Classification Learning System and Rule Extraction using Extended Genetic Programming for Data Mining

Ayahiko Niimi

A Dissertation

Submitted to Toin University of Yokohama
in partial fulfillment of the requirement
for the degree of Doctor of Engineering

January 2002

Acknowledgements

The author wishes to express his sincere gratitude to his supervisor Professor Eiichiro Tazaki whose guidance and support has been greatly helpful in completing this dissertation.

To Professor Takeshi Agui, Professor Chieko Furuichi, Professor Naimu Kuramochi, and Professor Masaaki Takeuchi, he is indebted for their valuable comments and suggestions during the preparation of this dissertation.

He further wishes to thank Dr. Kenneth James Mackin and Mr. Yasser Hassan, fellow doctorate student, for them valuable comments and suggestions during the course of the research.

To the colleagues in Tazaki Intelligent Informatics Laboratory, the author expresses his thanks for their collaboration and assistance.

The author would like to express his thanks to Dr. Katusmi Yoshida at the Department of Preventive Medicine, St. Marianna University School of Medicine for the medical dataset for the occurrence of hypertension, and helpful discussions. The author would like to express his thanks to Dr. Shusaku Tsumoto at the Department of Medical Informatics, Shimane Medical University, School of Medicine for the meningoencephalitis medical dataset and helpful discussions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, a large amount of data is accumulated, and so more effective methods to extract significant knowledge from these data are needed. Knowledge Discovery in Databases (KDD), often called data mining, aim at discovery of useful knowledge from large collection of data. There are various data as a large amount of data, such as text data by web pages and/or mailing list on internet, the convenience store's POS(point-of-sale) data, the utilization history of credit cards, genome data, medical data, graphics about star, the weather data, the earthquake data, and the space observation data, etc. Especially, the maintenance of the database is advanced in the company, and a large amount of data is held in the company. On the other hand, some databases related with physics, chemistry, biology, economics, etc. have been constructed by scientists of each domain. Furthermore, researchers have been researched using large quantity of numerical value data and graphics data.

It was thought to use positively these databases which were with great pains. Because these databases contain too a large amount of for operating by manually, it is very difficult to find significant knowledge by manual operation. It was thought whether data mining by the computer was able to apply to databases, or not. Many of these databases, data was collected without purpose, or was collected more than enough amount of original purpose.

There were techniques by which knowledge was extracted from database since before. In these techniques, the knowledge which wanted to be extracted was decided at the stage before data was collected, and there were a lot of techniques which to confirm whether the knowledge which wanted to be extracted by statistical operation was correct, or not. However, in the hopeful technique of knowledge extraction, whether the knowledge which is wanted to be extracted, beforehand is not decided, or is not clearly decided. In the data mining technique, we can gradually decide the knowledge what we want, while operating of data mining. Data mining by a statistical operation is still used. Data mining of a statistical technique is applied to the field where data contains a large amount of continuous value attribute.

On the other hand, the data mining technique by an artificial intelligence approach is being requested by data mining which narrows the wanted knowledge. The approach tries to operate which is able to be narrowed knowledge by an artificial intelligence learning technique. There are some techniques used as an artificial intelligence technique, such as neural network, genetic algorithm, and ILP (Inductive Logical Programming), etc. Recently, it is thought to do data mining from the database distributed on the network by using the network technology. However, in data mining by an artificial intelligence

technique, the interpretation of the extracted knowledge is difficult, and it is not easy to extract as knowledge.

In the data mining procedure, it is necessary to choose a concrete data mining technique according to the knowledge which wants to be extracted and the used database. The feature of the extracted knowledge can be predicted by experience of chosen technique. For instance, by using association rule extraction method, a significant association is lost in other many meaningless rules, or only the well known knowledge is extracted. By using the decision tree construction method, tree becomes too large to understood the whole of tree, or small tree contains only evident knowledge.

In the field of data mining, it is requested, that data mining tool can be corresponded to a lot of knowledge representations and it is easy-to-use. Being ultimately requested data mining tool is, not to depend on the database attributes, not to depend on the result's knowledge representation, its background knowledge is minimum, it is easy-to-use and high-speed.

In this dissertation, we propose the technique by which genetic program are combined with an existing data mining technique by such a background.

Using genetic programming for the data mining technique has two advantages.

The first advantage by using genetic programming for the data mining technique is, that the knowledge representation can become more rich. In genetic programming, the structure can be dynamically learned. Moreover, at the genetic programming system design, the system designer can decide the learned structure. In a word, it is constructed the system which can use for the decision tree and the rule, by the one learning framework.

The second advantage by using genetic programming for the data mining technique is, that data mining process can be contain a probabilistic operation. The unexpected knowledge at design process, cam be extracted by a probabilistic operation. The unexpected knowledge includes the knowledge which cannot be interpreted. However, even if system designer cannot interpret its knowledge, the interpretation might be possible because of other background knowledge. Moreover, a new interpretation might become possible by a new experiment plan by the not interpreted knowledge. Because the database contains huge data, only the common sense can be interpreted, new common sense might be able to be constructed by the unexpected knowledge.

Not only using genetic programming for data mining technique has two advantages. The first advantage is to be able to cover the slowness of learning speed by genetic programming. Other data mining techniques are treated as the preprocessing process of genetic programming. Because genetic programming can be learned by a variety of knowledge structure, the result knowledge of the preprocessing can be easily taken into genetic programming. According to situation, it is possible to make preprocessing results to the design indicators of the initial population design and the node design of genetic

programming. Therefore, designing the system even if the system designer does not know genetic programming so much becomes possible to use.

The second advantage is that treating the result of other data mining techniques as prior background knowledge, it is possible to construct the system even if the background knowledge at data mining is a little become possible to use. Therefore, designing the system even if system designer's data knowledge is a little becomes possible to use.

A lot of researches on data mining are proposed. [Arikawa 1999, Arikawa 2000, Liu 1998a, Liu 1998b, Lu 1997, Adriaans 1996, Berry 1997] Especially, decision tree construction method (C4.5) [Quinlan 1995] and association rule algorithm (apriori algorithm) [Terabe 2000, Kitsuregawa 1997], which we use our proposed method both, have became a general technique. Therefore, these algorithms are done to an initial stage of data mining, searching for the direction of data, and approaching to the data based on the these results as other data mining policy. However, in these approaches, a decision tree construction method and an apriori algorithm are used as preliminary experimental data mining. [Lu 1997]

Though it is not an approach by the combination of plural techniques, it is proposed that, the technique divides some data mining techniques into element methods and rearranges the elements automatically to do data mining. [Suyama 1999]

Various data mining techniques which use common databases are evaluated among researchers. [Tsumoto 1998, Tsumoto 1999a, Tsumoto 2000, Tsumoto 1999b, Washio 2001] In these evaluations, data mining is done from the same database by using the technique proposed each researcher. As a result of data mining, they got the comment of other researchers with the comment by the experts. There is [Blake 1998] as a comparison of the data mining techniques which use common databases. The differences between [Blake 1998] and [Tsumoto 1998, Tsumoto 1999a, Tsumoto 2000, Tsumoto 1999b, Washio 2001] are whether one researcher compares each techniques or different researchers do, and whether it can get the expert comments and feedback or not. In [Tsumoto 1998, Tsumoto 1999a, Tsumoto 2000, Tsumoto 1999b, Washio 2001], there is the datasets which is not done data cleaning. So this point is another different policy of [Blake 1998].

The proposal method can be discussed as examples of applied genetic programming. A lot of applied examples of genetic programming are proposed. [Koza 1992b, Koza 1994c, Koza 1994d, Koza 1994e, Langdon 1998, Benyahia 1998] However, it is not usual application using a lot of of nodes with genetic programming like this dissertation. Some approaches are proposed as genetic programming application for data mining. [Nakayama 2000].

The proposal method can be discussed as an improvement of genetic programming. The improvements of genetic programming as followings, the modifications of crossover and mutation, the modifications of gene design, the parallel genetic programming, etc.

[Koza 1992a, Koza 1994b, Koza 1999, Iba 1996, Iba 1994, Iba 1993, Iba 1999, Iba 2000, Koza 1994a, Ito 1998] In this dissertation, the combination with reinforced learning and the design of crossover are such approaches. However, it is not usual that the genetic programming is improved by combination of other techniques.

In this dissertation, we propose the application of genetic programming to data mining. And, to use genetic programming for data mining, we propose the expansion of genetic programming. Therefore, this dissertation has the side as applied case with genetic programming and the side as the data mining tool. As a side of applied case with genetic programming, there is a novelty as following: using a lot of nodes with genetic programming, using learning with the continuous vale attributes directly, to improve the learning speed. As a side as the data mining tool, there is a novelty as following: doing data mining combined with probability operations, the possibility of learning structured knowledges, not necessary background knowledge for system construction.

The field of KDD which treats with this dissertation is limited to data mining. Especially, the extraction of the decision tree, the classification rule, and the correlation rule is discussed. The data of the table form is a targeted database. As for the size of database, the number of elements is the one of several thousand levels. However, the data size more than thousands can be treated by proposed techniques. The problems and the improvements when data mining is done by genetic programming are discussed. The improvements not specialized to data mining are proposed as an improvement of genetic programming in some part.

Data mining method using genetic programming proposed with this chapter are mainly three methods. The first method is the method which combined genetic programming and decision tree construction method. This is effective to the data mining method by which the decision tree is extracted. The second method is the method which combined genetic programming and association rule extraction method. This is effective to data mining to a large-scale database. Moreover, it is a method to be able to treat the association rule and the classification rule at the same time by using genetic programming. The third method is the method which combined genetic programming and logical functions. It is the data mining method by logical functions by defining a logical function in the function node set of genetic programming. On the others, we propose as improvements of genetic programming not specialized to data mining, the method of combining with reinforced learning and the method of modification of crossover.

This dissertation will be organized at the following:

In chapter 2, we explain an overview of genetic programming, which is main of the proposed method. we discuss about general problems of genetic programming and solution methods. The problems and the design policy when genetic programming is used for data mining are described.

In chapter 3, we explain the method which combined genetic programming and decision tree construction method. The decision tree mining method is chiefly proposed.

In chapter 4, we explain the method which combined genetic programming and association rule extraction method. The focus is chiefly applied and described to data mining to a large-scale database. Moreover, because the association rule and the classification rule are treated at the same time by using genetic programming, we discuss this.

In chapter 5, we explain the method which combined genetic programming and logical functions. We propose the data mining method by logical functions by defining a logical function in the function node set of genetic programming.

In chapter 6, we explain the improvement methods of genetic programming. Because of some problems, genetic programming is difficult to use for data mining. We propose three methods which improve these problems. The first methods improves the problem that genetic programming is difficult to construct its system. The second methods improves the problem that genetic programming is difficult to treat with continuous value attributes. The third method improves the problem that genetic programming is difficult to use with a lot of nodes.

In chapter 7, we conclude with a discussion of the results and possible improvements.

Chapter 2 Summary:

The genetic programming paradigm continues the trend of dealing with the problem of representation in genetic algorithms by increasing the complexity of the structures undergoing adaptation. In particular, the structures undergoing adaptation in genetic programming are general, hierarchical computer programs of of dynamically varying size and shape.

Chapter 3 Summary:

There are many learning methods for classification systems. Genetic programming can change trees dynamically, but its learning speed is slow. Decision tree methods using C4.5 construct trees quickly, but the network may not classify correctly when the training data contains noise. For such problems, we proposed a learning method that combines decision tree making method (C4.5) and genetic programming. To verify the validity of the proposed method, we develop two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for the meningoencephalitis. We compared the results of propose method with prior ones.

Chapter 4 Summary:

Genetic programming (GP) usually has a wide search space and a high flexibility. So, GP may search for global optimum solution. But, in general, GP's learning speed is not so fast. Apriori Algorithm is one of association rule algorithms. It can be applied to large database. But, it is difficult to define its parameters without experience. We propose a rule generation technique from a database using GP combined with association rule

algorithm. It takes rules generated by the association rule algorithm as initial individual of GP. The learning speed of GP is improved by the combined algorithm. To verify the effectiveness of the proposed method, we apply it to the decision tree construction problem from UCI Machine Learning Repository, and rule discovery problem from the occurrence of hypertension database. We compare the result of proposed method with prior ones.

Chapter 5 Summary:

It is easy for an unexpected decision tree to be generated in the decision tree construction with genetic programming because the probability operation is contained. In the description of the decision tree by normal genetic programming, the division conditions by the attribute are connected with AND operator, and the tree evaluates effectiveness as a rule. However, if the description of the function node is modified, a more flexible rule is sure to be able to be generated. In this chapter, we show that the description of a more flexible decision tree is possible by the addition of the OR function and NOT function to the function node group.

Chapter 6 Summary:

In chapter 6, we explain the three improvement methods of genetic programming. Each improvement has each section.

In the first section, we discuss about object oriented combined approach.

There are many learning methods for classification systems. Genetic programming (one of the methods) can change trees dynamically, but its learning speed is slow. Decision tree methods using C4.5 construct trees quickly, but the network may not classify correctly when the training data contains noise. For such problems, we proposed an object oriented approach, and a learning method that combines decision tree making method (C4.5) and genetic programming. To verify the validity of the proposed method, we developed two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for the meningoencephalitis. We compared the results of proposed method with prior ones.

In the second section, we discuss about new genetic operators.

Genetic programming(GP) usually has a wide search space and a high flexibility, so GP may search for a global optimum solution. But GP has two problems. One is slow learning speed and huge number of generations spending. The other is difficulty to operate continuous numbers. GP searches many tree patterns including useless node trees and meaningless expression trees.

In general, GP has three genetic operators (mutation, crossover and reproduction). We propose an extended GP learning method including two new genetic operators, pruning (pruning redundant patterns) and fitting (fitting random continuous nodes). These operators have a reinforcement learning effect, and improve the efficiency of GP's search.

To verify the validity of the proposed method, we developed a medical diagnostic system for the occurrence of hypertension. We compared the results of proposed method with prior ones.

In the third section, we discuss about modified crossover.

Many GP learning methods have been proposed to decrease node combinations in order to keep the node combinations from explosively increasing.

We propose a technique using an opposite approach which tests a greater number of combinations in order to decrease the chances of the search being 'trapped' in a local optimum. In the proposed technique, how 'different' the individual structure is is used as an index in selecting individuals for genetic operations. Therefore, variety in the GP group is strongly maintained, and it is expected that GP learning is always done to a new combination.

In this session, we modify the normal crossover operation by using this proposed technique, and expect that GP search becomes more effective.

To verify the validity of the proposed method, we developed a medical diagnostic rule generation system for the occurrence of hypertension. We compared the results of the proposed method with prior ones.

Chapter 7 Summary:

We conclude the dissertation with an cverall discussion of the implications of this research, and possible enhancements to the proposed systems. We also include discussions on the possibility of combination learning in genetic programming.

Chapter 2

Overview of Genetic Programming

The genetic programming paradigm continues the trend of dealing with the problem of representation in genetic algorithms by increasing the complexity of the structures undergoing adaptation. In particular, the structures undergoing adaptation in genetic programming are general, hierarchical computer programs of of dynamically varying size and shape.

2.1 Definitions of Genetic Programming

Evolutionary Algorithm (EA) is a calculation method that models on gene of natural species and its evolution, it searches for the solution using the individual group as population. It seems a optimization which can be applied to a wide problem with high robustness. EA is a generic name of some similar techniques which have been independently progressed, and the following techniques are included.

ES:Evolutionary Strategy: Generations changes of selecting the child of m individual from the parent of n individual by chiefly using the mutation as an operator. A quantitative research is not difficult, the effect of the mutation are analyzed mathematically.

EP:Evolutionary Programming: mainly for automaton adaptive learning.

GA:Genetic Algorithm: The chromosome of the character string is operated by crossover, mutation etc. The research field of classifier system which applies GA is approved.

GP:Genetic Programming: The one that chromosome expression of GA was expanded to treat graph structure and tree structure.

Especially, in evolutionary approach for information, an adaptive learning which modified internal information by the interaction with the environment and the process of problem solving, is able to apply to solving problem and learning which looks like natural spices, without asking "What is intelligence?". It is different approach from historically approach. (See figure 2.1)

The idea of natural selection (selection) and the genetic mutation (mutation) is especially paid to attention about the natural theory of evolution. The gene of the individual with an excellent ability is left for the following generation by natural selection, GP

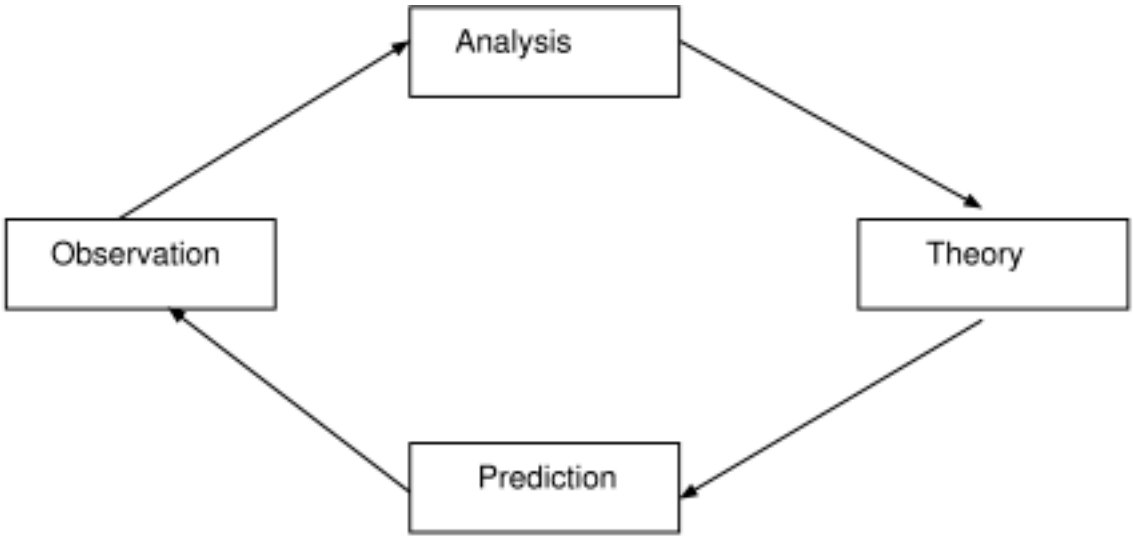


Figure 2.1: The Research Ring of Experienced Principle

searches for a new possibility by adding the mutation to the gene by an genetic operation. Crossover and the mutation, etc. are used for the genetic operation.

Expression of chromosome: The difference between GP and GA is that GP has extended its chromosome to allow structural expression using function nodes and terminal nodes [Koza 1992a, Koza 1994b]. GA uses character string (generally, one dimensional character string, such as 0, 1). On the other hand, GP uses structural expression like S type expression of LISP. Using function nodes and terminal nodes, GP can search hierarchical computer programs by undergoing adaptation. (See figure 2.2)

In applying GP to a problem, GP has five major preparatory steps. The five steps are followings.

1. The set of terminals nodes
2. The set of primitive functions
3. The fitness function
4. The parameters for controlling the run
5. The method for designating a result and criterion for terminating a run

The decision tree construction with GP is given by the following procedures. (refer to figure 2.3.)

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.
2. The fitness value, which relates to the problem solving ability, for each individual of the GP population is calculated.
3. The next generation is generated by genetic operations.
   (a) The individual is copied according to fitness value (reproduction).
   (b) A new individual is generated according to intersection (crossover).
   (c) A new individual is generated by random change (mutation).
4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

Application to various problems becomes possible designs the following five basic elements in GP. Depending on design of the following five basic elements in GP, application to various problems can be possible.

- non-terminal sign: Sign used by non-terminal node. Function in S type of LISP.
- terminal sign: Sign used by terminal node (leaf). Atom in S type of LISP.
- fitness value
- parameter: Probability to which crossover and mutation happen, the size of population, etc.

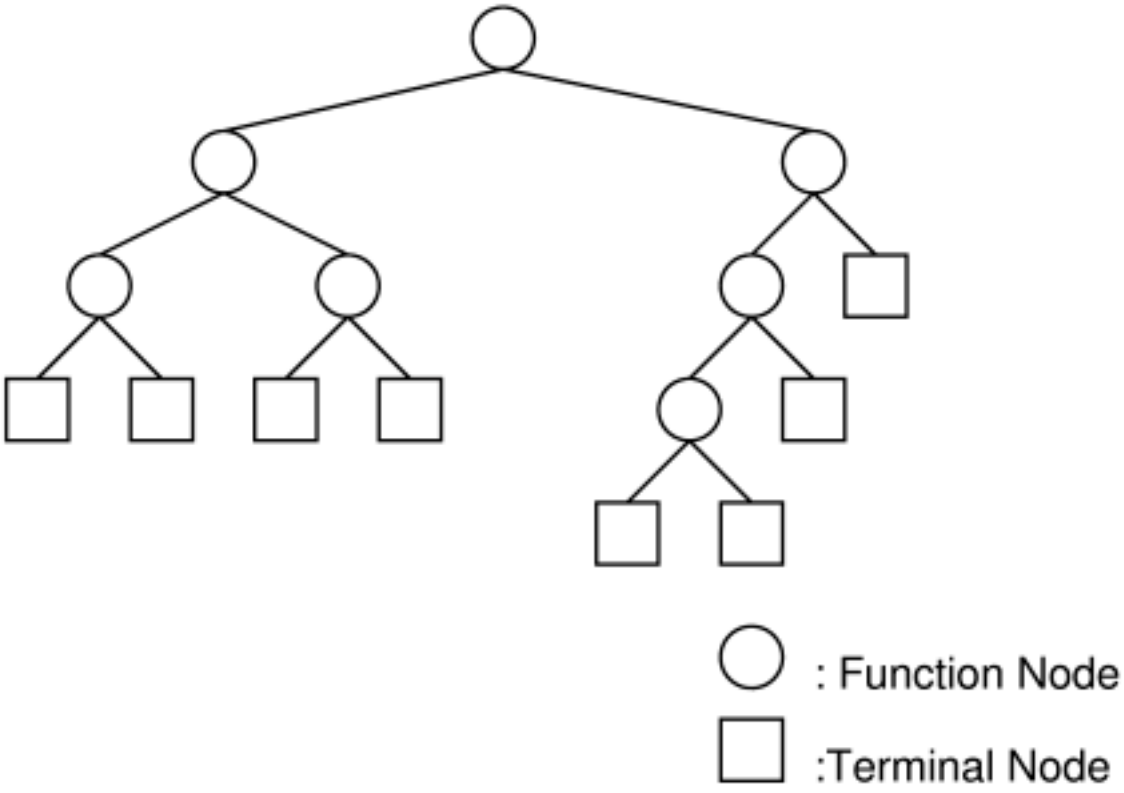


Figure 2.2: Examples of Genetic Programming Tree Structure

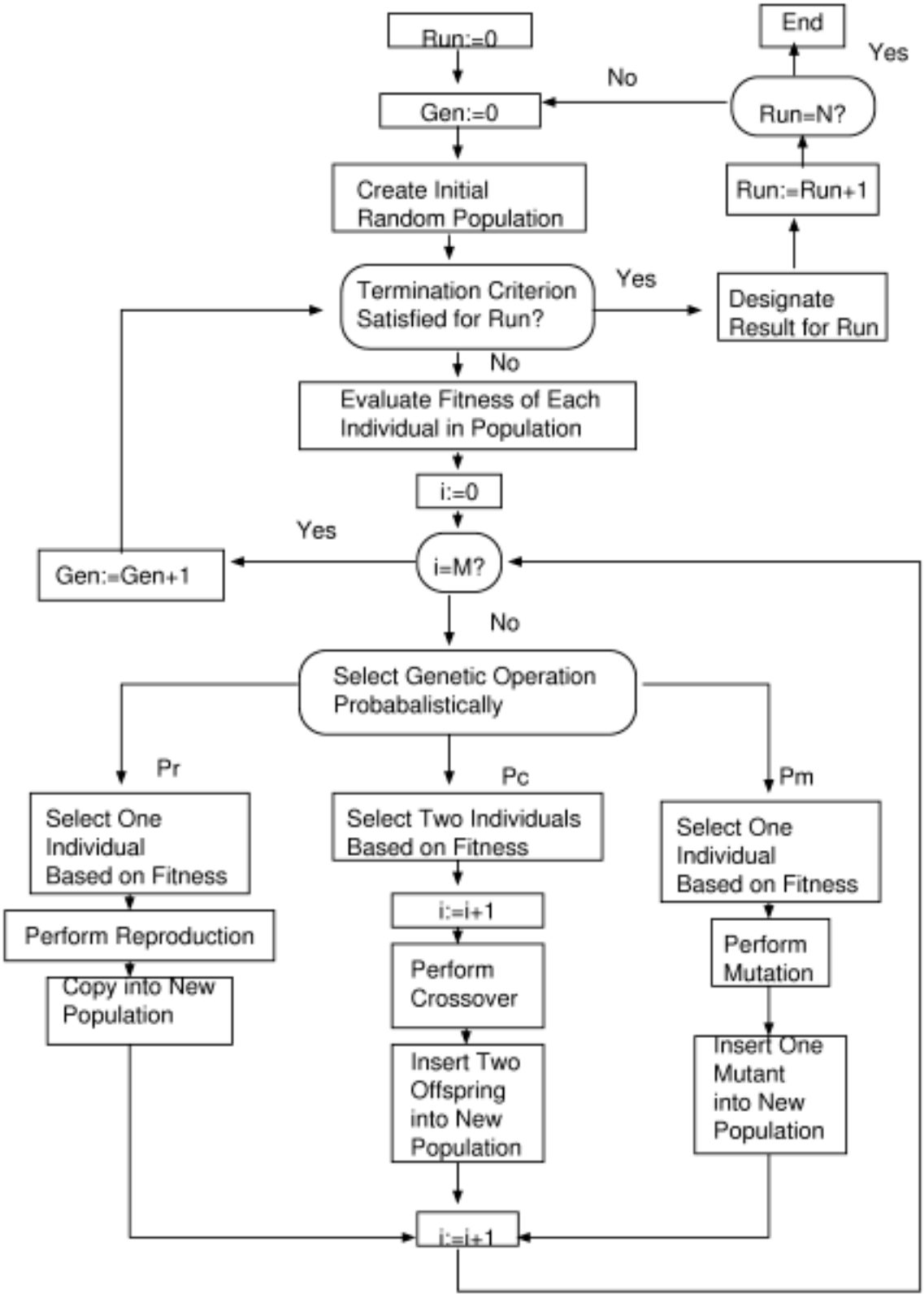


Figure 2.3: Flowchart of GP

- end condition

Morecver, the genetic operation is occasionally changed depending on the problem.

