CHAPTER 5

IMPLEMENTING CONTEXTUAL STRUCTURES FOR DATA MINING

In this chapter the formalisms defined in chapters 3 and 4 are explored computationally. The applications developed pertain to the broad field of *data mining*. This area of Information Sciences and Machine Learning is dedicated to building computational tools that can help us search and categorize the immense information resources available today. One of the most significant subareas of data mining research is that of knowledge discovery in databases. Particularly with the exponential growth of the Internet, it is becoming harder and harder to effectively search databases without being overwhelmed by a cascade of unrelated results to users' queries. Section 1 of this chapter presents a conversational knowledge discovery system for relational databases that uses evidence sets as categorization mechanisms. Its objective is the definition of a human-machine interface that can capture more efficiently the user's interests through an interactive question-answering process.

Another area of interest to data mining is that of the search of solutions for a problem in very large state spaces. Contextual genetic algorithms (CGA's) with fuzzy indirect encoding are used in two different problems. The first is the search of the appropriate set of weights for a very large Neural Network. The GA is used as a learning algorithm. It is a continuous variable problem. The second problem is the evolution of Cellular Automata rules to solve non-trivial tasks. It is a discrete variable problem. The fuzzy indirect encoding CGA of chapter 4 is shown to deal with such large problems efficiently.

1. Computing Categories in Relational Databases: Linguistic Categories as Consensual Selection of Dynamics[33]

The notion of uncertainty, is very relevant to any discussion of the modeling of linguistic/mental abilities. From Zadeh's [1971, 1975] approximate reasoning to probabilistic and even evidential reasoning [Schum, 1994], uncertainty is more and more recognized as a very important issue in cognitive science and artificial intelligence with respect to the problems of knowledge representation and the modeling of reasoning abilities [Shafer and Pearl, 1990]. Engineers of knowledge based systems can no longer be solely concerned with issues of linguistic or cognitive *representation*, they must describe "*reasoning*" procedures which enable an artificial system to answer queries. In many artificial intelligence systems, the choice of the next step in a reasoning procedure is based upon the measurement of the system's current uncertainty state [Nakamura and Iwai, 1982; Medina-Martins et al, 1992, 1994]. Now that a theory of evidential approximate reasoning and measures of uncertainty are defined for evidence sets (see chapter 3), we can extend fuzzy data-retrieval systems to an evidence set formulation.

1.1 Nakamura and Iwai's Fuzzy Information Retrieval System[34]

Nakamura and Iwai's [1982] data-retrieval system is based on a structure with two different kinds of objects: *Concepts* x_i (e.g. Keywords) and *Properties* p_i (e.g. data records like books). Each concept is associated with a number of properties which may be shared with other concepts. Based on the amount of properties shared with one another, a measure of similarity, s, can be constructed for concepts x_i and x_j:

[33] Some of the work presented in this section was first presented in Rocha[1991].

[34] Nakamura and Iwai's work has been expanded formally and in its interpretation from its original form.

$$s(x_i,x_j) = \frac{N(x_i \cap x_j)}{N(x_i \cup x_j)} = \frac{N(x_i \cap x_j)}{N(x_i) + N(x_j) - N(x_i \cap x_j)} \quad \textbf{(1)}$$

where:

- $N(x_i)$ represents the number of properties that directly qualify concept x_i.
- $N(x_i \cap x_j)$ represents the number of properties that directly qualify both x_i and x_j.
- $N(x_i \cup x_j)$ represents the number of properties that directly qualify either x_i or x_j.

The inverse of the similarity, s, creates a measure (semi-metric) of distance[35], d (figure 1):

$$d(x_i,x_j) = \frac{1}{s(x_i,x_j)} \quad \textbf{(2)}$$

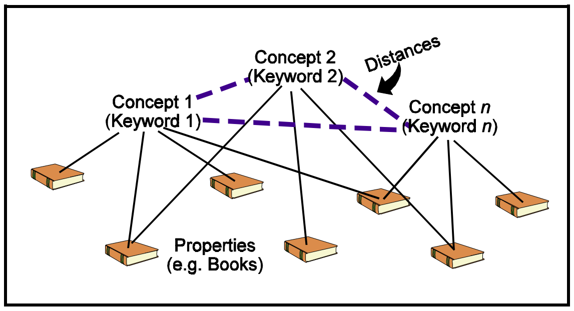


Figure 1: Structure of Nakamura and Iwai's database. A measure of distance is calculated between concepts according to the number of properties in common.

Naturally, some concepts will not share any properties between them. Several approaches can be pursued to calculate the distance between every concept in the network. Nakamura and Iwai propose the unfolding of the structure starting at any two concepts, and digging into deeper levels until common properties are reached. Instead of this, I use an algorithm that calculates the shortest path between two concepts. First, the distances between directly linked concepts are calculated using (2). After this, the shortest path is calculated between indirectly linked concepts. To speed up the process, the algorithm allows the search of indirect distances up to a certain level. The set of *n-reachable* concepts from concept x_i, is the set of concepts that have no more that n direct paths between them. Thus if x_i is directly linked to x_j which is turn directly linked to x_k, then x_k is a 2-reachable concept of x_i. If we set the algorithm to paths up to level n, all concepts that are only reachable in more than n direct paths from x_i will have their distance to x_i set to infinity. The larger the network of concepts and properties, the larger is the need to have a small n to avoid lengthy computations.

1.1.1 The Long-Term Networked Memory Structure: Semantic Closure

[35] This measure of distance calculated in a large network of nodes, is usually not a Euclidean metric because it does not observe the triangular inequality. In other words, the shortest distance between two nodes of the network might not be the direct path. This means that two nodes may be closer to each other when another node is associated with them. Such measures of distance are referred to as semi-metrics [Galvin and Shore, 1991]. For the purposes of the work here presented, we do not need to worry about the mathematical significance of semi-metrics.

After calculating the distances between concepts, a substructure can be recognized that is defined solely by the concepts and their relative distance: the *Knowledge Space X*. It is a computational structure whose purpose is to capture human knowledge by keeping a record of relationships and a measure of their similarity (or distance). It is not a connectionist memory system in the sense that concepts are not distributed (see chapter 3, sections 1 and 3) over the network but localized in recognizable nodes. However, in addition to localized memory it does possess a semi-metric space relating all knowledge as desired of connectionist engines [Clark, 1993]. This space is the lower-level structure of the system, the long-term memory banks of the relational database. It is from this semantic semi-metric that temporary prototype categorizations can be formed (see the discussion in section 3 of chapter 3).

The long-term relational network, simulates the uniqueness of the developmental history of cognitive systems. It captures the personal construction of relations or subjectivity. The system's relations are unique to it, and reflect the actual semantic constraints established by the set of data records (properties) it stores. Thus, the semantic semi-metric defined by (2), reflects the actual inter-significance of keywords (concepts) for the system itself. The same keyword in different databases will be related differently to other keywords, because the actual data records stored will be distinct. The data records a database stores are a result of its history of utilization and deployment of information by its users. In this sense, the long-term networked memory structure, defines a unique developmental artificial subjectivity established by the dynamics of information storage and usage. The inter-significance of concepts is specific to the system, which is thus semantically closed regarding conceptual inter-relations.

1.1.2 Short Term Categorization Processes

Given the underlying relations imbedded in the knowledge space, the system uses a question-answering process to capture the user's interests in terms of the system's own relational structure. In other words, the system constructs its own internal categories from its interaction with the user. The question-answering algorithm follows:

1. Users input an *initial concept of interest* (one of the keywords) $x_i \in X$.
2. The system creates a bell-shaped fuzzy membership function centered on x_i and affecting all its close neighbors (as calculated from the semantic semi-metric). This resulting fuzzy set of X represents a category that keeps the user's interests in terms of the system's own relations: *The learned category $A(x)$*.
3. The system calculates the fuzziness of the learned category.
4. Another concept $x_j \in X$ is selected. x_j is selected in order to potentially minimize the fuzziness of the learned category, that is, the most fuzzy concepts with the most fuzzy neighborhoods are selected.
5. The user is asked whether or not she is interested in x_j.
6. If the answer is "YES" another bell-shaped membership function is created over x_j, and a fuzzy union is performed with the previous state of the learned category.
7. If the answer is "NO" the inverse of the bell shaped membership function is created, and a fuzzy intersection is performed.
8. The system calculates the fuzziness of the learned category.
9. If the fuzziness of the learned category is smaller than half the maximum value attained previously, the system stops since the learned category is considered to have been successfully constructed. Otherwise computation goes back to step 4.

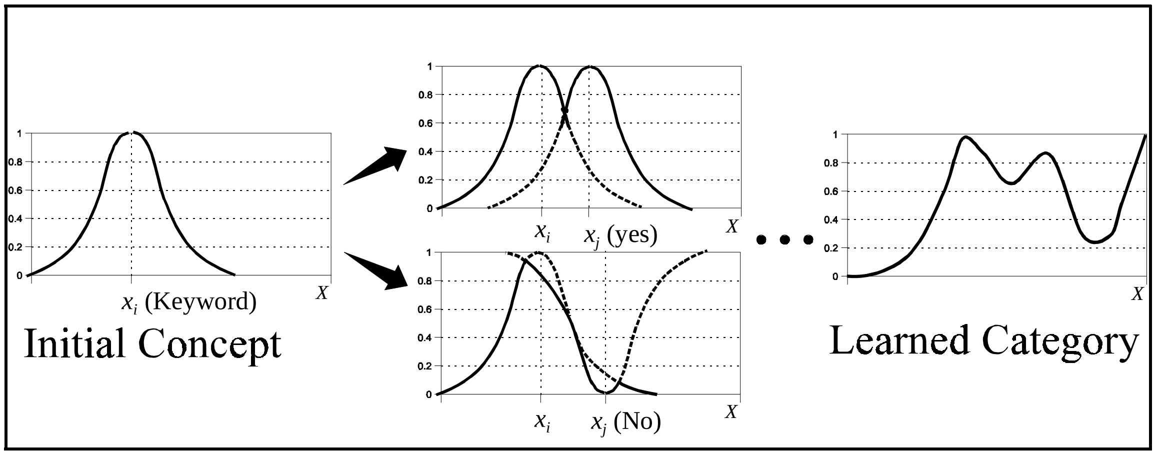


Figure 2: An initial concept x_i is selected and a bell-shaped fuzzy set is created over it in the Knowledge Space X. The system then selects another concept x_j which can potentially reduce the fuzziness of the learned category. If the user is interested in x_j another bell-shaped fuzzy set is created over x_j and a fuzzy union is performed. If the user is not interested the complement of the bell-shaped fuzzy set is created and a fuzzy intersection is performed. The process is repeated until the fuzziness of the learned category is acceptable.

The algorithm is depicted in figure 2. The bell-shaped function used is given by the following equation:

$$f_{YES,x_i}(x) = e^{(-\alpha|x-x_i|^2)} \tag{3}$$

The inverse function is defined as:

$$f_{NO,x_i}(x) = 1 - f_{YES,x_i}(x) \tag{4}$$

Both $f_{YES,xi}$ and $f_{NO,xi}$ define fuzzy sets of the knowledge space X. These functions spread the interest posited on a concept x_i to their neighbors based upon the semi-metric of X. In other words, the system assumes that if the user is (not) interested in x_i, it will also (not) be interested in its neighbors to a degree proportional to their distance to x_i. The parameter α in (3) controls the spread of this inference. A small (large) value will result in wide (narrow) spread of functions (3) and (4), which causes the system to learn the user's interest roughly (carefully). In fact, the value of α is adaptively changed in steps 6 and 7 of the algorithm, so that the current answer by the user does not contradict previous answers. Specifically, when the function for x_j is constructed, its width should not reduce or increase the value of membership previously computed for all prior x_i to which the user had given specific "YES" or "NO" answers. If x_i had received "NO" for an answer its membership should be 0, if x_j receives now an "YES" answer, the value of $f_{YES,xj}(x_i)$ should not be greater than 0, as this would mean contradicting (to a degree) a previous answer. Therefore, the parameter α is increased so that the spread decreases, and the answer to x_j does not affect the previous answer to x_i. In practice, another parameter ϵ is defined as the very small amount of membership value later answers are allowed to affect previous answers, it is usually set to 0.01.

