International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:11, 2007

Learning Classifier Systems Approach for Automated Discovery of Crisp and Fuzzy Hierarchical Production Rules

Suraiya Jabin, and Kamal K. Bharadwaj

***Abstract*—**This research presents a system for post processing of data that takes mined flat rules as input and discovers crisp as well as fuzzy hierarchical structures using Learning Classifier System approach. Learning Classifier System (LCS) is basically a machine learning technique that combines evolutionary computing, reinforcement learning, supervised or unsupervised learning and heuristics to produce adaptive systems. A LCS learns by interacting with an environment from which it receives feedback in the form of numerical reward. Learning is achieved by trying to maximize the amount of reward received. Crisp description for a concept usually cannot represent human knowledge completely and practically. In the proposed Learning Classifier System initial population is constructed as a random collection of HPR–trees (related production rules) and crisp / fuzzy hierarchies are evolved. A fuzzy subsumption relation is suggested for the proposed system and based on Subsumption Matrix (SM), a suitable fitness function is proposed. Suitable genetic operators are proposed for the chosen chromosome representation method. For implementing reinforcement a suitable reward and punishment scheme is also proposed. Experimental results are presented to demonstrate the performance of the proposed system.

***Keywords*—**Hierarchical Production Rule, Data Mining, Learning Classifier System, Fuzzy Subsumption Relation, Subsumption matrix, Reinforcement Learning.

I. Introduction

KNOWLEDGE discovery in databases (KDD) has become a very attractive discipline both for research and industry within the last few years. Its goal is to extract pieces of knowledge or patterns from usually very large databases. It portrays a robust sequence of procedures or steps that have to be carried out so as to derive reasonable and understandable results [3].

The most predominant representation of the discovered knowledge is the standard production rules (PRs) in the form **If P then D**. The PRs, however, are unable to handle exceptions and do not exhibit variable precision [2]. Much world knowledge is best expressed in the form of hierarchies. Hierarchies give comprehensible knowledge structure that allows us to manage the complexity of knowledge, to view the knowledge at different levels of details, and to focus our attention on the interesting aspects [6]. Automatic generation of hierarchies can be a post-processing step. Crisp description for a concept usually cannot represent human knowledge completely and practically. Automated discovery of fuzzy hierarchical structure from large database plays fundamentally important role in data mining because it provides comprehensible results that capture real-life inheritance of objects. Several efforts [4], [6], [14] have been made in the recent past towards automated discovery of hierarchical structure in large databases.

Bharadwaj and Jain [1] and [2] introduced the concept of Hierarchical Censored Production Rules (HCPRs) by augmenting Censored Production Rules (CPRs) with specificity and generality information. The general form of the HCPR is given as:

Decision **If** <condition>
Unless <censor>
Generality <general info>
Specificity <specific info>.

HCPRs are used to handle trade-off between the precision of an inference and its computational efficiency leading to trade-off between the certainty of a conclusion and its specificity. As a special case (dropping the **Unless** operator) Hierarchical Production Rule (HPR) takes the form:

Decision **If** <condition>
Generality <general info>
Specificity < specific info>

Learning Classifier Systems (LCSs) are rule based classifiers, often called Genetics Based Machine Learning tools, consisting of a set of rules and procedures for performing classifications and discovering rules using genetic and nongenetic operators. The most common applications of LCSs have been from the domain of reinforcement learning (e.g. Markov decision problems [17]). However, the potential for LCS in supervised learning for data mining has been known for some time [5]. The rationale for believing in this potential is based in part on the following observations concerning the following characteristics of LCS [19]:

- LCS have been shown to be capable of learning complex, n-linear classification functions that can be used to accurately predict new cases, on a variety of problem

Suraiya Jabin is a Ph.D. scholar at Dept. of Computer Science, Jamia Hamdard and also working as lecturer in Department of Computer Science, Jamia Millia Islamia, New Delhi, India (phone: +919810822834; e-mail: suraiya224@gmail.com).