How the individual which was each generation adjusted is evaluated according to the fitness function. Generally, these fitness values are deeply related to the end condition of program, the probability of the selection and genetic operations.

There are the following four in the fitness values. [Zongker 1996]

Raw Fitness Calculated from each individuals by fitness function. (f_r)

Standard Fitness Change the raw fitness values as normalization that raw fitness 0 become best. (f_s)

Adjusted Fitness Calculated from standard fitness values that it takes values between 0 to 1, and 1 is best. ($f_a = 1/(1 + f_s)$)

Normalized Fitness Calculated from adjusted fitness values. It can becomes an individual's index of the contribution degree in a certain generation.
($f_n(i) = f_a(i)/\sum_j f_a(j)$)

The selection is a mechanism that the gene of individual which has more excel character than the other individual is existed high possibility among the group. The ratio of genetic operations is different depending on the fitness value obtained by the evaluation by the fitness function.

A typical selection method is introduced as follows.

Roulette Selection Method Roulette which has the area proportional to the fitness value is made, the roulette is turned, and the individual of the hit place is selected. Method that number of individuals chooses and turns roulette repeatedly.

Tournament Selection Method The number of individuals (tournament size) which is is chosen at random from among the group, and the best in that one is selected. Method to repeat this process until number of groups is obtained.

Rank Selection Method Method to display each individual from the one with large fitness value sequentially, and to decide number of children according to function corresponding to this order.

Elite Strategy Method which always copies some of parent with good result, and leaves them for the next generation.

Genetic operations are used to search for the search space in GP. The chromosome with a new gene composition is made from modified the gene of each individual, and the search for a new area by the search space is attempted. This modified operation is called a genetic operation, and the operation based on the change which takes place to the gene of the natural species. The operations generally used with GP is shown below.

- Individual copy (reproduction)
- Genetic intersection (crossover)
- Replace each gene in genetic coding (inversion)

- Random gene change (mutation)

Reproduction is an operation by which the individual in the population which is is copied to the next generation's population as it is. A reproduced individual is decided by the selection is decided, and the individual with a high fitness value will be succeeded by the next generation.

Crossover is an operation by which the chromosome of the individual is recomposed from two parents'. By crossover, the child can succeeds parent's character, morecver the child become having new character which is not same as parents' one. (See figure 2.4)

Because GA ties the character string with the crossover point, the follcwing kinds of crosscver are proposed by how to decide the crossover point.

- one-point crossover
- n-point crossover
- uniform crossover

On the other hand, two individuals of the child are newly generated with the exchange of each parent's partial structure in GP. As for the chromosome of GP, the generated chromosome is effective even if the partial tree is arbitrarily replaced, and the replacement of a partial tree between two individuals is also possible.

example.1

```
(* (- X Z) (+ X Y)) (- (* X 1.2) (+ Y (* Z X)))
          ↓
(* (+ Y (* Z X)) (+ X Y)) (- (* Z 1.2) (-X Z))
```

example.2

```
(- (* X 1.2) (+ Y (* Z X))) (- (* X 1.2) (+ Y (* Z X)))
          ↓
(- (+ Y (* Z X)) (+ Y (* Z X))) (- (* X 1.2) (* X 1.2))
```

example.3

```
(- (* X 1.2) (+ Y (* Z X)))
          ↓
(- (* Z X) (+ Y (* X 1.2)))
```

The replacement of a partial tree in the parent of one individual is called inversion.

At an initial stage of evolution, it seems that the chromosome of two individuals chosen for crossover is greatly different. Therefore, the generated chromosome is expected to be able to add a big change by crosscver, and to have the gene composition which does not exist until then. At a stage advanced by evolution, the gene composition resembled spreads in the population, and the difference between chromosomes become small. As a result, it is difficult to be expected that an individual with a new gene composition is generated by crossover.

The mutation is an operation by which a certain node is replaced with other nodes. As follows, it is possible to classify mutations with the acting node. (See figure 2.5)

- Mutation from terminal node to non-terminal node : generation new subtree

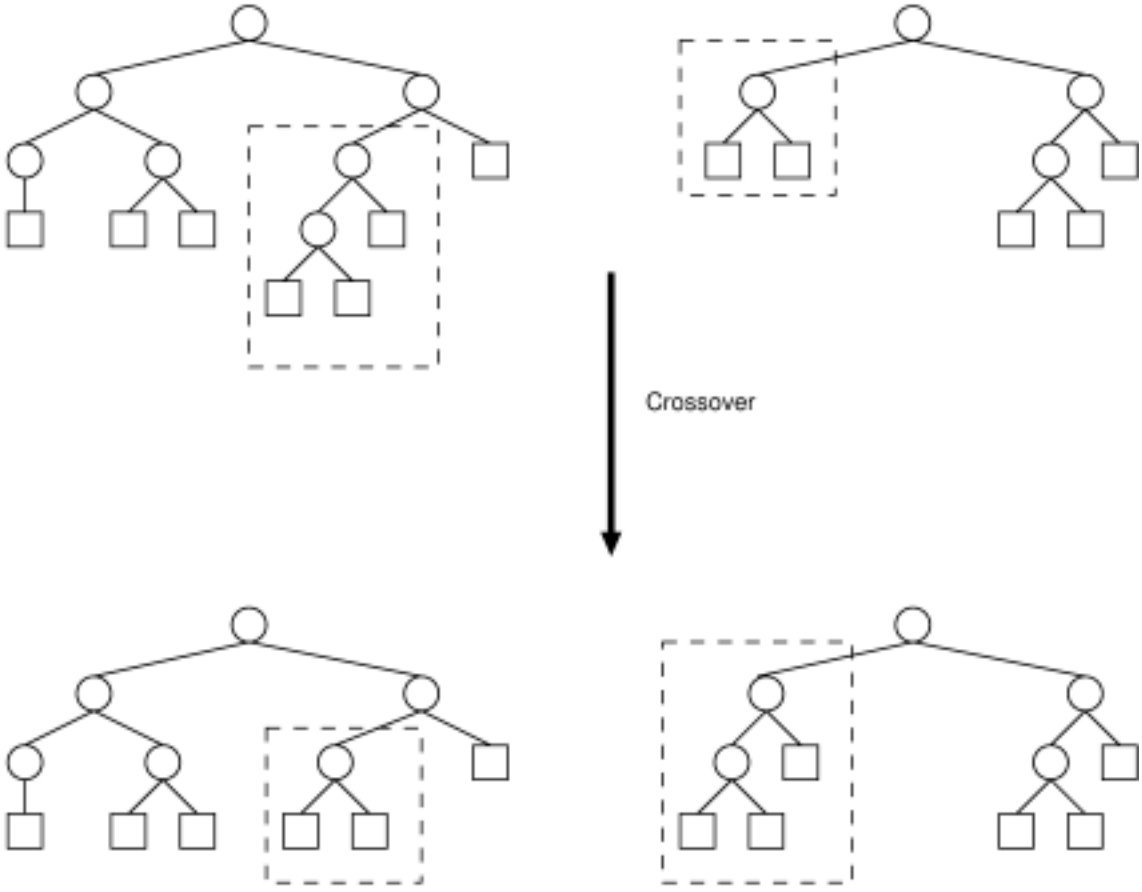


Figure 2.4: Examples of Crossover in GP

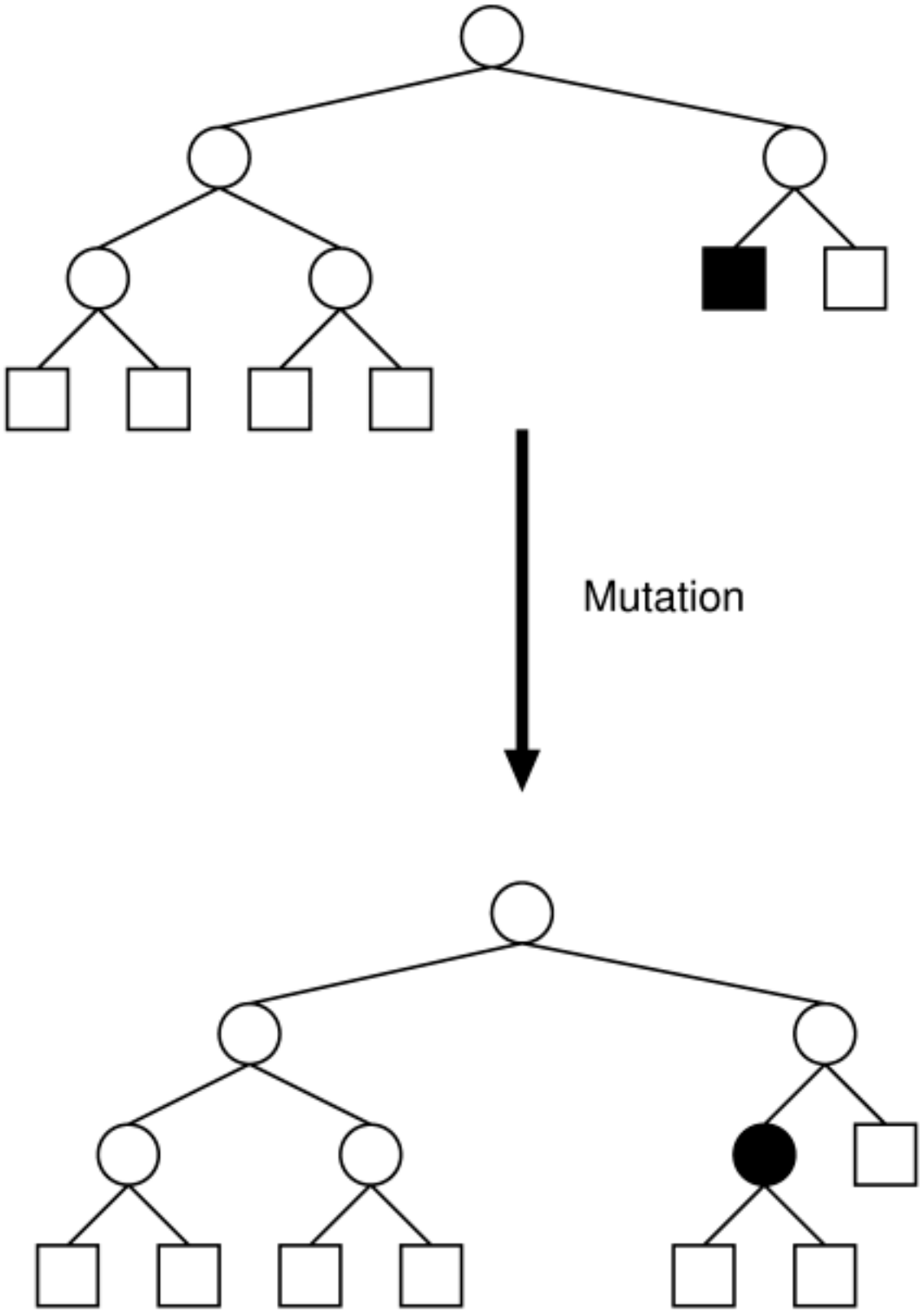


Figure 2.5: Examples of Mutation in GP

- Mutation from terminal node to terminal node : change the node label
- Mutation from non-terminal node to terminal node : delete subtree
- Mutation from non-terminal node to non-terminal node : if the number of new non-terminal nodes and the number of old terminal nodes are same, it becomes changing the node label. Otherwise, it becomes generating / deleting subtree.

example.

```
(* (- X Z) (+ X Y))
        ↓
(* (* X Z) (+ Z Y))
```

In the mutation, there is a role of various maintenance of the group and the limited part searches by the gene type. However, it is thought that the variety of the gene in the group is secured enough because an arbitrary partial tree is formed in an arbitrary place of the tree structure in GP. Moreover, the mutation between terminal nodes can be achieved by crossover. The mutation rate is not necessary so high in GP by abcve reasons.

2.2 General Problems of Genetic Programming and Solution Methods

We will introduce about application of GP, some problems of GP, expansion of GP, theoretical research of evolutionary algorithm, and research on dynamics of evolution. The problems and the design policy when genetic programming is used for data mining are described.

GP is applied to various fields. The applied fields of GP is various from the problem solving of AI to a practical problem such as the robot operation and molecular biology. The tree is transformed into the tree structure by using crossover and mutation in GP, and GP can search for LSIP (S type) program and concept tree, etc.

Some examples are shcwn as follcws.

- CLS:Concept Learning System
- learning artificial neural network
- programming for artificial life

example. XOR learning

- XOR by GP

```
(P (* 1.841 (P (* 1.66 D0)
               (* -1.387 D1) ) )
   (* 1.901 (P (* 1.191 D1)
               (* -0.989 D0) ) ) )
```

- XOR by artificial neural network

```
y = (P (* 0.5 (P (* 1 D1)
                 (* -2 D0) ) )
       (* 0.5 (P (* 1 D0)
                 (* -2 D1) ) ) )
```

p: calculate sum of inputs. If the calculated results is bigger than the threshold (i.e., 1.0), then return 1. Otherwise, return 0. *,+: Multiplication, calculate sum

Other GP applications are the programming which applied to the game programming, the key word extraction, and the number of application problem, etc.

GP has the following problems now.

- A mathematical background is not proven.
- Increase calculation cost by parallel calculation and increasing population
- Destruction of schema by crossover
- Expression problem when node is designed
- How to evaluate the tree structure

Because the problem of GP is solved, the following techniques are proposed.

- ADF:Automatic Defined Function
- other module structure extraction algorithms [Naemura 1999]
- fitness function based on MDL(Minimum Description Length)
- GP based on type theory (Strongly Typed GP)
- Parallel GP

It is difficult that a strict solution of the applied problem is not efficiently obtained in general, EA (Evolutionary Algorithm) is defined as an approximate calculation, and strong theory development is difficult. In crossover, the generated child depends on two or more parents, and the range of child's genes changing depends of parents' gene types. Because the operations are more complex than the mutations, the analysis is difficult. Only some analysis are proposed.

- Analysis of dynamics that individual which adjusts by environment leaves more descendants
- Analysis concerning shape of fitness value function searched by mutation and crossover

The following topics are about research on dynamics of evolution.

- Basic theorem and schema theory of group genetics
- Analysis by Markov chain model
- Genetic flowting: Phenomenon that individual group gradually loses variety even if there is no difference of fitness value between individuals when uncertainty by which limited of number of individuals and descendants are generated is considered

- Building block hypothesis

The following standards are proposed about encoding GP. [Yamamura 1994]In followings, because another can achieve some conditions in either of sacrifice, all need not be completely filled. The character preservingness definitely influences the settling speed in GP though the area dependency should greatly examine each problem area.

Completeness The solution candidate can express all as a chromosome.

Soundness All chromosomes correspond to the solution candidate.

Non redundancy The chromosome and the solution candidate become respondent a couple of one.

Character preservingness Parent's character is appropriately succeeded to to the child.

GP is one kind of the probabilistic search method based on a random specimen extraction. There is a feature in the point to manipulate the selection from the group and the solution candidate such as two or more intersection of the chromosome compared with other probabilistic searches. Here, the performance of GP is classified into three. [Yamamura 1994]

Random search level This is a performance level which can be achieved by a random search, quite a lot of trials are needed to raise accuracy, and the lowest performance level.

Local search level This is a performance level which can be achieved by repeating a local search (mutation).

Global search level This is a performance level not achieved if not depending on a global search (selection and crossover).

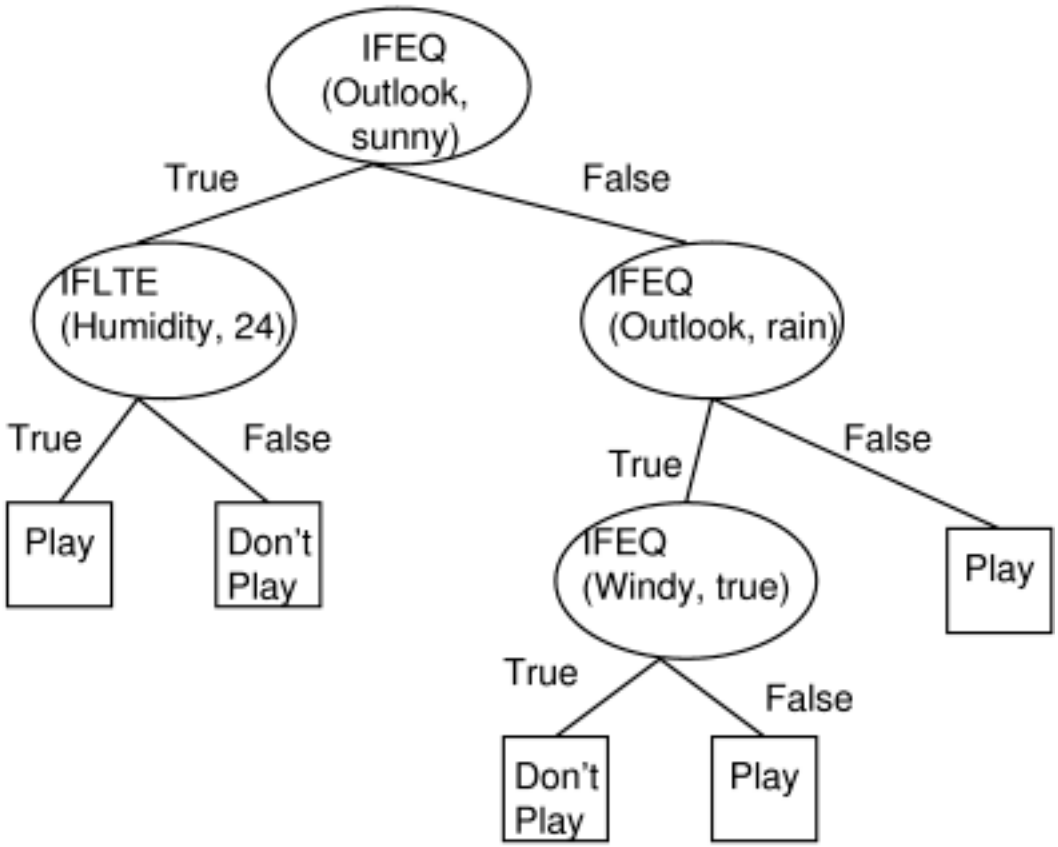
Using GP application, if the performance at the global search level is not demonstrated, it is not significant. The performance level drops if the character preservingness by which the global search level is achieved is not generated by encoding and the crossover design without supported by various maintenance.

In this dissertation, we will think about the individual expression as shown in Figure 2.6 generated from Table 2.1. In Figure 2.6, gp expresses like LISP-code as decision tree. "RPB" is defined as main GP tree. Both "ADF0" and "ADF1" are defined as each ADF tree. "IFLTE", "IFEQ" are function nodes. These functions requires four arguments, $arg1, arg2, arg3 and arg4$. The definitions of them are followings.

(IFLTE $arg1, arg2, arg3, arg4$**)** if $arg1$ is less than or equal to ($\leq$) $arg2$ then evaluate $arg3$, else then evaluate $arg4$

(IFEQ $arg1, arg2, arg3, arg4$**)** if $arg1$ is equal to(=) $arg2$ then evaluate $arg3$, else then evaluate $arg4$.

Decision Tree:



GP's Chromosome:

```
(IFEQ Outlook sunny
        (IFLTE  Humidity  24  Play  Don't Play)
        (IFEQ   Outlook rain
                (IFEQ  Windy true  Don't Play   Play)
                 Play))
```

Figure 2.6: One Example of Tree Structure, "Play or Don't Play"

Table 2.1: A Small Training Set of "Play or Don't Play"

Outlook	Temp(°C)	Humidity (%)	Windy?	Class
sunny	24	70	true	Play
sunny	27	90	true	Don't Play
sunny	30	85	false	Don't Play
sunny	22	95	false	Don't Play
sunny	21	70	false	Play
cvercast	22	90	true	Play
cvercast	28	78	false	Play
cvercast	18	65	true	Play
cvercast	27	75	false	Don't Play
rain	22	80	true	Don't Play
rain	18	70	true	Don't Play
rain	24	80	false	Play
rain	20	80	false	Play
rain	21	96	false	Play

In the figure, illustrate figure also expresses decision tree. "IFLTE" and "IFEQ" are expressed as circle. Other items are expressed as box. "IFLTE" and "IFEQ" figures contain $arg1, arg2$. $arg3, arg4$ are expressed as lines from circle.

In general, C4.5 is often used for decision tree learning. C4.5 is a high-speed algorithm, and it can be generate a high accuracy decision tree. On the other hand, the decision tree construction by GP is inferior to C4.5 by the calculation time. But, there is an advantage that various criterions can be used, and the decision tree by a higher-order knowledge representation can be constructed in learning by GP.

The nodes which are used compared node "IFLTE", such as "75" in the figure, is an ephemeral random constant terminal. These nodes are defined as "constant nodes with a continuous value attribute". Morecver, the node, such as "Humidity" and "Temp", is defined a terminal nodes which receive the input from a database.

One of the reasons why GP takes much time in calculation is that many types of nodes are defined in the decision tree construction. The discrete value attribute are treated as different nodes for each attribute. Morecver, in classification test of the continuous value attribute, the number of nodes is defined by the number of values which can be taken by random numbers, e.g. when the constant of the threshold of the divergence is given by random numbers. This causes an increase in the types of the node used.

When there are too many types defined, this may cause the GP learning speed to become very slow, or cause the GP to fail to reach the global optimum solution. This is caused by the explosive increase in combinations. However, useless and meaningless expressions might be included in the combination. For such problems, many techniques have been proposed. The ADF technique defines its cwn partial tree to make learning more efficient [Koza 1994a]. The MDL technique uses a fitness function defined by the size of the tree based on minimum description length (MDL) [Iba 1994].

Chapter 3

Combined Learning of C4.5 and Automatic Defined Function Genetic Programming for Construction of Decision Trees

There are many learning methods for classification systems. Genetic programming can change trees dynamically, but its learning speed is slow. Decision tree methods using C4.5 construct trees quickly, but the network may not classify correctly when the training data contains noise. For such problems, we proposed a learning method that combines decision tree making method (C4.5) and genetic programming. To verify the validity of the proposed method, we develop two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for the meningoencephalitis. We compared the results of propose method with prior ones.

3.1 Introduction

Now, we have a lot of huge databases about various fields. And we want to get new knowledge from them. Knowledge Discovery in Databases (KDD) is one of such topics.

The KDD process contains following steps [Liu 1998a, Liu 1998b, Berry 1997]. (See also figure 3.1.)

