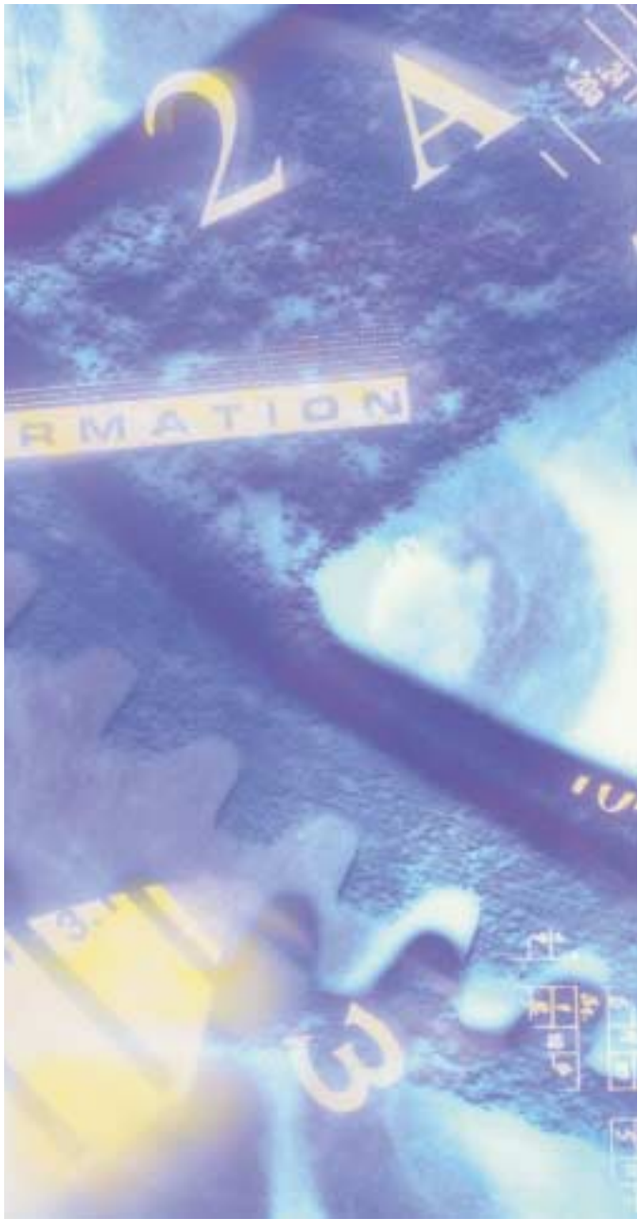
Structural Web Search Using a Graph-Based Discovery System



Our structural search engine searches not only for particular words or topics, but also for a desired hyperlink structure.

A structural search engine searches not only the text, but also the structure formed by sets of linked Web pages, to find matches to a query.

The Web consists of hyperlinks that connect one page to another, and the graph structure formed by these hyperlinks contains an enormous amount of information. Specifically, the creation of a hyperlink by the author of a Web page represents a relationship between the source and destination pages. Such relationships may include a hierarchical relation between a top-level page and a child page containing more detailed information; relationships between the current, previous, and next sites in a sequence of pages; or an implicit endorsement of a page that represents an authority on a particular topic (Chakrabarti, Dom, et al. 1999). A structural search engine uses this hyperlink structure along with the textual content to find sites of interest. Structural Web queries can locate online presentations, Web communities, or nepotistic sites. By searching the organization of a set of pages as well as the content, a search engine can provide richer information on the content of Web information.

Even a purely textual search can present challenges because human language is expressive and full of synonymy. A query for "automobile," for example, may miss a deluge of pages that focus on the term "car." One strategy to overcome this limitation is to augment search techniques with stored information about semantic relations between words. Such compilations, typically constructed by a team of linguists, are sometimes known as semantic networks, following the seminal work on the WordNet project (Miller, Beckwith, et al.

Nitish Manocha, **Diane J. Cook**, and **Lawrence B. Holder**
Department of Computer Science and Engineering
University of Texas at Arlington
manocha@cse.uta.edu,
cook@cse.uta.edu,
holder@cse.uta.edu
cygnus.uta.edu/subdue

1991). A structural search engine could employ this information to improve the retrieval of Web sites that reflect the structure and the semantics of the user query.

