Identifying Aggregates in Hypertext Structures

Rodrigo A. Botafogo[1] and Ben Shneiderman[2]

Human-Computer Interaction Laboratory &
Department of Computer Science
University of Maryland
College Park, MD 20742

ABSTRACT

Hypertext systems are being used in many applications because of their flexible structure and the great browsing freedom they give to diverse communities of users. However, this same freedom and flexibility is the cause of one of its main problem: the "lost in hyperspace" problem. One reason for the complexity of hypertext databases is the large number of nodes and links that compose them. To simplify this structure we propose that nodes and links be clustered forming more abstract structures. An abstraction is the concealment of all but relevant properties from an object or concept.

One type of abstraction is called an *aggregate*. An aggregate is a set of distinct concepts that taken together form a more abstract concept. For example, two legs, a trunk, two arms and a head can be aggregate together in a single higher level object called a "body." In this paper we will study the hypertext structure, i.e., the way nodes are linked to each other in order to find aggregates in hypertext databases. Two graph theoretical algorithms will be used: biconnected components and strongly connected components.

KEYWORDS

Hypertext, structural analysis, graph theory, abstraction, aggregation, generalization.

INTRODUCTION

Hypertext systems are used in many applications because of their flexible structure and the great browsing freedom they give to users. However, this same freedom and flexibility is the cause of one of its main problem: the "lost in hyperspace" problem. Previous attempts to solve this problem have concentrated in improving the user interface by having *multiple windows* [Mar89], *maps* [Nie90a,b][Bro89], and *tours* or *path mechanisms* [Tri88][Zel89]. Unfortunately, these techniques do not scale up nicely. Multiple windows help only in a very localized way, maps lose their appeal when hypertexts have more than a few dozen nodes, and path mechanisms are very hard to author and maintain.

---

[1] Partially supported by the Brazilian National Research Council (CNPq) under grant 201008/88.2 and NCR Corporation. Currently at NEC Corporation, Tokyo.

[2] Address correspondence to Ben Shneiderman.

One reason for the complexity of hypertexts is the large number of nodes and links that compose them. Halasz [Hal88] suggested that *composition* will be an important mechanism in future generations of hypertext systems. A composition mechanism is a way of dealing with a set of nodes and links as a single object. A similar notion was also introduced by Smith & Smith [Smi77a,b] in the database field: the formation of abstraction.

An abstraction is the concealment of all but relevant properties of an object or concept. Only information relevant to the task being performed is shown in an abstract object. Smith & Smith claim that abstraction allows for the intellectual manageability of highly structured objects. Since hypertext systems are very complex structured objects which suffer severely from a lack of intellectual manageability ("lost in hyperspace"), they can profit from abstraction. Smith & Smith define two types of abstraction: *aggregation* and *generalization*.

Aggregation is an operation that clusters related objects and forms a higher level object. For example, two legs, a trunk, two arms and a head can be aggregated together in a single higher level object called a "body." Halasz, in his paper, only identifies aggregation as a composition mechanism. However, generalization seems to be a very important property to simplify hypertexts. Generalization happens when a set of similar objects are regarded as a generic object. A hummingbird, a hawk, an eagle, etc., can be generalized to a single concept: that of "bird."

Developers of future generations of hypertext systems will face two important challenges: how to deal with abstraction and how to help authors to create comprehensible structures. The first challenge was described by Halasz [Hal88] and some systems such as NLS/Augment [Eng68] or IGD [Fei82; Fei88] already have preliminary solutions to it. The second issue is the one of interest in this paper. Botafogo, Rivlin, and Shneiderman [Bot91] studied hypertext structures to provide authors with different views and with useful metrics. Brown [Bro90] stresses the importance of analytical tools and metrics to assess hypertext quality. In this paper structural analysis will be used to help authors find abstractions in hypertexts.

Section 1 introduces some preliminary concepts and section 2 shows how to use two graph theoretical algorithms, *biconnected* and *strongly connected* components, to create aggregates. The interested reader is directed to [Bot90] [Bot91] for a complete discussion of generalization and metrics.

In this paper the word "hypertext" does not refer to every type of hypertext system. We focus on a card-based hypertext systems where nodes and links are untyped, or systems which can be represented by a directed graph. All the hypertexts that were used for testing our ideas were created in Hyperties (a hypertext system developed by the Human-Computer Interaction Laboratory; it has been expanded and distributed by Cognetics Corporation, Princeton Junction, NJ). We believe, however, that the ideas presented here could be extrapolated to other hypertext systems with a more complex structure.

