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

Genetic Programming Approach to Hierarchical Production Rule Discovery

Basheer M. Al-Maqaleh, and Kamal K. Bharadwaj

***Abstract*—**Automated discovery of hierarchical structures in large data sets has been an active research area in the recent past. This paper focuses on the issue of mining generalized rules with crisp hierarchical structure using Genetic Programming (GP) approach to knowledge discovery. The post-processing scheme presented in this work uses flat rules as initial individuals of GP and discovers hierarchical structure. Suitable genetic operators are proposed for the suggested encoding. Based on the Subsumption Matrix(SM), an appropriate fitness function is suggested. Finally, Hierarchical Production Rules (HPRs) are generated from the discovered hierarchy. Experimental results are presented to demonstrate the performance of the proposed algorithm.

***Keywords*—**Genetic Programming, Hierarchy, Knowledge Discovery in Database, Subsumption Matrix.

I. Introduction

KNOWLEDGE Discovery in Database (KDD) can be defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [16].

The most predominant representation of the discovered knowledge is the standard production rules in the form **If** P **Then** D. As the number of rules becomes significant, they are not comprehensible to people as meaningful knowledge, from which they can gain insight into the basis of decision-making. 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. Several efforts have been made in the recent past towards automated discovery of hierarchical structure in large data bases [3]-[9].

There has been increasing interest in applying evolutionary computation methods [11]-[15] as data mining tasks to KDD. In Genetic Programming (GP) [10],[14], the basic idea is the evolution of a population of “program”, i.e., candidate solutions to the target problem. A program (an individual of the population) is usually represented as a tree, where the internal nodes are the functions (operators) and the leaf nodes are terminal symbols. Both the function set and terminal set must include symbols suitable for the target problem. Each individual of the population is assessed regarding its ability to solve the target problem. This evaluation is conducted by a fitness function, which is problem-dependent. Individuals undergo the action of genetic operators such as reproduction, crossover and mutation. Once genetic operators have been applied to the population based on given probabilities, a new generation of individuals is created. These newly created individuals are evaluated by the fitness function. The whole process is repeated iteratively for a fixed number of generations or until other termination criterion is met. The result of GP (the best solution found) is usually the fittest individual created along all the generations [14].

In the present work, a post-processing scheme based on GP is presented that takes flat rules as input and discovers crisp hierarchical structure. Further, Hierarchical Production Rules(HPRs)[2], are generated using the discovered hierarchy. A concept of Subsumption Matrix (SM) is used to summarize the relationship between the classes.

An appropriate encoding, suitable genetic operators and effective fitness function are suggested for the proposed scheme.

II. Background

Bharadwaj and Jain [1]-[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>.

These 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) HPR takes the form:

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

Basheer Mohamad Al-Maqaleh is a Ph.D scholar at School of Computer and Systems Sciences (SC&SS), Jawaharlal Nehru University (JNU), New Delhi, India, on leave from Thamar University, Republic of Yemen. Mobile +91-9811515462; (e-mail: bmaa100@yahoo.com).

Kamal K. Bharadwaj, is a professor at the SC&SS, JNU, New Delhi, India.(e-mail: kbharadwaj@gmail.com).

As an example, consider the following HPRs [1]:

level 0
change_car_status **If** [obstacle_ahead]
Generality [ ]
Specificity [use_breaks, turn_off_road]
level 1
use_breaks **If** [speed_distance_ratio_high]
Generality [change_car_status]
Specificity [ ]
turn_off_road **If** [not_on_bridge]
Generality [change_car_status]
Specificity [ ].

The above related HPRs form a tree (called HPR-tree) giving the following hierarchical structure Fig.1.

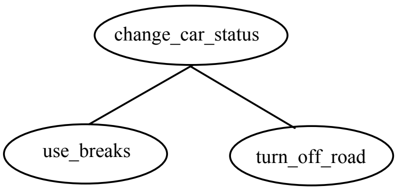

Fig. 1 HPR-tree (Hierarchy)

III. Subsumption Matrix

A class Di can be defined by 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.

