Optimal Demand-Oriented Topology for Hypertext Systems

Scott Aaronson
aaronson@voicenet.com
Department of Mathematics and Computer Science
Clarkson University
Potsdam NY 13699-5815

Abstract

This paper proposes an algorithm to aid in the design of hypertext systems. A numerical index is presented for rating the organizational efficiency of hypertexts based on (1) user demand for pages, (2) the relevance of pages to one another, and (3) the probability that users can navigate along hypertext paths without getting lost. Maximizing this index under constraints on the number of links is proven NP-complete, and a genetic algorithm is used to search for the optimal link topology. An experiment with computer users provides evidence that a numerical measure of hypertext efficiency might have practical value.

1 Introduction

The form of information retrieval called *hypertext* has grown vastly in importance over the past few years. Hypertext, wherein information is organized into pages containing links to one another, is the basis of not only the World Wide Web, but also voice-prompt telephony services; online software documentation; many multimedia applications; and emerging technologies such as network computers and interactive television. A large body of research, since the 1960's, addresses how to organize information in hypertext systems so that users can quickly access the information they want (see [8]). Riner [13] considers automated methods of converting reference materials and other structured documents into hypertext, while Rearick [12] proposes choosing links for hypertext systems by searching for pages with similar word patterns. However, previous research has generally been qualitative, with mathematical activity focused on the converse problem of *finding* information (for instance, [4]). Parunak [11] views hypertexts as directed graphs, and categorizes patterns of links, but does not consider algorithms for choosing links automatically.

In this paper, we consider a graph-theoretic model of hypertext that emphasizes user demand for information. We also propose a numerical index for rating how well-organized a hypertext system is. We show that rearranging links to maximize this index is an NP-complete problem. Since finding the absolute maximum would be too computationally costly, we use a genetic algorithm to approximate the solution. Finally, we describe an experiment in which 70 computer users searched for information using three different versions of the same hypertext. The results show that the mathematical model described here might be useful to hypertext designers, though it does not eliminate the need for link layout by humans.

2 Definitions

For the purposes of this paper, we will define a *hypertext system* by the following: a set of *pages* P, a *degree* θ_p and a *demand* d_p for each $p \in P$, a set of *virtual links* Γ, a set of *real links* R, a set of *mandatory links* M, a *system degree* θ_s, and an *attention-span factor* α.

A page (often called a node in the literature), containing information and links to other pages, is the basic unit of a hypertext system [3, 8].

$\theta_p \in \mathbb{N}$ represents the maximum number of outgoing links page p can have. It is useful for a variety of reasons. Bell Laboratories researchers have found that, when users of a voice-prompt telephony service are presented with too many options at once, they often forget the first few [10]. In addition, "personal communicators" and other hand-held consumer electronics may have display screens that are too small to hold many links at once. Even without constraints imposed by the medium, however, hypertext system designers may restrict the number of outgoing links pages contain; ac-

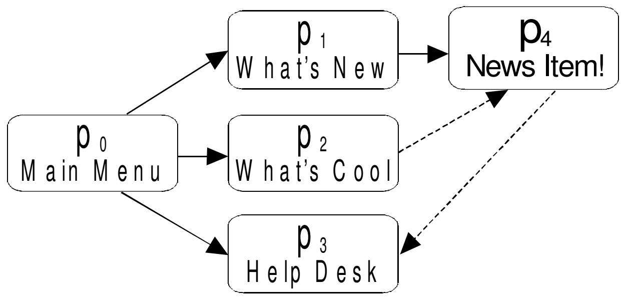


Figure 1: The hypertext system diagrammed here contains four real links and six virtual links.

cording to Benjamin [2], "when visitors are given too many choices at once, they may become overwhelmed, leave, and never come back."

The other page-specific constant, $d_p > 0$, represents how popular or important the hypertext designer judges p to be. For a World Wide Web site using a strictly demand-based model, d_p could be the number of accesses (or "hits") that p receives over a representative period—say, a month. d_p could also be determined through a controlled experiment to gauge user interest in each page p. Ultimately, the choice is an arbitrary one that rests with the hypertext designer.

Our definition of a hypertext system also includes three sets of links: the virtual links Γ, the real links $R \subset \Gamma$, and the mandatory links $M \subset R$. A virtual link is an ordered pair of pages $(p_a, p_b) \in \Gamma$ signifying that, based on the content of p_a and p_b, p_a could conceivably contain a link to p_b. Thus, $G = (P, \Gamma)$ forms a directed graph. The virtual links should be chosen by the hypertext designer, or someone else who has a thorough knowledge of the hypertext's content and purpose. A real link $(p_a, p_b) \in R \subset \Gamma$ signifies that p_a actually links to p_b in the hypertext system. For the problem of hypertext optimization as defined here, real links are the only variables. Figure 1 illustrates the difference between virtual and real links. To provide hypertext designers with more flexibility in adding constraints, we also define a set of *mandatory links* $M \subset R$. If there is a mandatory link (p_a, p_b), p_a *must* link to p_b in any rendition of the hypertext system.