Much research has focused on using hyperlink information in some method to enhance Web search. The Clever system (Chakrabarti, Dom, et al. 1999) scans the most central, authoritative sites on broad search topics by making use of hyperlinks. A respected authority is a page that is referenced by many good hubs; a useful hub is a location that points to many valuable authorities. Clever uses an iterative process to identify the best hub and authority pages for a particular topic according to the strength of the hub/authority pages to which they are connected. Google (Brin and Page 1999) is another search engine that uses the power of hyperlinks to search the Web. Google evaluates a site by summing the scores of other locations pointing to the page. When presented with a specific query, Google ranks relevant pages by their evaluation measure.

Even a purely textual search can present challenges because human language is expressive and full of synonymy.

Although these systems use certain types of hyperlink structures to rank retrieved Web pages, they do not perform searches for a range of structural queries. In contrast, WebSUBDUE searches for any type of query describing structure embedded with textual content. WebSUBDUE'S functions are enhanced using tools provided by the lexicon database WordNet (see sidebar).

DATA PREPARATION

Viewed as a graph, the structure of the Web exhibits properties that are of interest to a number of researchers. For this project, we transform Web data to a labeled graph for input to the WebSUBDUE system. Data collection is performed using a Web robot written in Perl. The Web robot follows only links to pages residing on specified servers. For example, if http://cygnus.uta.edu is a defined server, the Web robot will explore the URL http://cygnus.uta.edu/projects.html, but a URL outside the domain, such as http://ranger.uta.edu, will not be explored. As it traverses a Web site, the Web robot generates a graph file representing the specified site, described as a set of labeled vertices with labeled edges connecting the vertices.

The Web robot scans each page for URL references contained in that page. A depth-first search through the space of connected Web pages on the specified site is executed. The Perl script uses the built-in hash data type / associative array to store, for each Web page, a set of URLs that contain links to the Web page. For example, if URL A and URL B link to URL C, then the hash table entry for key C is defined as *Hash_Variable*{'C'} = 'A B'. In this case, URLs A and B are considered to be parent pages, and URL C is considered to be the child page. If the child URL has already been defined, then the parent URL is appended to its values. If the child URL has not been defined, then a new entry is created in the associative array with the child URL as the key and the parent URL as the value.

All the URLs in the current page are checked to see if they have been visited. If a URL has not been visited, it is pushed onto the stack and marked as visited. When all the URLs in the page have been pushed onto the stack, the topmost URL is popped off of the stack, and this URL becomes the new parent.

For each new parent URL on the specified site, the robot fetches the page header using the "request" routine in the Perl HTTP module and checks the page fetch response code. If the response code is RC_OK, then the page currently exists and is added to the graph. The page is then scanned for URLs and all reference entries are added to the associative array. This process continues until the stack is empty or the number of visited URLs exceeds the user-defined limit.

Once the URLs have been scanned and the associative array has been created, a labeled graph is generated representing the Web site. In this graph, each URL is represented as a

vertex labeled *_page_*, and an edge labeled *_hyperlink_* points from parent URLs to child URLs (from each value of a *Hash_Variable* entry to the key itself).

After adding an edge to the graph representing a hyperlink relation, the script determines the document type by examining the Content-Type tag in the page header. If the document is either an HTML file or a text file, the file is processed to obtain keyword information. All of the HTML tags, numbers, punctuation, single characters, articles, prepositions, pronouns, and conjunctions are removed from the file, and all the remaining unique words are added to the graph. A vertex labeled with the corresponding word is added to the graph, and an edge labeled *_word_* connects the page in which the word is found to the word vertex.