Kamal K. Bharadwaj is a professor at the School of Computer and Systems Sciences (SC & SS), Jawaharlal Nehru University (JNU), New Delhi, India (e-mail: kbharadwaj@gmail.com).

domains.

- LCS generalise over the attribute space and under ideal conditions can discover a maximally general, accurate rule set to perform classifications.
- The fact that LCS are rule based means they offer the potential for explanatory data analysis in addition to predictive modeling. In real world data mining exercises being able to explain how a technique forms classifications is often as important as accuracy, and techniques where this is difficult (such as neural networks) are often treated with suspicion in industry.
- Unlike most rule induction algorithms LCS do not discover and evaluate rules in isolation. Instead, they search of prediction accuracy, of a learning classifier system based on the space of possible rule sets defined for a particular problem.
- In addition to being able to form complete classifications LCS can also be used for nugget discovery (the discovery of classifications for some subset of the attribute space). The degree of coverage of the attribute space required can be controlled by careful parameterization.
- The way LCS evaluate rules and rule sets make them ideal for modeling problems where the model may be changing over time, and for maintaining and updating a classification function without the requirement of retraining on all the data.

In this paper we have presented Learning Classifier Systems approach for automated discovery of crisp and fuzzy Hierarchical Production Rule trees.

II. Fuzzy Subsumption Relation

A class Di can be defined by a set of properties (values of distinct attributes), class_prop(Di). Let Di and Dj be any two classes with the set of properties class_prop(Di) and class_prop(Dj), respectively.

First, we define a subsumption (i.e., knowing if a class is an ancestor of another) measure [9] between two attributes $P_i(x)$ and $P_j(y)$ (where x and y are frequencies of Pi and P_j with respect to classes Di and Dj respectively) as follows:

-**subsume**($P_i(x)$, $P_j(y)$)=1 if($x \neq \varnothing$) and ($y \neq \varnothing$) and ($x \leq y$) and (attribute P_i = attribute Pj).

-**subsume**(Pi(x), $P_j(y)$)= y if($x \neq \varnothing$)and ($y \neq \varnothing$)and (x > y) and (attribute P_i = attribute P_j).

-**subsume**($P_i(x)$, $P_j(y)$) =0 if (x =∅) or (y =∅) or (attribute $P_i \neq$ attribute P_j).

Further, we define a degree of subsumption (deg_sub) between two classes Di and Dj as follows, Let α_i is the i-th property in P_{Di} and βj is the j-th property in P_{Dj}.

$$deg\_sub(Di, Dj) = \frac{\sum_{i=1}^{|P_{Di}|} \sum_{j=1}^{|P_{Dj}|} subsume(\alpha_i, \beta_j)}{|P_{Di}|} \quad (1)$$

where $deg\_sub(Di, Dj) \in [0..1]$.

Only the deg_sub with max (deg_sub(Di , Dj), deg_sub(Dj , Di)) $\geq$ threshold will be considered during the construction of fuzzy hierarchy.

Class Dj is specific class of Di ($i \neq j$) if :

- deg_sub(Di, Dj) > deg_sub(D_k , Dj) $\forall$ k $\neq$ i $\neq$ j,
- deg_sub(Di, Dj) $\neq$ deg_sub(Dj , Di), and
- deg_sub(Di, Dj) $\geq$ threshold,

A general rule can be represented as:

P_1 (x1) $\wedge$ P_2 (x2) $\wedge$ ….$\wedge$ P_i (xi) $\wedge$…$\wedge$ P_m (xm) $\rightarrow$ D,

where xi ($1 \leq i \leq m$) denotes the frequency of the property P_i with respect to class D and is computed as follows:

$$x_i = \frac{|P_i \wedge D|}{|D|}, \text{ where } x_i \in [0..1] \quad (2)$$

For example, consider the following two rules:

if P1(1.0) and P2(0.8) then D1

if P1(0.9) and P2(0.6) and P10(1.0) and P11(0.9) then D2

We can compute the deg_sub between class D_1 and D_2 as follows:

$$deg\_sub(D_1, D_2) = \frac{0.9 + 0.6}{2} = 0.75, \text{ and}$$

$$deg\_sub(D_2, D_1) = \frac{1+1}{4} = \quad 0.25$$

If D1 subsumes D2, then D1 is more general than D2. The subsums quantitative measure provides information on whether a class almost subsumes another class [8].

