Web Page Ranking based on Fuzzy and Learning Automata

Zohreh Anari
Computer Eng.Department
Azad University of shabestar
Shabestar,Iran
zohreh_anari@iaushab.ac.ir

Mohammad Reza Meybodi
Computer Eng.Department
Amirkabir University of Technology
Tehran,Iran
mmeybodi@aut.ac.ir

Babak Anari
Computer Eng.Department
Azad University of shabestar
Shabestar,Iran
babak_anari@iaushab.ac.ir

ABSTRACT

The main goal of web pages ranking is to find the interrelated pages. In this paper, we introduce an algorithm called FPR-DLA. In the proposed method learning automata is assigned to each web page which its function is determining the weight of hyperlinks between web pages. Also for determining the weight of each web page parameters such as time duration on a web page and the importance of web pages are considered. Time duration on a web page and the importance of web pages are characterized as a fuzzy linguistic variable. The proposed algorithm calculates the rank of each web page as recursive according to the weights of each web page and hyperlinks between web pages. Experimental results show that the proposed method has a considerable efficiency in determining the rank of web pages.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – Data Mining, H.3.5 [**Information Storage and Retrieval**]: Online Information Services – Web-based Services

General Terms

Algorithms, Experimentation.

Keywords

Ranking, Learning Automata, Fuzzy Variable, Web mining

1. INTRODUCTION

With the rapid growth of the web, users get easily lost in the rich hyper structure. Providing relevant information to the users to cater to their needs is the primary goal of web site owners. Therefore, finding the content of the web and retrieving the user's interests and needs from their behavior have become increasingly important. Two page ranking algorithms, HITS and Page Rank, are commonly used in web structure mining [4,5].Both algorithms treat all links equally when distributing rank scores. The idea of Page Rank which is offered by Page and Brin is used in the search engine of Google to arrange the results of search. It measures the importance of the pages by analyzing the links. Even though the Page Rank algorithm is used successfully in Google, one problem still exists: in the actual web, some links in a web page may be more important than are the others. In some ranking algorithms such as [1,14] only pay attention to how user's use the hyperlink between web pages and the weight of every web page that can be effective factor in determining users interest and preference to this web page don't considered. There are many factors in determining the importance of each web page. The page is important if many users have referred to it. For example in ranking algorithm [6] the weight of every web page is determined only by the number of times the page was visited. The major shortcoming of this algorithm is that it considers only the number of access frequency to each web page by users.
In this paper, we propose a new method for computing the rank of web pages. Our method includes three steps. In the first step we use learning automata to determine the weight of hyperlinks between web pages. In the second step we compute the weight of each web page. We consider three parameters in determining the weight of web pages. The first parameter is time duration on a web page. It shows users' interest and preference. Since time durations are numerical and the page importance is linguistic, fuzzy-set concepts are used to process them [18].
The second factor is the importance of web pages which is gained by expert managers or other methods. This parameter can also be expressed by linguistic variables.
The third factor is the number of access frequency to each web page. We have used the product of these three factors for determining the weight of each web page. After determining the weight of web pages and hyperlinks we use the proposed recursive formula for computing the rank of web pages.
The paper is organized as follows: In section 2 the learning automata and the distributed learning automata will be explained. In section 3 we explain the proposed method for determining the weight of hyperlink between web pages and the weight of web pages. Section 4 includes some experimental evaluation of the proposed approach. Finally, section 5 draws a conclusion.

2. LEARNING AUTOMATA (LA)

An automaton can be regarded as an abstract model that has finite number of actions. This action is applied to the selected action of automata. The random environment evaluates the applied action and gives a grade to the selected action of automata. The response from environment (i.e. grade of action) is used by automata to select its next action. By continuing this process, the automaton learns to select an action with best grade. The learning algorithm used by automata to determine the selection of next action from the response of environment [13]. Figure 1 shows the relationship between the environment and the learning automata.