The question-answering algorithm above by utilizing the traditional approximate reasoning operations of fuzzy union and intersection, causes the system to construct its short-term representation of a category holding the user's interests in terms of its own long-term relations. Such a simple algorithm, implements many of the, temporary, "on the hoof" [Clark, 1993] category constructions ideas as discussed in chapter 3. In particular, it is based on a long-term memory bank of relations that implements the system's own conceptual relationships (subjective construction). Prototype categories are then built using fuzzy sets which reflect such relational metrics and the directed interest of a user. Later in this chapter this system is extended to an evidence set framework in order to improve the categorical representation especially when it comes to context representation. Before that, the next subsections present other aspects of this information-retrieval system such as the actual retrieval of stored information, and more importantly, how the long-term relational network is changed according to the short-term categorical construction as it interacts with successive users.

1.1.3 Document Retrieval

After construction of the learned category $A(x)$, the system must return to the user the properties (data records such as books) relevant to this category. Notice that every property p_i defines a crisp subset of the knowledge space X whose elements are all the concepts to which p_i is connected. Let this subset be represented by $S_{p_i}(x) \equiv \{x \in X \mid x \hookrightarrow p_i\}$, where the symbol $\hookrightarrow$ represents a direct link between concept x and property p_i. Since each property defines a crisp subset of X, the similarity between this crisp subset and the fuzzy subset defined by the learned category should be a measure of the relevance of the property to the learned category. There are several ways to define this measure of similarity, for instance:

$$R_1(p_i) = \frac{\sum_{x \in X} A(x) \cap S_{p_i}(x)}{\sum_{x \in X} S_{p_i}(x)} \qquad \textbf{(5)}$$

and

$$R_2(p_i) = \frac{\sum_{x \in X} A(x) \cap S_{p_i}(x)}{\sum_{x \in X} A(x)} \qquad \textbf{(6)}$$

R_1 yields the fuzzy cardinality of the fuzzy set given by the intersection of the learned category $A(x)$ with $S_{p_i}(x)$ over the cardinality of the latter: it is an index of the subsethood of $S_{p_i}(x)$ in $A(x)$. The more $S_{p_i}(x)$ is a subset of $A(x)$, the more relevant p_i is. As long as $S_{p_i}(x)$ is included to a large extent in A, the property p_i will be considered very relevant, even if the learned category contains many more concepts than those included in $S_{p_i}(x)$. This way, R_1 gives high marks to all those properties who form subsets of the learned category and not necessarily those properties that qualify (are related to) the entire learned category as a whole. It is an index that emphasizes the independence of the concepts of the learned category. It should be used when the cardinality of $A(x)$ is large, otherwise, very few properties will exist that qualify a such large set of concepts.

R_2, on the other hand, yields the fuzzy cardinality of the fuzzy set given by the intersection of the learned category $A(x)$ with $S_{p_i}(x)$ over the cardinality of the former: it is an index of the subsethood of $A(x)$ in $S_{p_i}(x)$. The more $A(x)$ is a subset of $S_{p_i}(x)$, the more relevant p_i is. This way, R_2 gives high marks to all those properties who form subsets that include the learned category as a whole. It is an index that emphasizes the dependence of the concepts of the learned category. It should be used when the cardinality of $A(x)$ is small.

Since users usually tend to look for properties (data records) that both qualify the learned category as a whole or those that qualify its subsets, it is a good idea to define an index that is the average of R_1 and R_2:

$$R_3(p_i) = \frac{R_1(p_i) + R_2(p_i)}{2} \tag{7}$$

Thus, after the system finishes its construction of the learned category $A(x)$, the user can select one of the indices given by (5), (6), or (7) and a value between 0 and 1. High values will result on the system returning only those data records highly related to $A(x)$ according to the index chosen. Lower values will result in many more items being included in the list of returned data.

1.1.4 Adaptive Alteration of Long-Term Structure by Short-Term Categorization: Pragmatic Selection

Fuzzy logic and approximate reasoning are used in this system as a model of the temporary construction of prototypical categories. As such, it functions as a remarkable relational database retrieval system (computational details are discussed ahead). A logical step now is to provide it with a mechanism by virtue of which the long-term relational structure, the knowledge space, can be adaptively changed according to system's interactions with its users. Due to the properties (data records) it stores, the system may fail to construct strong relations between concepts (keywords) that its users find relevant. Therefore, the more certain concepts are associated with each other, by often being simultaneously included with a high degree of membership in learned categories, the more the distance between them should be reduced. An easy way to achieve this is to have the values of $N(x_i)$ and $N(x_i, x_j)$ in (1) adapted, after a learned category $A(x)$ is constructed, to:

$$N_{t+1}(x_i) = N_t(x_i) + A(x_i) \tag{8}$$

and

$$N_{t+1}(x_i,x_j) = N_t(x_i,x_j) + \min[A(x_i),A(x_j)] \tag{9}$$

respectively (t indicates the current state and t+1 the new state). If x_i and x_j tend to have high membership degrees simultaneously in the learned categories the system constructs, then their relative distance will tend to be reduced. This implements an adaption of the system to its users according to repeated interaction. Thus, the system though constructing its categories according to its own relational long-term memory, will adapt its constructions as it engages in question-answering conversations with its users. The direction of this adaptation leads the system's relational structure to match more and more the expectations of the community of users with whom the system interacts. In other words, its constructions are consensually selected by the community of users. If we regard the system's learned categories, implemented as fuzzy sets, as linguistic prototypical categories, which are the basis of the system's communication with its users, then such categories are precisely a mechanism to achieve the structural perturbation of its long-term relational memory in order to lead it to increasing adaptation to its environment. In other words, linguistic categories function as a system of consensual structural perturbation of networked dynamic memory banks, as anticipated in the discussion of selected self-organization and evolutionary constructivism (see chapter 2, section 2.4 and chapter 3 section 3). It implements the pragmatic selection dimension of evolving semiotics.

1.2 Contextual Expansion With Evidence Sets

In chapter 3, section 3, the construction of prototypical categories from connectionist machines was examined. One of the problems referred was that connectionist machines do not easily allow the treatment of several different contexts in the categories they construct. Each network is trained in a given problem area or context, whose concepts are stored in a distributed manner and for which a semantic semi-metric is established. But one network will not be able to relate a completely novel concept with its semantic semi-metric unless it is retrained in a larger context with expanded training patterns. Similarly, the system described above in 1.1. though not connectionist, observes some of the key characteristics of such systems. In particular, the inclusion of a new concept in its knowledge space, implies the recalculation of the entire set of distances for all concepts related to the new one up to a desired level (*n*-reachable concepts)[36].

Ideally, for both machines with connectionist or relational long-term memory, there should exist a mechanism that bridges information from networks trained in specific contexts. In other words, we would like a mechanism to categorize information stored in different long-term contextual memory banks. It is also desirable that the higher-level, short-term categorization processes may re-shape the long-term memory banks with the history of categorizations. The expansion of the system in 1.1 to an evidence framework achieves precisely that.

1.2.1 Distances from Several Relational Databases: The Extended Long-Term Memory Structure

Consider that instead of one single database (with concepts and properties) we have several databases which share at least a small subset of their concepts (keywords). Since they are linked to different properties, the similarity relation between concepts as defined by equation (1) is different from database to database, and so will the distance semi-metric given by (2). For instance, keyword "fuzzy logic" will be differently related to keyword "logic" in the library databases of a control engineering research laboratory or a philosophy academic department. We may desire however to search for materials in both of these contexts. Figure 3 depicts such structure with two different relational databases.

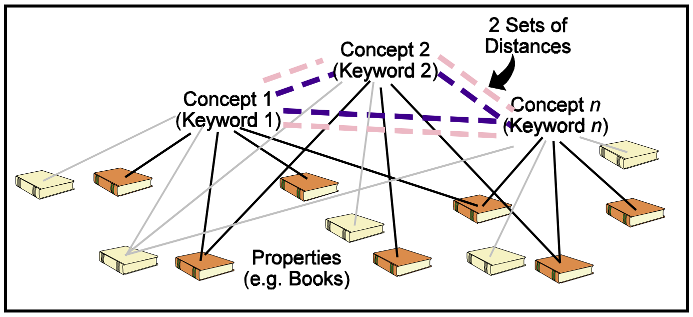


Figure 3: Structure of 2 distinct databases that possess at least some key-words in common. A different distance semi-metric is constructed for each one

The knowledge space X of this new structure is now the set of all concepts in the n_d included databases. Unlike the knowledge space of the system in section 1.1, that had only one distance semi-metric defined on set X, in this case the system has n_d different distance semi-metrics, d_k, associated with it. Each distance semi-metric is still built with equation (2) for some acceptable level of *n*-reachable concepts. But since each of the n_d databases has a different concept-property pattern of connectivity, each distance semi-metric d_k will be different. When a concept exists in one database and not on another, its distance to all other concepts of the second database is naturally set to infinity. If the databases derive from similar contexts,

[36] In general, it is still much easier to recalculate the distance of the n-reachable concepts to a new concept in a database, than to completely retrain a neural network.

naturally their distance semi-metrics will tend to be more similar. This distinction between the several contextual relational memory banks provides the system with intrinsic contextual conflict in evidence.

The ability to relate several relational databases is a very relevant endeavor for data-mining in this age of networked information. Users have available today a large number of networked client-server relational databases, which keep partially overlapping information gathered in different contexts. It is very useful to set up search applications that can relate the several sources of information to the interests of users. Such mechanisms need to effectively group several contextual sources of related long-term memory into temporarily meaningful categories.

1.2.2 Extended Short-Term Categorization

With their several intervals of membership weighted by the basic probability assignment of DST (see chapter 3), evidence sets can be used to quantify the relative interest of users in each of these information contexts. The extended evidence set approximate reasoning operations of intersection and union can be used to define a very similar question-answering process of prototypical category construction as the one in section 1.1. In the evidence set version, contexts can be explicitly accounted for. In addition, more forms of uncertainty are taken into account in the question-answering algorithm.

The system starts by presenting the user with the several networked databases available (see specific details of implementation below), who has to probabilistically grade them. That is, weights must be assigned to each database which must add to one (in order to build basic probability assignment functions). The selected databases define the several contexts which the system uses to construct its categories. Once this is defined, from the point of view of the user of the system, the question-answering interface is pretty much the same one as the system in section 1.1. Internally, however, the system is quite different, as the fuzzy set formulation is extended to evidence sets.

The main difference lies in the construction of evidence set membership functions that extend the fuzzy membership functions (3) and (4). Since the knowledge space *X* is now defined by n_d distance semi-metrics d_k, the shape of these functions cannot be a single bell-shaped function. It should somehow reflect the richness of different contexts (the n_d databases). Several approaches can be used to define these evidence set membership functions. The scheme I follow here starts by building bell-shaped fuzzy membership functions (equations 3 and 4) for each of the n_d distance semi-metrics d_k of *X*. Thus we obtain n_d different fuzzy subsets of *X* defined by fuzzy membership functions for "YES" or "NO" responses given to concept x_i as follows:

$$f^{d_k}_{YES,x_i}(x) = e^{\left(-\alpha|x-x_i|^{1/2}_{d_k}\right)},\ k = 1 \ldots n_d,\ x_i \in X \qquad \textbf{(10)}$$

and

$$f^{d_k}_{NO,x_i}(x) = 1 - f^{d_k}_{YES,x_i}(x) \qquad \textbf{(11)}$$

The next step is the construction of IVFS from these n_d fuzzy sets, using Turksen's DNF⊆CNF combinations (see chapter 3, section 3.2). All pairs of the n_d fuzzy sets given by either (10) or (11) for the "YES" or "NO" case respectively are combined to obtain $(n_d^2 - n_d)/2$ IVFS for the union combination and $(n_d^2 - n_d)/2$ for the intersection combination. Each pair of fuzzy sets is combined with the CNF and DNF forms of disjunction (union) and conjunction (intersection) in order to form two IVFS whose intervals of membership are defined by the CNF⊆DNF bounds for the standard union and intersection. Thus, from n_d fuzzy sets we obtain $n_d^2 - n_d$ IVFS. Since the "YES" and "NO" functions are combined in exactly the same way, in the following I define the IVFS combination for a series of n_d fuzzy set membership functions that can refer to either "YES" or "NO". Formally, a pair of fuzzy set membership functions (for semi-metrics d_k and d_l) is combined to obtain two IVFS:

$$IVf_{x_i}^{d_k \cup d_l}(x) = \left[f_{x_i}^{d_k}(x) \underset{DNF}{\cup} f_{x_i}^{d_l}(x) ;\; f_{x_i}^{d_k}(x) \underset{CNF}{\cup} f_{x_i}^{d_l}(x) \right] \qquad \textbf{(12)}$$

for union, and

$$IVf_{x_i}^{d_k \cap d_l}(x) = \left[f_{x_i}^{d_k}(x) \underset{DNF}{\cap} f_{x_i}^{d_l}(x) ;\; f_{x_i}^{d_k}(x) \underset{CNF}{\cap} f_{x_i}^{d_l}(x) \right] \qquad \textbf{(13)}$$

for intersection, where for two fuzzy sets $A(x)$, $B(x)$ the following definitions apply (the over line denotes set complement):

$$\begin{aligned} A \underset{DNF}{\cap} B &= A \cap B \\ A \underset{CNF}{\cap} B &= (A \cup B) \cap (A \cup \overline{B}) \cap (\overline{A} \cup B) \\ A \underset{DNF}{\cup} B &= (A \cap B) \cup (A \cap \overline{B}) \cup (\overline{A} \cap B) \\ A \underset{CNF}{\cup} B &= A \cup B \end{aligned}$$