III. Subsumption Matrix

A Subsumption Matrix (SM) that summarizes the relationship between the classes, D1, D2,……..,Dn is an n × n matrix defined as under:

$$SM[D_i, D_j] = deg\_sub(D_i, D_j) \quad (3)$$

SM gives an *ad hoc* insight into the hierarchical relationships present between classes [9].

IV. Learning Classifier System Approach

Successful data mining applications of Learning Classifier Systems have been shown in the past (Bernad_o, Llor_a, & Garrell, 2001) investigating and comparing performance of the accuracy-based Michigan-style LCS XCS (Wilson, 1995) and the Pittsburgh-style LCS GALE (Llor_a & Garrell, 2001). Both systems showed competent performance in comparison to six other machine learning systems. Recently, new systems have appeared in the LCS eld, like the Pitt-style LCS GAssist (Bacardit & Garrell, 2003a).

EpiCS [9] developed by John Homes, based on early works of Wilson was successful application of datamining using LCS. It was designed to use standard production and association rule as underlying knowledge but in our proposed work, designed automated system is using Hierarchical Production Rules as underlying knowledge representation.

John Holmes developed a stimulus-response learning classifier system, BOOLE++ and later, EpiCS which was

based on Wilson's early work on the Animat, BOOLE and NEWBOOLE. Since the designed automated system is completely based on EpiCS, so some more light is thrown on EpiCS. Like other LCS, EpiCS's representation scheme expresses a rule as a condition-action pair, or a *classifier*, where in attributes are encoded as "genes". The left-hand side (condition) of the classifier is commonly referred to as a *taxon*, and the right-hand, the *action*. More commonly, the action is described as the *action bit*, since the type of problems for which one uses EpiCS typically have a single, dichotomous classification, such as dead/alive, diseased/healthy, etc. Classifiers are contained in a *population* of constant size. EpiCS is constructed using the three-part framework of a typical LCS, the performance, reinforcement, and discovery components as shown in Fig. 1.

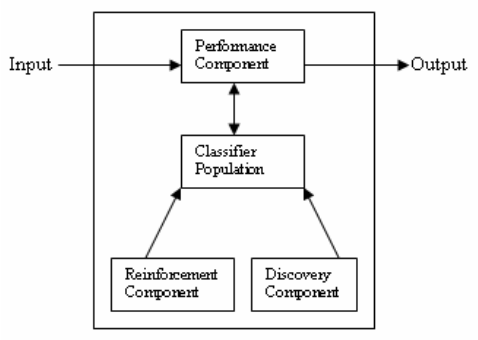

Fig. 1 High – level schematic of EpiCS [9]

A. Performance Component

The *performance component* creates a subset of all classifiers in the population whose taxa (premise part) match a stream of data received as input from the environment. In this way, the performance component is analogous to a *forward chaining rule-base* system. All classifiers in the population whose taxa match the input stream comprise a *Match Set* [M], even though some of these classifiers may advocate different actions. The process is equivalent to the triggering of rules, and [M] is analogous to an agenda in an expert system. From [M], the classifier with the proportionally highest strength is selected. The action of this classifier is then used as the output of the system; this process is analogous to the firing of a rule in an expert system [9].