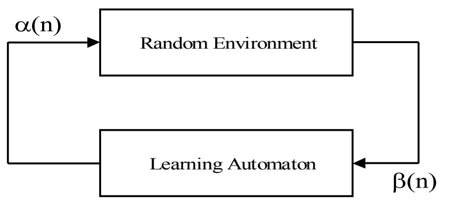

Figure1: The relationship between learning automata and the environment

2.1 Environment

The environment can be shown by E ≡{α, β, c} in which $\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_r\}$ represents a finite action / output set, $\beta=\{\beta_1, \beta_2,\ldots, \beta_m\}$ represents a input / response set, and $c=\{c_1, c_2,\ldots,c_r\}$ is the set of penalty probabilities, where each element c_i corresponds to one action α_i of the set α . The output (action) α_n of the automaton belongs to the set α, and it is applied to the environment at time t = n.

2.2 Learning Automata with Variable Structure

Variable structure learning automata is represented by < β, α, T, *p* >, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is a set of actions. β = {0, 1} is the set of inputs from the environment; where 0 represents a reward and 1 represents a penalty, $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ is learning algorithm and define the method of updating the action probabilities on receiving an input from the random environment. $p = \{p_1(n), p_2(n)\ldots p_r(n)\}$ is the action probability vector, where $p_i(n)$ represents the probability of choosing action α_i at time *n*. In these kinds of automata, if the action of α_i is chosen in the nth stage and receive the desirable response from the environment, the probability of $p_i(n)$ increases and the other probabilities decreases and in undesirable response, the probability of $p_i(n)$ decreases and the other probabilities increases. The following algorithm is one of the simplest learning schemes for updating action probabilities, and is defined as follows:

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$\forall j \quad j \neq i \quad p_j(n+1) = (1-a)p_j(n) \tag{1}$$

a) Desirable response

$$p_i(n+1) = (1-b)p_i(n)$$
$$\forall j \quad j \neq i \quad p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n) \tag{2}$$

b) Undesirable response

As seen from the definition, the parameter *a* is associated with reward response, and the parameter *b* with penalty response. According to the values of *a* and *b* we can consider three scheme. If the learning parameters *a* and *b* are equals, the scheme called reward penalty (L_{R-P}) . When *b* is less than *a*, we call it linear reward epsilon penalty ($L_{R\varepsilon P}$) scheme. When *b* equals to zero, we call it as linear reward inaction (L_{R-I}) scheme. For more information about the theory and applications of learning automata, refer to [9, 11].

2.3 Distributed Learning Automata (DLA)

A distributed learning automaton (DLA) is a network of learning automata which collectively cooperate to solve a particular problem. In DLA, the number of actions for any automaton in the network is equal to the number of outgoing edges from that automaton. When an automaton selects one of its actions, another automaton on the other end of edge corresponding to the selected action will be activated. At any time only one automaton in the network will be active. Formally, a DLA with n learning automata can be defined by a graph (V, E).Where V= $\{LA_1, LA_2\ldots LA_n\}$ is the set of automata and $E \subset V \times V$ is the set of edges in the graph in which an edge (LA_i, LA_j) corresponds to action α_j of automata LA_i. The action probability vector for automaton LA_i is shown by $p^i = \{p_1^i, p_2^i, \ldots, p_m^i\}$ where p_m^i denotes the probability of selecting action α_m, that is, edge (LA_i, LA_m). The choice of action α_m^i by LA_i activates LA_m. r_i shows the number of actions done by LA_i automata. For more information about distributed learning automata and its applications, refer to [3, 15].

3. THE PROPOSED METHOD

Our method includes three steps. In first step we compute the weight of each hyperlink between web pages based on learning automata. We use each user navigation path which has been stored in log file for computing weight of each hyperlink between web pages. In second step we compute weight of each web page. We consider product of three factors for computing weight of each web page. These factors are: the time duration on a web page, the importance of each web page and the number of access frequency to a web page. The more weight of web page shows that the users have more interested to this page.

