Journal of Intelligent Manufacturing, 16, 645–661, 2005

Discovering multiple diagnostic rules from coronary heart disease database using automatically defined groups

AKIRA HARA[1], TAKUMI ICHIMURA[1] and KATSUMI YOSHIDA[2]

[1]Faculty of Information Sciences, Hiroshima City University, 3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima, 731-3194, Japan
E-mail: {ahara, ichimura}@its.hiroshima-cu.ac.jp
[2]Department of Preventive Medicine, St.Marianna University School of Medicine, 2-16-1, Sugao, Miyamae-ku, Kawasaki 216-8511, Japan
E-mail: k2yosida@marianna-u.ac.jp

Received May 2004 and accepted November 2004

Much of the research on extracting rules from a large amount of data has focused on the extraction of a general rule that covers as many data as possible. In the field of health care, where people's lives are at stake, it is necessary to diagnose appropriately without overlooking the small number of patients who show different symptoms. Thus, the exceptional rules for rare cases are also important. From such a viewpoint, multiple rules, each of which covers a part of the data, are needed for covering all data. In this paper, we describe the extraction of such multiple rules, each of which is expressed by a tree structural program. We consider a multi-agent approach to be effective for this purpose. Each agent has a rule that covers a part of the data set, and multiple rules which cover all data are extracted by multi-agent cooperation. In order to realize this approach, we propose a new method for rule extraction using Automatically Defined Groups (ADG). The ADG, which is based on Genetic Programming, is an evolutionary optimization method of multi-agent systems. By using this method, we can acquire both the number of necessary rules and the tree structural programs which represent these respective rules. We applied this method to a database used in the machine learning field and showed its effectiveness. Moreover, we applied this method to medical data and developed a diagnostic system for coronary heart diseases.

Keywords: evolutionary computation, genetic programming, multi-agent system, rule extraction

1. Introduction

Due to the advance of information technology in recent years, patient diagnostic data in hospitals have been accumulated in databases. When deciding a treatment policy such as the necessity for surgery for a patient, the decision is based on the patient's current health. It is important to understand the general tendency of diseases from stored data, and to use this information in order to determine the treatment policy for new patients. Discovering effective diagnostic rules from a large amount of data in a database is a worthwhile but challenging effort which would help doctors administer medical treatment (Ichimura *et al.*, 1995, 2001, 2003 a, b).

However, there are some problems in realizing this goal. One problem results from the complexity of medical data. Diagnoses by doctors are based on various information, such as the numerical results of various tests, X-rays or endoscope images, palpations of diseased parts, patients' complexions, and so on. All of this information, including diagnostic results, are accumulated in a medical database. However, it is difficult to add all information to the database in a consistent, impartial manner. For example, sense information such as palpations of diseased parts or patients' complexions are biased by the doctor's subjective judgment. Image information may be converted into checklists of the features most important to the doctor. Occasionally, a special examination that is not included in the database could be performed. Thus, the data accumulated in the medical database may be missing a part of the information used when doctors diagnosed a disease. As another complexity of medical data, consider the case that a doctor diagnoses a progressive disease. When the disease progresses, it may be called by another name. However, the examination results of these diseases show similar tendencies. In another situation, a doctor may have to judge if the numerical result of a medical test is considered in the normal range or an abnormal range, which would indicate the necessity of treatment. In these situations, the doctor has a choice of diagnostic results (e.g., Disease A or B, normal or abnormal.). However, the borderline between two states or results is often ambiguous. Therefore, the judgments may be influenced by the doctor's experience. In addition, the medical data is imprecise information about a living body, and involves the influence of patients' personal equations, the doctors' measurement skills, and the noise of the measuring instruments. For these reasons, medical databases often include exceptional, inconsistent or incomplete data. Knowledge discovery from such medical databases is more difficult than that from standard test problems in the machine learning field. In this research, we focus particularly on the handling of exceptional data. Much research on extracting rules in the machine learning field has focused on the extraction of a general rule that covers as many data as possible. Some data which do not satisfy a general rule may be abandoned as exceptional data or noise. However, because medical treatment is a field where people's lives are at stake, knowledge that can be used for a small number of patients with exceptional symptoms is also very important. A key point to the achievement of high-quality medical treatment is whether such patients are treated appropriately without missing any symptoms. From such a viewpoint, multiple rules, each of which covers a part of the data, are needed to cover all data. In this paper, we plan to extract multiple rules, including exceptional rules, from a database.

