Learning acyclic rules based on Chaining Genetic Programming

Wing-Ho Shum, Kwong-Sak Leung
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong
{whshum, ksleung}@cse.cuhk.edu.hk

Man-Leung Wong
Dept. of Computing & Decision Sciences
Lingnan University
Hong Kong
mlwong@ln.edu.hk

Abstract

Multi-class problem is the class of problems having more than one classes in the data set. Bayesian Network (BN) is a well-known algorithm handling the multi-class problem and is applied to different areas. But BN cannot handle continuous values. In contrast, Genetic Programming (GP) can handle continuous values and produces classification rules. However, GP is possible to produce cyclic rules representing tautologic, in which are useless for inference and expert systems. Co-evolutionary Rule-chaining Genetic Programming (CRGP) is the first variant of GP handling the multi-class problem and produces acyclic classification rules [16]. It employs backward chaining inference to carry out classification based on the acquired acyclic rule set. It can handle multi-classes; it can avoid cyclic rules; it can handle input attributes with continuous values; and it can learn complex relationships among the attributes. In this paper, we propose a novel algorithm, the Chaining Genetic Programming (CGP) learning a set of acyclic rules and to produce better results than the CRGP's. The experimental results demonstrate that the proposed algorithm has the shorter learning process and can produce more accurate acyclic classification rules.

1 Introduction

BN is a powerful network model, which represents a set of attributes in the multi-class problem and provides the probabilistic relationships among them. But it cannot handle continuous values, they must be discretized and thus the order information is lost [4, 14]. Heckerman et al. proposed methods of learning a network that contains Gaussian distributions [5]. Monti et al. used neural networks to represent the conditional densities [13].

GP is a branch of evolutionary computation (EC). It is applied to different areas, like shortest path finding, pattern recognition, classification and clustering [3]. Teredesai el al. described the Recurrent Genetic Programming handling the on-line handwritten classification problem [17]. Kishore et al. proposed a GP model to solve the multi-category pattern classification problem [9]. Wilson's XCS [18], XCSI [19] and sUpervised Classifier System [2] are well-known classification algorithms based on learning classifier system. However, all of them cannot produce acyclic rules.

Acyclic rules are important and necessary for inference and expert systems. No meaningful result can be inferred from cyclic rules and they represent tautologic only. An example of cyclic rules is,

if class 0 = 1, then class 2 = 3
if class 3 = 0, then class 0 = 1
if class 2 = 3, then class 3 = 0

we cannot infer the class 2's values, as they ultimately depend on themselves though a cycle of rules.

Shum et al. proposed the CRGP, which is the first EC algorithm handling the multi-class problem [16]. It learns a set of acyclic classification rules representing the relationships among the attributes and it can handle input attributes with continuous values.

In this paper, we propose the CGP to solve the multi-class problem. Compared with the CRGP, it can learn more accurate acyclic rules and has the shorter learning process. The remaining parts of the paper are organized as follows. We describe the proposed algorithm in the next section. The experimental results are presented in Section 3. The summary appears in the last section.

2 The Proposed Algorithm

The problem addressed in this study is learning classification rules without cycle in the inference process for the multi-class problem. Each attribute of the problem is regarded as either an input attribute or a class. The relationships among the attributes will be represented by rules.

The CGP is a GP based on backward chaining [8]. It has a population of rules. Unlike the CRGP, it has no migration; it has no epoch; it has a co-operating scoring stage; and it does not infer the classes' values during the learning process. The CGP has the lower computation complexity and produces more accurate acyclic rules. The population is divided into sub-populations, in which increase and maintain the diversity of rules. Backward chaining can find out cycles through the inference process and cyclic rules have to be avoided.

2.1 Initialization of Population

The CGP employs the Michigan approach [7]. It starts with a population, which in turn consists of n sub-populations of rules, where n is equivalent to the number of classes. Each sub-population is assigned to learn a different class. For instance, $sub-population_0$ learns $class\ 0$, $sub-population_1$ learns $class\ 1$ and so on.

The rules represent the relationships among the attributes. They are of the form, $< antecedent > \rightarrow < consequent >$. The $< antecedent >$ and $< consequent >$ are in the prefix form. The function set is defined as $F = \{\wedge, >, \leq, =, \neq\}$. An example of rule is $< \wedge(= class1\ 1)(> attribute2\ 49) > \rightarrow <= class2\ 0 >$.