3.1 Determining the Weight of Hyperlinks between Web Pages based on Learning Automata

Web pages and users play the role of stochastic environment for the existing learning automata in DLA. In the proposed method for each web page like P_i the learning automata LA_i is considered. If n is the number of web pages, in this case, an automaton has n-1 actions. When a user moves from page P_i to P_j, the jth action of LA_i automata in DLA is activated and the action probability vector for LA_i automata is updated based on learning algorithm. This action probability vector shows the weight of hyperlinks between pages i and j. Let $p_m^k(n)$ be the probability of choosing action α_m in learning automata LA_k at the nth time. If the user moves from page D_k to page D_m ($D_k \rightarrow D_m$) learning automata LA_k updates its action probability vector based on learning algorithm. For example if the action vector and the action probability vector of learning automata LA_1 are (A_2, A_3, A_4), (0.2, 0.5, 0.3) respectively and the user moves $D1 \rightarrow D_2$ then $E_2^1 = -(0.2\log 0.2 + 0.8\log 0.8) = 0.21$ and $a_2^1 = 0.21/(1+0.21) = 0.17$. In this case the action probability vector of LA_1 automata changes to (0.35, 0.42, 0.23). In this paper we used this algorithm for determining the weight of hyperlinks between web pages. This algorithm is presented in Figure 2.

```
1- Create a DLA according to web pages structure.
2- For each learning automata do
    2-1 Initialize the action probability vector where
        $p_i^k(n) = \frac{1}{r}, for \quad 1 \le i \le r \quad \sum_{i=1}^{r} p_i^k(n) = 1$
3- For every user navigation path in the log file do
    3-1 If the user moves from $D_k \rightarrow D_m$ then update the action probability vector for $LA_k$ automata according to the following learning algorithm.
        $p_m^k(n+1) = p_m^k(n) + a_m^k[1 - p_m^k(n)]$    (3)
        $p_j^k(n+1) = (1 - a_m^k) p_j^k(n) \; j \ne m \; \forall j$    (4)
        $a_m^k = \frac{E_m^k}{1 + E_m^k}$    (5)
        $E_m^k = -(p_m^k \log p_m^k + (1 - p_m^k)\log(1 - p_m^k))$    (6)
```

Figure 2: Algorithm for determining weight of hyperlinks between web pages base on learning automata.

3.2 Determining the Weight of Web Pages

Suppose the weight of web page like k is w_k, in the proposed algorithm this weight will be determined by the product of three main factors: the time duration on a web page, the importance of each web page and the number of access frequency to a web page. Time duration denotes time spending on the current visited web page. All time duration on web pages are divided into three fuzzy regions Short, Middle, and Long according to the method introduced in Wang [16]. Each fuzzy region is characterized as a fuzzy linguistic variable. Assume the membership functions of time duration are shown in Figure 3. Fuzzy set was first introduced by Zadeh [18] in 1965.In this section some basic concepts and results of fuzzy variable are reviewed.

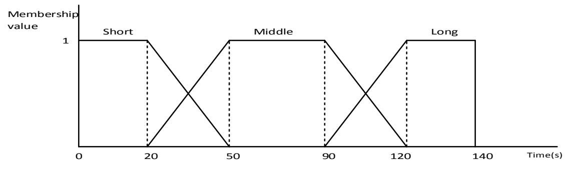

Figure 3: The membership functions of the time duration

Definition: (Liu and Liu [10]) the expected value of a trapezoid fuzzy variable (r_1, r_2, r_3, r_4) is defined as follows:

$$E[\eta] = \frac{1}{4}(r_1 + r_2 + r_3 + r_4) \qquad (7)$$

The expected value of a triangular fuzzy variable (r_1, r_2, r_3) is defined as follows:

$$E[\eta] = \frac{1}{4}(r_1 + 2r_2 + r_3) \qquad (8)$$

The expected value of a fuzzy variable can characterize this fuzzy variable by numerical value. The method of denoting the time duration on a web page k by a linguistic term is as follows [17]:

Step1. Assume that the domain of time duration on all web pages is divided into l fuzzy regions depicted by l different linguistic terms. Every linguistic term is characterized as a corresponding fuzzy variable η_j $j=1, 2,..., l$.

Step2. For all time durations $t_1, t_2, ..., t_v$ on the web page k, calculate the membership degrees in every fuzzy region $\mu_{\eta j}(t_i)$ $j=1, 2, ..., l$. respectively. Then calculate the scalar cardinality $count_j$ which definition is as

$$count_j = \sum_{i=1}^{v} \mu_{\eta j}(t_i) \qquad (9)$$

Step3. The fuzzy variable corresponding the largest scalar cardinality *Count =1, 2,..., l* is selected to depict the time duration on the web page k. The relative time duration T_k of the web page k can be described as follows:

$$T_k = \frac{E[\eta_k]}{T_{\max}} \qquad (10)$$

In this relation T_{max} is the given maximal time duration on web pages. $E[\eta_k]$ is the expected value of fuzzy variable η_k. Their expected values can be gained by equation 7.