1 PRELIMINARIES

1.1 Index and Reference nodes

Intuitively *index* nodes are nodes that can, as the name implies, be used as an index or guide to many other nodes. For example, an article that points to all the other articles in a hypertext is an index. In a hypertext about a computer science department an article pointing to all the professors that work in human factors is also an index. Formally, an index node is a node whose outdegree is greater than the mean outdegree of all nodes, plus a threshold value.

Reference nodes are in a certain way the converse of index nodes. For example, in a hypertext about animals, all birds might link to an article about “feathers,” dogs to an article about “sharp teeth,” etc. It is possible to make reference to the birds by saying that they are animals that have feathers. Formally, a reference node is a node whose indegree is greater than the mean indegree of all nodes, plus a threshold value. Since index and reference nodes usually point to or are pointed by whole classes of nodes it is natural to define index and reference nodes as a function of their out and in-degrees respectively.

Definitions

• Let μ be the mean of the outdegrees of the nodes in the hypertext and let μ' be the mean of the indegrees of the nodes in the hypertext. Note that $\mu = \mu'$ since every link that leaves a node has to reach another node. For this reason we will use μ for both means.

• Let σ be the standard deviation of the outdegrees of the nodes.

• Let σ' be the standard deviation of the indegrees of the nodes.

• Let τ be a threshold value.

• An index node is a node whose outdegree is greater than $\mu + \tau$.

• A reference node is a node whose indegree is greater than $\mu + \tau$.

We usually define τ as been equal to $3 * \sigma$ (σ'). The motivation for this choice is as follows: if the number of links follow a normal distribution, then a node that has in or out-degree exceeding three standard deviations will occur less than one percent of the time making them special in the context of the hypertext.

1.2 Metrics

Since hypertexts are composed of two main components, nodes and links, it is helpful to the author to have knowledge of the number of nodes and links in the hypertext. Many systems do provide the first piece of information, but few consider the number of links. With that information at hand the author can start forming an idea of the complexity of the hypertext. However, the number of nodes and links and possibly their ratio gives only a rough idea of the complexity. In order to better capture the notion of how complex a hypertext is the *compactness* metric will be developed. The compactness indicates the interconnectedness of a hypertext.

From a reader’s point of view a too high compactness means that each node has many links and that consequently there are potentially many cycles. Traversing many cycles can disorient users. Too many links might also indicate a poorly organized hypertext. For example, a hypertext that is fully connected is equivalent to a hypertext that is fully disconnected but where the user can access any node by using an index. In a fully connected hypertext the user has no clue to which article should be read next, which is equivalent to choosing the next article to be read from a general index. On the other hand, a too low compactness indicates insufficient links and that possibly parts of the hypertext are disconnected. For our purposes the following facts about the compactness are of importance [Bot90a,b]:

(a) The compactness (Cp) is a value in the range 0 to 1. It is 0 when the hypertext is disconnected and 1 when the hypertext is fully connected.

(b) Any graph has a unique compactness associated to it.

(c) From experience, appropriate compactness is in the range 0.3 to 0.8.

Formally the compactness is defined as:

• $Cp = (Max - \sum_i \sum_j C_{ij}) / (Max - Min)$

C_{ij} is the distance from node *i* to node *j*. If node *i* is not connected to node *j* the distance between them is infinite. However, in order to be able to calculate efficiently, when two nodes are not connected the distance C_{ij} is set equal to a constant K. This constant is called the converted distance (see [Bot91] for more detailed information about the constant K). Max is a parameter that depends only on the number of nodes in the graph and the converted distance. It is the distance from every node to every other node when the graph is completely disconnected. We note that:

• $Max = (n^2 - n)\ K$, where n is the number of nodes in the hypertext. The distance from a node to itself is always zero.

Min is defined in a similar way than Max, but in this case the graph is completely connected. Again we note that:

• $Min = (n^2 - n)$. When the graph is completely connected, the distance of a node to any other node is equal to 1.

