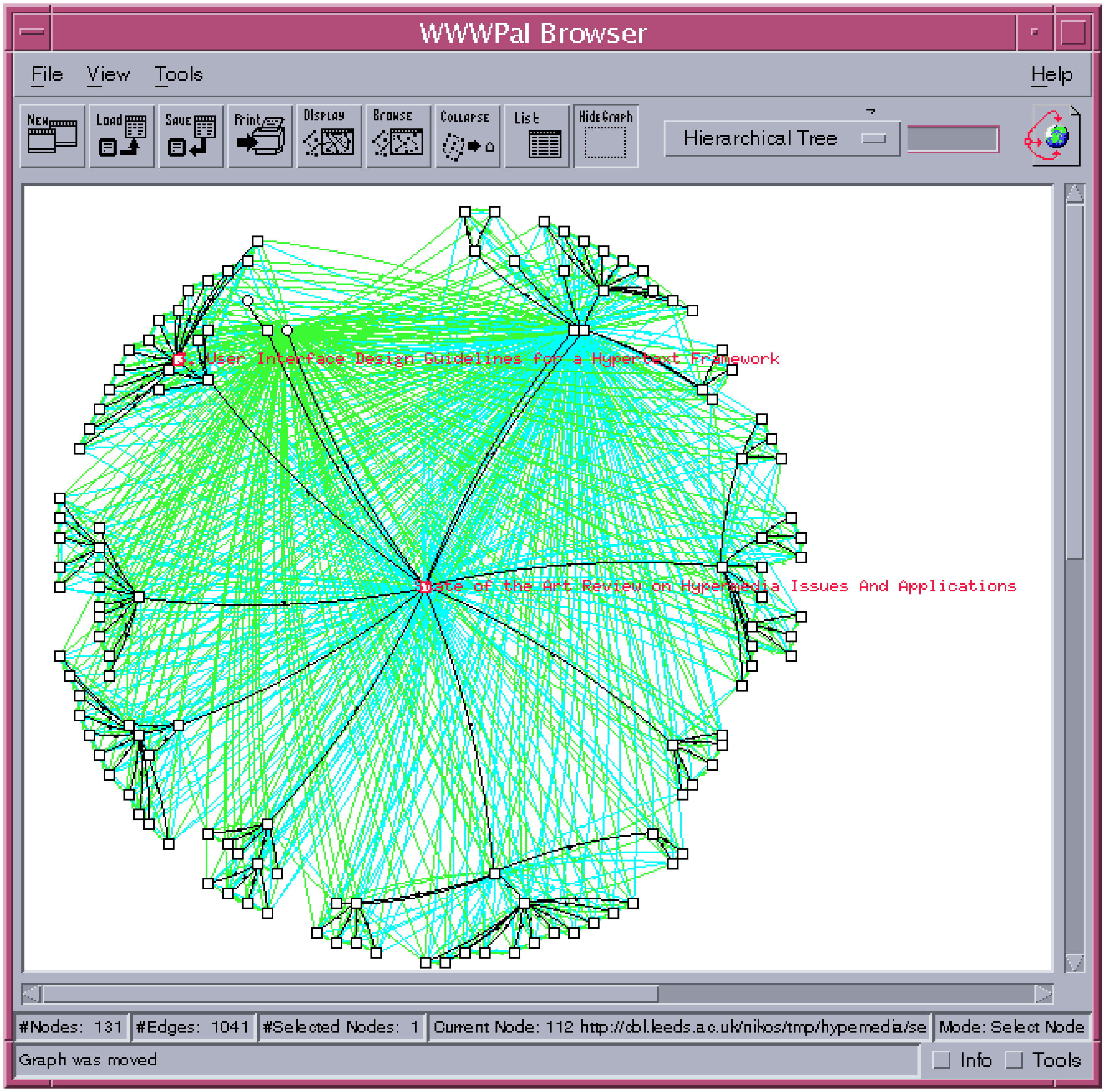
[12] M. Tabini et al, Professional Site Server 3.0, Wrox Press Inc, July 1999.


WWWPal Browser – /cs/puninj/LWWWPAL/WWWPAL/graphs/SMALL/BA
File
View
Tools
Help
New
Load
Save
Print
Display
Browse
Collapse
List
Hide Graph
Radial Tree
State of the Art Review on Hypermedia Issues And Applications
Info
Tools

#Nodes: 131 | #Edges: 1041 | #Selected Nodes: 0 | Current Node: 1 http://cbl.leeds.ac.uk/nikos/tmp/hypermedia/ | Mode: Draw Radial Tree

Node was not selected

Figure 6: Radial Tree Drawing of a Web Document


WWWPal Browser
File
View
Tools
Help
New
Load
Save
Print
Display
Browse
Collapse
List
HideGraph
Hierarchical Tree
User Interface Design Guidelines for a Hypertext Framework
State of the Art Review on Hypermedia Issues And Applications
Info
Tools

#Nodes: 131 | #Edges: 1041 | #Selected Nodes: 1 | Current Node: 112 http://cbl.leeds.ac.uk/nikos/tmp/hypemedia/se | Mode: Select Node

Graph was moved

Figure 7: Hierarchical Tree Drawing of a Web Document