B. Classifier Population

Since the proposed system is a post processing system, so input to this automated hierarchical production rule system is a set of standard production rules that must be result of some data mining algorithm. Consider the following rule sets shown below:

e.g. a) Crisp Rule Set

- if x_lives_in_city_y then x_is_in_city_y
- if x_lives_in_city_y and time(night) then x_is_at_home
- if x_lives_in_city_y and time(day) then x_is_outside_home
- if x_lives_in_city_y and time(day) and day(working) then x_is_working_outdoor
- if x_lives_in_city_y and time(day) and day(Sunday) then x_is_entertaining_outdoor

e.g. b) Fuzzy Rule Set

- if P1(1.0) then D1
- if P1(1.0) and P2(1.0) then D2
- if P1(0.8) and P3(1.0) and P4(1.0) then D3
- if P1(1.0) and P3(0.9) and P4(0.8) and P5(1.0) then D4
- if P1(1.0) and P3(1.0) and P6(1.0) and P15(1.0) then D5
- if P1(1.0) and P3(1.0) and P6(1.0) and P7(1.0) and P14(1.0) then D6
- if P1(1.0) and P3(1.0) and P4(1.0) and P5(1.0) and P8(1.0) and P9(1.0) then D7
- if P1(1.0) and P2(1.0) and P10(1.0) and P11(0.9) then D8
- if P1(1.0) and P2(1.0) and P12(1.0) then D9
- if P1(1.0) and P2(1.0) and P10(1.0) and P11(0.8) and P13(1.0) then D10

C. Discovery Component

The *discovery component* basically employs the genetic algorithm (GA) for generating HPR-trees. Genetic Algorithm is adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics [7]. It belongs to the class of stochastic search methods. Whereas most stochastic search methods operate on a single solution to the problem at hand, genetic algorithms operate on a population of solutions. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution.

To use genetic algorithm, we represent a solution to the problem as a *genome* (or *chromosome*). The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one(s). The fitness function determines how 'good' each individual is.

Chromosome Representation Method: Each chromosome is a random arrangement of only decision attributes of input production rules in the form of tree or hierarchy. This hierarchical structure is encoded as list representing a general tree:

Tree: (Root (sub-tree 1) (sub-tree 2)…… (sub-tree i)…… (sub-tree k)), where sub-tree i is either empty or has the same structure as ***Tree***. For example a hierarchy and its representation are shown in Fig. 2.

An individual, as hierarchy must satisfy the following condition: $Di \cap Dj = \emptyset$ for any two classes Di and Dj at the same level in the hierarchy. During crossover/mutation operators, if any of the offspring or mutated individuals does not satisfy the above condition, then it will be rejected as an illegal individual [8].

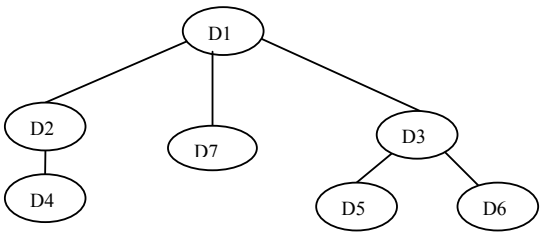

Fig. 2 Hierarchical Production Rule

would be encoded as (D1 (D2(D4))(D7) (D3 (D5) (D6))).

Fitness Function: Once the initial random population has been generated next step is to associate to each solution (hierarchy as chromosome) a value corresponding to the fitness function. The most difficult and most important concept of evolutionary algorithm is the fitness function. It varies greatly from one type of problem to another. However in the context of genetic search we must formulate a single numerical quantity that encapsulates the desirable features [11].

For the proposed system, the fitness measure of an individual is defined as [8]:

$$fitness = \sum_{\forall Di,Dj(i\neq j)} SM[Di, Dj] \quad (4)$$

The wining individual has the highest fitness such that $\forall i, j \;\; Di \xrightarrow{d1} Dj$, where d1 is deg_sub(Di , Dj) ∈ [0,1].

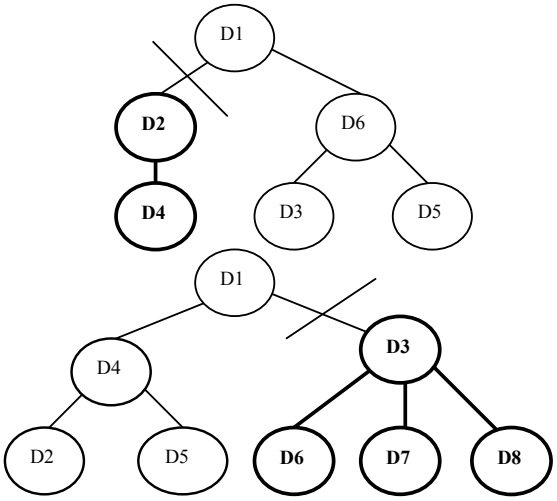

Fig. 3 Two parent chromosomes before crossover

Genetic Operators: We used conventional genetic operators after appropriate modifications that were necessary for our system requirements to generate HPR- trees.