We can define the degree of subsumption (deg_sub(Di, Dj)) as follows:

$$\text{deg\_sub(Di , Dj)} = \frac{\left|\text{class\_prop (Di)} \cap \text{class\_prop (Dj)}\right|}{\left|\text{class\_prop (Di)}\right|} \qquad (1)$$

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

A SM that summarizes the relationship between the classes, D1, D2,…….,D_n is an n × n matrix defined as under:

$$\text{SM[Di, Dj]} = \begin{cases} 1 & \text{if deg\_sub(Di, Dj)} = 1 \\ & \text{i.e., Di} \subseteq \text{Dj} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

IV. Genetic Programming Approach

As a post-processing scheme, we are using GP to discover crisp hierarchical production rules from the flat rules as input. The details of encoding, genetic operators and the fitness function for the proposed scheme are discussed in the following subsection:

A. Encoding

A 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 the hierarchy in Fig.2.

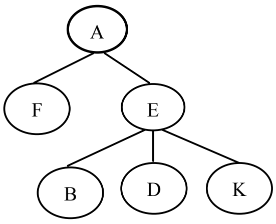

Fig. 2 Hierarchy

would be encoded as (A (F) (E (B) (D) (K))).

An individual, as hierarchy must satisfy the following condition: Di$\cap$Dj = Ø 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.

B. Genetic Operators

The new elements in the population are generated by means of three operators: reproduction, crossover and mutation.

a) Reproduction

The reproduction operator selects one individual of the present population in proportion to its fitness value, so that the fitter an individual is the higher the probability that it will take part in the next generation of individuals. After selection, the individual is copied into the new generation without any modifications. Reproduction reflects the principle of natural selection and survival of the fittest [14].

b) Crossover

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: (A **(B)** (C (H) (N)))
Parent 2: (A (F) **(E (B) (D) (K))**)

with corresponding hierarchical structure is given in Fig.3.

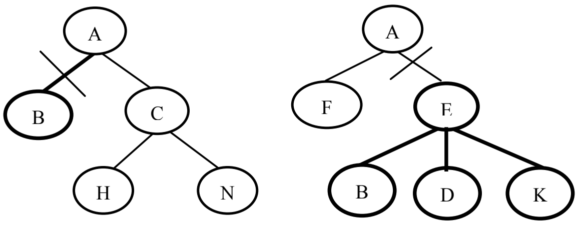

Fig.3 Two parents before crossover

The two offspring resulting from crossover are:
Offspring 1: (A **(E (B) (D) (K))** (C (H) (N))) and
Offspring 2: (A (F) **(B)**) are shown below in Fig.4.

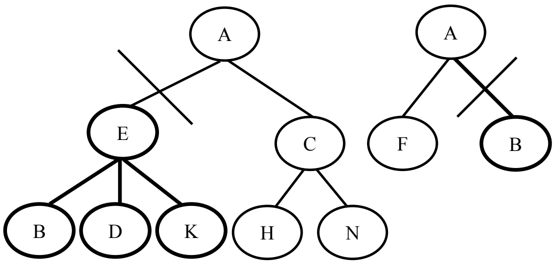

Fig. 4 Two offspring produced by crossover

c) *Mutation*

For the tree mutation a sub-tree/leaf is replaced by randomly chosen sub-tree/ leaf.

C. *Fitness Function*

The fitness function evaluates the quality of an individual in the population. For the proposed algorithm, the fitness measure of an individual is defined as:

$$fitness = \sum_{\forall\, Di,Dj} \mathrm{SM}\left[Di, Dj\right] \qquad (3)$$

The wining individual has the highest *fitness* such that $\forall$ i, j $Di \xrightarrow{1} Dj$.

V. Experimental Results

Each GP run consisted of a population of 20 individuals evolving over generations. The probability of crossover, reproduction and mutation set to 0.8, 0.1 and 0.1, respectively, and the selection method used for both parents was fitness proportionate. For the practical reason of avoiding the expenditure of large amounts of computer time on occasional oversized programs, the depth of initial programs was limited to 6 and during the run the maximum tree depth was set to 10.

Example 1: consider the following five flat rules as input for the proposed algorithm:

If x _lives_in_city _y Then x_is_ in_city_y
If x_ lives_in_city_ y $\wedge$ time(night) Then x_is_at_home
If x_ lives_in_city_ y $\wedge$ time(day) Then x_is_ outside_home
If x_lives_in_city_y $\wedge$ time(day) $\wedge$ day(working) Then x_is_ working_outdoor
If x_lives_in_city_y $\wedge$ time(day) $\wedge$ day(Sunday) Then x_is_ entertaining_outdoor.