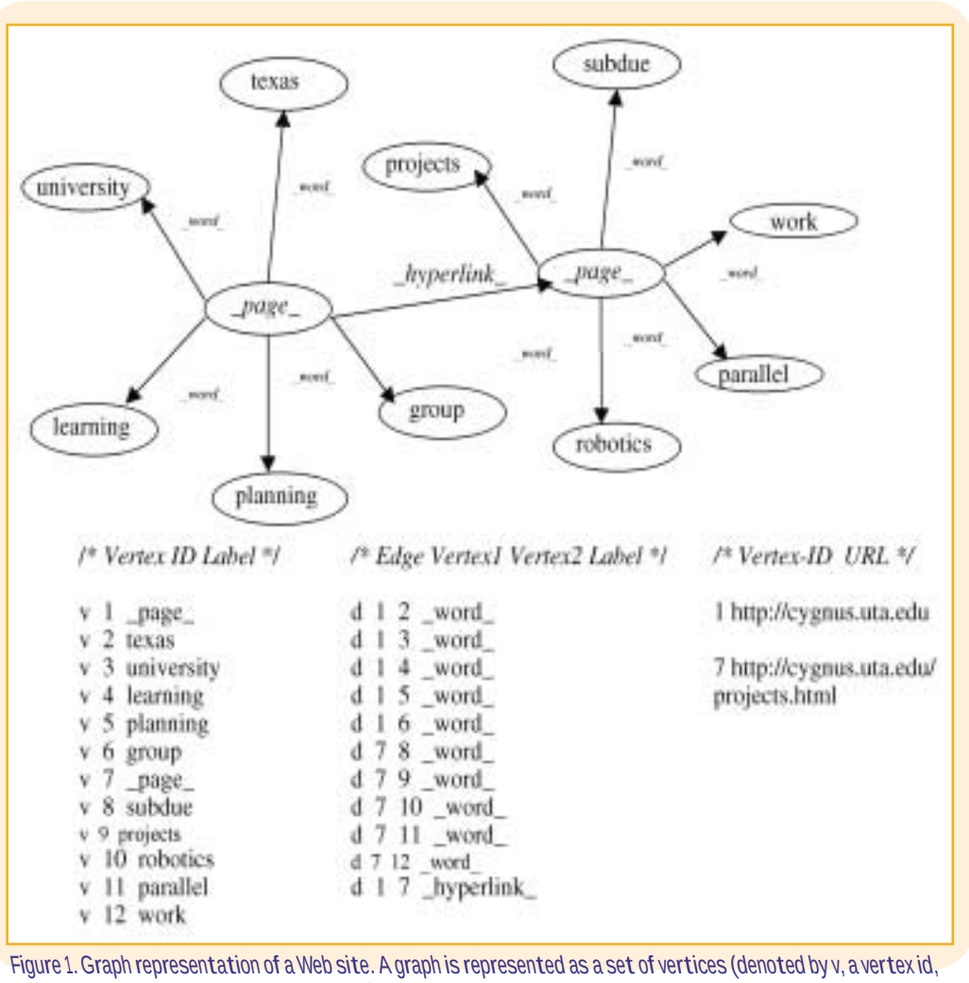


Figure 1. Graph representation of a Web site. A graph is represented as a set of vertices (denoted by **v**, a vertex id, and a label) and a set of edges (denoted by **d** for directed edge or **u** for undirected edge, the source and destination vertex IDs, and a label). Hits are represented by the vertex ID and corresponding URL.

The current version of this search engine allows the user to create a graph for a new domain or search an existing graph. An example graph representing a collection of Web pages for domain cygnus.uta.edu is shown in Figure 1. WebSUBDUE can process structural queries on this graph.

Although the Web robot creates a single graph through this process, new Web sites can be added to an existing graph, allowing Web pages to be incrementally added. In the current version, only files that are reachable from a root page are considered, which may exclude some pages. The types of files currently handled by the robot contain extensions .html (.htm), .txt, .exe, .gif, .pdf, .ps, .tar, .gz, .jpg, and .mpg.

THE SUBDUE KNOWLEDGE DISCOVERY SYSTEM

SUBDUE is a knowledge discovery system that discovers patterns in structural data (Cook, Holder, et al. forthcoming). Subdue accepts data in the form of a labeled graph and performs various types of data mining on the graph. As an unsupervised discovery algorithm, SUBDUE discovers repetitive patterns, or subgraphs, in the graph. As a concept learner, the system identifies concepts that describe structural data in one class and exclude data in

other classes. The system can also discover structural hierarchical conceptual clusters (Cook and Holder 2000). In this paper, we introduce the application of SUBDUE as a structural search engine, called WebSUBDUE, in which the search query and the Web are represented as labeled graphs. The discovered instances are reported as the results of the query. SUBDUE accepts input in the form of a labeled graph, where objects and attribute values map to vertices and relationships between objects map to edges.

