on-line preferential knowledge acquisition. Topics for future research include the following.

1) Use of this approach to conflict resolution in the context of an actual expert system in order to test its applicability. A system currently under development that will incorporate our approach to conflict resolution in the context of computer aided design is described in [11].
2) Generalization of the approach to consider preference-guided search in other search-related problems, more general functional forms of the utility structure, and the inclusion of risk in rule selection.
3) Examination of other approaches to blending production systems with formal reasoning techniques.

References

[1] A. Barr and E. A. Feigenbaum, Eds, *The Handbook of Artificial Intelligence*. Los Altos, CA: Kaufmann. 1981, vol. 1.

[2] R. Davis and J. J. King "An overview of production systems," *Machine Intelligence*, vol. 8, pp. 300–332, 1977.

[3] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*, Reading, MA: Addison-Wesley, 1983.

[4] J. K. Kastner and S. J. Hong, "A review of expert systems," *European J. Opns. Res.*, vol. 18, pp. 285–292, 1984.

[5] R. L. Keeney and M. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley, 1976.

[6] P. E. Lehner, M. A. Probus, and M. L. Donnell, "Building decision aids: exploiting the synergy between decision analysis and artificial intelligence," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 469–474, 1985.

[7] J. McDermott and C. Forgy, "Production systems conflict resolution strategies," in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth, Ed. New York: Academic, pp. 177–199. 1978.

[8] E. Rich, *Artificial Intelligence*. McGraw-Hill, New York: 1983.

[9] ——, "Users are individuals: Individualizing user models," *Int. J. Man-Machine Studies*, vol. 18, pp. 199–214, 1983.

[10] M. D. Rychener and A. Newell, "An instructable production system: Basic design issues," in *Pattern-Directed Inference Systems*, D. A. Waterman, and F. Hayes-Roth, Eds. New York: Academic Press, 1978.

[11] E. A. Sykes and C. C. White, "Specifications of a knowledge system for packet-switched data network topological design," IEEE Computer Society Symposium on Expert Systems in Government, October 1985.

[12] A. Whinston, "Knowledge-based systems, Abstract of a plenary address," ORSA/TIMS Bulletin, p. 1, Nov. 1985.

[13] C. C. White, A. P. Sage, and S. Dozono, "A model of multiattribute decisionmaking and trade-off weight determination under uncertainty," *IEEE Trans. Systems, Man, and Cybernetics, vol. SMC-14, pp.* 223–229, 1984.

Fuzzy Information Retrieval Based on a Fuzzy Pseudothesaurus

SADAAKI MIYAMOTO AND K. NAKAYAMA

***Abstract*—A fuzzy bibliographic information retrieval based on a fuzzy thesaurus or on a fuzzy pseudothesaurus is described. A fuzzy thesaurus consists of two fuzzy relations defined on a set of keywords for the bibliography. The fuzzy relations are generated based on a fuzzy set model, which describes association of a keyword to its concepts. If the set of concepts in the fuzzy set model is replaced by the set of documents, the fuzzy relations are called a pseudothesaurus, which is automatically generated by using occurrence frequencies of the keywords in the set of documents. The fuzzy retrieval uses two fuzzy relations in addition, that is, a fuzzy indexing and a fuzzy inverted file: the latter is the inverse relation of the former. They are, however, related to different algorithms for indexing and retrieval, respectively. An algorithm of ordering retrieved documents according to the values of the fuzzy thesaurus is proposed. This method of the ordering is optimal in the sense that one can obtain documents of maximum relevance in a fixed time interval. An example of the fuzzy retrieval is shown on a prototype database. This method shows one of the simplest way to realize fuzzy retrieval in practical database systems.**

Manuscript received March 8, 1985; revised October 14, 1985.

The authors are with the Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-Mura, Ibaraki 305, Japan.

IEEE Log Number 8406789.

I. Introduction

The concept of fuzziness plays an important role in information retrieval, since a user of a retrieval system has in general vague notions of his subjects: he tries some keyword that comes to his mind, then he evaluates the result of his retrieval by his fuzzy criteria. An example of the fuzziness is exhibited also in the thesaurus for information retrieval, where a set of keywords are shown with several related terms that may be more appropriate than the given keyword.