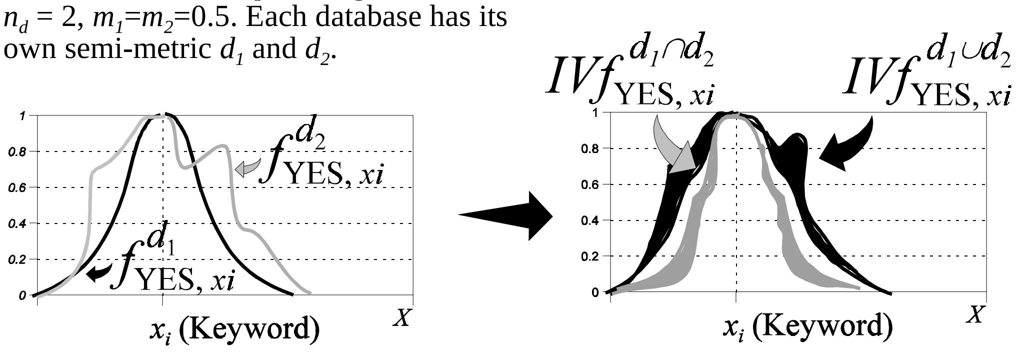
The final step is the construction of an evidence set from the $n_d^2 - n_d$ IVFS obtained from (12) and (13). At the onset, the user specifies the relative weight of the n_d databases utilized, m_k, which form a probability restriction since the sum of all m_k for k=1...n_d must equal one. When two fuzzy sets $f^{d_k}(x)$ and $f^{d_l}(x)$, with relative weights m_k and m_l respectively, are combined with (12) and (13) to obtain two IVFS, the total weight ascribed to this pair of IVFS is $(m_k + m_l)/(n_d - 1)$, and half of this quantity to each IVFS, which guarantees that the several IVFS are weighted by a probability restriction.

Thus, if we have n_d databases, with probability weight m_k (k=1...n_d), the evidence set membership function for an answer "YES" to concept x_i of knowledge space X, is given by:

$$ESf_{YES,x_i}(x) = \left\{ \left\langle IVf_{YES,x_i}^{d_k \cup d_l}(x),\ \frac{m_k + m_l}{2(n_d - 1)} \right\rangle, \left\langle IVf_{YES,x_i}^{d_k \cap d_l}(x),\ \frac{m_k + m_l}{2(n_d - 1)} \right\rangle \right\},\ k,l = 1 \ldots n_d \qquad \textbf{(14)}$$

that is, the evidence set has $n_d^2 - n_d$ focal intervals constructed from (12) and (13), weighted as described above. When several focal intervals coincide, their weights are summed and only one interval is acknowledged. The procedure for the "NO" evidence set is equivalent. Figure 4 depicts the construction of the "YES" evidence set membership function for a case of n_d=2.

2 Databases with equal weight:
$n_d = 2$, $m_1=m_2=0.5$. Each database has its own semi-metric d_1 and d_2.



Intermediate Fuzzy Sets Associated with each Database

YES Evidence Set: The weights of the basic probability assignment are 0.5 for both intervals $(m_1+m_2)/(2 *1)$

Figure 4: Evidence Set Membership Function for answer "YES" to concept x_i

Notice how the "YES" evidence set of figure 4, with only two databases, is much more interesting than the simple fuzzy set version of section 1. The two different semi-metrics d_1 and d_3 for the knowledge space X cause the category constructed for the "YES" answer to concept x_i to be more than just fuzzy, also nonspecific and conflicting. It is important to stress that this more accurate construction of prototypical categories includes more uncertainty forms as a result of structural differences in the information stored in the several relational memory banks utilized. It is the lower level conflicts of the long-term memory that the short-term construction of categories tailored by users reflects. The algorithm of section 1.1 is also expanded to account for the different kinds of uncertainty present in the prototypical categories. The new algorithm is now defined as follows:

1. The user selects the n_d databases of interest and their weights m_k.
2. The user inputs an *initial concept of interest* (one of the key-words) $x_i \in X$.
3. The system creates an evidence set membership function centered on x_i and affecting all its close neighbors using (14). This resulting evidence set of X represents a category that keeps the user's interests in terms of the system's own relations: *The learned category* $A(x)$.
4. The system calculates the total uncertainty of the learned category in its forms of fuzziness, nonspecificity, and conflict (as defined in chapter 3).
5. Another concept $x_j \in X$ is selected. x_j is selected in order to potentially minimize the uncertainty of the learned category, that is, the most uncertain concepts with the most uncertain neighborhoods are selected.
6. The user is asked whether or not she is interested in x_j.
7. If the answer is "YES" another membership function as defined by (14) is created over x_j, and an evidence set union is performed with the previous state of the learned category.
8. If the answer is "NO" the inverse of (14) is created, and an evidence set intersection is performed.
9. The system calculates the total uncertainty of the learned category in its forms of fuzziness, nonspecificity, and conflict.
10. If the uncertainty of the learned category is smaller than half the maximum value attained previously, the system stops since the learned category is considered to have been successfully constructed. Otherwise computation goes back to step 5.

The algorithm can be supplemented with the inclusion of the uncertainty decreasing operation for evidence sets (as described in section 6 of chapter 3), in addition to the evidence set union and interaction. Often, the question-answering process might be lengthy, as the system tries to explore the uncertainty sources in the several databases. If the user decides to terminate the process to achieve a faster search, the uncertainty decreasing operation can be used with pre-existing simple categories reflecting some often used queries. Such categories can be referred to as *templates*. When they are applied to the current knowledge space with the uncertainty-decreasing operation, the former is quickly simplified. This allows the system to escape a lengthy associative process if a fast result is required. The details of this mechanism are not explored here since it does not affect the essentials of the algorithm and is not fundamental for understanding artificial category construction.

1.2.3 Document Retrieval

The mechanism of document retrieval is very similar to the one described in 1.1.3, except that the indices given by equations (5), (6) and (7) are adapted for evidence sets. First the evidence set category is simplified to its closest fuzzy set by a process of elimination of nonspecificity and conflict. Basically, the fuzzy membership is defined as the center of all weighted interval focal elements. Once this fuzzy set is obtained (5), (6), and (7) can be used.

1.2.4 Adaptive Alteration of Long-Term Structure

The adaptive alteration of the long-term structure is also essentially equivalent to the one described in section 1.1.4. When concepts tend to be present in short-term learned categories, their relative distance in the long-term relational structure is adaptively reduced. The only difference is that now the long-term structure is divided in several sub-databases. Thus, when two highly activated concepts in the learned category are not present in the same database (each one exists in a different database) they are added to the database which does not contain them, with property counts given by equations (8) and (9). If the simultaneous activation keeps occurring, then a database that did not previously contain a certain concept, will have its presence progressively strengthened, even though such concept does not really possess any properties in this database – properties end up being associated with it through the concept's relations to native concepts of the database. This way, short-term memory not only adapts an existing structure to its users as the system in section 1.1, but effectively creates new elements in different, otherwise independent, relational databases, solely by virtue of its temporary construction of categories.

1.2.5 Categories as Linguistic, Metaphorical, Structural Perturbation

The evidence set question-answering system follows essentially the algorithm presented in section 1.1 except that now the constructed categories capture more of the prototypical effects discussed in chapter 3. Such "on the hoof" construction of categories triggered by interaction with users, allows several unrelated relational networks to be searched simultaneously, temporarily generating categories that are not really stored in any location. The short-term categories bridge together a number of possibly highly unrelated contexts, which in turn creates new associations in the individual databases that would never occur within their own limited context. Therefore, the construction of short-term linguistic categories in this artificial system, implements the sort of structural perturbation of a long-term system of associations discussed in chapter 2, section 2.4. It is in fact a system of recontextualization of otherwise, contextually constrained, independent relational networks.

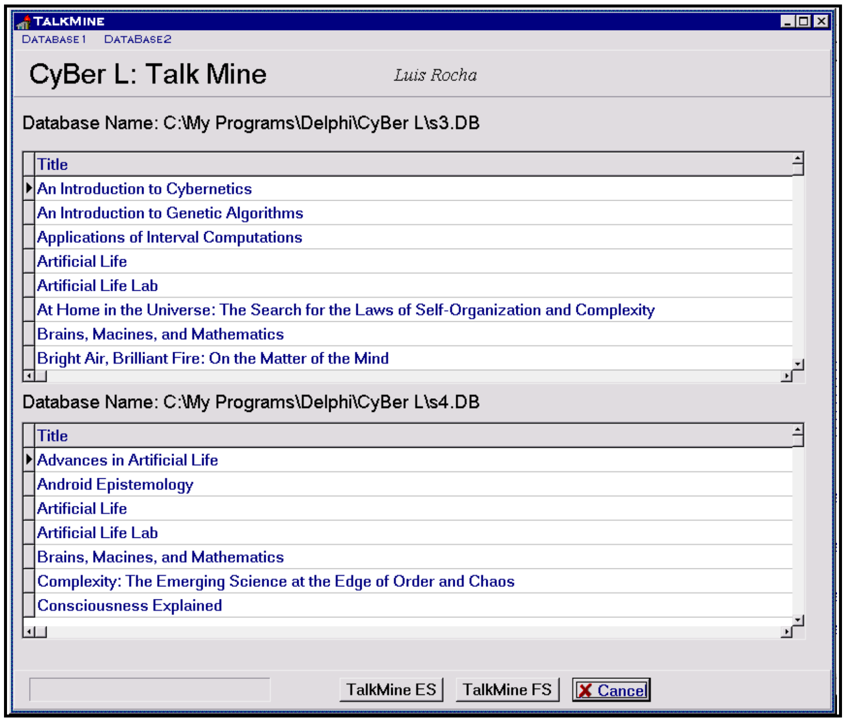
This transference of information across dissimilar contexts through short-term categorization models some aspects of what metaphor offers to human cognition: the ability to discern correspondence in non-similar concepts [Holyoak and Thagard, 1995; Henry, 1995]. Consider the following example. Two distinct databases are going to be searched using the system described above. One database contains the books of an institution devoted to the study of computational complex systems (e.g. the library of the Santa Fe Institute), and the other the books of a Philosophy of Biology Department . I am interested in the concepts of Genetics and Natural Selection. If I were to conduct this search a number of times, due to my own interests, the learned category obtained would certainly contain other concepts such as Adaptive Computation, Genetic Algorithms, etc. Let me assume that the concept of Genetic Algorithms does not initially exist in the Philosophy of Biology library. After I conduct this search a number of times, the concept of Genetic Algorithms is created in this library, even though it does not contain any books in this area. However, with my continuing to perform this search over and over again, the concept of Genetic Algorithms becomes highly associated with Genetics and Natural Selection, in a sense establishing a metaphor for these concepts. From this point on, users of the Philosophy of Biology library, by entering the keyword Genetic Algorithms would have their own data retrieval system output books ranging from “The Origin of Species” to treatises on Neo-Darwinism – at which point they would probably bar me from using their networked database! The point is, that because of the Evidence Set system of short-term categorization that uses existing, fairly contextually independent relational sub-networks, an ability to create correspondence between somewhat unrelated concepts is established.