Discovery Search Algorithm

When performing unsupervised knowledge discovery, SUBDUE uses a polynomial-time beam search for its main discovery algorithm. The goal of SUBDUE's search is to find the subgraph that yields the best compression of the input graph. A substructure in SUBDUE consists of a subgraph defining the substructure and all of the substructure instances throughout the graph. The initial state of the search is the set of subgraphs consisting of all uniquely labeled vertices. The only search operator is the *Extend Subgraph* operator, which extends a subgraph in all possible ways by a single edge and a vertex or by a single edge only if both vertices are already in the subgraph. Substructures are kept on an ordered queue of length determined by the beam width.

To evaluate subgraphs the Minimum Description Length (MDL) principle is used, which states that the best theory is one that minimizes the description length of a database (Rissanen 1989). According to this principle, a substructure is evaluated by its ability to minimize the description length of the graph when compressed using the substructure.

The search terminates upon reaching a user-specified limit on the number of substructures extended, or upon exhaustion of the search space. Once the search terminates and returns the list of best subgraphs, the graph can be compressed using the best subgraph. The compression procedure replaces all instances of the subgraph in the input graph by a single representative vertex. Incoming and outgoing edges to and from the replaced subgraph will connect to the new vertex that represents the subgraph. The SUBDUE algorithm can iterate on this compressed graph to generate a hierarchical description of discovered substructures.

Search for Pre-defined Substructures

SUBDUE supports biasing the discovery process toward specified types of substructures. Pre-defined substructures can be provided as input by creating a pre-defined substructure graph that is stored in a separate file. SUBDUE will try to locate and expand these substructures, thereby jump-starting the discovery process. The inclusion of background knowledge has been shown to be of great benefit. This allows the user to provide SUBDUE with background knowledge about the kinds of substructures for which the database is being mined.

For our structural search engine, WebSUBDUE invokes SUBDUE in pre-defined substructure mode, where the search query is represented in the form of a substructure. SUBDUE discovers the instances of the pre-defined substructure in the graph, representing the results of the query. WebSUBDUE reports the graph vertices, edges, and corresponding URLs for each discovered instance. For example, Figure 2 shows the representation of a struc-

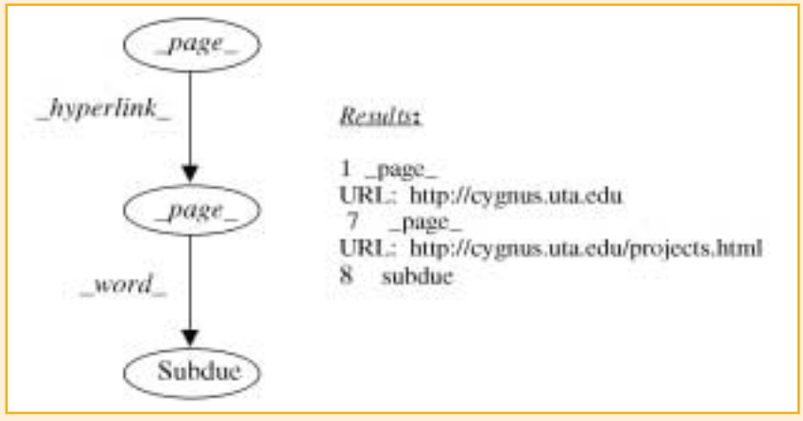


Figure 2. Structural search query.

tural search query to "*find all pages that link to a page containing the term Subdue*," along with the results of applying the search to the graph shown in Figure 1.

Inexact Graph Match

In many databases, patterns exist in the database with small variations between instances. SUBDUE applies a polynomial-time, inexact match algorithm that allows for small differences between a substructure definition and a substructure instance. The dissimilarity of two graphs is determined by the number of transformations needed to make one graph isomorphic to the other. The allowed transformations include adding or delete an edge or vertex, changing an edge or vertex label, and reversing the direction of an edge. Two graphs match if the number of required transformations is no more than a user-defined threshold value multiplied by the size of the larger graph. If the threshold value is defined as 0, an exact match is required.

