Applications of Graph Probing to Web Document Analysis

Daniel Lopresti and Gordon Wilfong
Bell Labs, Lucent Technologies Inc.
600 Mountain Avenue
Murray Hill, NJ 07974 USA
{dpl,gtw}@research.bell-labs.com

1 Introduction

Graphs are a fundamental representation in much of computer science, including the analysis of both traditional and Web documents. Algorithms for higher-level document understanding tasks often use graphs to encode logical structure. HTML pages are usually regarded as tree-structured, while the WWW itself is an enormous, dynamic multigraph. Much work on attempting to extract information from Web pages makes explicit or implicit use of graph representations [1, 3, 4, 7, 11].

It follows, then, that the ability to compare two graphs is basic functionality, as demonstrated in such applications as query-by-structure, wrapper generation for information extraction, performance evaluation, etc. Because most problems relating to graph comparison have no known efficient, guaranteed-optimal solution, researchers have developed a wide range of heuristics. For the problem of determining isomorphism, for example, many heuristics rely on the existence of certain *vertex invariants*, which consist of a value $f(v)$ assigned to each vertex v, so that under any isomorphism I, if $I(v) = v'$ then $f(v) = f(v')$. One commonly used invariant is the degree of a vertex. In fact **nauty**, a successful software package for determining graph isomorphism (see [9]), relies on such vertex invariants.

This observation can be seen as forming the basis for *graph probing*, a paradigm we have recently begun exploring for graph comparison [5, 8]. However, we desire more than a simple "yes/no" answer; we are interested in quantifying the similarity between two graphs, not just in whether they may be isomorphic. Conceptually, the idea of probing is to place each of the two graphs under study inside a "black box" capable of evaluating a set of graph-oriented operations (*e.g.*, returning a list of all the leaf vertices, or all vertices labeled in a certain way). We then pose a series of probes and correlate the responses of the two systems.

Our past work in the area treats graph probing as an on-line process; both the query graph and the database graph are available for synthesizing the probe set. While this is an appropriate assumption when one is comparing, say, the output of a recognition algorithm with its associated ground-truth, it is not a workable model for retrieval applications when the database contains anything other than a small number of documents.

In this paper, we describe our first steps towards adapting the graph probing paradigm to allow pre-computation of a compact, efficient probe set for databases of graph-structured documents in general, and Web pages coded in HTML in particular. This new model is shown in Figure 1, where the portion of the computation bounded by dashed lines is performed off-line. We consider both comparing two graphs in their entirety, as well as determining whether one graph contains a subgraph that closely matches the other. We present an overview of work in progress, as well as some preliminary experimental results.

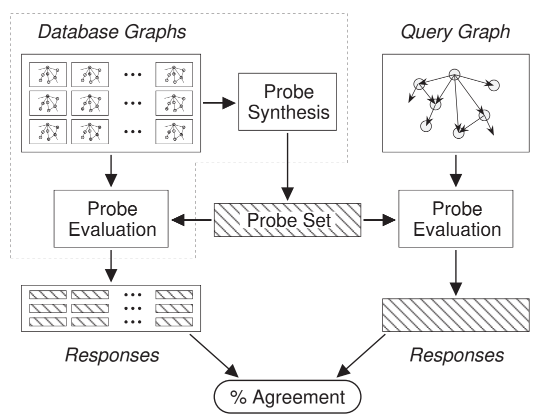

Figure 1. Overview of graph probing.

2 Related Work

Graph comparison is an important yet difficult problem, so it should come as no surprise that a large number of researchers have proposed heuristics or solutions designed for special cases. For example, Bunke and Messmer present a decision-tree-based precomputation scheme for solving the subgraph isomorphism problem [2], although their data structure can be exponential in the size of the database graphs in the worst case.

Lazarescu *et al.* propose a machine learning approach to building decision trees for eliminating from further consideration graphs that cannot possibly be isomorphic to a given query [6]. While they employ a similar set of features to the ones we use, they do not consider the approximate matching or subgraph problems.

