Acknowledgements

First of all I would like to thank my advisors, dr. De Bra and dr. ir. Houben for the many stimulating discussions and for improving my English. Furthermore I am grateful to all people who participated in the experiments and especially to drs. Erica de Vries for her advise on how to conduct these experiments. I would also like to thank my housemates, Alexander van de Pas, ir. Jean Moonen and René de Leijer for their support. And last but certainly not least I would like to thank my parents for their love and encouragement.

Contents

Chapter 1

Introduction

Hypertext is often considered the successor of the book. Apart from the medium in which it is presented most of the time (screen versus paper), the main difference is that hypertext adds non-linearity as a key-feature. Novels are meant to be read in a linear fashion, but books that contain information on a number of subjects are often read in a non-linear way. The pages containing the desired information are looked up in the index or table of contents. No one reads an encyclopedia from page 1 to retrieve information about Rembrandt. A hypertext system supports non-linear retrieval of information in a natural way.

Hypertext can be defined ([4]) as *a database of (textual) information fragments (nodes) that has active links and allows the reader to "jump" to other parts of the database as desired*. The jumps that most hypertext systems offer are following hard-wired links forwards and backwards, jumping to the first node or to a node appearing in a list. As a hypertext system allows a number of jumps in a given hypertext, it defines a structure.

This thesis concentrates on retrieving the structure of a hyperdocument. This structure embodies part of the semantics of the hyperdocument. These semantics are important for readers to retrieve information, as their perception of the hyperdocument's structure is essential for a fast discovery of the information they are searching for. If a user has a good idea of the structure, then the so-called "Lost in Hyperspace" problem will occur less frequently. This "Lost in Hyperspace" problem just consists of the user having no clue of the hypertext's structure and his position in it. It is of course the author's job to make the structure clear and logical. From a structure that is a mess, it may well be impossible to obtain a proper perception. Moreover, little is gained by a proper perception of such structure, because it won't help the user to find the desired information significantly faster.

In order to help users obtain a proper perception of a hypertext structure, different approaches can be followed. For smaller hypertexts a valid approach is to show readers a graphical overview of the hypertext structure. By considering this graphical overview, the users would be able to better understand how the information in the hypertext is linked together. The problem with presenting a graphical overview of a (distributed) hypertext is that it is often too large. Furthermore an incremental local drawing is too restricted to contribute much to the user understanding the complete or a significant part of the hypertext structure. Whenever readers want to find information, they will have to browse through the document in order to mentally construct a perception of the structure of the parts of the document, that obtain the information they are searching for.

In this thesis we consider a number of experiments that are aimed at obtaining a better insight into the relationship between a document's structure and the reader's perception of that structure. The ways in which readers build a perception of a document's structure are influenced both by the navigation strategy they use while browsing, and by the facilities the hypertext systems offers to aid the navigation process. The navigation strategies differ in the way a reader browses through a hypertext. The facilities hypertext systems offer, to help the user navigate through a hyperdocument, differ in the additional moves they offer the user and in the information they supply for constructing a users perception of the structure.

The experiments we describe, try to measure how the perception of a hyperdocument structure depends on the strategy and facilities used by the reader. From the measurements we conclude what the best approach is to browse through a given hypertext, if we are interested in obtaining a good perception of the document's structure. The experiments consist of simulating users browsing through a graph, with a program. By examining whether the structure of the discovered hypertext-part is a good representation of the structure of the hypertext, we draw conclusions on the usefulness of the strategy and facilities used. By doing so, it is obvious that a lot of properties of the simulated users have to be stated. Hence, the crucial properties stated for the simulated users are supported by experiments conducted with "real" users.

1.1 Overview

Chapter 2 describes the different strategies applied by our simulated users and the facilities that may be offered by a hypertext. Furthermore, chapter 2 describes the properties of the users we simulate.

Chapter 3 describes the settings used in the experiments. This chapter also describes the way we differentiate between different kinds of users and hypertexts.

In chapter 4 metrics are described that are used to characterize the structure of a graph. These metrics are important if one wants to say something about the suitability of a strategy for abstracting the structure of a graph.

In chapter 5 conclusions are drawn concerning the efficiency of the different strategies in combination with facilities that may be offered by a hypertext system.

Chapter 6 contains the conclusions drawn on the usefulness of the tested strategies in abstracting a good perception of the structure of a graph.

In chapter 7 we examine the way nodes, accessed due to some navigation strategy, are distributed over the graph.

Chapter 8 describes experiments and results that justify the properties stated on our simulated users in chapter 3.

Chapter 9 discusses the major conclusions of this thesis.

Chapter 2

Navigation Strategies, Facilities & Behaviour

In this chapter we describe the strategies we consider in our experiments. We justify not considering some of the known facilities and we divide the facilities into two categories: a category of *static* facilities and a category of *dynamic* facilities. Furthermore, we discuss properties we state for our simulated users.

2.1 Navigational Strategies

By deciding what links must be considered to be followed next, a navigation strategy determines the way a graph is searched. We distinguish the following strategies:

1. Breadth-first search.
2. Depth-first search.
3. Random search 0.
4. Strategy 1.
5. Strategy 2.
6. Random search 1.
7. Strategy 3.

Remark 1 *By the* current node *we mean the node the user is currently visiting.*

Remark 2 *By a* new *link we mean a link that is not followed yet during the current browsing session.*

Remark 3 *By the next link to follow we mean the next* new *link to follow.*

Definition 1 *A level is the set of all nodes situated at the same distance from the root. For a node, the distance from the root is equal to the minimal number of links that need to be followed from the root to that node, according to the strategy used.*

By a node at a high level we mean a node close to the root regarding the distance from the root to this node.

Remark 4 *The strategies we describe in this chapter are ambiguous. This means that if such a strategy is put into practice, the order in which the nodes are visited is still not fixed. In the examples used to clarify the strategies, a second* hidden *strategy is used:*

Of all links that are considered to be the next link to follow regarding a certain strategy, the one is chosen that leads to the node with the smallest id.

We used this particular *second strategy because it simplifies the examples. In the actual experiments we will not always use* this *second strategy. Other second strategies will be discussed in chapter 3.*

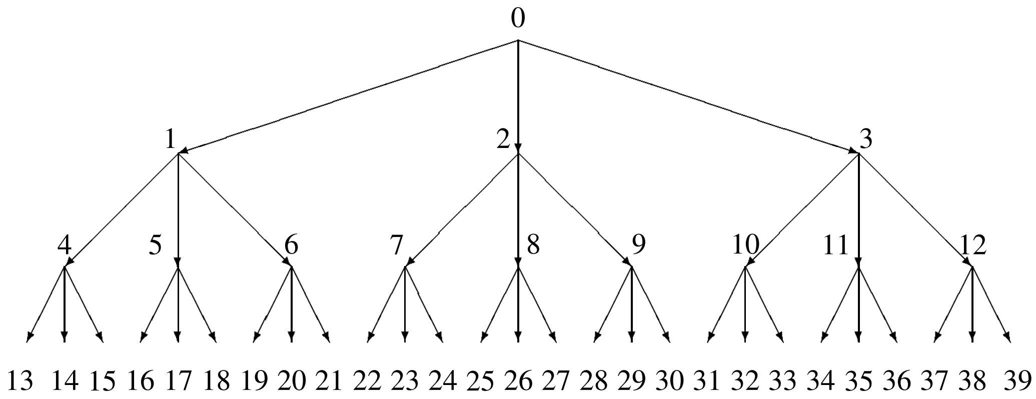


Figure 2.1: Example of a structure

The strategies we enumerated have the following properties:

Breadth-first search: All nodes at a level are considered *before* nodes at a lower level.

Applying breadth-first search on the structure shown in figure 2.1 yields to visiting the nodes of this graph in the following order:

$0 \rightarrow 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \overset{back}{\rightarrow} 0 \rightarrow 3 \overset{back}{\rightarrow} 0 \rightarrow 1 \rightarrow 4 \overset{back}{\rightarrow} 1 \rightarrow 5 \overset{back}{\rightarrow} 1 \rightarrow 6 \overset{back}{\rightarrow} 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \rightarrow 7 \overset{back}{\rightarrow} 2 \rightarrow 8 \overset{back}{\rightarrow} 2 \rightarrow 9 \overset{back}{\rightarrow} 2 \overset{back}{\rightarrow} 0 \rightarrow 3 \rightarrow 10 \overset{back}{\rightarrow} 3 \rightarrow 11 \overset{back}{\rightarrow} 3 \rightarrow 12 \overset{back}{\rightarrow} 3 \overset{back}{\rightarrow} 0 \rightarrow 1 \rightarrow 4 \rightarrow 13 \overset{back}{\rightarrow} 4 \rightarrow 14$, and so on.

Depth-first search: A link is followed that leads to a node situated on a lower level than the current node. If this is no longer possible (the current node has no new links), then the path is followed backwards until a new link is available.

If we apply depth-first search on the structure shown in figure 2.1, then the nodes of the graph are visited in the following order:

$0 \to 1 \to 4 \to 13 \overset{back}{\to} 4 \to 14 \overset{back}{\to} 4 \to 15 \overset{back}{\to} 4 \overset{back}{\to} 1 \to 5 \to 16 \overset{back}{\to} 5 \to 17 \overset{back}{\to} 5 \to 18 \overset{back}{\to} 5 \overset{back}{\to} 1 \to 6 \to 19 \overset{back}{\to} 6 \to 20 \overset{back}{\to} 6 \to 21 \overset{back}{\to} 6 \overset{back}{\to} 1 \overset{back}{\to} 0 \to 2 \to 7 \to 22$, and so on.

The Random search 0 strategy: From all (new) outgoing links from nodes the user *visited previously* (including the current node), the next link to follow is chosen completely at random.

No user will ever follow this strategy because of its inefficiency in terms of the number of steps that have to be done (we will explain this later). Furthermore, a user always has a purpose while browsing through a hypertext. Even if a user wants to browse without the intention of finding a certain piece of information, he won't choose the links to follow completely random. These users tend to follow links close to the current node.

Even though no user will browse through a graph completely at random, we added this "strategy" to compare it with the other strategies for the sake of completeness.

If the random 0 strategy is put into practice while browsing through the graph with its structure shown in picture 2.1, then the order in which the nodes are visited might for example be:

$0 \to 1 \overset{back}{\to} 0 \to 2 \overset{back}{\to} 0 \to 1 \to 4 \overset{back}{\to} 1 \overset{back}{\to} 0 \to 2 \to 7 \overset{back}{\to} 2 \overset{back}{\to} 0 \to 1 \to 4 \to 13 \overset{back}{\to} 4 \overset{back}{\to} 1 \overset{back}{\to} 0 \to 2 \to 7 \to 22$, and so on.

As we can already see before having conducted the actual experiments, such a strategy is not optimal regarding the number of steps that have to be done.

Strategy 1: This strategy involves the *strategy 1 constant*. It behaves the way depth-first search does with the difference that from every node only

$$\left\lfloor \frac{1}{\textit{strategy 1 constant}} \times \textit{outdegree of the node} \right\rfloor,$$

with a minimum of one, outgoing links are considered. When this restricted depth-first search has completed, the remaining links and nodes are searched using the depth-first strategy.

A *strategy 1 constant* equal to 1 leads to performing plain depth-first search.

If strategy 1 is put into practice while browsing through the graph with its structure shown in picture 2.1 (in this example the constant is set on 2, hence half of the outgoing links of the current node, are followed *before* the outgoing (new) links, of the nodes we visited earlier), then the order in which the nodes are accessed is:

$0 \to 1 \to 4 \to 13 \overset{back}{\to} 4 \overset{back}{\to} 1 \overset{back}{\to} 0 \to 2 \to 7 \to 22 \overset{back}{\to} 7 \overset{back}{\to} 2 \overset{back}{\to} 0 \to 3 \to 10 \to 31 \overset{back}{\to} 10 \overset{back}{\to} 3 \overset{back}{\to} 0 \to 1 \to 5 \to 16 \overset{back}{\to} 5 \overset{back}{\to} 1 \to 6 \to 19 \overset{back}{\to} 6 \overset{back}{\to} 1 \to 4 \to 14 \overset{back}{\to} 4 \to 15 \overset{back}{\to} 4 \overset{back}{\to} 1 \overset{back}{\to} 0 \to 2 \to 8 \to 25 \overset{back}{\to} 8 \overset{back}{\to} 2 \to 9 \to 28 \overset{back}{\to} 9 \overset{back}{\to} 2 \overset{back}{\to} 0 \to 3 \to 12 \to 37 \overset{back}{\to} 12 \overset{back}{\to} 3 \overset{back}{\to} 0 \to 1 \to 4 \to 14 \overset{back}{\to} 4 \to 15$, and so on.

As expected, node 14 is not visited directly after node 13. This can be explained by the fact that the outdegree of node 4 equals 3, hence only one outgoing link is directly searched after reaching node 4.

Strategy 2: This strategy involves the *strategy 2 constant*. It behaves the way depth-first search does, with the difference that of every node only *strategy 2 constant*, with a minimum of one and a maximum of the outdegree of the current node, outgoing links are considered. When this restricted depth-first search has completed, the remaining links and nodes are searched with the depth-first strategy.

A *strategy 2 constant* equal to the largest outdegree of any node in the graph, leads to performing a plain depth-first search.