1.2.6 Open-Ended Structural Perturbation

Given a large number of sub-networks comprised of context-specific associations, the categorization system is able to create new categories that are not stored in any one location, changing the long-term memory banks in an open-ended fashion. Thus the linguistic categorization Evidence Set mechanism achieves the desired system of open-ended structural perturbation of long-term networked memory. As discussed in chapter 2, open-ended in terms of all the available dynamic building blocks that establish the personal construction of associations. Open-endedness does not mean that the categorizing system is able to discern all physical details of its environment, but that it can permutate all the associative information that it constructs in an open-ended manner. Each independent network has the ability to associate new knowledge in its own context (e.g. as more books are added to the libraries of the prior examples): these are the building blocks. To this, the categorization scheme adds the ability of open-ended associations built across contexts. Therefore, a linguistic categorization mechanism as defined above, offers the ability to re-contextualize lower level dynamic memory banks, as a result of pragmatic, consensual, interaction with an environment. This linguistic consensual selection of dynamics, was identified in chapter 2 as the main idea behind Evolutionary Constructivism as the cognitive version of a theory of embodied, evolving, semiosis.

1.2.7 *TalkMine*: The Implemented Application

An application named *TalkMine* was developed that implements the above specified system. It was built with the *Delphi 2.0* application development environment for *Windows 95*. TalkMine accepts database files from such standard databases such as *Paradox* and *dbase*. It is also equipped to deal with client-server databases using *SQL*. The example shown below refers to a database of 150 of my own books. From the pool of 150 books three databases were created by randomly picking books from this pool with equal probability. Each sub-database is comprised of about 50 books, some of which exist in more than one of the sub-databases. Book records are the properties of the system described above. The fields created for these records were Title, Date, Authors, Publisher, plus 6 key-words describing the contents of the books. These keywords are the concepts of the system described above. Naturally, many of these keywords overlap. From the 150 books, 89 keywords were identified. Thus the system has 89 concepts and 150 properties. Figure 5 shows TalkMine's first screen. Two databases are selected *S3.DBD* and *S4.DBD*.



Database Name: C:\My Programs\Delphi\CyBer L\s3.DB

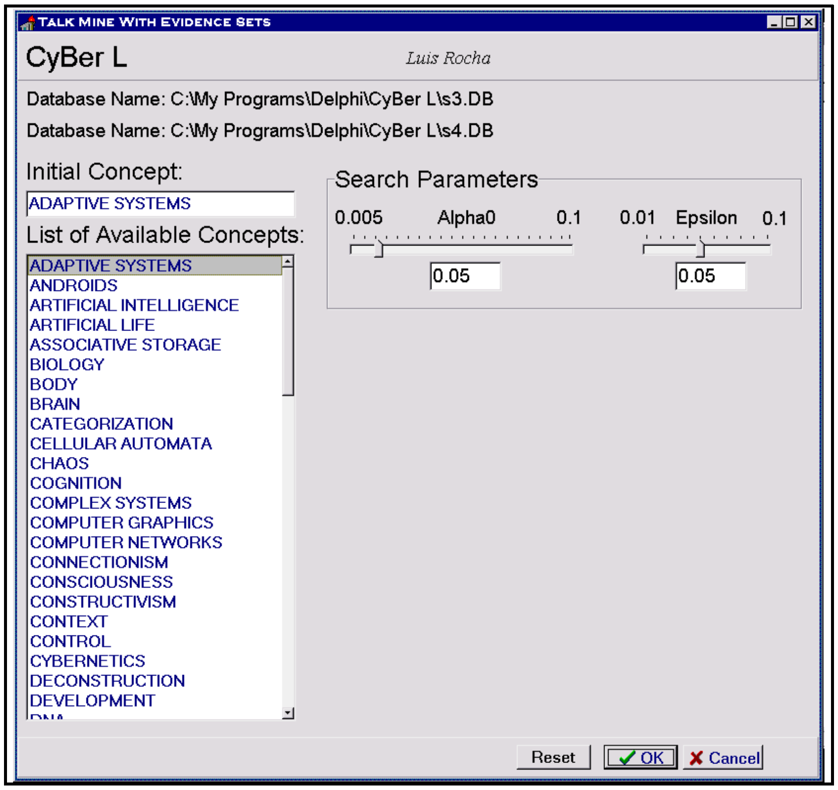
Title
An Introduction to Cybernetics
An Introduction to Genetic Algorithms
Applications of Interval Computations
Artificial Life
Artificial Life Lab
At Home in the Universe: The Search for the Laws of Self-Organization and Complexity
Brains, Macines, and Mathematics
Bright Air, Brilliant Fire: On the Matter of the Mind

Database Name: C:\My Programs\Delphi\CyBer L\s4.DB

Title
Advances in Artificial Life
Android Epistemology
Artificial Life
Artificial Life Lab
Brains, Macines, and Mathematics
Complexity: The Emerging Science at the Edge of Order and Chaos
Consciousness Explained

Figure 5: TalkMine with two Sub-Databases

After selecting two databases, the user selects the *TalkMine ES* option to mine the databases using evidence sets (the *TalkMine FS* option uses Fuzzy Sets). The next screen shows the user all the concepts TalkMine found in the sub-databases. The user can also set the α and ∊ parameters introduced above, that specify how rough or precise the search is to be made and the membership threshold to preserve previous answers. Figure 6 shows this screen.


TALK MINE WITH EVIDENCE SETS

CyBer L *Luis Rocha*

Database Name: C:\My Programs\Delphi\CyBer L\s3.DB

Database Name: C:\My Programs\Delphi\CyBer L\s4.DB

Initial Concept: ADAPTIVE SYSTEMS

List of Available Concepts:

ADAPTIVE SYSTEMS
ANDROIDS
ARTIFICIAL INTELLIGENCE
ARTIFICIAL LIFE
ASSOCIATIVE STORAGE
BIOLOGY
BODY
BRAIN
CATEGORIZATION
CELLULAR AUTOMATA
CHAOS
COGNITION
COMPLEX SYSTEMS
COMPUTER GRAPHICS
COMPUTER NETWORKS
CONNECTIONISM
CONSCIOUSNESS
CONSTRUCTIVISM
CONTEXT
CONTROL
CYBERNETICS
DECONSTRUCTION
DEVELOPMENT

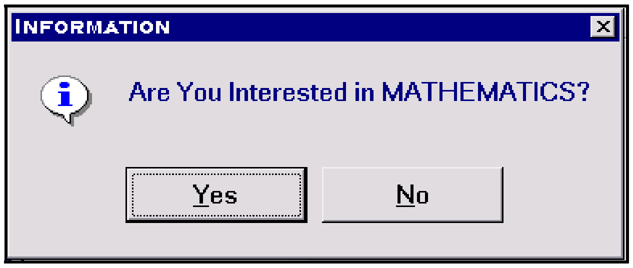
Search Parameters

Alpha0: 0.005 – 0.1, value 0.05

Epsilon: 0.01 – 0.1, value 0.05

Figure 6: Search Screen for TalkMine

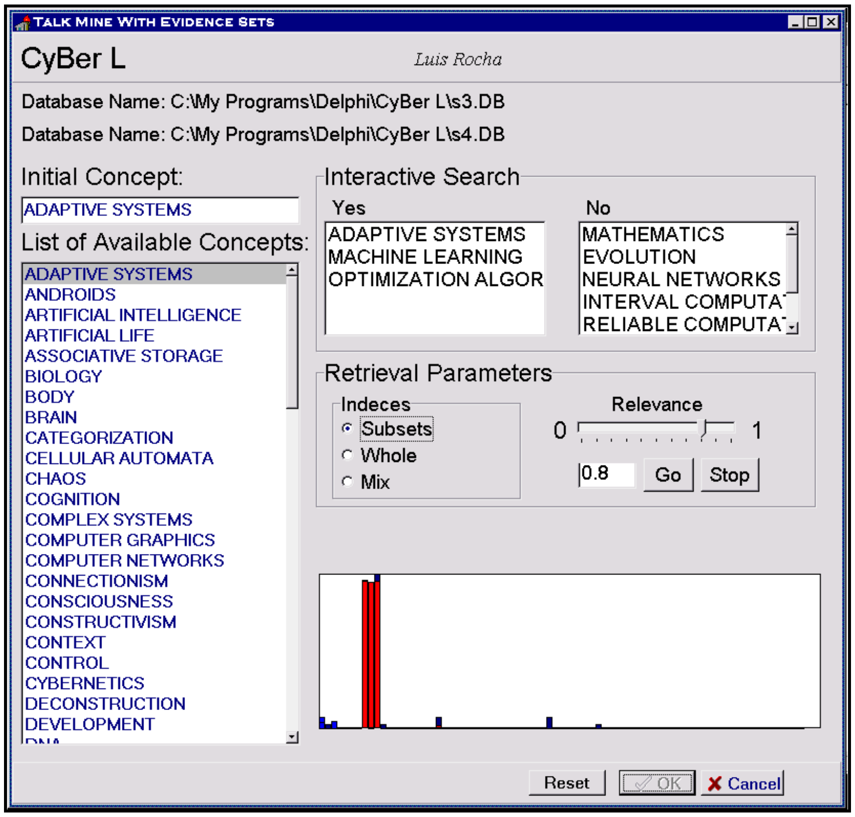
The user then selects an initial concept to start the search, in this case “ADAPTIVE SYSTEMS” is chosen. From this point on, the system starts the question-answering process described earlier, an example of this interaction is shown in figure 7.


INFORMATION

Are You Interested in MATHEMATICS?

Figure 7: Example of Question Posed by TalkMine

After a few steps, the system stops and a few data retrieval options are shown to the users (figure 8). Notice that, all concepts that received “YES” and “NO” answers are posited in their own respective boxes. Furthermore, a graphical interface showing the state of the (evidence set) knowledge space is drawn throughout the question-answering process (figure 8).



TALK MINE WITH EVIDENCE SETS

CyBer L — *Luis Rocha*

Database Name: C:\My Programs\Delphi\CyBer L\s3.DB

Database Name: C:\My Programs\Delphi\CyBer L\s4.DB

Initial Concept: ADAPTIVE SYSTEMS

List of Available Concepts:

ADAPTIVE SYSTEMS
ANDROIDS
ARTIFICIAL INTELLIGENCE
ARTIFICIAL LIFE
ASSOCIATIVE STORAGE
BIOLOGY
BODY
BRAIN
CATEGORIZATION
CELLULAR AUTOMATA
CHAOS
COGNITION
COMPLEX SYSTEMS
COMPUTER GRAPHICS
COMPUTER NETWORKS
CONNECTIONISM
CONSCIOUSNESS
CONSTRUCTIVISM
CONTEXT
CONTROL
CYBERNETICS
DECONSTRUCTION
DEVELOPMENT

Interactive Search

Yes	No
ADAPTIVE SYSTEMS	MATHEMATICS
MACHINE LEARNING	EVOLUTION
OPTIMIZATION ALGOR	NEURAL NETWORKS
	INTERVAL COMPUTA
	RELIABLE COMPUTA

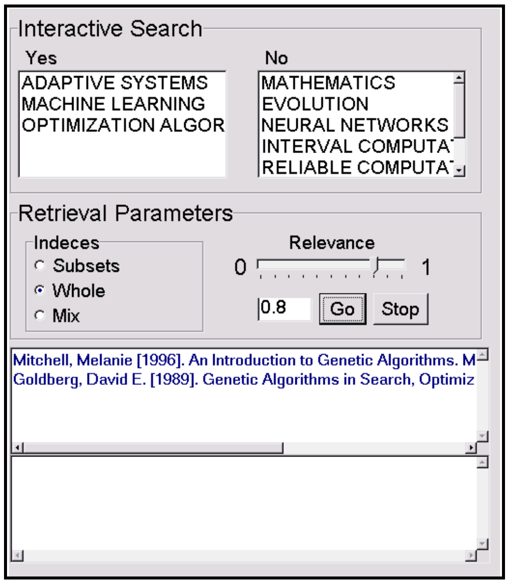
Retrieval Parameters

Indeces: [✓ Subsets] Whole Mix

Relevance: 0 – 1, 0.8

Figure 8: TalkMine’s Retrieval Screen

Examples of retrieval results are shown in figures 9 and 10 for different retrieval options. Notice that the options “Subsets”, “Whole”, and “Mix” refer to retrieval indices (5), (6), and (7) respectively. Figure 11 shows another search this time for the concept “CATEGORIZATION”.