Figure 3 shows an example pair of graphs. The least-cost transformation to make graph G1 isomorphic to G2 would be to map the node labeled A in G1 to the node labeled A in G2, and to map the node labeled B in G2 to

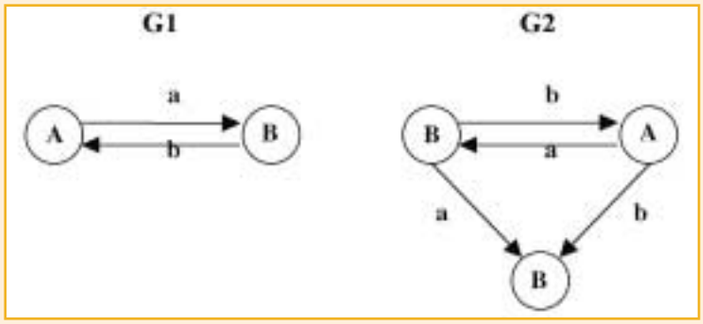


Figure 3. Example of an inexact graph match.

WordNet

WordNet is an electronic lexicon database that attempts to organize information according to the meanings of the words instead of the forms of the words (Cook and Holder 2000). The WordNet database and related tools can be downloaded from www.cogsci.princeton.edu/~wn. In this project, our goal is to augment the structural search technique with stored information about relationships between words. In particular, WebSUBDUE uses WordNet to increase its search capability by searching for words similar to the query term rather than searching only for the query term itself.

WordNet contains standard information found in dictionaries and thesauri but also stores relationships between words including hypernyms, hyponyms, and synonyms. A hypernym of a term is a more general term that fits the statement "_is a kind of". For example, a dog is a kind of canine, so canine is a hypernym of dog. Hyponyms are the opposite of hypernyms. Different words having the same meaning are called synonyms.

WordNet provides several ways to identify related terms. Synsets provide one method of grouping synonym terms. A word can be part of several different synsets, each representing a different context. WordNet also has morphological features. Although word base forms are usually stored in WordNet, searches may be performed on inflected forms. We apply a set of morphology functions to the input search string, generating a form that is present in WordNet. For example, if a user searches for the word *teaching*, WordNet will return the base form *teach*. Similarly, if the query term *Oct* is not present in WordNet, WordNet will return the inflected form *October*, which does reside in its dictionary.

We expanded the capabilities of WebSUBDUE's structural search engine by integrating some of the functions of WordNet. WebSUBDUE uses WordNet to expand its search capabilities by finding words similar to the search terms. In particular, vertices representing a word within a page are matched with vertices containing a different label if they hold one of the valid relationships defined by WordNet. Valid considerations include matching the base form of the word (e.g., match *base* or *basis* with query *bases*), derived forms of the word (e.g., match Jan with query *January*), and synonyms (e.g., match *employment*, *position*, or *work* with query *job*).

the top left node labeled B in G2. The corresponding cost is calculated as the cost for adding a node (the bottom node labeled B in G2) and two corresponding edges. Because the size of the larger graph is 3 vertices + 4 edges = 7, graph G1 would be considered an instance of G2 if the threshold value is 3/7 or greater.

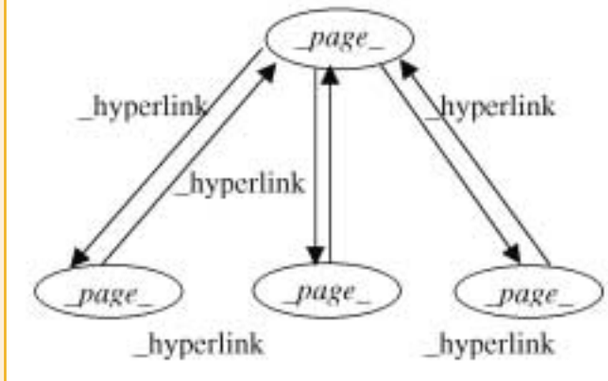
To find the least-cost match, SUBDUE searches in a branch-and-bound fashion through the space of possible matches. The first complete match found is thus the least-cost match and is returned. If the number of partial matches considered exceeds a user-defined function of the size of the larger graph, the search switches to a hill climbing search. As a result, the inexact graph match is constrained to run in time polynomial in the size of the larger graph. SUBDUE's inexact graph match algorithm can be used by WebSUBDUE to efficiently find Web sites that closely, but not exactly, match the user query.

EXPERIMENTAL RESULTS

We conducted a number of experiments to evaluate the capabilities of the WebSUBDUE structural search engine. When possible, we compare query results of WebSUBDUE with search results generated using a popular search engine, AltaVista (Cook, Holder, et al. forthcoming). AltaVista offers features such as a simple keyword search, searches with advanced options, image searches, audio searches, and video searches. AltaVista's advanced search features include the use of Boolean expressions, the limitation of a search to a particular host or domain, and other search criteria that provide a valuable point of comparison for the results discovered by WebSUBDUE.