Finally, we define two constants that apply to the entire hypertext system. The first is the system degree,

$$\theta_s \leq \sum_{p \in P} \theta_p$$

representing the total number of real links the hypertext system may have. θ_s is useful for regulating "link density"; when used in conjunction with page degrees, it can prevent the user from being overwhelmed with choices. The second is the attention-span factor $0 < \alpha < 1$, which is chosen arbitrarily by the hypertext designer (for the experiment described here we use $\alpha = 0.7$). α is a constant used in computing an index associated with the hypertext system; it represents the likelihood that users will follow a path through the system without becoming sidetracked. When α is close to 1, systems with paths connecting disparate pages will be favored; when α is close to 0, systems that link all closely related pages will be favored. That α may be chosen arbitrarily, and that it is constant throughout the hypertext system, are simplifying assumptions used in our model.

3 The Organizational Index

Our goal is to rearrange links to maximize hypertext efficiency. Before we can do this, however, we need a numerical measure of hypertext efficiency: an index corresponding to how well-organized, or easy to navigate, a hypertext system is. Here, we present one such index. In Section 5, we present empirical evidence indicating that it is indeed a meaningful measure of hypertext system organization.

Benjamin [2] advises World Wide Web site designers: "Don't make people click through too many successive pages to get from point A to point B... The further your visitors have to travel, the more of them you'll lose along the way." In that spirit, we propose the following working definition: *hypertext organization* is a measure of the ease or rapidity with which users can navigate from any page to any other page in a hypertext system.

How do we measure "ease or rapidity" numerically? We assume that at each intermediary page along a path from page p_a to page p_b, there is a certain probability that the user will become sidetracked, get lost, or give up in his or her quest; furthermore, we assume that this probability is equal to $1 - \alpha$, the negation of the attention-span factor. Thus, we say the *strength* of a path containing ℓ real links is $\alpha^{\ell-1}$.

Clearly, there can be more than one path of real links from p_a to p_b in the hypertext system. However, since considering *all* paths would be computationally impractical, we simplify matters by considering only the length of the *shortest* path from p_a to p_b, which we call r_{p_a,p_b}. If there is no path from p_a to p_b, then we say that $r_{p_a,p_b} = \infty$.

In order to compute a realistic aggregate of the strengths of *all* shortest paths, we need to weight each path by the likelihood that a user would actually want to traverse it. We assume the likelihood of users following a path from p_a to p_b is proportional to d_{p_a}, d_{p_b}, and the strength of the shortest path from p_a to p_b in *virtual* links (which is an estimate of how germane the content of p_b is to p_a). Let γ_{p_a,p_b} represent the length of the shortest virtual path from p_a to p_b; note that, since $R \subset \Gamma$, $\gamma_{p_a,p_b} \leq r_{p_a,p_b}$. Thus, the *weight factor* for a path from p_a to p_b is

$d_{p_a} d_{p_b} \alpha^{\gamma_{p_a,p_b}-1}$.

Multiplying the path strength by the weight factor, we obtain a *weighted path strength* of

$d_{p_a} d_{p_b} \alpha^{r_{p_a,p_b}+\gamma_{p_a,p_b}-2}$

Then, summing the weighted path strengths for all $(p_a \in P, p_b \in P)$ such that $p_a \neq p_b$, we find that, if we call the organizational index Ψ, then

$$\Psi = \sum_{\substack{p_a,p_b \in P \\ p_a \neq p_b}} \left(d_{p_a} d_{p_b} \alpha^{r_{p_a,p_b}+\gamma_{p_a,p_b}-2} \right)$$

(Note that if $r_{p_a,p_b} = \infty$, then $\alpha^{r_{p_a,p_b}+\gamma_{p_a,p_b}-2} = 0$.)

Arranging links in a hypertext system to maximize this index is likely to ensure that:

- The pages with the highest demand are easily reachable from one another.
- Closely related pages are easily reachable from one another.
- All of the links on a given page are relevant to that page.

However, one drawback of the index presented here is that it does *not* ensure that every page is reachable from every other. Hypertext designers who want to ensure that even pages with low demand are reachable have several options:

- Increase the attention-span factor α.
- Create a tree of mandatory links, originating at a central home page and connecting every page in the hypertext.
- After links have been arranged so that Ψ is maximized, add or rearrange links manually (or with a separate algorithm) so that every page is reachable.

4 Genetic Algorithm Approximation

Our problem is to find a set of real links that maximizes Ψ under the given constraints. Appendix A contains a proof that this problem is NP-complete. Thus, no known method can solve this problem in polynomial time, and it is necessary to consider heuristic methods. Originally, we implemented a greedy algorithm, which simply selected the virtual link that increased Ψ the most during each pass. However, the greedy algorithm performed poorly, and did not converge to optimal Ψ in the general case. Thus, we decided to use a more robust genetic algorithm.

4.1 Overview

A genetic algorithm (GA) uses selection, crossover, and mutation heuristics inspired by biological evolution to approach the solution to a problem [7, 9]. We have implemented a GA that, given a hypertext system, converges to R such that Ψ is maximal or near-maximal. To improve performance, it uses special crossover and mutation heuristics that are designed for the degree constraints of our hypertext model. It also includes several user-specified parameters, which can be adjusted for each hypertext system to yield optimal convergence time.

4.2 Procedure: GA for approximating maximal Ψ

Table 1 shows the set of input data and parameters for the GA. Algorithm 1 shows our implementation of the GA written in pseudocode.