1. data warehousing,
2. target data selection,
3. cleaning,
4. projection and reduction,
5. model selection,
6. data mining,
7. evaluation and interpretation,
8. consolidation of the discovered information

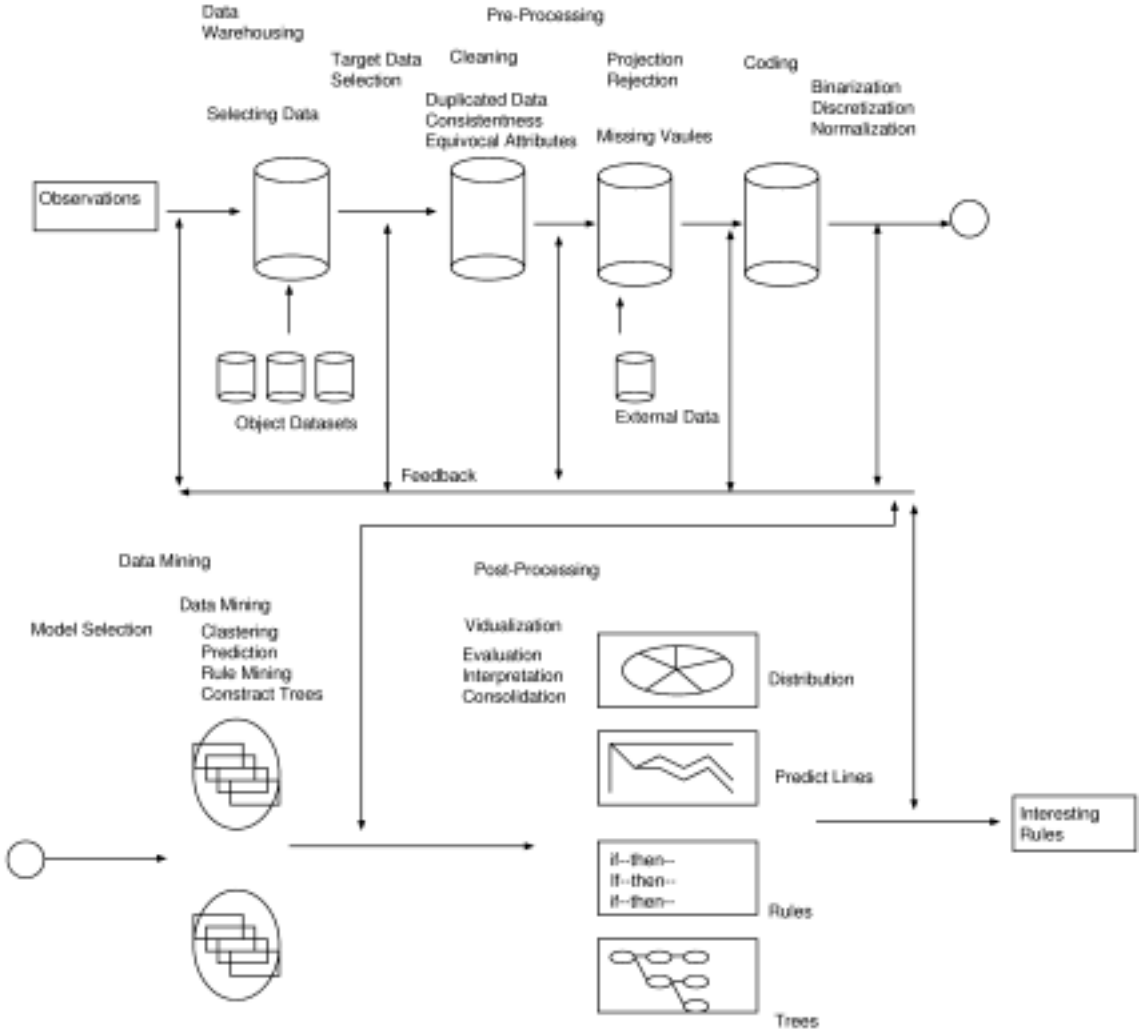


Figure 3.1: Flowchart of KDD Process

Data warehousing step means data collecting from a lot of different sources (they are a kind of database.) depend on applying applications. Target data step selection means select data to make a dataset. Cleaning step means adjusting dataset with deleting missing values and low association values. In model selection step, we need decide best modeling expression and modeling algorithms (classification or pattern matching algorithm) for application. In data mining step, we use some modeling algorithms (classification or pattern matching algorithm), to extract interesting and/or useful models. Evaluation and interpretation step means validating the results and consider them meanings. Final step, consolidation means arrangement the discovered information for more easy to be checked and reused.

In common, each step is called following,

- data warehousing ⇒ data warehousing,
- target data selection, cleaning, projection and reduction ⇒ pre-processing,
- model selection, data mining⇒ data mining
- evaluation and interpretation, consolidation⇒ post-processing

For data mining, various techniques have been proposed for the construction of the inference system using classification learning. In general, the learning speed of a system using a genetic programming is slow. Moreover, both problem background knowledge and design skill are demanded for the system designer. However, a learning system which can acquire higher-order knowledge by adjusting to the environment can be constructed, because the structure is treated at the same time.

On the other hand, a learning system which uses the decision tree can be trained shorter time than other compared techniques. An effective network structure constructed by the classification model is obtained by decision tree. But, there is a problem with deteriorated classification accuracy when the training data contains noise.

Each technique has advantages and disadvantages like this. In this chapter, we propose a combining method to construct a classification system trained by combining decision tree construction methods. It is expected that learning will occur while mutually making up for the advantage and the disadvantage of each technique.

To verify the validity of the effectiveness of the proposed learning method, we test the two learning methods: the decision tree construction method (C4.5), and genetic programming with automatic function definition (ADF). It is applied to two medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for meningoencephalitis. We compare the results with prior methods.

The remainder of this chapter is organized as follows: Section 3.2 and 3.3 briefly introduces the decision tree construction algorithm and genetic programming respectively and how to use in data mining. Section 3.4 presents our proposed method. Section 3.5 presents two experiments of verify the proposed method's effectiveness. Final Section 3.6 draws conclusions.

3.2 Decision Tree Construction Algorithm

There are some methods of inductive construction model by examining the recorded classification data and generalizing a specific example. The classification learning by

decision tree can achieve certain classification ability in a comparatively short time. C4.5 is one of decision tree construction methods, and it classifies data based on the gain criterion which selects a test to maximize expected information gain [Quinlan 1995]. As a result, important attributes can be collected at the root of the decision tree. Morecver, the algorithm contains branch pruning by estimating error rates to prevent the construction of excessively classified decision trees.

In the framework of decision tree construction algorithm, training data is divided to some subset by decision tree. Each subset has its class and rules. The decision tree can express easy understanding rules and hierarchical structure of database's attributes. The decision tree can use both categorical attributes and numerical attributes for its rules.

A decision tree has two items.

- leaf: class.
- branch: decision attribute node. Its expresses a checking one attribute value. One subtree expresses one attribute value checking.

Repeating branch checking from root to leaf, a decision tree classify database. The branch is checking its attribute and data's attribute, the root express that the data belongs to its leaf's class. The branch of decision tree express test for dataset. A dataset is divided by branch's attribute. When we use decision tree, we check branches attribute from its root to leaf. Some decision construction algorithm can use only binary decision tree. But C4.5 can construct decision tree in which each branch can have different number of routes.

The decision tree construction with C4.5 follcws the follcwing procedures proposed by Quinlan [Quinlan 1995].

1. Construction of initial decision tree.
2. Branch pruning of constructed decision tree.

In the construction step, to decide attribute which is better for branch, information gain is used. In the branch pruning step, the subtree of decision tree is pruned by reduced error rate.

C4.5 uses information gain to construct decision tree, and its information gain is similar to ID3's one. We show the definition of ID3's information gain, and what is extended definition from ID3 to C4.5.

For example, we construct a decision tree from a set T of training cases. It has the classes be denoted $\{C_1, C_2, \ldots, C_k\}$. T is expressed by some possible n test cases $(T_1, T_2, \ldots, T_n)$.

The original ID3 used a criterion called gain, defined belcw [Quinlan 1995].

Imagine selecting one case at random from a set S of cases and announcing that it belongs to some class C_j. This message has probability

$$\frac{freq(C_j, S)}{|S|}.$$

We define S is any set of cases, $freq(C_i, S)$ denotes for the number of cases in S that belong to class C_i, and $|S|$ denotes the number of cases in set S. And so the information it conveys is

$$-\log_2(\frac{freq(C_j, S)}{|S|})\text{bits.}$$

To get $info(T)$ which measures the average amount of information needed to identify the class of a case in T. (This quantity is also known as the entropy of the set S.), calculating

$$info(S) = -\sum_{j=1}^{k} \frac{freq(C_j, S)}{|S|} \times \log_2(\frac{freq(C_j, S)}{|S|})\text{bits.}$$

We want to get the measurement after T has been partitioned in accordance with the n outcomes of a test X.

$$info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T_i).$$

The quantity

$$gain(X) = info(T) - info_X(T)$$

measures the information that is gained by partitioning T in accordance with the test X. The gain criterion means selecting a test to maximize this information gain.

For some cases, its gain criterion is good criterion for selecting of a test attribute. But if an attribute has a lot of cases, its criterion is bigger than other test's criterion. Because, if we use such attribute for test, most of data is divided by its only one tests for many small sets.

C4.5 is extended to express of the gain criterion which can have a kind of normalization bias. So, the definition of $info(S)$ is modified as follows.

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2(\frac{|T_i|}{|T|})$$

And, the definition of $gain\ ratio(X)$ is modified as follows.

$$gain\ ratio(X) = gain(X)/split\ info(X)$$

Its $gain\ ratio(X)$ is considered the ratio of useful part of information for calcification. The implementation of C4.5 contains three types of tests by gain ratio.

- The "standard" test on a discrete attribute, with one outcome and branch for each possible value of that attribute.
- A more complex test, based on a discrete attribute, in which the possible values are allocated to a variable number of groups with one outcome for each group rather than each value.

- If attribute A has continuous numeric values, a binary test with outcomes $A \leq Z$ and $A > Z$, based on comparing the value of A against a threshold value Z.

Using these tests, C4.5 can use multi-value attributes and continuous attributes. A division process has the minimum number of splitting training cases. Its number can restrict self-evident division and too small division.

The construction process of C4.5 continues until dataset of subtree contains only single class. By this process, very complex tree and "overfites the data" tree are produced. C4.5 is pruned its decision tree that we expect reduction of tree size and abatement of overfites the data. The pruning of C4.5 applies for each decision tree's subtree from leaf. We evaluate each efficiency by using predicted error rate, before exchanging from subtree to leaf and after exchanging. If difference of predicted error rates is less than its threshold, the subtree is replaced by leaf. This process is repeated to root node, recursively. Using predicted error rate is considered for dealing with unknown dataset. The decision tree processed by such pruning, is expected that its size becomes small, and it has effectiveness for unknown dataset.

In the decision tree construction method by C4.5, the decision tree is constructed with the recurrent division of the training data. Therefore, there is a possibility of constructing a different decision tree when the number of training data is modified. In general, using a large number of training data tends to construct a more complex decision tree.

3.3 Genetic Programming

Genetic programming (GP) is a learning method based on the natural theory of evolution, and the flow of the algorithm is similar to genetic algorithm (GA). The difference between GP and GA is that GP has extended its chromosome to allow structural expression using function nodes and terminal nodes [Koza 1992a, Koza 1994b]. Using function nodes and terminal nodes, GP can search hierarchical computer programs by undergoing adaptation.

GP can be used for a search problem if its problem can be expressed as computer programs. GP's search space depend on function nodes and terminal nodes.

In using GP to a problem, GP has five major preparatory steps. The five steps are followings.

1. the set of terminals
2. the set of primitive functions
3. the fitness measure
4. the parameters for controlling the run
5. the method for designating a result and criterion for terminating a run

The decision tree construction with GP is given by the following procedures.

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.
2. The fitness value, which relates to the problem solving ability, for each individual of the GP population is calculated.

3. The next generation is generated by genetic operations.

   (a) The individual is copied by fitness value (reproduction).

   (b) A new individual is generated by intersection (crossover).

   (c) A new individual is generated by random change (mutation).

4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

In this chapter, the tree structure is used to express the decision tree. Therefore, we express the decision tree by using each attribute value and the class name as the terminal node and the condition sentence as the function node. (See example in figure 3.2) We defined some expressions for decision trees. In Figure 3.2, decision tree is expressed like LISP-code. "RPB" is defined as main GP tree. Both "ADF0" and "ADF1" are defined as each ADF tree. "IFLTE", "IFEQ" are function nodes. These functions requires four arguments, $arg1, arg2, arg3 and arg4$. The definitions of them are followings.

(IFLTE $arg1, arg2, arg3, arg4$**)** if $arg1$ is less than or equal to ($\leq$) $arg2$ then evaluate $arg3$, else then evaluate $arg4$

(IFEQ $arg1, arg2, arg3, arg4$**)** if $arg1$ is equal to(=) $arg2$ then evaluate $arg3$, else then evaluate $arg4$.

A, B, C and D express the attribute values from database. "T" and "F" express attribute value, and "N" and "P" express class name.

Using these expressions, GP can use database attributes more easily. Moreover, GP can apply continuous attributes. However, in general, database attributes are a lot of numbers, so GP's learning speed may become more slowly. The definition for continuous attributes may be cause of slow learning speed. It has trade-off relation between learning speed and expression compatibility.

Ordinarily, there is no method of adequately controlling the growth of the tree, because GP does not evaluate the size of the tree. Therefore, during the search process the tree may become overly deep and complex, or may settle to a too simple tree. There has been research on methods to have the program define functions itself for efficient use. One of the approaches is automatic function definition (or Automatically Defined Function: ADF), and this is achieved by adding the gene expression for the function definition to normal GP [Koza 1994a]. By implementing ADF, a more compact program can be produced, and the number of generation cycles can be reduced. More than one ADF can be defined in one individual.

In addition, the growth of the tree is controlled by evaluating the size of the tree. For example, an approach based on minimum description length (MDL) has been proposed concerning the evaluation of the size of the tree. In this chapter, we used the classification success ratio ($f_{hits(n)}$) and the number of composed nodes ($f_{nodes(n)}$), to define the fitness of the individual ($f_{fitness(n)}$).

$$f_{fitness(n)} = \alpha f_{hits(n)} + (1 - \alpha) f_{nodes(n)}$$

α is weight defined by ($0 \leqq \alpha \leqq 1$).

Here, two things are expected. One is that accuracy of the decision tree is raised while avoiding over-training in GP. And the other is that the decision tree generated can be

```
RPB:
    (IFLTE A "T"
        (IFEQ B "T"
            (IFEQ C "F" N P) ADF0) P) P)

ADF0:
    (IFEQ D "F" P N)

ADF1:
    C
```

Figure 3.2: Expression of GP's Chromosome

made comparatively compact. The fitness value and rule size of the best individual is influenced by the weighting of success rate or node size. We set weight α by pre-experiment.

3.4 Proposed Combined Learning Method

The classification learning by the decision tree can be trained in a short time, but when the noise is contained in the training data, the classification ability is rapidly deteriorated. On the other hand, a high classification ability is obtained by GP through training structural information, but more learning time is needed as the degree of learning freedom increases.

For such problems, we propose a combined learning method that combines the decision tree method (C4.5) and genetic programming. The proposed method consists of the following three steps:

1. First, using C4.5, construct appropriate decision trees.

2. Next, generate the genetic programming population which includes initial individuals converted from the decision trees.

3. Train the genetic program to construct the classification system.

It is observed that the discursive accuracy of classification becomes highly improved by the emergent property of interaction between two combined methods.

An outline of our proposed method is shown in figure 3.3.

When the decision tree is taken into the initial population of GP, it is necessary that variety in the initial population is not upheld [Niimi 1999a]. To ensure a variety of patterns taken into the initial population, we created decision trees by C4.5 using different number of training data.

In this GP, the classification system is expressed by decision tree. If you need decision rules, it is easy to convert from trees to rules by the process proposed by Quinlan [Quinlan 1995].

The key ideas of Quinlan's conversion process are follcwing.

- Every path from the root of an unpruned tree to a leaf gives one initial rule. The left-hand side of the rule contains all the conditions established by the path, and the right-hand side specifies the class at the leaf.

- Each such rule is simplified by removing conditions that do not seem helpful for discriminating the nominated class from other classes, using a pessimistic estimate of the accuracy of the rule.

- For each class in turn, all the simplified rules for that class are sifted to remcve rules that do not contribute to the accuracy of the set of rules as a whole.

- The sets of rules for the classes are then ordered to minimize false positive errors and a default class are chosen.

A decision tree can express feature of all data. On the other hand, it is difficult to express feature of all data by using a rule except using rule sets. GP is evaluated by all training data checking as one element of fitness function. One GP's individual need to

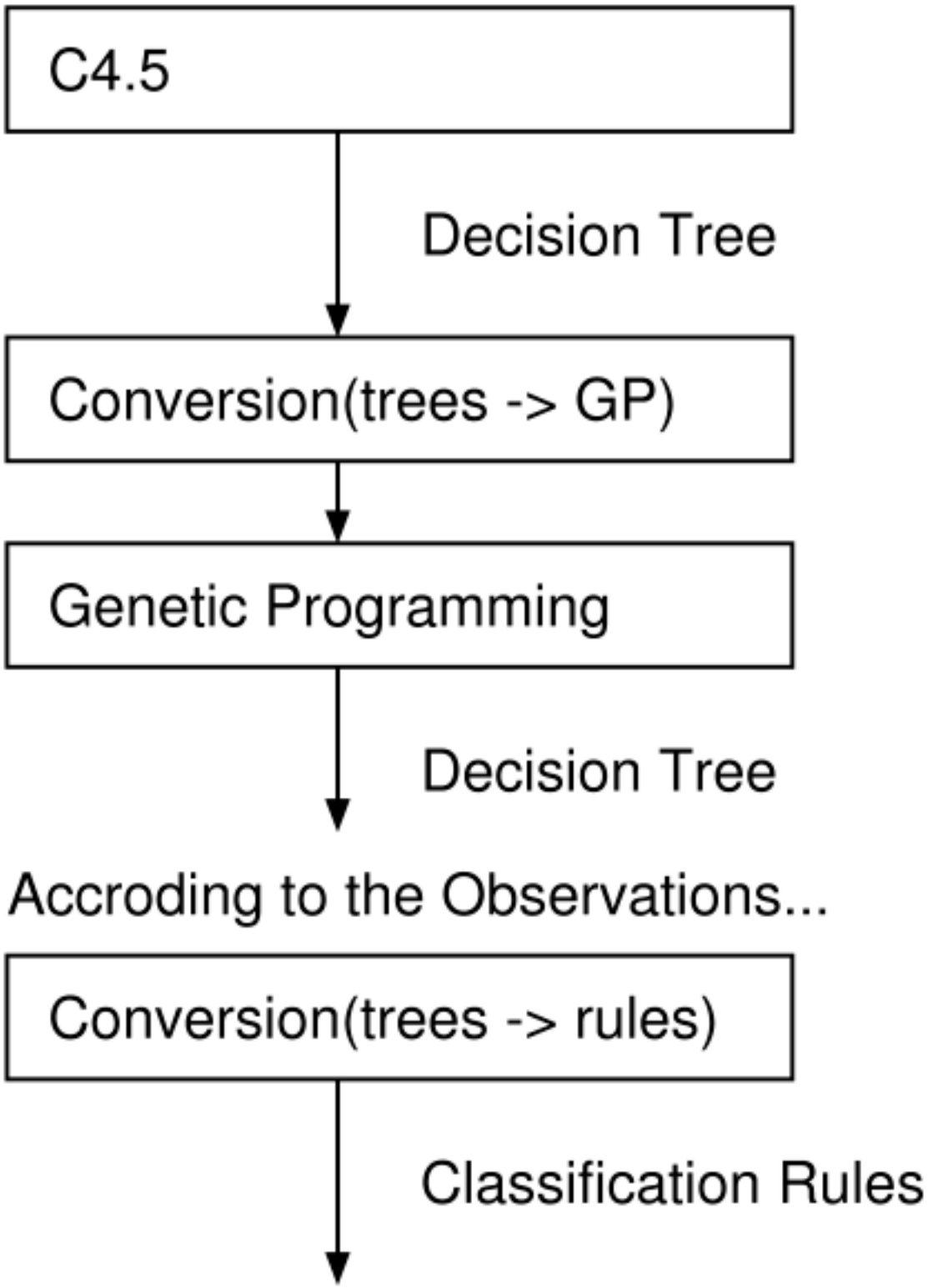


Figure 3.3: Flowchart of Approach of Proposed Combined Learning

express feature of all data, then we use GP as decision tree. If you need classification rules, it is easy to use GP that classification rules are converted from decision tree after GP's decision tree learning. (Refer to figure 3.4.)

The conversion from GP's tree to rules is an effective approach as post-processing of data mining. GP's tree often contains meaningless rules or useless rules because GP's structure expression has high flexibility and main GP operations are based on probability operations. For this problem, many approaches have been proposed, for example, modified scheme of GP's individual expression, expanded genetic operation, pruning operation on GP learning [Niimi 1999d]. But these techniques may become over-head operation on GP learning. In our proposed method, GP's initial population is improved and unless search is reduced, by taking C4.5 results into GP's initial population. In addition, conversion from decision tree to classification rules as post-processing, can be remove unable expressed rules and unable explained rules.

For conversion from decision tree to classification rules as post-processing, we propose the following steps.

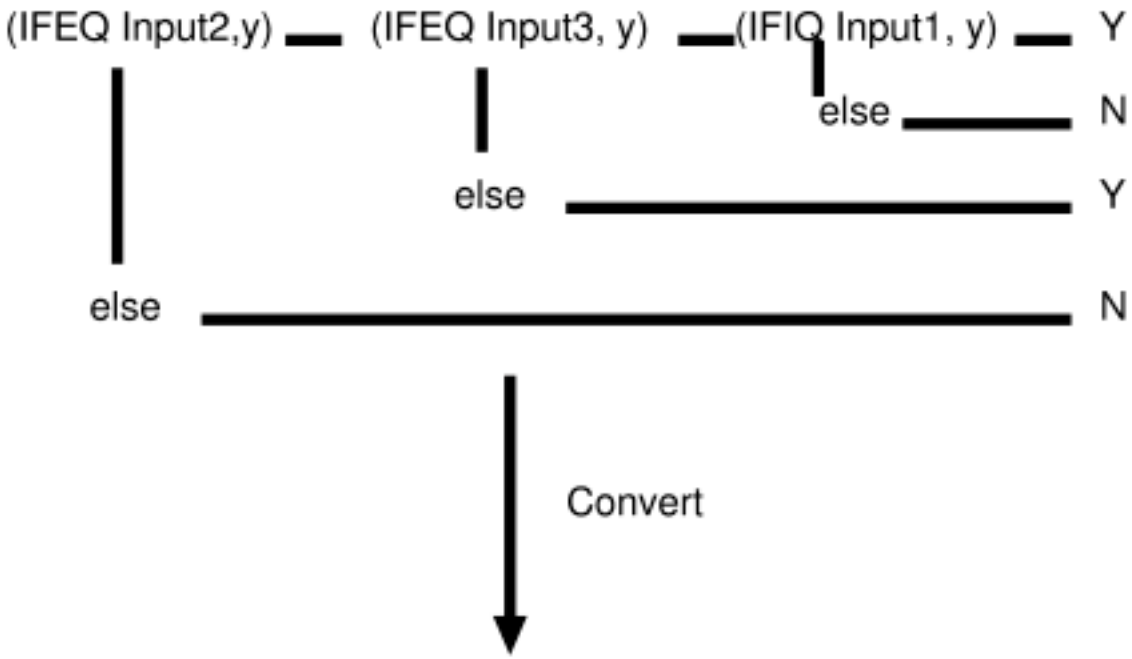
1. prune meaningless subtree in decision tree by GP
2. convert from tree to rules using C4.5 rule conversion method
3. remove meaningless rules or unable explained rules
4. check the rules by expert

Some examples of 1. are "$if A = Athen \ldots$", "$if A = BthenCelseC$". An example of 2. is comparative equation which is overflowed input data range.

The other consideration is calculation increasing by combined method. GP is most heavy calculation in the following four calculations, C4.5, conversion from C4.5 to GP, GP, conversion from GP to rules. The GP's calculation is far heavier compared with the other calculations. For such reason, the increase of calculation in the combined method is able to be actually ignored by GP calculation. For implementation of GP, both the set of terminals and the set of functions can be automatically defined by C4.5's result. For fitness function, general definition of evaluation tree can be used. By this way, our proposed method makes GP's implementation more easily, and the implementation times can be reduced.

3.5 Applications to Medical Diagnostic System

To verify the validity of the proposed method, we developed two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for meningoencephalitis. The occurrence of hypertension database contains a lot of continuous attributes. In experiment, continuous attributes are converted binary by threshold which is learned by GP. In general, it is difficult to use continuous values in GP. So this experiment is valuation for database which contains a lot of continuous attributes. The meningoencephalitis experiment contains many categorical attributes. In general, a lot of definition node makes GP slowly. So this experiment is validation for database which contains many attributes. We compared the results of the proposed method with prior ones. We got some comments for our results from domain experts. (In these case, domain expert means doctor.)



(Input2 = y) and (Input3 = y) and (Input1 = y) then Y

(Input2 = y) and (Input3 = y) and (Input1 != y) then N

(Input2 = y) and (Input3 != y) then Y

(Input2 != y) then N

Figure 3.4: Convert from Decision Tree to Rules

The database used for the occurrence of hypertension contains fifteen input terms and one output term. There are two kinds of Intermediate assumptions between the input terms and the output term [Ichimura 1997, Niimi 1999c]. Among the input terms, ten terms are categorized into a biochemical test related to the measurement of blood pressure for past five years, and the rest are "Sex", "Age", "Obesity Index", "γ-GTP", and "Volume of Consuming Alcohol". One output term represents whether the patient has had an attack of hypertension for the input record. The database has 1024 patient records. In this chapter, we selected 100 occurrence data and 100 non-occurrence data by random sampling, and this was used as the training data.

The parameters for GP used the following. (Refer to Table 3.1.) In this experiment, only a small percentage of GP individuals could be used as decision trees due to the large number of node types and large freedom for tree construction. Moreover, most of the input data have continuous value attributes. This seems to have caused the decrease in the fitness value of the GP search close to that of a random search. It was confirmed that the decision tree taken in as an initial individual was succeeded to the best individual, and it can be concluded that this influenced the improvement of the learning efficiency. (The results shown in Table 3.2, Figure 3.5, Figure 3.6.) The number of nodes in Table 3.2, and Figure 3.6 are not contained unused ADF tree.

The meningoencephalitis database is a medical treatment database concerning the discrimination diagnosis of meningoencephalitis. It consists of 140 patients. The database is described by 34 attribute about the past illness history, physical examinations, laboratory examinations, diagnosis, therapies, clinical courses, final status, and risk factors. The two classifications were bacillus and virus meningitis [Tsumoto 1999b, Niimi 2000d]. In this chapter, 32 attributes regarding sex, ages, etc. were used as well.

The parameters for GP used the following. (Refer to Table 3.3.) For this experiment, all methods achieved high accuracy, the medical database had removed noise well, and many attributes have discrete values. For the construction of decision tree by ADF-GP only, the fitness value did not increase quickly, but for construction of decision tree by combined ADF-GP and C4.5, the decision tree reached high accuracy comparatively quickly. It was confirmed that the decision tree taken in as an initial individual was succeeded to the best individual, and influenced the improvement of the learning efficiency. (The results shown in Table 3.4, Figure 3.7, Figure 3.8.) The number of nodes in Table 3.4, and Figure 3.8 are not contained unused ADF tree.