Assume the membership functions for importance of the web page are given in figure 4. In figure 4, the importance of the web page is divided into five fuzzy regions: Very unimportant, Unimportant, Ordinary, Important, Very important. Each fuzzy region is represented by a membership function. The membership function in figure 4 can be represented as follows: Very unimportant (vu): (0, 0, 0.25), Unimportant (u): (0, 0.25, 0.5), Ordinary (o): (0.25, 0.5, 0.75), Important (I): (0.5, 0.75, 1) and Very Important (VI): (0.75, 1, 1).

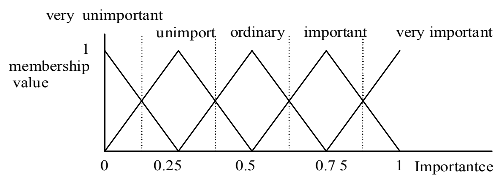

Figure 4: The membership functions of importance of web pages

To determine the importance of web pages, we first calculate the rank of each web page with one of the suggested ranking algorithm [1, 2]. In this paper we have used algorithm [1].Then the ranks are normalized between zero and one. According to the rank of every page, the fuzzy region of each page is determined according to Figure 4. Then the expected value for each web page can be gained by equation 8. By having the rank of each page and predefined membership functions for the importance of web pages, the values of importance for every web page can be calculated. The importance of k^{th} web page L_k calculates as follows:

$$L_k = E[\eta_k] \qquad (11)$$

The following relations between real numeric importances of web page V_k with fuzzy linguistic variable η_k can be gained from figure 4 as

$$\eta_k = \begin{cases} 0, & v_k = 0 \\ Veryunimportant, & 0 < v_k \le 0.125 \\ Unimportant, & 0.125 < v_k \le 0.375 \\ Ordinary, & 0.375 < v_k \le 0.625 \\ Important, & 0.625 < v_k \le 0.875 \\ Veryimportant, & 0.875 < v_k \le 1 \end{cases} \qquad (12)$$

Therefore the weight of web page *k* is described as follows:

$$w_k = L_k \times T_k \times S_k \qquad (13)$$

L_k is the importance of k^{th} web page and T_k shows the relative time duration of the k^{th} web page. The number of access frequency shown by S_k.

3.3 FPR-DLA Page Rank

We define $w_{i \to j}$ as the weight of hyperlink between pages *i* and *j*, this weight is computed based on section 3.1 we also compute w_i as the weight of page *i* , this weight is determined based on section 3.2. We can consider DLA as an N×N matrix. Such that every row *i* of matrix represent a web page (introduces automata *i*) and each column *j* represent the j^{th} action for automata *i*. Since the addition of matrix rows equals to one therefore, M is a stochastic transition matrix. At first, the value of each element from m_{ij} matrix is (1/N). When a user enters to the system and observes the page P_i, the learning automata for that page (i.e. LA_i) becomes activated. When the user moves from page P_i to P_j, the action is chosen in LA_i automata. According to the characteristics of Markov theory and the Page Rank computation, the proposed algorithm is the solution of the following equation [4]:

$$PR = (1-d)\ M \times PR + dp \qquad (14)$$

d is damping factor (usually set to 0.15). The Probability value for weight of each page described as a vector such as p:

$$p = \left[\frac{w_i}{\sum_{p_j \in In(w_j)} w_j}\right]_{N \times 1} \qquad (15)$$

Algorithm convergence guarantees only when *M* is irreducible and aperiodic [12]. The aperiodic limitation guarantees in action in the web environment. Irreducibility is usually done by adding the damping factor (*1-d*) to the rank propagation. Since the value of each entry m_{ij} from the matrix is at first equal to (1/N), therefore, it is guaranteed that equation 14 will converge to a unique vector FPR-DLA.