After parsing the input data, the algorithm stores the hypertext system's pages and virtual links as a directed graph G_Γ using an adjacency matrix (line 2 of the pseudocode). Then, it computes the length of the shortest path in virtual links from each page to each other page, if such a path exists, and stores the results in a new matrix (defined on line 3). This is a classical problem called *all-pairs shortest paths*; we solve it using *Floyd's algorithm* (described in [14]), which runs in $O\left(|P|^3\right)$ time. There are algorithms with better time complexity, such as one presented in [5], but we judged Floyd's algorithm to be sufficient for use with the GA.

Input	**Type**	**Represents**
P	Set	Pages in hypertext
θ_p	$\mathbb{N}$	Degree of page p
d_p	$\mathbb{R} > 0$	Demand of page p
Γ	Set	Virtual links
R	Set	Real links
M	Set	Mandatory links
θ_s	$\mathbb{N}$	System degree
α	$0 < \mathbb{R} < 1$	Attention-span factor
ρ	$\mathbb{N} > 1$	Population size
m	$0 \leq \mathbb{R} < 1$	Mutation rate
elite	Boolean	Elitist selection on
fitness	Boolean	Fitness scaling on
t	$\mathbb{R} > 0$	Seconds until halt

Table 1: Inputs to the GA

```
1   Input $P, \Gamma, R, M, \theta_s, \alpha, \rho, m, elite, fitness, t$;
2   Let $G_\Gamma := (P, \Gamma)$ be a directed graph;
3   Let $H_\Gamma :=$ all_pairs_shortest_path$(G_\Gamma)$;
4   Let $D := \Gamma \cup M'$ be a set of links;
5   Let $\Omega :=$ an array of $\rho$ subsets of $D$;
6   Let $b_{i,j} = 1$ if $D_j \in \Omega_i$, 0 otherwise;
7   Let $b_{i,j} := 0 \; \forall \; 0 \le i < |\Omega|, 0 \le j < |D|$;
8   for $i := 0$ to $|\Omega| - 1$ {
9     while $(|\{\text{links } \in \Omega_i\}| < \theta_s)$ {
10      $n :=$ random_integer$(0...|D| - 1)$;
11      if $\left( \left| \left\{ \begin{array}{c} \text{links } \ell \in \Omega_i \text{ with} \\ \text{source}(\ell) = \text{source}((\Omega_i)_n) \end{array} \right\} \right| < \theta_{\text{source}((\Omega_i)_n)} \right)$
12      then $b_{i,n} := 1$; } }
13  do {
14    for $i := 0$ to $\rho - 1$ {
15      Let $R := b_i \cup M$;
16      Let $G_R := (P, R)$ be a directed graph;
17      Let $H_R :=$ all_pairs_shortest_path$(G_R)$;
18      Let $\Psi_i :=$ org_index$(H_R)$;
19      if $(\Psi_i > \Psi_{MAX})$ then{$\Psi_{MAX} := \Psi_i$; $\xi := i$;}
20      if $(\Psi_i < \Psi_{MIN})$ then $\Psi_{MIN} := \Psi_i$; }
21    if $(fitness)$ then for $i := 0$ to $|\Omega| - 1$
22      $\Psi_i := \Psi_i - \Psi_{MIN} + 1$;
23   Let $\Omega' :=$ an array of $\rho$ subsets of $D$;
24   Let $c_{i,j} = 1$ if $D_j \in \Omega'_i$, 0 otherwise;
25   Let $c_{i,j} := 0 \; \forall \; 0 \le i < |\Omega'|, 0 \le j < |D|$;
26    for $i := 0$ to $\rho - 1$ {
27      do { $x :=$ random_integer$(0...\rho - 1)$; }
28      until (random_real$(0...\Psi_{MAX}) \le \Psi_x$)
29      do { $y :=$ random_integer$(0...\rho - 1)$; } until
30      (random_real$(0...\Psi_{MAX}) \le \Psi_y$ and $y \ne x$)
31      for $j := 0$ to $|D| - 1$ { $c_{i,j} := 0$;
32        if $(b_{x,j} = 1$ and $b_{y,j} = 1)$ then $c_{i,j} := 1$; }
33      while $(|\{\text{links } \in \Omega'_i\}| < \theta_s)$ {
34        $n :=$ random_integer$(0... |D| - 1)$;
35        if $\left( \begin{array}{c} (b_{j,x} = 1 \text{ or } b_{j,y} = 1) \text{ and} \\ \left| \left\{ \begin{array}{c} \text{links } \ell \in \Omega'_j \text{ with} \\ \text{source}(\ell) = \text{source}\left((\Omega'_j)_n\right) \end{array} \right\} \right| \\ < \theta_{\text{source}\left((\Omega'_i)_n\right)} \end{array} \right)$
36        then $c_{i,j} := 1$; }
37      while (random_real$(0...1) \le m$) {
38        do {$n :=$ random_integer$(0... |D| - 1)$;}
39          until $(c_{i,n} = 1)$;  $c_{i,n} := 0$;
40        do { $n :=$ random_integer$(0... |D| - 1)$; }
41          until $(c_{i,n} = 0)$ and
42          $\left| \left\{ \begin{array}{c} \text{links } \ell \in \Omega'_j \text{ with} \\ \text{source}(\ell) = \text{source}\left((\Omega'_j)_n\right) \end{array} \right\} \right| < \theta_{\text{source}\left((\Omega'_i)_n\right)}$ ;
43        $c_{i,n} := 1$; } }
44      if $(elite)$ then $\Omega'_0 := \Omega_\xi$;
45      Let $\Omega := \Omega'$;
46  } until (seconds_so_far$() \ge t$ )
```