At both of the two experiments, the accuracy of the decision tree, the size, and the learning speed of our proposed method was obtained better result than normal GP's. The table of the results shows only generation cycle of GP, the calculation time of C4.5 and the time for conversion process from C4.5 to GP are shorter than calculation time for one generation of GP. By this experiment result, an increasing calculation by the combination was not seen.

The result decision tree of GP did not contain subtree defined by ADF-GP part. It is thought that subtree of ADF-GP worked for saving subtree which is not used for main tree. The effect is taken in the reduction of tree size of the main tree though it is not a control of the tree size which normal ADF-GP expects. According to the results of each method, the result of C4.5 is reflected considerably strongly in the proposed method. This is a result to which it is confirmed that an initial group with good performance can generate by building in C4.5, and this initial group has a strong influence in the GP learning.

For two experimental result, the expert comment is "appropriateable result". And,

Table 3.1: Parameters of GP (Hypertension)

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node	IFLTE, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (15 attributes), P, N, R
Number of training data	P(100), N(100)
Number of test data	1024

IFLTE,IFEQ: if less than or equal to($\leq$),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
P, N: occurrence (P), no-occurrence (N)

Table 3.2: Experimental Result (Reasoning Precision) by Each Technique.

	training data (%)	all data (%)	nodes (%)	generations
C4.5	98.5	77.1	29	—
ADF-GP	75.0	66.9	148	14328
C4.5 +ADF-GP	96.0	81.4	17	41

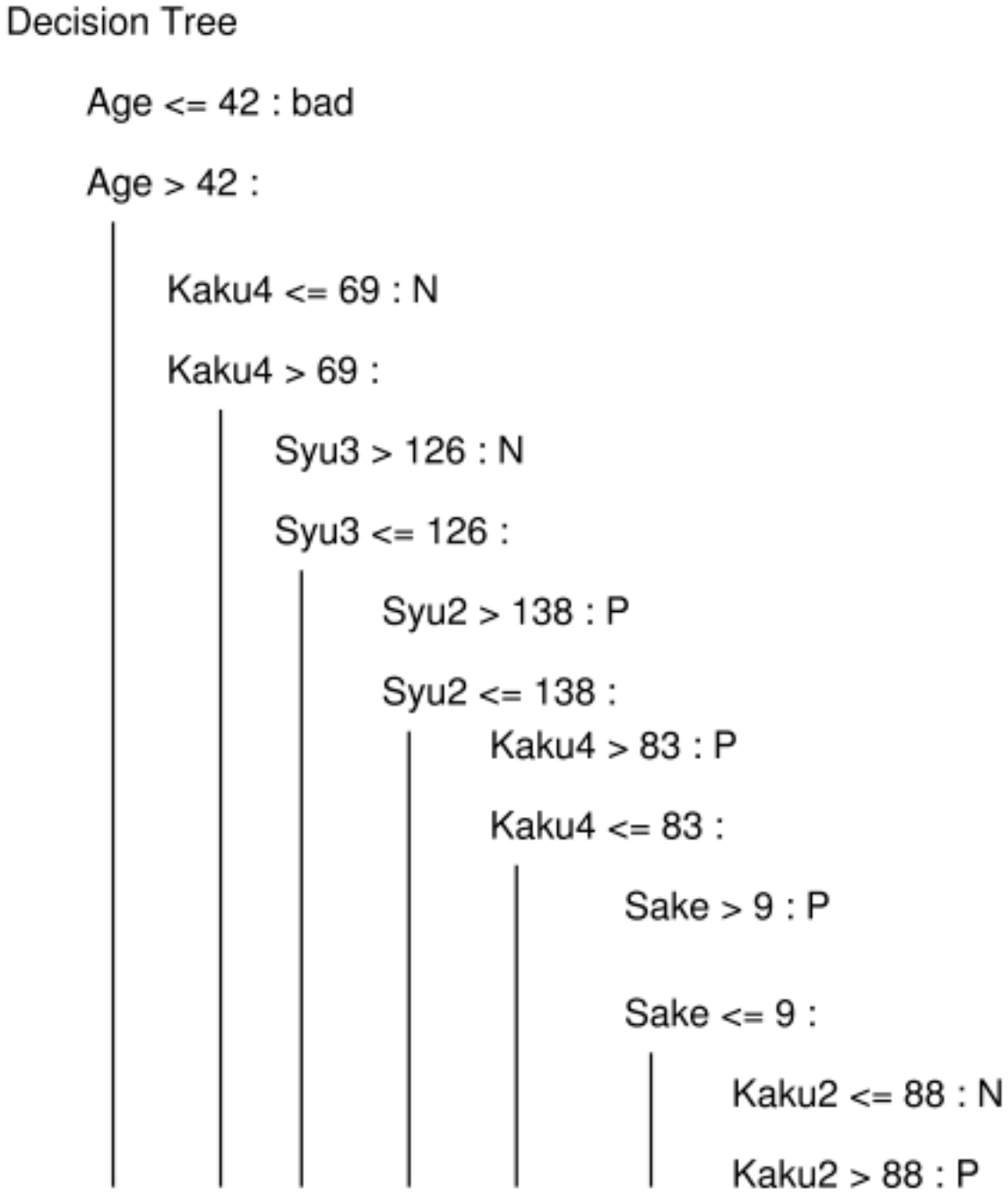


Figure 3.5: The Result from C4.5 (Hypertension)

```
(IFLTE Age 42 N
       (IFLTE Kaku4 83
              (IFLTE Syu3 126
                     (IFLTE Syu2 138 N P) P) P))
```

Figure 3.6: The Result from C4.5+ADF-GP (Hypertension)

Table 3.3: Parameters of GP (Meningoencephalitis)

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node	IFLTE, IFEQ, $*$, /, $+$, $-$, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (32 attributes), VIRUS, BACTERIA, R
Number of training data	VIRUS(49), BACTERIA(21)
Number of test data	140

IFLTE,IFEQ: if less than or equal to($\leq$),if equal to($=$)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
VIRUS,BACTERIA: Virus meningitis and bacillus meningitis

Table 3.4: Comparison of Generated Decision Trees (Number of Nodes and Fitness Value) by Each Technique

	training data (%)	all data (%)	nodes (%)	generations
C4.5	98.6	94.3	11	—
ADF-GP	88.6	82.9	21	7673
C4.5 +ADF-GP	95.7	97.1	11	686

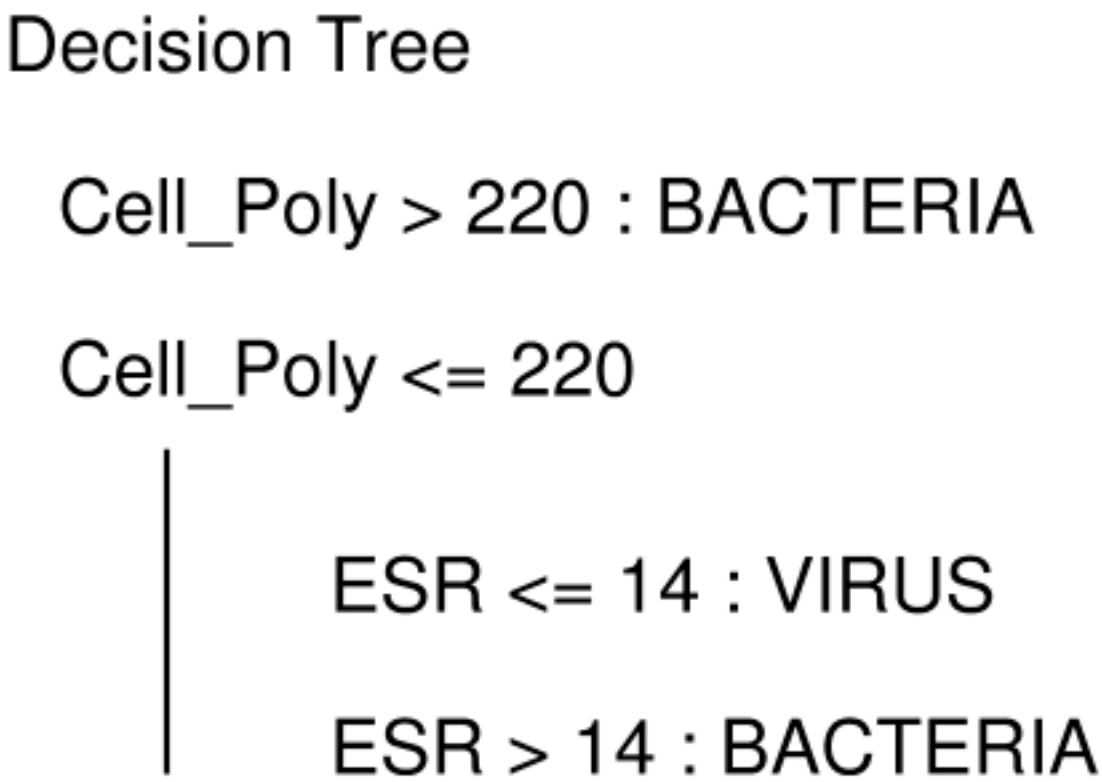


Figure 3.7: The Result from C4.5 (Meningoencephalitis)

```
(IFLTE Cell_Poly 225
        (IFLTE N
                (- Cell_Mono 14) N CT_FIND) P)
```

Figure 3.8: The Result from C4.5+ADF-GP (Meningoencephalitis)

other comment is that more interesting rules were contained by normal GP's result than the proposed method's. This comment is expressed for that GP rules include some interesting attributes (such as sex). A expert do not consider so much some attributes (sex etc.) to the diagnosis, but it can be understood according to background knowledges. The result is proof of the idea that an unexpected rule comes out easily by GP's probability operations. However, it seems that GP's probability operations could not present the effect by the influence from the initial individual taken from C4.5 in the proposed method. In general, unexpected rules can be applied only small number of data. Therefore, it is thought that our fitness function cannot express unexpected rules well. The accuracy of an unexpected rule and the decision tree becomes the relation of the trade-off. It is necessary to design fitness function which can strongly express the unexpectedness to generate an unexpected rule by using the proposed method.

In our proposed method, it is not necessary to consider about the inside of each algorithm. We think about C4.5 as the technique to construct the decision tree with the database and GP as the technique which can take the decision tree and can construct the decision tree with the database. Therefore, even if other decision tree constructing techniques are used instead of C4.5, the proposed method can be applied. It is thought that considering only to the method combination without considering each internal algorithm, it is possible to construct the combination learning method by using proposed method's analogy.

3.6 Conclusive Discussion

In this chapter, we proposed a decision tree construction method combined with C4.5 and GP. The proposed method has three steps. First, using C4.5, construct appropriate decision trees. Next, generate the genetic programming population which includes initial individuals converted from the decision trees. Finally, train the genetic program to construct the classification system.

The proposed method has some advantages. The first advantage is that the proposed method can improve decision tree's accuracy. The second advantage is that its learning speed becomes faster than normal GP's. The third advantage is that the design of a genetic programming becomes easy. And, it is possible to correspond also to datasets with a lot of numbers of data and attribute which contains the continuous value which is not treated easily by normal GP. It can be consider that method combination makes calculation heavier. But, the proposed method can save its combination overhead. Because it is controlled by the result of C4.5, there is a possibility that unexpected rules may not be extracted than a normal genetic programming.

We experimented by using two kinds of medical datasets to verify of proposed method. The domain experts gave the evaluation and comments on the results. The problem was taken up two things. One is intended for dataset which contained a lot of continuous values. The other is intended for dataset with a lot of numbers of attributes. We compared the results of three methods (using C4.5 only, using ADF-GP only, and using combined C4.5 and ADF-GP using the proposed method).

In the proposed method, the decision tree has been improved compared with GP and C4.5. The number of generations until best individual has been improved. The combination overhead was able to be disregarded because of the speed improvement of GP.

It can be concluded that the proposed method is more effective method as the decision tree construction method from the database.

Chapter 4

Combined Method of Genetic Programming and Association Rule Algorithm

Genetic programming (GP) usually has a wide search space and a high flexibility. So, GP may search for global optimum solution. But, in general, GP's learning speed is not so fast. Apriori Algorithm is one of association rule algorithms. It can be applied to large database. But, it is difficult to define its parameters without experience. We propose a rule generation technique from a database using GP combined with association rule algorithm. It takes rules generated by the association rule algorithm as initial individual of GP. The learning speed of GP is improved by the combined algorithm. To verify the effectiveness of the proposed method, we apply it to the decision tree construction problem from UCI Machine Learning Repository, and rule discovery problem from the occurrence of hypertension database. We compare the result of proposed method with prior ones.

4.1 Introduction

KDD and data mining were explained in the previous chapter, and the advantage which used genetic programming for data mining was described. The technique by which genetic programming was combined with the decision tree construction technique was proposed. It is easy to understand fairly complex knowledge because the decision tree expresses knowledge by using the hierarchical structure. However, growing the decision tree makes understanding the whole knowledge to become difficult. Therefore, it has possibility to miss relations between attributes in large-scale data mining. In this chapter, we think that genetic programming is applied to large-scale data mining. Large-scale data mining should understand the relations between a lot of attributes. Then, we think the association rules are used for data mining. The association rule is not classification rule, and it expresses a rule of co-occurrence relations between attributes. Therefore, using association rules can easy to understand co-occurrence relations between attributes.

The feature of genetic programming has already been described in Chapter 2. In Chapter 3, the decision tree was used as an individual expression, but the association rule is taken in this chapter.

On the other hand, there is the Apriori algorithm [Terabe 2000], a rule generating

technique for large databases. This is an association rule algorithm. The Apriori algorithm uses two values for rule construction: a support value and a confidence value. Depending on the setting of each index threshold, the search space can be reduced, or the candidate number of association rules can be increased. However, experience is necessary for setting an effective threshold.

Both techniques have advantages and disadvantages as above. In this chapter, we propose an extended genetic programming using apriori algorithm for rule discovery. By using the combined rule generation learning method, it is expected to construct a system which can search for flexible rules in large databases.

To verify the effectiveness of the proposed method, we discuss the learning method by genetic programming combined with the association rule construction method by the Apriori algorithm and the automatically defined function technique. We apply it to the decision tree construction problem from UCI Machine Learning Repository, and rule discovery problem from the occurrence of hypertension database. We compare the result of proposed method with prior ones.

The remainder of this chapter is organized as follows: Section 4.2 and Section 4.3 briefly introduces the Apriori and genetic programming respectively and how to use in data mining. Section 4.4 presents our proposed method. Section 4.5 presents two experiments for validation and we discuss the results. Final Section 4.6 draws conclusions.

4.2 Algorithm of Association Rule Construction

The association rule is one of the expressions which are often used by the basket analysis. In the association rule analysis, each case generally targets the transaction form data. The transaction form data is item set as a minimum unit of the data description. Co-occurrence pattern among item sets in the case is extracted as an association rule. (4.1) is one of association rule samples. The association rule expresses as co-occurrence pattern of the item set in the case data [Kitsuregawa 1997, Terabe 2000].

$$B \Rightarrow H \tag{4.1}$$

B is conditions of association rule.
H is conclusions of association rule.

By the ordinary analysis technique, a lot of calculation time needs to extract the association rule from a large database. The Apriori algorithm can extract the association rule from a large database by achieving the high efficiency of the search in realistic time.

In the basket analysis, there are two key principles which are called the monotonicity.

- If a set of items S is frequent (appears in at least fraction s of the baskets), then every subset of S is also frequent.
- Conversely, a set S cannot be frequent unless all its subsets are.

The following ideas can be used to find the basket's frequent itemsets in the basket analysis [Ullman 2000].

1. Proceed levelwise, finding first the frequent items (sets of size 1), then the frequent pairs, the frequent triples, etc. In our discussion, we concentrate on finding frequent pairs because:

(a) Often, pairs are enough.

(b) In many data set, the hardest part is finding the pairs; proceeding to higher levels takes less time than finding frequent pairs.

2. Find all maximal frequent itemsets (i.e., sets S of any size, such that no proper superset of S is frequent) in one pass or a few passes.

The following is taken from [Agrawal 1993, Agrawal 1994]. The algorithm called Apriori algorithm.

1. Given support threshold s, in the first pass we find the items that appear in at least fraction s of the baskets. This set is called L_1, the frequent items. Presumably there is enough main memory to count occurrences of each item.

2. Pairs of items in L_1 become the candidate pairs C_2 for the second pass. We hope that the size of C_2 is not so large that there is not room in main memory for an integer count per candidate pair. The pairs in C_2 whose count reaches s are the frequent pairs, L_2.

3. The candidate triples, C_3 are those sets $\{A, B, C\}$ such that all of $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ are in L_2. On the third pass, count the occurrences of triples in C_3; those with a count of at least s are the frequent triples, L_3.

4. Proceed as far as you like (or the sets become empty). L_i is the frequent sets of size i; C_{i+1} is the set of sets of size $i + 1$ such that each subset of size i is in L_i.

In the Apriori algorithm, the candidate of the association rule is evaluated by using two values, support value and confidence value, while searching for the association rules. The support value of association rule R ($sup(R)$) is defined as (4.2). The confidence value of association rule R ($sup(R)$) is defined as (4.3).

$$sup(R : B \Rightarrow H) = \frac{n(B \cup H)}{N} \tag{4.2}$$

$n(B \cup H)$ means a number of cases containing B or H items, N means a number of all cases, and $B \cap H = \emptyset$.

$$conf(R : B \Rightarrow H) = \frac{n(B \cup H)}{n(B)} \tag{4.3}$$

$n(B \cup H)$ means a number of cases containing B or H items, $n(B)$ means a number of cases containing B items, and $B \cap H = \emptyset$.

The Apriori algorithm, for support value and confidence value, is used minimum support value and minimum confidence value as those thresholds. If a candidate of association rule cannot fill these minimum values, we assume that its rule expresses low association. Then, its rule is excluded evaluation to reduce search space consecutively. Therefore, the Apriori algorithm can search faster than the other association rule analysis techniques.

By operating each minimum value, the candidate number of association rule can be increased or the range of the search space can be reduced. However, it is possible that an unexpected rule cannot be extracted by reducing the range of the search space. Moreover, the load of the expert who analyzes the rule increases when there are a lot of association rule candidates, and it is a possible that it becomes difficult to search for a useful rule.

4.3 Genetic Programming

Genetic programming (GP) is a learning method based on the natural theory of evolution, and the flow of the algorithm is similar to genetic algorithm (GA). The difference between GP and GA is that GP has extended its chromosome to allow structural expression using function nodes and terminal nodes [Koza 1992a, Koza 1994b]. Using function nodes and terminal nodes, GP can search hierarchical computer programs by undergoing adaptation.

GP can be used for a search problem if its problem can be expressed as computer programs. GP's search space depend on function nodes and terminal nodes.

In using GP to a problem, GP has five major preparatory steps. The five steps are followings.

1. the set of terminals
2. the set of primitive functions
3. the fitness measure
4. the parameters for controlling the run
5. the method for designating a result and criterion for terminating a run

The decision tree construction with GP is given by the following procedures.

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.
2. The fitness value, which relates to the problem solving ability, for each individual of the GP population is calculated.
3. The next generation is generated by genetic operations.
   (a) The individual is copied by fitness value (reproduction).
   (b) A new individual is generated by intersection (crossover).
   (c) A new individual is generated by random change (mutation).
4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

In this chapter, the tree structure is used to express the decision tree. Therefore, we express the decision tree by using each attribute value and the class name as the terminal node and the condition sentence as the function node. (See example in figure 4.1) We defined some expressions for decision trees. In figure 4.1, decision tree is expressed like LISP-code. "RPB" is defined as main GP tree. Both "ADF0" and "ADF1" are defined as each ADF tree. "IFLTE", "IFEQ" are function nodes. These functions requires four arguments, $arg1, arg2, arg3 and arg4$. The definitions of them are followings.

(IFLTE $arg1, arg2, arg3, arg4$**)** if $arg1$ is less than or equal to ($\leq$) $arg2$ then evaluate $arg3$, else then evaluate $arg4$

(IFEQ $arg1, arg2, arg3, arg4$**)** if $arg1$ is equal to($=$) $arg2$ then evaluate $arg3$, else then evaluate $arg4$.

```
RPB:
    (IFLTE A "T"
        (IFEQ B "T"
            (IFEQ C "F" N P) ADF0) P) P)

ADF0:
    (IFEQ D "F" P N)

ADF1:
    C
```

Figure 4.1: Expression of GP's Chromosome

A, B, C and D express the attribute values from database. "T" and "F" express attribute value, and "N" and "P" express class name.

Using these expressions, GP can use database attributes more easily. Moreover, GP can apply continuous attributes. However, in general, database attributes are a lot of numbers, so GP's learning speed may become more slowly. The definition for continuous attributes may be cause of slow learning speed. It has trade-off relation between learning speed and expression compatibility.

Ordinarily, there is no method of adequately controlling the growth of the tree, because GP does not evaluate the size of the tree. Therefore, during the search process the tree may become overly deep and complex, or may settle to a too simple tree. There has been research on methods to have the program define functions itself for efficient use. One of the approaches is automatic function definition (or Automatically Defined Function: ADF), and this is achieved by adding the gene expression for the function definition to normal GP [Koza 1994a]. By implementing ADF, a more compact program can be produced, and the number of generation cycles can be reduced. More than one ADF can be defined in one individual.

In addition, the growth of the tree is controlled by evaluating the size of the tree. For example, an approach based on minimum description length (MDL) has been proposed concerning the evaluation of the size of the tree. In this chapter, we used the classification success ratio ($f_{hits(n)}$) and the number of composed nodes ($f_{nodes(n)}$), to define the fitness of the individual ($f_{fitness(n)}$).

$$f_{fitness(n)} = \alpha f_{hits(n)} + (1 - \alpha) f_{nodes(n)}$$

α is weight defined by $(0 \leqq \alpha \leqq 1)$.

Here, two things are expected. One is that accuracy of the decision tree is raised while avoiding over-training in GP. And the other is that the decision tree generated can be made comparatively compact. The fitness value and rule size of the best individual is influenced by the weighting of success rate or node size. We set weight α by pre-experiment.

4.4 Approach of Proposed Combined Learning

Apriori Algorithm can be applied to large database. But, it is difficult to define its parameters without experience. On the other hand, a high classification ability is obtained by GP through training structural information, but more learning time is needed as the degree of learning freedom increases.

It is observed that the discursive accuracy of classification becomes highly improved by the emergent property of interaction between two combined methods.

To make up for the advantage and the disadvantages of the Apriori algorithm and GP, we propose a rule discovery technique which combines GP with the Apriori algorithm. By combining each technique, it is expected searching for flexible rules from a large database. An outline of our proposed technique is shown in figure 4.2.

The following steps are proposed for the rule discovery technique.

1. First, the Apriori algorithm generates the association rule.

2. Next, the generated association rules are converted into decision trees which are taken in as initial individuals of GP. The decision trees are trained by GP learning.

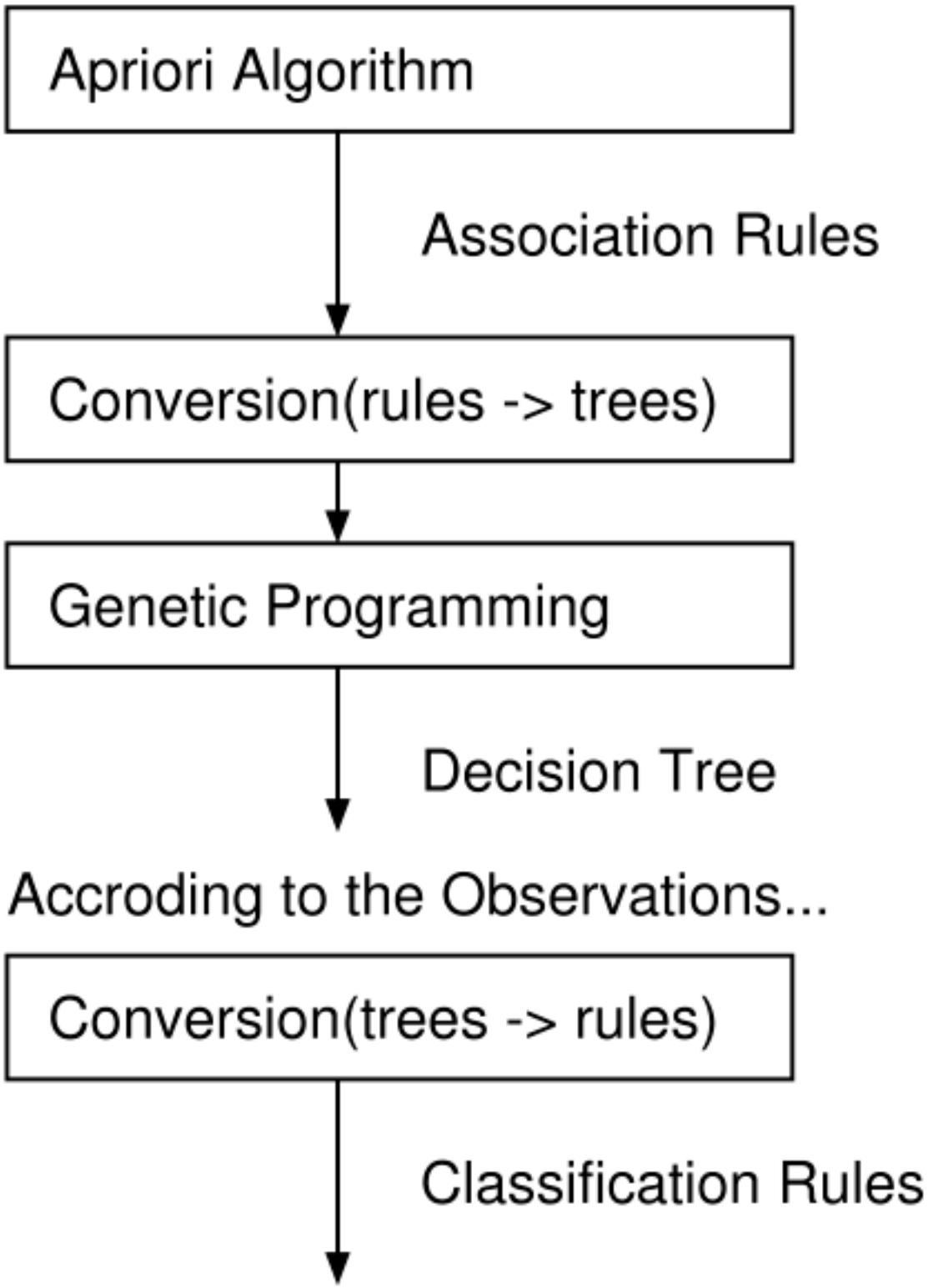


Figure 4.2: Flowchart of Approach of Proposed Combined Learning

3. The final decision tree is converted into classification rules.

This allows effective schema to be contained in the initial individuals of GP. As a result, it is expected to improve the GP's learning speed and its classification accuracy.