Definition (FPR-DLA): We define the page ranking based on distributed learning automata (FPR-DLA) as a recursive formula.

$$FPR-DLA\ (LA_k) = d \times \left\{ \sum_{i=1, i \ne k}^{n} PR(LA_i) \times V(LA_i, k) \right\} + (1-d) \times \frac{W_i}{\sum_{P_j \in WS} W_j} \qquad (16)$$

In this equation, LA_k show the learning automata for page *k*, and *V* (LA_i , *k*) is the value of action probability *k* in LA_i automata. In this fact, this probability value shows the weight of hyperlinks between web pages. Parameter *n* shows the number of web pages and *d* is damping factor.

4. EXPRIMENTAL EVALUATION

4.1 The Metrics of Ranks Evaluation

In our experiments we used two publicly available data sets. The first one includes the page visits of users who visited the NASA data set [19]. Its information has been recorded since 1998/8/1 up to 1998/8/30 has been used in this simulation. In this log file the number of users sessions have been 45000 and the number of web pages have been 863.

We used two measures for comparing two top-n rankings r_1 and r_2. The first measure denoted as *Osim* (r_1, r_2) [7] indicates that the degree of overlap between the top-n pages of two sets *A* and *B* (each of size *n*) to be:

$$Osim(r_1, r_2) = \frac{|A \cap B|}{n} \qquad (17)$$

In our comparisons we created top-n rankings. The second Ksim (r_1, r_2) is based on Kendall's distance measure [8] and indicates the degree to which the relative orderings of two top-n lists are in agreement and is defined as:

$$Ksim(r_1, r_2) = \frac{\left|(u,v) : r_1', r_2' \ havesameordering\ \ of(u,v), u \ne v\right|}{\left(|A \cap B||A \cap B| - 1\right)} \qquad (18)$$

Where r_1' is an extension of r_1, containing all elements included in r_2 but not r_1 at the end of the list (r_2' is defined analogously) [7]. We performed the experiments across four methods proposed method (FPR-DLA), UPR, LA base method, Page Rate and compared the average Osim and Ksim similarity measure for the top-n ranking generated by four methods. The diagrams presented here, show the average Osim and Ksim similarities over four methods. Figure 5 and 6 depicts the average Osim and Ksim values for the top-10 and top-50 rankings generated for the NASA data set. In the first case (top-10) we observe that the Osim in the proposed method is higher than other methods (UPR, LA base method, Page Rate) and is more than 80%. The Ksim, on the other hand, is more than 90% in the case of the proposed method and is higher than other methods. In the second case (top-50) we observe that the Osim in the proposed method is higher than other methods (UPR, LA base method, Page Rate) and is around 85%. The Ksim for three ranking methods (Proposed method, UPR, Page Rate) returned more than 90% whereas, it is reaches around 68% for the LA base method.