Fuzzy thesauri and fuzzy retrieval have been proposed by several authors [1], [2], [3]. However, algorithms in these studies are complicated, and it is difficult to realize them on a practical information retrieval system for large-scale databases.

In the present paper we show a simple way to perform a fuzzy retrieval. The main points here are as follows.

1) The fuzzy retrieval is based on a fuzzy thesaurus or a fuzzy pseudothesaurus [4], [5], which means that the fuzzy retrieval is easily implemented on existing systems of bibliographic information retrieval.
2) The fuzzy retrieval uses an ordering of the retrieved documents according to their fuzzy relevance. Therefore the more relevant documents are displayed to a user earlier than the less relevant ones.

A fuzzy pseudothesaurus can be easily generated based on a fuzzy set model [4]. Then a sorting algorithm gives the ordering of the retrieved documents without severe restriction to the amount of bibliographic data.

A problem in a fuzzy retrieval is that too many documents are found in general by one trial of the retrieval. The method of the ordering and the fuzzy thesaurus overcome this problem: the former gives more important documents prior to less important ones so that one can safely neglect those coming later, and the latter controls the amount of retrieved documents by managing the size of the thesaurus file.

II. Fuzzy Retrieval Based on a Fuzzy Pseudothesaurus

A. Fuzzy Pseudothesaurus

With an essential modification, we briefly review the method of generating a fuzzy pseudothesaurus based on a fuzzy set model which has been proposed by the authors [4], [5].

Let $W = \{w_1, w_2, \cdots, w_m\}$ be the set of keywords, and let $C = \{c_1, c_2, \cdots, c_n\}$ be the set of various concepts. A function $h\colon W \to [0,1]^C$ is a mapping from each word w_i to its associated fuzzy set of concepts $h(w_i)$. We assume also an additive measure M defined on fuzzy subsets of C. (In the sequel M is taken to be a measure of counting numbers of elements: if $h(w_i) = h_{ij}$ on c_j, then $M(h(w_i)) = \sum_j h_{ij}$.) Then the following two fuzzy relations are introduced.

1) Proximity Relation: Related terms RT are represented as

$$s(w_i, w_j) = \frac{M(h(w_i) \cap h(w_j))}{M(h(w_i) \cup h(w_j))}$$

where w_i and w_j are fuzzy related terms with value s.

2) Inclusion Relation: Narrower terms NT are represented as

$$t(w_i, w_j) = \frac{M(h(w_i) \cap h(w_j))}{M(h(w_i))}$$

where w_i is a fuzzy narrower term of w_j with value t.

Remark: Other choices of the proximity and inclusion relations are possible. For example

$$t(w_i, w_j) = \frac{M(h(w_i) \cap h(w_j))}{M(h(w_i))}$$

$$s(w_i, w_j) = \min\left[t(w_i, w_j), t(w_j, w_i)\right].$$

These two relations are equivalent to the measure suggested by Salton [6] in his method of generating thesauri.

In the authors' foregoing studies α-cuts on s and t have been introduced to have a thesaurus-like structure. Here, however, the fuzzy relations themselves are considered to be a fuzzy thesaurus.

Remark: The point of view that a fuzzy relation between a pair of keywords is regarded as a fuzzy thesaurus has been discussed by Reisinger [1] and Radecki [2].

In a practical situation a fuzzy pseudothesaurus instead of the fuzzy thesaurus is generated by using co-occurrence relations of the keywords. In case of the pseudothesaurus, the set of concepts are replaced by the set of various articles or bibliographic citations [5]. Moreover the map h is derived from the matrix (h_{ij}) of the co-occurrences, where h_{ij} is the frequency that the keyword w_i occurs in the article c_j. Then the two relations are calculated as

$$s(w_i, w_j) = \frac{\sum_k \min(h_{ik}, h_{jk})}{\sum_k \max(h_{ik}, h_{jk})}$$

$$t(w_i, w_j) = \frac{\sum_k \min(h_{ik}, h_{jk})}{\sum_k h_{ik}}$$

where we assume that $M(h(w_i)) = \sum_k h_{ik}$. An algorithm has been given [4] to have a thesaurus-like structure on great numbers of keywords and articles. The algorithm given in [4] is applicable to the fuzzy relations here with no essential modification, therefore we omit the details.