Selection is based on the idea that better individuals get higher chance of selection proportional to their fitness.

Crossover is a key operator for natural evolution. The crossover operator replaces a randomly selected sub-tree of an individual with a randomly chosen sub-tree from another individual and creates new offspring by exchanging sub-trees (i.e., sub-lists) between the two parents. The crossover point was chosen at random for both parents. For example, consider the following two individuals as parents (the "crossover point" is indicated by a tilted line and the sub-trees swapped by crossover are shown in bold):

Parent 1: (D1 **(D2 (D4))** (D6 (D3) (D5)))

Parent 2: (D1 (D4 (D2) (D5)) **(D3 (D6) (D7) (D8))**)

with corresponding hierarchical structure as given in Fig. 3. The two offspring resulting from crossover are:

Offspring 1: (D1 **(D3 (D6) (D7) (D8))** (D6 (D3) (D5))) and

Offspring 2: (D1 (D4 (D2) (D5)) **(D2 (D4))**) are shown below in Fig. 4.

Mutation is the other way to get new chromosomes. It is an operator that acts on a single individual at a time. For mutation a sub-tree/leaf is replaced by randomly chosen sub-tree/ leaf.

D. Reinforcement Component

The idea that we learn by interacting with our environment is probably the first to occur to us when we think about the nature of learning. When an infant plays, waves its arms, or looks about, it has no explicit teacher, but it does have a direct sensorimotor connection to its environment.

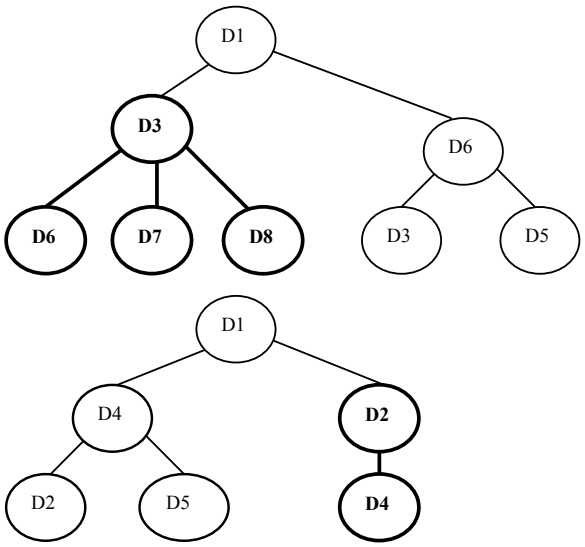

Fig. 4 Two offspring chromosomes after crossover

Reinforcement learning refers to a class of problems in machine learning which postulate an agent exploring an environment in which the agent perceives its current state and takes actions. The environment, in return, provides a reward (which can be positive or negative). Reinforcement learning algorithms attempt to find a policy for maximizing cumulative reward for the agent over the course of solving the problem. After every time step of discovery component, the following **Reward and Punishment Scheme** is applied to evolve the best hierarchy. For each HPR-tree:

Maximum Possible Fitness = total no. of arcs in HPR-tree

e.g. For a tree having total no. of arcs = 10, The Maximum Possible Fitness = 10. And

Difference = Maximum Possible Fitness – Actual Fitness

where Actual Fitness is computed using equation (4).

Now:

i. If **Actual Fitness ≥ 60 %** of the Maximum Possible Fitness, then a reward is given which is some percentage (say 10 %) of Difference.

ii. If **Actual Fitness < 60 %** of the Maximum Possible Fitness, then a tax is applied that is some percentage (10% is taken for the proposed system) of

Difference.