Using (1) and (2) the SM is constructed for the five classes D1=x_is_in_city_y ; D2=x_is_at_home ; D3=x_is_outside_home ; D4=x_is_working_outdoor ; D5=x_is_entertaining_outdoor , as shown below (see TABLE I).

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

The proposed algorithm produced the following individual with the highest *fitness* = 4 :
(D1(D2)(D3(D4)(D5))).

The corresponding hierarchy is shown in Fig.5.

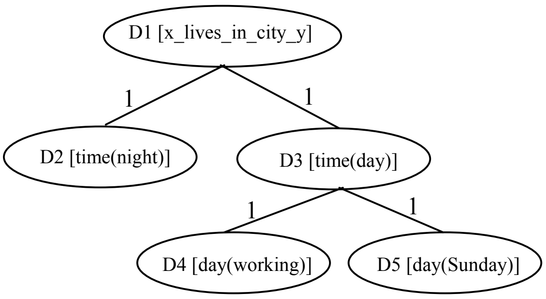

Fig. 5 Hierarchy-the individual with the highest *fitness*=4

From the discovered hierarchy shown in Fig.5, the following HPRs are generated:

level 0
D1 **If** [x_lives_in_city_y]
Generality [ ]
Specificity [D2,D3]

level 1
D2 **If** [time (night)]
Generality [D1]
Specificity [ ]
D3 **If** [time (day)]
Generality [D1]
Specificity [D4 ,D5]

level 2
D4 **If** [day (working)]
Generality [D3]
Specificity [ ]
D5 **If** [day (Sunday)]
Generality [D3]
Specificity [ ]

Example 2: suppose we have 10 flat rules with 10 different classes as follows:

If P1 $\wedge$ P2 $\wedge$ P5 Then D1
If P1 $\wedge$ P2 $\wedge$ P4 $\wedge$ P7 Then D2
If P1 $\wedge$ P2 $\wedge$ P4 Then D3

If P1 $\wedge$ P2 Then D4
If P1 $\wedge$ P2 $\wedge$ P5 $\wedge$ P6 Then D5
If P1 $\wedge$ P2 $\wedge$ P3 Then D6
If P1 $\wedge$ P2 $\wedge$ P4 $\wedge$ P8 $\wedge$ P11 $\wedge$ P12 Then D7
If P1 $\wedge$ P2 $\wedge$ P4 $\wedge$ P8 $\wedge$ P9 $\wedge$ P10 Then D8
If P1 $\wedge$ P2 $\wedge$ P4 $\wedge$ P8 Then D9
If P1 $\wedge$ P2 $\wedge$ P4 $\wedge$ P8 $\wedge$ P13 $\wedge$ P14 $\wedge$ P15 Then D10.

Finally, the proposed algorithm produced the following individual with the highest *fitness* = 9: (D4(D6)(D3(D2)(D9(D7)(D8)(D10)))(D1(D5))).

The corresponding hierarchy is shown in Fig.6.

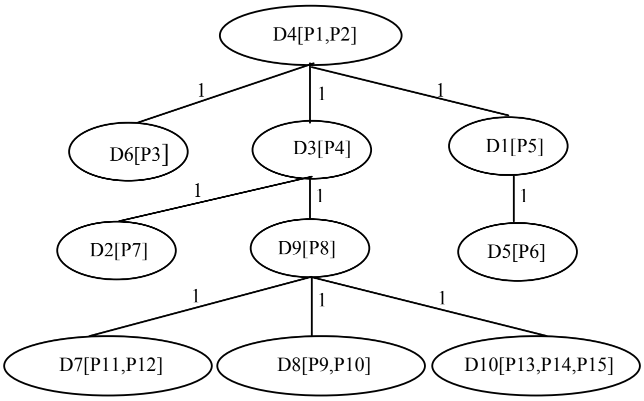

Fig.6 Hierarchy-the individual with the highest *fitness*=9

From the discovered hierarchy shown in Fig.6, the following HPRs are generated:

level 0
D4 **If** [P1, P2]
Generality [ ]
Specificity[D6,D3, D1]

level 1
D6 **If** [P3]
Generality [D4]
Specificity [ ]
D3 **If** [P4]
Generality [D4]
Specificity [D2, D9]
D1 **If** [P5]
Generality [D4]
Specificity [D5]

level 2
D2 **If** [P7]
Generality [D3]
Specificity [ ]
D9 **If** [P8]
Generality [D3]
Specificity[D7,D8,D10]
D5 If [P6]
Generality [D1]
Specificity [ ]

level 3
D7 **If** [P11, P12]
Generality [D9]
Specificity [ ]
D8 **If** [P9, P10]
Generality [D9]
Specificity [ ]
D10 **If** [P13, P14, P15]
Generality [D9]
Specificity [ ]