B. Fuzzy Indexing and Fuzzy Inverted File

Assume that W is the set of keywords as is described above and D is the set of articles. Let T: $D \to 2^W$ be a mapping of indexing from an article to its associated keywords and U: $W \to 2^D$ be the inverse mapping (inverted file in the implementation) from a keyword to the corresponding articles. In other words, for $x \in D$ and $w \in W$, $w \in T(x)$, if and only if $x \in U(w)$. A fuzzy relation f on $W \times W$ is assumed to be given. We assume that $f(w_i, w_j) = s(w_i, w_j)$ or $f(w_i, w_j) = t(w_i, w_j)$. In the argument below a reasonable assumption is made that the number of elements in D is far greater than the number of elements in W, and that the average number of w_i's that satisfy $f(w_i, w_j) \neq 0$ for an arbitrarily given w_j is not very large.

Remark: Sometimes the number of pairs (w_i, w_j) such that $f(w_i, w_j) \neq 0$ becomes large in generating a pseudothesaurus. In such a case low values of $f(w_i, w_j)$ should be set to zero to keep the thesaurus file to a reasonable size, which does not alter the argument here essentially.

Now, the indexing T in an ordinary sense and the inverted file U are generalized to a fuzzy indexing T_f and a fuzzy inverted file U_f by using the fuzzy relation f. For each $x \in D$, $T_f(x)$ should give a fuzzy set on W. In other words T_f is a fuzzy relation on $W \times D$. Using f we define

$$T_f(w, x) = \sup_{v \in T(x)} f(v, w), \qquad w \in W, \quad x \in D. \tag{1}$$

Namely, the grade of relevance of a keyword to an article is determined as the maximum value in the fuzzy thesaurus through the terms that exist in the original nonfuzzy index. Note that $T_f(v, x) = 1$, for $v \in T(x)$, since $f(v, v) = 1$. Next, for a given keyword $w \in W$ the relation U_f should give a fuzzy set on D. That is, U_f is a fuzzy relation on $D \times W$. It is natural to define U_f as the inverse relation of T_f, namely

$$U_f(x, w) = T_f(w, x).$$

It is necessary to distinguish the two quantities T_f and U_f, since they lead to different algorithms. As the fuzzy indexing T_f is derived from T, U_f satisfies

$$U_f(x, w) = \sup_{\substack{x \in U(v) \\ \text{all } v \in W}} f(v, w). \tag{2}$$

To prove the above relation, note that the inequalities

$$\sup_{v \in T(x)} f(v, w) \geqslant f(v', w)$$

for any $v' \in W$ such that $x \in U(v')$, and

$$\sup_{\substack{x \in U(v') \\ \text{all } v' \in W}} f(v', w) \geqslant f(v'', w),$$

for any $v'' \in T(x)$ hold, since $x \in U(v')$ means $v' \in T(x)$, and $v'' \in T(x)$ implies $x \in U(v'')$. Taking the suprema of the right hand sides of the above inequalities, we have

$$U_f(x, w) = T_f(w, x) = \sup_{v \in T(x)} f(v, w) \geqslant \sup_{\substack{x \in U(v') \\ \text{all } v' \in W}} f(v', w) \geqslant$$
$$\geqslant \sup_{v'' \in T(x)} f(v'', w) = T_f(w, x) = U_f(x, w).$$

The relations (1) and (2) are used to have algorithms for a fuzzy indexing and the fuzzy retrieval, respectively.

III. Ordering Algorithms

Although conventional (nonfuzzy) thesauri have been widely used, they have an inherent disadvantage. A retrieval through a conventional thesaurus means that articles having any of the associated terms with a given keyword are retrieved. Therefore great many articles are found by one thesaurus retrieval in general. Moreover, no measure of their relevance are given in the retrieved set of articles. Therefore one can not know which of the retrieved documents are more important than others.