Another problem for realizing scientific diagnoses is the management of medical data. Because medical data relates to a patient's privacy, an actual database cannot be offered to researchers on machine learning. In order to offer the medical database to researchers, people in the medical field need to construct a modified database in which data violating patient privacy are deleted. Moreover, it is necessary to have a database which contains a large amount of data that has accumulated for many years from the same viewpoint, such as the collection of patient data from continuing examinations and long-term observations of disease progress in specific patients. Thus, it is impossible to wrestle with the important problem of acquisition of diagnostic knowledge Without the cooperation of medical people and a reliable database. Therefore, researchers have employed open scientific databases (e.g., Iris plant database, Monk database) for the verification of their proposed methods. As a result, neither a comparison nor an examination of various rule extraction methods on actual complicated problems such as a medical database has been done. In order to resolve this impasse, Suka *et al.* (2004) developed open medical databases which reproduce real-world data related to the development of coronary heart diseases. This is very meaningful because we can discuss the advantages and disadvantages of various methods for an actual complicated health care problem. In this paper, we perform experiments using the medical databases.

We plan to extract not only general diagnostic rules but also exceptional diagnostic rules. Moreover, we intend not only to improve the prediction accuracy but also to acquire useful and comprehensible knowledge. Genetic Programming (GP) (Koza, 1992) is used for the rule extraction method. Moreover, we also use a multi-agent approach for the rule extraction. We assume that some agents extract a general rule from frequently observed data and other agents extract exceptional rules from a small quantity of exceptional data. By the cooperation of multiple agents, rules that cover all of the data are extracted. In order to realize these rules, we use an optimization method called Automatically Defined Groups (ADG) (Hara and Nagao, 1999, 2002; Hara *et al*., 2003, 2004), which was derived from Genetic Programming. The ADG has been proposed with the goal of the realization of effective cooperative behavior among heterogeneous agents. This method can optimize both the group structure of agents and the action rule of each group.

The contents of this paper are as follows. In Section 2, we describe GP and rule extraction using GP. In Section 3, we explain some conventional models for cooperative problem-solving and ADG. In Section 4, we propose a new method using ADG for rule extraction, and show its effectiveness with a preliminary experiment on the Monk database. In Section 5, we apply this method to a medical database for coronary heart diseases. In Section 6, we describe our conclusions.

2. Rule extraction by genetic programming

2.1. *Genetic programming*

In the process of evolution of living things, genetic code is transmitted from parents to children. As the genetic code is transmitted, recombination and mutation of the genes occur. As a result, the genetic code of some children is different from their parents. This variability of generations is repeated according to the principle of the survival of the fittest. This principle means that individuals that adjust better to the environment survive and reproduce at a higher rate, and less fit individuals survive, and reproduce at a lower rate. Living things thus evolve by these processes.

Evolutionary computations are optimization algorithms based on the mechanics of the evolution of living things. We prepare multiple individuals that represent candidate solutions for a given problem, and try to generate more suitable solutions by the repetition of genetic operations. Genetic Programming is one of these evolutionary computations, and it was proposed by Koza (1992). There are also Genetic Algorithms which can optimize individuals that are mainly expressed by one-dimensional bit strings (Holland, 1995; Goldberg, 1989). GP is an enhanced method that considers the tree structural program.

In GP, each individual representing a candidate solution is expressed by the tree structural program. The tree structural program corresponds to the chromosome of living things. The symbols of the tree structural program which can be internal nodes are called functional symbols, and the symbols which can be leaf nodes are called terminal symbols. Depending on the particular problem, the functions may be standard arithmetic operations, standard logical functions, or domain-specific functions. Depending on the setting, tree structural programs can represent numerical expressions, logical expressions, decision trees, action control programs of robots, and so on.

A population in each generation consists of a set of individuals. In the initial population, each individual is created by the random combination of functional and terminal symbols. Next, each individual in the population is measured in terms of how well it performs in a particular problem environment. This value is called the fitness.

Genetic operations are performed based on the fitness value. The basic operations in GP are selection, crossover and mutation. At first, individuals for the next generation are selected by the selection operation. Individuals with higher fitness will survive and reproduce at a higher rate. This reflects the principle of natural selection. Next, the genetic process of reproduction between two parental programs is performed to create new offspring programs from the two parental programs selected in proportion to fitness. This operation is called crossover. In a basic crossover operation, a node is selected at random from each parental tree program, and then subtrees with the nodes as root nodes are exchanged. After that, mutation operations are performed on each individual. The symbols of some nodes change to other symbols at random by mutations. A new population is created by these operations. Then, evaluations and genetic operations to this population are performed again. The performance of the individuals improves with this repetition. The acquired best individual may be a solution or an approximate solution to the problem.

2.2. *Rule extraction from data containing multiple rules*

Classification is an important problem extensively studied in several research areas, such as pattern recognition, machine learning and data mining. In the classification process, when the values of the predictor attributes of an instance are given, we have to predict the class of the instance. One of the rule formats used in classification is the IF_THEN rule. The rule's condition (IF) part is made by the conjunction of multiple terms. Each term is the combination of an attribute and the value which can be taken. The following expression is an example of the IF_THEN rule.

$$\text{Rule1}: \text{IF } (Attribute\_a = 1) \wedge (Attribute\_b < 2)$$
$$\text{THEN } (\textit{The data is Positive.})$$

This rule format has the advantage of being intuitively comprehensible to the user. So, the user can combine the knowledge contained in the discovered rules with his own knowledge in order to make intelligent decisions about a target classification problem—for example, to make a medical diagnosis.