Algorithm 1: GA to maximize Ψ

Line 4 defines D as the set of virtual links that are not mandatory. We call these *discretionary links*, since candidate solutions to the hypertext optimization problem can be represented as subsets of them. Line 5 initializes an array Ω of these subsets; Ω will be used as the *parent population* in genetic reproduction. Line 6 represents Ω as a set of bit strings b in which '1' signifies that a discretionary link is included in a candidate solution and '0' signifies that it is excluded; this bit notation will be helpful for describing genetic operations. Line 7 clears all bits in b to 0.

On lines 8 to 12, the algorithm forms an initial population of candidate solutions randomly. It repeatedly selects a discretionary link at random (employing the "random_integer" function on line 10). Then, it evaluates whether adding this link to the current candidate solution would violate constraints on page degree (line 11; note that the "source" function returns a link's source page). If not, it adds the link (line 12); it repeats this process until the number of discretionary links equals θ_s, the system degree.

Lines 13-46 form the main body of the GA, which keeps iterating through generations until a user-specified amount of seconds have elapsed (line 46).

Lines 14-20 evaluate the organizational index Ψ for each member of the current parent population Ω. On lines 16-17, the algorithm forms a directed graph G_R of *real* links and solves the all-pairs shortest path problem over them. Now that page distances have been computed for both the virtual links (which are constant) and the current set of real links, the index Ψ can be easily computed in $O\left(|P|^2\right)$ time by the formula in Section 3. Lines 19-20 simply store the maximum and minimum values of Ψ in the parent population, as well as the position of the candidate solution with maximal Ψ.

If the user selected *fitness scaling* as a parameter, then lines 21-22 subtract the minimal Ψ from the index of each candidate solution, and then add 1 (so that all indices are nonzero). This technique significantly improved performance in empirical tests (see Section 6).

Line 23 defines a second array of candidate solutions Ω', which will be used as the *child population* in genetic reproduction. Line 24 represents Ω' as a set of bit strings c analogous to b; line 25 clears all bits in c to 0.

Lines 26-43 loop through the child population Ω', building each new candidate solution through genetic selection, crossover, and mutation operations on the parent population Ω. Lines 27-28 select the first parent for genetic crossover from Ω, with the probability of any candidate being selected proportional to its organizational index Ψ. Here, we perform weighted selection using a Monte Carlo technique, in which a candidate solution and a real number $0 \leq \mathbb{R} \leq \Psi_{MAX}$ are repeatedly selected at random until the real number is less than or equal to the candidate's index Ψ. However, another possibility would be *roulette-wheel selection* (described in [7]), in which the sum

$$\sum_{i=0}^{k} \Psi_i$$

is stored in position k of an array A, and binary search is used to select a candidate solution based on a random real number $0 \leq \mathbb{R} \leq A[\rho - 1]$. Monte Carlo selection tends to perform well when most values of Ψ are close to Ψ_{MAX}; roulette-wheel selection might run more quickly when the values of Ψ are widely distributed. Lines 29-30 select the second parent by a method identical to that for the first, except that an extra test ensures that both parents are distinct.

Lines 31-36 perform genetic crossover on the two selected parents. With our crossover function, discretionary links that appear in both parents will definitely appear in the child; those that appear in only one parent might appear in the child; and those that appear in neither parent will not appear in the child. Lines 31-32 set the child's bit array equal to the binary AND of the two parents' bit arrays. Then, lines 33-36 choose '1' bits at random from the binary OR of the two parents' bit arrays; test whether copying them to the child would violate constraints on page degree; and if not, copy them. The 'while' loop continues until the number of discretionary links in the child equals the system degree θ_s. (This algorithm is analogous to the one used for random candidate generation on lines 9-12.)

After a child has been generated through crossover, the GA performs random mutations on it (lines 37-43). In each mutation, a '1' bit is selected at random and changed to a '0' bit (lines 38-39). Then, '0' bits are repeatedly selected at random until one is found for which changing it to a '1' bit would not violate constraints on page degree[1] (lines 40-43); this bit is then flipped. The GA continues to mutate while a random real number $0 \leq \mathbb{R} \leq 1$ is less than or equal to the *mutation rate* m (line 37); thus, in principle, any number of mutations are possible.

If the user selected *elitist selection* as a parameter, then line 44 copies the best candidate solution from the parent population Ω into the first position in the child population Ω'. Like fitness scaling, this technique improved performance in empirical tests (see Section 6).

Line 45 sets the parent population for the next generation equal to the current child population. (This is implemented as a simple pointer exchange, not a transfer of data.) The GA loop continues until t seconds have elapsed, at which time the GA breaks and returns the best candidate solution in the current child population. Depending on user needs, particular implementations could have other criteria for breaking, such as whether the GA has iterated through a specified number of generations, or whether Ψ_{MAX} has attained a specified value.

[1]There is guaranteed to be such a '0' bit—if nothing else, it will be the one which was just before changed from a '1' bit.