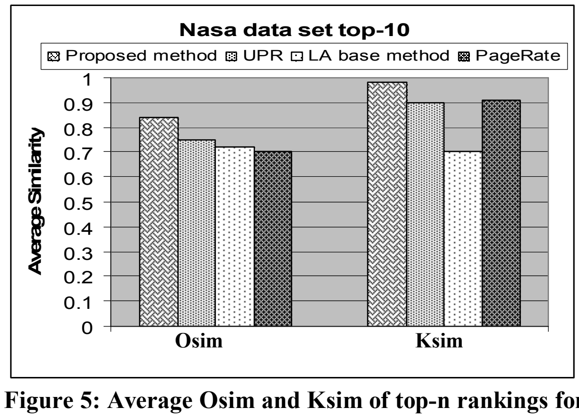

Figure 5: Average Osim and Ksim of top-n rankings for NASA data set

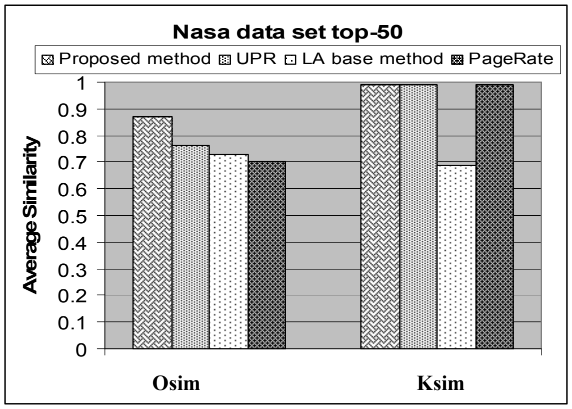

Figure 6: Average Osim and Ksim of top-n rankings for NASA data set

5. CONCLUSION - FUTURE WORK

In this research, we have proposed a new algorithm based on DLA and fuzzy for ranking pages in web. The proposed algorithm has considered main parameters in calculating web pages ranking. The parameters which can be fuzzy variables are time duration on a web page and the importance of web pages. The results of simulation show that proposed algorithm has considerable efficiency in determining the ranking of web pages. In the future we will attempt to use proposed algorithm in other kind of learning automata such as learning automata with variable actions. It is useful for dynamic web sites that pages or links change dynamically. Also we will attempt to automatically derive the membership functions based on learning automata.

6. REFERENCES

[1] Anari, B., Meybodi, M. R., and Anari, Z. *A New Method based on Distributed Learning Automata for Page Ranking in Web.* Proceedings of the 13th Annual International CSI Computer Conference of Iran, Kish Island, Iran, (March 2008).

[2] Anari, B., and Meybodi, M. R. *A New Method based on Distributed Learning Automata for Determining Web Documents Structure.* Proceedings of the 12th Annual International CSI Computer Conference of Iran, CSICC2007, (Tehran, Iran, Feb 20- 02, 2007) 2276-2281.

[3] Beygi, H., and Meybodi, M.R. *Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problem.* International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, World Scientific Publishing Company, 2005.

[4] Brin, S., and Page, L. *The Anatomy of a Large-scale Hyper Textual Web Search Engine*. Computer Networks and ISDN System, 30(1-7), 1998, pp. 107-117.

[5] Chakrabarti, S., Dom, B. E., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, A. D., and Kleinberg, J. *Mining the Web Link Structure*. Computer, 1999, 32(8): pp. 60-67.

[6] Eirinaki, M., and Vazirgiannis, M. *UPR: Usage-based Page Ranking for Web Personalization*. Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05), 2006.

[7] Haveliwal, T. *Topic-Sensitive Page Rank*. Proceedings of the WWW2002 Conference, Hawaii. 2002.

[8] Kendal, M., and Gibbons, J. D. *Rank Correlation Methods*. Oxford University Press, 1990.

[9] Lakshmivarahan, S. *Learning Algorithms: Theory and Applications*. New York: Springer-Verlag, 1981.

[10] Liu, B., and Liu, Y. *Expected Value of Fuzzy Variable and Fuzzy Expected Value Models*. IEEE Transactions on Fuzzy Systems. 2002, pp. 445- 450.

[11] Meybodi, M. R., and Lakshmivarahan, S. *On a Class of Learning Algorithms which have Symmetric Behavior under Success and Failure*. Lecture Notes in Statistics, Berlin: SpringerVerlag, 1984, pp. 145-155.

[12] Motwani, R., and Raghavan, P. *Ranomized Algorithms*. Cambrig University Press, United Kingdom, 1995.

[13] Narendra, K. S. and Thathachar, M .A. L. *Learning Automata: An Introduction*. Prentice Hall, 1989.

[14] Saati, S., and Meybodi, M. R. *Document Ranking using Distributed Learning Automata*. Proceedings of 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab, Tehran, Iran, May 24-26, 2006.

[15] Thathachar, M. A. L., Bhaskar, B., and Harita, R. *Learning Automata with Changing Number of Actions*. IEEE Transaction on System, Man and Cybernetice, Vol. SMC-17, 6(Nov. 1987).

[16] Wang, X., and Ha, M. *Note on Maxmin μ/E Estimation*. Fuzzy Sets and Systems. 94(1998), pp. 71-75.

[17] Wu, R., Tang, W., and Zhao, R. *Web Mining of Preferred Traversal Patterns in Fuzzy Environments*. Springer-Verlag Berlin Heidelberg, LNAI 3642, 2005, pp. 456-465.

[18] Zadeh, L. A. *Fuzzy Sets*. Information and control, 8(1965): pp. 338-353.

[19] http://ita.ee.lbl.gov/html/traces.html.