2 CLUSTERING BY INTERRELATIONS: AGGREGATION

Aggregation was defined previously as an operation that clusters *related objects* and forms a higher level object. In order to help authors in the formation of aggregates, the notion of related objects (or related articles in our case) must be formally defined. In information retrieval there is a long history of multi-dimensional clustering algorithms based on text analysis [Cro89]. By contrast, hypertexts are explicitly structured by links between articles which establish that there is a semantic relation. Assuming that the semantic relation is a transitive one, a path between two nodes also indicates a semantic relation. However, as the distance between two nodes in the path increases, the strength of the relation diminishes. On the other hand, the more paths there are between any two nodes the stronger their relation.

The compactness is then a good measure of how related a set of nodes is. The higher the compactness the more related they are. A *semantic cluster* – a set of related objects appropriate for aggregation – will be defined as:

Definition:

• A *semantic cluster* of a hypertext is a set of nodes and links that have the following two properties:

(a) they are a subgraph of the hypertext graph.

(b) the compactness of the subgraph is higher than the compactness of the whole graph.

With a formal mathematical definition of related objects, it becomes possible to apply graph theoretical algorithms to help in the formation of aggregates. Unfortunately, hypertexts are written in a way that does not always make our assumptions true. For example, the aggregate "body," made of two legs, a trunk, two arms, a head, etc., will be represented in a hypertext as in figure 1. Note that the links between the parts is not present, since the abstract notion has already been formed. When one sees a leg it is usually followed by another leg, both of them are seen with a trunk; unless you are seeing a cartoon or a horror movie, all the previous part are crowned by a head. figure 2 is then a more realistic, although less structured, representation for the hypertext. This suggests that the more unstructured the hypertext is, with all relevant links present, the more it will profit from the use of semantic clusters.

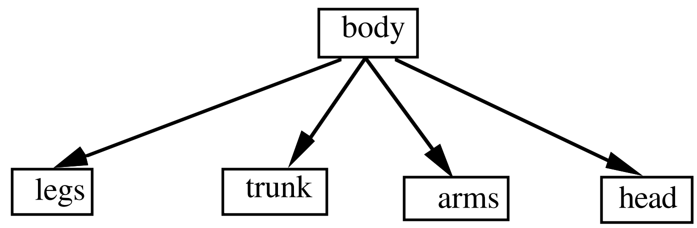

Figure 1

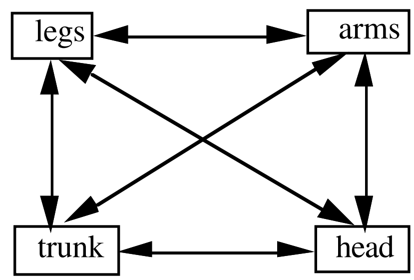

Figure 2

In order to make hypertexts intelligible for readers, authors remove links as explained above, but in many cases they also add extra links. Figure 3 shows a hypertext in which all the birds seem to be highly related, since there are many paths between them (if direction of links is not considered).

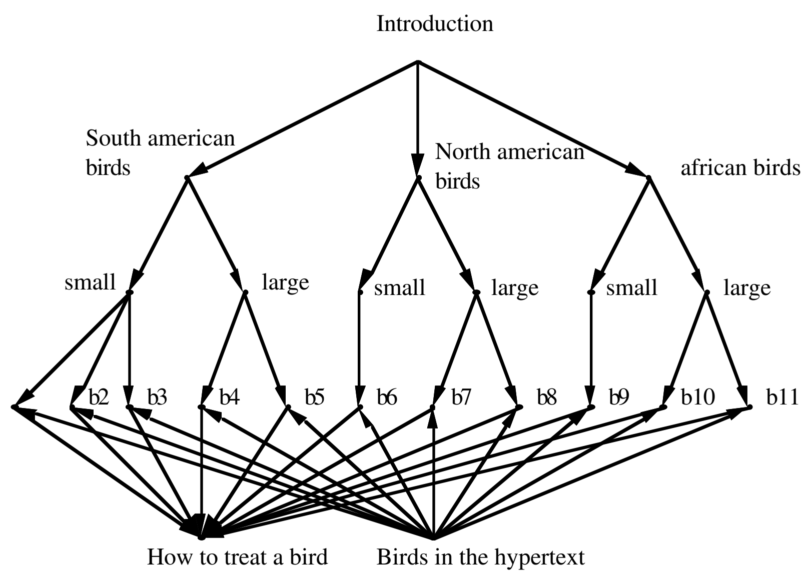

Figure 3 Hypertext where the removal of links is very important for identifying aggregates.