5 Experiment with Human Subjects

Thus far, we have not presented evidence that the "organizational index" Ψ is related to the organizational efficiency of actual hypertext systems. Here, we describe an experiment wherein 70 computer users searched for information using three real link configurations with the same underlying hypertext.

5.1 Procedure

Our goal was to study the relationship between a hypertext system's organizational index and the perceived and actual speed with which users could find specific information in it. To accomplish this, we constructed several hypertext systems, each with equivalent information but with real links corresponding to different values of the index, Ψ. We intended to base these systems on an existing hypertext that had content of wide general interest; included a large number of pages, each with specific information; was truly "hypertextual," rather than simply hierarchical; contained some indication as to what virtual links should be; had some constraints on page degree; included detailed usage statistics; and was not copyrighted. Failing to find such a hypertext, we constructed our own, a 46-page system concerning the topic of the Watergate scandal. We arbitrarily chose $\alpha = 0.7$, $\theta_p = 6$ for each $p \in P$, and $\theta_s = 120$. d_p was 1 for almost every $p \in P$, 2 for a few pages of pivotal importance ("The White House," for instance), and 100 for the central home page of the system. In addition, each page had a mandatory link to the home page. Using the genetic algorithm described in Section 4, we created three renditions of this hypertext, System A with $\Psi \cong 7382$, System B with $\Psi \cong 8186$, and System C with $\Psi \cong 9039$. We made a few manual changes to Systems A and B to ensure that pages were reachable from the home page. We then converted these hypertexts into sets of HTML pages and placed them on a World Wide Web server. We configured this server to require users' names before they entered any of the hypertexts, and to store for later statistical analysis the sequence of links each individual user chose.

We then created a written instruction and question sheet (reproduced in Appendix B) to guide participants in an experiment that would compare the three hypertext systems. The instructions directed participants to seek information in all three systems on both of two *research topics*; then, if they found information, to rate their perception of how quickly they found it on a scale of 1 to 10. If users reported that they could not find relevant information at all, we would record a rating of 0 out of 10. The *research topics*, which were different for each participant, were randomly selected titles of pages in the system, with double weight given to pages with $d_p = 2$. The sheet also asked participants to write down two facts about their research topics if they found information, to ensure that they were following instructions.

For the actual experiment, we reserved a computer lab at Clarkson University for two days. Participants were Clarkson University students with varying computer experience. On the first day, participants were presented System A first, then System C, then System B; on the second, they were presented System B first, then System A, then System C. The ordering of the hypertext systems was thus minimized as a source of error.

5.2 Results

In all, 70 students participated in the experiment, 28 on the first day and 42 on the second. Since each participant was given two research topics, we thus had 140 *datasets* - user ratings of how quickly some piece of information was found in each of the three systems, if it was found at all. However, since some participants did not register their names before entering the hypertext systems, went from one system to another out of order, or otherwise failed to follow directions, only 125 datasets could be retrieved from the Internet server's access logs for statistical analysis.

The results confirm the prediction made in Section 3 that the organizational index is positively correlated with the speed of finding information. For System A ($\Psi_A = 7382$), the mean user rating on the 0 to 10 scale was 3.7 and the median rating was 1; for System B ($\Psi_B = 8186$), the mean rating was 5.3 and the median was 7; for System C ($\Psi_C = 9039$), the mean rating was 6.6 and the median was 8. Figure 2 and Table 2 show the distribution of ratings in each of the three systems. Note that the ratings are highly polarized - the average standard deviation for the three systems is 4.0.

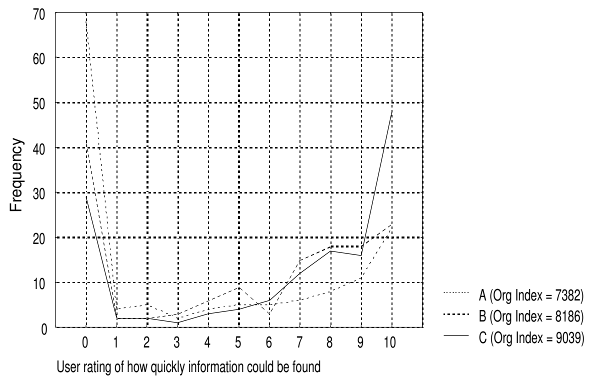


Figure 2: Distribution of user ratings (note that 0 was recorded when a user could not find information)

Score	**A** (7382)	**B** (8186)	**C** (9039)
0	68	41	29
1	4	2	2
2	5	2	2
3	2	3	1
4	4	6	3
5	5	9	4
6	5	3	6
7	6	15	12
8	8	18	17
9	11	18	16
10	22	23	48
Datasets	140	140	140
Mean	3.7	5.3	6.6
Median	1	7	8
Mode	0	0	10
St. Dev.	4.1	3.9	3.9

Table 2: User ratings

Analysis of the Internet server's access logs shows that, as Ψ increased, there was a corresponding decrease in the number of pages participants visited before they either found information on their research topics or gave up. (Note that by "number of pages" we do not mean number of *unique* pages.) For System A, the mean number of pages visited while looking for information on a research topic was 26.2 and the median number of pages was 20; for System B, the mean number of pages was 16.4 and the median was 9; for System C, the mean number of pages was 12.0 and the median was 7. Figure 3 and Table 3 show the distribution of number of pages accessed for each of the three systems. The standard deviation of number of pages accessed for Systems A, B, and C are 23.5, 16.8, and 11.9 respectively, indicating a wide range of path lengths needed to access information.