Papadopoulos and Manolopoulos discuss an idea that is philosophically quite similar to ours [10]. However, they focus on a single invariant: node degree. It is clear this is not sufficient for catching all of the interesting differences that can arise between HTML documents. Moreover, their histogram technique is applied only to the problem of comparing complete graphs, whereas we wish to examine the subgraph matching problem as well.

Valiente and Martínez describe an approach for subgraph pattern-matching based on finding homomorphic images of every connected component in the query [12]. Again, the worst-case time complexity is exponential, but such features could also perhaps be incorporated in the heuristics we are about to present.

Instead of trying to solve the problem for graphs in general, some leeway can be had by limiting the discussion to trees, for which efficient comparison algorithms are known. Schlieder and Naumann consider a problem closely related to ours: error-tolerant embedding of trees to judge the similarity of XML documents [11]. Likewise, Dubois *et al.* write about tree embedding for searching databases of semi-structured multimedia documents [4].

Finally, a number of researchers have studied techniques for identifying smaller, coherent substructures within Web pages (*e.g.*, lists, tables) [1, 3, 7].

3 Graph Probing

The graph model we assume for HTML documents includes the standard tree-structured hierarchy generated when parsing the tags (the "contains"/"contained-by" relationship). In addition, we also make use of the order in which content and the various substructures are encountered (in many cases this corresponds to the natural reading order for the material in question). We represent this via "next"/"previous" cross-edges that connect nodes at a given level in the hierarchy, rather than assuming an implicit fixed ordering on the children of a node as some other researchers have done. Lastly, we record hyperlinks as either back-edges (in the case of targets on the same page) or a distinguished node type (in the case of external references). No provision is currently made for incoming links from outside documents.

Several criteria are desirable when designing a probing strategy: probes should be invariant across graph or subgraph isomorphism, they should be easy to evaluate and the responses easy to compare, similar graphs should agree more often than dissimilar graphs, and the probes in a set should be independent.

While the probing paradigm is open-ended, our initial efforts for Web documents have focused on the following categories:

Class 0 These probes count the number of occurrences of a given type of vertex in the graph. A representative Class 0 probe might be paraphrased as: *How many vertices labeled "Table" does the graph have?*

Class 1 These probes examine the vertex and edge structure of the graph by counting in- and out-degrees, tabulating different types of incoming and outgoing edges separately. An example is: *How many vertices have one incoming edge labeled "contains", another labeled "next", and none labeled "href", along with two outgoing edges labeled "contains", one labeled "next", and one labeled "href"?*

In addition to these classes, which we hope to expand for this present work, our research on table understanding makes use of two other, more sophisticated types of probes [5].

We define a *discriminating probe* to be a probe that demonstrates a difference between two graphs. Two fundamental questions are of interest: (1) For two graphs that are different, does there exist at least one discriminating probe? and (2) Over the entire set of probes, how many are discriminating? The first of these reflects the graph isomorphism problem. The second can serve as a measure of how similar the two graphs are. To make this more explicit, we define the *agreement* between two probe sets to be:

$$\textit{agreement} \;\equiv\; 1.0 \;-\; \frac{\textit{\# of discriminating probes}}{\textit{total \# of probes}} \qquad (1)$$

Our aim is to equate "agreement" with the traditional concept of "accuracy."

When comparing two graphs in their entirety, it suffices to correlate their responses to the probes and count the number of times they agree. For the problem of subgraph matching, however, we cannot expect to be able to compare directly the outputs for the larger graph to those for the smaller. For example, consider a query graph consisting of a single table that corresponds to a table in some database document, but where that document also contains dozens of other, unrelated tables.

To assure that the Class 0 probes are invariant across subgraph isomorphism, the manner in which the results are compared must account for this. Clearly, if the query graph contains a certain number of nodes labeled in a given way, the target graph may possibly contain an isomorphic subgraph so long as it contains at least that many nodes labeled in the same way. If the target graph has fewer such nodes, we know there cannot be a subgraph isomorphism. Similarly, the way in which the Class 1 probes are correlated needs to be modified as well, since the nodes present in the query graph may have fewer incoming and outgoing edges than their corresponding nodes in an isomorphic subgraph of a larger graph.