V. Experimental Results

The performance of the proposed system was tested over a no. of examples from which few are demonstrated below. In experiments (a) and (b) the parameters used are as under:

Each iteration of the proposed system consisted of a population of 30 individuals evolving over 500 generations. The proposed algorithm was terminated when the best Fitness did not change continually throughout 10 generations. The probability of crossover and mutation was set to 0.8 and 0.2 respectively. For time efficiency purpose and for avoiding oversized HPR-trees the maximum tree depth and the maximum no. of offspring were fixed and hash defined so that one can change these settings to see the effect over output. The threshold value of degree of subsumption for fuzzy hierarchy was set to 0.60.

Example (a): Consider the crisp rule set mentioned in example (a) of Classifier Population Componenet as input for the proposed system.

Using equation (1) and equation (3) the SM is constructed for the following six classes: D1=x_is_in_city_y; D2=x_is_at_home; D3=x_is_outside_home; D4=x_is_working_outdoor; and D5=x_is_entertaining_outdoor, as shown in Table I.

The proposed system produced the following individual with the highest fitness= 4

(D 1 (D 2) (D 3 (D 5) (D 4)))

The corresponding HPR-tree is shown in Fig. 5.

TABLE I
Subsumption Matrix (5×5)

	D1	D2	D3	D4	D5
D1	1	1	1	1	1
D2	0	1	0	0	0
D3	0	0	1	1	1
D4	0	0	0	1	0
D5	0	0	0	0	1

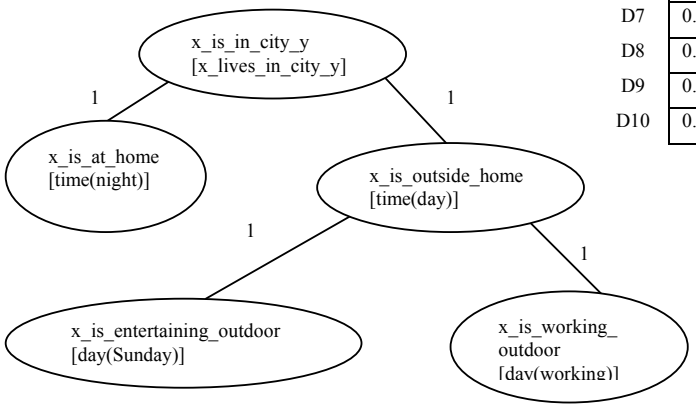

Fig. 5 Crisp HPR-tree with the highest fitness = 4

From the discovered Hierarchy shown in Fig. 5, the following Production Rules with Fuzzy Hierarchy are generated:

LEVEL 0

X_IS_IN_CITY_Y If [X_LIVES_IN_CITY_Y,]
GENERALITY []
SPECIFICITY [X_IS_AT_HOME (1.000000) ,
X_IS_OUTSIDE_HOME (1.000000) ,]

LEVEL 1

X_IS_OUTSIDE_HOME If [TIME (DAY),]
GENERALITY [X_IS_IN_CITY_Y]
SPECIFICITY [X_IS_ENTERTAINING_OUTDOOR (1.000000),
X_IS_WORKING_OUTDOOR(1.000000) ,]

X_IS_AT_HOME If [TIME (NIGHT),]
GENERALITY [X_IS_IN_CITY_Y]
SPECIFICITY[ ]

LEVEL 2

X_IS_WORKING_OUTDOOR If [DAY(WORKING),]
GENERALITY [X_IS_OUTSIDE_HOME]
SPECIFICITY[]

X_IS_ENTERTAINING_OUTDOOR If [DAY (SUNDAY),]
GENERALITY [X_IS_OUTSIDE_HOME]
SPECIFICITY[ ]

Example (b): Consider the fuzzy rule set mentioned in example (b) of Classifier Population Componenet as input for the proposed system.