On the other hand, the apparent advantage of a fuzzy thesaurus is that it has the measure of relevance for the associated keywords. Thus the fuzzy thesaurus overcomes the above problem in two ways: 1) articles of higher values of relevance can be processed prior to articles of lower values of relevance, and 2) articles of low values of relevance can be ignored by specifying an α-cut [7].

For these two purposes we propose an ordering algorithm that displays the retrieved articles, which are ordered according to their values of relevance. Namely, documents of higher values of relevance are printed earlier than those of lower values.

Obviously this method of the ordering is optimal in two senses. First, one wishes to know more important informations earlier than less important ones. Second, if one breaks the output of the retrieved documents which are ordered, the rest of the documents which are ignored has lower values of relevance than the documents already printed.

A user of the fuzzy retrieval may or may not specify α-cuts on the retrieved set. Even when he does not use α-cuts and requires all the retrieved articles, the ordering method is optimal in the first sense. When he uses α-cuts, he must determine a threshold for the relevance measure. In the ordering method, however, explicit specification of a threshold for an α-cut is unnecessary. A user can break his output at any time when he feels appropriate, then the documents already printed forms the α-cut. In other words, we can use the number of printed articles as a more natural parameter for α-cuts than threshold on the relevance measure. Note that in the commercial databases amount of charges is decided by the number of printed documents.

Thus the advantage of this method can be stated as follows. If utility of an article at the retrieval is proportional or monotonically dependent on the value of the fuzzy relation, the method of the ordering is optimal in the sense that one can obtain articles of maximum utility in a fixed time interval.

Algorithms for this ordering use a routine device of the sorting. In the following description of the algorithms, $(\boldsymbol{a}, \boldsymbol{b})$, for example, denotes a record with fields $\boldsymbol{a}$ and $\boldsymbol{b}$: $\{(\boldsymbol{a}, \boldsymbol{b})\}$ is a sequential file of records $(\boldsymbol{a}, \boldsymbol{b})$'s. Sorting of $\{(\boldsymbol{a}, \boldsymbol{b})\}$ by the key $\boldsymbol{b}, \boldsymbol{a}$ means that the first key for the sorting is $\boldsymbol{b}$ and the second key is $\boldsymbol{a}$.

In Algorithm 1 of the fuzzy retrieval two sortings are used. The second sorting is for the above ordering, whereas the first sorting is to find values of U_f for each instance (occurrence) of articles. Since an instance $\hat{x}$ of articles has several keywords which may be related, $\hat{x}$ may occur several times with different values of relevance in the retrieval through the fuzzy thesaurus. If $\hat{x}$ has the values $f_1, f_2, \cdots, f_p$, we have records $(\hat{x}, f_1)$, $(\hat{x}, f_2), \cdots, (\hat{x}, f_p)$ in the retrieved set by the following algorithm. When the file of the retrieved articles is sorted in descending order by the first key x of article and the second key f of the fuzzy relation, then the records for the same instance $\hat{x}$ are accumulated and form a subsequence $(\hat{x}, f_q), \cdots, (\hat{x}, f_r)$ in the sorted sequence of the retrieved records. Moreover f_q is the maximum value of $f_1, \cdots, f_p$. It is clear that the top $(\hat{x}, f_q)$ of the subsequence represents the value of U_f, and the rest of the subsequence is unnecessary. Therefore the rest of the subsequence should be deleted and the second sorting gives the required ordering of the articles.

Algorithm 1 of the fuzzy retrieval is based on (2).