If strategy 2 is put into practice while browsing through the graph with its structure shown in picture 2.1, then the order in which the nodes are visited is (in this example the constant is set on 2. Hence two of the outgoing links (provided that two such links exist) of the current node, are followed *before* the outgoing (new) links, of the nodes we visited previously):

$0 \rightarrow 1 \rightarrow 4 \rightarrow 13 \overset{back}{\rightarrow} 4 \rightarrow 14 \overset{back}{\rightarrow} 4 \overset{back}{\rightarrow} 1 \rightarrow 5 \rightarrow 16 \overset{back}{\rightarrow} 5 \rightarrow 17 \overset{back}{\rightarrow} 5 \overset{back}{\rightarrow} 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \rightarrow 7 \rightarrow 22 \overset{back}{\rightarrow} 7 \rightarrow 23 \overset{back}{\rightarrow} 7 \overset{back}{\rightarrow} 2 \rightarrow 8 \rightarrow 25 \overset{back}{\rightarrow} 8 \rightarrow 26 \overset{back}{\rightarrow} 8 \overset{back}{\rightarrow} 2 \overset{back}{\rightarrow} 0 \rightarrow 3 \rightarrow 10 \rightarrow 31 \overset{back}{\rightarrow} 10 \rightarrow 32 \overset{back}{\rightarrow} 10 \overset{back}{\rightarrow} 3 \rightarrow 11 \rightarrow 34 \overset{back}{\rightarrow} 11 \rightarrow 35 \overset{back}{\rightarrow} 11 \overset{back}{\rightarrow} 3 \overset{back}{\rightarrow} 0 \rightarrow 1 \rightarrow 6 \rightarrow 19 \overset{back}{\rightarrow} 6 \rightarrow 20 \overset{back}{\rightarrow} 6 \overset{back}{\rightarrow} 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \rightarrow 9 \rightarrow 28 \overset{back}{\rightarrow} 9 \rightarrow 29$, and so on.

As opposed to the case where strategy 1 is put into practice, now node 14 is visited directly after node 13. This is a consequence of the fact that strategy 2 always visits two nodes (provided this node has an outdegree larger than 1), that are situated at the end of the outgoing links of the current node, *before* new links are followed that initiate from nodes the user visited before.

Remark 5 *If we take a look at the definitions of strategy 1 and strategy 2, we can see that using certain constants, will cause these strategies to overlap. Hence when conducting the actual experiments, some constant settings for strategy 1 or 2 don't have to be examined since for example, a strategy 1 constant equal to 2, makes this strategy perform the same way as strategy 2 with its constant set to 1, if the graph shown in figure 2.1 is searched.*

The random search 1 strategy: This strategy behaves the way depth-first search does, with the difference that of every node only a randomly chosen number, with a minimum of one, outgoing links are considered. When this restricted depth-first search has completed, the remaining links and nodes are searched with the depth-first search strategy.

If random search 1 is put into practice while browsing through the graph with its structure shown in picture 2.1, the order in which the nodes are visited could for example be:

$0 \rightarrow 1 \rightarrow 4 \rightarrow 13 \overset{back}{\rightarrow} 4 \rightarrow 14 \overset{back}{\rightarrow} 4 \overset{back}{\rightarrow} 1 \rightarrow 5 \rightarrow 16 \overset{back}{\rightarrow} 5 \overset{back}{\rightarrow} 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \rightarrow 7 \rightarrow 22 \overset{back}{\rightarrow} 7 \rightarrow 23 \overset{back}{\rightarrow} 7 \rightarrow 24 \overset{back}{\rightarrow} 2 \overset{back}{\rightarrow} 0 \rightarrow 3 \rightarrow 10 \rightarrow 31 \overset{back}{\rightarrow} 10 \overset{back}{\rightarrow} 3 \overset{back}{\rightarrow} 0 \rightarrow 1$, and so on.

Strategy 3: At first the depth-first search strategy is put into practice until a certain level is reached. What level, is determined by the $level - constant$. The node the user is visiting at the moment level $level - constant$ is reached, can be looked at as the root of a subgraph. The user now searches this subgraph using one of the other described strategies. If the subgraph is searched completely, the steps described are repeated until the graph is fully searched or the maximum number of allowed steps is reached[1].

If strategy 3 is put into practice while browsing through the graph with its structure shown in picture 2.1, then the order in which the nodes are visited is (in this example the $level - constant$ is set on 2, so a straight path is followed until a node at level 2 is reached. After that, the subgraph with this node as its root is searched using the breadth-first strategy.):

$0 \rightarrow 1 \rightarrow 4 \rightarrow 13 \overset{back}{\rightarrow} 4 \rightarrow 14 \overset{back}{\rightarrow} 4 \rightarrow 15 \overset{back}{\rightarrow} 4 \overset{back}{\rightarrow} 1 \rightarrow 5 \rightarrow 16 \overset{back}{\rightarrow} 5 \rightarrow 17 \overset{back}{\rightarrow} 5 \rightarrow 18 \overset{back}{\rightarrow} 5 \overset{back}{\rightarrow} 1 \rightarrow 6 \rightarrow 19 \overset{back}{\rightarrow} 6 \rightarrow 20 \overset{back}{\rightarrow} 6 \rightarrow 21 \overset{back}{\rightarrow} 6 \overset{back}{\rightarrow} 1 \overset{back}{\rightarrow} 0 \rightarrow 2 \rightarrow 7 \rightarrow 22 \overset{back}{\rightarrow}$

[1] See section 3.2.1.

$7 \rightarrow 23 \stackrel{back}{\rightarrow} 7 \rightarrow 24 \stackrel{back}{\rightarrow} 7 \stackrel{back}{\rightarrow} 2 \rightarrow 8 \rightarrow 25 \rightarrow 8 \rightarrow 26 \stackrel{back}{\rightarrow} 8 \rightarrow 26 \stackrel{back}{\rightarrow} 8 \rightarrow 27 \stackrel{back}{\rightarrow} 2$, and so on.

2.2 Facilities

Browsing facilities support the user in building a perception of the graph structure in his mind (and of course the discovery of the information he is looking for.) Hypertexts usually offer the user a combination of the facilities shown below. These utilities are the most common ones [1][5]:

1. Bread crumbs.
2. Backtracking.
3. The highlighting of links.
4. The history list 0.
5. The history list 1.
6. The bookmark facility.
7. The ability to save bookmarks.
8. Thumb tabs.
9. Graphical overviews.

Breadcrumbs: We won't take the breadcrumbs facility into account because of the property of our simulated users, described in assumption 1.

Assumption 1 *The user we simulate, is supposed to have a perfect memory concerning the recognition of the current node, as being visited before. Hence, if he visits a node, the user is able to determine whether he visited this node before or not.*

This assumption appears to be too strong to resemble human user behaviour. It is however indispensable! If a user is not able to recognize nodes as being visited before, he is also not able to obtain a proper perception of a hypertext structure by browsing it. Such user will probably search a part of the graph more than once. As a result he thinks he has visited a larger number of nodes than he actually did, and there is no way this user obtains the right picture of the structure of the graph!

Of course no *human* user has a perfect memory. But he is often helped by one of the static[2] facilities offered to the user. Especially the breadcrumbs-facility will help the user recognize nodes as being visited before. Although we do not consider this facility in our experiments, if implemented it helps users behave in a way according to the property described in assumption 1. Moreover, the experiments described in chapter 8 show us that the average user does recognize nodes as being visited before.

[2] See paragraph 2.3, for the definition of a static facility.

Assumption 2 *For a user to get an idea of the structure of a graph, it is not enough to just recognize a node as being visited before. He must also be able to recall its position in the graph. Therefore we presume that the user has this ability.*

Assumption 2 seems a bit exaggerated. It is however not possible to get a good idea of the structure if the user only knows that "he has seen a certain node before", and doesn't know when or where. Hence assumption 2 is indispensable. Again the experiments described in chapter 8 affirm that the average user has this property.

Backtracking: The backtracking facility enables the user to go back to the node he visited prior to the current node, even if there exists no link that makes this specific move possible.

Because of the indispensability of this backtracking-facility, we assume it is always available to the user. This means that in every experiment we conduct, the backtracking-facility is considered. Without backtracking mechanism a user would be stuck when reaching a node with no outgoing links. Moreover, every hypertext we worked with offered the user the backtracking-facility.

Highlighting links: This facility marks anchors, that represent links that lead to nodes the user visited before, differently from anchors representing links that lead to "new" nodes. It also gives the user information about the node the link (represented by the anchor) leads to. Thus supplying valuable information on the hyperdocument structure.

The first mentioned property just speeds up searching. The last mentioned property is essential if a user wants to discover the structure of a hyperdocument. To get an idea of the structure, it is not enough to know that a link leads to a node visited before. The user must be able to tell to *what* earlier visited node it leads.

History list 0: The history list 0 facility offers the user a complete list of nodes he visited before. Hence there is no limit to the number of nodes in this list. By selecting a node in the list the user is able to jump to that node, as if a link exists from within the current node to the node he chose from the list.

Some browsers that offer this facility, use a restricted history list. They only allow a limited number of nodes in it. The maximum number of nodes in such a list provided by these browsers, is however big enough to be simulated with an unrestricted one.

History list 1: The history list 1 facility is a restriction of the history list 0 facility. The history list 1 facility doesn't keep track of *all* visited nodes. Instead, it offers the user a list containing the nodes that are positioned on the direct "path" from the root to the current node. Our implementation of this history list 1 corresponds to the one used by Mosaic for X. Its behaviour is explained by the following example.

Example 1 *If the structure shown in figure 2.2 is browsed through then the history list 1 (*HL1*) contains: (the contents of the history list 1 at each moment are printed at the right of the id of the current node)*

0, HL1: [] $\rightarrow$ 1, HL1: [0] $\rightarrow$ 2, HL1: [0|1] $\rightarrow$ 3, HL1: [0|1|2] $\rightarrow$ 2, HL1: [0|1] $\rightarrow$ 4, HL1: [0|1|2] $\rightarrow$ 5, HL1: [0|1|2|4] $\rightarrow$ 4, HL1: [0|1|2] $\rightarrow$ 6, HL1: [0|1|2|4].

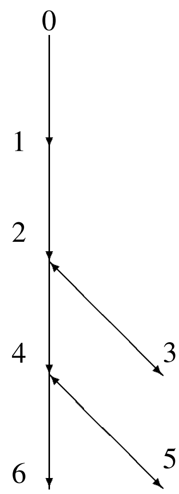


Figure 2.2: Structure of a graph

Bookmarks: This facility enables the user to mark a node with a bookmark. In case a node is marked, it is possible to jump to that node from every location in the graph as if a link exists from the current node to the node he chooses to jump to.

Placing bookmarks on every visited node resembles the use of the history 0 facility.

Saving bookmarks: This facility allows the user to save bookmarks for future use. Because only browsing behaviour during *one* session is investigated, we won't take this facility into account while conducting the experiments.

Thumb tabs: The thumb tab facility offers the author of the document the possibility to place bookmarks on nodes he thinks are important to all users. It is typically used by the author to create index and reference nodes. Because of the fact that this facility is *only* presented to the author, we won't investigate it.

The graphical overview facility: This facility shows the user the complete structure or a part of it. This facility is not considered. The problem with presenting a graphical overview of a hypertext structure is that it is often too large to be shown entirely and hence only a part of it can be shown. And the problem with showing just a part of the structure is that it is difficult to determine what part should be shown to give the user a proper idea of the complete structure. Often parts of the structure are shown that have not been searched yet. This means that the perception of the hypertext structure may be disturbed by this facility. Hence, some of the results of the experiments we will conduct may also be useful when designing a graphical overview facility.

Some of these facilities also enable the user to jump to a node shown in the overview. We may look at this as a variant of the bookmark facility and therefore it does not need to be considered separately.

2.3 Categorizing the facilities

We divide the described facilities into two categories. The first category contains the highlighting link facility, the bread-crumb facility and the graphical overview facilities. We will call these facilities *static*. The second category contains the rest of the facilities. We will call them *dynamic*. A static facility has the property of *not adding* any possible actions. This facility just gives the user some extra information. This information can be used to speedup the search. What actions, concerning this facility, are possible, *only* depends on the structure of the hypertext. Whereas in case of a dynamic facility the number of possible actions can be expanded by using these facilities. If for instance, the structure of the hyperdocument contains a link from node A to node B, (and no link from node B to node A), then the backtrack facility *adds* the possibility of, after following that link, going back to node A. This move is impossible if the backtrack mechanism is not available. This also holds if a history list is applied. In that case, when reaching node B, node A is an element of the history list and therefore directly reachable from node B. By using the bookmark facility, one can place a bookmark on node A and therefore is able to directly jump from node B to node A. So the use of one of these facilities makes it possible to (sometimes) jump to a node that is not directly accessible according to the structure of the document.

2.4 User behaviour

The users we simulate in the experiments have the following properties:

1. A perfect memory concerning the recognition of a node visited before. See assumption 1.

2. A perfect memory concerning the position of a node recognized as being visited before. See assumption 2.

3. If the user wants to navigate to the node that initiates the next link to follow, then he performs backsteps until he reaches a node that is situated on the last followed path to the node he wants to visit. After that, he follows this path until he reaches the desired node.