VI. Conclusion

As an attempt towards automated generation of hierarchies, a GP approach is proposed to organize, summarize and present the discovered rules in the form of HPRs. Suitable genetic operators are proposed for the suggested encoding. Based on the SM, an appropriate *fitness* function is suggested. Performance of the proposed algorithm is demonstrated through experimental results, which are quite encouraging.

Development of a GP based algorithm for the automated discovery of Production Rules with Fuzzy Hierarchy is under progress. One of the most important future research directions would be the discovery of HPRs with exceptions from large databases using GP approach.

References

[1] K. K. Bharadwaj and R. Varshneya, "*Parallelization of hierarchical censored production rules*," Information and Software Technology, 37, 1995, pp.453-460.

[2] K. K. Bharadwaj and N. K. Jain, "*Hierarchical censored production rules (HCPRs) systems*," Data and Knowledge Engineering, North Holland, vol. 8, 1992, pp.19-34.

[3] S. Levachkine and A. Guzman-Arenas, "*Hierarchies measuring qualitative variables,*" Springer-Verlag Berlin Heidelberg 2004, A. Gelbukh (Ed.):CICLing 2004,2004,pp.262-274.

[4] J. Han, 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, July 1994, pp. 157-168.

[5] H. Surynato and P. Compton, "*Learning classification taxonomies from a classification knowledge based system*," Proceedings the First Workshop on Ontology Learning in Conjunction with ECAI-2000, Berlin, pp.1-6.

[6] B. Liu, M. Hu, and W. Hsu, "*Multi-level organization and summarization of the discovered rules*," Boston, USA, SIGKDD-2000, Aug 20-23, 2000.

[7] D. Richards and U. Malik, "*Multi-level rule discovery from propositional knowledge bases*," International Workshop on Knowledge Discovery in Multimedia and Complex Data (KDMCD'02), Taipei, Taiwan, May 2002, pp.11-19.

[8] R. Srikant, Q. Vu and R. Agrawal, "*Mining association rules with item constraints*," in Proc of the 3rd International Conf on Knowledge Discovery and Data Mining (KDD'97), 1997, pp.67-73.

[9] M. Suan, "*Semi-Automatic taxonomy for efficient information searching*," Proceeding of the 2nd International Conference on Information Technology for Application (ICITA-2004), 2004.

[10] J. R. Koza, "*Genetic programming: on the programming of computers by means of natural selection*," MIT Press, 1994.

[11] A. A. Freitas, "*A survey of evolutionary algorithms for data mining and knowledge discovery*," In: A. Ghosh, and S. Tsutsui (Eds.) Advances in Evolutionary Computation, Springer-Verlag, 2002.

[12] I. De Falco, A. Della Cioppa, and E. Tarantiono, "*Discovering interesting classification rules with genetic programming*," Applied Soft Computing, 1, 2002, pp.257-269.

[13] M. V. Fidelis, H. S. Lopes, and A. A. Freitas, "*Discovering comprehensible classification rules with a genetic algorithm*," Proc. Congress on Evolutionary Computation-2000 (CEC'2000), La Jolla, CA, USA,IEEE, July 2000, pp.805-810.

[14] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, " *Genetic programming for knowledge discovery in chest pain diagnosis*," IEEE Engineering in Medical and Biology magazine-special issue on data mining and knowledge discovery, 19(4), July/Aug 2000,pp.38-44.

[15] M. C. J. Bot and W. B. Langdon, " *Application of genetic programming to induction of linear classification trees*," Genetic Programming: Proceedings of the 3rd European Conference (EuroCP'2000), Lecture Notes in Computer Science 1802, Springer, 2000, pp.247-258.

[16] U.M. Fayyad, G.P. Shapiro, and P. Smyth, "*The KDD process for extracting useful knowledge from volumes of data*," Communication of ACM. Nov, 1996, vol. 39 (11), pp.27-34.