Algorithm 1 (fuzzy retrieval):

```
FOR a given $w$
    FOR all $v$ such that $f(v, w) \neq 0$
        FOR all $x \in U(v)$
            MAKE record $(x, f(v, w))$;
    SORT $\{(x, f(v, w))\}$ in descending order by the key $x, f(v, w)$;
    SCAN the sorted file $\{(x, f(v, w))\}$, for any subsequence of records that represents the same instance of $x$, take the first record and delete the rest of the records in that subsequence;
    SORT the resulting $\{(x, f(v, w))\}$ in descending order by the key $f(v, w)$;
    /* Note that any instance of $x$ occurs at most once in the records at this stage. */
    OUTPUT the documents $\{x\}$ according to the sorted order.
```

Remark: Practically, the second sorting should be postponed until when output commands like DISPLAY, TYPE, or PRINT are issued. The reason is that operations such as AND, OR are expected on the retrieved sets of articles; these operations are more easily performed on the records which are ordered according to the key x before the second sorting than on the records after the second sorting. As usual, these set operations in the fuzzy case result in minimum or maximum of the values of the fuzzy relation on each articles. That is, with the fuzzy retrieval with keywords w_i and w_j

$$\min\left(U_f(x, w_i), U_f(x, w_j)\right)$$

and with w_i *or* w_j

$$\max\left(U_f(x, w_i), U_f(x, w_j)\right).$$

Remark: As another usage of a thesaurus, a user may select some of the associated terms according to his order of preference and search articles having any one of these selected terms. Even in this case the associated terms with a given keyword should be displayed according to the order of relevance given by the fuzzy relation, since terms of higher values of relevance are more frequently selected than terms of lower values. On the other hand, if a user specify the selected keywords according to his order of preference for the fuzzy retrieval, the previous algorithm should be replaced by another algorithm that displays articles according to the order of preference, since in this case the order of preference specified by the user should be considered prior to the order given by the fuzzy relation.

On the other hand, the relation (1) for T_f provides a fuzzy indexing algorithm.

Algorithm 2 (fuzzy indexing):

```
FOR a given $x$
    FOR all $v \in T(x)$
        FOR all $w$ such that $f(v, w) \neq 0$
            MAKE record $(w, f(v, w))$;
    SORT $\{(w, f(v, w))\}$ in descending order by the key $w, f(v, w)$;
    SCAN the sorted file $\{(w, f(v, w))\}$, for any subsequence of records that represents the same instance of $w$, take the first record and delete the rest of the records in that subsequence;
    SORT the resulting $\{(w, f(v, w))\}$ in descending order by the key $f(v, w)$;
    OUTPUT the keywords $\{w\}$ according to the sorted order.
```

Now we consider why the present algorithm is ready for the implementation on a practical information retrieval system. Our basic assumption is that a fuzzy retrieval should be implemented on a practical bibliographic retrieval system of an ordinary (non-fuzzy) type. For the purpose of comparison let us consider the following three alternatives of implementing the fuzzy retrieval.

1) Construct the fuzzy indexing T_f and U_f directly by some manual fashion. The fuzzy inverted file U_f is stored in the form of an ordinary inverted file (physically stored U_f) with values of the fuzzy relation.

2) Construct U_f by the fuzzy thesaurus or pseudothesaurus, but U_f is physical and stored like the ordinary inverted file.

3) Use the thesaurus file and the ordinary inverted file U_f to perform the fuzzy retrieval as in the present method (logical U_f).

For the comparison of resources required in these methods, note that m is the number of keywords and n is the number of articles. Moreover let p be the average number of keywords per one article and q be the average number of v's satisfying $f(v, w) \neq 0$ for one element $w \in W$. As was stated before, $m \ll n$, and p and q are far smaller than m. For example, $m =$ order(10^3) (m has the order of 10^3), $n =$ order(10^5), $p =$ order(10), and $q =$ order(10). Moreover note that an inverted file has m entries for the keyword search and np items to retain locations of the articles as the total amount of data in the inverted file.

First, manual construction of T_f and U_f by alternative 1) needs $mp \sim mpq$ times of the specification of the keywords and the values of the fuzzy relation, which is almost impossible on a large-scale database. Next, if we manage U_f directly in the form of the inverted file by alternatives 1) or 2), m entries and npq items are needed as the total amount of data. On the other hand, when we use the alternative 3), the number of entries is $2m$ as the sum of the number for U and that for the thesaurus file; the amount of data is $np + mq \ll 2np$, which is of quite a manageable size.