The sub-populations have the same and fixed number of rules and they are initialized randomly.

2.2 Fitness Evaluation

The CGP uses a support-confidence based fitness function and token competition to evaluate the rules' fitness [1, 10]. The support measures the coverage of a rule. It is the ratio of the number of data items classified correctly by the rule to the total number of data items. The confidence factor (cf) is the confidence of the consequent part to be true when the antecedent part is satisfied, which reflects the rule's classification accuracy. It is calculated as the ratio of the number of data items satisfying both of the antecedent and consequent parts to the number of data items satisfying the antecedent part only. The fitness function is defined as follows:

$$fitness = \begin{cases} support, & if\ support < min \\ support + cf\_part, & otherwise \end{cases} \quad (1)$$

and cf_part is

$$cf\_part = cf * log(\frac{cf}{prob}) \quad (2)$$

where *cf is both/consq*, *prob is consq/total*, min is the user-defined threshold, $both$ is the number of data items satisfying both of the antecedent and consequent parts, $consq$ is the number of data items satisfying the consequent part only and $total$ is the total number of data items.

2.3 Co-operation and Co-evolution

The sub-populations co-operate together through backward chaining. Backward chaining is a well-known inferential methodology. Given a class, some facts and some rules, it forms backward chains of rules and proves if the class can be satisfied.

Co-operating scoring stage occurs after the fitness evaluation and the cyclic relationships in the rules will be detected and eliminated. The rules are assessed if they can well co-operate with the others and co-operating scores are then assigned. Those rules form no cycle with the others and produce accurate inferential results will have the higher co-operating scores.

Tables 1 and 2 show the pseudocodes of the co-operating scoring and backward chaining procedures respectively. The co-operating score is calculated as follows:

$$co-operating\_score = \sum_k fit_k \quad (3)$$

where fit_k is the fitness value of the k^{th} rule in the backward chain of rules.

In the beginning, all the rules' co-operating scores are set to 0. They are then assessed one by one. 1) The rule being assessed becomes the first and only rule in the backward chain of rules; 2) if the rule contains one or more classes in the $< antecedent >$, a copy of the backward chaining procedure will be invoked for each of the classes; 3) the backward chaining procedure selects the rules inferring the class value that is being looking for from the corresponding sub-population and collect them as a pool; 4) in the pool, a rule forming no cycle with the others in the backward chain is selected by Roulette Wheel method, according to their fitness value [11]. Higher fitness value, higher chance to be selected; 5) if the selected rule further contains one or more classes in the $< antecedent >$, another copy of the backward chaining procedure will be invoked for each of the classes; 6) if no rule in the pool can form no cycle with the others in the backward chain, the rule selection process will be ended; 7) after finished all the copies of the backward chaining procedure, the co-operating score of the backward chain is calculated by the co-operating scoring function, Equation 3. It is the sum of the fitness values of all the rules in the backward chain; and 8) check all the rules in the backward chain if their co-operating score is smaller than the backward chain one. If so, their co-operating score will be updated to the latter one.

The backward chaining procedure selects the better rules, instead of the best one to form backward chain. Higher fitness value higher chance to be selected. If the backward chaining procedure always selects the best rule, the learning process would likely to stick into a local optimal and the learning performance is likely heavily degraded

Table 1. The pseudocode of the Co-operating Scoring Procedure.

1. set i = 0.
2. while $i <$ the number of sub-populations,
2.1 set j = 0.
2.2 while $j <$ the number of rules of $sub - population_i$,
2.2.1 set the co-operating score of the j^{th} rule in $sub - population_i$ = 0
2.2.2 j = j + 1.
2.3 i = i + 1.
3. set i = 0.
4. while $i <$ the number of sub-populations,
4.1 set j = 0.
4.2 while $j <$ the number of rules of $sub - population_i$,
4.2.1 the j^{th} rule in $sub - population_i$ becomes the first and only rule in the backward chain of rules.
4.2.2 if the j^{th} rule contains one or more classes in its $< antecedent >$,
4.2.2.1 a copy of the backward chaining procedure is invoked for each of the classes.
4.2.3 calculate the co-operating score of the backward chain by the co-operating scoring function.
4.2.4 set k = 0.
4.2.5 while $k <$ the number of rules in the backward chain.
4.2.5.1 if the co-operating score of the k^{th} rule in the backward chain $<$ the co-operating score of the backward chain
4.2.5.1.1 set the co-operating score of the k^{th} rule in the backward chain = the co-operating score of the backward chain
4.2.5.1.2 k = k + 1.
4.2.6 j = j + 1.
4.3 i = i + 1.