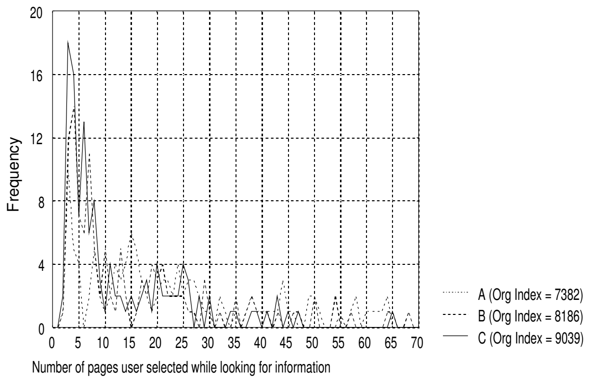
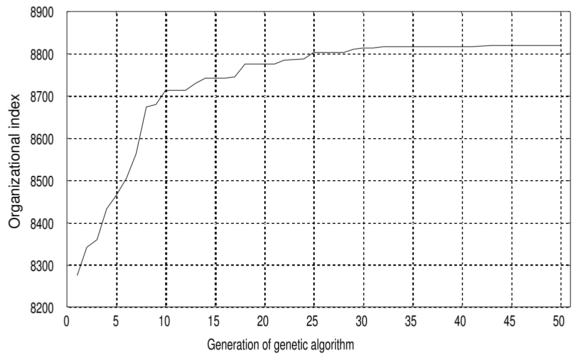


Figure 3: Distribution of number of pages accessed

Pages	**A** (7382)	**B** (8186)	**C** (9039)
Datasets	125	125	125
Mean	26.2	16.4	12.0
Median	20	9	7
Mode	2	3	2
St. Dev.	23.5	16.8	11.9

Table 3: Pages accessed



Example of GA convergence, with population=20, mutation=0.2, elitism, scaling

6 Performance of Genetic Algorithm

Any performance analysis of the GA described in Section 4 is necessarily subjective, since the algorithm by its nature evades computational classification and may deliver different performance based on the hypertext systems it receives as input.

For all of the hypertext systems we tested with $|P| \leq 10$, the GA converged to the optimum very quickly. For the hypertext described in Section 5 with $|P| = 46$, we ran the GA through tens of thousands of generations

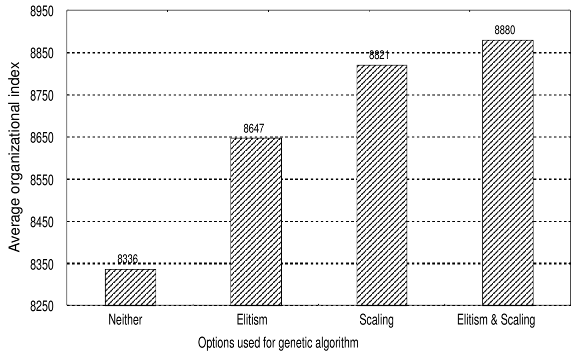


Figure 4: Effects of elitism and scaling on convergence rate, with generations=50, population=20, mutation=0.2

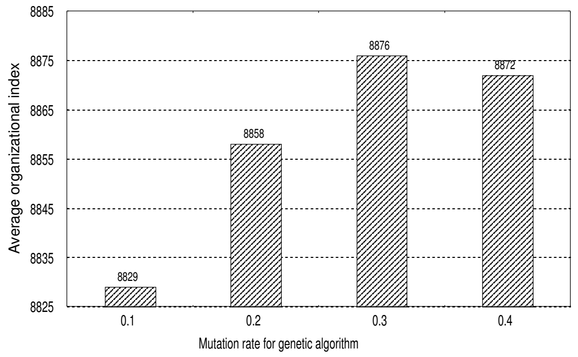


Figure 5: Effect of mutation on convergence rate, with generations=50, population=20, elitism, scaling

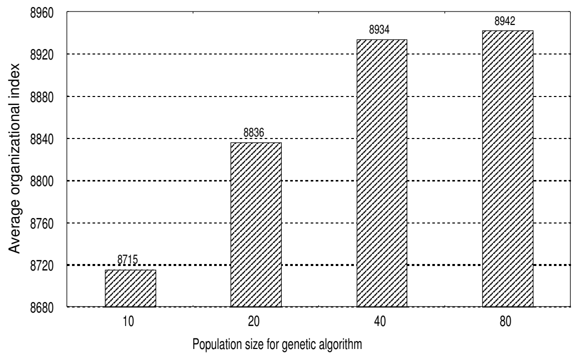


Figure 6: Effect of population size on convergence rate, with generations=50, mutation=0.2, elitism, scaling

and are still reasonably certain that we did not find the optimum value. Figure 4 illustrates how the GA converges toward a maximal value of Ψ; as with most optimization methods, Ψ initially increases very rapidly, then increases more slowly, finally reaching a plateau. (The organizational index would only surpass 9000 after thousands of generations.)