Interactive Search

Yes: ADAPTIVE SYSTEMS, MACHINE LEARNING, OPTIMIZATION ALGOR

No: MATHEMATICS, EVOLUTION, NEURAL NETWORKS, INTERVAL COMPUTA, RELIABLE COMPUTA

Retrieval Parameters

Indeces: Subsets [✓ Whole] Mix

Relevance 0 – 1: 0.8

Go Stop

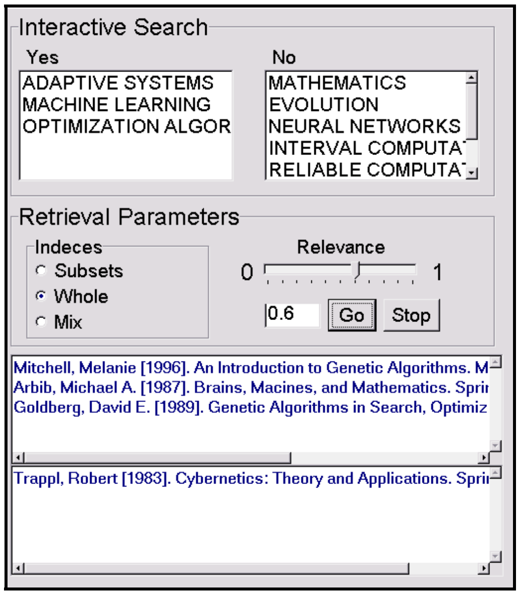
Mitchell, Melanie [1996]. An Introduction to Genetic Algorithms. M
Goldberg, David E. [1989]. Genetic Algorithms in Search, Optimiz

Figure 9: Results for "ADAPTIVE COMPUTATION" with R_2=0.8

TalkMine implements the contextual construction of short-term categories from several long-term relational structures as described above. It is based on prototypical categories represented by evidence sets defined in Chapter 3. It is a reflection of the evolutionary constructivist position discussed in chapter 2, as categories are constructed by the system's own relational structure, which is in turn pragmatically adapted to the consensus of its users.

The long-term relational memory structure implements the personal, subjective, aspects of cognitive systems. It is this structure that ultimately dictates how categories are constructed. The categories constructed are short-term structures not stored in any location, but constructed "on the hoof" as the system relates its several relational sub-networks to the interaction (conversation) users provide. Its syntax is based on evidence sets and their extended theory of approximate reasoning. Semantics is established in accordance to the system's internal relational semi-metrics, and how it pragmatically relates to the users needs.

TalkMine is therefore imbedded in the evolving semiotics framework also described in chapter 2. Furthermore, it explores contextual conflicting uncertainty as a source of artificial category construction, since the selection of concepts for the question-answering process is based on reducing the total uncertainty present in learned categories. Hence, TalkMine is a computational exploration of uncertainty, context, and subjectivity in cognitive systems with very useful results for the field of data mining. It is an implementation of the ideas defended philosophically and mathematically in chapters 2 and 3.


Interactive Search

Yes: ADAPTIVE SYSTEMS, MACHINE LEARNING, OPTIMIZATION ALGOR

No: MATHEMATICS, EVOLUTION, NEURAL NETWORKS, INTERVAL COMPUTA, RELIABLE COMPUTA

Retrieval Parameters

Indeces: Subsets [✓ Whole] Mix

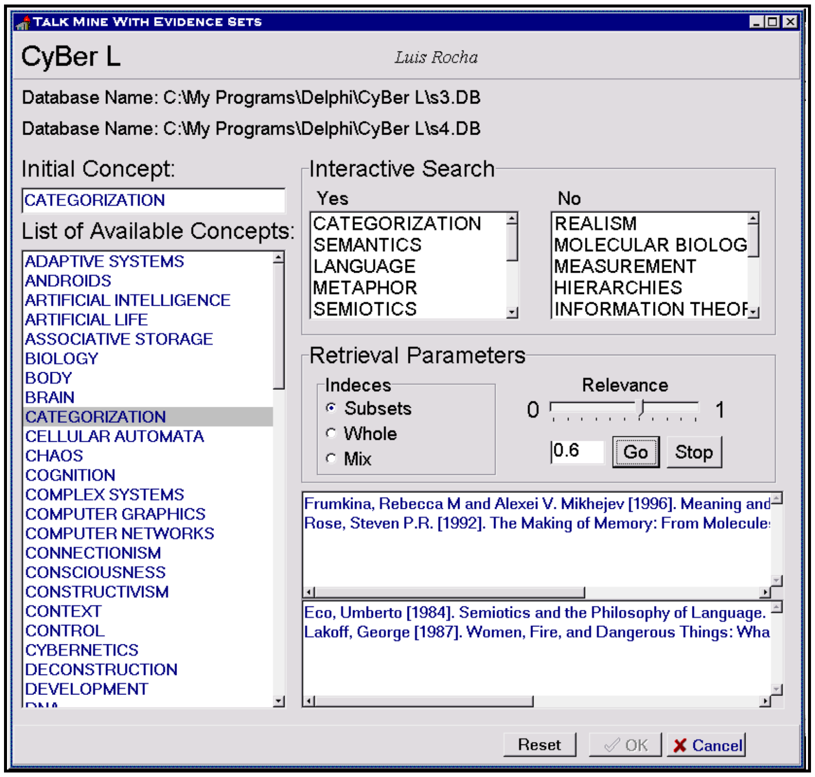
Relevance 0 – 1: 0.6

Go Stop

Mitchell, Melanie [1996]. An Introduction to Genetic Algorithms. M
Arbib, Michael A. [1987]. Brains, Macines, and Mathematics. Sprii
Goldberg, David E. [1989]. Genetic Algorithms in Search, Optimiz

Trappl, Robert [1983]. Cybernetics: Theory and Applications. Sprii

Figure 10: Results for "ADAPTIVE COMPUTATION" with R_2=0.6



Database Name: C:\My Programs\Delphi\CyBer L\s3.DB

Database Name: C:\My Programs\Delphi\CyBer L\s4.DB

Initial Concept: CATEGORIZATION

List of Available Concepts:

ADAPTIVE SYSTEMS
ANDROIDS
ARTIFICIAL INTELLIGENCE
ARTIFICIAL LIFE
ASSOCIATIVE STORAGE
BIOLOGY
BODY
BRAIN
CATEGORIZATION
CELLULAR AUTOMATA
CHAOS
COGNITION
COMPLEX SYSTEMS
COMPUTER GRAPHICS
COMPUTER NETWORKS
CONNECTIONISM
CONSCIOUSNESS
CONSTRUCTIVISM
CONTEXT
CONTROL
CYBERNETICS
DECONSTRUCTION
DEVELOPMENT

Interactive Search

Yes	No
CATEGORIZATION	REALISM
SEMANTICS	MOLECULAR BIOLOG
LANGUAGE	MEASUREMENT
METAPHOR	HIERARCHIES
SEMIOTICS	INFORMATION THEOF

Retrieval Parameters

Indeces: [✓ Subsets] Whole Mix

Relevance: 0 — 1, 0.6

Frumkina, Rebecca M and Alexei V. Mikhejev [1996]. Meaning and
Rose, Steven P.R. [1992]. The Making of Memory: From Molecule

Eco, Umberto [1984]. Semiotics and the Philosophy of Language.
Lakoff, George [1987]. Women, Fire, and Dangerous Things: Wha

Figure 11: Results for search on "CATEGORIZATION" with R_1=0.6

2. Emergent Morphology and Evolving Solutions in Large State Spaces[37]

The Contextual Genetic Algorithm (CGA) with fuzzy indirect encoding defined by Fuzzy Development Programs (FDP'S) described in chapter 4 (section 5.2), was applied to two different cases. The first one refers to the evolution of weights for a large neural network; it is a continuous variable problem. The second, refers to the evolution of Cellular Automata (CA) rules for solving non-trivial problems; it is a discrete variable problem.

2.1 Implementation Details

Both problems were approached with the same FDP architecture. 16 fuzzy set shapes are defined in pool $\mathscr{F}$ ($n_F = 16$), specified by the following functions defined for $x \in [0, L]$:

$F_1 = x/L$	$F_2 = 1\text{-}F_1$	$F_3 = \begin{cases} 2x/L, & x<L/2 \\ 2-2x/L, & x \geq L/2 \end{cases}$	$F_4 = 1\text{-}F_3$
$F_5 = \begin{cases} 0.25, & x<L/2 \\ 0.75, & x \geq L/2 \end{cases}$	$F_6 = 1\text{-}F_5$	$F_7 = \begin{cases} 4x/L, & x<L/4 \\ 1, & x<3L/4 \\ 4-4x/L, & x \geq 3L/4 \end{cases}$	$F_8 = 1\text{-}F_8$
$F_9 = \begin{cases} 0.5-2x/L, & x<L/4 \\ 2x/L-0.5, & x<3L/4 \\ 5/2-2x/L, & x \geq 3L/4 \end{cases}$	$F_{10} = 1\text{-}F_9$	$F_{11} = \begin{cases} 2x/L, & x<L/2 \\ 2x/L-1, & x \geq L/2 \end{cases}$	$F_{12} = 1\text{-}F_{11}$
$F_{13} = \begin{cases} 3x/2L, & x<L/3 \\ 0.5, & x<2L/3 \\ 3x/2L-0.5, & x \geq 2L/3 \end{cases}$	$F_{14} = 1\text{-}F_{13}$	$F_{15} = \begin{cases} 0.5-x/L, & x<L/2 \\ x/L, & x \geq L/2 \end{cases}$	$F_{16} = 1\text{-}F_{15}$

16 fuzzy set operations are defined in pool $\mathscr{O}$ ($n_O = 16$): $\odot_1 = \cup$, $\odot_2 = \cap$, $\odot_3 = Avg$, $\odot_3 = Avg$, $\odot_4 = 1 - Avg$, $\odot_5 = \overline{A \cap B}$, $\odot_6 = \overline{A \cup B}$, $\odot_7 = A \cap \overline{B}$, $\odot_8 = A \cup \overline{B}$, $\odot_9 = \overline{A} \cap B$, $\odot_{10} = \overline{A} \cup B$, $\odot_{11} = A.B$, $\odot_{12} = 1 - A.B$, $\odot_{13} = \overline{A} \cap (A \cdot B)$, $\odot_{14} = \overline{A} \cup (A \cdot B)$, $\odot_{15} = \overline{B} \cap (A \cdot B)$, and $\odot_{16} = \overline{B} \cup (A \cdot B)$. All these operations are simple linear functions of two generic fuzzy sets A and B.

[37] Portions of the work here presented were published in Rocha [1997d]

The universal set X of the problems presented next was divided in 16 Parts ($n_X = 16$). Some experiments were also conducted with $n_X = 32$ which will be identified ahead. Experiments were conducted with FDP's of lengths $n = 8$, 16, and 24, which according to equation (1) in chapter 4, section 5.2.3, require binary chromosomes of lengths $\nu = 144$, 288, and 432 respectively.

2.2 Continuous Variables: Evolution of Neural Network Weights

Interesting results have been obtained when using genetic algorithms (GA's) to evolve the weights of a neural network with a fixed architecture. For instance, Montana and Davis [1989] utilized a GA instead of a standard training algorithm such as back-propagation to find a good set of weights for a given architecture. It is reasoned that the advantage of using GA's instead of back-propagation lies in the latter's tendency to get stuck at local optima, or the unavailability of a large enough set of training patterns for certain tasks [Mitchell, 1996a]. I explore the problem here, not to prove or disprove its merits, but simply because it is usually a very computationally demanding problem, which can potentially benefit from the chromosomal information compression offered by CGA's with fuzzy indirect encoding.

The set of weights of a neural network defines a vector of real values. Montana and Davis' network required a vector of 126 weights to evolve. Because the weights are real-valued, they used what is usually referred to as the *real encoding* of chromosomes in a GA. That is, instead a binary strings as chromosomes, a real-encoded GA uses strings of real numbers. The operation of mutation is now different: a random number is added to each element of the vector with low probability. Montana and Davis also used a different kind of crossover operation, the details of which are unnecessary for the current presentation (see Montana and Davis [1989] for more details).