For conversion from the association rules into decision trees, we use the following procedures. (Refer to figure 4.3.)

1. For the first process, the route of the decision tree is constructed, assuming the conditions of the association rule as the attribute-based tests of the decision tree.

2. In the next process, the conclusions of the association rule is appended on the terminal node of this route.

3. Finally, the terminal nodes which are not defined by the association rule are assigned candidate nodes at random.

In this conversion, one decision tree is converted by one rule. When the decision tree is taken into the initial population of GP, it is necessary that variety in the initial population is not upheld [Niimi 1999a]. Because Apriori algorithm can make a lot of rules, it is easy to maintain the variety of an initial population of GP. To later GP learning, the accuracy of the decision tree need not be so high though it is possible to convert the decision tree with higher accuracy from more rules. Moreover, because the rules only have to be able to be extracted for GP's initial population, the number of rules is not so necessary.

In this GP, the classification system is expressed by decision tree. If you need decision rules, it is easy to convert from trees to rules by the process proposed by Quinlan [Quinlan 1995].

The key ideas of Quinlan's conversion process are following.

- Every path from the root of an unpruned tree to a leaf gives one initial rule. The left-hand side of the rule contains all the conditions established by the path, and the right-hand side specifies the class at the leaf.

- Each such rule is simplified by removing conditions that do not seem helpful for discriminating the nominated class from other classes, using a pessimistic estimate of the accuracy of the rule.

- For each class in turn, all the simplified rules for that class are sifted to remove rules that do not contribute to the accuracy of the set of rules as a whole.

- The sets of rules for the classes are then ordered to minimize false positive errors and a default class are chosen.

In the proposed technique, a large number of rules made from Apriori algorithm can be brought together by GP. The rules are made from Apriori algorithm, the decision tree is learned by GP, and the result is converted into the rules. This algorithm seems to cause the over-head in this operation seemingly by a lot of conversions. But, a decision tree can express feature of all data. On the other hand, it is difficult to express feature of all data by using a rule except using rule sets. GP is evaluated by all training data checking as one element of fitness function. One GP's individual need to express the classification rule set of all data, then we use GP as a decision tree. If you need classification rules, it is easy to use GP that classification rules are converted from decision tree after GP's

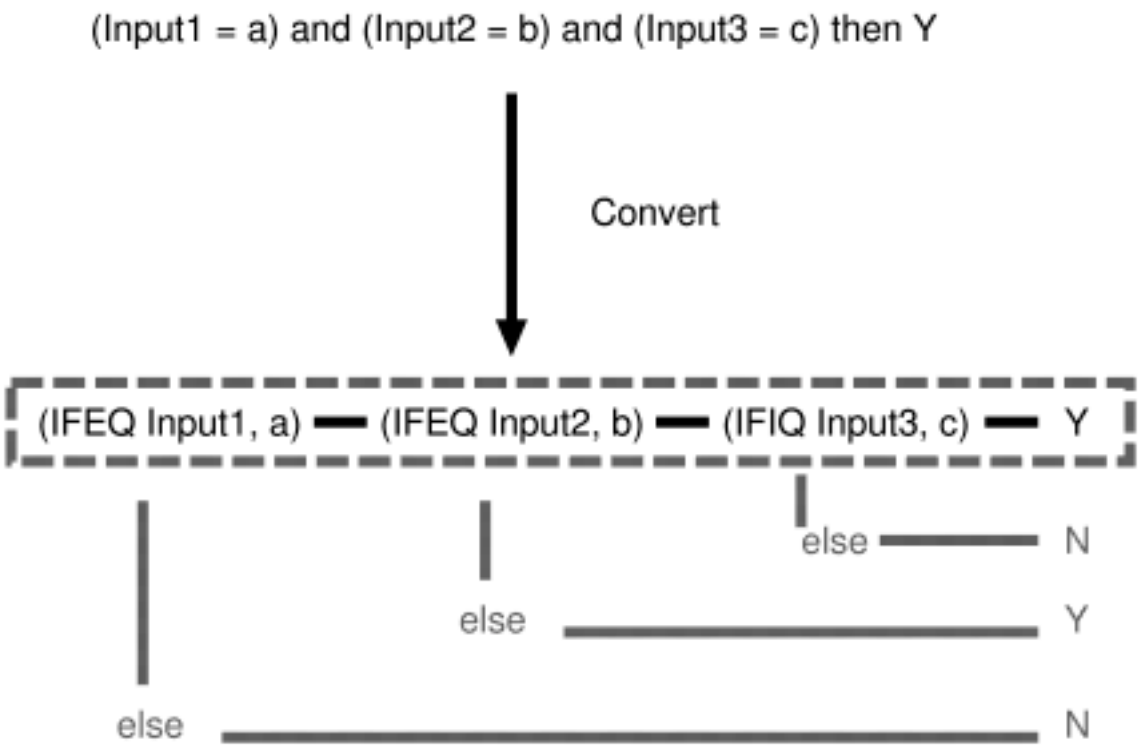
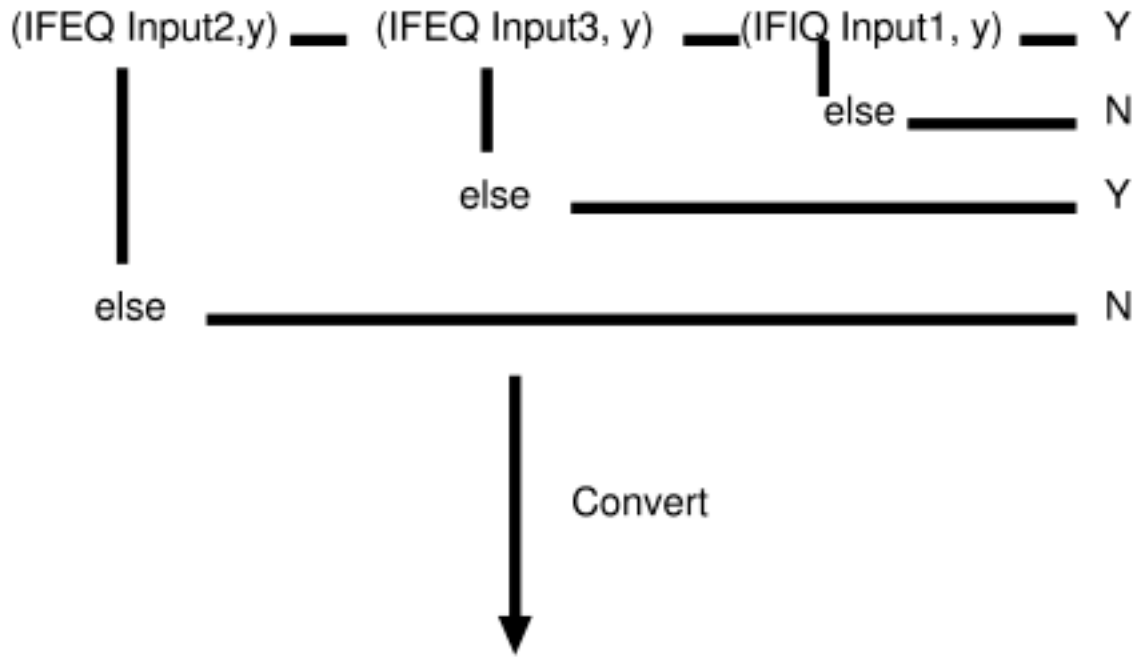


Figure 4.3: Convert from Association Rules to Decision Tree



(Input2 = y) and (Input3 = y) and (Input1 = y) then Y

(Input2 = y) and (Input3 = y) and (Input1 != y) then N

(Input2 = y) and (Input3 != y) then Y

(Input2 != y) then N

Figure 4.4: Convert from Decision Tree to Rules

decision tree learning. (Refer to figure 4.4.) Morecver, because GP is learned after the rules are extracted, the rule which contains a continuous value can be easily learned.

The rule which contains a continuous value is learned by the change of the threshold in the function node of GP.

The conversion from GP's tree to rules is an effective approach as post-processing of data mining. GP's tree often contains meaningless rules or useless rules because GP's structure expression has high flexibility and main GP operations are based on probability operations. For this problem, many approaches have been proposed, for example, modified scheme of GP's individual expression, expanded genetic operation, pruning operation on GP learning [Niimi 1999d]. But these techniques may become cver-head operation on GP learning. In our proposed method, GP's initial population is improved and unless search is reduced, by taking Apriori algorithm results into GP's initial population. In addition, conversion from decision tree to classification rules as post-processing, can be remcve unable expressed rules and unable explained rules.

For conversion from decision tree to classification rules as post-processing, we propose the following steps.

1. prune meaningless subtree in decision tree by GP
2. convert from tree to rules using C4.5 rule conversion method
3. remcve meaningless rules or unable explained rules
4. check the rules by expert

Some examples of 1. are "$if A = Athen \ldots$", "$if A = BthenCelseC$". An example of 2. is comparative equation which is overflowed input data range.

The other consideration is calculation increasing by combined method. GP is most heavy calculation in the following four calculations, Apriori algorithm, conversion from rules to GP, GP, conversion from GP to rules. The GP's calculation is far heavier compared with the other calculations. For such reason, the increase of calculation in the combined method is able to be actually ignored by GP calculation. For implementation of GP, both the set of terminals and the set of functions can be automatically defined by Apriori algorithm's result. For fitness function, general definition of evaluation tree can be used. By this way, our proposed method makes GP's implementation more easily, and the implementation times can be reduced.

4.5 Apply to Decision Tree Construction from Database

To verify the validity of the proposed method, we applied it to the house-votes data from UCI Machine Learning Repository [Blake 1998], and medical database for occurrence of hypertension [Ichimura 1997, Niimi 1999c]. From here on all occurrence of GP uses Automatically Defined Function Genetic Programming (ADF-GP) [Koza 1994a] including the proposed method. In the proposed method, we took the association rule generated by Apriori algorithm as initial individuals of GP. We compared the results of the proposed method against GP. We did not compare GP with the Apriori algorithm because of the difference of the expression such as rules and decision trees . We use house-votes database as small test database expressed by discrete values, and hypertension database

as large test expressed by continuous values. Validity of the expert (doctor) was had to be evaluated to the result concerning the experiment on hypertension.

For evaluation, we used house-votes data from UCI Machine Learning Repository [Blake 1998]. We compared the results of the proposed method with GP. The evaluation data contains 16 attributes and 2 classes. The attributes are for example "handicapped-infants" and "water-project-cost-sharing " etc. They are expressed by 3 values: "y ", "n", and "?". And the 2 classes are "democrat" and "republican ". 50 cases out of the total 435 data of house-votes were used for training data.

We extracted the association rule from the database by the Apriori algorithm. We applied the Apriori algorithm to a data set excluding data with the "?" value, because "?" value means "others". In the following experiment, we used minimum support value (= 30) and minimum confidence value (= 90). As a result of the experiment, 75 rules were generated.

Next, the above generated 75 rules were taken into the initial individual. Table 4.1 was used concerning the parameter of GP in each experience. In 4.1, GP makes "y" and "democrat", "n" and "republican" to the same treatment as dividing attributes because of mounting. The result of the evaluation of each fitness value is shown in figure 4.5, and the result of best individual is shown in table 4.2. The best decision tree of each method is shown by figure 4.6, figure 4.7. In these tables do not contain unuseful branches using post-processing.

By using GP, fitness value did not improve rapidly. However, the proposed method showed fast learning and achieved high accuracy. Comparing the best individual results, the proposed method showed better results than GP, except for accuracy against the training data. Concerning the results of training data, GP may have shown overfitting. Furthermore, in figure 4.5, GP's fitness value did not change at 100–300 generations. We consider this no-learning term was caused by overfitting. But its proof could not be obtained by only this result.

The rules were converted from the constructed decision tree removing invalid rules and meaningless rules. The rules' total accuracy was 94.8%.

We applied a medical diagnostic system for the occurrence of hypertension. We compared the results of proposed method with GP. Most of the data values are expressed as continuous values, and the size of the database is larger than the house-votes database. The domain expert gave the evaluation and comments on the results.

The occurrence of hypertension database contains 15 input terms and 1 output term. There are 2 kinds of intermediate assumptions between the input terms and the output term [Ichimura 1997, Niimi 1999c]. Among the input terms, 10 terms are categorized into a biochemical test related to the measurement of blood pressure for past five years, and the other terms are "Sex", "Age", "Obesity Index", "γ-GTP", and "Volume of Alcohol Consumption". 1 output term represents whether the patient has an attack of hypertension for the input record. The database has 1024 patient records. In this chapter, we selected 100 occurrence data and 100 no-occurrence data by random sampling, and this was used as the training data.

The association rule has been extracted from the database by the Apriori algorithm. The Apriori algorithm was used after these attributes had been converted into binary attributes using the average of each data, because the continuous value attributes were included in this database. To search for the relationship between the minimum support value and the minimum confidence value and the number of rules, we experimented with the threshold patterns. (Refer to the result table 4.3) In the following experiment, we

Table 4.1: Parameters of GP(House-votes)

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node	IFEQ, ADF0, ADF1
Terminal node	Attribute value of database (16 attributes), y, n, ?, democrat, republican
Number of training data	democrat:31, republican:19
Number of test data	435

IFEQ: if equal to (=)
ADF0, ADF1: the function definition gene expanded by ADF
y, n, ?: yes(y), no(n), others(?)

Table 4.2: Experiment Best Individuals Result (House-votes).

	training (%)	all data (%)	nodes	depths	generations
ADF-GP	100.0	86.0	11	3	586
Apriori + ADF-GP	98.0	92.9	9	2	235

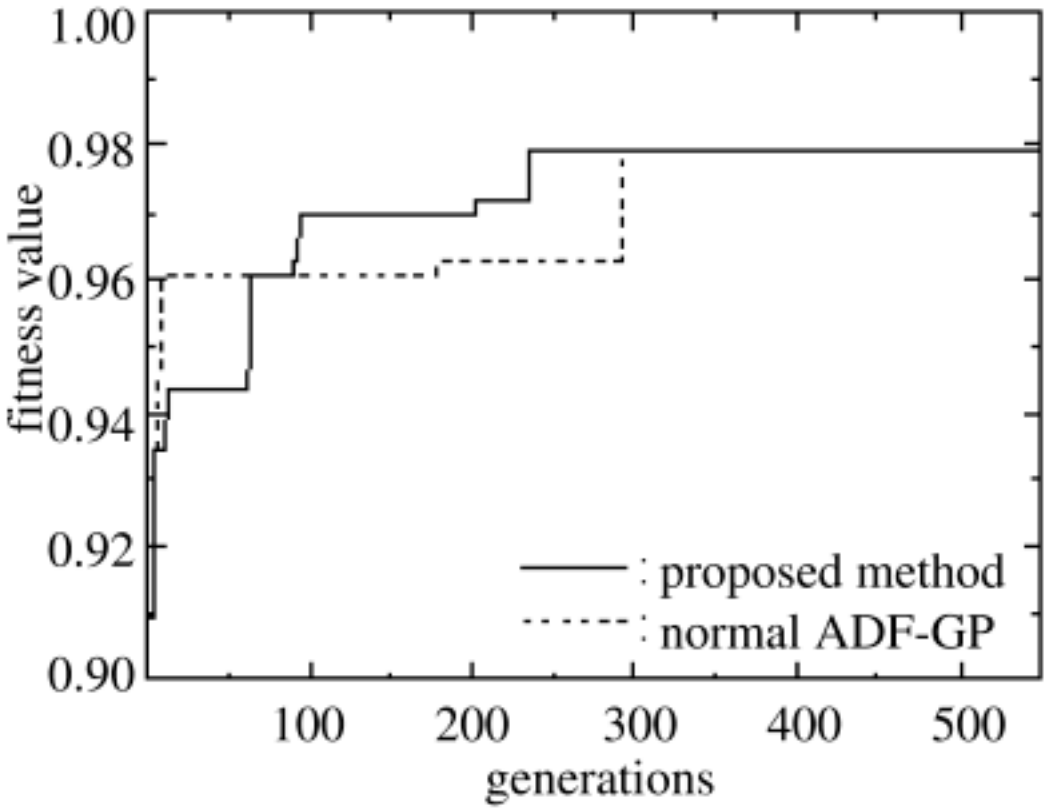


Figure 4.5: Evaluation of Each Fitness Value (Training Data)

```
RPB:
    (IFEQ y physician-free-freeze
        (IFEQ physician-fee-freeze mx-missile
            ADF0 adoption-of-the-budget-resoltion)
            y)
ADF0:
    handicapped-infants
ADF1:
    el-salvador-aid
```

Figure 4.6: The Result from ADF-GP (House-votes)

```
RPB:
    (IFEQ mx-missile physician-free-freeze synfuels-corporation-cutback
        (IFEQ mx-missile y
            mx-missile adoption-of-the-budget-resolution)

ADF0:

    handicapped-infants

ADF1:

    democrat
```

Figure 4.7: The Result from Apriori+ADF-GP (House-votes)

used minimum support value (= 30) and minimum confidence value (= 90).

Next, the 33 rules generated by the Apriori algorithm were taken into the initial individual. The continuous values are learned at the same time by the change of the threshold of the function node while GP learning. Table 4.4 was used concerning the parameter of GP in each experience. "P" and "1", "N" and "0" do the same treatment by convenience in mounting in GP. The result of best individual is shown in table 4.5. The best decision tree of each method is shown by figure 4.8, figure 4.9. In these tables do not contain unuseful branches using post-processing.

By using GP, fitness value did not improve rapidly. However, the proposed method showed fast learning and achieved high accuracy.

When the rules were converted from the decision tree, invalid rules and meaningless rules were removed. Each ratio of the number of effective rules to generation rules was 37.5% (by GP) and 50.0% (by proposed method). (Table 4.6 shows 3 rules generated with each technique, chosen by the highest support value.)

By using GP, many invalid rules and many rules which were difficult to interpret were generated. Compared to GP, the proposed method showed decrease in the support value and improvement in accuracy. The proposed technique improved the ratio of effective rules and the accuracy.

At both of the two experiments, the accuracy of the decision tree, the size, and the learning speed of our proposed method was obtained better result than normal GP's. The table of the results shows only generation cycle of GP, but, the calculation time of Apriori algorithm and the time for conversion process from Apriori algorithm to GP are shorter than calculation time for one generation of GP. By this experiment result, an increasing calculation by the combination was not seen.

When the rules were extracted from the result of both experiment's generated decision trees, it was so difficult to extract as post-processing. It is thought that the reason is that a lot of rules with a difficult interpretation are included by the probability operation of GP. However, the threshold to which the proposed technique is more significant than normal GP is often especially learned in the learning of continuous values in the experiment on hypertension. ADF part was not used too effectively concerning the use of ADF in the experiment on house-votes. The reason is that the small decision tree had a enough size for the problem because the problem was too easy to make the decision tree.

For hypertension experimental result, the expert comment is "appropriateable result". And, other comment is that more interesting rules were contained by normal GP's result than the proposed method's. This comment is expressed for that GP rules include some interesting attributes (such as sex). A expert do not consider so much some attributes (sex etc.) to the diagnosis, but it can be understood according to background knowledges. The result is proof of the idea that an unexpected rule comes out easily by GP's probability operations. However, it seems that GP's probability operations could not present the effect by the influence from the initial individual taken from Apriori algorithm in the proposed method. In general, unexpected rules can be applied only small number of data. Therefore, it is thought that our fitness function cannot express unexpected rules well. The accuracy of an unexpected rule and the decision tree becomes the relation of the trade-off. It is necessary to design fitness function which can strongly express the unexpectedness to generate an unexpected rule by using the proposed method.

In our proposed method, it is not necessary to consider about the inside of each algorithm. We think about Apriori algorithm as the technique to extract the association rules with the database and GP as the technique which can take the decision tree and

Table 4.3: Relations between Thresholds and Number of Rules

Minimum Support Value	Minimum Confidence Value	Rules
25	75	396
30	75	125
25	90	187
30	90	33

Table 4.4: Parameters of GP(Hypertension)

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method Elitist strategy
Function node	IFLTH, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (15 attributes), P, N, R
Number of training data	Occurrence (100) , No-occurrence (100)
Number of test data	1024
Maximum generations	20000

IFLTH,IFEQ: if less than or equal to ($\leq$),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
P, N: occurrence (P), no-occurrence (N)

Table 4.5: Experiment Best Individuals Result (Hypertension).

	training (%)	all data (%)	nodes	depths	generations
ADF-GP	89.5	66.3	41	6	18553
Apriori + ADF-GP	89.5	74.9	49	4	671

Table 4.6: Comparison of Generated Rules (Size and Fitness Value)

Technique	Size	Support Value(%)	Wrong(%)
ADF-GP	4	41.4	48.4
	2	36.0	24.1
	4	22.6	8.2
Apriori+ADF-GP	2	17.7	39.2
	4	15.5	13.2
	4	13.8	19.2

```
RPB:
        (/ N
           (- ADF1 Kaku2))

ADF0:
        (/ 152 Sex)

ADF1:
        (IFLTE (IFEQ Syu4 Syu2 P Kaku3)
                (/ (- 120 Age) ADF0) Syu2

                (IFLTE Kaku4
                        (- 120 Age)

                        (IFLTE (- Syu2 Kaku4) Kaku4 Syu2
                                (IFLTE Kaku4
                                (/ (- Syu2 Kaku4) ADF0) Syu4 Kaku2)) Kaku2))
```

Figure 4.8: The Result from ADF-GP (Hypertension)

```
RPB:
        (IFLTE 45 Age
                (IFLTE 76 Kaku4 P
                        (IFLTE 75 Kaku5
                                (IFLTE 75 Kaku2 P N) N))
                (IFLTE (- (IFEQ ADF1 Sex Age Kaku1)
                        (/ Kaku2 Sake)) Kaku5
                        (IFLTE (IFLTE 76 Kaku4 P 90) Kaku5
                                (IFLTE 75 Kaku2 P N) N) N))

ADF0:
        (/ Syu4 Kaku2)

ADF1:
        (/ Kaku3
            (+ Kaku5 P))
```

Figure 4.9: The Result from Apriori+ADF-GP (Hypertension)

can construct the decision tree with the database. Therefore, even if other association rules extracting techniques are used instead of Apriori algorithm, the proposed method can be applied. It is thought that considering only to the method combination without considering each internal algorithm, it is possible to construct the combination learning method by using proposed method's analogy.

4.6 Conclusive Discussion

In this chapter, we proposed the rule discovery technique from the database using genetic programming combined with Apriori algorithms. The proposed method has three steps. First, using Apriori algorithm, extract association rules. Next, generate the genetic programming population which includes initial individuals converted from the association rules. Finally, train the genetic program to construct the classification system.

The proposed method has some advantages. The first advantage is that the proposed method can improve decision tree's accuracy. The second advantage is that its learning speed becomes faster than normal GP's. The third advantage is that the design of a genetic programming becomes easy. And, it is possible to correspond also to datasets with a lot of numbers of data and attribute which contains the continuous value which is not treated easily by normal GP. It can be consider that method combination makes calculation heavier.

We experimented by using two kinds of datasets to verify of proposed method. For one experiment, the domain expert gave the evaluation and comments on the results. Two problems were discussed. One is intended for dataset with small test database expressed by discrete values. The other is intended for dataset which contained a lot of continuous values. We compared the results of these methods (using ADF-GP only, and using combined Apriori algorithm and ADF-GP using the proposed method).

In the proposed method, the decision tree has been improved compared with GP and Apriori algorithm. The number of generations until the best individual has been decreased. The combination over-head was able to be disregarded because of the speed improvement of GP. Though the result of suppressing the overfitting was seen, we did not consider the analysis of its mechanism.

In the future, we will research the following 4 topics. The first topic is to apply the method to other verifications [Niimi 2000b, Niimi 2000c, Niimi 2000d]. The second topic is to discuss the conversion algorithm from the association rule to a decision tree with high accuracy. The third topic is to extend the proposed method to multi-value classification problems. The fourth topic is to study a theoretical analysis about the mechanism of the overfitting.

Chapter 5

Extended Boolean Decision Tree using Genetic Programming with Logical Functions for Knowledge Discovery

It is easy for an unexpected decision tree to be generated in the decision tree construction with genetic programming because the probability operation is contained. In the description of the decision tree by normal genetic programming, the division conditions by the attribute are connected with AND operator, and the tree evaluates effectiveness as a rule. However, if the description of the function node is modified, a more flexible rule is sure to be able to be generated. In this chapter, we show that the description of a more flexible decision tree is possible by the addition of the OR function and NOT function to the function node group.