   As a result our user won't try to find some other path to get to the desired node.

4. A user only retrieves information from a highlighting link if this link would have been the next link to follow if there was no highlighting link facility available.

   Since some browsers that support a highlighting link facility, make the user place the mouse over the anchor, to see to what node it leads, or make him perform some other action to get the same result, this behaviour seems reasonable. With most hyperdocuments it is not the case that just looking at the anchor leads to the knowledge of to what node, the link it represents, leads to. Hyperties for example, gives the user a short description of the node a link leads to after clicking the corresponding anchor. A double click on the anchor makes the browser follow the link "connected" to this anchor.

   Different anchors presented to the user may represent links that lead to the same node. As an example we can take the HTML-language that is used to construct hypertexts. This language enables the author to present an anchor with a picture or some text. Hence the author, and *not* the node it leads to, determines what the anchor looks like.

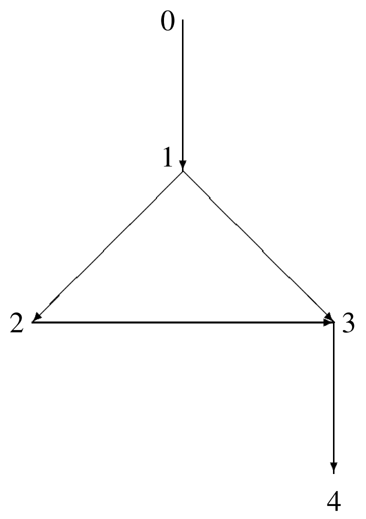


Figure 2.3: The structure of a graph

Example 2 *If we look at the structure shown in figure 2.3 and the user conducts a depth-first search browsing through it, then the nodes are visited in the following order (*no *highlighting link facility available):*

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \stackrel{back}{\rightarrow} 3 \stackrel{back}{\rightarrow} 2 \stackrel{back}{\rightarrow} 1 \rightarrow 3.$

Example 3 *If we put depth-first search into practice while browsing through the structure shown in figure 2.3, using the highlighting link facility, then our user accesses the nodes in the following order:*

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \stackrel{back}{\rightarrow} 3 \stackrel{back}{\rightarrow} 2 \stackrel{back}{\rightarrow} 1$

As we can see, the user does *not* follow the link that leads from node 1 to node 3. This is a result of the highlighting link facility showing the user that this link leads to a node he visited before.

However, if a link is highlighted *and* it leads to a node that initiates the next link to follow, then the link is followed.

Example 4 *If our user is browsing through the structure shown in figure 2.3, using the breadth-first search, and the highlighting link facility is available, he navigates through the graph as follows:*

$0 \rightarrow 1 \rightarrow 2 \stackrel{back}{\rightarrow} 1 \rightarrow 3 \stackrel{back}{\rightarrow} 1 \rightarrow 2 \stackrel{!!!}{\rightarrow} 3 \rightarrow 4$

and not:

$0 \rightarrow 1 \rightarrow 2 \stackrel{back}{\rightarrow} 1 \rightarrow 3 \stackrel{back}{\rightarrow} 1 \rightarrow 2 \stackrel{back}{\rightarrow} 3 \stackrel{back}{\rightarrow} 2 \stackrel{back}{\rightarrow} 1 \rightarrow 3 \rightarrow 4$

Example 5 *If the structure in figure 2.3 is browsed through applying breadth-first search, then the following path is followed (*no *highlighting link facility available):*

$0 \rightarrow 1 \rightarrow 2 \stackrel{back}{\rightarrow} 1 \rightarrow 3 \stackrel{back}{\rightarrow} 1 \rightarrow 2 \rightarrow 3 \stackrel{!!!}{\rightarrow} 4.$

Notice that, as a result of property 3, after visiting node 3 for the second time, the link leading to node 4 is followed without first performing: $3 \stackrel{back}{\rightarrow} 1 \rightarrow 2 \rightarrow 3.$

5. Our user *only* 'jumps' to a node contained in one of the history lists or to a node carrying a bookmark, if this node initiates a link the user wants to follow next. The reason for the restrictive use of the history lists and bookmark-facility, is the fact that it is very hard (if not impossible) to recollect what bookmark is placed somewhere on a path to the next link to follow. Hence, a more extensive use of this facility is not considered.

Chapter 3

Graphs & Experiment settings

In this chapter we discuss the settings in which the experiments are conducted. We will show what graphs are used, and we will show the values of the different constants used in the experiments.

3.1 The graphs

In our experiments we will browse the following graphs:

1. 5 randomly generated graphs with hypertext-like structures.
   - The first graph (graph 0) contains 80 nodes and 97 links.
   - The second graph (graph 1) contains 100 nodes and 224 links.
   - The third graph (graph 2) contains 100 nodes and 210 links.
   - The fourth graph (graph 3) contains 100 nodes and 188 links.
   - And the last randomly generated graph (graph 4) contains 100 nodes and 127 links.
2. - The HHO structure: This structure belongs to a commercial hypertext [4]. The graph representing the structure of HHO contains 238 nodes and 687 links. (After deleting an index and a reference node. These nodes disturb the structure of the hypertext [1].)
3. 3 structures generated by Erica de Vries [3] for experiments conducted with human users.
   - The first one (E0) has a network-like structure. This graph contains 57 nodes and 225 links;
   - the second one (E1) is strictly hierarchical. This graph contains 81 nodes and 80 links. This combination of number of nodes and number of links comes to no surprise since the strictly hierarchical structure is just a tree-structure, which implies a number of links equal to the number of nodes minus one;
   - and the last one (E2) is a combination of E0 and E1. This graph contains 81 nodes and 225 (E0) + 80 (E1) = 305 links.

All graphs are fully connected and it is clear what node has to be chosen to be the root.

3.2 Experiment settings

Every graph we conduct our experiments on, will be browsed through using all combinations of navigation-strategies and browser-facilities. (By all combinations of browser facilities, we mean all combinations of dynamic and static facilities. Since they overlap, there is no use in combining dynamic facilities. Think of the history list 0, history list 1 and bookmark facilities.)

Definition 2 *One step either represents:*

1. *Following a single link for the first time;*
2. *following a single link for the second or more time;*
3. *moving one step back using the backtracking-facility;*
4. *jumping to a node contained in one of the history lists;*
5. *jumping to a node that is marked with a bookmark.*

The steps defined above do not only exist of the selection of a link, a backstep or some jump. They also include the system providing the selected information and the user picking up the desired information. What information the user needs to retrieve from a node depends on the situation he is in. See assumption 3.

Assumption 3 *If a user reaches a node for the first time, then more or at least the same time is spent on absorbing the information contained in this node, than is the case if a node is visited for the second or more time.*

The reason for this assumption is the fact that in our opinion, if a node is reached for the first time, the same or more information is absorbed then in case the node was visited before. Moreover, the results from the experiments conducted in chapter 8 support this assumption.

To simulate different kinds of users we assign different weights to the different steps.

1. The first category contains users who spend equal time on every step.

   This behaviour is of course a little strange. But it represents an extreme user behaviour. The time such a user needs to get to a node merely depends on the number of steps that have to be done and *not* on *the kind* of steps performed.

2. The second category contains users that do not need much time to recognize a node as being visited before, compared to the time needed to pick up all the information contained in a node that is visited for the first time.

To what category a user belongs also depends on the properties of the hypertext the user is browsing through. These properties are:

1. The average amount of information stored in the nodes of the hypertext:

   A hypertext containing a lot of information in each node, implies a high percentage of time spend by the user retrieving information, and a low percentage of time spend on actually navigating through the hypertext. Whereas if the average amount of information is small, the overhead of plain navigation grows.

2. The way the information is presented to the user:

   Information containing pictures or other graphics probably lead to faster recognition than information that merely consists of text. It also depends on the layout and colors used [3].

3. To what category of hypertexts it belongs.

   We will distinguish two categories:

   Category 1 hypertexts: This category contains hypertexts like HHO. An element of this category has the following property:

   The time between the selection of a link by the user and the hypertext showing the contents of the node at the end of the selected link, is short.

   Category 2 hypertexts: This category contains hypertexts, like Mosaic for X, that for example use the Internet. These hypertexts need a lot more time to show the information contained in the node at the end of the selected link.

   If of course, the hypertext has some cache facility to store the information contained in nodes the user visited before, it belongs to category 1. Such a hypertext doesn't have to retrieve the information of a node visited before, by linking to the source of the information, sending a request, and waiting for the information to arrive. It just retrieves the desired information from its memory, so the time spend on retrieving this information is considerably shorter.

So a category 1 hypertext that presents the information in such manner that it is difficult to distinguish different nodes and that has a complex structure, implies that most users browsing through this hypertext will automatically become users that behave the way category 0 users do. This also counts for hypertexts that contain *very* little information in a node.

The differences between the categories of hypertexts and the categories of users described above can be expressed by assigning different weights to the steps performed by the user while browsing through a hypertext. Major difference is the time spend on following links the user followed before.

If the hypertext belongs to the first category and so does the user, then following a link followed before, takes about as much time as taking a backstep. Whereas a second category user, browsing through a hypertext belonging to the second category, needs less time to perform a backstep than following a link he followed before. Hence, if a hypertext is browsed through that belongs to the first category, the time needed to perform a step that represents following a link the user followed before, will be set equal to the weight of a backstep. Whereas in case of a hypertext that is an element of the second category, the time needed to follow a link that was followed before will be considerably larger.

Some users need much time to decide what link to follow next after performing a backstep. This may be a result of a very complex structure in combination with nodes that are hard to distinguish from another. We simulate this by assigning a weight to a backstep equal to the weight of the step that represents following a link for the first time.

The weights assigned to the different steps are:

- A user of category 1 and a hypertext of category 1 or a user of category 1 and a hypertext of category 2:

  1. Following a link the first time: weight = 1;

2. Following a link followed before: weight = 1;
3. Performing a backstep: weight = 1;
4. Jumping to a node in one of the history lists: weight = 1;
5. Jumping to a node a bookmark is placed on: weight = 1.

In the remaining part of this paper we will refer to these settings as *group 0.*

- A user of category 2 and a hypertext of category 1:

1. Following a link the first time: weight = 8;
2. Following a link followed before: weight = 1;
3. Performing a backstep: weight = 1;
4. Jumping to a node in one of the history lists: weight = 1;
5. Jumping to a node a bookmark is placed on: weight = 1.

In the remaining part of this paper we will refer to these settings as *group 1.*

- A user of category 2 and a hypertext of category 2:

1. Following a link the first time: weight = 8;
2. Following a link followed before: weight = 6;
3. Performing a backstep: weight = 1;
4. Jumping to a node in one of the history lists: weight = 1;
5. Jumping to a node a bookmark is placed on: weight = 1.

In the remaining part of this paper we will refer to these settings as *group 2.*

The weights assigned do *not* represent all users. Slow or fast readers can be represented by larger respectively smaller weights assigned to the step representing following a link for the first time. Our weight assignments just register the difference between the categories of users and hypertexts. We are *certainly not* suggesting that they represent all the different hypertext systems and user categories. See chapter 8.

3.2.1 Browsing time

The results from the experiments conducted by Erica de Vries [2] with real users browsing through hypertexts, show us that depending on the goal given to the users, the average number of different nodes visited is:

- about 45, if the users are asked to just browse around in the hypertext. This implies a total weight of steps performed of about 650 if the step that represents following a link for the first time is weighted 8. We will however take our allowed maximum total weight somewhat higher, since the hypertexts we conduct our experiments on are larger on the average, (see the description of those graphs.) We allow our simulated user a maximum total weight of 800, if the weight assigned to following a link for the first time equals 8. If the settings are such that this weight equals 1, then a maximum total weight of 100 is allowed.

- If users are asked to find a certain piece of information in the hypertext, then the total number of different nodes visited by these users is about 25 on the average. This implies a total weight of the steps performed of about 250 if the step that represents following a link for the first time is weighted 8. We will (again) choose the maximum total weight to be somewhat higher, since the hypertexts we conduct our experiments on are larger on the average. We allow our simulated user a maximum total weight of 320 if the weight assigned to following a link for the first time equals 8. If the settings are such that this weight equals 1, then a maximum total weight of 40 is allowed.

To draw conclusions on the speed of the strategies in combination with the different facilities, we will also simulate users allowing a total weight of ∞, to allow them to search this graph completely.

Hence, we conduct all our experiments on all the graphs with a maximum number of:

1. 40 respectively 320 steps;
2. 100 respectively 800 steps and
3. enough steps to *discover* every link in the graph.

   In case no highlighting link facility is offered, *discover* can be read as "follow". See the description of the highlighting link mechanism in section 2.2.

3.2.2 The remaining settings

- While simulating a user who is offered the bookmark-facility, bookmarks are placed randomly. The experiments conducted using the bookmark facility were repeated several times using different numbers of bookmarks placed. These numbers were chosen to be high, because the chance at placing a bookmark that is never used is quite big.
- Each combination of a random search strategy and a browser facility is tested a number of times to avoid extremities of the random search strategy.
- The experiments conducted with Strategy 1 and 2 were conducted with their constants set to 1, 2, 3 and 4.
- The experiments involving strategy 3 are conducted using $level - constant$s equal to 2, 3 and 4.
- If a strategy is applied and, regarding this strategy, multiple links are considered to be the next link to follow, then a link is chosen randomly. Hence the strategies tested in the experiments sometimes differ from the ones described in section 2.1. The examples in section 2.1 used a *second* strategy that chooses the link that leads to the node with the smallest id to be the next link to follow.