Table 2. The pseudocode of the Backward Chaining Procedure.

input: the class and the class value being looking for
the backward chain of rules

1. set i = 0
2. set j = the class number of the class being looking for.
3. set the pool of rules = $\{\}$.
4. while $i <$ the number of rules of $sub - population_j$,
4.1 if the i^{th} rule in $sub - population_j$ infers the class value being looking for,
4.2 add the i^{th} rule into the pool.
4.3 i = i + 1.
5. if the pool $\neq \{\}$,
5.1 select a rule from the pool by Roulette Wheel method according to the fitness value
5.2 remove the selected rule from the pool.
5.3 while the selected rule form cycle with the others in the backward chain and the selected rule $\neq$ NULL
5.3.1 if the pool $\neq \{\}$,
5.3.1.1 select another rule from the pool by Roulette Wheel method, according to the fitness value
5.3.1.2 remove the selected rule from the pool.
5.3.2 else
5.3.2.1 set the selected rule to NULL.
5.4 if the selected rule $\neq$ NULL,
5.4.1 the selected rule is appended to the backward chain.
5.4.2 if the selected rule further contains one or more classes in the $< antecedent >$,
5.4.2.1 another copy of backward chaining procedure is invoked for each of the classes

<= class0 1>→<= class2 1> fitness value = 1.3
<= class3 4>→<= class0 1> fitness value = 0.8
<= class1 2>→<= class3 4> fitness value = 0.71
<= class5 1>→<= class1 2> fitness value = 1.22
<= class4 2>→<= class5 1> fitness value = 0.6

co-operating score = 4.63

<= class3 3>→<= class2 2> fitness value = 1.2
<= class1 0>→<= class3 3> fitness value = 0.9
<= class4 1>→<= class1 0> fitness value = 0.8

co-operating score = 2.90

Figure 1. Examples of backward chain of rules.

by an accidentally poor performed sub-population. Figure 1 shows two examples of backward chain of rules. The left backward chain is longer and its rules have higher fitness values, thus the co-operating score of the backward chain is higher. The only condition stopping the right backward chain to grow is cycle. Since there is no suitable rule that would form no cycle with the others in the backward chain, the chain is stopped to grow and thus has smaller co-operating score.

The co-operating score of the backward chain is the sum of the fitness values of all the rules in the chain, which reflects the chain's overall performance. More better rules higher the co-operating score. The rules only keep one co-operating score, regardless the number of backward chains that they have participated in, i.e. only the highest one is kept.

Therefore, the rules' co-operating scores memorize the best overall performance that the rules have achieved with the others. Those rules that have participated in a long backward chain of better rules will have the higher co-operating scores. In result, the co-operating scoring stage encourages well co-operated acyclic rules.

The sub-populations co-evolve together. Initially, the sub-populations only have poor-performed rules, so it is difficult to form backward chain with higher co-operating score. As the learning process goes on, the quality of the rules improves and better backward chain becomes possible.

2.4 Genetic Operators and Reproduction

The CGP employs four canonical genetic operators of GP to reproduce offsprings; they are the crossover, the mutation, the reproduction and the dropping conditions [12]. Firstly, the rules are selected by Roulette Wheel method, according to their co-operating scores [11]. Higher co-operating score higher chance to be selected. Secondly, the selected rules are applied with one of the genetic operators, to reproduce offsprings. The rules can breed with the others in the same sub-population only. Then, all the rules and offsprings in the same sub-population compete with each other. The best half of them is preserved as the new sub-population for the next generation.

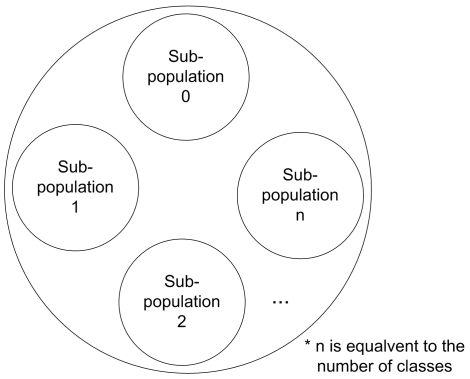


Figure 2. The architecture of the CGP.

