How To Make World Wide Web Sites Faster and Easier to Use

Misha Koshelev
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
mkosh@cs.utep.edu

Abstract
We propose a new idea of organizing Web sites so that the Web will be easier and faster to use.

Problem
The World Wide Web is currently a very important way of disseminating information. Most WWW sites are organized as follows:

- The main page of the site usually contains a small amount of information and links to other pages in this Web site.
- These other pages, in turn, consist of either (more or less) “pure” information or of other links.

Sometimes, to be able to access the information he or she needs, the user has to go through several (intermediate) pages to get to the desired page. These intermediate pages contain, to the user, no useful information other than a link to either the next intermediate page or to the final page for which the user is looking. The retrieval of the intermediate WWW pages takes up the user’s time (especially if either the user or the server has a relatively slow Internet connection, or if the server is very busy).

Therefore, if too many clicks are necessary to reach the desired page, the user will simply give up and leave the site.

A Trivial “Solution” Does Not Work
To solve this problem, one could place all the links on one page, but this would be inconvenient: It is known in psychology that, in general, humans are most comfortable with 5 to 9 items to choose from (“7 plus minus 2” law, see, e.g., (Milner 1970)).

Precise Formulation of the Problem
It is desirable to organize a Web site in such a way that every intermediate page has no more than nine links, and the average number of clicks is the smallest possible.

If we have \( n \) pages, and we know the number of people \( p_i \) who have visited the \( i \)-th page and the number of clicks \( c_i \) necessary to access the \( i \)-th page, then the average number of clicks is equal to

\[
C = \frac{p_1 \cdot c_1 + p_2 \cdot c_2 + \ldots + p_n \cdot c_n}{p_1 + p_2 + \ldots + p_n}.
\]

The existing structure of a Web site can be represented by a tree of its links, with the main page as the tree’s root, etc. We must also get information from the Web site’s maintainer about the possible new intermediate pages that we can create, i.e., about all possible meaningful groupings of the links. For example, if a page contains eight links, the maintainer may prompt us that, if necessary, this page can be subdivided into three meaningful groups: links 1–4, 5–6, and 7–8.

The Main Idea of the Solution
To minimize \( C \), we suggest adding “hit counters” to all of the “leaf” WWW pages on a Web site (i.e., the pages which contain the actual information of the site). After the hits have been counted, we can try different tree rearrangements and find the tree with the smallest \( C \).

Preliminary Results
Our experiments show that we can get up to a 50% decrease in the average number of clicks by using this method.

Toy Example: Formulation of the Problem
Let’s describe the Web site of a fictitious company called El Paso Hot Dogs Co. Its main page has three links:

- Our Products
- Our Services
• Miscellaneous
Each of these links, in turn, leads to a new page. For example, the link to Our Products leads to a new page with the following three links:

• Hot Dog
• Hot Dog with Cheese
• Hot Dog with Chili

The last two links leads directly to the corresponding leaf pages, while the first link leads to a new page with 2 links in it: to Picture and Hot Dog Coupon Book. The resulting “link tree” (with the numbers of hits in parentheses) is as follows:

• Our Products
  – Hot Dog
    * Picture (5)
    * Hot Dog Coupon Book (50)
  – Hot Dog with Cheese (10)
  – Hot Dog with Chili (10)

• Our Services
  – Hot Dog Parties/Catering (50)
  – Hot Dog Weddings (10)
  – Hot Dog Prom (1)

• Miscellaneous
  – Our History (1)

The average number of clicks \( C \) for this site is 2.4.

Toy Example: Solution
If we move the two most frequently accessed pages in this site (the Hot Dog Coupon Book and Hot Dog Parties/Catering) to the main page, then the resulting link structure can be represented by the following tree:

• Our Products
  – Hot Dog
    * Picture (5)
  – Hot Dog with Cheese (10)
  – Hot Dog with Chili (10)

• Hot Dog Coupon Book (50)

• Our Services
  – Hot Dog Weddings (10)
  – Hot Dog Prom (1)

• Hot Dog Parties/Catering (50)

• Miscellaneous
  – Our History (1)

with \( C = 1.3 \) (\( \ll 2.4 \)).

Similar Problem
A similar problem of organizing information kiosks was considered by J.-Y. Jaffray (Jaffray 1995), (Jaffray and Lorit 1996), (Jaffray 1997) (his toy example is a kiosk for the Louvre).

Our problem is much simpler than the kiosk problem because most information kiosks provide no method of determining the number of “hits” to a certain page.