Chapter 4

Metrics that classify a structure

Since the purpose of this research is to advise a user what browsing strategy leads to the best understanding of the structure of a hypertext, we need tools to classify structures. We distinguish three metrics, borrowed from [1], that tell us something about the structure of a graph. The next section describes the way these metric values are calculated.

4.1 Calculating metric values

The metrics we distinguish are:

1. The number of cross-reference links in the graph and the percentage cross-reference links of the total number of links.
2. Compactness.
3. Stratum.

The combination of the values of these metrics for a given graph, gives us an idea of its structure.

Cross-reference links: The number of cross-reference links in a graph can be found by computing a "spanning tree" of that graph. This is done by grabbing the structure at its root and letting the links dangle down. The number of links that do not dangle vertically equals the *number* of cross-reference links in the graph.

In figure 4.2 the structure of a graph is shown. Figure 4.3 shows the same graph after grabbing it at node 0. The links shown that are not straight, are the cross-reference links that are discovered by applying breadth-first search. Figure 4.4 shows the cross-reference links (the ones that are not straight) in case depth-first search is applied. If we compare the figures showing the cross-reference links according breadth-first and depth-first search, we observe that the *number* of cross-reference links does not depend on the strategy used. Whereas, *what links* are considered to be cross-reference ones, completely depends on the strategy applied.

As a result we draw the conclusion that when classifying a structure, that is searched by our simulated user, we should only look at the number of cross-reference links found. Hence it is not important *what* links are considered cross-reference ones according to the strategy used. Moreover, it is impossible, by just browsing a graph, to tell what links are intended to be cross-reference ones, by the author of the document.

Compactness: This metric shows us whether each node can easily be reached from another node. The compactness metric assumes values that vary between 0 and 1. A value of 1 indicates that each node can easily reach any other node in the graph, suggesting a large amount of cross-reference links. In fact, a compactness value of 1 indicates a complete graph (each node is connected to every other node). A small value for the compactness metric indicates that on the average, a lot of links have to be followed to navigate from one node in the graph to another, suggesting a small amount of cross-reference links. We can already see that these metrics are closely related.

To calculate the compactness value for a given graph we need the following definitions:

Definition 3 *Let D be the distance matrix representing a graph. Then the converted distance matrix C of that graph is calculated as follows:*

$$C_{ij} = \begin{cases} D_{ij} & \textit{if } D_{ij} \neq \infty. \\ K & \textit{if } D_{ij} = \infty. \end{cases}$$

Where K is a constant, the conversion constant.

Figure 4.1 shows a graph with its distance matrix and its converted distance matrix.

The compactness value C_p for a graph is calculated as follows:

$$C_p = \frac{(MAX - \sum_i \sum_j C_{ij})}{(MAX - MIN)}. \tag{4.1}$$

with MAX and MIN defined as

$$MAX = (n^2 - n)\mathbf{C}\ ;\ \ MIN = (n^2 - n). \tag{4.2}$$

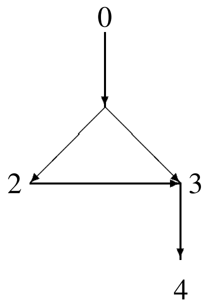
where n is the number of nodes in the graph and **C** the maximum value an entry in the converted distance matrix can assume. Usually $\mathbf{C} = K$. Hence, if we calculate the compactness values for the graph shown in figure 4.1 we get:

Let us assume $C = K = 10$, then

$n = 5$, so $MAX = 200$ and $MIN = 20$,

hence $C_p = \frac{(200 - (16 + 10K))}{180} = 0.47$.

When the compactness value is calculated for a graph during the experiments, the conversion constant K is assigned a value equal to the number of nodes of the graph. This value gives us the best insight in the structure of the graph. A conversion-constant equal to the number of nodes in the graph leads to compactness values that are spread more widely than in case a larger conversion constant is used. A high value for K means that it becomes very important that from every node all other nodes can be reached via some path. To put it differently; it is



	0	1	2	3	4
0	0	1	2	2	3
1	∞	0	1	1	2
2	∞	∞	0	1	2
3	∞	∞	∞	0	1
4	∞	∞	∞	∞	0

	0	1	2	3	4
0	0	1	2	2	3
1	K	0	1	1	2
2	K	K	0	1	2
3	K	K	K	0	1
4	K	K	K	K	0

Figure 4.1: A graph with its distance matrix and converted distance matrix representation.

very important that the graph is connected. Looking at the structures of the graphs we conduct our experiments on, this is always the case. Hence the best value for K is a value as small as possible. Such value must be larger than all the other values in the converted distance matrix, hence it equals the number of nodes in the graph, because the longest possible distance between two nodes in a (connected) graph equals the number of nodes minus one.

Example 6 *If K is set equal to 5 and we calculate the compactness value for the graph shown in figure 4.1 we get:*

$C = K = 5$ *then*

$n = 5$, *so* $MAX = 100$ *and* $MIN = 20$,

hence $C_p = \frac{(100 - (16 + 10K))}{80} = 0.42$.

Stratum: The stratum metric indicates whether it is important to begin reading the hypertext from a particular node. If the structure of a hypertext is linear, then there exists only one node that can be picked as a starting point from which it is possible to reach all other nodes. A linear structure has a stratum value of 1. If on the contrary the structure is circular then all nodes can be reached from all other nodes. This structure has a stratum value of 0.

The value for the stratum metric is calculated as follows:

Definition 4 *The Linear Absolute Prestige (LAP) is given by the following formula:*

$$LAP = \begin{cases} \frac{n^3}{4}, & \text{if } n \text{ is even.} \\ \frac{n^3 - n}{4}, & \text{if } n \text{ is odd.} \end{cases}$$

Definition 5 *The* status *of a node equals the sum of all distances to all other nodes, as given by the distance matrix. Distances equal to ∞ are not considered. See figure 4.5.*

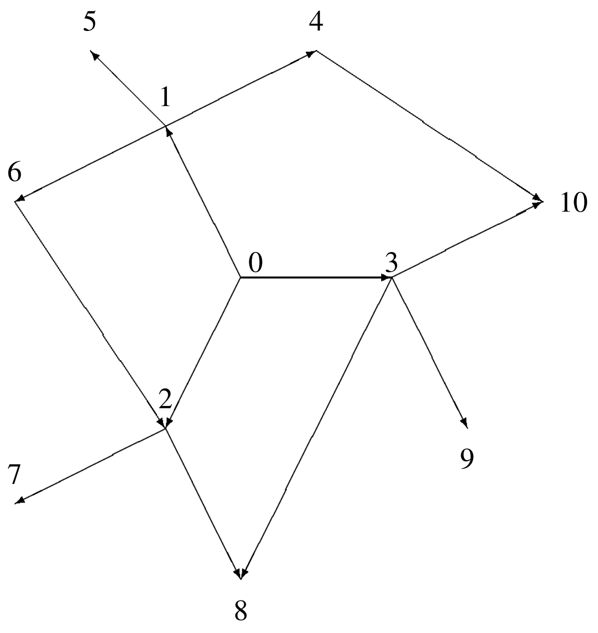


Figure 4.2: The structure of a graph.

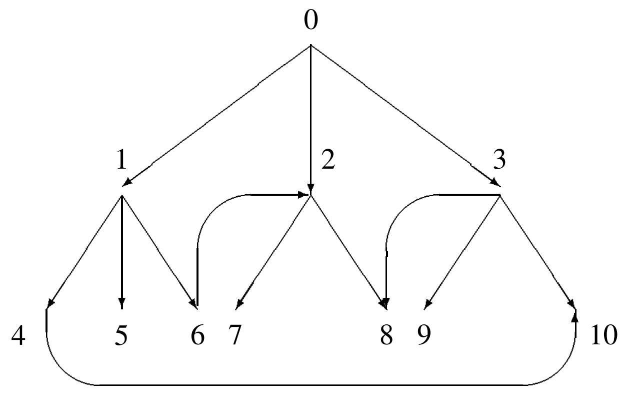


Figure 4.3: The same structure presented hierarchically (Breadth-first search).

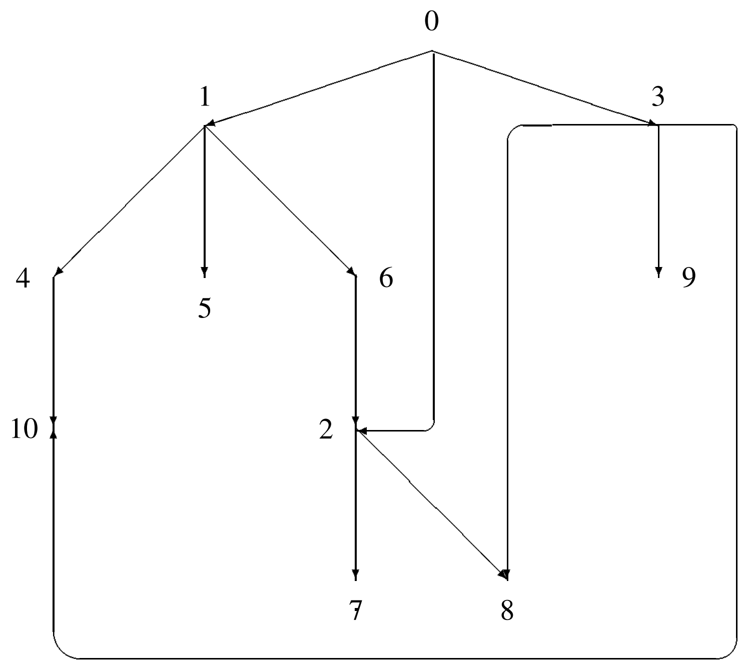


Figure 4.4: The same structure presented hierarchically (Depth-first search).

Definition 6 *The* contrastatus *of a node equals the summation of all distances, from all other nodes to this node, as given by the distance matrix. Distances equal to* ∞ *are not considered. See figure 4.5.*

Definition 7 *The Absolute Prestige is given by:*
AP = $\sum_i(|status_i \; - \; contrastatus_i|)$

The stratum value of a graph equals AP/LAP.

In figure 4.5 the contrastatus and status values for the graph in figure 4.1 are shown.

From these values we can calculate AP.

AP = 22, so the stratum value for this graph equals $22/30 \; = \; 0.73$.

4.2 The way we use the metrics

In the experiments we conduct, we simulate users browsing through a graph. These users browse the graphs using a limited number of steps. If a user is finished browsing, we examine the subgraph he discovered. We differentiate between two types of discovered subgraphs:

1. The subgraph consisting of all visited nodes and all followed links plus all links *not* followed due to using the highlighting link facility.

	0	1	2	3	4	status
0	0	1	2	2	3	8
1	∞	0	1	1	2	4
2	∞	∞	0	1	2	3
3	∞	∞	∞	0	1	1
4	∞	∞	∞	∞	0	0
contrastatus	0	1	3	4	8	$AP = 22$

Figure 4.5: Status and contrastatus.

2. The subgraph consisting of all visited nodes and all followed links plus all links *not* followed due to using the highlighting link facility *plus* the links represented by anchors that are ignored due to the strategy followed or due to the limited number of steps.

   Since the anchors representing some links are ignored, the user is not able to tell to what node they lead[1]. Hence to the subgraph, links represented by these anchors, are added that lead to *new* nodes.

   Example 7 *If the graph shown in figure 4.2 is searched putting depth-first search into practice and only three links are followed, then the subgraph discovered according to item 1 resembles the one shown in figure 4.6 on the left.*

   The subgraph according to item 2 is shown in figure 4.6 on the right.

The metric values of each subgraph are calculated. By comparing these values to the values obtained from the complete hypertext structure, we can draw conclusions on the usefulness of a strategy, in giving a user a proper perception of the hypertext structure. We do this by looking at the combination of the values of the described metrics. We thereby keep in mind that the subgraph according to item 1 probably has a larger compactness value than the subgraph according to item 2 (fewer nodes and the same number of cross-reference links).

We concentrate on the discovery of the cross-reference links and the value for the compactness metric. (The results for the stratum metric are generally worse). We have to keep in mind that the compactness and stratum values differ from the values for the complete hypertext structure.

4.3 Metric values of the graphs used in the experiments

Graph 0: Number of cross-reference links: 18;

Percentage cross-reference links: 19;

[1]See item 4 on page 14.

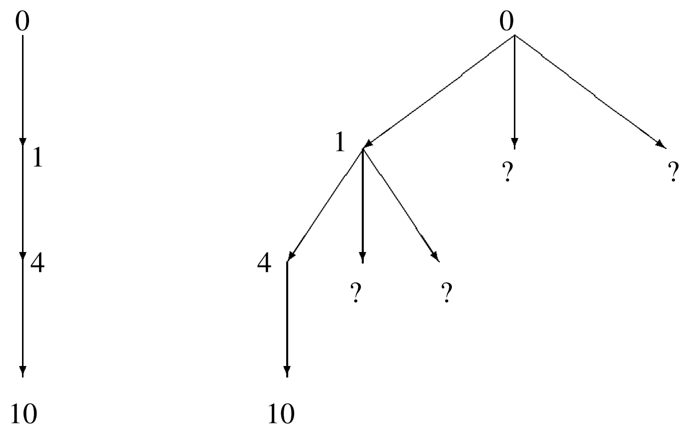


Figure 4.6: Left: subgraph 1, right: subgraph 2.