2.2.1 Hand-Written Character Recognition: The Network Architecture

I applied the CGA with fuzzy indirect encoding to a problem that requires larger real-valued chromosome vectors. The problem I approached was that of the recognition of hand-written characters in an 8 by 10 grid[38]: each pattern is defined by an 80 bits long vector. There were a total of 260 patterns available, which contained 5 different categories, that is 5 different hand-written characters with 52 instance patterns each. The patterns were divided equally between the *learning* and the *validation sets*, preserving the ratios per category. In other words, the learning and the validation sets contained both 130 patterns, 26 patterns for each of the 5 categories.

Since each pattern is an 80-length bit vector, the input layer to the neural network has also 80 nodes, plus one for the bias unit. The output layer requires only 3 nodes, whose binary output encode 5 categories (3 nodes can encode up to 8 categories). Experiments were ran with a hidden layer of 2 and 5 Nodes. The information requirements of the sets of weights for these architectures are summarized in the following table:

[38] These patterns were obtained from Don Gause with his permission from the data files of the Neural Network and Genetic models class.

Neural Network Architecture with 81 input nodes and 3 output nodes		
Hidden Nodes	2	5
Number of Weights	171	423
Bits of Information (4 byte real number)	5472	13536

Thus, the real encoded GA requires chromosomes defined by real-valued vectors of length 171 and 423, for the 2 and 5 hidden nodes architectures, respectively. Such vectors actually cost some computer implementation 5472 and 13536 bits of information each (assuming an implementation where real numbers are only 4 bytes long which is actually rather small).

2.2.2 Results from Back-Propagation

The results reached by the back-propagation algorithm are irrelevant to establish CGA's with fuzzy indirect encoding, since the results from these are to be compared instead with traditional GA's. In any case, for this problem, back-propagation classified the patterns much better than either GA, and can be regarded as an upper limit for the desired behavior of classifying systems. A typical run is shown in figure 12. The network converges easily to the 130 patterns of the learning set. The number of correctly classified patterns in the validation set does not surpass 100 patterns. Networks with more hidden nodes converge a bit faster

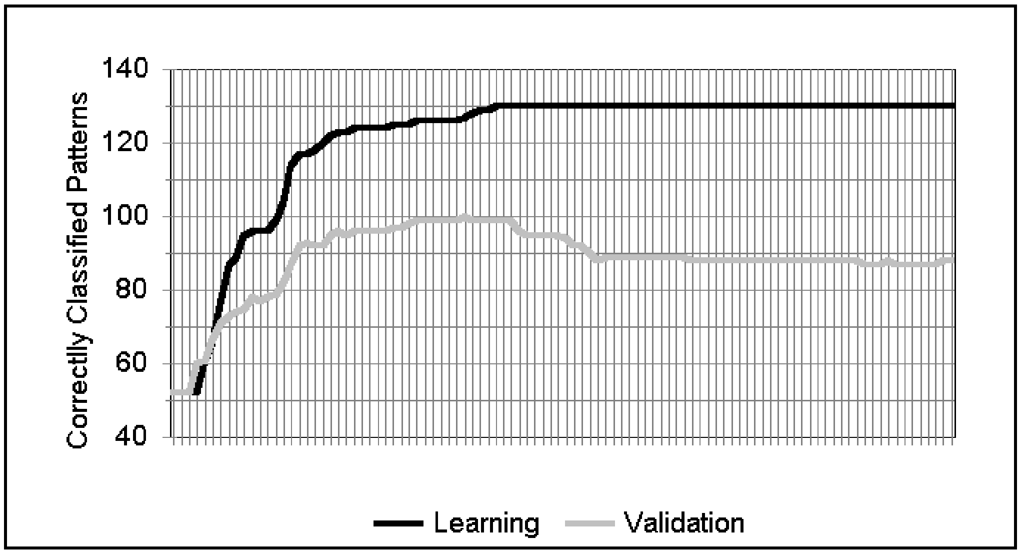


Figure 12: Typical run for back-propagation network with 2 hidden nodes

to the learning set, but do not result in better classifications of the validation set. 100 runs were performed, 50 for each architecture. The learning set converged every time, while the average cross-validation was close to 94 patterns out of 130. The following table summarizes the results.

Back-Propagation Algorithm		
Number of Hidden Nodes	2	5
Number of Runs	50	50
Average number of correctly classified patterns in learning set	130	130
Average number of correctly classified patterns in validation set	93.2	94.4

A typical weight vector obtained from the state of the network whose run is depicted in figure 12 when highest cross-validation was reached, is shown in figure 13.

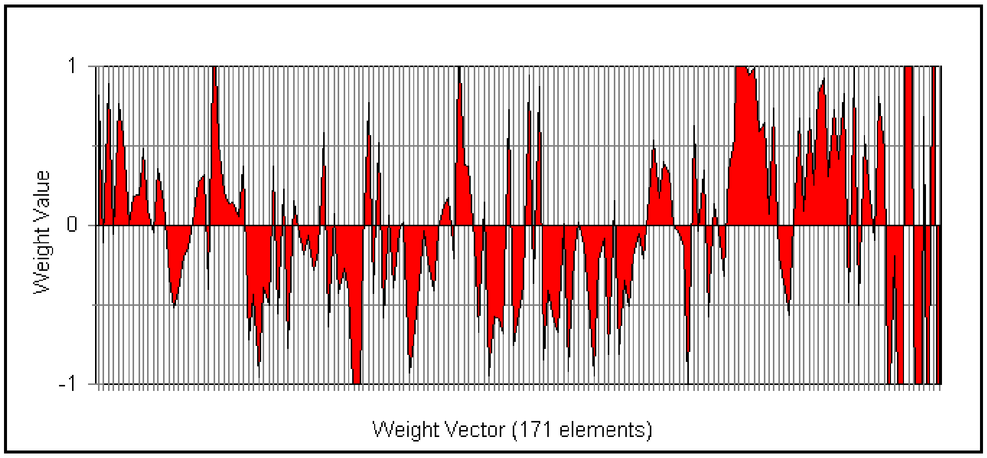


Figure 13: Typical weight vector for back-propagation network with 2 hidden nodes.

2.2.3 Results from Real-encoded GA

The runs with a standard GA were performed with real-encoded chromosomes (the weight vectors). The population size was set to 100 elements. Each run consisted of 500 generations. The fitness function was defined simply as the number of correct classifications of the patterns in the learning set. In each generation, every chromosome was decoded into a network and the learning set presented to it. The network weight vectors that classified more patterns had higher probability of selection into the next generation. Probability of crossover was set to 0.7, and probability of mutation to 0,002[39]. The following table presents the results obtained.

[39] Other values of these probabilities were tried, as well as other fitness functions. The set of parameters described above proved to be the most efficient.

Real-Encoded GA		
Number of Hidden Nodes	2	5
Number of Runs	50	50
Average Time of runs (min)	7	24
Average number of correctly classified patterns in learning set	108.7	114.7
Average number of correctly classified patterns in validation set	81.8	79.9

The average time of runs refers to the particular computational setup I used: an *Intel Pentium* 166MHz. The code was written using Delphi 2.0. This algorithm is very computationally demanding due to extensive floating point operations, since the weight vectors are real-valued. The average time for each run increases dramatically with the number of hidden nodes included. The algorithm never converges completely to the learning set, but does reach comparable, though lower, results to the back-propagation algorithm for the validation patterns. Notice that the network with 5 hidden nodes did not yield as good results for validation as the one with 2 hidden nodes, though the reverse happened for the learning set. Figure 14 shows a typical weight vector obtained by the GA for a network with two hidden nodes. Notice that it is more

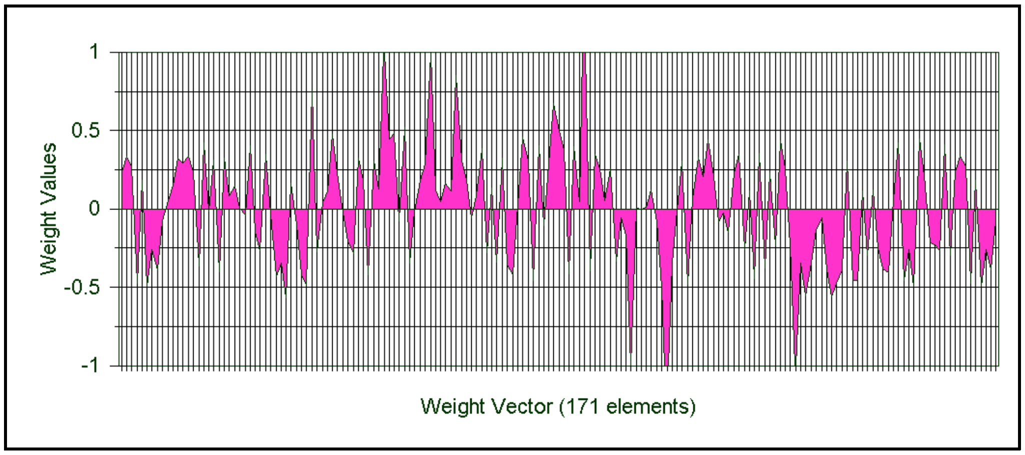


Figure 14: Weight Vector for Network with 2 hidden nodes obtained with GA

random that the one obtained from back-propagation. I believe that this more random pattern causes the network to overly adapt to the learning set and fail to generalize the categorical relationships properly. This might explain why networks with more hidden nodes present worse cross-validation behavior, since the weight vector that overly adapted to the specific relationships of the learning set is much larger, and being very random, the probability that it will also categorize the validation set will be smaller.

2.2.4 Results from CGA with Fuzzy Indirect Encoding

The fuzzy indirect encoding of the networks' weight vectors, transforms them from real-valued vectors to binary vectors represented by FDP's. That is, the CGA uses as chromosomes not the weight vectors directly, but binary FDP's that indirectly encode weight vectors. The length of the binary FDP's is in bits presented in the following table according to the parameters specified in section 2.1. The length in bits is calculated with equation (1) defined in chapter 4, section 5.2.3.

FDP Length (n)	8	16	24
16 Partitions (n_X=16)	144	288	432
32 Partitions (n_X=32)	168	336	504

Compare the bit sizes of these chromosomes to the ones required of real-encoded chromosomes as shown in the table in section 2.2.1. For instance, to encode a 5 hidden nodes network with direct real-valued chromosomes we need 13536 bits, while with fuzzy indirect encoding we only need 144 to 504 bits depending on the parameters of the FDP. Notice that the size of the FDP is not dependent on the size of the network. From the point of view of the chromosomes, it is the same to encode a 2 or 5 hidden layer network. However, though the chromosomes do not depend on the size of the network, the fitness evaluation naturally does. Indeed, each chromosome will be decoded into a specific network that is eventually evaluated regarding the problem at stake. Indirect encoding can only off-load computational effort from the encoding and variation portions of GA's, not from the evaluation steps. The results from the indirect encoding runs are shown on the following tables:

	Fuzzy Indirect Encoding CGA with n_X=16					
Hidden Nodes	2			5		
FDP Length	8	16	24	8	16	24
Number of runs	50	50	50	35	50	20
Avg. Run time (min)	3	3	3	15	16	17
Avg. Correct Patterns in Learning Set	95.0	98.2	107	92.2	98.3	96
Avg. Correct Patterns in Validation Set	80.1	81.6	82.9	77.5	84.1	77.5

	Fuzzy Indirect Encoding CGA with n_X=32					
Hidden Nodes	2			5		
FDP Length	8	16	24	8	16	24
Number of runs	20	20	-	-	20	-
Avg. Run time (min)	3	4	-	-	18	-
Avg. Correct Patterns in Learning Set	98.2	96.1	-	-	96.6	-
Avg. Correct Patterns in Validation Set	79.8	78	-	-	80.3	-

The first thing to observe from these tables is that there was no advantage in increasing the number of partitions of the universal set of the FDP's. Even though a considerable smaller number of experiments was conducted for 32 partitions, the results were generally worse than the experiments with 16 partitions. The results from the 16 partitions cases were, however, very promising. The indirect encoding runs took as luttle as half the time as the direct encoding ones with comparable results. The standard GA showed better results for the patterns in the learning set, however, better cross validation results were obtained on average by the fuzzy indirect encoding scheme. For the 2 hidden nodes network, the highest average of correct patterns in the validation set was obtained by the CGA with fuzzy indirect encoding with FDP's with 24 fuzzy sets (82.9), slightly better than the cross-validation obtained by the regular GA (81.8). For the 5 hidden nodes network, the difference was even higher: 84.1 for the CGA, and 79.8 for the GA.