When the maximum number of generations is met, the learning process is terminated and all of the rules in the population are collected together as the resultant rule set. Figures 2 and 3 show the architecture and the flow of the CGP respectively.

3 Experiments

3.1 Experimental setting

We compared the CGP with the CRGP, the Multi-population Genetic Programming (MGP) and the GP. The maximum number of generations of the CGP is equalvent to the product of the number of generations in an epoch and the number of epochs of the CRGP. The MGP is a multi-population GP. It has no migration and no backward chaining. The GP is a canonical GP, the number of individuals in the population is equalvent to the total number of individuals of all populations of the CRGP and the maximum number of generations is equalvent to the product of the number of generations in an epoch and the number of epochs of the CRGP. In short, all the algorithms have the same total num-

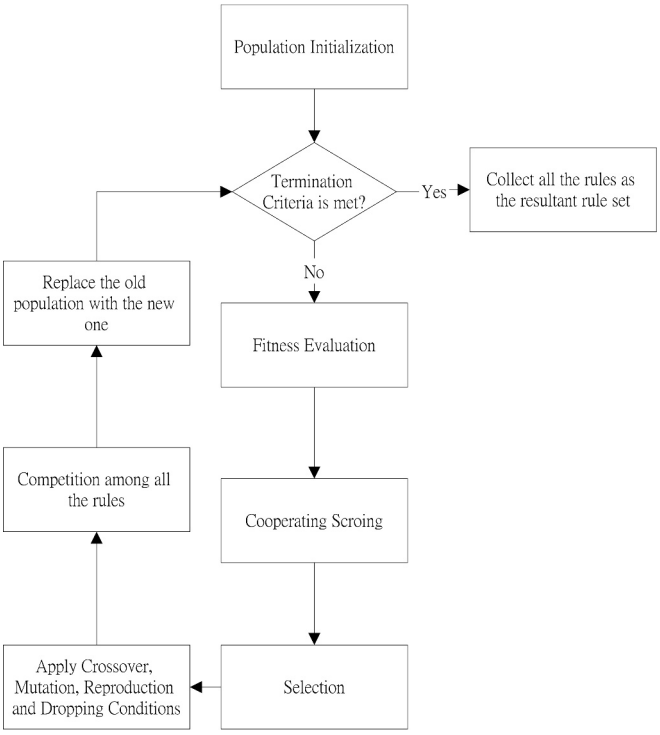


Figure 3. The flow of the CGP.

ber of generations and the same total number of individuals. We implemented the CGP, the CRGP, the MGP and GP in C++. They have the same implementation details, operating parameters and operating environment.

3.2 Data Sets

We evaluated the CGP on two real-life data sets. The first one, "Asia" is from UCI Machine Learning Repository [6]. It has 7 attributes and 1000 data items. Figure 4 shows the relationships among the attributes of the Asia. The second one, "Fracture" is from the Orthopaedie Department of the Prince of Wales Hospital of Hong Kong. Figure 5 shows the relationships among the attributes of the Fracture. It consists of records of children with limb fractures admitted to the hospital in the period 1984-1996. The Fracture was used in [15]. It has 6 attributes and 4884 data items. The data items were originally recorded in text format, we pre-processed them to remove the typo-errors and those data items having missing value were removed, since it is not viable to replace them with average values.

3.3 Evaluation Methods

The algorithms were evaluated in term of the inferential accuracy. After finished the learning process, the rules pro-

Figure 4. The relationships among the attributes of the Asia.