Compactness: 0.0998

Stratum: 0.0274.

Graph 1: **Number of cross-reference links:** 125;

Percentage cross-reference links: 56;

Compactness: 0.280;

Stratum: 0.0559.

Graph 2: **Number of cross-reference links:** 111;

Percentage cross-reference links: 53;

Compactness: 0.795;

Stratum: 0.0780.

Graph 3: **Number of cross-reference links:** 89;

Percentage cross-reference links: 47;

Compactness: 0.275;

Stratum: 0.0629.

Graph 4: **Number of cross-reference links:** 28;

Percentage cross-reference links: 22;

Compactness: 0.0586;

Stratum: 0.0112.

HHO: **Number of cross-reference links:** 856;

Percentage cross-reference links: 73;

Compactness: 0.591;

Stratum: 0.0558;

E0: **Number of cross-reference links:** 169;
Percentage cross-reference links: 75;
Compactness: 0.811;
Stratum: 0.0863.

E1: **Number of cross-reference links:** 0;
Percentage cross-reference links: 0;
Compactness: 0.0435;
Stratum: 0.00912.

E2: **Number of cross-reference links:** 225;
Percentage cross-reference links: 74;
Compactness: 0.600;
Stratum: 0.0830.

Chapter 5

Speed considerations

In this chapter we examine the speed of the strategies described in section 2.1. Furthermore we investigate the influence the different facilities have on the speed of these strategies. The speed of a strategy may be important when looking at the suitability of a strategy in offering the user a good insight in the structure of the graph he is browsing through. When applying a fast strategy, the user is able to follow/discover more links and thus examine a larger part of the graph. Discovering a large part of a graph *may* lead to a better understanding of its structure. (Later we will see that this isn't always the case).

The speed of a strategy is measured by examining the total weight of the steps needed to browse the graph completely.

Separate conclusions will be drawn for each of the three user/hypertext groups.

5.1 Group 0 speed-conclusions

Since the weights assigned to the different steps[1] all equal one, the total weight needed to browse a graph equals the number of steps needed to search a graph completely. Hence in this section, steps and weight are used interchangeably.

5.1.1 Only backtracking allowed

We first examine the speed of the strategies in combination with a minimal set of facilities, which means that only the backtracking facility is offered. (Remember that this facility is considered indispensable. See section 2.2.)

Looking at the data obtained by experimenting with all graphs, offering our simulated user just the backtracking facility, we can draw the following conclusions:

- Depth-first search is the fastest strategy.

[1] See definition 2 in section 3.2.

Remark 6 *While examining the number of steps needed by the depth-first search, we observe that it needs 750 steps to search graph 2 completely. This number of steps is conspicuous since the number of steps needed by the depth-first search strategy to search a graph completely, has a maximum of* –two times the number of links– *in the graph. Hence the maximum number of steps should be 422.*

We can explain the loss of speed by the depth-first search by the fact that during our experiments, strategies are combined with the properties of the user. Property 3 in section 2.4 states that a user, navigating to the node that initiates the (new) link he wants to follow next, follows the last-*followed path to the current node backwards, until a node is reached that is situated on the last followed path from the root to the goal node.*

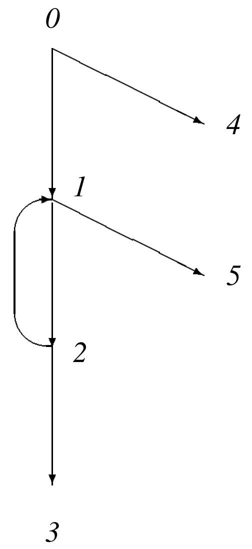


Figure 5.1: The structure of some graph

If the graph shown in figure 5.1 is browsed, using the depth-first search and taking the properties of the simulated user into account, then the nodes are accessed in the following order:

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 1 \stackrel{back}{\rightarrow} 2 \rightarrow 3 \stackrel{back}{\rightarrow} 2 \stackrel{back}{\rightarrow} 1 \rightarrow 5 \stackrel{back}{\rightarrow} 1 \stackrel{property\ 3,\ section\ 2.4}{\longrightarrow} 2 \rightarrow 1 \stackrel{back}{\rightarrow} 0 \rightarrow 4$$

Hence 13 steps are performed whereas the number of links in the graph equals 6.

The fact that depth-first search may need more than –two times the number of links– *steps to search a graph, explains depth-first search sometimes not being the fastest strategy.*

- Random 0 is the slowest strategy to discover a graph completely.
- Breadth-first search is faster than the random search 0 strategy and slower than depth-first search strategy.
- The speed of strategy 1, strategy 2 and random 1 heavily depends on the graph browsed. If the graph contains less than about 25% cross-reference links, then the strategies need about the same number of (or even more) steps as in case breadth-first search is put into practice. If the graph contains more than about 40% cross-reference links (HHO, graph 1, graph 2, graph 3,

erica 0 and erica 2), then the number of steps needed by strategy 1, 2 and random 1 is often smaller than the number of steps performed by the breadth-first search.

Furthermore, when varying the constants involved in strategy 1 and 2 search, we observe (as expected) that the larger the *Strategy 2 constant*, the faster it becomes.

- Applying breadth-first search or Random 0 search in combination with Strategy 3 speeds up the complete search of a graph. The value for the optimal *level-constant* heavily depends on the graph that is searched. A wrong value of the *level-constant* may decrease the speed, so that it can be compared to using the second strategy without strategy 3. See the results on graph 2 with *level-constant* equal to two.

  A strategy that is combined with strategy 3 with *level-constant* equal to two or three, gaines most if graphs are searched that contain a small number of (cross-reference) links. This comes as no surprise since a small graph in combination with strategy 3 often yields performing plain depth-first search, since the number of levels in a small graph is usually also small.

Remark 7 *Strategy 3 in combination with depth-first search yields (as expected) to the same results as just applying depth-first search.*

The conclusions drawn come as no surprise. If the depth-first strategy is put into practice, the next link to follow is always the nearest link not followed yet. So the number of backsteps is restricted to a minimum[2]. In fact, most of the time, the next link to follow initiates from the current node. Backsteps are only needed in case a leaf is reached, or in case the current node is visited before.

Random 0 search behaves unstructured and therefore can hardly be called a strategy. The "unstructuredness" of this strategy implies a large number of steps that represent backsteps and following links for the second or more time. As a result it performs very poorly regarding speed.

The other strategies are all intermediate forms of the depth-first and breadth-first search. Hence, depending on the values of the constants, the speed these strategies produce lie somewhere in between the speed of the depth-first strategy and the speed of the breadth-first strategy.

5.1.2 Combining strategies and facilities

In this section we examine the influence facilities have on the speed of strategies. It may well be the case that a strategy, like breadth-first search, gains that much by a certain facility that looking at the speed of the strategy, it becomes more interesting.

Conclusions concerning the different search strategies:

- If we examine the speed of the random search 0 in combination with the different facilities, then the following order is obtained (from slow to fast discovery of the complete graph):

  1. Backtracking.
  2. Backtracking, Highlighting links.
  3. Backtracking, History list 1.
  4. Backtracking, History list 1, Highlighting links.

[2] With some exceptions, as shown in the example on the graph with its structure shown in figure 5.1.

5. Backtracking, Bookmarks.
6. Backtracking, Bookmarks, Highlighting links.
7. Backtracking, History list 0.
8. Backtracking, History list 0, Highlighting links.

Property 1 *Dynamic facilities allow a user to directly jump to the node that initiates the next link to follow. So navigating over the path leading from the current node to the node that initiates the next link to follow can be skipped. Hence the speed gained by a dynamic facility is a result of skipping these paths.*

Property 2 *If the next link to follow leads to a node visited before, then the highlighting link mechanism allows a user to skip the step representing following a link for the first time and the step representing a backstep, Hence the speed gained by this facility consists of skipping these steps for every cross-reference link encountered in the graph.*

We conclude that usage of the dynamic browser facilities most accelerates the complete discovery of the graph. This can be explained by the combination of: the fact that performing random search 0 yields the discovery of very few cross-reference links (see chapter 6) in combination with property 2, and the unstructuredness of this strategy in combination with property 1.

- Examining the speed of the breadth-first search in combination with the different facilities, yields the following order (from slow to fast discovery of the complete graph):

  1. Backtracking.
  2. Backtracking, History list 1.
  3. Backtracking, Bookmarks.
  4. Backtracking, History list 0.
  5. Backtracking, Highlighting links.
  6. Backtracking, History list 1, Highlighting links.
  7. Backtracking, Bookmarks, Highlighting links.
  8. Backtracking, History list 0, Highlighting links.

  We observe that, in contrast to using random search 0, most speed is gained by using the highlighting link facility. This can be explained by the fact that in contrast to using the random search, the overhead of backsteps performed is not that big. As a result, not as much speed (but still a lot) is gained by using one of the dynamic facilities. However, usage of the highlighting link facility decreases the number of steps needed at least with the number of cross-reference links (property 2).

  If the graph searched contains very few cross-reference link then the reverse counts and the results are the same as in the random search 0 case.

Remark 8 *The history list 1 facility doesn't add any speed in case breadth-first search is performed. This is a consequence of the fact that our simulated users have a property of only jumping to a node if it initiates the next link to follow. See property 5 at page 16.*

If breadth-first search is applied then:

- *the node that initiates the next link to follow is not contained in the history list 1, or*
- *the node is contained in the history list 1 but can also be reached by a single backstep.*

Hence nothing is gained by using the history 1 facility in combination with breadth-first search.

- If we examine the speed of the depth-first search in combination with the different facilities, then the following order is obtained (from slow to fast discovery of the complete graph):

  1. Backtracking.
  2. Backtracking, Bookmarks.
  3. Backtracking, History list 0.
  4. Backtracking, History list 1.
  5. Backtracking, Highlighting links.
  6. Backtracking, Bookmarks, Highlighting links.
  7. Backtracking, History list 0, Highlighting links.
  8. Backtracking, History list 1, Highlighting links.

  This behaviour can be explained by the fact that while using depth-first search, both the number of backsteps and number of steps that represent following a link for the second or more time are very small. Hence due to property 1, little is gained by a dynamic facility.

- The results obtained from the experiments conducted using the strategy 1, 2 and random 3 search, make us draw the same conclusions as we did when the results concerning the depth-first search were examined. If however the constants are chosen in such a way that the behaviour of the strategy 1 and 2 searches resembles breadth-first search, then the results and the conclusions we draw from them will too.

Remark 9 *Using depth-first search, there is no difference in speed between using the history list 0 and history list 1 facility. In contrast to the breadth-first search strategy, the nodes the user jumps to, using a history list facility are all situated on the last followed direct path from the root to the current node. These nodes are just the ones that are contained in the history list 1. Hence when applying the depth-first search, it doesn't matter which of the two history lists is offered.*

Remark 10 *We have to keep in mind that if the bookmark facility is offered, the speed of the strategy heavily relies on the way the bookmarks are placed. A 'fast' strategy in combination with poorly placed bookmarks is possibly slower than a 'slow' strategy in combination with well placed bookmarks.*

Since our algorithm to simulate a user randomly chooses nodes to place bookmarks on, bookmarks will *sometimes be poorly placed. Take for example bookmarks placed on leaves. The simulated user never visits a leaf twice. Hence those bookmarks are useless.*

5.2 Group 1 speed conclusions

5.2.1 Only backtracking allowed

As expected, there is a major change in proportions between the different strategies, with respect to the proportions obtained using group 0 settings. The relative differences between strategies, in number of steps needed, became a lot smaller. This is caused by the small weight of the backstep and the step representing following a link for the second or more time, compared to the weight assigned to following a link for the first time. The differences in speed between the strategies, are a result of the number of backsteps that have to be performed and the number of times a link has to be followed more than once, in proportion to the number of links that have to be followed only once. Hence these steps (weighted 1) do not influence the total weight as much. Which implies a major decrease in the differences regarding speed.

However, we still obtained the same ordering of strategies regarding speed:

- Except for situations explained by figure 5.1, applying depth-first search yields the fastest discovery of the complete graph.
- Random 0 search is the slowest strategy to search a graph completely.
- Breadth-first search is faster than the random 0 search strategy and most of the times slower than depth-first search, strategy 1, strategy 2 and random 1 search.
- And again, since the strategy 1, 2 and the random 1 search are all some combination of breadth-first and depth-first search, the number of steps needed to browse a graph completely is larger than in case depth-first search was applied and smaller (or larger if the graph searched contains very few cross-reference links) than in case breadth-first search is put into practice.
- If we combine strategy 3 with one of the other strategies we get results that make us draw the same conclusions as for the depth-first search. In fact, the larger the level-constant, the more the results resemble those of the depth-first search.

  If the level-constant is chosen to be small, then the behaviour of the combination of Strategy 3 and some other strategy will strongly resemble the behaviour of the second strategy. As a result the same conclusions are drawn for this combination, as the ones we drew after examining the results of the second strategy. Again as in the case of group 0 users/hypertexts, graphs with a relative small number of links gain the most by combining a strategy with strategy 3.