Notice that the objective was not to top GA's, but simply to obtain comparable results. The indirect encoding scheme reduced the chromosome size dramatically. Consequently, the computation time was strongly cut down, not just because of the size of chromosomes was strongly reduced but because instead of floating-point operations the genetic operations are now based on boolean variables. FDP's work by constructing simple linear functions (the fuzzy set shapes in section 2.1) over the solution space (the weight vector in this case). Therefore, it restricts solutions into the reduced space of possibilities built out of simple linear functions.

This reduction is apparent in Figure 15 that shows a typical weight space obtained by the CGA for a network with 2 hidden nodes. Indeed, the weight vector is not random, we can detect the simple fuzzy set shapes used by the FDP's repeated here an there. It is obvious that this scheme tries to explore non-random relationships in the solutions reached, whereas the traditional GA reaches much more random solutions. It is remarkable that the sort of ordered solutions reached by the CGA with fuzzy indirect encoding are actually good solutions. It shows that the problem being approached is rich in simple relationships that can be explored. This might explain why the CGA obtained better results for cross-validation than the traditional GA. The latter seems to converge well to specific solutions that solve the particular patterns in the validation set, but it is not as good at generalizing the categorical relationships in the data needed for good cross-validation results.

We can also observe that the indirect encoding scheme introduces a number of parameters that may end up dictating its behavior. For instance, FDP's of length 16 encoding networks with 5 hidden nodes, seemed to work better for this problem than all others. Moreover, and more importantly, ultimately the

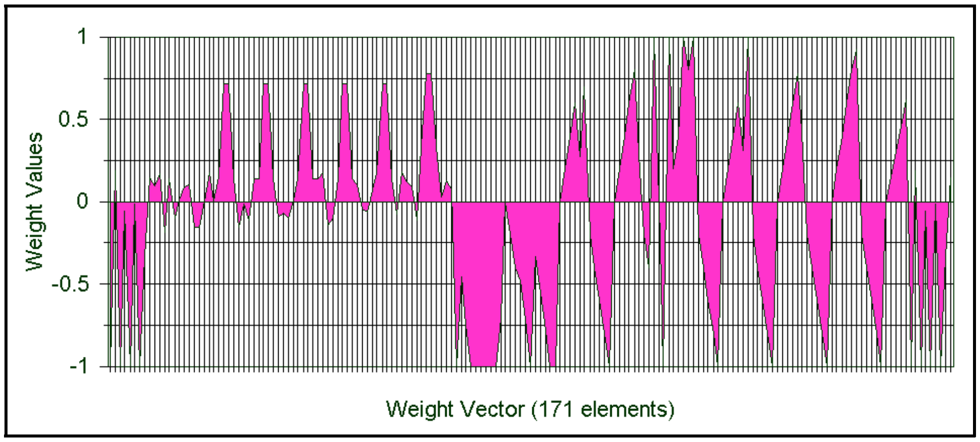


Figure 15: Typical Weight Vector for Network with 2 Hidden Nodes obtained with the fuzzy indirect encoding CGA

solutions obtained depend on the kind of fuzzy set shapes used to define FDP's (section 2.1). In this case, only very simple linear shapes were used. Such simple functions might prevent us from reaching large, important, portions of the solution space. Perhaps more complicated shapes would yield even better results. This dependence on the building blocks of the intermediate developmental stage in the translation of chromosomes into solutions, was precisely defended in chapters 2 and 4 as the constraints on evolutionary systems that both enable and restrict the evolution of fit solutions. With the previous example, we can see that this behavior can be explored computationally; the FDP's used did ease computational description efforts, but we can also appreciate the kinds of constraints it poses on solutions evolved by merely looking to figure 15. We can understand FDP's as the simulation of material, dynamic, constraints on evolution as desired of selected self-organization. These issues are discussed again in section 2.4, and chapter 6.

2.3 Discrete Variables: Evolution of Cellular Automata Rules

The evolution of neural network weights with CGA's with fuzzy indirect encoding proved to be computationally relevant because very large real-valued chromosomes were reduced to smaller binary versions, in some cases, cutting in half the computation time. However, as it is well known in evolutionary computation, often the bulk of the computation in GA's lies on the implementation of the fitness function and not on the genetic variation engine. In such cases, unless the reduction of chromosome is dramatic, there will be no perceivable advantage of fuzzy indirect encoding schemes. Moreover, if the solutions of the problem we wish to solve are already discrete, the advantages of switching from floating-point to binary operations observed in the previous section will not occur. Thus, when present with discrete domain problems, particularly with computationally demanding fitness functions, the reduction of chromosome size has to be large for fuzzy indirect encoding to prove profitable. The problem presented in this section offers precisely that.

A very interesting problem that GA's have been used successfully in, is the evolution of Cellular Automata (CA's) rules for the solution of non-trivial tasks[40]. CA's are often used to study the behavior of

[40] This work has been mainly pursued by the Adaptive Computation Group of the Santa Fe Institute, whose members I am indebted to for many conversations on this problem. I particularly wish to thank Melanie Mitchell and Wim Hordijk for making many of their unpublished research available to me,

complex systems, since they capture the richness of self-organizing dynamical systems which from local rules amongst their components, observe emergent behavior (see chapter 2). “CAs are decentralized spatially extended systems consisting of a large number of simple identical components with local connectivity.” [Mitchell, 1996b, page 1].The components (cells) are finite-state machines arranged in a lattice *N*, each with an identical state transition rule based upon the state of their immediate neighbors. The CA’s used in this section are one-dimensional with binary states.

Certain CA rules are capable of solving global tasks assigned to their lattices, even though their transition rules are local. One such tasks is usually referred to as the *density task*: given a randomly initialized lattice configuration, the CA should converge to a global state where all its cells are turned “ON” if there is a majority of “ON” cells in the initial configuration (IC’s), and to an all “OFF” state otherwise. This rule is not trivial precisely because the local rules of the component cells do not have access to the entire lattice, but can only act on the state of their immediate neighborhood. Crutchfield and Mitchell [1995] used a GA to evolve the CA rules for such a task. They used one-dimensional CA’s with 149 cells. The local rule of this CA works on a neighborhood of 7 cells (the rule’s cell plus 3 cells to each side), which is often referred to as a neighborhood of radius 3. Such a rule has 128 possible states since the CA is binary (2^7=128), and so it is defined by a lookup table with 128 entries. To encode such a lookup table in a GA we require only a 128-bit string if we fix the order of the lookup table entries. Their GA used a population of 100 of such randomly initialized GA’s. The fitness of each rule was computed by iterating the corresponding CA on 100 randomly chosen ICs uniformly distributed over the density of “ON” cells, $\rho_0 \in [0, 1]$, half with $\rho_0 < 0.5$ (correct classification all “OFF’s”) and half with $\rho_0 > 0.5$ (correct classification all “ON’s”), and by recording the fraction of correct classifications performed in a certain number of steps (usually a bit over 2N) [Ibid, page 10743]. Notice how computationally demanding this fitness function is, as every single chromosome has to be decoded into a CA and run for about 300 cycles.

The GA found a number of fairly interesting rules, but 7 out of the 300 runs evolved very interesting rules (with high fitness) which create an intricate system of lattice communication. Basically, groups of adjacent cells propagate certain patterns across the lattice, which as they interact with other such patterns “decide” on the appropriate solutions for the lattice as a whole. Several types of signaling patterns and their communication syntax have been identified, and can be said to observe the emergence of

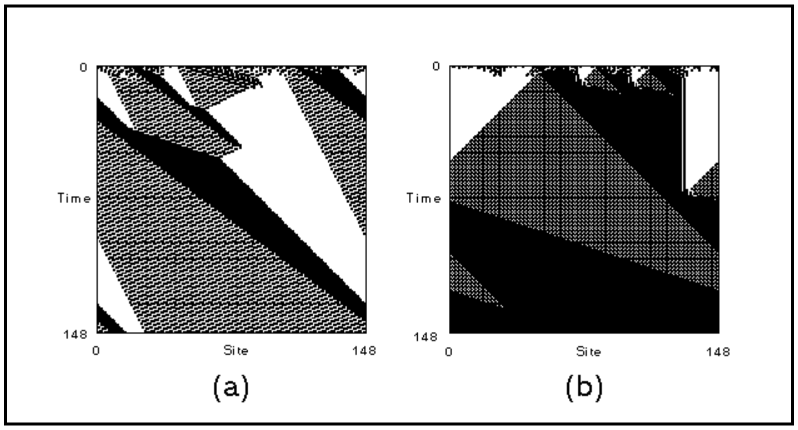


Figure 16: Space-time diagram of one-dimensional CA’s as they deal with the density task.

---

without which I would not be able tackle the problem.

embedded-particle computation in evolved CA's [Ibid; Hordijk, Crutchfield, Mitchell, 1996]. Figure 16 (from [Crutchfield and Mitchell, 1995] shows 2 examples of these rules as they tackle the same IC, the first rule (a) classifies the IC properly while the second (b) misclassifies. The space-time diagram shows the temporal development of the one-dimensional lattice. Each line represents the state of the lattice at a particular instance. Black cells are "ON" states, while white cells are "OFF" states. Notice the existence of several types of areas, which define the signaling particles. The best rules for this system have a fitness of 0.74, while the average fitness for all rules evolved by their GA was 0.64[41].

I replicated their experiments and obtained the same value for average fitness, but the rule with highest fitness I encountered rated only 0.67. This may have been because, due to very lengthy computations, I only tried 75 runs (they found only 7 particle-computation rules out of 300 runs). In any case, my aim was to apply the fuzzy indirect encoding scheme to this problem, so the bulk of my effort was pursuing this aim and not to replicate their experiments. I used the CGA with fuzzy indirect encoding with the parameters described in section 2.1, with FDP's of length 8 and 16, which require binary chromosomes of length 144 and 288 respectively. Since their chromosomes use 128 bit-strings, there is no advantage in using fuzzy indirect encoding for this problem, except that it is important to compare the results. 120 runs were performed and the same average fitness 0.64 was obtained. The maximum fitness I observed was once again 0.67. In other words, I obtained the same results with the CGA than with my runs of the regular GA.

I never evolved one of the particle-computation rules for CA's with neighborhoods of radius 3. At least in my experiments, there was no difference between the performances of the directly and indirectly

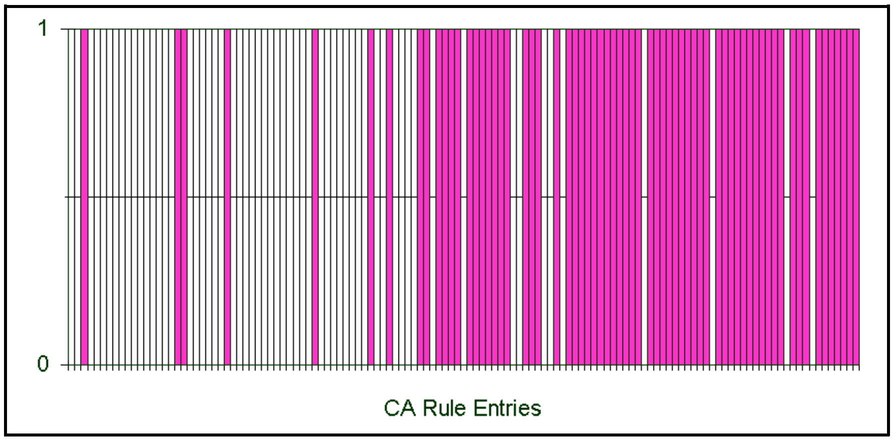


Figure 17: Typical CA rule of radius 3 evolved with the traditional GA. Each element of the horizontal axis represents an entry of the rule's lookup table. A bar denotes an "ON" state. The hexadecimal representation of the 128-bit string this rule defines is:
8000C0400010216FEFC9EFFDFFEFFEEF

encoded GA's. Figures 17 and 18 show the highest rules evolved for the two cases. Both are similar and quite typical of the rules evolved: the lookup entries towards the end of the binary vector tend to be "ON", while the ones on the left ten to be "OFF". This is natural since the first (last) elements in the lookup table refer

[41] This fitness is not the same as the one used in the GA. The latter uses IC's whose density of ON's is uniformly distributed in the unit interval, while the former is the so-called unbiased fitness. It is computed by randomly flipping each bit of the IC's with equal probability. This results in densities closer to 0.5 which are the toughest for the rule to tackle.

to lower (higher) binary numbers which means that most cells in the neighborhood are “OFF” (“ON”). However, the CGA with fuzzy indirect encoding tends to explore relationships in the solution vector by lumping entire neighborhoods into one state, so its rules are a bit more correlated that the ones obtained with the standard GA.