We have found that elitist selection and fitness scaling, two optional GA parameters described in Section 4, almost always accelerate the convergence of Ψ. Figure 5 shows the average values of Ψ returned after running the GA ten times with and without both elitist selection and fitness scaling. We have also found that mutation rate, within the range from 0.1 to 0.4, does not significantly affect the GA's performance. Figure 6 shows the results of ten-trial GA runs with varying mutation rates, in which $m = 0.3$ produces Ψ slightly higher than that for the other mutation rates tested. Finally, we have found that, although increasing population generally increases the convergence rate *per generation*, the convergence rate *per processing time* is usually maximized when $|\Omega| \cong 20$. Figure 7 shows the average organizational indices returned for ten runs of the GA with each of four population sizes.

7 Conclusion and Future Directions

The results described in Section 5 indicate a strong correlation between the organizational index Ψ and both the actual number of pages users access while seeking information and users' perceptions of how quickly they can find information. These results should not be taken as proof that the organizational index is a reliable measure of hypertext organization, especially since we tested only three sets of real links for one underlying hypertext system. However, there seems to be enough evidence that a numerical rating of hypertext organization has some usefulness to warrant further investigation into mathematical modeling of hypertext systems.

We have implemented the GA described in Section 4 as cross-platform console-based software. In the future, we plan to evolve our software into a class library that other developers can easily use. Some open problems are:

- How well does the organizational index perform for larger, more realistic hypertexts than the one used in Section 5?
- Can the organizational index be improved by taking into account the hierarchical structure of most hypertexts, or the need for every page to be reachable from every other?
- Can a model of hypertexts be designed in which the set of pages is not given a priori, but rather is constructed by the optimization algorithm?
- Can the genetic crossover and mutation heuristics described in Section 4, which account smoothly for constraints on vertex degree and total number of edges, be applied to other graph-theory problems?

Algorithmic optimization of hypertext link topology, such as that described here, is unlikely to replace design of hypertext systems by humans. There are structural and aesthetic nuances in hypertext design that algorithms would have extreme difficulty handling. Rather, the numerical methods described in this paper should be considered tools to aid hypertext designers in understanding and resolving organizational issues.

8 Acknowledgments

I would like to thank Dr. Christopher Lynch of Clarkson University for helpful advice on all aspects of this project, and especially for assistance with the NP-completeness proof. Special thanks are also due to Dr. Ramnath A. Lakshmi-Ratan, Director of Technical Excellence at Bell Laboratories, for explaining research on voice-prompt telephony services, and to Alex Halderman, for providing technical advice.

A Appendix: Computational Complexity

Our problem is to find a set of real links that maximizes Ψ under the given constraints. Here, we prove that no known method can solve this problem in polynomial time, and hence that it is necessary to consider heuristic methods.

Theorem 1 *Maximizing Ψ is NP-complete.*

Proof. Let us first rephrase the optimization problem as a decision problem: *Given a hypertext system and a constant $K > 0$, does there exist R such that $\Psi \geq K$?* Call this problem Hypertext Optimization (or HTO). Clearly, HTO is not any "harder" than the original problem (see [6] for example).

It is apparent that $HTO \in NP$, since a nondeterministic algorithm need only guess all permissible configurations of real links and check in polynomial time whether any of them have $\Psi \geq K$.

We will transform HC (the Hamiltonian cycle problem on a directed graph; known to be NP-complete [6]) into HTO. Given an instance $G = (V, E)$ of HC with $|V| \geq 2$, we map each $v \in V$ onto $p \in P$, and each $(v_0, v_1) \in E$ onto $(p_0, p_1) \in \Gamma$. For each $p \in P$, let $\theta_p = d_p = 1$. Furthermore, let $\theta_s = |P|$ and choose

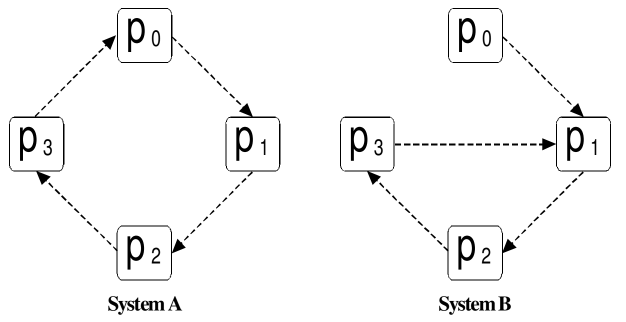


Figure 7: System A contains a Hamiltonian cycle and System B does not, so System A has a higher organizational index (given the bound on alpha)

$\alpha > \sqrt[2|P|-2]{1 - \frac{2}{(|P|-1)|P|}}$

(the reason for this restriction on α will be clarified shortly).

If G contains a Hamiltonian cycle, then for each $p_0, p_1 \in P$ with $p_0 \neq p_1$, the length of the shortest virtual path from p_0 to p_1 is at most $|P|$ - that is, $\gamma(p_0, p_1) < |P|$. Thus, if for each $(p_0, p_1) \in \Gamma$ along the Hamiltonian cycle we let $(p_0, p_1) \in R$, then

$y(p_0, p_1) < |P| \Rightarrow r(p_0, p_1) < |P|$

and we can give a lower bound on the organizational index $\Psi_{\exists HC}$. Since there are $\frac{1}{2}(|P|-1)|P|$ pairs of distinct (p_0, p_1) with $\gamma, r < |P|$,

$\Psi_{\exists HC} > \frac{1}{2}(|P|-1)|P|\alpha^{2|P|-2}$.