However, a close analysis shows that the relationship between those birds is mainly limited to two facts:

- They are birds. But this is the topic of the hypertext and thus is not a relevant property for clustering, and

• The techniques to treat them are the same. Again, since they are birds it is not surprising that one should use the same process for treating them and no special clustering should come from this fact.

Observe that in this example the nodes “how to treat a bird” and “birds in the hypertext” are reference and index nodes respectively. Those links do not contribute at all in the formation of aggregates and will consequently not be considered when forming aggregates.

In the two subsections that follow two graph theoretic algorithms that identify semantic clusters in a hypertext will be presented. An example showing how they perform in an actual hypertext will also be shown.

2.1 Biconnected components

Definitions:

• An articulation point in a connected graph is a vertex “a” for which there exists two other vertices “v” and “w” such that any path between “v” and “w” does include “a”.

• If a graph has no articulation points then the graph is said to be biconnected.

• The removal of all the articulation points of a graph will split the graph into biconnected components.

Biconnected components in a graph have then the property that there are at least two paths between any two nodes in this component. Finding biconnected components is a quite simple task and can be implemented in O(V+E) [Sed83]. Bicomponents, by the way they are defined, are only applicable to non-directed graph. In the algorithms presented below, the graphs are considered non-directed. A directed graph can easily be made undirected by adding for each link a link in the opposite direction.

Since at least two paths exists between two nodes in a biconnected component, it is likely that bicomponents will be semantic clusters of the hypertext. Using the notions described above, a clustering algorithm by interrelation of nodes can be developed:

Cluster 1:

Step 1) Find the index and reference nodes in the hypertext. If none exist and the algorithm has run at least once then return.

Step 2) Remove outgoing edges from index nodes and incoming edges to reference nodes.

Step 3) Treating the graph G as undirected, find the biconnected components.

Step 4) Build the reduced graph G’ from G.

Step 5) For each of the bicomponents go back to step 1.

Some important features of this algorithm should be observed. First, the algorithm is recursive (step 5). This implies that every bicomponent found will be treated as an independent graph (with fewer nodes than the original graph) and consequently it will be possible to find new index and reference nodes. Finding those nodes and removing their links will allow for a more precise clustering of the hypertext with the intrinsic relationship between nodes assuming an important role.

The second important property of this algorithm lies in step 4. The reduced graph G' of G is a tree. This simplifies enormously the structure of the hypertext that goes from a complex graph to a very simple tree structure. And each new iteration of the algorithm will take a complex graph and reduce it to a tree. Since trees are an easy structure to show to an author, this might help the author have an idea of the contents of the hypertext being created.

Finally the reduced tree G' created has an interesting structure with one level formed by biconnected components (with many nodes in it), the next level formed by articulation points, then again biconnected components, followed by articulation points, and so on. Figure 4 gives an example of this property where the “blobs” represent bicomponents and the “dots” are articulation points. It is possible for a “blob” (a bicomponent) to degenerate and have only 1 node. This will happen for example when the structure is already a tree.

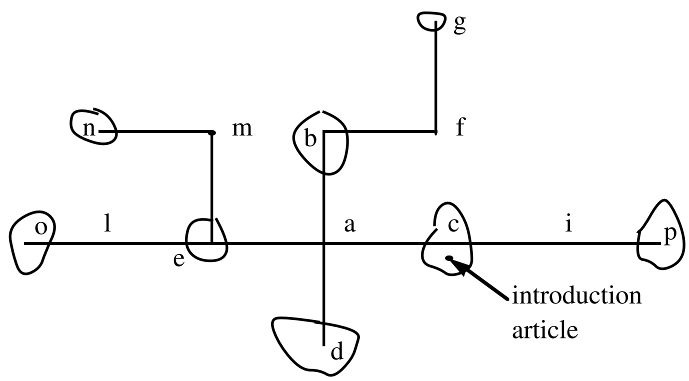

Figure 4(a) Unrooted tree obtained after the bicomponent algorithm has been applied and the reduced graph formed.