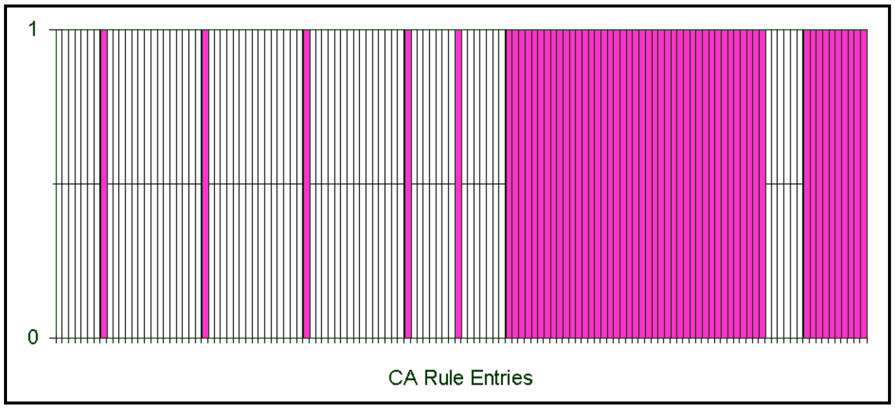


Figure 18: Typical CA rule of radius 3 evolved with the fuzzy indirect encoding CGA. The hexadecimal representation of the 128-bit string this rule defines is: 01000100010001010 1FFFFFFFFFF03FF

The indirect encoding scheme can only be useful once we try to evolve CA rules of radius larger than 3. The following table shows the sizes of lookup tables, and consequently chromosome size, for rules of radius 3, 4, and 5.

CA Radius	3	4	5
Lookup Table size (bits)	128	512	2048

Notice, as shown in section 2.2.4, that FDP’s of length 8 and 16, require binary chromosomes of

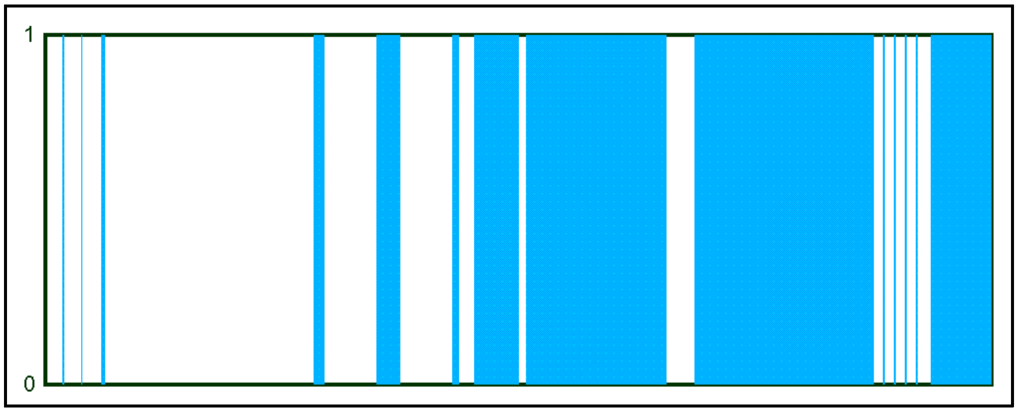


Figure 19: Best CA rule with radius 4 evolved with the fuzzy indirect encoding CGA. The hexadecimal representation of the 512-bit string this rule defines is:
00401003000000000000000000000000007E0000001FFF0000000F00FFFFFF0FFFFFFF
FFFFFFFFFFFF0001FFFFFFFFFFFFFFFFFFFFFFFF04104101FFFFFFFF

length 144 and 288 respectively (for 16 partitions of the universal set of the FDP's). Thus it is profitable to use such FDP's to encode the CA rules of radius 4 and 5. I ran 150 experiments with CA rules of radius 4, and 50 experiments with CA rules of radius 5. With FDP's of length 8, the results of the evolution of CA rules of radius 4 echoed the results for rules of radius 3, that is, average fitness 0.64 and maximum fitness 0.67. However, for FDP's of length 16, better results were attained. The average fitness increased to 0.65, and the maximum fitness found was 0.73 for the ruleshown in figure 19, which observes the desired characteristics of embedded particle-computation (figures 20 and 21). The sort of signal interaction this CA rule creates, is quite different from the ones created with the CA rules of radius 3, which is to be expected of a different dynamics. It is not the purpose of the present work to study in detail this rule. The point is that with fuzzy indirect encoding, we were able to peek into a much higher dimensionality space than the one that thus far has been used for this particular problem. It shows that particle computation also exists in CA's with rules of radius larger than 3.

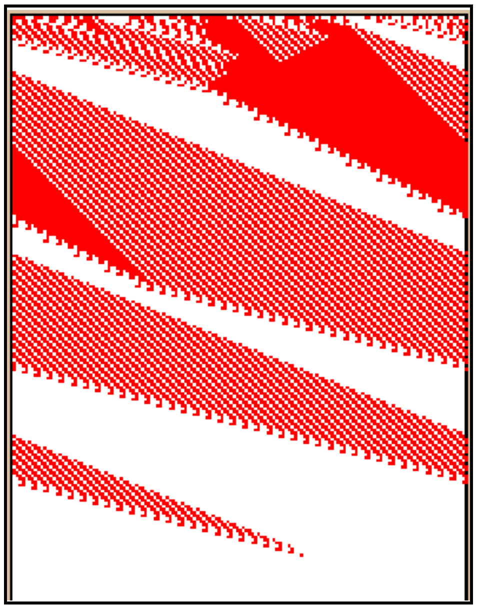


Figure 20: Space-time diagram of best CA rule of radius 4

As shown in this section, fuzzy indirect encoding CGA's can be used successfully in discrete domain variables. Indirect encoding can be useful because it reduces the memory requirements of the populations of chromosomes in GA's. In this particular case, the advantage is not so much the reduction in computation time, since whether we use direct or indirect encoding, the bulk of computational effort has to be posited on the fitness evaluation function that requires large CA's to be run. In any case, the size of the memory required to store the chromosomes of the GA is significantly reduced, which is important in computer implementations.

This is particularly obvious in the evolution of CA rules of radius 5, whose lookup table requires 2048 entries. By using the fuzzy indirect encoding scheme we can reduce the size of chromosomes from 2048 to 144 or 288 depending on the size of FDP's used. I ran only 50 experiments for this case. The average fitness was again 0.64, but the maximum fitness observed was 0.68 for FDP's of size 8 (only 144 bits per chromosome!). The results were not impressing from the point of the fitness values, but we were at least able to peek into the dynamics of such a high dimensionality space. Indirect encoding explores correlations between neighbors in the solution vectors, which restrict the search to smaller volumes of the state space. If good solutions exist in these restricted volumes of the solution space, then indirect encoding will be very profitable. Indeed, it may be the only alternative to even peek at state spaces that are just too large for regular GA searches. The following table presents all the results obtained from the experiments in this section.

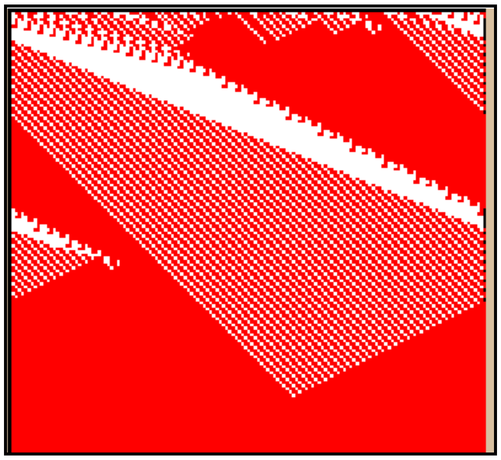


Figure 21: Space-Time View of the Best rule evolved for a CA rule of radius 4

	Regular GA	Fuzzy Indirect Encoding CGA				
CA Rule Radius	3	3	4		5	
Number of Runs	75	120	150		50	
Average Fitness	0.64	0.64	0.64 (*n*=8)	0.65 (*n*=16)	0.64 (*n*=8)	0.64 (*n*=16)
Best Fitness	0.67	0.67	0.67 (*n*=8)	**0.73** (*n*=16)	0.68 (*n*=8)	0.67 (*n*=16)

2.4 The Effectiveness of Computational Embodiment: Epistasis and Development

The introduction of an intermediate development layer in CGA's serves to reduce the size of genetic descriptions (see chapter 4). This is done because some fixed computationally building blocks are assumed for any problem we encode. Chromosomes code for these building blocks which will themselves self-organize (develop) into a final configuration. The algorithm used in this chapter achieves precisely that. It is an instance of the selected self-organization ideas presented in chapter 2. The developmental stage is in effect simulating some specific dynamic constraints on evolution posed by a simulated embodiment of the evolving semiotic system.

Fuzzy indirect encoding captures computationally the advantages of materiality by reducing genetic descriptions, which may be very relevant in practical domains such as data mining. However, it does also capture the reverse side of embodiment, that is the limiting constraints that a given materiality poses on evolving systems. Given a specific simulated embodiment defined by the particular fuzzy set shapes and operations used (see section 2.1), the algorithm cannot reach all portions of space of solutions. It can only reach those solutions that can be constructed from the manipulation of the allowed fuzzy sets and operations – the building blocks. This echoes my early observation (see chapter 2 and 4), that the genetic system is similarly not able to evolve anything whatsoever, but only forms that can be built out of aminoacid chains. Such constraints are observed on the problems of sections 2.2 and 2.3, as the solution vectors obtained by the indirect encoding scheme are much less random than the solutions obtained by direct encoding. The solutions reached are much more ordered, in other words, the inclusion of the developmental stage introduced a lot of order "for free". Such order was not a result of the selection mechanism, but of the intrinsic dynamics the system possesses (the developmental rules) that tends to produce mostly ordered solutions.

Another way to think of this, is that in traditional GA's, each position of the solution vector maps to only one position in the chromosomes (allele), which can be independently mutated. In other words, the mutation of one bit in the chromosome will affect only one component in the solution vector. Whereas in the indirect encoding scheme, each bit of the chromosomes affects several elements of the solution vector non-linearly. In the scheme here utilized, flipping one bit in the FDP may result in changing a fuzzy set operation to another, thus causing the fuzzy sets it operates on to be connected in a totally different manner for all its elements. Thus, one single bit of indirectly encoded chromosomes can affect many, potentially all, elements of the solution vector as the development program is changed. This introduces *epistasis* to evolutionary computation, which we know exists in natural genetic systems.

All of these aspects of indirectly encoded GA's, both enabling constraints such as genetic information compression, and limiting constraints such as reduction of the space of solutions, are desired of models of evolutionary systems. But what does it mean for practical applications in data mining? Genetic compression is obviously a plus, but the limiting constraints may be problematic if the reduced search space includes only mediocre solutions to our problems. When genetic descriptions are not very large, the only reason to use indirect encoding is if we wish to avoid very random solution vectors that tend to be produced with GA’s. This was the case of the evolution of neural network weights in section 2.2. The indirect encoding scheme was able to produce better cross-validation results. Due to its intrinsic order, its solutions did not overly adapt to the patterns in the learning set. When the genetic descriptions are very large, or require real-encoding (such as the case of section 2.2), then it is advantageous to use indirect encoding. Sometimes, even if the reduced solution space is mediocre, such less that optimum solutions might be all that we can hope to find in huge solution spaces, inaccessible to standard GA’s due to computational limitations.