To demonstrate the structural searching capability of WebSUBDUE, we posed several queries to the system and compared the results with similar searches using AltaVista. The search ranges of WebSUBDUE and AltaVista are restricted to include only Web pages starting with http://www-cse.uta.edu, a graph which represents 5,825 URLs in the CSE-UTA domain using 113,933 vertices and 125,657 edges.

One type of structural query might be to find online lecture materials or HTML papers discussing a particular topic. A query to search for all presentation-style pages was posed to WebSUBDUE. A presentation page is defined



1. ~holder/courses/cse5311/lectures/l26/l26.html
2. ~cook/ai1/syllabus/syllabus.html
3. ~holder/courses/cse2320/lectures/l3/l3.html
4. ~holder/courses/cse5311/lectures/l12/l12.html
5. ~holder/courses/cse2320/lectures/l16/l16.html
6. ~holder/courses/cse2320/lectures/l10/l10.html
7. ~holder/courses/cse5311/lectures/l19/l19.html
8. ~holder/courses/cse5311/lectures/l9/l9.html
9. ~cook/ai1/lectures/l13/l13.html
10. ~cook/ai1/lectures/l1/l1.html
11. ~reyes/node48.html
12. ~reyes/node1.html
13. ~cook/ai1/lectures/l6/node6.html
14. ~cook/ai1/lectures/l7/l7.html
15. ~holder/courses/cse5311/lectures/l23/l23.html
16. ~holder/courses/cse2320/lectures/l2/l2.html
17. ~cook/ai1/lectures/l10/l10.html
18. ~cook/ai1/lectures/l12/l12.html
19. ~holder/courses/cse5311/lectures/l10/l10.html
20. ~holder/courses/cse2320/lectures/l15/l15.html
21. ~holder/courses/cse5311/lectures/l4/l4.html
22. ~holder/courses/cse5311/lectures/l2/l2.html

Figure 4. Presentation page structural query.

here as series of pages focusing on a topic, where each page contains links to the *next* page, the *previous* page, and the top page, the *top* page being a menu page that has links to all the other pages. This query structure is shown in Figure 4.

In response to the presentation page query, WebSUBDUE discovered 22 unique instances or 22 presentation pages on the CSE-UTA site. The URLs discovered are listed in Figure 4. All of these Web pages are presentation pages covering various topics with a top menu page containing a link to subtopic pages and each subtopic page pointing to the next subtopic page, the previous page, and the top menu page.

AltaVista has no method of responding to structural search queries. To determine whether AltaVista could find the sites discovered by WebSUBDUE, we provided AltaVista with additional information. All the discovered pages represent documents that were generated using the LaTeX2HTML conversion program and contain links to the icon files *next_motif.gif*, *up_motif.gif*, and *previous_motif.gif*, representing contiguous sections of the document. To assist AltaVista, we provide information about icon file links using the advanced search option *host:www-cse.uta.edu AND image:next_motif.gif AND image:up_motif.gif AND image:previous_motif.gif*. This search option asks AltaVista to search in the CSE-UTA domain for pages containing links to next icon, previous icon, and top icon files.

AltaVista discovered 12 instances, and WebSUBDUE discovered 22 instances. Because the actual names of the icons varied between presentations, not all presentation pages were discovered using AltaVista. This shows the difficulty of posing these types of structural queries using a text search engine, whereas WebSUBDUE just requires the structure information.

WebSubdue can be used with the synonym-searching capability to find pages containing text on *jobs in computer science*. The structural query to perform this search is shown in Figure 5.

WebSUBDUE discovered 33 instances of the search query, as summarized in Figure 5 with the matched terms found in the corresponding page. WebSUBDUE finds *employment*, *work*, *job*, *problem*, and *task* as synonyms and allowable forms of the search terms. Only exact matches were found for computer and science in these pages.

A similar query was posed to AltaVista using the syntax *host:www-cse.uta.edu AND text:jobs AND text:computer AND text:science*. AltaVista discovered two instances of the search query. WebSUBDUE can thus cover a wider search range using synonyms along with the keywords themselves.

Viewed as a graph, the structure of the Web exhibits properties that are of interest to a number of researchers.