Using equation (1) and equation (3) the SM is constructed for the nine classes D1 to D9 and is shown below in Table II. The proposed system produced the following individual with the highest fitness = 8.066667

(D1 (D3 (D5 (D6)) (D4 (D7))) (D2 (D8 (D10)) (D9)))

From the discovered Hierarchy shown in Fig. 6, the following Production Rules with Fuzzy Hierarchy are generated:

TABLE II
Subsumption Matrix (10×10)

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
D1	1.00	1.00	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00
D2	0.50	1.00	0.40	0.50	0.50	0.50	0.50	1.00	1.00	1.00
D3	0.33	0.33	1.00	0.90	0.66	0.66	1.00	0.33	0.33	0.33
D4	0.25	0.25	0.70	1.00	0.50	0.50	1.00	0.25	0.25	0.25
D5	0.25	0.25	0.45	0.47	1.00	0.75	0.50	0.25	0.25	0.25
D6	0.20	0.20	0.36	0.38	0.60	1.00	0.40	0.20	0.20	0.20
D7	0.16	0.16	0.46	0.61	0.33	0.33	1.00	0.16	0.16	0.16
D8	0.25	0.50	0.20	0.25	0.25	0.25	0.25	1.00	0.50	0.95
D9	0.33	0.66	0.26	0.33	0.33	0.33	0.33	0.66	1.00	0.66
D10	0.20	0.40	0.16	0.20	0.20	0.20	0.20	0.80	0.40	1.00

LEVEL 0

D1 If [P1]
GENERALITY [ ]
SPECIFICITY [D3 (0.800000) , D2 (1.000000)]

LEVEL 1

D2 If [P2]
GENERALITY [D1]
SPECIFICITY [D8 (1.000000), D9 (1.000000)]

D3 If [P3, P4]
GENERALITY [D1]
SPECIFICITY [D5 (0.666667), D4 (0.900000)]

LEVEL 2

D9 If [P12]
GENERALITY [D2]
SPECIFICITY [ ]

D8 If [P10, P11]
GENERALITY [D2]
SPECIFICITY [D10 (0.950000),]
D4 If [P5,]
GENERALITY [D3]
SPECIFICITY [D7 (1.000000)]
D5 If [P6, P15]
GENERALITY [D3]
SPECIFICITY [D6 (0.750000)]
LEVEL 3
D10 If [P13]
GENERALITY [D8]
SPECIFICITY [ ]
D7 If [P8, P9]
GENERALITY [D4]
SPECIFICITY [ ]
D6 If [P7, P14]
GENERALITY [D5]
SPECIFICITY [ ]

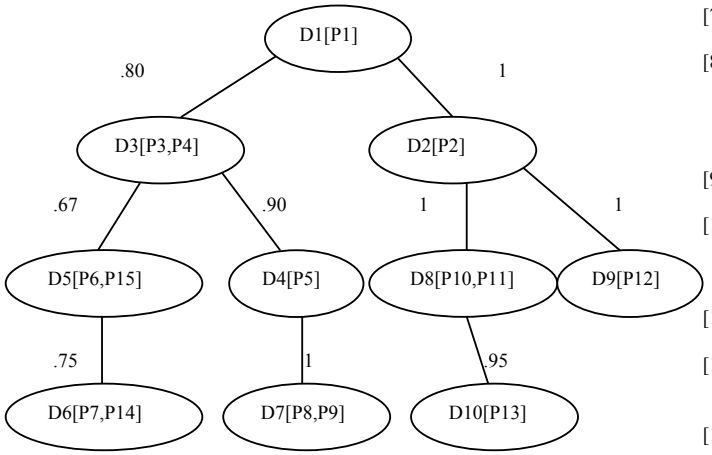

Fig. 6 HPR with the highest fitness = 8.066667

VI. Conclusion

In the present work, LCS approach is proposed as post-processing scheme to organize, summarize and present the discovered rules in the form of Production Rules with Crisp/Fuzzy Hierarchy. An appropriate chromosome representation scheme and suitable genetic operators are proposed. A suitable fitness function is suggested using Subsumption Matrix (SM) based on proposed fuzzy subsumption relation. Also an appropriate domain specific reward and punishment scheme is suggested. Experimental results both for synthetic and real life data sets are quite interesting. One of the most important extensions of the present work would be discovery of Hierarchical Censored Production Rules (HCPRs) [1], [2] and [6] from large datasets using Learning Classifier Systems Approach.