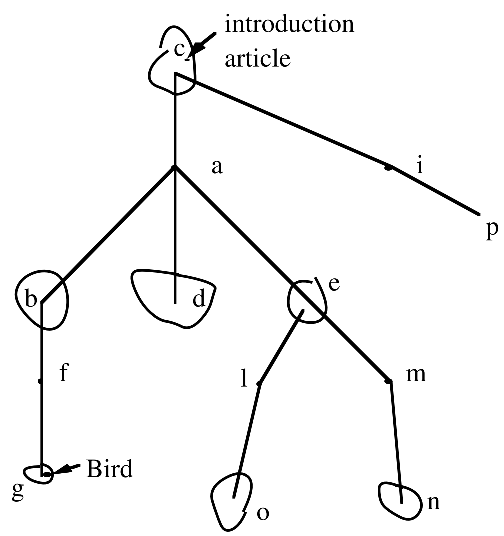

Figure 4(b) To reach the node labeled "Bird" the reader has to read articles "a" and "f."

Figure 4 emphasizes the fact that the reduced tree G' is an unrooted tree. However, we could make it a rooted tree by taking as the root the bicomponent that contained the introduction article or any other article that might qualify as a good root (see [Bot91]). Let T be a rooted tree generated from G'. Given any article in the hypertext it is possible to locate which node of T contains this article, call this node N. By following the path that links the root of T to N, we will pass trough many articulation points. Those articulation points are articles whose reading is required before we are able to read the selected article. Figure 4(b) shows the same structure as figure 4(a), but in it, node “c” was taken as the root of the tree. In this case, to read the article “birds,” readers will have to read articles “a” and “f” before. They will also be required to read some article in “blob” “b.” If this blob deals with a certain subject in particular, the author of the hypertext will know that readers that have reached the article “bird” will have at least some notion of this subject.

It is important for the author to know the articulation points in the hypertext, since in some cases it might not make sense to require the reading of some text before others, while in other cases it might be desired. For instance, suppose an author wants to write a book where at the end of each chapter there is a test that the reader must pass before going on reading. If the test is not an articulation point of the hypertext, then this constraint will not be enforced by the hypertext structure. However, if the node is an articulation point, then there is no way the reader will be able to go on reading without passing the test.

2.2 Strongly connected components

In the previous section, direction were not considered, but many systems such as Hyperties have directional links. Although some researches argue that links should always be bidirectional it seems that in many cases directional links might be useful. In this section we make use of the direction of the links to further enhance the aggregation of nodes.

Definition

• Two nodes “a” and “b” are in the same strongly connected component if there is a path from node “a” to node “b” and a path from node “b” to node “a”

We will improve our previous algorithm by adding step (6):

Cluster 2:

Step 1) Find the index and reference nodes in the hypertext. If none exist and the algorithm has run at least once go to step 6.

Step 2) Remove outgoing edges from index nodes and incoming edges from reference nodes.

Step 3) Treating the graph G as undirected, find the biconnected components.

Step 4) Build the reduced graph G’ from G.

Step 5) For each of the bicomponents go back to step 1.

Step 6) For each bicomponent left, decompose it in strongly connected components.

Several hypertexts ranging in size from 100 nodes and 400 links to 250 nodes and 1600 links were analyzed in order to check the effects of the algorithm presented. The results were similar in all of them and we will focus our discussion on the *Hypertext Hands-On!* book (HHO) because it is widely available [Shn89]. HHO is the first electronic book commercially published. It has 243 nodes and 803 links.

This book was carefully crafted and extensively reviewed to help ensure clear structure and ease of reading. It covers the basic concepts of hypertext, typical hypertext applications, and currently available authoring systems. It also describes design and implementation issues such as user interface, performance and networks. The first iteration of the algorithm decomposed the hypertext into 85 biconnected components and articulation points. Two of those bicomponents are large, with 146 and 31 nodes, while the majority of them are composed of just a few nodes. However, those small bicomponents are parts of separate trees.

Figure 5 shows a picture of part of the reduced graph of the HHO hypertext, when only the first iteration of the algorithm was performed. The central core contains the unbroken 146 nodes and connected to it are three branches of the tree. Although almost all the biconnected components drawn contain only one article, subtrees focus on the same subject. The whole hypertext was subdivided by the algorithm into a large core and 8 subtrees: Tours, Travel Guide, The Interactive Fiction, Resumes, Contracts, Job Aids, Blueprints, and Jokes.