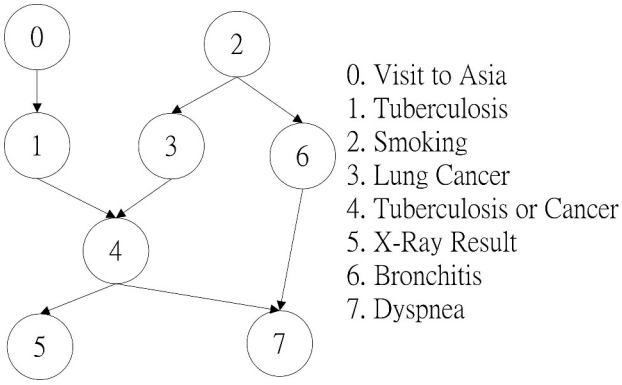
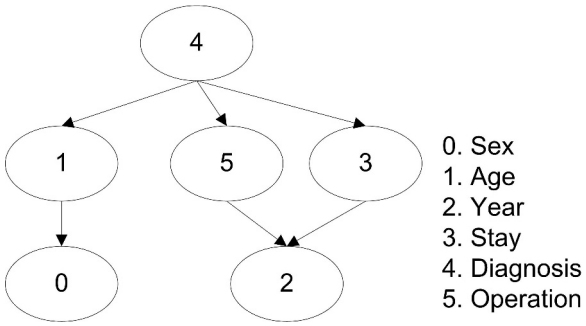


Figure 5. The relationships among the attributes of the Fracture.



duced by the algorithms were collected together as the resultant rule set as the typical evaluation method of EC [15]. Then the rule sets were used to infer the values of the classes though the backward chaining inference, in other words, the classes' values were inferred by the input attributes' values though the inference process. Finally, we compared the inferred values with the real ones in the data sets and calculated the inferential accuracies. We also compared the inferential accuracies among the algorithms by T-test. Smaller T-test values more significance of the improvements (and less chances were caused by stochastic effect). The T-test values were calculated by Microsoft Excel 2003 SP1.

Since cyclic rules cannot be fired in the inference process (and expert systems) and the MGP, the GP and the other algorithms are possible to produce cyclic rules, their inferential accuracies should be lower than the CGP's and the CRGP's.

We compared the computation times between the CGP and the CRGP as well. Since they are different in architecture and flow, we cannot compare their learning performance in term of the number of evaluations. Instead, the number of seconds of a run is used.

The data sets were split into two parts. 66% of the data items were used for the learning, the rest of them were used for the evaluation. For all of the algorithms, we specified the input attributes. All the results are based on 10 independent runs.

3.4 The Asia results

The values of the number of epochs, the maximum number of generations, the number of rules, the maximum depth of a rule, the crossover rate, the mutation rate, the reproduction rate and the dropping condition rate are 100, 10, 10, 15, 0.6, 0.4, 0 and 0.1 respectively.

Tables 3 and 4 show the inferential accuracy results and the comparisons respectively. The values in between parenthesis are the best results of the 10 independent runs. The first columns in the tables show the sum of the inferential accuracies of all the classes. They show both of the CGP and the CRGP outperformed than the others, especially for classes 1, 2 and 3. It was because both of the MGP and the GP produced a lot of cyclic rules which could not be fired in the inference process (and expert systems). Furthermore, as their rules are difficult to form longer and better backward chains of rules and infer all the classes' values then, they cannot achieve better inferential accuracy for all of the classes simultaneously, but some of them. Different runs very different inferential accuracies for the classes, i.e. their results are very fluctuated. In result, their best and average results are quite different with each other. Besides, the CGP did better than the CRGP, it got 22% improvement over the CRGP on the sum of the inferential accuracies.

Table 9. Computation time comparison results between the CGP and the CRGP.

	Asia	Fracture
CGP (second)	58.49	66.74
CRGP (second)	197.36	364.94
Improvement(%)	70.36	81.71
T-test	$5.70 * 10^{-3}$	$2.54 * 10^{-4}$

Table 5 shows the T-test comparison results. As the t-test values are very small, the improvements of the CGP over the others are significant.

3.5 The Fracture results

The values of the number of epochs, the maximum number of generations, the number of rules, the maximum depth of a rule, the crossover rate, the mutation rate, the reproduction rate and the dropping condition rate are 100, 10, 10, 15, 0.4, 0.4, 0.1 and 0.1 respectively.

Tables 6 and 7 show the inferential accuracy results and the comparisons respectively. Similar to the ones of the Asia, they show both of the CGP and the CRGP did better than the others, for all of the classes and the CGP got better than the CRGP as well. Table 8 shows the T-test comparison results. It shows the improvements of the CGP are significant and were not simply caused by stochastic effect.

Table 9 shows the computation time comparisons between the CGP and the CRGP. It shows the CGP got the significant less computation time than the CRGP. Inference is a time consuming process, unlike the CRGP, the CGP does not infer the classes' values during the learning process and thus a large portion of the computation time is saved.