In GP, the list structure (tree structural program), created by compositions of logical operators (AND) and relational operators (=, >, <) as functional symbols, along with attributes $(Att_a, \ldots, Att_n)$ and constants as terminals, can represent the IF_THEN rule. Each individual encodes the IF part of a rule, but not the THEN part (the predicted class). The reason for this is that in a given GP run, all individuals represent the rules predicting the same target class. For example, if we use $\{\text{AND},=,>,<\}$ as functions and $\{Att_a, \ldots, Att_n, Constant\}$ as terminals, the following tree structural program can be created by compositions of these functions and terminals.

$$(\text{AND}(= \ Att_a \ 1)(< \ Att_b \ 2)).$$

This expression is equivalent to Rule 1. Thus, GP is used as one method for rule extraction.

However the knowledge of a data set is unlikely to be sufficiently described by a single rule. The following Rule 2, which is different from Rule 1, may be needed. In this case, the class of instance will be judged by Whether either of these rules is satisfied.

$$\text{Rule 2}: \text{IF } (Attribute\_b = 1) \wedge (Attribute\_c > 1)$$
$$\text{THEN } (\textit{The data is Positive.})$$

We encounter similar situations in medical data. For instance, respective attributes represent a patient's condition, and the class of the instance represents a result of diagnosis. If we apply simple GP to such problems, however, only a single rule can be extracted. We cannot discover multiple rules. It is important for the realization of high-quality medical treatment to recognize two or more possibilities. In order to solve this problem, we propose a new method using ADG, by which the clustering of data and rule extraction in each cluster are performed simultaneously. This method uses a multi-agent approach. Next, we describe the approach and ADG.

Incidentally, it is also possible to include the OR symbol in the function set of GP in order to represent multiple rules by a single tree. However, when such a symbol setting is used, the rule is not necessarily expressed by the disjunctive normal form. Therefore, it is difficult to understand the rule. Moreover, extracting multiple rules without using the OR symbol leads to the important concept of the reliability of each rule. This idea is described in Section 4.

3. Automatically defined groups

3.1. *Co-operative problem-solving by multiple agents*

Co-operative problem-solving ,by multi-agent systems has attracted increasing attention in recent years. In the field of artificial intelligence, a number of attempts to generate co-operative behavior by means of Genetic Programming have been made in domain such as multi-robot control, and RoboCup soccer agents.

The acquisition of the action control rules of multi-agents by GP are mainly classified into two common methods. The first uses co-evolution, which means each agent is controlled by an individual. We perform an evaluation of the individual based on the agent's actions, and improve the performance of the team consisting of the respective agents. However, this method causes a credit assignment problem. This means that it is difficult to judge how much each agent contributes to the success when the problem is solved by cooperation.

The other method generates a team consisting of all agents, which together are considered an individual of GP. We perform an evaluation of the individual based on the performance of the team. This method facilitates the evaluation and can produce complex behavior such as the division of labor, because all agents are evaluated equally by the team's performance, regardless of their respective roles. Also, various models.(e.g., homogeneous model, heterogeneous model) have been proposed for this approach and experiments have been performed to compare the models (Luke and Spector, 1996; Iba, 1996, 1997).

When all agents in the environment take actions under identical rules, this team is called a homogeneous team. In GP, each agent refers to the same tree, as shown in Fig. 1. All agents decide their movements according to the same rules derived from the GP tree. However, since each agent is situated in a different environment, it is possible that each agent takes different actions according to the environmental conditions and solves the problem by cooperation with each other.

When each agent in the environment takes actions under distinct rules, this team is called a heterogeneous team. In GP, an individual maintains multiple trees, each of which is referred to by the corresponding agent, as illustrated in Fig. 1. In the heterogeneous model, various breeding strategies (restricted breeding, free breeding, and so on) have been proposed (Luke and Spector, 1996; Iba, 1996, 1997). Free breeding allows any member of a team to freely breed with any other member of another team. In restricted breeding, crossover operations are restricted to corresponding branch pairs. For example, restricted breeding allows team member 1 to breed only with another team member 1, and team member 2 to breed only with another team member 2, and so forth. Generally, restricted breeding works better than free breeding from the viewpoint that the restriction further promotes the preservation of diversity and specialization of each agent since it divides team members into separate breeding pools.