5.2.2 Combining strategies and facilities

We can draw the following conclusions concerning the browser-facilities:

Breadth-first search: The order is the same as that obtained by conducting the group 0 experiments:

Still, the biggest decrease in number of steps needed to search a graph completely, is caused by using the static facilities. There exists a major difference in using the history 0 and history list 1 facility. In fact the history list 1 doesn't add any speed (as explained before) whereas the history list 0 facility does.

Depth-first search: The order is still the same as the order obtained from conducting the experiments on group 0 users/hypertexts. The static facilities still speedup the search more than the dynamic ones. Compared to the group 0 approach the difference is even bigger, because of the fact that the steps representing following a link for the second or more time or representing a backstep don't affect the total weight as much as they do due to group 0 settings.

If however the number of cross-reference links in the graph is very small, then the differences between group 0 and group 1 results become very small and all facilities add about the same speed to the strategies.

Random 0 search: Although we retrieved the same order in combinations of facilities with respect to the speed of browsing a graph completely, there is a difference compared to the results we got by experimenting with group 0 settings. The dynamic browser facilities (especially the history list 0) still add a lot of speed, but not as much as they did using the group 0 approach. This can be totally attributed to the fact that weighting the steps the way we did leads to less influence of the enormous overhead of backsteps and steps representing following a link more than once, on the total weight.

We can draw the overall conclusion, that assigning weights that represent group 1 users/hypertexts to the different steps, does not alter the order of combinations, but does decrease the differences regarding speed between the combinations, compared to results obtained with group 0 settings. This can totally be attributed to the fact that the influence of the number of backsteps and links followed more than once, on the total amount of moves the user performs, diminishes a lot.

5.3 Group 2 speed conclusions

5.3.1 Only backtracking allowed

Next, we examine the speed results concerning group 2. Compared to group 1 settings, only the step representing following a link for the second or more time is weighted differently. Now a weight of 6 is assigned instead of a weight of 1.

With respect to the results we got from experimenting with group 1 settings, the breadth-first and random 0 strategies used in combination with only the backtracking mechanism, suffer most from the different settings, due to the fact that breadth-first search and random 0 search are the only strategies that force the user to often follow a link more than once.

If strategy 1 and 2 are performed in a manner that they behave close to breadth-first search, then the change in step weight has the same effect as it did on the breadth-first search.

Hence the conclusions drawn are the same as the ones drawn from experimenting with group 1 settings. Only the size of the gaps between the strategies differ.

5.3.2 Combining strategies and facilities

If together with the backtracking facility a dynamic facility (except for the history list 1) is available, then the steps that make the difference between group 1 and group 2 users/hypertexts (following a link more than once) can be skipped by jumping to earlier placed bookmarks or jumping to nodes in the history list 0. Hence, if these facilities are available, the user doesn't notice the change in weight of the step that represents following a link he followed before.

The ordering of the strategies in combination with the facilities is the same ordering as the one established after conducting the group 1 experiments. The same conclusions are drawn. Only difference is, because of the larger weight assigned to a step that represents following a link for the second or more time, the history list 0 and bookmark facility having a bigger influence on the speed.

Chapter 6

Discovering the structure of a graph

In this chapter we examine what strategies are most suitable for the user in order to obtain a proper perception of the structure of a graph. We do this by examining the results of the experiments described in section 4.2. In doing so, we especially look at the values of the results regarding cross-reference links, compactness and stratum. The basic idea is that from the point of view of the user, a hypertext appears to have a hierarchical structure that is disturbed by cross-reference links. Whenever the structure of a hypertext is strictly hierarchical, the user is unlikely to get lost. Hence, special attention will be paid to the cross-reference links metrics and the compactness metric.

Separate conclusions are drawn for all three user/hypertext groups[1]. Furthermore, we distinguish between browsing a graph using a small number of steps (40 and 320 respectively, depending on the weights assigned to the steps, see definition 2) and using a larger number of steps (100 and 800 respectively).

6.1 Conclusions concerning group 0

6.1.1 A maximum allowed weight of 40

From the results we got from the simulated user browsing through a graph making a small number of steps, we draw the following conclusions:

Breadth-first, depth-first search: In case cross-reference links are rare ($< 20\%$) in the graph we experiment on (graph 0, 4 and E1), and if the breadth-first strategy is used, the user discovers a lot more cross-reference links compared to using the depth-first search. The depth-first search has the tendency to find very few cross-reference links if a small number of steps is taken and the graph that is searched contains a low ($< 20\%$) percentage of cross-reference links.

In case there are a lot of cross-reference links contained in the graph (graph 1, 2, HHO, E0 and 2) the reverse counts. When breadth-first search is put into practice, experimenting on the HHO structure even leads to not discovering any cross-reference links! This can be explained by the fact that the root has 25 outgoing links. Hence all of the allowed steps are used to follow links that initiate from the root.

[1] For the definition of these groups, see chapter 3.

The reason for not discovering many cross-reference links while applying depth-first search on a graph that contains a small percentage of cross-reference links, is the fact that if the user puts this strategy into practice, he then navigates in a way that he will follow a linear path as long as possible. The circumstances that force the user to leave this trail, are the arrival in a leaf or the arrival in a node he visited before. Hence, the consistent following of linear paths and the limited number of steps, yields the discovery of very few cross-reference links. If however the percentage of cross-reference links is high (> 50), then depth-first search will discover more cross-reference links because of its speed.

In case only a small number of steps is taken, the breadth-first search doesn't gain much by using the history list 0 or bookmark facility. Because of the very small number of discovered nodes (about 15 if only 40 steps are allowed, remember that we are speaking of group 0 users/hypertexts), the increase in number of nodes discovered (about 5) when using one of the facilities is very small. As a result the increase of discovered cross-reference links will also be very small.

The same for the highlighting link facility, albeit for a different reason. Because using the breadth-first search for only 40 steps yields to visiting very few nodes, the number of cross-reference links found will be that small that using the highlighting link facility will lead to the discovery of only slightly more nodes. Hence, the structure found does not differ significantly from the structure discovered not using this facility.

Adding facilities in case depth-first search is put into practice, also leads to not discovering a higher percentage or larger number of cross-reference links. This can be explained by the fact that it is very efficient by itself (concerning speed)[2] regarding the settings belonging to group 0 users/hypertexts. Hence, very little is gained by using the facilities described.

Random 0 search: Using random search 0 while making a small number of steps results in the discovery of a number of cross-reference links larger than found while using the depth-first search, and smaller than found while using breadth-first search. In case the experiments were conducted on graphs with high percentages of cross-reference links *and* more than 100 nodes (HHO, graph 1, 2 and 3), the reverse counts.

When allowing a small number of steps, the use of facilities does increase the number and percentage of cross-reference links found by using random 0 search. This can be explained by the fact that using random 0 search, with only the backtracking facility available, is that inefficient that using history lists, the bookmark facility or the highlighting link facility, *does* increase the number of nodes discovered considerably.

Strategy 1, 2 and random 1: Strategy 1 and 2 combine the results of the breadth and depth-first search. Which of the two behaviours they resemble most, depends on the values of the constants.

Strategy 3: Using strategy 3 in combination with another strategy yields results that are close to those of the depth-first search.

Remark 11 *We have to remember that the highlighting link facility only provides extra information to the user if cross-reference links are encountered*[3]*. Hence, if only a few cross-reference links are discovered, very little is gained by using this facility.*

[2] See the group 0 speed conclusions.
[3] See chapter 2.2 for the description of the highlighting link facility.

We conclude that in case a small number of steps is done, to find a number and percentage of cross-reference links that comes closest to the actual number and percentage of cross-reference links in the graph, a strategy 1 or 2 search is the best strategy to put into practice. If however the user is told that the hypertext he is about to browse is big (more than 200 nodes) or contains a high percentage of cross-reference links, he then better applies the depth-first strategy or a strategy 1 or 2 search close to it.

With the settings simulating group 0 users/hypertexts and only performing 40 steps it is merely impossible to get a good idea, in terms of the metrics we examined, of the structure of a graph that contains more than 20% cross-reference links. The found percentage of cross-reference links doesn't even come close to the actual percentage of cross-reference links contained in the graph. From the other two metrics, especially the stratum values differ enormously from the values of the complete structure of the graph. This can be explained by the fact that because of the small number of links followed, concerning the subgraph searched, it is much clearer from what root this subgraph can best be searched. Hence, as a consequence, we get a much higher value of the stratum metric for this subgraph than for the complete graph.

6.1.2 A maximum allowed weight of 100

The results we got from the experiments conducted with group 0 settings show us that:

- If the percentage of cross-reference links in the graph is high, the depth-first strategy makes the user find a lot more cross-reference links compared to the number of cross-reference links found while using the breadth-first search. Using random search 0 leads to the discovery of even fewer cross-reference links than in case breadth-first search is applied.
- In case there are only a small number of cross-reference links contained in the graph, still depth-first search leads to the discovery of the most cross-reference links. The difference with the other strategies is however smaller.
- The strategy 1 and 2 strategies tend to find percentages close to the number and percentages found by the depth-first search strategy. With the right constant settings, they do slightly better.
- Only random search 0 and breadth-first search are significantly improved by using them in combination with strategy 3. Which comes as no surprise, since the results using these strategies without strategy 3 were very poor. And applying strategy 3 in combination with another strategy means that the graph is partly searched using depth-first search.

Remark 12 *If the strategies are put into practice in combination with the highlighting link facility and/or a history list facility, the differences between the strategies, with respect to the number and percentage of cross-reference links, become smaller. Looking at the results on graph 3 the breadth-first search even discovers a higher percentage and larger number of cross-reference links than in case depth-first search is used. This can be explained by the fact that the cross-reference links in graph 3 are situated at the levels close to the root.*

We draw the conclusion that if about a hundred steps are planned, the depth-first search or a strategy 1 or 2 search close to it, is the best strategy to use. But again, the values of the metrics don't come near the original ones of the complete graph.

It is clear that the advantage of using the depth-first search in case the number of steps allowed to make is not too small, totally relies on the fact that using the depth-first search, enables the user to follow a lot more links than in case another strategy is used. This can solely be attributed to the type of user/hypertexts we simulated in this section.

6.2 Conclusions concerning group 1

This section shows the results of the experiments conducted with group 1 users/hypertexts. Because the settings used, yield a bigger similarity to 'normal' user behaviour, in comparison with the behaviour of group 0 users/hypertexts, we take the compactness and stratum metric more often into account (especially the compactness metric.)

Remarkably, the results retrieved with group 1 settings show us that with these settings the user more often gets a better insight in the structure of the graph than in case group 0 settings are applied. Hence, behaviour close to "normal" behaviour yields the best results!

6.2.1 A maximum allowed weight of 320

In this section we examine the results obtained while experimenting with a maximum allowed weight of 320 by our simulated user. If we look at these results we see that they differ a lot from graph to graph. Especially the results we got from the experiments conducted on the HHO structure differ from the results we got from graph 0 through 4 and E0 through 2. Main reason for this divergent behaviour is the number of nodes it contains. Hypertexts containing more than about 200 nodes, typically have roots and nodes at the first level with very large outdegrees. The root of the HHO hypertext has for example an outdegree of 25. Hence, if a strategy is applied, on such a graph that searches the structure in a breadth-first search-like manner, a lot of time (weight) is spent by just following the links from the root. To be exact, with the settings belonging to group 1 users/hypertexts, breadth-first search uses a weight of 225 just to follow all outgoing links from the root. Hence breadth-first like strategies have problems penetrating into the document. Only the "surface is searched".

The HHO graph

We will now examine the results concerning searching the HHO graph, using a maximum weight of 320. Looking at these results, we draw the following conclusions:

Breadth-first search: Of all strategies, breadth-first search leads to the discovery of the smallest number and percentage of cross-reference links. The values of the compactness and stratum metric are also very poor. They don't come near the metric values of the complete HHO-structure. Furthermore, it doesn't matter whether dynamic facilities are offered by the browser. The results stay the same. This also is a consequence of the large out-degree of the root. At most one backstep is needed to get to the next link to follow, so possible jumps to a bookmark or to a node contained in one of the history lists, do not decrease the total weight used. Hence it doesn't matter whether just backtracking is offered, or one of the more advanced facilities.

The highlighting link facility doesn't significantly increase the number and percentage of cross-reference links found too, since the extra number of links that can be followed due to using the highlighting link facility, depends on the number of cross-reference links found, if this facility were not available. This number is very small, so very little is gained by using the highlighting link facility. The percentage and number of cross-reference links found, using the highlighting link facility, are equal to the percentages found without putting this facility into practice. Only difference is that a few more links are discovered, and so the compactness value slightly drops.

Depth-first search: Compared to breadth-first search, depth-first search does a much better job discovering high percentages and numbers of cross-reference links. Putting this strategy into practice leads to the discovery of 10 or 11 cross-reference links and a percentage of 28. Unlike putting breadth-first search into practice, using different facilities *does* lead to different numbers (although the differences are minimal) of cross-reference links found (10 and 28% when using backtracking against 11 an 28% when using one of the other dynamic facilities together with the backtracking mechanism) and different values of the stratum and compactness metrics. Still, the number and percentage found don't come near the number and percentage belonging to the complete structure. Because depth-first search has the property of following as many links as possible before using the backstep mechanism, the enormous number of cross-reference links in the graph causes the user to follow many links before the first backstep is performed. So it takes a lot of steps before discovering the first cross-reference link. The stratum and compactness values are however very close to the ones of the complete HHO-graph.