In our final example we demonstrate how WebSubdue can use a combination of text/synonym match and structure inexact match to satisfy a query for hub and authority pages. Web search engines typically index several thousand relevant pages on a topic, but the user will only be willing to look at a small number of these sites. Hyperlink information can be used to filter this search by identifying hub and authority pages. A hub page on a topic is a page that has links to many other pages on that topic, or that links to many authorities on a topic. A good authority is a page that is pointed to by many good hubs, while a good hub is a page pointed to by many authorities (Kleinberg 1998). The HITS algorithm (Cook and Holder 2000) uses a series of matrix calculations to find good hub and authority pages. WebSubdue can discover these hub and authority pages by specifying the hub and authority page structure.

A query to search for all hub and authority pages on *algorithms* was posed to WebSUBDUE (AltaVista cannot process this type of structural query). The substructure input to WebSUBDUE is shown in Figure 6.

The query structure represents both an authority page as a page that is pointed to by many hubs, and a hub as a page pointed to by many authorities. WebSUBDUE discovered one instance of the search query, and the corresponding three hub sites and three authority sites are summarized in Figure 6.

SUBDUE's inexact graph match capability

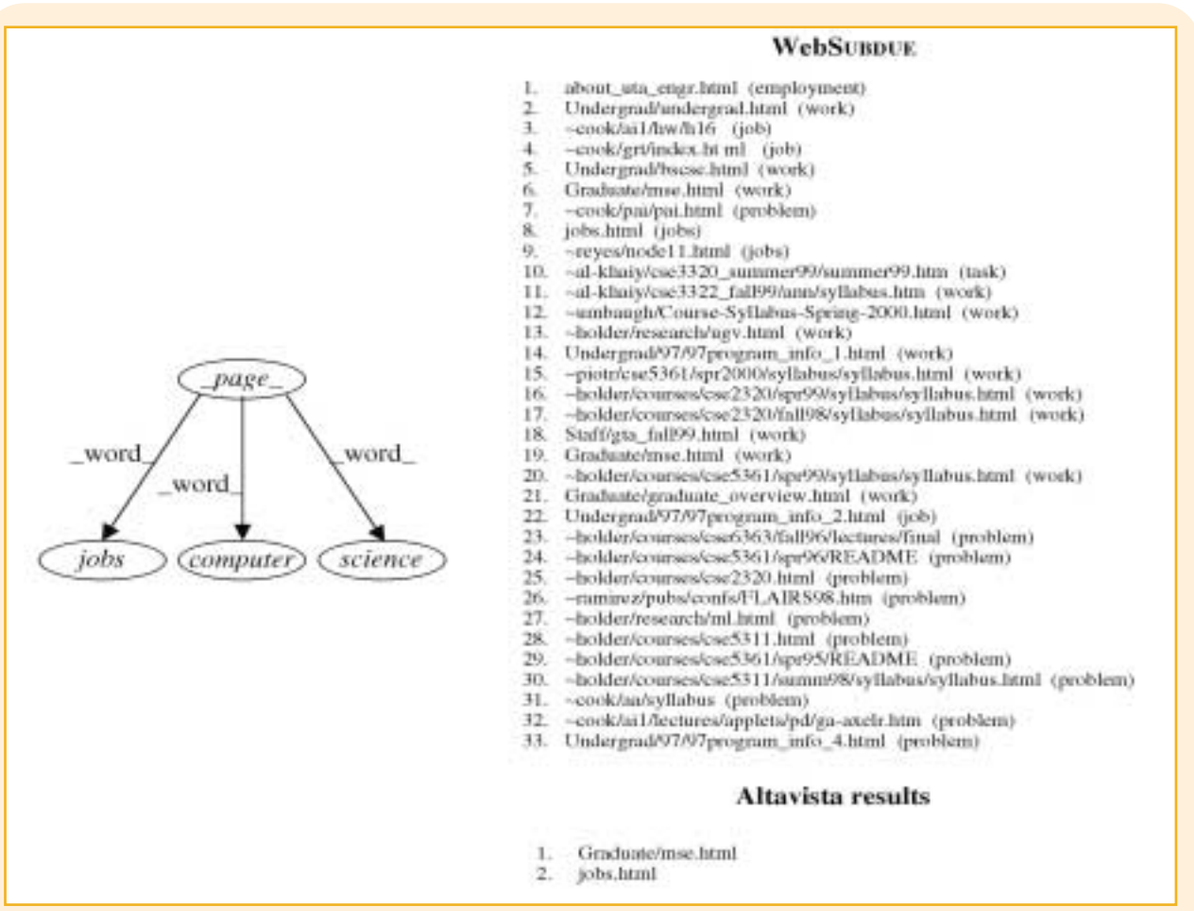


Figure 5. Structural query to find pages on jobs in computer science.