The experimental results show the multi-class problem cannot be solved by traditional classification algorithms, like the MGP and the GP; Shum et al. also demonstrated that even the sophisticated decision tree algorithm, C5.0 cannot manage this problem [16]. They are likely to produce cyclic results representing tautologic. The CGP and the CRGP are the only EC algorithms producing acyclic rules for inference and expert systems. The results also show the CGP outperformed than the CRGP, both in terms of the inferential accuracy and the computation time.

4 Summary

In this paper, we propose a novel algorithm, the CGP to learn a set of acyclic rules. It has the shorter learning process and can produce more accurate acyclic classification rules. Unlike BN, it can handle input attributes with continuous values. We evaluated its performance on two real-life

Table 3. Inferential accuracy results for the Asia.

	sum (%)	class 0 (%)	class 1 (%)	class 2 (%)	class 3 (%)	class 4 (%)	class 5 (%)
CGP	488.52 (496.06)	99.09 (99.09)	94.64 (94.64)	92.73 (92.73)	88.12 (88.18)	53.97 (60.91)	60.97 (61.52)
CRGP	399.64 (495.70)	94.76 (96.70)	72.24 (94.55)	63.79 (90.24)	62.33 (89.70)	54.21 (64.24)	52.30 (57.27)
MGP	313.39 (418.48)	98.67 (99.70)	64.94 (90.60)	34.27 (90.00)	14.85 (14.85)	51.15 (66.36)	49.52 (56.97)
GP	309.61 (421.82)	98.55 (99.09)	66.55 (93.33)	38.36 (92.42)	13.09 (13.64)	43.33 (62.12)	49.73 (61.21)

Table 4. Comparison results between the CGP and the others for the Asia.

	sum (%)	class 0 (%)	class 1 (%)	class 2 (%)	class 3 (%)	class 4 (%)	class 5 (%)
CGP and CRGP	22.24 (0.07)	4.57 (-0.60)	29.61 (-0.96)	45.37 (2.75)	41.37 (-1.69)	-0.44 (-5.10)	16.57 (7.40)
CGP and MGP	55.88 (18.54)	0.43 (-0.06)	44.19 (3.34)	170.56 (3.03)	493.47 (493.87)	5.50 (-8.22)	23.13 (7.98)
CGP and GP	57.78 (17.60)	0.55 (0.00)	40.71 (0.32)	141.70 (0.32)	573.14 (546.67)	24.55 (-1.95)	22.60 (0.50)

Table 5. T-test comparison results between the CGP and the others for the Asia.

	class 0	class 1	class 2	class 3	class 4	class 5
CGP and CRGP	$3.14 * 10^{-3}$	0.08	$5.97 * 10^{-4}$	$1.53 * 10^{-34}$	0.30	$1.97 * 10^{-5}$
CGP and MGP	$2.56 * 10^{-3}$	0.03	$2.46 * 10^{-4}$	$1.21 * 10^{-45}$	0.39	$1.85 * 10^{-4}$
CGP and GP	$3.14 * 10^{-3}$	0.03	$5.79 * 10^{-4}$	$5.59 * 10^{-43}$	0.37	$1.07 * 10^{-4}$

Table 6. Inferential accuracy results for the Fracture.

	sum (%)	class 0 (%)	class 1 (%)	class 2 (%)	class 3 (%)	class 4 (%)
CGP	239.55 (263.40)	67.85 (73.33)	24.58 (32.69)	38.97 (42.12)	72.80 (74.88)	35.35 (40.38)
CRGP	220.43 (255.65)	63.08 (72.52)	27.29 (32.82)	36.79 (41.56)	60.98 (71.09)	32.30 (37.66)
MGP	77.08 (191.50)	19.71 (72.15)	5.42 (25.31)	5.64 (24.07)	27.94 (43.98)	18.37 (25.99)
GP	55.99 (187.03)	10.83 (71.84)	4.94 (22.95)	8.06 (18.92)	19.73 (45.97)	12.43 (27.36)

Table 7. Comparison results between the CGP and the others for the Fracture.

	sum (%)	class 0 (%)	class 1 (%)	class 2 (%)	class 3 (%)	class 4 (%)
CGP and CRGP	8.67 (3.03)	7.56 (1.11)	-9.91 (-0.37)	5.92 (1.34)	19.38 (5.32)	9.45 (7.25)
CGP and MGP	210.78 (37.54)	244.26 (1.63)	353.95 (29.17)	591.09 (75.00)	160.55 (70.24)	92.37 (55.37)
CGP and GP	327.82 (40.83)	526.76 (2.07)	397.24 (42.43)	383.60 (122.62)	268.90 (62.89)	184.33 (47.62)