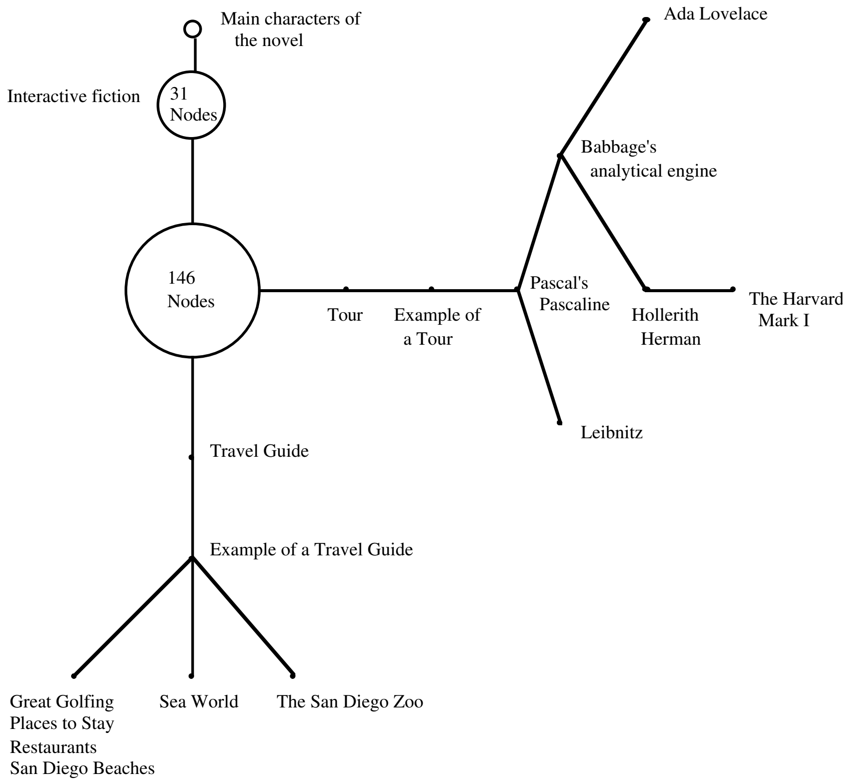

Figure 5 Part of the reduced graph of HHO.

In this first iteration 40% of the hypertext was separated from the core. Even though no experiments were performed to check users understanding of the hypertext it is expected that they will grasp it much more easily when dealing with 9 subparts, where the larger has 146 nodes, than when dealing with a single hypertext of 243 nodes.

HHO comes in two versions: a hypertext disk and a paper book. The two versions although similar are not identical. For example, the interactive fiction appears only in the hypertext version. The book version contains mainly the central core of the hypertext while the subtrees appear only in the hypertext version. This reflects the actual construction process in which the authors first wrote the core of the hypertext and later finished building the disk version by including examples and extra information.

The two larger bicomponents, the core and the interactive fiction, had their compactness measured. For the interactive fiction it was found that: Cp = 0.86. A compactness index of 0.86 is very high. This strong linkage was intended by the author, Robin Platt, whose clever story depends on tight interweaving across nodes. For the core of the hypertext the indices were: Cp = 0.78. Again a very high compactness index. This degree of linkage was debated by the authors. Although redundant links from several points in an article were eliminated, there is still a high level of interconnection. From these examples we see that the core and the interactive fiction are semantic clusters of the hypertext. The compactness of HHO as a whole is Cp = 0.55 indicating that the two pieces are semantic clusters of the hypertext.

The algorithm treated recursively the two large pieces of the hypertext: the Interactive Fiction and the core. No further clustering was done in the interactive fiction or in the core. Since no further breaking was possible with the biconnected algorithm the strongly connected algorithm was run in those two pieces. The Interactive Fiction part was further subdivided in 9 parts. Eight of them have only one article and the ninth has the rest of this subgroup. Almost the same thing happens with the core. It was subdivided in 34 parts, 33 of them containing one or two nodes and the last part containing 103 nodes. While the first iteration decomposition reflects the high level structure as perceived by authors, the second iteration did not so clearly produce the chapter organization that the authors intended. On reflection, emphasizing links within chapters might have been a worthwhile goal. In summary, the clustering algorithm succeeded in revealing the hypertext structure and raised legitimate questions for the authors.