5.1 Introduction

It can be expected as the use of genetic programming to data mining to find an unexpected knowledge, because genetic programming has a probabilistic operation in the evolution calculation. In genetic programming, the knowledge representation can be used a structural expression to express the chromosome, so it can be widely applied from decision trees to rules.

However, by the evaluation of the individual according to the fitness value function, it is better that genetic programming's chromosome expression can cover the entire knowledge like the decision tree. In the description of the decision tree by genetic programming, in general the division condition by the attribute is connected with AND, and it is evaluated as a rule.

However, it is possible to mount in genetic programming if the chromosome expression and the fitness value function can be defined. Then, other knowledge representations can be mounted on genetic programming.

In this chapter, we extend the decision tree and the rule expression using if-else function. First of all, we define the rule expression by the AND function which is referred to the association rule. Then, a more flexible expression of the decision tree and the rule can be used by using the OR function.

GP which uses the boolean function as a function node has already been proposed. However, these are designed for the logical function operation, and it is not easy to use the function nodes in data Mining. Therefore, to take the attribute value from the data base easily, the function node was defined in the proposal technique. The modification can be produced by replacing the definitions of the function node of genetic programming. Therefore, the modification of the framework of genetic programming is minimum.

To evaluate of the effectiveness of learning by modified decision tree and rule expression using the modified function nodes and the automatic function definitions. The modified genetic programming is applied to the decision tree construction problem from the evaluation data of UCI Machine Learning Repository, and we compare the results by each function node definitions.

The purpose of the research is in the study of the decision tree with high accuracy with the tree with small size by using two or more logical functions properly.

5.2 Genetic Programming

Genetic programming (GP) is a learning method based on the natural theory of evolution, and the flow of the algorithm is similar to genetic algorithm (GA). The difference between GP and GA is that GP has extended its chromosome to allow structural expression using function nodes and terminal nodes [Koza 1992a, Koza 1994b]. Using function nodes and terminal nodes, GP can search hierarchical computer programs by undergoing adaptation.

GP can be used for a search problem if its problem can be expressed as computer programs. GP's search space depend on function nodes and terminal nodes.

In using GP to a problem, GP has five major preparatory steps. The five steps are followings.

1. the set of terminals
2. the set of primitive functions
3. the fitness measure
4. the parameters for controlling the run
5. the method for designating a result and criterion for terminating a run

The decision tree construction with GP is given by the following procedures.

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.
2. The fitness value, which relates to the problem solving ability, for each individual of the GP population is calculated.
3. The next generation is generated by genetic operations.
   (a) The individual is copied by fitness value (reproduction).
   (b) A new individual is generated by intersection (crossover).
   (c) A new individual is generated by random change (mutation).

4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

In this chapter, the tree structure is used to express the decision tree. Therefore, we express the decision tree by using each attribute value and the class name as the terminal node and the condition sentence as the function node. (See example in figure 5.1) We defined some expressions for decision trees. In figure 5.1, decision tree is expressed like LISP-code. "RPB" is defined as main GP tree. Both "ADF0" and "ADF1" are defined as each ADF tree. "IFLTE", "IFEQ" are function nodes. These functions requires four arguments, $arg1, arg2, arg3 and arg4$. The definitions of them are followings.

(IFLTE $arg1, arg2, arg3, arg4$**)** if $arg1$ is less than or equal to ($\leq$) $arg2$ then evaluate $arg3$, else then evaluate $arg4$

(IFEQ $arg1, arg2, arg3, arg4$**)** if $arg1$ is equal to(=) $arg2$ then evaluate $arg3$, else then evaluate $arg4$.

A, B, C and D express the attribute values from database. "T" and "F" express attribute value, and "N" and "P" express class name.

Using these expressions, GP can use database attributes more easily. Moreover, GP can apply continuous attributes. However, in general, database attributes are a lot of numbers, so GP's learning speed may become more slowly. The definition for continuous attributes may be cause of slow learning speed. It has trade-off relation between learning speed and expression compatibility.

Ordinarily, there is no method of adequately controlling the growth of the tree, because GP does not evaluate the size of the tree. Therefore, during the search process the tree may become overly deep and complex, or may settle to a too simple tree. There has been research on methods to have the program define functions itself for efficient use. One of the approaches is automatic function definition (or Automatically Defined Function: ADF), and this is achieved by adding the gene expression for the function definition to normal GP [Koza 1994a]. By implementing ADF, a more compact program can be produced, and the number of generation cycles can be reduced. More than one ADF can be defined in one individual.

In addition, the growth of the tree is controlled by evaluating the size of the tree. For example, an approach based on minimum description length (MDL) has been proposed concerning the evaluation of the size of the tree. In this chapter, we used the classification success ratio ($f_{hits(n)}$) and the number of composed nodes ($f_{nodes(n)}$), to define the fitness of the individual ($f_{fitness(n)}$).

$$f_{fitness(n)} = \alpha f_{hits(n)} + (1 - \alpha) f_{nodes(n)}$$

α is weight defined by ($0 \leqq \alpha \leqq 1$).

Here, two things are expected. One is that accuracy of the decision tree is raised while avoiding over-training in GP. And the other is that the decision tree generated can be made comparatively compact. The fitness value and rule size of the best individual is influenced by the weighting of success rate or node size. We set weight α by pre-experiment.

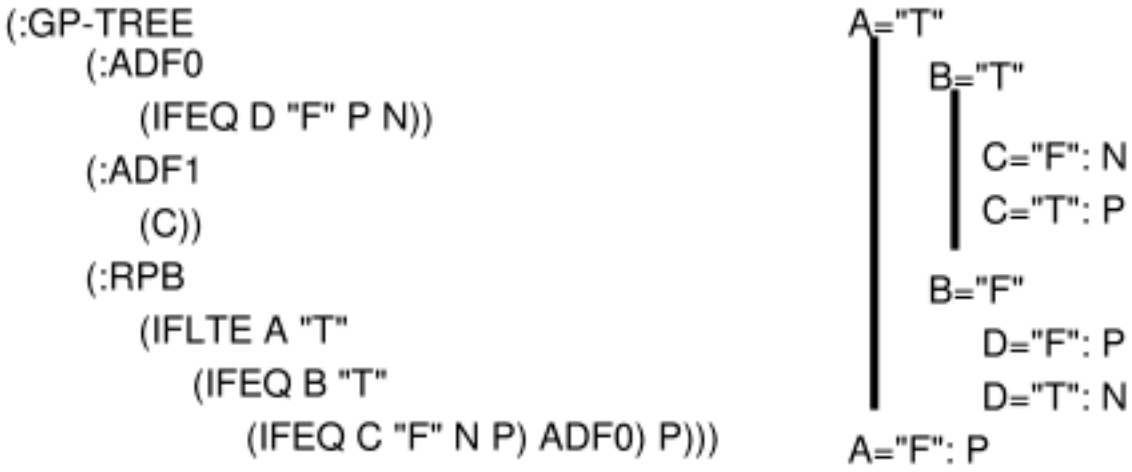


Figure 5.1: Expression of GP's Chromosome

5.3 Decision Tree and Expression of Rule by GP used AND, OR and NOT

When the decision tree is expressed by GP, if-else form can be used as a function node. Its idea from the technique by which the rule is extracted from the decision tree. [Quinlan 1995] (A, C and D are defined as arguments of function.)

(IF A C D) if (A) then C else D.

It is also possible to be used the following expanded definition which can compare the attributes from the database. [Niimi 1999b, Niimi 2000a] (A, B, C and D are defined as arguments of function.)

(IFEQ A B C D) if (A = B) then C else D.

In addition, the following operators can be used for rule expressions.

- AND
- OR
- NOT

In a general association rule, the rule is expressed as the conditions are connected by AND. [Kitsuregawa 1997, Terabe 2000] If if-else form is used for rule expression, it is difficult to treat a part which is not defined by association rule. Therefore, it is thought that it is difficult to learn the association rule with GP.

Morecver, it is necessary to have the same partial structure many times to express OR part by using if-else form in the decision tree.

On the other hand, AND part can be expressed by simply extending the route of the decision tree. It is thought that in the decision tree expression with if-else form, an increase in the size of the decision tree by the part expressing OP part is easier to occur than an increase in the size of the decision tree by the part expressing AND part.

Some GP which has already used a logical function is proposed. In common GP with boolean operations, GP is applied boolean function synthesis problem. On these implementations of AND, OR, NOT, the functions can return True or False. But its definition is not useful for data mining and knowledge discovery. To apply data mining and knowledge discovery, the function nodes of our proposed GP are defined as logical functions. Therefore, each function node is defined as extended logical functions which can return multi values. In our proposed method, return values of function nodes are almost databases' classname. Then, we extended these functions and defined functions.

In this chapter, to make the rule which contains OR part and NOT part easy to express by GP, the rule expression by AND, OR and NOT is defined as the following function nodes. And an outline of our proposed method is shown in figure 5.2.

(AND A B C D) if (A and B) then C else D.

(OR A B C D) if (A or B) then C else D.

(NOT A) if (A is True) then False else True.

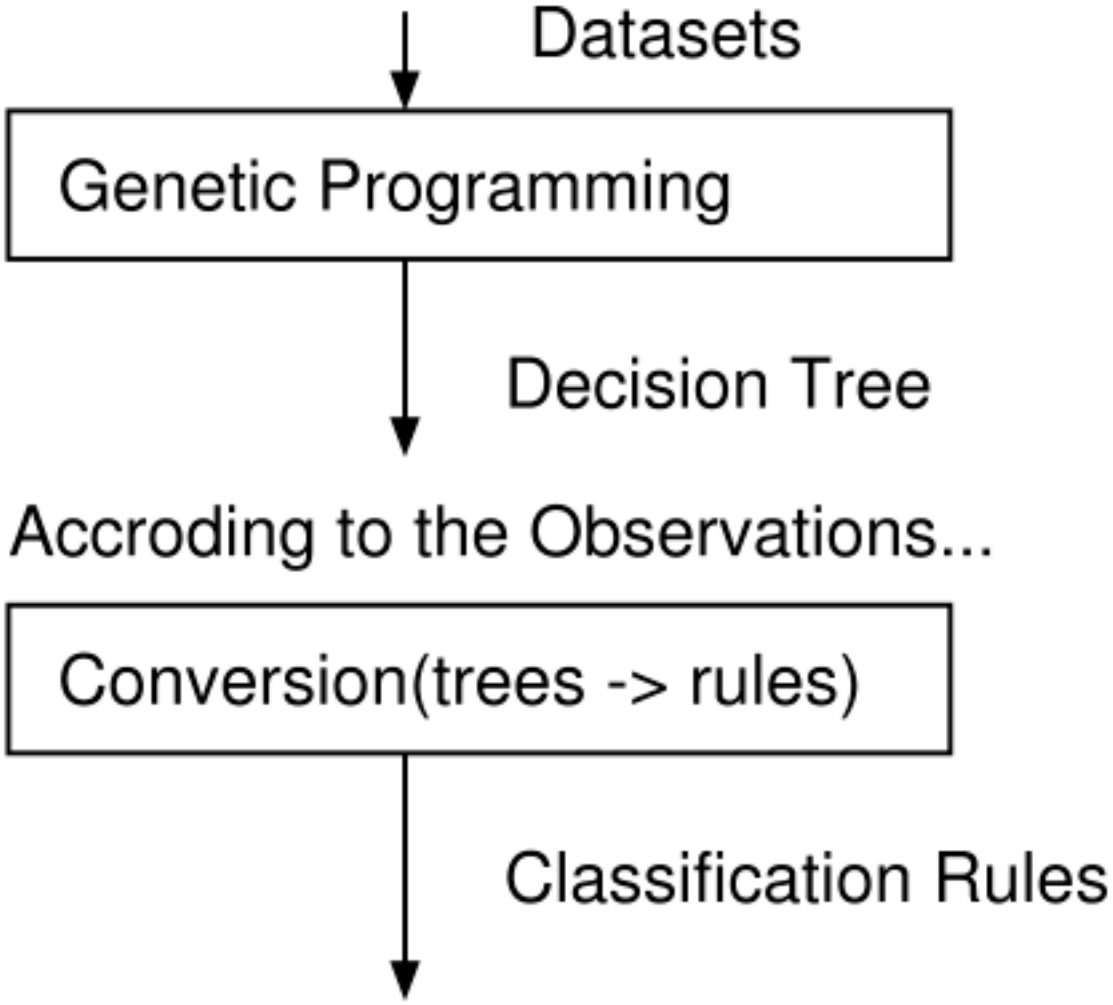


Figure 5.2: Flowchart of Approach of Proposed Combined Learning

In GP, it is comparatively easy to mount the expression method of various rules.

Mounting such as if-else, AND and OR can be done only by modification of the definition of the function nodes. Because it is only to modify the definition of the function nodes, the framework of GP need not be modified. Therefore, there is a possibility that the fitness value function and other parameters need not be modified.

By this modification, the modification is only definition of the function nodes, the definition of the fitness value function and the parameters need not to modify. Because the rule which contains AND and OR is expressed easily in this definition than using only if-else form, the reduction of the size of the constructed decision tree is expected. However, because the defined function node increases, and an increase in the combination happens, the improvement may not be so expected concerning the learning speed.

Only as for NOT, the definition is complicated. The purpose of it is to have to define the boolean values such as True and False.

The definition of AND and OR are understood by using only if-else form. (See also in figure 5.3.)

(AND A B C D) if(A and B)then C else D

→ (if A then(if B then C else D)else D)

(OR A B C D) if(A or B)then C else D

→ (if A then C else(if B then C else D))

In this case, the number of used nodes increases, and the depth of the decision tree become deeply. In addition, one individual must have plural similar parts for expressing OR part.

NOT can be expressed by using the part of else of if-else. But an expression of (NOT conditional expression) is easy to understand as the decision tree than an expression of if else (conditional expression). The easy understanding comes to interpret individuals easily to conversion into the rules as decision trees.

The decision tree construction which uses these logic operators is not so general. But, in the learning by the proposed GP, using these logical operators is possible by only modification of the definition of the function nodes, it is easy to use proposed GP.

If NAND and NOR use for rule expression, AND, OR and NOT can be theoretically expressed by only NAND or NOR. As a result, the definition of the function nodes in GP can be decreased such as used only if-else form. Therefore, the speed-up of learning is expected. However, GP learning might not good because an effective usage becomes difficult in NAND and NOR like if-else form.

In GP, it can save the meaningless rule and the unuseless rule, that the defined function nodes can be used comparatively freely in the GP individual. The combination of a lot of easy function nodes may be more effective than the combination of small number of complex function nodes.

Moreover, NAND and NOR also have the fault that it is difficult to understand the meaning on the rule. The rule described with NAND and NOR is not generally and can not be intuitively understood.

A general other logical operator, such as XOR (exclusive-OR), has the possibility to make many rules with a difficult interpretation if the operator is not effectively used.

If the problem contains many rules which can be expressed by only AND function, it can be analyzed by only if-else form. However, If the problem contains many OR rules,

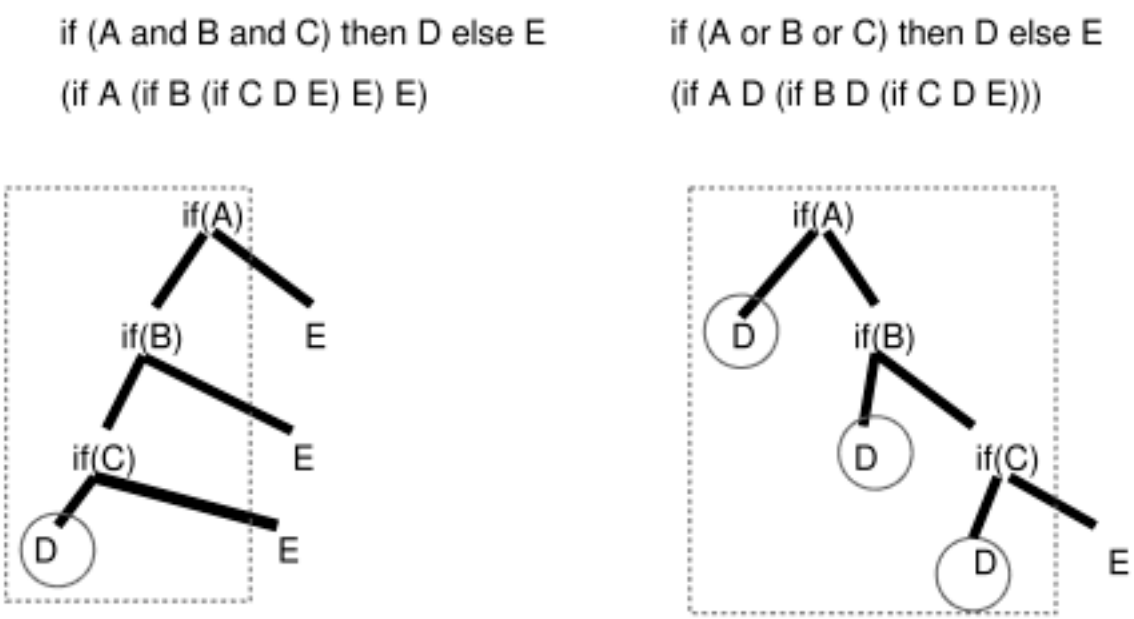


Figure 5.3: "if"Expression of AND, OR

it might be difficult in the analysis of the expression only with if-else form. It is thought that the proposed modification is effective for such a problem.

It has been enumerated that the Building block hypothesis does not consist easily of GP which uses the boolean function as a problem. It is thought that the combination of nodes can be prevented being destroyed by expressing the combination of easy nodes by using a more complex node. In the proposal technique, as for the inclusion of a similar expression, we thinks whether it is possible to evade by richly expressing nodes so that the combination of nodes is not destroyed as much as possible.

The expression richness often works profitably as a system when knowledge is expressed with man's doing the knowledge representation and the knowledge base system. However, there is a possibility to connect with the increase of the search space because of increasing of the combination of nodes when there are two or more methods of expression GP for one event, and increasing the amount of the calculation. Because the expression of AND, OR, and NOT has already been contained as a decision tree, and the function node of AND and OR contains the NOT expression in the proposal node, each expression can be substituted by the De Morgan's law. Therefore, it is thought that the repetition of the expression is contained.

The result like the size reduction and the accuracy improvement, etc. of the decision tree was seen from the experiment result by each function node's being effectively used in the proposal technique. It is thought that having contained the part where the size and the accuracy of the tree are evaluated to the adjustment degree function led to effective use for the function node.

The classification rules converted from the decision tree into the rule to evaluate each rule included in the decision tree in the experiment result and the classification rule was generated. It was small number of rule groups, and the one that the accuracy of each rule was high, and the rule availability is high was evaluated to the rule generated from the decision tree as a rule group better.

5.4 Application to Decision Tree Construction Problem from Database

To verify the validity of the proposed method, we applied it to the house-votes data from UCI Machine Learning Repository [Blake 1998]. From here on all occurrence of GP uses Automatically Defined Function Genetic Programming (ADF-GP) [Koza 1994a] including the proposed method. We compared the results of the proposed method against GP.

For evaluation, we used house-votes data from UCI Machine Learning Repository [Blake 1998]. We compared the results of the proposed method with GP. The evaluation data contains 16 attributes and 2 classes. The attributes are for example "handicapped-infants" and "water-project-cost-sharing " etc. They are expressed by 3 values: "y ", "n", and "?". And the 2 classes are "democrat" and "republican ". 50 cases out of the total 435 data of house-votes were used for training data.

We extracted the association rule from the database by the Apriori algorithm. We applied the Apriori algorithm to a data set excluding data with the "?" value, because "?" value means "others".

To verify the validity of the proposed modification, we apply the modified GP to an evaluation experiment. For evaluation data, we use house-votes data from UCI Machine

Learning Repository. [Blake 1998] By using this evaluation database, we can compare the results of the proposed modification with normal GP. The evaluation data contains 16 attributes and 2 classes. The attributes are for example "handicapped-infants" and "water-project-cost-sharing " etc. They are expressed by 3 values: "y ", "n", and "?". And the 2 classes are "democrat" and "republican ". 50 cases out of the total 435 data of house-votes are used for training data. The overview of its dataset is shown in table 5.1.

The decision tree was generated from the learning data with GP. The parameter of modified GP is same one which is tuned for if-else experiment. The following GP parameters were used. (Refer to table 5.2.) (The result is a table 5.3.) In the table, an unused ADF definition part is excluded concerning the size of the individual and the depth of the tree. The best decision tree of each method is shown by figure 5.4, figure 5.5. The error distribution of each method is shown by table 5.4, table 5.5.

The fitness function contains the part where the size of the tree is evaluated and the part where accuracy is evaluated. Evaluating the description length and the size at the same time like the MDL standard becomes possible.

The classification rules converted from the decision tree into the rule to evaluate each rule included in the decision tree in the experiment result and the classification rule was generated. It was small number of rule groups, and the one that the accuracy of each rule was high, and the rule availability is high was evaluated to the rule generated from the decision tree as a rule group better.

The database using for experiment is small-scale example of data mining. In the proposal technique, the study time proportional to the number of data can be expected of the data base with a lot of numbers of data. However, it is thought that the amount of the calculation of the proposal technique increases explosively because the combination of nodes increases for the data base with a lot of numbers of attributes. When the proposal technique is applied to the data base with a lot of numbers of attributes or more, it is necessary to decrease the number of attributes used by the preprocessing with GP.

Learning the rule with AND and OR was able to generate the rule with high accuracy from the AND function.

An increase in the combination occurs because the function node defined when both AND and OR are used increases. Therefore, it seems that learning slows, and the number of generations until the best individual acquiring has become long.

The size and the depth of the decision tree were able to be improved from the one that if-else was used.

By using NOT function, the accuracy of the decision tree was improved.

The expression richness often works profitably as a system when knowledge is expressed with man's doing the knowledge representation and the knowledge base system. However, there is a possibility to connect with the increase of the search space because of increasing of the combination of nodes when there are two or more methods of expression GP for one event, and increasing the amount of the calculation. Because the expression of AND, OR, and NOT has already been contained as a decision tree, and the function node of AND and OR contains the NOT expression in the proposal node, each expression can be substituted by the De Morgan's law. Therefore, it is thought that the repetition of the expression is contained.

The result like the size reduction and the accuracy improvement, etc. of the decision tree was seen from the experiment result by each function node's being effectively used in the proposal technique. It is thought that having contained the part where the size and

Table 5.1: House-votes Dataset.

Class Name	2 (democrat, republican)
handicapped-infants	2 (y,n)
water-project-cost-sharing	2 (y,n)
adoption-of-the-budget-resolution	2 (y,n)
physician-fee-freeze	2 (y,n)
el-salvador-aid	2 (y,n)
religious-groups-in-schools	2 (y,n)
anti-satellite-test-ban	2 (y,n)
aid-to-nicaraguan-contras	2 (y,n)
mx-missile	2 (y,n)
immigration	2 (y,n)
synfuels-corporation-cutback	2 (y,n)
education-spending	2 (y,n)
superfund-right-to-sue	2 (y,n)
crime	2 (y,n)
duty-free-exports	2 (y,n)
export-administration-act-south-africa	2 (y,n)
Missing Attribute Values	Denoted by "?"

Number of Instances: 435 (267 democrats, 168 republicans)
Number of Attributes: 16 + Class Name = 17 (All Boolean Valued)

Table 5.2: Parameters of GP

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node(1)	IFEQ, ADF0, ADF1
Function node(2)	AND, ADF0, ADF1
Function node(3)	AND, OR, ADF0, ADF1
Terminal node	Attribute value of database (16 attributes), y, n, ?, democrat, republican
Number of training data	democrat:31, republican:19
Number of test data	435

IFEQ: if equal to (=)
(AND A B C D): if (A and B) then C else D.
(OR A B C D): if (A or B) then C else D.
ADF0, ADF1: the function definition chromosome expanded by ADF
y, n, ?: yes(y), no(n), others(?)

Table 5.3: Experiment Best Individuals Result by Each Technique.

	training (%)	all data (%)	nodes	depths	generations
AND	96.0	65.5	10	1	29
AND+OR	96.0	92.0	5	1	1730
AND+NOT	98.0	92.9	8	2	1331
AND+OR+NOT	98.0	94.3	7	2	122
if-else	100.0	86.0	11	3	586

No used ADF parts are removed from size and depth.

```
RPB:
    (AND physician-fee-freeze ADF0
         adoption-of-the-budget-resolution democrat-y)
ADF0:
    (AND duty-free-exports democrat-y
         physician-fee-freeze el-salvador-aid)
ADF1:
    anti-satellite-test-ban
```

Figure 5.4: Decision Tree - AND only

```
RPB:
    (OR physician-fee-freeze adoption-of-the-budgest-resolution
        adoption-of-the-budget-resolution democrat-y)
ADF0:
    (AND religious-group-in-schools crime
        export-administration-act-south-africa
        adoption-of-the-budget-resolution)
ADF1:
    mx-missile
```

Figure 5.5: Decision Tree - AND + OR

Table 5.4: Error Distribution - AND only.

(a)	(b)	→ classified as
264	2	(a):class democrat
147	21	(b):class republican
total	hits	= 285

Table 5.5: Error Distribution - AND + OR.

(a)	(b)	→ classified as
260	6	(a):class democrat
27	140	(b):class republican
total	hits	= 400

the accuracy of the tree are evaluated to the adjustment degree function led to effective use for the function node.

5.5 Conclusive Discussion

In this chapter, the decision tree and the rule expression by genetic programming was proposed, and the expression which used AND, OR and NOT were mounted on genetic programming besides the rule expression of the if-else form. Moreover, the decision tree was constructed by using the house-votes data from UCI Machine Learning Repository to verify the effectiveness of the modified rule expressions, and we compared the results.