4 Preliminary Experimental Results

This paper describes research that is very much still in progress. As we have noted, we have implemented the on-line version of graph probing to help in the evaluation of table understanding algorithms. However, the utility of graph probing in that specialized domain is not an assurance that it is an appropriate paradigm for searching databases for matches in a retrieval application, especially since the classes of probes we have at our disposal are also different here.

To begin examining some of these issues, we performed two simple experiments to test graph probing in an information retrieval setting. In both cases we used Class 0 and Class 1 probe sets. The first test examined the ability of the two classes to detect changes as the structure of a specific commercial Web page evolved naturally over several days. The second studied the subgraph matching problem by searching for a Web page that had been edited by deleting a significant portion of its content.

The small database consisted of 75 current WWW homepages collected from a variety of sources: commercial, educational institutions, personal, news organizations, and portal pages. We implemented a parse tree generator capable of recovering from the kinds of simple errors that often arise in real-world HTML (*e.g.*, missing end tags). The size of the resulting graphs ranged from a minimum of 60 nodes to a maximum of 1,204, with an average of 423 nodes. The probe set, generated automatically, consisted of a total of 219 Class 0 and Class 1 probes.

For the first experiment, we used as our query the May 31, 2001 homepage from *The New York Times* (http://www.nytimes.com/). This is a relatively complex page, its parse tree containing 1,089 nodes. The results for using graph probing to compare this page to the 75 pages in the database are shown in Figure 2. The May 31 page is, of course a perfect match for itself, but also a very good match for the June 4, 5, and 6 homepages as well (the only other examples from *The New York Times* in the database). Clearly some sort of structural change in the page was made between May 31 and June 4. Conversely, a number of other regularly-updated Web pages we have examined show no structural changes from day-to-day, although obviously the content is constantly varying.

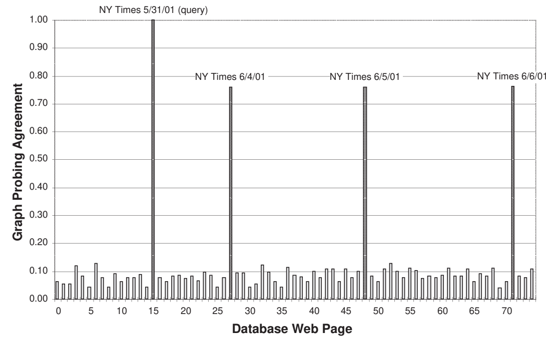

Figure 2. Graph probing results for Experiment 1 (full graph comparison measure).

For our second experiment, we used the homepage for the WDA'2001 workshop, a screen snapshot of which is given in Figure 3. The corresponding parse tree contained 328 nodes. To create the query, the page was edited by deleting the sidebar on the left side of the page. The parse tree for the edited page had 125 nodes.

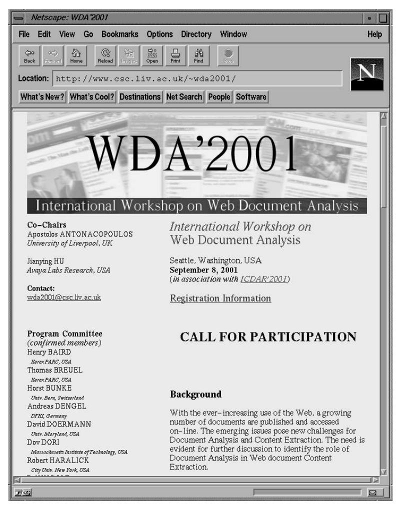

Figure 3. Snapshot of the WDA'2001 homepage (http://www.csc.liv.ac.uk/˜wda2001/).