Fuzzy Case
Optimizing a Web Site Without Hit Counters
In many cases, setting up hit counters is the best way to optimize a Web site. However, there are two situations in which optimization without hit counters is desirable:

• Infrequently Accessed Web Sites. Sometimes, we may want to optimize a Web site before we have any hit count data. For example, on Web sites which have few visitors, it may take a very long time to collect enough information to be able to correctly optimize the site.

• Multimedia-Oriented Web Sites. A hit counter, just like other material on a Web page, takes time to load. So, for multimedia-oriented Web sites with a large amount of graphics, adding hit counters to a page may make it take an intolerably long time to load.

Fuzzy Logic
When we do not have hit counts, we can ask experts to predict the number of hits. It is unrealistic to expect that experts can predict exact numbers, but they are often able to express their predictions with words such as “few,” “many,” etc. In order to describe these words in mathematical terms, we can use fuzzy numbers (see, e.g., (Klir and Yuan 1995)). To obtain these fuzzy numbers, we can perform the following standard steps:

1. Suppose that there are \( E \) experts who can answer questions about this Web site. For each page \( i \), select the expert most knowledgeable about this page, ask his or her opinion on how many users will visit this page, and record his or her answer \( w_i \) (e.g., “a few,” “lots,” etc.).

2. For each word \( w \) used by at least one of the experts, ask each of the \( E \) experts if he/she believes that this word can be used to describe 0, 1, 2, 3, . . . users (e.g., that 2 users is a few users). Then, record the number of experts who believe this to be true as \( e_{w}(0), e_{w}(1), e_{w}(2), \ldots \).
3. Having collected this information, define, for each term \( w \), the membership function \( \mu_w \) as follows:

\[
\mu_w(0) = \frac{e_w(0)}{E}, \mu_w(1) = \frac{e_w(1)}{E}, \mu_w(2) = \frac{e_w(2)}{E}, \ldots
\]

4. For each page \( i \), we take \( \mu_i(\alpha) = \mu_w(\alpha) \), where \( w \) is the natural-language word that the expert used to identify the number of users that will visit the \( i \)-th page.

**How to Formulate the Problem in Precise Terms**

Our goal is to minimize the average number of clicks \( C \).

- In the previous section, we considered the case when we know the exact number of visitors \( p_i \) to our Web site. In this case, for each rearrangement of the Web site, we can compute the exact value of \( C \).
- Now, we consider the cases when we do not know the exact values of \( p_i \). Instead, we know the membership functions \( \mu_i(\alpha) \) that describe the number of visitors. In other words, for every leaf \( i \), we know the fuzzy number \( p_i \).

Since \( p_i \) are fuzzy numbers, the resulting value \( C \) is also a fuzzy number. To define the corresponding fuzzy number \( C \), with a membership function \( \mu_C(\alpha) \), we can use a standard tool of fuzzy logic: the extension principle (Klir and Yuan 1995). According to this principle, if we know the membership functions \( \mu_i(\alpha) \) that describe the fuzzy variables \( x_1, \ldots, x_n \), then the membership function \( \mu(\alpha) \) that describes the fuzzy variable \( y = f(x_1, \ldots, x_n) \) is defined as

\[
\mu(\alpha) = \max_{x_1, \ldots, x_n : y = f(x_1, \ldots, x_n)} (\min(\mu_1(x_1), \ldots, \mu_n(x_n))). \quad (2)
\]

In our case, \( x_i = p_i \) and the function \( f(x_1, \ldots, x_n) \) is given by formula (1).

We want to choose the site organization with the smallest value of \( C \).

- When we knew the exact number of visitors, we could directly compare different site organizations by simply comparing their corresponding average numbers of clicks \( C \).
- When the values of \( C \) are fuzzy numbers, it is difficult to compare them directly. So, we need to have some numerical (crisp) characteristic by which we can compare different site organizations.

Which characteristics should we choose? A similar problem of choosing a crisp number that characterizes a membership function has been thoroughly analyzed in fuzzy control. In fuzzy control, the corresponding methods are called defuzzification methods. The most widely used defuzzification method is the centroid defuzzification procedure \( D(\mu) = (\int x \mu(x) dx) / (\int \mu(x) dx) \). Therefore, we will select a site organization for which this characteristic \( D(\mu_C) \) is the smallest possible.

Now, we are ready to formulate the problem in precise terms.

**Precise Formulation of the Problem**

**GIVEN:**

- Fuzzy numbers \( p_1, \ldots, p_n \) describing the number of visitors to the leaf pages of an \( n \)-page Web site.
- A tree describing the current organization of the Web site.