The number of keyword searches needed in one fuzzy retrieval is greater in the method 3) than in 1) or 2). Namely, q associated keywords which are obtained in the thesaurus file for a given keyword should be searched in U by alternative 3), whereas in alternatives 1) and 2) only one keyword is searched in the inverted file U_f.

It is well known that sorting of k records requires computation of order$(k \log k)$. In this case, however, the sorting is not a burden, since it is unnecessary to sort the retrieved documents themselves, but it is sufficient to sort only pointers to the documents and the values of the fuzzy relation. Therefore the amount of data to be sorted is comparatively small and an in-core version of the sorting program is sufficient for them. In an experimental sorting of data of sizes 10^3 and 10^4 on FACOM M380 computer at Science Information Processing Center, University of Tsukuba, the required CPU times are 18 ms and 447 ms, respectively. On the other hand, the number of retrieved articles by one fuzzy retrieval is at most order(10^3).

Considering these points, we conclude that alternative 3) is easier to realize on a practical retrieval system, although it requires a greater number of the keyword searches. The reason is that the huge size of the physical U_f in the form of the inverted file is a more serious problem, whereas the number of the keyword searches in alternative 3) is at most $q =$ order(10), if the size of the thesaurus file is adequately controlled.

Another advantage, perhaps a more important one, is that the method of alternative 3) is realized with no modification on the underlying retrieval system of an ordinary type on which the fuzzy retrieval is implemented. Note that a fuzzy retrieval will be expected in the midst of ordinary retrievals. In alternatives 1) and 2), two inverted files U_f and U should be used, since the use of U_f instead of U in an ordinary retrieval disimproves performance of the system. The method of alternative 3) uses the ordinary inverted file U alone, where the thesaurus file is used to have the logical U_f. Thus in the present method 3), the fuzzy retrieval is implemented on the full use of functions of an ordinary retrieval system with a comparatively small size of the thesaurus file.

IV. Example of the Fuzzy Retrieval

A prototype database and its pseudothesaurus were used to show how the fuzzy retrieval is performed. Here the database contains 189 articles of *IEEE Transactions on Automatic Control* in 1980. The database INSPEC was referred to have keywords for articles in the prototype database. Then the fuzzy pseudothesaurus was generated by C as the set of articles and (h_{ij}) as the matrix of occurrences of the keywords w_i's in the articles c_j's [4].

Fig. 1 shows an example of the fuzzy retrieval. The INSPECT command specified a keyword MAN-MACHINE SYSTEMS for the fuzzy retrieval by the pseudothesaurus. The term RT means that the fuzzy relation $s(w_i, w_j)$ for RT should be used. Then the program found two related keywords MAN-MACHINE SYSTEMS, and DECISION THEORY AND ANALYSIS with their values in the pseudothesaurus and retrieved the articles having these two keywords. The command DISPLAY shows the retrieved articles which are ordered according to the values of the relation U_f. First an article having MAN-MACHINE SYSTEMS itself is shown, then articles with DECISION THEORY AND ANALYSIS follow.