Remark 13 *If no backsteps are performed, only those cross-reference links are discovered that are part of a cycle. The remaining cross-reference links are not.*

Because of the fact that the depth-first search, compared to breadth-first search, leads to better results concerning the discovery of cross-reference links, more is gained by offering the user the highlighting link facility. If this facility is offered, then 20 instead of 10 cross-reference links are discovered, which equals a percentage of 37. This is quite an improvement on browsing without the highlighting link facility.

Random 0 search: Applying the random 0 strategy while taking a maximum of 320 steps leads to percentages and numbers of cross-reference links that are close to those of the breadth-first search. It does slightly better because it isn't as inefficient as the breadth-first search because not necessarily all outgoing links from the root are followed.

Although a step representing a backstep has a small weight compared to the step representing the following of a link for the first time, still the overhead of backsteps compared to the other strategies is considerable. This explains the bad results on discovering cross-reference links when random 0 is put in practice. The number and percentage of cross-reference links found using the different static and dynamic facilities, run parallel with the results we obtained by looking at the speed of the different combinations of static and dynamic facilities, using random 0 search.

Strategy 1, 2 and random 1 search: The strategy 1 and 2 search, (with the right constant settings), yield about the same results as the depth-first search. Applying one of these strategies even leads to the discovery of a larger number of cross-reference links. Strategy 1 with its constant equal to two gives the best results regarding the number and percentage of cross-reference links found. Examining the stratum and compactness values we draw the conclusion that they are not as good as the ones obtained by applying the depth-first search.

The differences in the results we get by applying different dynamic facilities correspond to the results we obtained by looking at the speed properties of these facilities while putting depth-first search into practice. However, when using them in combination with the highlighting link facility, it actually leads to different percentages and numbers, whereas using them without this facility only leads to minor differences in the stratum and compactness values.

Strategy 3: It depends on the *level-constant* whether using strategy 3 in combination with another strategy yields results that are close to that of the depth-first search, or results close to those of the second strategy. The bigger the constant, the more the results resemble those of the depth-first search.

Hence we conclude that the best strategies to find a number and percentage of cross-reference links that come close to the actual number and percentage of the HHO graph, are the strategy 1, 2 and depth-first search strategies. The best results are obtained by using the random 1 search with strategy1-constant equal two. What dynamic facility is offered doesn't matter that much. Only slight differences in number, percentage, stratum and compactness occur while experimenting with the different dynamic facilities.

The best results are obtained if one of these strategies is applied in combination with the highlighting link facility.

If we look at the values of the compactness metric, we draw the conclusion that if only a total weight, of the steps performed, of 320 is allowed, then the depth-first search yields values very close to the compactness value of the complete HHO graph, if no highlighting link facility is available. If, however this facility is offered to the user, then regarding the compactness values, strategy 1 with its constant set to two is best put into practice.

As a result we conclude that if a hypertext contains more than about 200 nodes and its structure contains a high percentage of cross-reference links, the user is best advised to perform depth-first or strategy 1 search, to get the best insight in the structure of the hypertext.

The rest of the graphs, allowing 320 steps

If we take a look at the results obtained from experimenting on these graphs allowing our simulated user a maximum of 320 steps, we draw the following conclusions:

Breadth-first, depth-first search: As in the results we got from experimenting on the HHO graph, of all strategies, breadth-first search leads to the discovery of the smallest number and percentage of cross-reference links. Again this can be explained by the effort needed to follow the links from the root. Since the outdegrees, of the nodes contained at the high levels of the randomly generated graphs, are not as big as they are in the HHO graph, the difference in number of cross-reference links found between applying breadth-first and depth-first search is a lot smaller.

The fact that, by using breadth-first search as well as using depth-first search, percentages of about 25 cross-reference links are discovered (except for graph 0 and 4), makes the highlighting link facility very useful.

The small number of steps allowed and the small weight of the backstep are cause for the dynamic facilities not to contribute much to the number of links discovered and the percentage and number of cross-reference links found.

Random 0 search: Putting the random 0 strategy into practice yields to results that are just about the average of the results obtained by applying the depth-first and breadth-first search.

strategy 1, 2 and Random 1 search: As in the HHO experiments, the best strategies to find numbers and percentages of cross-reference links that come close to the actual numbers and percentages of the graphs, are strategy 1, 2 and random 1. The best results are obtained by using the strategy 1 search and strategy 2 search with strategy1-constant larger than 2. It depends of course, on the graph the experiments are conducted on. But overall we can conclude that the strategy 1-constant needs to be larger than in case the experiments are conducted on the HHO graph.

Strategy 3: It depends on the *level-constant* whether using strategy 3 in combination with another strategy yields results that are close to that of the depth-first search, or results close to those of the second strategy.

Looking at the values of the other metrics we conclude that as in the experiments concerning HHO, the depth-first search and strategy 1 give the best results concerning the compactness value.

6.2.2 A maximum allowed weight of 800

The HHO graph

The results obtained from experimenting with a maximum of 800 steps are about the same as the results we get from experimenting with a maximum number of 320 steps. The major differences are:

- The number of cross-reference links found about tripled and the percentage of cross-reference links found are about $1\frac{1}{2}$ times larger. The exception is:
  - Breadth-first search: The nodes at level 1 also have (like the root) large out-degrees, so a lot of steps are needed to follow the adjacent links. Furthermore, the number of cross-reference links at the higher levels is quite small, so a user applying breadth-first search doesn't reach to the places (the lower levels) where most cross-reference links are situated.
- Because of the larger number of cross-reference links found, the highlighting link facility becomes more important.

The rest of the graphs

If a weight of 800, instead of a weight of 320, is allowed while browsing through graph 0, 1, 2, 3, 4, E0, E1 or E2, most is gained by breadth-first search. The main reason is the fact that in these randomly generated graphs, the root has a large outdegree but the nodes at level 1 have out-degrees that are smaller on the average. This implies that using the breadth-first search, our simulated user reaches to lower levels and as a result has a better chance at discovering cross-reference links.

Still, the compactness values for the subgraphs searched, using depth-first search or strategy 1 or 2, are the only ones that come very close to the values of the complete graphs. So a user is best advised to use one of the three strategies mentioned above, if he is going to browse through a hypertext that is contains less than 200 nodes, with the intention of just browsing around (a maximum weight of 800.)

6.3 Conclusions concerning group 2

6.3.1 A maximum allowed weight of 320

If we compare these results with the results we obtained from the experiments conducted with the group 1 settings, we draw the conclusion that the differences between the results are negligible. Looking at the conclusions we have drawn concerning the speed of the different strategies applied on the group 2 hypertexts, this comes as no surprise.

Even the breadth-first search and random 0 search in combination with just the backtracking mechanism lead to the same results as the ones obtained in the group 1 experiments. This can be explained by the fact that when only a few (320) steps are allowed, then the number of times a link is followed more than once, is *very* small. Since these steps are the kinds that make the difference between the settings of the group 1 and group 2, the results are almost equal.

6.3.2 A maximum allowed weight of 800

The HHO graph

If a maximum weight of 800 is allowed, then the difference in the weighting of the step representing following a link followed before, that was masked by the limited number of steps (320) before, does show. Again, in case of the experiments conducted on the HHO graph they don't show as clearly because of the large out-degree of the root and the nodes at level 1. Whereas in case of applying the depth-first search with only the backtracking facility offered, the number and percentage of cross-reference links found tends to be slightly smaller.

The same counts if the backtracking mechanism is offered in combination with the highlighting link facility. This combination together with the depth-first search, strategy 1 or strategy 2 also shows a difference in metric values compared to the results obtained with the group 1 settings. Fewer links are discovered.

The rest of the graphs

In case graph 0, 1, 2, 3, 4, E0, E1 or E2 is browsed through, the random 0 and breadth-first search do lead to the discovery of less cross-reference links and a lower percentage as when they did in the experiments on category 1 graphs. Main reason is the fact that the out-degrees of the root and the nodes at the higher levels are not as large as those in the HHO graph. This implies that the breadth-first and random 0 search yield following more links and as a consequence more often a link is followed more than once (if the backtracking mechanism is the only dynamic facility available. If another dynamic strategy is offered then following a link more than once can be avoided.)

Again, we draw the conclusion that the best strategies to find cross-reference percentages and metric-values closest to those of the complete graphs are depth-first search and strategy 1, 2 and 3 with settings that make them behave in a way that comes close to that of the depth-first search.

6.4 Overall conclusion concerning the discovery of a structure

If we take a look at the conclusions drawn in this chapter, we can give the following overview:

Small number of steps and *no* highlighting links facility available	
Property of the graph	What strategy is most suitable?
Less than 20% cr-links:	All strategies
About a hundred nodes and about 50% cr-links:	depth-first search, strategy 1, strategy 2 and random 1
More than 200 nodes and more than 70% cr-links:	Just depth-first search, all other poor.

Small number of steps and highlighting links facility available	
Property of the graph	What strategy is most suitable?
Less than 20% cr-links:	All strategies
About a hundred nodes and about 50% cr-links:	depth-first search, strategy 1, strategy 2 and random 1
More than 200 nodes and more than 70% cr-links:	depth-first search, strategy 1 and strategy 2

Large number of steps and *no* highlighting links facility available	
Property of the graph	What strategy is most suitable?
Less than 20% cr-links:	All strategies
About a hundred nodes and about 50% cr-links:	depth-first search, strategy 1, strategy 2 and random 1
More than 200 nodes and more than 70% cr-links:	Just depth-first search, all other poor.

Large number of steps and highlighting links facility available	
Property of the graph	What strategy is most suitable?
Less than 20% cr-links:	All strategies
About a hundred nodes and about 50% cr-links:	depth-first search, strategy 1, strategy 2 and random 1
More than 200 nodes and more than 70% cr-links:	Just depth-first search, all other poor.

Remark 14 *If in a table shown above strategy 1 and strategy 2 are mentioned together with just depth-first search we then mean strategy 1 and strategy 2 with such constants that they closely resemble depth-first search. This implies a small* strategy 1-constant *and a large* strategy 2-constant*.*

However, if these strategies are mentioned together with the random 1 strategy then the constants may vary more.

As we can see for almost all combinations of a graphs and a strategy, the best strategies to perform to get metric values close to the value of the complete graph, are depth-first search and strategy 1 or strategy 2 search. The only exceptions are:

1. Graphs that contain small ($< 20\%$) percentages of cross-reference links.

   This is caused by the fact that, especially the compactness value, heavily depends on the percentage of cross-reference links found. Since this percentage in these graphs, is relatively small, the differences in percentages of cross-reference links found by the strategies are also very small; hence, the different strategies will yield compactness values that hardly deviate.

2. Graphs with about 50% cross-reference links no matter what static facility is available and no matter the number of steps done. Strategy 1 and strategy 2 may deviate more from depth-first search and still yield good results.

The more properties of group 0 settings the user and the hypertext have, the more important it becomes to perform a strategy that is very close to the depth-first search. If these settings closely resemble the settings of group 0, then even in case of graphs containing low percentages of cross-reference links, depth-first search is far more suitable than the other strategies, if the total allowed weight isn't too small.

The same holds for group 2 settings, albeit in a lesser degree.

One important conclusion is that for group 1 and 2 users *depth-first navigation is best*, regardless the size of the document, the percentage of cross-reference links or the presence of navigation facilities. The minor variations of depth-first navigation, strategies 1–3 and random 1, also work well in most cases. This suggests that the depth-first navigation is a "stable" optimum. The structure of the hyperdocument, the number of steps made, and the presence of navigation facilities only have some influence on *how better* the depth-first navigation is compared to all other browsing strategies. The experiments also show that this optimum is more stable with documents with around 50% cross-reference links than with documents with very few cross-reference links.

Another observation from the experiments is that whether a strategy works well or not depends a lot on the structure "near" the starting point. A hyperdocument like HHO has many outgoing links in the nodes near the starting points. As a consequence, the breadth-first navigation spends a lot of time (many steps) near the starting point, and may never make it out into the body of the hyperdocument, which has a different structure from the immediate environment of the starting-point.

Chapter 7

Node distribution

In this chapter we examine the behaviour of the different strategies regarding the distribution of the nodes that are accessed during a browsing session. We do this by considering the results from experiments that are described in this chapter. We are especially interested in whether the strategies that yielded the best results in chapter 6 lead to a reasonable node distribution.

7.1 Strategies and node distribution

Until now we have examined what combinations of browser-facilities and strategies are most suitable for the user to get a proper perception of the structure of a graph. However, we didn't examine the way the nodes visited or discovered are distributed over the graph. This is especially important if the graph is browsed by some automatic-search tool instead of a user. If only a limited number of nodes is searched, it is important that these nodes are well distributed over the graph structure. A search facility can operate better if it is able to find a significant part of the structure in the limited time it is allowed to search. It is undesirable for the nodes searched to be all situated in the same region.