**FIND:**

- A new organization (tree) for the given Web site for which the defuzzified value \( C = D(\mu_C) \) of the (fuzzy) average number of clicks

\[
C = \frac{p_1 \cdot c_1 + p_2 \cdot c_2 + \ldots + p_n \cdot c_n}{p_1 + p_2 + \ldots + p_n} \quad (3)
\]

is the smallest possible.

**Motivation for the Algorithm**

To solve this problem, we will first develop an algorithm for computing \( \hat{C} \), and then use a genetic algorithm to find the desired Web site organization.

To compute the membership function \( \mu_C(\alpha) \) (that corresponds to \( \hat{C} \)), we will use a known result (see, e.g., (Klir and Yuan 1995)) to break down this problem from a problem with fuzzy numbers to several problems with intervals. Namely, if \( y \) is determined by formula (2), then for every \( \alpha \in [0, 1] \), the \( \alpha \)-cut \( y(\alpha) = [y^-(\alpha), y^+(\alpha)] \) (defined as \( \{x : \mu(x) \geq \alpha\} \) if \( \alpha > 0 \) and \( \{x : \mu(x) > 0\} \) for \( \alpha = 0 \)) is related to the \( \alpha \)-cuts of \( x_i \) by the formula

\[
y(\alpha) = f(x_1(\alpha), \ldots, x_n(\alpha)) = \{f(x_1, \ldots, x_n) : x_i \in x_i(\alpha), \ldots, x_n \in x_n(\alpha)\}.
\]

To use this idea, we must repeat these computations for several values of \( \alpha : \alpha = 0, \Delta \alpha, 2 \cdot \Delta \alpha, \ldots, 1.0 \). The value of \( \alpha \) describes an expert’s degree of belief. It is difficult to imagine that an expert would be able to distinguish between, say, the degree of
belief 0.87 and 0.88. Therefore, to describe the experts’ beliefs, it is quite sufficient to use eleven values $\alpha = 0, 0.1, 0.2, \ldots, 1.0$ that correspond to $\Delta \alpha = 0.1$.

In our case, the function $f(x_1,\ldots,x_n)$, as described by formula (1), is fractionally-linear. Therefore, to compute the corresponding intervals $C(\alpha)$, we can use an existing algorithm developed in (Lea, Kreinovich, and Trejo 1996).

To compute the defuzzification result, we will use a method of computing the centroid defuzzification procedure described in (Nguyen and Kreinovich 1996).

Now, we are ready to describe the algorithm.

**Algorithm for Computing $C$**

Take $\Delta \alpha = 0.1$ (any value of $\Delta \alpha \leq 1.0$ may also be chosen) and repeat the following steps 1–5 for $\alpha = 0, \Delta \alpha, 2 \cdot \Delta \alpha, \ldots, 1.0$:

1. For each leaf page $i$, use the binary search algorithm to find the values $p_i^- (\alpha) \leq p_i^+ (\alpha)$ for which $\mu_i (p_i^- (\alpha)) = \mu_i (p_i^+ (\alpha)) = \alpha$. (For $\alpha = 0$, $p_i^- (\alpha)$ is the smallest value for which $\mu_i > 0$ and $p_i^+ (\alpha)$ is the largest value for which $\mu_i > 0$.)

Then, in the following four steps, we will use an algorithm from (Lea, Kreinovich, and Trejo 1996) to compute $C$:

2. If $c_i = c_j$ for some $i \neq j$, replace $p_i(\alpha)$ and $p_j(\alpha)$ with a single new interval $p_i(\alpha)$, for which $c_i$ stays the same as before, but for which the new interval $p_i(\alpha)$ is equal to $p_i(\alpha) + p_j(\alpha)$, i.e., for which $p_i(\alpha) = p_i(\alpha) + p_j(\alpha)$ and $p_i(\alpha) = p_i(\alpha) + p_j(\alpha)$.

3. Sort the intervals $p_i(\alpha)$ in the increasing order of the corresponding coefficients $c_i$.

4. Compute $C^+ (\alpha) = \max(C_0^+ (\alpha), C_1^+ (\alpha), \ldots, C_n^+ (\alpha))$, where

$$C_k^+ (\alpha) = \frac{N_k^+ (\alpha)}{D_k^+ (\alpha)},$$

$$N_k^+ (\alpha) = p_1^+ (\alpha) \cdot c_1 + p_2^+ (\alpha) \cdot c_2 + \ldots + p_k^+ (\alpha) \cdot c_k + p_{k+1}^+ (\alpha) \cdot c_{k+1} + \ldots + p_n^+ (\alpha) \cdot c_n,$$

and

$$D_k^+ (\alpha) = p_1^- (\alpha) + p_2^- (\alpha) + \ldots + p_k^- (\alpha) + p_{k+1}^- (\alpha) + \ldots + p_n^- (\alpha).$$