V. Conclusion

Although the fuzzy framework is appropriate for mathematical representation of human subjective processing and judgment of information, excessive introduction of subjectivity should be

```
? INSPECT FUZZY RT D 'MAN-MACHINE SYSTEMS

 MAN-MACHINE SYSTEMS                  RELATION  1.0000 ,     1 RECORDS FOUND
 DECISION THEORY AND ANALYSIS         RELATION  0.2000 ,     5 RECORDS FOUND
 5  RECORDS IN TOTAL

? DISPLAY

 TI AN INTERACTIVE RECTANGLE ELIMINATION METHOD FOR BIOBJECTIVE
 TI  DECISION MAKING
 D  MAN-MACHINE SYSTEMS;DECISION THEORY AND ANALYSIS
 JI IEEE TRANS. AUTOM. CONTROL (USA)
 AI PAYNE, A.N.;POLAK, E.
 YI JUNE 1980
 PG P0421-0431

 TI INCONSISTENCY OF THE AIC RULE FOR ESTIMATING THE ORDER OF A
 TI UTOREGRESSIVE MODELS
 D  TIME SERIES;DECISION THEORY AND ANALYSIS
 JI IEEE TRANS. AUTOM. CONTROL (USA)
 AI KASHYAP, R.L.
 YI OCT. 1980
 PG P0996-0998

 TI A DECENTRALIZED TEAM DECISION PROBLEM WITH AN EXPONENTIAL C
 TI OST CRITERION
 D  DECISION THEORY AND ANALYSIS;OPTIMAL CONTROL
 JI IEEE TRANS. AUTOM. CONTROL (USA)
 AI SPEYER, J.L.;MARCUS, S.;KRAINAK, J.
 YI OCT. 1980
 PG P0919-0924

 TI ON RESOLUTION AND EXPONENTIAL DISCRIMINATION BETWEEN GAUSSI
 TI AN STATIONARY VECTOR PROCESSES AND DYNAMIC MODELS
 D  RANDOM PROCESSES;DECISION THEORY AND ANALYSIS
 JI IEEE TRANS. AUTOM. CONTROL (USA)
 AI KAZAKOS, D.
 YI APRIL 1980
 PG P0294-0296

 TI SPECTRAL DISTANCE MEASURES BETWEEN GAUSSIAN PROCESSES
 D  FILTERING AND PREDICTION THEORY;RANDOM PROCESSES;SPECTRAL A
 D  NALYSIS;MATRIX ALGEBRA;DECISION THEORY AND ANALYSIS
 JI IEEE TRANS. AUTOM. CONTROL (USA)
 AI KAZAKOS, D.;PAPANTONI-KAZAKOS, P.
 YI OCT. 1980
 PG P0950-0959
```

Fig. 1. Example of the fuzzy retrieval, where underlined parts are inputs by the user. First command INSPECT indicates the use of fuzzy RT relation on the search of descriptor $D =$ MAN-MACHINE SYSTEMS. The DISPLAY command requires output of the retrieved records to the terminal.

avoided in a formulation of a problem. In this aspect the above method of information retrieval has the following two characteristics.

1) Subjective and direct determination of values for the fuzzy thesaurus is unnecessary, since a pseudothesaurus is available that is generated automatically based on the co-occurrence frequencies of the keywords.
2) Explicit specification of thresholds for α-cuts by a user is avoided owing to the ordering method.

Moreover, realization of the above fuzzy retrieval of documents is comparatively easy, due to the following reasons.

a) The sorting algorithm is not resource-consuming.
b) The retrieval with a pseudothesaurus is easily implemented on existing (nonfuzzy) database management systems.
c) Conventional thesauri themselves essentially have fuzzy characteristics and their fuzzy generalizations are not difficult.

In summary, the method described here shows one of the simplest way to realize fuzzy retrieval in a practical environment. Also, there is much room for further applications of the present method in knowledge information systems. For example, research of psychological association is promising.

References

[1] L. Reisinger, "On fuzzy thesauri," in *Proc. 1974 Symp. Comput. Stat.*, G. Bruckman *et al.*, Eds. Vienna: Physica-Verlag, 1974, pp. 119–127.
[2] T. Radecki, "Mathematical model of information retrieval system based on the concept of fuzzy thesaurus," *Inform. Processing Management*, vol. 12, pp. 313–318, 1976.
[3] ——, "Mathematical model of time-effective information retrieval system based on the theory of fuzzy sets," *Inform. Processing Management*, vol. 13, pp. 109–116, 1977.
[4] S. Miyamoto, T. Miyake, and K. Nakayama, "Generation of a pseudothesaurus for information retrieval based on cooccurrences and fuzzy set operations," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-13, no. 1, pp. 62–70, 1983.
[5] ——, "Structure generation on bibliographic databases with citations based on a fuzzy set model," in *Proc.* IFAC *Symp. Fuzzy Inform.*, 1983, pp. 225–230.
[6] "*The SMART Retrieval System, Experiments in Automatic Document Processing*, G. Salton, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1971.
[7] D. Dubois and H. Prade, *Fuzzy Sets and Systems, Theory and Applications*, New York: Academic, 1980.