Table 8. T-test comparison results between the CGP and the others for the Fracture.

	class 0	class 1	class 2	class 3	class 4
CGP and CRGP	$1.71 * 10^{-4}$	$1.73 * 10^{-5}$	$6.00 * 10^{-8}$	$1.26 * 10^{-7}$	$1.68 * 10^{-4}$
CGP and MGP	$6.61 * 10^{-5}$	$5.94 * 10^{-6}$	$1.59 * 10^{-8}$	$2.93 * 10^{-7}$	$2.31 * 10^{-5}$
CGP and GP	$1.70 * 10^{-4}$	$1.73 * 10^{-5}$	$6.17 * 10^{-8}$	$2.67 * 10^{-7}$	$6.05 * 10^{-5}$

data sets. The experimental results show the CGP outperformed than the CRGP, the MGP and the GP. With the flexibility of GP, the CGP can learn complex relationships that cannot be handled by BN.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of otems in large databases. In *Proceedings of the 1993 International Conference on Management of Data*, pages 207–216, 1993.

[2] Ester Bernad-Mansilla and Josep M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation, MIT Press*, 11:209 – 238, 1998.

[3] Alex A. Freitas, editor. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer, 2002.

[4] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1998.

[5] D. Heckerman and D. Geiger. Learning bayesian networks: a unification for discrete and gaussian domains. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence*, pages 274–284, 1995.

[6] S. Hettich, C.L. Blake, and C.J. Merz. Uci repository of machine learning databases, 1998.

[7] JH Holland. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In *Machine Learning*, pages 593–623, 1986.

[8] Verlyn M. Johnson and John V. Carlis. Sharing and reusing rules: A feature comparison of five expert system shells. In *IEEE Expert: Intelligent Systems and Their Applications*, pages 3–17, 1994.

[9] J.K. Kishore, L.M. Patnaik, V. Mani, and V.K. Agrawal. Application of genetic programming for multicategory pattern classification. In *IEEE Transactions on Evolutionary Computation*, pages 242–258, 2000.

[10] L. So K.F. Yam K.S. Leung, Y. Leung. Rule learning in expert systems using genetic algorithm: 1, concepts. In *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks (Iizuka, Japan)*, pages 201–204, 1992.

[11] Z. Michalewic, editor. *Genetic Algorithms+DataStructures=Evolution Programs*. New York: Springer-Verlag, 1994.

[12] R.S. Michalski. A theory and methodology of inductive learning. In J.G. Carbonell R.S. Michalski and T.M. Mitchell, editors, *Machine Learning - An Artificial Intelligence Approach*, chapter 4. Los Altos, Calif., 1983.

[13] S. Monti and G. F. Cooper. Learning bayesian belief networks with neural network estimators. In *Advances in Neural Information Processing Systems*, pages 579–584, 1997.

[14] P. Myllymaki, T. Silander, H. Tirri, and P. Uronen. B-course: a web service for bayesian data analysis. In *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, pages 247–256, 2001.

[15] Ngan P. S., Wong M. L., Lam W., Leung K. S., , and J. C. Y. Cheng. Medical data mining using evolutionary computation. In *Artificial Intelligent in Medicine, Special Issue On Data Mining Techniques and Applications in Medicine*, pages 73–96, 1999.

[16] Wing-Ho Shum, Kwong-Sak Leung, and Man-Leung Wong. Co-evolutionary rule-chaining genetic programming. In *Intelligent Data Engineering and Automated Learning - IDEAL 2005: 6th International Conference. Lecture Notes in Computer Science.*, pages 546–554, 2005.

[17] A. Teredesai, V. Govindaraju, E. Ratzlaff, and J. Subrahmonia. Recurrent genetic programming. In *IEEE International Conference on Systems, Man and Cybernetics*, page 5, 2002.

[18] S.W. Wilson. Generalization in the xcs classifier system. In *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674, 1998.

[19] S.W. Wilson. Mining oblique data with xcs. In *International Workshop on Learning Classifier Systems*, page Extended Abstract, 2000.