5. Compute $C^- (\alpha) = \min(C_0^- (\alpha), C_1^- (\alpha), \ldots, C_n^- (\alpha))$, where

$$C_k^- (\alpha) = \frac{N_k^- (\alpha)}{D_k^- (\alpha)},$$

$$N_k^- (\alpha) = p_1^- (\alpha) \cdot c_1 + p_2^- (\alpha) \cdot c_2 + \ldots + p_k^- (\alpha) \cdot c_k + p_{k+1}^- (\alpha) \cdot c_{k+1} + \ldots + p_n^- (\alpha) \cdot c_n,$$

and

$$D_k^- (\alpha) = p_1^- (\alpha) + p_2^- (\alpha) + \ldots + p_k^- (\alpha) + p_{k+1}^- (\alpha) + \ldots + p_n^- (\alpha).$$

Let’s now find a defuzzified value $\bar{C}$ of $C$ (by using an algorithm from (Nguyen and Kreinovich 1996)). First, find the midpoint

$$m(\alpha) = \frac{C^- (\alpha) + C^+ (\alpha)}{2}$$

and the width $l(\alpha) = C^+ (\alpha) - C^- (\alpha)$ of each interval $C(\alpha)$. Then, compute:

$$\bar{C} = \frac{m(0) \cdot l(0) + m(\Delta \alpha) \cdot l(\Delta \alpha) + \ldots + m(1.0) \cdot l(1.0)}{l(0) + l(\Delta \alpha) + \ldots + l(1.0)}$$

**Optimization Using Genetic Algorithms**

Since the Web site optimization problem is discrete, we cannot use differentiation to find the organization corresponding to the smallest average number of clicks. Instead, we will use genetic algorithms (see, e.g., (Alander 1993), (Kreinovich, Quintana, and Fuentes 1993), (Goldberg 1994), (Mitchell and Forest 1993), (Srinivas 1994)).

In this algorithm, each possible organization of the Web site will be represented in the computer by the depths $c_1, c_2, \ldots, c_n$ of each leaf page within the newly organized site (i.e., the number of clicks that are necessary to get to this page). Therefore, it is desirable to know the maximum possible depth $M_i$ of each leaf page $i$. This number can be computed based on the initial information about the meaningful groupings.

In order to be able to check whether a given intermediate page has no more than nine links, we assign a unique label to each intermediate page and to each meaningful grouping. For every leaf, we store the list of all labels which describe its ancestors.

To use genetic algorithms, we represent each organization by its genetic code: a sequence $\tilde{c}$ of $n$ bytes that contain integers $c_1, c_2, \ldots, c_n$. In each generation, we will have the same number $N$ of organisms: $\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_N$.

To simulate selection, we use the following fitness function:

- First, we compute the average number of clicks $C(\tilde{c}_k)$ (in the crisp case) or the defuzzified average number of clicks $\bar{C}(\tilde{c}_k)$ (in the fuzzy case). 

Then, we penalize the organizations with more than nine links on some pages by computing the corrected objective function \( P(\vec{c}_k) = C(\vec{c}_k) + b(\vec{c}_k) \cdot p \), where \( b(\vec{c}_k) \) is the number of such "bad" pages, and \( p \) is a large positive number.

The classical genetic algorithm maximizes the objective function \( J(x) \). Since we are trying to minimize our objective function, we take \( J(\vec{c}_k) = P_{\text{max}} - P(\vec{c}_k) \), where \( P_{\text{max}} = \max(P(\vec{c}_1), \ldots, P(\vec{c}_N)) \).

Mutation is also simulated in a slightly different way:

- We use 1 byte (= 8 bits) to store each number of clicks \( c_i \).
- In principle, eight bits can store all integers from zero to \( 2^8 - 1 = 255 \).
- However, even 16 = \( 2^4 \) clicks is too much. Therefore, we will use at most four bits to store \( c_i \), and the four remaining bits will be simply zeroes.

Thus, instead of changing a bit, we mutate the entire byte by replacing it with a random value between 0 and \( M_i \).

Acknowledgments

This work was partially supported by the SC-COSMIC Consortium in Computational Sciences and by NASA through grants No. NCCW-0089 and NCC 5-97. The author is very grateful to Professor Jean-Yves Jaffray (LAFORIA–IBP, Université Paris VI) for his encouragement and support, and to all the participants of the 1996 SC-COSMIC Conference for valuable discussions.

References