The results for this experiment, using graph probing with the subgraph comparison measure, are shown in Figure 4. Here we can see that the two probe classes are able to distinguish the original page and the smaller, edited version from the rest of the database, but just barely. The current probes, which consider only structure (and high-level structure at that), need to be supplemented with new classes that are able to make use of content and other aspects of the page layout before the probing paradigm can be effective for the subgraph matching problem. This is work in progress.

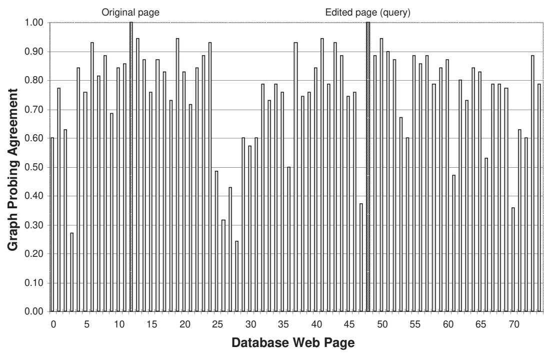

Figure 4. Graph probing results for Experiment 2 (subgraph comparison measure).

5 Discussion

In this paper we have described our initial efforts to adapt the graph probing paradigm to searching databases of graph-structured documents. We considered both the comparison of two graphs in their entirety, as well as the subgraph matching problem, and gave some preliminary experimental results.

Currently, our graph probing provides a measure of how similar two graphs are or how similar one graph is to some subgraph of another. It does not, however, provide a mapping from one graph to the other. Clearly, to extract information from semi-structured sources we need to recognize more than the fact that some matching is likely to exist, we must be able to identify the actual correspondence between the graphs (or between one graph and a subgraph of the other). As mentioned, at the very least graph probing can be used to identify graphs that are a likely match and then run more computationally expensive methods on this smaller collection of graphs to find the desired matches.

In addition to information retrieval, other possible applications of graph comparison via probing could include wrapper generation and maintenance (*e.g.*, [1, 3]) and analysis of HTML-coded tables (*e.g.*, [7]). This would require retargeting our graph probing language to information extraction applications, a task that would be challenging but seems feasible.

References

[1] N. Ashish and C. Knoblock. Wrapper generation for semi-structured Internet sources. In *Proceeding of the Workshop on the Management of Semistructured Data*, Tucson, AZ, June 1997.

[2] H. Bunke and B. T. Messmer. Recent advances in graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(1):169–203, November 1997.

[3] W. W. Cohen. Recognizing structure in Web pages using similarity queries. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 59–66, Orlando, FL, 1999.

[4] D. Dubois, H. Prade, and F. Sedes. Some uses of fuzzy logic in multimedia databases querying. In *Proceedings of the Workshop on Logical and Uncertainty Models for Information Systems*, London, England, July 1999.

[5] J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table structure recognition and its evaluation. In *Proceedings of Document Recognition and Retrieval VIII*, volume 4307, pages 44–55, San Jose, CA, January 2001.

[6] M. Lazarescu, H. Bunke, and S. Venkatesh. Graph matching: Fast candidate elimination using machine learning techniques. In *Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 236–245. Springer-Verlag, Berlin, Germany, 2000.

[7] S.-J. Lim and Y.-K. Ng. An automated approach for retrieving hierarchical data from HTML tables. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 466–474, Kansas City, MO, November 1999.

[8] D. Lopresti and G. Wilfong. Evaluating document analysis results via graph probing. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 201–210, Columbia, MD, April 2001.

[9] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[10] A. N. Papadopoulos and Y. Manolopoulos. Structure-based similarity search with graph histograms. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, pages 174–178. IEEE Computer Society Press, 1998.

[11] T. Schlieder and F. Naumann. Approximate tree embedding for querying XML data. In *Proceedings of the ACM SIGIR Workshop on XML and Information Retrieval*, Athens, Greece, July 2000.

[12] G. Valiente and C. Martínez. An algorithm for graph pattern-matching. In *Proceedings of the Fourth South American Workshop on String Processing*, pages 180–197. Carleton University Press, 1997.