CONCLUSION

By analyzing the structure of a hypertext, i.e., how the nodes are linked, it was possible to identify groups of nodes that had a high semantic relation. We suggested that those nodes should be aggregated to form a more abstract node. With the formation of abstraction it may be possible to simplify the hypertext structure, making it easier for readers and authors to understand the hypertexts and find useful information in them. We believe that structural analysis methods may provide potentially useful ways of giving authors a better understanding of their hypertext so that they can revise it to reduce the “lost in hyperspace” problem. However, our study of three modest-sized hypertexts needs to be repeated with many other hypertexts, and tested with much larger hypertexts to see what adjustments are needed to support scaling up.

DIRECTIONS FOR FUTURE WORK

• We concentrated our analysis on hypertext systems that have a simple underlying structure. Will the ideas that we presented work in more complex hypertext systems? For instance, dealing with typed links may make the analysis more complex, but the additional information has the potential of supporting more effective clustering methods. Another source of relationship information would be textual analysis of article contents.

• Since the hypertexts we studied were authored with Hyperties, one can argue that the system induced authors to produce similarly structured hypertexts. The results need to be replicated on hypertexts created with other systems such as Interleaf, HyperCard, or Guide.

• Authoring should be an interactive activity, with the computer suggesting the formation of abstraction and authors deciding exactly how this process should happen. However, for that to be feasible a good user interface is a must. Showing abstractions over a network is still a difficult problem.

• Only two algorithms for forming aggregation were considered. New algorithms should be studied in order to improve aggregation and verify that it is effective and rapid enough with much larger networks.

REFERENCES

[Bot90] Botafogo, R. A. (1990). *Structural Analysis of Hypertexts*. Unpublished master's thesis, University of Maryland, College Park.

[Bot91] Botafogo, R. A., Rivlin, E., & Shneiderman, B. (1991). Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Transactions on Information Systems* (In Press).

[Bro89] Brown, P. J. (1989). Do we need maps to navigate round hypertext documents? *Electronic Publishing*, *2* (2), 91-100.

[Bro90] Brown, P. J. (1990). Assessing the quality of hypertext documents. Hypertext: Concepts, Systems and Applications; *Proceedings of the European Conference on Hypertext,* 1-12, Cambridge University Press, Cambridge, UK.

[Cro89] Crouch, D. B., Crouch, C. J., & Andreas, G. (1989). *Proceedings of the Hypertext 89 Conference*, 225-237, ACM, New York, NY.

[Eng68] Englebart, D. C. (1968). Authorship provisions in Augment. *Proceedings of FJCC,* 395-410. San Francisco, CA.

[Fei82] Feiner, S., Nagy, S., & van Dam, A. (1982). An experimental system for creating and presenting interactive graphical documents. *ACM Transactions on Graphics 1* (1), 59-77.

[Fei88] Feiner, S. (1988). Seeing the forest for the trees: Hierarchical display of hypertext structure. *Proceedings of the Conference on Office Information Systems,* 205-212, ACM, New York, NY.

[Hal88] Halasz, F. G. (1988). Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM, 31* (7), 836-852.

[Mar89] Marshall, C. C. (1989). Guided Tours and on-line presentations: How authors make existing hypertext intelligible for readers. *Proceedings of the Hypertext 89 Conference*, 15-26, ACM, New York, NY.

[Nie90a] Nielsen, J. (1990). The art of navigating through hypertext, *Communications of the ACM, 33* (3), 296-310.

[Nie90b] Nielsen, J. (1990). *Hypertext & Hypermedia.* Academic Press, Inc., New York, NY

[Sed83] Sedgewick, R. (1983). *Algorithms*. Addison-Wesley Publishing, Reading, MA.

[Shn89] Shneiderman, B., & Kearsley, G. (1989). *Hypertext Hands-On!*. Addison-Wesley Publishing, Reading, MA.

[Smi77a] Smith, J. M., & Smith, D. C. P. (1977). Database abstraction: Aggregation. *Communications of the ACM, 20* (6), 405-413.

[Smi77b] Smith, J. M., & Smith, D. C. P. (1977). Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems*, *2* (2), 105-133.

[Tri88] Trigg, R. (1988). Guided tours and tabletops: Tools for communicating in a hypertext environment. *ACM Transactions on Office Information Systems, 6* (4), 398-414.

[Zel89] Zellweger, P. T. (1989). Scripted documents: A hypermedia path mechanism. *Proceedings of the Hypertext 89 Conference*, 1-14, ACM, New York, NY.