One of these automatic-search tools is *fish-search*. *Fish-search* is an information retrieval tool for the World Wide Web (WWW) built on top of Mosaic for X. It operates by retrieving documents from WWW sites, determining the measure of relevance for such a document and extracting the links from it. Since the WWW is very large and constantly growing, only a fraction of the WWW can be searched in a reasonable amount of time. It is clear that the strategy used is extremely important for the quality of the search.

For an automatic-search tool a strategy is preferred that leads to following a lot of links between different sites. Following links to other sites implicates a better distribution of the nodes that are visited during the search session.

Property 3 *Following a lot of links considered as cross-reference links in a complete breadth-first search of the graph implies that the accessed nodes are well distributed.*

Property 3 is a result of the following observation:

Removing the cross-reference links, *obtained by breadth-first search*, from a graph leaves a tree-like structure. The nodes in a subtree of this structure contain information about a common subject. This subject is determined by the subtree's root. Hence it is the cross-reference link that connects the different subjects (according to the tree's structure).

7.2 Measuring the node distribution

To determine what strategies described in section 2.1 yield a good distribution of discovered nodes, we conduct the following experiments:

All graphs described in section 3.1 are browsed using all strategies described in section 2.1. Since the results of these experiments are especially meant for improving the automated search, the weights assigned to the different steps are:

1. Following a link the first time: weight = 1;
2. Following a link followed before: weight = 0;
3. Performing a backstep: weight = 0;

Since a jump to a node in one of the history lists or a jump to a node that was marked as a bookmark are not applicable in case an automated search is performed, the steps representing them are not considered.

The weights assigned to performing a backstep and the following of a link for the second or more time are set to 0. This is a result of the fact that the *fish-search* algorithm uses a queue in which the nodes that still need to be searched, according to a strategy, are contained. Accessing the next node, due to a strategy, is done by deleting it from the front of the queue and performing some search on the contents of this node. Hence, the distance between two successive nodes that are accessed plays no role in the automated search. The only step that takes time is the step that represents the following of a link for the first time (deleting the node from the front of the queue).

Remark 15 *The differences between the strategies are obtained by the place the nodes are put in the queue.*

Because of the fact that an automated search retrieves information much faster than a human reader of a hypertext does, the maximum allowed weight used in the experiments must be chosen (relatively) much higher than was the case in the experiments in which human users were simulated. We choose two maxima:

1. A maximum weight of 75 and
2. a maximum weight of 150.

Remark 16 *The settings used in these experiments imply that the maximum allowed weight equals the number of different links followed.*

To examine whether the nodes accessed, due to a strategy, are well distributed over the graph, we execute the following steps:

1. The complete graph is browsed using the breadth-first search. While searching, the cross-reference links, *according to this strategy*, are marked.

2. The graph is searched with a maximum allowed weight of 75 or 150. This search is performed by using the strategy we want to examine. And the links, that were marked in the first step, that are followed (not discovered, since no facilities are implemented), are counted.

3. The percentage of these "marked" links of the total number of links in the subgraph searched, is calculated.

The percentage of "marked" links of the total number of followed links gives us an indication on the way the accessed nodes are distributed. We hereby use property 3.

Remark 17 *We have to take into account that the results in these experiments will be very poor for the breadth-first search. The number of "marked" links that will be followed by the limited breadth-first search, will equal the number of cross-reference links discovered. We know from the results shown in chapter 6 that this number is small. However, these poor results do* not *mean that the nodes accessed due to breadth-first search are not well distributed over the graph structure. The distribution of the nodes accessed by applying breadth-first search is not a result of following a lot of "marked" links, but a result of following the right "unmarked" links. For the depth-first-like strategies the opposite holds.*

7.3 Results and conclusions

Looking at the results we conclude that the highest percentages of "marked" links are followed by the depth-first search, strategy 1, strategy 2 and strategy 3. These percentages equal or even exceed the percentage of cross-reference links in the graph.

As a result we conclude, due to property 3, that these strategies access nodes that are well distributed over the graph. The allowed total weight only influences the results regarding breadth-first and random 0. They do about 15% better if a total weight of 150 instead of 75 is allowed.

Remarkably the results make us draw the same conclusion whatever second strategy is put into practice:

1. Random;

2. Choose the link with the smallest id;

3. Choose the link that is situated on the left.

Hence whatever "measure of relevance"-mechanism is used by the automatic-search tool, the same conclusions are applicable.

7.4 Results and conclusions on non-automated search

The same experiments as described in the preceding part of this chapter, are conducted with simulated users. The differences are:

- The weights assigned to the different steps.
- The maximum allowed total weight during a browsing session.
- Browsing-facilities are considered.

Equal settings are used as the ones described in chapter 3.

The same conclusions as for the automated-search, with respect to the distribution of visited nodes, can be drawn for all three classes that represent the non-automated search. The browsing facilities have the same influence on the percentage of "marked" links discovered, as they do on the speed of a strategy, described in chapter 5. Hence again we conclude that the strategies that performed best in chapter 6, also yield good node distribution.

Chapter 8

Justifying the experiments

In this chapter we compare the behaviour of *human* users with the behaviour of the users we *simulated* in the experiments. We do this by experimenting with human users and comparing the results with the results we got from experimenting with simulated users. We examine whether the properties we imputed on simulated users are met in reality. In other words: do human users have the properties we stated regarding the users we simulated.

8.1 Recognizing structures

In the experiments described in chapter 3 the simulated user browsed through a graph with some maximum allowed number of steps. The subgraph discovered was stored and the metric values were calculated for it. From these values, conclusions were drawn in chapter 6, on the suitability of a strategy for browsing a certain graph.

If we want to draw the same conclusions for human users as we did for the ones we simulated, then human users should have the following property:

Property 4 *The human user's "impression" of the structure of the subgraph discovered has to be very close to the structure of the subgraph he actually searched. In other words: the structure he thinks he has discovered must be close to the actual structure of the subgraph*[1] *searched.*

As a consequence, the user must be able to remember a node as being visited before and he also has to know its position in the structure. If it can be shown that human users have the ability of recognizing a node as being visited before and that they can recall the position of this node in the graph, then we can state that the conclusions drawn in the earlier chapters also hold for *human* users, which is the goal of this thesis.

In the experiments described in this chapter, we try to find out what picture, of the part of the graph discovered, a user builds in his mind, and whether this picture resembles the subgraph he actually searched. We do this by asking people to browse a non-trivial hypertext. While these participants are browsing the hypertext, their moves are automatically registered. From this data we can abstract the

[1]The subgraph according to item 2 at page 28.

structure of the subgraph searched. Hence by letting the participant draw a picture of the subgraph browsed, and comparing this picture with the actual subgraph this user searched, we hope to be able to draw some conclusions on differences and/or similarities between *human* users and simulated users.

8.2 The experiment-settings

The hypertext that is used in these experiments has the following properties:

1. It was designed using GUIDE and has a structure equal to graph E2[2]. In fact, we obtained the structure we experimented on in section 3.1 by abstracting it from this hypertext.
2. The only dynamic facility offered to the user is the backtracking facility.
3. There is no highlighting links facility available.
4. There is no breadcrumbs facility available.
5. The contents of every node is small enough to fit on the screen at once. Hence, scrolling is not necessary.

   The contents of a node exists of text and pictures.
6. The anchors give some information about the contents of the node that is situated at the end of the link the anchor represents.

   Furthermore, the anchors in this hypertext have the following property:

   All anchors representing links that lead to the same node are identical.

   A hypertext having this property can be looked at as a hypertext providing some kind of highlighting link facility. The experiments, however, showed us that most (almost all) people who cooperated did *not* recognize the anchors, that lead to nodes they visited before, as such. Hence, item 3 is still valid.

Item 3 and 4 imply that there are no facilities offered to help the user, recognize nodes as being visited before.

Item 2 implies that there are no facilities available to the user that may help him remember the position of an earlier visited node in the graph.

Hence, the experiments are conducted with a minimum of facilities.

The participants were asked to browse the hypertext twice. One time using the breadth-first search and the second time putting the depth-first search into practice. In both cases they were asked to stop browsing if they thought to have discovered 25 different nodes.

8.3 Conclusions on structure recognition

In table 8.1 some of the results of the experiments are shown. It contains the results of four participants with average results. The first column contains the values of the metrics concerning the subgraph the

[2]See section 3.1.

user searched. The second column contains the values of the metrics concerning the subgraph the user *thinks* he has searched. The data shown in table 8.1 is obtained from users performing depth-first search as well as breadth-first search.

As we can see, the metric values of the subgraph, the user thinks he has searched, do not divert much from the subgraph this user actually searched. Especially the number and percentage of cross-reference links, the user thinks he has discovered are very close, if not the same, to the number and percentage he has actually seen. This means that *human* users have the ability of (most of the time/all the time) recognizing nodes as being visited before. (Remember that in these experiments *no* facilities were offered to help the user. Hence the results may very well become even better, if the highlighting link or the breadcrumbs facility is offered.)

The fact that, on the average, the values of the stratum and compactness metrics, of the subgraph the user thinks he has browsed, differ less than 10% from the metric values of the subgraph these users actually searched, make us draw the conclusion that these users are also able to depict, with very small deviation, the *position* of the node they visited before. Again if, for example, a history list were available the results would probably have been better since examining this list may give the user some extra information about the position of a node in a graph.

Person	Metric	Subgraph value	Drawing value
1:	Compactness	0.271	0.277
	Stratum	0.407	0.408
	# cr	3	3
	% cr	10	10
2:	Compactness	0.427	0.394
	Stratum	0.774	0.711
	# cr	7	7
	% cr	29	29
3:	Compactness	0.435	0.399
	Stratum	0.586	0.535
	# cr	6	6
	% cr	26	23
4:	Compactness	0.128	0.121
	Stratum	0.083	0.076
	# cr	1	0
	% cr	5	0

Table 8.1: Some results

8.4 Conclusions on time spend regarding different steps

During the experiments described in this chapter, apart from the nodes the user visits, also the time spend at each node is registered. With the time spend at a node, we mean the time spend from the moment the anchor is "clicked" that represents a link that leads to this node (or from selecting the backtracking mechanism), until the moment the contents of the node are erased from the screen due to selecting another anchor (or selecting the backtracking mechanism).

We will differentiate between three situations:

1. The time spend on a node visited for the first time.
2. The time spend on a node visited for the second or more time.
3. The time spend on a node that is reached using the backtrack mechanism.

The times spend on the nodes by the people who cooperated in the experiments vary that much that no reasonable conclusion can be drawn on the average time spent on nodes. We can however conclude that the time spent on a node that was visited for the first time, is *at least* as long as the time spend on a node that is visited for the second or more time and the time spent on a node that is reached after performing a backstep.

The three situations described before resemble of course the definitions of the steps in chapter 3. Because of the diversity of the times for the different steps registered, we are not able to say if the weights of the steps we used in our experiments are optimal regarding *human* user behaviour. We just know that they are close.

Chapter 9

Conclusions

In retrospect, we may state that we can distinguish four major results:

1. The depth-first search strategy is the most suitable navigation strategy for obtaining a proper perception of a graphs structure. The main reason is the depth-first search strategy being very fast. But even if the settings are such that backsteps and the step representing following a link that is followed before, have very small weights compared to the step representing following a new link, the depth-first search strategy is the best one to put into practice.

   Furthermore, we have seen that this strategy also yields the access of nodes that are well distributed over the graph structure.

2. Another important property of the depth-first strategy is that it is a "stable" optimum. Meaning that minor variations of this strategy and variations in the graph structure do not have a big influence on this strategy's supremacy over the other strategies.

   Only graphs containing less than about 20% cross-reference links, level the differences with the other strategies, for a strategy's suitability in letting a user obtain a proper perception of the structure. This comes to no surprise, since it is the cross-reference link that disturbs the user's view of an underlying hierarchical structure. Hence hypertexts containing a small number of cross-reference links are unlikely to be the cause of a user getting "Lost in Hyperspace". As a result it isn't that important what strategy is used to search such hypertext.

3. As a result of the supremacy of depth-first search, the highlighting link mechanism is the best facility to offer a user. As seen in chapters 5 and 6, little or nothing is gained by using dynamic facilities in combination with depth-first search.

   Furthermore, when applying depth-first search, the history list 0 facility is completely superfluous, as the advantages gained by using this facility are also obtained by using the more restricted history list 1 facility.

4. The experiments described in chapter 8 show us that human users have the same properties as those stated for our simulated users. If this wasn't the case, then it would be nearly impossible, for a human user, to obtain a proper perception of a hypertext structure.

Bibliography

[1] R.A. Botafogo and E. Rivlin and B. Shneiderman, Structural Analysis of Hypertexts : Identifying Hierarchies and Useful Metrics; *ACM Trans. on Information Systems, volume 10, pages 142–180*.

[2] E. de Vries; Stretching the initial problem space for design problem solving : Browsing versus searching in network and hierarchy structures; *OCTO-report 93/02, Dept. of Philosophy and Social Sciences, Eindhoven Univ. of Technology*.

[3] E. de Vries, J. van Andel, T. de Jong; Computer-Aided Transfer of Information on Environment and Behavior : Network versus Hierarchical Structures *Proc. of the 23rd Conference of the Environmental Design Research Association*.

[4] B. Shneiderman and G. Kearsley; Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information; *Addison-Wesley* 1989.

[5] J. Nielsen; Hypertext and hypermedia; London : *Academic Press*.