Figure 8 illustrates systems with and without Hamiltonian cycles.

If G does not contain a Hamiltonian cycle, then for some p_0, p_1, $\gamma(p_0, p_1) = \infty$ (because the page degree $\theta_p = 1$ for each $p \in P$). Since there are at most $\frac{1}{2}(|P|-1)|P| - 1$ pairs of distinct (p_0, p_1) with γ, r finite, we can specify an upper bound on Ψ:

$\Psi_{\nexists HC} < \frac{1}{2}(|P|-1)|P| - 1$.

Now, notice that because of the bound on α, it always holds that $\Psi_{\exists HC} > \Psi_{\nexists HC}$. Thus, if G contains a Hamiltonian cycle, there exists R such that

$\Psi \geq \frac{1}{2}(|P|-1)|P|\alpha^{2|P|-2}$.

Otherwise, there does not exist such an R. ■

References

[1] G. Bartlett, "Genie: A First GA," Practical Handbook of Genetic Algorithms, Vol I. CRC Press, 1995.

[2] B. Benjamin, Elements of Web Design. *www.cnet.com/Content/Features/Howto/Design*, 1996.

[3] P.D. Bruza & Th.P. Van Der Weide, "Assessing the Quality of Hypertext Views," ACM SIGIR Forum, 24(3):6-25, 1990.

[4] O. Etzioni, S. Hanks, T. Jiang, R.M. Karp, O. Madoni, O. Waarts, "Efficient Information Gathering on the Internet," 37^{th} Proceedings of Foundations of Computer Science, 234-243, 1996.

[5] D.R. Karger, D. Koller, S.J. Phillips, "Finding the Hidden Path: Time Bounds for All-Pairs Shortest Paths," SIAM Journal on Computing, 1199-1217, Vol. 22-6, 12/1993.

[6] M.R. Garey & D.S. Johnson, Computers and Intractability. Freeman, 1979.

[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[8] R.E. Horn, Mapping Hypertext. The Lexington Institute, 1989.

[9] S.R. Ladd, Genetic Algorithms in C++. M&T Books, 1996.

[10] R.A. Lakshmi-Ratan, Director of Technical Excellence, Bell Laboratories. Personal correspondence, 1996.

[11] H.V.D. Parunak, "Ordering the Information Graph," Chapter 20 in Hypertext/Hypermedia Handbook. McGraw Hill, 1991.

[12] T.C. Rearick, "Automating the Conversion of Text Into Hypertext," Chapter 10 in Hypertext/Hypermedia Handbook. McGraw Hill, 1991.

[13] R. Riner, "Automated Conversion," Chapter 9 in Hypertext/Hypermedia Handbook. McGraw Hill, 1991.

[14] R. Sedgewick, Algorithms in C++. Addison-Wesley, 1992.

B Appendix: Instructions and questions given to experiment participants

(Condensed to fit on page)

Name: ______________________________ Phone: x_____ Email: ___________________________________

If you win a prize, you would prefer to be contacted by (check one): ____ Phone ____ Email

Welcome to this experiment! Please read and follow the directions carefully.

Your first research topic is ____

1. Click **Home** on Netscape's toolbar. When the **Welcome Page** appears, select the link that says **Login**. Then, type in your *full* name (as it appears on this sheet) in the **Name** box, and press the **Continue** button.
2. When you are back to the **Welcome Page**, select the link that says **System A**.
3. Navigate through the hypertext system that appears, looking for information on your **first research topic**. *Do not become sidetracked - try to find the information by clicking on as few links as possible.* Do not click Home on the Netscape toolbar while you are looking for information. If, after a few minutes, you cannot find anything, **give up**.
4. Fill out the following questionnaire:

    Did you find any information on the **first research topic**? Yes No

    If so, jot down two facts about your topic:

    1. ____
    2. ____

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

5. Click **Home** on Netscape's toolbar. On the **Welcome Page**, select the link that says **System B**. Again, look for information on your **first research topic**, following exactly the same rules as before. Then, answer the following questions:

    Did you find any information on the **first research topic**? Yes No

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

6. Click **Home** on Netscape's toolbar. On the **Welcome Page**, select the link that says **System C**. *Once again* (we know it's boring, but it's in the name of science), look for information on your **first research topic**, following the rules from Step #3. Then, answer the following questions:

    Did you find any information on the **first research topic**? Yes No

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

Your second research topic is ____

7. Click **Home** on Netscape's toolbar. On the **Welcome Page**, select **System A**. Now, look for information on your **second research topic**, following the same rules. Then, answer these questions:

    Did you find any information on the **second research topic**? Yes No

    If so, jot down two facts about your topic:

    1. ____
    2. ____

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

8. You know what to do…click **Home**, select **System B**, and look for information on your **second research topic**. Then, answer these questions:

    Did you find any information on the **second research topic**? Yes No

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

9. Click **Home**, select **System C**, and look for information on your **second research topic**. Then, answer these questions:

    Did you find any information on the **second research topic**? Yes No

    On a scale of 1 to 10, how quickly were you able to find the information (if you found it)?

    Very slowly 1 2 3 4 5 6 7 8 9 10 *Very quickly*

Thanks for participating! Please turn in this sheet at the front.