As a result, the size of the decision tree was able to be improved. Moreover, the improvement of accuracy was admitted in the one that AND and OR were used. By using NOT function, the accuracy of the decision tree was improved. The expanding rule expression is mounted by replacing the definition of the function node of genetic programming. Therefore, the modification of the framework of genetic programming is minimum. It can be said that the rule expression which uses AND, OR and NOT from this is effective in genetic programming.

Future works are scheduled to do the evaluation which uses data for other verifications, to examine whether the rule expression by NAND, NOR and XOR etc. can be used or not, and to make a policy which rule expressions to be used.

Chapter 6

Extension of Genetic Programming for Knowledge Discovery

In this chapter, we discuss about the improvement methods of genetic programming. Because of some problems, genetic programming is difficult to use for data mining. We propose three methods which improve these problems. The first methods improves the problem that genetic programming is difficult to construct its system. The second methods improves the problem that genetic programming is difficult to treat with continuous value attributes. The third method improves the problem that genetic programming is difficult to use with a lot of nodes. Other proposed improvement method is an extended genetic programming approach for initial condition problem. [Yoshiie 2000]

This chapter is organized as the following:

In section 6.1, we propose object-oriented combined method. In this method, genetic programming and the decision tree construction method are treated respectively as an object, combined to construct learning system. In section 6.2, we propose redundant patterns pruning operation and adjustment operation of continuous value attribute. In genetic programming with these operations, genetic programming is able to learn the continuous value attributes by these operation as genetic operation. In section 6.3 , we propose extended crossover to maintain variety. The maintenance of variety in the genetic programming's population is expected by using not only fitness value but also the difference of individual for the index of crossover.

6.1 Object Oriented Approach to Combined Learning of Decision Tree and ADF GP

There are many learning methods for classification systems. Genetic programming (one of the methods) can change trees dynamically, but its learning speed is slow. Decision tree methods using C4.5 construct trees quickly, but the network may not classify correctly when the training data contains noise. For such problems, we proposed an object oriented approach, and a learning method that combines decision tree making method (C4.5) and genetic programming. To verify the validity of the proposed method, we developed two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for the meningoencephalitis. We compared the results of proposed method with prior ones.

Various techniques are proposed for the construction of the inference system using

classification learning. In general, the learning speed of a system using a genetic programming is slow. Moreover, both problem background knowledge and design skill are demanded of the system designer. However, a learning system which can acquire higher-order knowledge by adjusting to the environment can be constructed, because the structure is treated at the same time.

On the other hand, a learning system which uses the decision tree can be trained in a short time compared to other techniques. An effective network structure is constructed by the classification model is obtained by decision tree. But, there is a problem with deteriorated classification accuracy when the training data contains noise.

Each technique has advantages and disadvantages like this. In this section, we propose an object oriented method to construct a classification system trained by combining various learning methods treated as objects. It is expected that learning will occur while mutually making up for the advantage and the disadvantage of each technique.

To verify the validity of the effectiveness of the proposed learning method, we test the two learning methods: the decision tree construction method (C4.5), and genetic programming with automatic function definition (ADF). This is applied to two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for meningoencephalitis. We compared the results of the proposed method with prior approaches using only one learning method.

The effectiveness of combining techniques has been verified by the study of neural network training using decision tree construction method and back-propagation learning method [Banerjee 1997], and by the study of neural network training using decision tree construction method and genetic programming [Matsumoto 1998].

There is a method of inductively constructing a model by examining the recorded classification data and generalizing a specific example. The classification learning by decision tree can achieve a certain classification ability in a comparatively short time. C4.5 is one of decision tree construction methods, and it classifies data based on the gain criterion which selects a test to maximize expected information gain. As a result, important attributes can be collected at the root of the decision tree. Moreover, the algorithm contains branch pruning by estimating error rates to prevent the construction of excessively classified decision trees.

The decision tree construction with C4.5 follows the following procedures [Quinlan 1995].

1. Construction of initial decision tree.

2. Branch pruning of constructed decision tree.

In the decision tree construction method by C4.5, the decision tree is constructed with the recurrent division of the training data. Therefore, there is a possibility of constructing a different decision tree when the number of training data is modified. (Refer to table 6.1.) In general, using a large number of training data tends to construct a more complex decision tree.

Genetic Programming(GP) is a learning method based on the theory of natural evolution, and the flow of the algorithm is similar to that of Genetic Algorithm(GA). The difference with GA is that in GP the chromosome expression has been extended to express

Table 6.1: Example of C4.5 and Number of Training Data

Number of Training Data	Tree Size	Wrong (rate)	
		Training	Test
140.0	7.0	1.0(0.7%)	1.0(0.7%)
70.0	5.0	1.5(2.2%)	10.0(7.2%)
46.7	4.3	1.0(2.1%)	11.0(7.8%)
35.0	4.0	0.5(1.5%)	18.2(13.0%)
28.0	4.6	0.6(2.2%)	13.0(9.3%)
23.3	3.7	0.5(2.1%)	23.0(16.4%)
20.0	3.3	0.6(2.9%)	17.0(12.1%)
17.5	4.0	0.4(2.1%)	22.0(15.7%)
15.6	3.4	0.4(2.9%)	31.8(22.7%)
14.0	3.2	0.6(4.3%)	27.0(19.3%)

structure using function nodes and terminal nodes. In this section, the tree structure was used to express the decision tree.

The decision tree construction with GP follows the following procedures.

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.

2. The fitness value, which relates to the problem solving ability, for each individual of the GP population is calculated.

3. The next generation is generated by genetic operations.

   (a) The individual is copied by fitness value (reproduction).

   (b) A new individual is generated by intersection (crossover).

   (c) A new individual is generated by random change (mutation).

4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

Ordinarily, there is no method of adequately controlling the growth of the tree, because GP does not evaluate the size of the tree. Therefore, during the course of the search the tree may become overly deep and complex, or may settle to a too simple tree . There has been research on methods to have the program define functions itself for efficient use. One of the approaches is automatic function definition (or Automatically Defined Function:ADF), and this is achieved by adding the gene expression for the function definition to normal GP [Koza 1994a]. By implementing ADF, a more compact program can be produced, and the number of generation cycles can be reduced. More than one ADF can be defined in one individual.

In addition, the growth of the tree is controlled by evaluating the size of the tree. For example, an approach based on minimum description length (MDL) has been proposed concerning the evaluation of the size of the tree. In this section, we used the classification success ratio and the number of composed nodes, to define the fitness of the individual.

$$f_{fitness(n)} = \alpha f_{hits(n)} + (1 - \alpha) f_{nodes(n)}$$

α is weight defined by $(0 \leqq \alpha \leqq 1)$.

Here, two things are expected. One is that accuracy of the decision tree is raised while avoiding over-training in GP. And the other is that the decision tree generated can be made comparatively compact. The inference accuracy and rule size of the best individual is influenced by the weighting of success rate or node size.

In an object oriented design, a program is built from independent information processing units, called objects, and information is processed by exchanging messages between objects. The object receives input information, performs a series of processes, and sends output information. In this case, the object's own internal information is hidden. Therefore, the external data flow can be designed independently from the internal processing.

The classification learning by the decision tree can be trained in a short time, but when the noise is contained in the training data, the classification ability is rapidly deteriorated. On the other hand, a high classification ability is obtained by GP through training structural information, but more learning time is needed as the degree of learning

freedom increases.

For such problems, we propose an object oriented approach to the learning method that combines the decision tree method (C4.5) and genetic programming. The proposed method consists of the following three steps:

1. First, using C4.5, construct appropriate decision trees.

2. Next, generate the genetic programming population which includes initial individuals converted from the decision trees.

3. Train the genetic program to construct the classification system.

The proposed method simplifies the construction of classification systems. It is observed that the discursive accuracy of classification becomes highly improved by the emergent property of interaction between two combined methods.

When the decision tree is taken into the initial population of GP, it is necessary that variety in the initial population is not upheld [Niimi 1999a]. For that purpose, we created decision trees by C4.5 using different number of training data, to ensure a variety of patterns taken into the initial population.

To verify the validity of the proposed method, we developed two different medical diagnostic systems. One is a medical diagnostic system for the occurrence of hypertension, the other is for meningoencephalitis. We compared the results of the proposed method with prior ones.

The database used for the occurrence of hypertension contains fifteen input terms and one output term. There are two kinds of Intermediate assumptions between the input terms and the output term [Ichimura 1997]. Among the input terms, ten terms are categorized into a biochemical test related to the measurement of blood pressure for past five years, and the rest are "Sex", "Age", "Obesity Index", "γ-GTP", and "Volume of Consuming Alcohol". One output term represents whether the patient has had an attack of hypertension for the input record. (Example data is shown in Table 6.2.) The database has 1024 patient records. In this section, we selected 100 occurrence data and 100 non-occurence data by random sampling, and this was used as the training data.

The parameters for GP used the following. (Refer to Table 6.3.) In this experiment, only a small percentage of GP individuals could be used as decision trees due to the large number of node types and large freedom for tree construction. Moreover, most of the input data have continuous value attributes. This seems to have caused the decrease in the inference accuracy of the GP search close to that of a random search. It was confirmed that the decision tree taken in as an initial individual was succeeded to the best individual, and it can be concluded that this influenced the improvement of the learning efficiency. (The results shown in Table 6.4.)

The meningoencephalitis database is a medical treatment database concerning the discrimination diagnosis of meningoencephalitis. It consists of 140 patients. The database is described by 34 attribute about the past illness history, physical examinations, laboratory examinations, diagnosis, therapies, clinical courses, final status, and risk factors. The two classifications were bacillus and virus meningitis [Tsumoto 1999b]. In this section, 32 attributes regarding sex, ages, etc. were used as well.

Table 6.2: Example Data (The Ocurence of Hypertension)

Occur	Sex	Age	Obesity Index	γGTP	Consuming Alcohol	Systolic Blood Pressure (one per year)					Diastolic Blood Pressure (one per year)				
						1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
+	m	48	25.86	54	13	120	122	137	116	153	76	73	83	80	93
+	f	45	25.56	13	0	132	141	120	151	139	84	74	82	85	79
-	f	47	21.05	10	0	120	113	136	111	126	70	58	78	64	69
-	m	43	26.61	17	0	120	136	120	130	136	66	80	80	80	83
+	m	49	28.73	40	0	138	140	134	138	140	78	80	88	88	90
+	m	50	17.70	18	0	120	146	110	124	110	80	93	80	86	75
+	m	49	24.50	83	26	138	135	128	138	136	74	86	74	90	85
-	f	52	27.60	20	0	120	129	124	128	125	80	61	74	80	67
-	f	51	21.10	17	0	120	126	124	126	112	80	73	74	72	63
-	m	42	20.80	34	6.5	114	110	128	130	125	76	66	82	66	76

Table 6.3: Parameters of GP

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node	IFLTH, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (15 attributes), P, N, R
Number of training data	P(100), N(100)
Number of test data	1024

IFLTH,IFEQ: if less than (<),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
P, N: ocurence (P), no-occurence (N)

Table 6.4: Experiment Result (Reasoning Precision) by Each Technique.

	training data (%)	all data (%)	nodes (%)	generations
C4.5	98.5	77.1	29	—
ADF-GP	75.0	66.9	148	14328
C4.5 +ADF-GP	96.0	81.4	17	41

The parameters for GP used the following. (Refer to Table 6.5.) For this experiment, all approaches achieved high accuracy, the medical database had removed noise well, and many attributes have discrete values. For the construction of decision tree by ADF-GP only, the inference accuracy did not increase quickly, but for construction of decision tree by combined ADF-GP and C4.5, the decision tree reached high accuracy comparatively quickly. It was confirmed that the decision tree taken in as an initial individual was succeeded to to the best individual, and influenced the improvement of the learning efficiency. (The results shown in Table 6.6.)

In this section, we proposed an inference system construction method based on an object oriented approach, combining two or more learning methods. We compared the results of three methods (using C4.5 only, using ADF-GP only, and using combined C4.5 and ADF-GP using the proposed method).

As a result, an improvement of learning efficiency was seen. It can be concluded that the proposed technique is an effective method for the improvement of learning efficiency of an inference system.

6.2 Extended GP using Reinforcement Learning Operation

Genetic programming(GP) usually has a wide search space and a high flexibility, so GP may search for a global optimum solution. But GP has two problems. One is slow learning speed and huge number of generations spending. The other is difficulty to operate continuous numbers. GP searches many tree patterns including useless node trees and meaningless expression trees.

In general, GP has three genetic operators (mutation, crossover and reproduction). We propose an extended GP learning method including two new genetic operators, pruning (pruning redundant patterns) and fitting (fitting random continuous nodes). These operators have a reinforcement learning effect, and improve the efficiency of GP's search.

To verify the validity of the proposed method, we developed a medical diagnostic system for the occurrence of hypertension. We compared the results of proposed method with prior ones.

Genetic programming(GP) usually has a wide search space and a high flexibility. So, GP may search for global optimum solution. But GP has two problems. One is slow learning speed and huge number of generations spending. The other is difficulty to operate continuous numbers. GP searches many tree patterns including useless node trees and meaningless expression trees.

For the construction technique of an inference system by classification learning, we propose two new genetic programming (GP) techniques. The pruning operation of an invalid sub-tree structure increases the search efficiency. The fitting operation GP can adjust continuous value nodes using local learning. This technique improves the efficiency when treating continuous value attributes.

Table 6.5: Parameters of GP

GP population	500
Reproduction probability	0.1
Intersection probability	0.8
Mutation probability	0.1
Selection method	Tournament method
Function node	IFLTH, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (32 attributes), VIRUS, BACTERIA, R
Number of training data	VIRUS(49), BACTERIA(21)
Number of test data	140

IFLTH,IFEQ: if less than (<),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
VIRUS,BACTERIA: Virus meningitis and bacillus meningitis

Table 6.6: Comparison of Generated Decision Trees (Number of Nodes and Inference Accuracy) by Each Technique

	training data (%)	all data (%)	nodes (%)	generations
C4.5	98.6	94.3	11	—
ADF-GP	88.6	82.9	21	7673
C4.5 +ADF-GP	95.7	97.1	11	686

By adding these two new techniques, it is expected that the classification learning system, which contains continuous value attributes, can be more efficiently constructed compared to normal GP.

To verify the validity of the proposed techniques, we developed a medical diagnostic system for the occurrence of hypertension. We compared the results of the proposed method with normal GP.

Genetic programming(GP) is a learning method based on the natural theory of evolution, and the flow of the algorithm is similar to genetic algorithm(GA). The difference between GP and GA is that GP has extended its chromosome to allow structural expression using function nodes and terminal nodes [Koza 1992a].

Various topics about the examples of applying GP have already been researched by others. In this section, we aimed to construct a classification learning system with GP. We took up the decision tree as a classification learning system, and expressed the decision tree using tree structural individuals. (Refer to Figure 6.1.)

The decision tree construction with GP follows the following procedures.

1. An initial population is generated from a random grammar of the function nodes and the terminal nodes defined for each problem domain.
2. The fitness value, which relates to the problem solving ability, is calculated for each individual of the GP population.
3. The next generation is generated by genetic operations.
   (a) The individual is copied according to fitness value (reproduction).
   (b) A new individual is generated according to intersection (crossover).
   (c) A new individual is generated by random change (mutation).
4. If the termination condition is met, then the process exits. Otherwise, the process repeats from the calculation of fitness value in step 2.

In general, C4.5 is often used for decision tree learning. C4.5 is a high-speed algorithm, and it can be generate a high accuracy decision tree [Quinlan 1995]. On the other hand, the decision tree construction by GP is inferior to C4.5 by the calculation time. But, there is an advantage that various criterions can be used, and the decision tree by a higher-order knowledge representation can be constructed in learning by GP.

In this section, we will think about the individual expression as shown in Figure 1. The nodes which are used compared node "IFLTE", such as "135" and "139" in the figure, is an ephemeral random constant terminal. These nodes are defined as "constant nodes with a continuous value attribute". Moreover, the node, such as "Input1" and "Input2", is defined a terminal nodes which receive the input from a database.

One of the reasons why GP takes much time in calculation is that many types of nodes are defined in the decision tree construction. The discrete value attribute are treated as

```
(IFLTE Input3 139
       (IFLTE Input2
              (IFLTE Input4 135
                     (IFLTE Input5 139
                            (IFLTE Input1 85 139 Input3)
                      P) P) N P) P)

      Description of Function Node
      (IFLTE A B C D) : If (A < B), then C else D
```

Figure 6.1: Expression of GP's Chromosome

different nodes for each attribute. Moreover, in classification test of the continuous value attribute, the number of nodes is defined by the number of values which can be taken by random numbers, e.g. when the constant of the threshold of the divergence is given by random numbers. This causes an increase in the types of the node used.

When there are too many types defined, this may cause the GP learning speed to become very slow, or cause the GP to fail to reach the global optimum solution. This is caused by the explosive increase in combinations. However, useless and meaningless expressions might be included in the combination. For such problems, many techniques have been proposed. The ADF technique defines its own partial tree to make learning more efficient [Koza 1994a]. The MDL technique uses a fitness function defined by the size of the tree based on minimum description length (MDL) [Iba 1994]. The technique combined GP and C4.5 increase the accuracy of the initial population [Niimi 1999a].

In this section, we define redundant nodes as useless expression patterns and/or meaningless expression patterns. We propose the pruning of redundant patterns to improve learning speed. We also propose an adjustment operation to the node with continuous value attributes, based on a similar idea. For the pruning of redundant patterns, a technique combined with a fuzzy rules has been proposed [Maeda 1998].

Useless structures can be removed from the GP individual by redundant patterns pruning operation. Therefore, the reduction of tree structure and the reduction in the amount of calculation are expected. This operation is applied to sub-trees selected by random. It searches redundant node patterns, and replaces the node patterns found with an effective terminal node. This operation is based on a reinforcement learning approach. Because the redundant node patterns depend on the problem, it is necessary to define them for each problem.

For this experiment, the GP tree structure was first converted into a node object array, and the pattern of the redundant node was checked using pattern matching. Examples of redundant node patterns are shown in table 6.7. By converting the tree structure into an array, it becomes possible to look for a redundant node pattern without being affected by the depth of the tree. Moreover, by converting the tree to an object array, that operation does not need to worry about the content of each node.

The redundant patterns pruning operation consists of following steps.

1. An arbitrary node is chosen from the target individual of GP. The tree structure of the sub-tree below the node is converted into a node object array.

2. The redundant node pattern is checked by pattern matching against the converted array.

   If a corresponding pattern is found, the node pattern is replaced with a terminal node.

3. The checking pattern is changed and process 2. is repeated, until all patterns are checked.

Table 6.7: Examples of Redundant Node Patterns

(**IFLTE** A A B C)	Replaced by C
(**IFLTE** R1 R2 B C)	**If** (R1 $<$ R2), then replaced by B else by C
(**IFLTE** A1 A2 C C)	Replaced by C
(**IFEQ** A A B C)	Replaced by B
(**IFEQ** R1 R2 B C)	**If** (R1 $=$ R2), then replaced by B else by C

Description of Function Node
(IFLTE A B C D) : If (A $<$ B), then C else D
(IFEQ A B C D) : If (A $=$ B), then C else D

4. The processed pattern for the array is converted back into a tree structure, and the converted tree structure is replaced with the original sub-tree structure.

Let us assume the following algorithm. For a given subtree, create all possible combinations by replacing the nodes with the set of all defined nodes, without changing the original tree structure of the subtree. Select the combination which gives the highest fitness value. Using this algorithm, it is possible to look for the local optimum solution for a given structure, and a strong learning effect is expected. However, there is a possibility that the amount of the calculation increases. In this section, we apply this idea to the training of GP containing many continuous value attributes.

In the decision tree construction, the terminal node is occasionally defined the outputs of the continuous value attribute from training data. At the method that the constant node compared with such a continuous value attribute is defined by random, there is a possibility not being selected for a quite effective constant value because the range of a continuous value attribute is wide. Then, we proposed the algorithm of the replace constant value node to rise adjustment degree by the training data.

At the replaced processing, the constant value node is processed replacing after the tree structure of the individual of GP is converted into the node object array. Examples of continuous value adjustment process are shown in table 6.8. The method is similar method of pruning redundant patterns, and the advantage of this process is same thing which listed at redundant patterns pruning operation. In addition, because the continuous node patterns depend on the problem, it is necessary to define them by each problem.

The adjustment operation to the continuous value attribute consists of the following steps.

1. An arbitrary node is chosen from the target individual of GP. The tree structure of a sub-tree below the node is converted into the node object array.

2. The pattern of the continuous value node pattern is checked pattern matching to the converted array. The continuous value node pattern is a pattern to compare the continuous value attribute and the constant node mutually.

   (a) If a corresponding pattern is found, the node pattern is replaced with the node pattern which contains the constant by which the new node pattern is generated.

3. Until all patterns are checked, the checking pattern is changed and process 2. is repeated.

4. The processed pattern for the array is converted into the tree structure, and the converted tree structure is replaced with an original sub-tree structure.

When the adjustment is operated, the new constant node is generated so that the adjustment degree may rise. At this time, generating the individual with a high degree of the more adjustment becomes possible by devising the constant generation algorithm. The following two algorithm are taken up as a constant generation algorithm.

Table 6.8: Examples of Continuous Value Adjustment Processing

(IFLTE A R1 B C)	If ($A_{min} > R1$ or $A_{max} < R1$), then re-define node by random.
(IFLTE R1 A B C)	Ditto
(IFEQ A R1 B C)	Ditto
(IFEQ R1 A B C)	Ditto

Description of Function Node
(IFLTE A B C D) : If (A < B), then C else D
(IFEQ A B C D) : If (A = B), then C else D

Strongly Continuous Value Nodes Fitting Function: To become the best accuracy, the value of a continuous node is local optimum learned by using all the training data. This operation makes surely its accuracy rise. But, the continuous value node and its tree structure will have strongly relationship, and it is possible not to work other genetic operations well.

Weakly Continuous Value Nodes Fitting Function: The continuous node from random numbers are taken so that the range of the comparison object and the numerical value may come in succession. It seems that the freedom degree to other genetic operations is preserved considerably high though accuracy does not necessarily rise. In addition, to preserve the freedom degree highly, it is possible to use that the generating operation can occasionally take outside the range in the distribution of random numbers (regular distribution which center on compared ranges, etc.).

We propose the redundant pattern pruning operation and the adjustment operation to the continuous value attribute are built in GP. The adjustment operation was examined at section 6.2. Here, it is consider the various maintenance etc. when pruning operation is built in further. Therefore, the following two approaches are devised by how to build operations in GP.

1. The operations are built in as an genetic operation, and process the method to a part of individual.

2. The operations are built in as the processing between generations, and process the method to all individuals simultaneously.

At the approach of 1, each processing is treated just like the mutation and crossover. Because of processing of pruning and adjustment operation, it is possible that a various maintenance may be lost and a potential effective schema may be destroyed. However, it is considered that these problems can be avoided by processing as a probabilistic operations like other genetic operations.

At the approach of 2, learning speed will be strongly advanced. Therefore, it is possible that learning causes an initial settling. However, because shortening the number of generations until settling becomes possible, it is possible that these operations apply in the field of the interactive evolute calculation etc.

To verify the validity of the proposed method, we developed a medical diagnostic systems for the occurrence of hypertension. We compare the results of proposed method with prior ones.

The occurrence of hypertension database is contained fifteen input terms and one output term. There are two kinds of Intermediate assumptions between the input terms and the output term[6]. Among the input terms, ten terms are categorized into an biochemical test related to the measurement of blood pressure for past five years, and the rest terms are "Sex", "Age", "Obesity Index", "γ-GTP", and "Volume of Consuming Alcohol". One output term represents whether the patient has an attack of hypertension for the input record. The database has 1024 patient records. In this section, we selected 100 occurrence data and 100 no-occurence data by random sampling, and this was used as the training data.

The parameter of GP used the follcwing. (Refer to Table 6.9.) Pruning redundant patterns operation and fitting random continuous value nodes operation were built in GP respectively as an genetic operation. Moreover, weakly continuous nodes fitting function was used.

The tendency which the adjustment degree usually improved at the learning initial stage was seen in the results of three methods (only redundant patterns pruning operation built in GP, only adjustment continuous value attribute operation built in GP, both two operation built in GP). Morecver, the number of generations until settling has decreased. (Refer to table 6.10.) As a result, it can be said that the proposed technique influenced the improvement of the learning efficiency. However, the difference between normal GP and proposed GP was not seen in the adjustment degree in the learning finality stage. The result is thought to be an occurrence because the parameter setting is defined experience in the experiment at this time. We will think that the learning efficiency influences by optimizing the parameter setting in the learning finality stage in the future.

In this section, we proposed the redundant node patterns operation to raise the search efficiency, and added to normal GP for the construction technique of inference system by the used GP classification learning. Moreover, we proposed the GP technique with adjustment continuous value nodes as local learning to improve to treat the continuous value attribute more easily. To verify the validity of the proposed techniques, we developed a medical diagnostic system, and compared the results of proposed methods and normal GP.

As a result, an improvement of learning efficiency was seen. It can be said that the pruning for redundant node improve the efficiency of GP's search, and the fitting for continuous number node range makes continuous node more effective. Thereafter, it can be concluded that the proposed technique is an effective method for the improvement of learning efficiency of an inference system.

6.3 Rule Discovery Technique using GP with Crossover to Maintain Variety

Many GP learning methods have been proposed to decrease node combinations in order to keep the node combinations from explosively increasing. We propose a technique using an opposite approach which tests a greater number of combinations in order to decrease the chances of the search being 'trapped' in a local optimum. In the proposed technique, hcw 'different' the individual structure is is used as an index in selecting individuals for genetic operations. Therefore, variety in the GP group is strongly maintained, and it is expected that GP learning is always done to a new combination.