Acknowledgment

Ms. Deepa Anand is a name that deserves special mention; she has really helped a lot during implementation of the proposed system in this paper.

References

[1] Bharadwaj, K. K. and R. Varshneya, "Parallelization of Hierarchical Censored production Rules (HCPRs)", Information and Software Technology, 37, 1995, pp. 453 - 460.
[2] Bharadwaj, K. K. and N.K. Jain, 'Hierarchical Censored production Rules System', Data and Knowledge Engineering, North Holland, vol. 8, page 19 – 34, 1992.
[3] Bruha , Ivan, "Pre – and Post – processing in Machine learning and Data mining", ACAI '99, LNAI 2049, pp. 258 – 266, 2001, Copyright – Verlag Berlin Heidelberg 2001.
[4] Han, J. and Y. FU, "Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases," *AAAI'94 Workshop Knowledge in Databases (KDD'94)*, Seattle, WA, 1994, pp. 157-168.
[5] Holland, J.H. (1986a). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, Carbonell, & Mitchell (eds) *Machine learning, an artificial intelligence approach*. Morgan Kaufmann.
[6] B. Liu, M. Hu, and W. Hsu, "Multi-level organization and summarization of the discovered rules," *SIGKDD-2000*, Boston, USA, Aug 20-23, 2000, pp. 208-217.
[7] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
[8] Basheer M. Al-Maqaleh, and K. K. Bharadwaj, "Genetic programming approach for automated discovery of production rules with fuzzy hierarchy," *in Proc. of National Conference on Methods and Models in Computing (NCM2C'2006),* Jawaharlal Nehru University, New Delhi, India, 18-19 December 2006, pp.127-134.
[9] Alwyn Barry, John Holmes, and Xavier Llora, 'Data Mining using Learning Classifier Systems', pages 21 – 23
[10] Jabin, Suraiya and K. K. Bharadwaj, "Learning Classifier Systems Approach for Automated Discovery of Censored Production Rules", In 14. International Enformatika Conference, v14-57, Vol 14 Aug 2006 ISSN 1305 – 5313.
[11] N.J Radcliffe and P.D. Surry, 'Co – operation through hierarchical competition in genetic data mining', *EPCC – TR94 – 09, 1994.*
[12] R. A. Angryk, and F. E. Petry, "Mining multi-level associations with fuzzy hierarchies," *The 2005 IEEE International Conference on Fuzzy Systems*, 2005, pp.785-790.
[13] Lanzi, Pier Luca, Wolfgang Stolzmann, and Stewart W. Wilson, editors. *Learning Classifier Systems. From Foundations to Applications*, volume 1813 of *LNAI*, Berlin, 2000. Springer-Verlag.
[14] Suryanto, H. and P. Compton, "Discovery of class relations in exception structured knowledge bases", ICCS - 2000, Springer, Germany, 2000, pp. 113 - 126.
[15] John H. Holmes, 'Discovering Risk of Disease with a Learning Classifier System', In Thomas Back, editor, *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA97)*. Morgan Kaufmann, 1997.
[16] Jain, N. K. and K. K. Bharadwaj, "*Some learning techniques in hierarchical censored production rules (HCPRs) systems*" International Journal of Intelligent Systems, John Wiley & Sons, Inc.(NY), vol.13,1998, pp.319-344.
[17] Lanzi , P. L. and R. L. Riolo. A roadmap to the last decade of learning classifier research (from 1989 to 1999). In Lanzi et al. [20], pages 33–61.
[18] Sutton, R. S. and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[19] Bagnall , A. J. Bagnall, G. C. Cawley, in 'Learning Classifier Systems for Data Mining: A Comparison of XCS with Other Classifiers for the Forest Cover Data Set'.