Absorbing and Ergodic Discretized Two-Action Learning Automata

B. JOHN OOMMEN

***Abstract*—A learning automaton is a machine that interacts with a random environment and that simultaneously learns the optimal action that the environment offers to it. Learning automata with variable structure are considered. Such automata are completely defined by a set of probability updating rules. Contrary to all the variable-structure stochastic automata (VSSA) discussed in the literature, which update the probabilities in such a way that an action probability can take any real value in the interval [0, 1], the probability space is discretized so as to permit the action probability to assume one of a finite number of distinct values in [0, 1]. The discretized automaton is termed linear or nonlinear depending on whether the subintervals of [0, 1] are of equal length. It is proven that 1) discretized two-action linear reward–inaction automata are absorbing and ϵ-optimal in all environments; 2) discretized two-action linear inaction–penalty automata are ergodic and expedient in all environments; 3) discretized two-action linear inaction-penalty learning automata with artificially created absorbing barriers are ϵ-optimal in all random environments; and 4) there exist nonlinear discretized reward–inaction automata that are ϵ-optimal in all random environments. The maximum advantage gained by rendering any finite-state discretized automaton nonlinear has also been derived.**

Manuscript received May 26, 1985; revised September 27, 1985. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of this paper was presented at the 1985 International Conference on Systems, Man and Cybernetics, Tucson, Arizona.

The author is with the School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada.

IEEE Log Number 8406787.

I. Introduction

Learning automata have been extensively studied by researchers in the area of adaptive learning. The intention is to design a learning machine that interacts with an environment and that dynamically learns the optimal action which the environment offers. The literature on learning automata is extensive. We refer the reader to a review paper by Narendra and Thathachar [9] and a recent excellent book by Lakshmivarahan [3] for a review of the various families of learning automata. The latter reference also discusses in fair detail some of the applications of learning automata which include game playing [5], pattern recognition and hypothesis testing [9], priority assignment in a queueing system [7], and telephone routing [10], [11].

Broadly speaking, learning automata can be classified into two categories: fixed structure automata and variable-structure stochastic automata (VSSA). A fixed-structure automaton is one in which the transition and output functions are time-invariant. Examples of such automata are the Tsetlin, Krylov, and Krinsky automata [17], [18]. By far, most of the research in this area has involved the second category, namely, variable-structure stochastic automata. Automata is this category possess transition and output functions which evolve as the learning process proceeds. It can be shown that a VSSA is completely defined by a set of action probability updating functions [8], [9], [20].

VSSA are implemented using a random-number generator (RNG). The automaton decides on the action to be chosen based on an action probability distribution. Nearly all the VSSA discussed in the literature permit probabilities which can take any value in the interval [0, 1]. Hence the RNG required must theoretically possess infinite accuracy. In practice, however, the probabilities are rounded off to a certain number of decimal places depending on the architecture of the machine that is used to implement the automaton.

Learning automata can also be broadly classified in terms of their Markovian representations. Generally speaking, learning automata are either ergodic [10], [13]–[15], [19] or possess absorbing barriers [6], [9], [12]. Automata in the former class converge with a distribution that is independent of the initial distribution of the action probabilities. This feature is desirable when interacting with a nonstationary environment—for the automaton does not "lock" itself into choosing any one action. However, if the environment is stationary an automaton with an absorbing barrier is preferred. Various absolutely expedient schemes which ideally interact with such environments have been proposed in the literature [3], [6], [8], [9].

To minimize the requirements on the RNG *and to increase the speed of convergence* of the VSSA the concept of discretizing the probability space was recently introduced in the literature [12], [16]. As in the continuous case, a discrete VSSA is defined using a probability updating function. However, as opposed to the functions used to define continuous VSSA, discrete VSSA utilize functions that can only assume a *finite* number of values. These values divide the interval [0, 1] into a finite number of subintervals. If the subintervals are all of equal length the VSSA is said to be linear. Using these functions discrete VSSA can be designed—the learning being performed by updating the action probabilities in discrete steps.