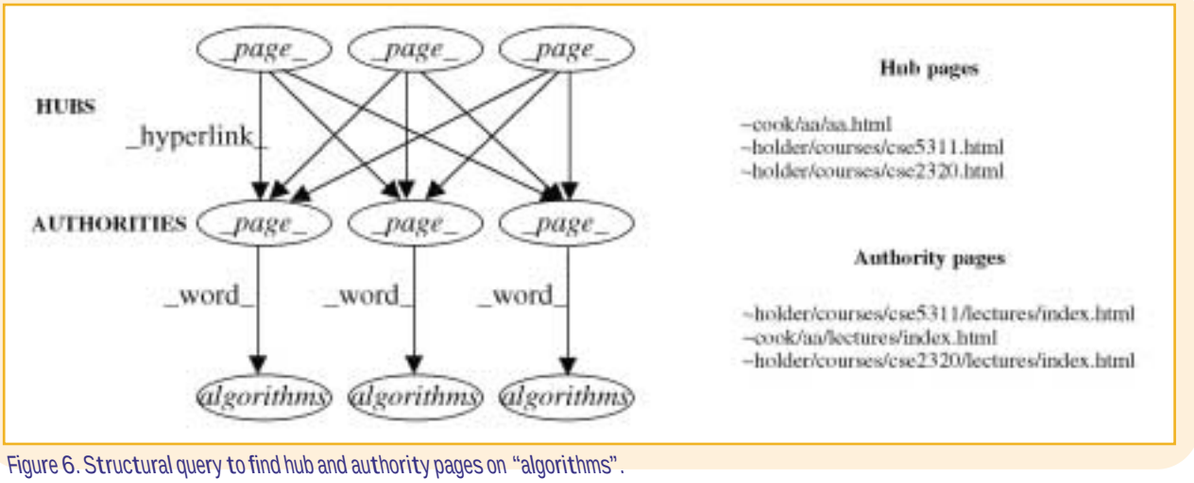


Figure 6. Structural query to find hub and authority pages on "algorithms".

can be used to improve the WebSUBDUE's Web search results. When an inexact match is used, Web sites can be retrieved that approximate the query structure. The amount of allowable difference is controlled by the user-defined match threshold.

To demonstrate the effect of using an inexact graph match, we repeat the search for hub and authority pages that focus on *algorithms*, but this time allow an inexact threshold of 0.2. Because all transformations are assigned a unit cost and the size of the query structure is 9 vertices + 12 edges = 21, up to 0.2 * 21 = 4 (4.2) transformations are allowed. A total of 13 instances are identified with this query, as opposed to the single instance found using an exact match. The instances differ from the query structure by the addition of a node with an associated edge, the lack of one or more of the connecting edges, or a change in a node or edge label. These matches indicate that the related sites are fairly good hubs and authorities, and the match cost provides a useful ranking measure among the retrieved sites.

Results of these experiments indicate that WebSUBDUE can successfully perform structural search, text search, synonym search, and combinations of these searches. These experiments also demonstrate the need for structural search engines and the inability of existing search engines to perform these functions.

REFERENCES

Brin, S. and Page, L. The anatomy of a large-scale hypertextual Web-search engine. 1999. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia. April 14-18, 1998.

Chakrabarti, S., Dom, B.E., Gibson, D., Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., and Tompkins, A. 1999. Mining the link structure of the World Wide Web. *IEEE Computer 32*, 8, pp. 60–67.

Cook, D.J. and Holder, L.B. 2000. Graph-based data mining. *IEEE Intelligent Systems 15*, 2, pp. 32–41.

Cook, D.J., Holder, L.B., Galal, G., and Maglothin, R. Forthcoming. Approaches to parallel graph-based knowledge discovery. *Journal of Parallel and Distributed Computing*.

Kleinberg, J. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms*.

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K.J. 1991. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography 3*, 4, pp. 235–244.

Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company.