In general, when there are a large number of node types defined, this may cause the Genetic Programming (GP) [Koza 1992a] learning speed to become very slcw, or cause the GP to fail to reach the global optimum solution. This is caused by the explosive increase in combinations. For this problem, We propose a technique using an opposite approach which tests a greater number of combinations in order to decrease the chances of the search being 'trapped' in a local optimum. In the proposed technique, how 'different' the

Table 6.9: Parameters of GP

GP population	500
Reproduction probability	0.1
Intersection probability	0.6
Mutation probability	0.1
Pruning redundant patterns probability	0.1
Fitting random continuous value nodes probability	0.1
Selection method	Tournament method
Function node	IFLTH, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (15 attributes), P, N, R
Number of training data	Occurrence (100) , No-occurence (100)
Number of test data	1024

IFLTH,IFEQ: if less than (<),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
P, N: ocurence (P), no-occurence (N)

Table 6.10: Experimental Result (Reasoning Precision) by Each Technique

	training data (%)	all data (%)	nodes	generations
GP	75.0	66.9	148	14328
+ pruning	82.1	71.4	17	247
+ fitting	86.0	72.2	35	5739
+ pruning and fitting	88.2	82.3	22	10327

individual structure is is used as an index in selecting individuals for genetic operations. Therefore, variety in the GP group is strongly maintained, and it is expected that GP learning is always done to a new combination.

In normal GP, genetic operators (crossover and mutation) are operated based on the individual fitness. In the proposed technique, genetic operators are operated based on the evaluation of the tree difference. As for the difference of the tree, two types of comparisons can be made. One is tree structure difference between two individuals. The other is difference in each individual's composed node types and numbers. In this session, we use the difference in composition node types and numbers as the tree difference. Following, the evaluation of the difference of the tree was defined by the following expression.

$$f_{diff}(A,B) = \sum_{i=1}^{K_{types}} |n_i(A) - n_i(B)|$$

$f_{diff}(A,B)$: Evaluation value of difference in tree structures A and B
K_{types}: the total number of defined node types
$n_i(\bullet)$: the number of i-th node type used in $\bullet$

The proposed evaluation value is built in to the genetic operator in GP. In this session, we used crossover as the modified genetic operator. In normal crossover operation, two target individuals are chosen based on the individual fitness. In the crossover operation modified by the proposed technique, one individual is chosen based on the individual fitness, and another individual is chosen based on the evaluation value of the difference of the tree structure with the first one.

In previous GP techniques, when there are a large number of node types defined, the search efficiency is raised by decreasing the examined node combination [Koza 1994a, Niimi 1999b]. However, this approach may cause the search to find only a local optimum solution and fail to reach the global optimum solution. In the modified GP using the proposed evaluation value, the examined combination has effectively been increased, allowing a better probability for the GP to reach the global optimum solution.

To verify the validity of the proposed method, we developed a rule generation system from a medical database. We compared the result of the proposed method with normal GP, using a database for the occurrence of hypertension. The parameter of GP used the following. (Refer to table 6.11.) The proposed method was shown to give a more accurate rule set compared to normal GP. The proposed method also showed a greater improvement in fitness towards the end of the training cycle compared to normal GP. However, the number of generations until termination had increased for the proposed method. (Refer to table 6.12.)

The proposed method allows GP to strongly maintain variety within the GP group. Therefore when there are a large number of node types defined, the proposed GP increases the probability to reach the global optimum solution.

Table 6.11: Parameters of GP

GP population	500
Reproduction probability	0.1
Crossover probability	0.4
Mutation probability	0.1
Proposed crossover probability	0.2
Selection method	Tournament method, Elite preservation
Function node	IFLTH, IFEQ, *, /, +, −, ADF0, ADF1
Terminal node	Attribute value of database such as SEX and AGE (15 attributes), P, N, R
Number of training data	P(100) , N(100)
Number of test data	1024

IFLTH,IFEQ: if less than (<),if equal to(=)
ADF0, ADF1: the function definition gene expanded by ADF
R: randomly generated constant
P, N: ocurence (P), no-occurence (N)

Table 6.12: Experimental Result (Reasoning Precision) by Each Technique

	training data (%)	all data (%)	nodes	generations
GP	75.0	66.9	148	10327
Proposed Methods	88.2	82.3	22	14328
C4.5 (to reference)	98.5	77.1	29	—

Chapter 7

Conclusion

A lot of researches on data mining are proposed. A lot of researches on genetic programming are proposed. However, the research by which both are treated is not proposed.

In this dissertation, we propose the technique by which genetic program are combined with an existing data mining technique. By using genetic programming for the data mining technique is, that the knowledge representation can become more rich and data mining process can be contain a probabilistic operation. It is expected to be able to cover the slowness of learning speed by genetic programming, and to be possible to construct the system even if the background knowledge at data mining is a little become possible to use.

At first, we discuss the problems of genetic programming allied to data mining.

Secondly, we proposed a decision tree construction method combined with C4.5 and GP.

The proposed method has some advantages. The first advantage is that the proposed method can improve decision tree's accuracy. The second advantage is that its learning speed becomes faster than normal GP's. The third advantage is that the design of a genetic programming becomes easy.

We experimented by using two kinds of medical datasets to verify of proposed method. The domain experts gave the evaluation and comments on the results. The problem was taken up two things. One is intended for dataset which contained a lot of continuous values. The other is intended for dataset with a lot of numbers of attributes. We compared the results of three methods (using C4.5 only, using ADF-GP only, and using combined C4.5 and ADF-GP using the proposed method).

In the proposed method, the decision tree has been improved compared with GP and C4.5. The number of generations until best individual has been improved. The combination overhead was able to be disregarded because of the speed improvement of GP.

Thirdly, we proposed the rule discovery technique from the database using genetic programming combined with Apriori algorithms.

The proposed method has some advantages. The first advantage is that the proposed method can improve decision tree's accuracy. The second advantage is that its learning speed becomes faster than normal GP's. The third advantage is that the design of a

genetic programming becomes easy.

We experimented by using two kinds of datasets to verify of proposed method. For one experiment, the domain expert gave the evaluation and comments on the results. In the proposed method, the decision tree has been improved compared with GP and Apriori algorithm. The number of generations until the best individual has been decreased. The combination over-head was able to be disregarded because of the speed improvement of GP. Though the result of suppressing the overfitting was seen, we did not consider the analysis of its mechanism.

In the forth, the decision tree and the rule expression by genetic programming was proposed, and the expression which used AND, OR and NOT were mounted on genetic programming besides the rule expression of the if-else form. Moreover, the decision tree was constructed by using the house-votes data from UCI Machine Learning Repository to verify the effectiveness of the modified rule expressions, and we compared the results.

As a result, the size of the decision tree was able to be improved. Moreover, the improvement of accuracy was admitted in the one that AND and OR were used. By using NOT function, the accuracy of the decision tree was improved. The expanding rule expression is mounted by replacing the definition of the function node of genetic programming. Therefore, the modification of the framework of genetic programming is minimum. It can be said that the rule expression which uses AND, OR and NOT from this is effective in genetic programming.

In finally, we discussed about three improvement methods of genetic programming. The first methods improves the problem that genetic programming is difficult to construct its system. The second methods improves the problem that genetic programming is difficult to treat with continuous value attributes. The third method improves the problem that genetic programming is difficult to use with a lot of nodes. We applied them to experiments of data mining. As a result of each experiments, each method shows improvement learning efficiency.

Considering these discussions and proposed method, it can be concluded that the combined method with genetic programming and data mining techniques has the improvement of the calculation speed and accuracy. In general, a relationship between search space and discovering unexpected rules becomes trade-off, but using our proposed techniques are able to improve both. Because an existing data mining technique is smaller than the amount of the calculation of genetic programming, concerning an increase in the amount of the calculation by the combination does not become a problem. It can be said that correspondence will become possible by added the improvement technique by genetic programming concerning difficult learning of the continuous value attributes and a lot of defined nodes. It can be concluded that using genetic programming for data mining, is possible not only to improve accuracy of discovered knowledge, but also to extract knowledge which is not easy to be obtained by existing techniques.

In the future, we consider to apply to the character-strings based databases (such as text mining), expansion for application to the time series databases, development for large scale databases. Moreover, we will discuss to make a policy how to combined genetic programming with other techniques.

References

[Adriaans 1996] P. Adriaans, D. Zantinge: Data Mining. Addison Wesley Longman, 1996

[Agrawal 1993] R. Agrawal, T. Imielinski, and A. Swami: Mining Associations between Sets of Items in Massive Databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington, D.C., USA, pp.207–216, 1993

[Agrawal 1994] R. Agrawal and R. Srikant: Fast Algorithms for Mining Association Rules. In Proceedings 20th International Conference on Very Large Databases, Santiago, Chile,1994

[Aiyoshi 1998] E. Aiyoshi, H. Tamaki, N. Sannomiya, A. Yoshikawa, Y. Kobayashi, K. Handa, N. Harada, H. Mori, S. Mori, S. Harada, K. Tanaka: Genetic Algorithms and Neural Networks : Scheduling and Combinattorial Optimization . The Institute of Electrical Engineers of Japan(IEEJ), Corona Publishing Co., 1998 (In Japanese)

[Arikawa 1998] S. Arikawa, M. Haraguchi: Predicate Logic and Logical Programming. Ohmsha, 1998 (In Japanese)

[Arikawa 1999] S. Arikawa, K. Furukawa(Eds.): Discovery Science 1999. Lecture Notes in Artificial Intelligence 1721, Springer, 1999

[Arikawa 2000] S. Arikawa, K. Furukawa(Eds.): Discovery Science 2000. Lecture Notes in Artificial Intelligence 1967, Springer, 2000

[Banerjee 1997] A. Banerjee: Initializing Neural Networks Using Decision Trees. Computational Learning Theory and Natural Learning Systems Volume 4: Making Learning Systems Proctical, pp.3–15, 1997

[Benyahia 1998] I. Benyahia, J. Potvin: Decision Support for Vehicle Dispatching Using Genetic Programming. IEEE Transactions on Systems, Man and Cybernetics-Part A:Systems and Humans, Vol.28, No.3,pp.306–314 ,1998

[Berry 1997] M. J. A. Berry, G. S. Linoff: Data Mining Techniques: for Marketing, Sales, and Customer Support. John Wiley & Sons, Inc., 1997

[Blake 1998] C. L. Blake, C. J. Merz: UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science, 1998

[Borgelt 1996] C. Borgelt:apriori — find association rules/hyperedges with apriori algorithm. [http://fuzzy.cs.uni-magdeburg.de/~borgelt], 1996

[Boros 2000] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik: An Implementation of Logical Analysis of Data. IEEE Transactions on Knowledge and Data Engineering, Vol.12, No.2, pp.292–306 ,2000

[Cattral 1999] R. Cattral, F. Oppacher, D. Deugo: Rule Acquisition with a Genetic Algorithm. Proceedings of the Genetic and Evolutionary Computation Conference, Vol. 1, p. 778, Morgan Kaufmann ,1999

[Cohen 2001] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullmann, C. Yang: Finding Interesting Associations without Support Pruning. IEEE Transactions on Knowledge and Data Engineering, Vol.13, No.1, pp.64–78 ,2001

[Crow 1983] J. F. Crow: Genetics Notes. Eighth Edition, Burgess Publishing Company, 1983

[Dingsoyr 1997] T. Dingsoyr, E. M. Lidal: An Evaluation of Data Mining Methods and Tools. Student Project Reports, Knowledge Systems Group, Norwegian University of Science and Technology ,1997

[Gruau 1994] F. Gruau: Genetic Micro Programming of Neural Networks. Advances in Genetic Programming, edited by K. E. Kinnear, Jr., The MIT Press,pp.495–518 ,1994

[Hemsathapat 2001] K. Hemsathapat, C. H. Dagli, D. Enke: Using a Neuro-Fuzzy-Genetic Based Data Mining Architecture. Working Notes of JSAI KDD Challenge 2001, JKDD01, Matsue, Shimane, Japan,pp.17–24 ,2001

[Ho 1998] T. B. Ho, T. D. Nguyen, N. B. Nguyen, T. Ito: Development of Some Methods and Tools for Discovering Conceptual Knowledge. Discovery Science 1998, pp.415–416 ,1998

[Iba 1993] H. Iba: Genetic Algorithms and Genetic Search. Instrument and Control, Journal of the Society of Instrument and Control Engineers Vol.32 No.1:39-45,1993 (In Japanese)

[Iba 1994] H. Iba: Genetic Programming and Evolutionary Learning. Journal of Japanese Society for Artificial Intelligence, Vol.9, No.4, pp.512–517, 1994 (In Japanese)

[Iba 1996] H. Iba: Genetic Programming. Tokyo Denki University Press, 1996 (In Japanese)

[Iba 1999] H. Iba: An Introduction to Evolutionary Computation. University of Tokyo Press, 1999 (In Japanese)

[Iba 2000] H. Iba: Genetic Programming and Evolutionary Computation. Journal of Japanses Socierty for Artificial Intelligence Vol.15 No.2, pp.251-258,2000 (In Japanese)

[Ichimura 1997] T. Ichimura, K. Ooba, E. Tazaki, H. Takahashi and K. Yoshida: Knowledge Based Approach to Structure Level Adaptation of Neural Networks. In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'97), Orlando, Florida, USA,pp548–553,1997

[Ikeda 2001] T. Ikeda, T. Washio, H. Motoda: Basket Analysis on Meningitis Data. Working Notes of JSAI KDD Challenge 2001, JKDD01, Matsue, Shimane, Japan, pp.33–40 ,2001

[Ito 1998] T. Ito, H. Iba, S. Sato: Depth-Dependent Crossover for Genetic Programming. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation(ICEC'98), pp.775–780 ,1998

[Kita 1997] H. Kita: An Overview of Theoretical Studies on Evolutionary Algorithms. Proceedings of 13th Fuzzy System Symposium, pp.327–329,1997 (In Japanese)

[Kitsuregawa 1997] M. Kitsuregawa: Mining Algorithms for Association Rules. Journal of Japanese Society for Artificial Intelligence, Vol.12 No.4,pp.513–520,1997 (In Japanese)

[Kobayashi 1993] S. Kobayashi: Perspective of Genetic Algorithms, Instrument and Control. Journal of the Society of Instrument and Control Engineers Vol.32 No.1,pp.2–9,1993 (In Japanese)

[Koza 1992a] J. R. Koza: Genetic Programming, On the Programming of Computers by Means of Natural Selection. The MIT Press,1992

[Koza 1992b] J. R. Koza: Evolution of Emergent Behavior. Genetic Programming, On the Programming of Computers by Means of Natural Selection, The MIT Press, pp.329–355 ,1992

[Koza 1994a] J. R. Koza: Scalable Learning in Genetic Programming Using Automatic Function Definition. Advances in Genetic Programming, edited by K. E. Kinnear, Jr., The MIT Press, pp.99–117, 1994

[Koza 1994b] J. R. Koza: Genetic Programming II, Automatic Discovery of Reusable Programs. The MIT Press,1994

[Koza 1994c] J. R. Koza: Prediction of Transmembrane Domains in Proteins. Genetic Programming II, Automatic Discovery of Reusable Program, The MIT Press,pp.445–492 ,1994

[Koza 1994d] J. R. Koza: Prediction of Omega Loops in Proteins. Genetic Programming II, Automatic Discovery of Reusable Program, The MIT Press, pp.493–504 ,1994

[Koza 1994e] J. R. Koza: Lookahead Version of the Transmembrane Problem. Genetic Programming II, Automatic Discovery of Reusable Program, The MIT Press,pp.505–524 ,1994

[Koza 1999] J. R. Koza, F. H Bennett III, D. Andre, M. A. Keane: Genetic Programming III, Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, 1999

[Langdon 1998] W. B. Langdon: Genetic Programming and Data Structures. Kluwer Academic Publishers ,1998

[Liu 1998a] H. Liu, H. Motoda: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, 1998

[Liu 1998b] H. Liu, H. Motoda: Feature Extraction, Construction and Selection: A Data Mining Perspective. Kluwer Academic Publishers, 1998

[Lu 1997] H. Lu, H. Motoda, H. Liu(Eds.): KDD: Techniques and Applications. Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining, World Scientific, 1997

[Maeda 1998] Y. Maeda, S Kawaguchi: Adaptive Search Method for Genetic Programming Using Fuzzy Reasoning. Proceedings of 14th Fuzzy Symposium, pp.267–269, 1998 (In Japanese)

[Mannila 1997] H. Mannila: Inductive Databases and Condensed Representations for Data Mining. International Logic Programming Symposium,pp.21–30 ,1997

[Maruyama 2000] K. Maruyama, K. Uehara: Knowledge Integration of Rule Mining and Schema Discovering. Discovery Science 2000, Kyoto, Japan, Lecture Notes in Artificial Intelligence 1967, Springer, pp.285–289 ,2000

[Matsumoto 1998] N. Matsumoto, E. Tazaki: Emergence of Learning Rule in Neural Networks Using Genetic Programming Combined with Decision Trees. IEEE International Conference on Systems, Man, and Cybernetics, (SMC'98), San Diego, California, USA,pp.1801–1805 ,1998

[Naemura 1999] T. Naemura, T. Hashiyama, S. Okuma: Hierarchical Module Generation in Genetic Programming and Its Incremental Evolution. Transactions of the Society of Instrument and Control Engineers Vol.35, No.10, pp.1292–1299,1999 (In Japanese)

[Nakayama 1999] H. Nakayama, Y. Hattori, R. Ishii: Rule Extraction based on Rough Set Theory and its Application to Medical Data Analysis. IEEE International Conference on Systems, Man, and Cybernetics (SMC'99), Yurakucho, Tokyo, Japan, pp.924–929 ,1999

[Nakayama 2000] Y. Nakayama, N. Otani, M. Shimura: Optimization of Decision Trees by Using Symbiotic Evolution. Proceedings of 14th Fuzzy System Symposium,pp.327–328, 2000 (In Japanese)

[Niimi 1999a] A. Niimi, E. Tazaki: Combined Learning Method of Decision Tree and ADF GP by Object Oriented Approach. Proceedings of 42nd Knowledge Based System meeting,pp.75–82, Japan Artificial Intelligence Society, SIG–KBS–9802, 1999 (In Japanese)

[Niimi 1999b] A. Niimi, E. Tazaki: Extended Genetic Programming using pruning redundant patterns and fitting random continuous nodes. Proceedings of the 13th Annual Conference of Japanese Society for Artificial Intelligence, pp.257–258, 1999 (In Japanese)

[Niimi 1999c] A. Niimi, E. Tazaki: Object oriented approach to combined learning of decision tree and ADF GP. 1999 International Joint Conference on Neural Networks, Washington DC, USA, 1999

[Niimi 1999d] A. Niimi, E. Tazaki: Extended Genetic Programming using Reinforcement Learning Operation. In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'99), Tokyo, Japan, pp.596–600, 1999

[Niimi 2000a] A. Niimi, E. Tazaki: Learning Method using Genetic Programming Combined with Association Rule Algorithm. Proceedings of the 14th Annual Conference of Japanese Society for Artificial Intelligence, pp.270–271, 2000 (In Japanese)

[Niimi 2000b] A. Niimi, E. Tazaki: Genetic Programming Combined with Association Rule Algorithm for Decision Tree Construction. Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering System & Allied Technologies, volume 2,pp.746–749, 2000.

[Niimi 2000c] A. Niimi, T. Tazaki: Knowledge Discovery using Extended Genetic Programming from Biochemical Data. Proceedings of 49th Knowledge Based System meeting, pp.45–49, Japan Artificial Intelligence Society, SIG–KBS–A002, 2000 (In Japanese)

[Niimi 2000d] A. Niimi, E. Tazaki: Rule Discovery Technique Using Genetic Programming Combined with Apriori Algorithm. In Proceedings of the Third International Conference on Discovery Science, Kyoto, Japan, Lecture Notes in Artificial Intelligence 1967, Springer, pp.273–277, 2000

[Niimi 2001a] A. Niimi, E. Tazaki: Extended Genetic Programming using Apriori Algorithm for Rule Discovery. Working Notes of JSAI KDD Challenge 2001, JKDD01, Matsue, Shimane, Japan:41–48 ,2001

[Niimi 2001b] A. Niimi, E. Tazaki: Combined Method of Genetic Programming and Association Rule Algorithm. Applied Artificial Intelligence, Volume 15, Number 9, 2001 October, pp.825–842 ,2001

[Okada 2000] T. Okada: Finding Discrimination Rules Using the Cascade Model. Journal of Japanses Socierty for Artificial Intelligence Vol.15 No.2, pp.321–330 ,2000 (In Japanese)

[Ono 2000] I. Ono, M. Yamamura, H. Kita: Real-Coded Genetic Algorithms and Their Applications. Journal of Japanses Socierty for Artificial Intelligence Vol.15 No.2:259-266,2000 (In Japanese)

[Quinlan 1995] J. R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers,1993

[Radcliffe 1994] N. J. Radcliffe, P. D. Surry: Co-operation through Hierarchical Competition in Genetic Data Mining. EPCC-TR94-09: Parallel Problem Solving From Nature, pp.1–10 ,1994

[Sanchez 2000] L. Sanchez: Interval-Valued GA-P Algorithms. IEEE Transactions on Evolutionary Computation, Vol.4, No.1,pp.64–72 ,2000

[Shimohara 1998] K. Shimohara, T. S. Ray, H. Garis, J.Mizoguchi, H. Hemmi, J. Vaario, K. Wada, A. Imada: Artificial Life and Evolutionary System. Tokyo Denki University Press, 1998 (In Japanese)

[Sigmund 1993] K. Sigmund: Games of Life. Oxford University Press, 1993

[Suyama 1999] A. Suyama, N. Negishi, T. Yamaguchi: Design and Evaluation of an Environment to Automate the Construction of Inductive Applications. Discovery Science 1999, Tokyo, Japan, Lecture Notes in Artificial Intelligence 1721, Springer, pp.103–114,1999

[Terabe 2000] M. Terabe, O. Katai, T. Sawaragi, T. Washio, H. Motoda: Attribute Generation Based on Association Rules. Journal of Japanese Society for Artificial Intelligence, Vol.15 No.1,pp.187–197,2000 (In Japanese)

[Tsumoto 1998] S. Tsumoto(Eds.): Proceedings of 42nd Knowledge Based System meeting. Japan Artificial Intelligence Society, SIG–KBS–9802, 1998 (In Japanese)

[Tsumoto 1999a] S. Tsumoto(Eds.): Proceedings of 38th SIG-FAI and 45th Knowledge Based System meeting. Japan Artificial Intelligence Society, SIG–FAI/KBS–9902, 1999 (In Japanese)

[Tsumoto 1999b] S. Tsumoto, T. Terano, M. Inada, A. Niimi, E. Tazaki, N. Negishi, A. Suyama, T. Yamaguti, Y. Tachibana, J. Dong, N. Zhong, S. Ohsuga, Y. Kamiishi, S. Sugaya, E. Suzuki, M. Tsukada, A. Inoguchi, T. Washio, H. Marsuzawa, T. Okada, M. Oyama, T. B. HO, T. D. NGUYEN, N. B. NGUYEN: Comparison of Data Mining Methods using Common Medical Datasets. Proceeding of Data Mining and Knowledge Discovery in Data Science(ISM Symposium), The Institute of Statistical Mathematics,pp.63–72,1999

[Tsumoto 2000] S. Tsumoto(Eds.): Proceedings of 49th Knowledge Based System meeting. Japan Artificial Intelligence Society, SIG–KBS–A002, 2000 (In Japanese)

[Ullman 2000] J. D. Ullman: A Survey of Association-RUle Mining. In Proceedings of the Third International Conference on Discovery Science, Kyoto, Japan, Lecture Notes in Artificial Intelligence 1967, Springer, pp.1–14, 2000

[Umano 1996] M. Umano, M. Yoshimura, I. Hatono, H. Tamura: Generation of Fuzzy Decision Trees by Genetic Algorithm. Proceedings of 12th Fuzzy System Symposium, pp.823–826,1996 (In Japanese)

[Washio 2001] T. Washio(Eds.): Working Notes of JSAI KDD Challenge 2001. JKDD01, Matsue, Shimane, Japan ,2001

[Wong 1999] M. L. Wong, W. L., K. S. Leung, J. C.Y. Cheng: Applying Evolutionary Algorithms to Discover Knowledge from Medical Databases. IEEE International Conference on Systems, Man, and Cybernetics (SMC'99), Yurakucho, Tokyo, Japan,pp.936–941 ,1999

[Yamamura 1994] M. Yamamura, S. Kobayashi: Toward Application Methodology of Genetic Algorithms. Journal of Japanses Socierty for Artificial Intelligence Vol.9 No.4, pp.506–511,1994 (In Japanese)

[Yoshida 1996] K. Yoshida, M. Yamamura, S. Kobayashi: Buiding Probabilistic Decision Trees Considering Locality of the Discriminating Power of Each Attribute. Journal of Japanses Socierty for Artificial Intelligence Vol.11 No.2:264-272,1996 (In Japanese)

[Yoshiie 2000] S. Yoshiie, T. Senaha, A. Niimi, E. Tazaki: An Extended Genetic Programming Approach for Initial Condition Problem. Proceedings of 16th Fuzzy System Symposium, pp.69–72,2000 (In Japanese)

[Zongker 1996] D. Zongker, B. Punch, B. Rand: lil-gp 1.01 User's Manual. Michigan State University, 1996