In order to solve a complex task requiring teamwork, the sharing of roles among agents is needed. Generally, it has been shown that the performance of heterogeneous agents, in which each member has distinct action rules, is higher than that of homogeneous agents, because each agent in the heterogeneous model is specialized according to each role.

This multi-agent approach is effective in solving problems that seem to be unrelated to the concept of agents. Soule applied a multi-agent approach to even-parity problems and linear regression problems and showed that the performance of this approach is higher than that of conventional solutions (Soule, 1999, 2000). In these experiments, a solution is obtained by collecting each agent's output.

In this research, we apply the same idea, which is the search for various solution by heterogeneous agents, to the domain of knowledge acquisition from the data. We handle data containing multiple rules in order to carry out the clustering of data as well as rule extraction from each cluster. We use a multi-agent approach to reach a solution. That is, the data are divided among agents. This corresponds to the clustering of data. And, each agent generates an approximate function for the assigned data. This corresponds to the rule extraction in each cluster. As a result, all rules are extracted by multi-agent cooperation.

In order to use this approach, however, we do not know the number of rules hidden in the data and how to allot the data to each agent. Moreover, as we prepare abundant agents, the number of tree structural programs in an individual increases. Therefore, the search performance declines.

3.2. *Automatically defined groups*

In order to solve the problems described in the previous section, we have proposed an improved GP method, ADG. The ADG optimizes both the grouping of agents and the program of each group in the process of evolution. By grouping multiple agents, we can prevent the increases of search space and perform an efficient optimization. Moreover, we can easily analyze the agents'

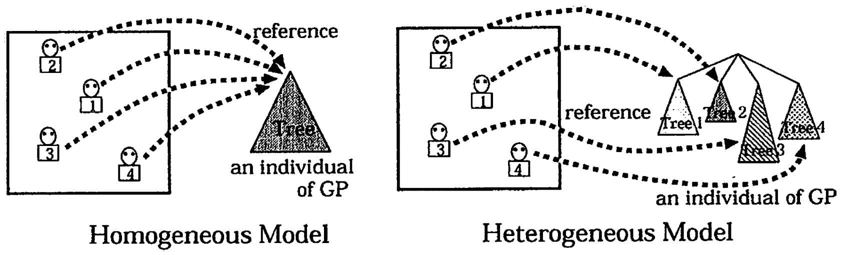

Fig. 1. Conventional models for multi-agent control.

behavior. The acquired group structure is utilized for understanding how many roles are needed and which agents have the same role. That is, the following three points are automatically acquired by using ADG.

- How many groups (roles) are required to solve the problem?
- Which group does each agent belong to?
- What is the program of each group?

A team that consists of all agents is regarded as one GP individual. One GP individual maintains multiple trees, each of which functions as a specialized program for a distinct group, as shown in Fig. 2. We define a group as the set of agents referring to the same tree for the .determination of their actions. All agents belonging to the same group use the same program.

When an initial population is generated, agents in each GP individual are divided into random groups, as shown in Fig. 3. The upper left-hand individual in Fig. 3 is a simple representation of the individual in Fig. 2. Each GP individual represents a multi-agent system.

Basically, crossover operations are restricted to corresponding tree pairs. For example, a tree referred to by an agent 1 in a team breeds with a tree referred to by an agent 1 in another team. However, we consider the sets of agents that refer to the trees used for the crossover. The group structure is optimized by dividing or unifying the groups according to the relationship of the sets. Individuals search for solutions as their group structures gradually approach the optimal structure.

The concrete processes are as follows: We arbitrarily choose an agent for two parental individuals. A tree referred to by the agent in each individual is used for crossover. We use T and T' as expressions of these trees, respectively. In each parental individual, we decide a set $A(T)$, which is the set of agents referring to the selected tree T. When we perform a crossover operation on trees T and T', there are the following three cases.

Type a: If the relationship of the sets is $A(T) = A(T')$, the structure of each individual is unchanged.

Type b: If the relationship of the sets is $A(T) \supset A(T')$, the division of groups takes place in the individual with T, so that only the tree referred to by the agents in $A(T) \cap A(T')$ can be used for crossover. The individual which maintains T' is unchanged.

For example, Fig. 4 shows the case in which the trees referred to by agent 2 are used for crossover. When we examine the sets of agents which refer to the trees, the set for the first parent is $\{2\}$ and the set for the second parent is $\{1, 2, 3, 4\}$. The set for the second parent includes the set for the first parent. The subtree used for crossover in the first parent is considered to be the special program to agent 2. If we perform the crossover operation, this operation affects not only the program of agent 2 but also the program of agents 1, 3, 4 in the second parent. Such an influence is considered unfavorable. Therefore, in the second parent we generate a new tree, which is the same tree that agent 2 refers to, and we shift agent 2 to the new group referring to the new tree. As a result, the division of groups takes place so that only the tree referred to by agent 2 can be used for crossover, and then we perform a crossover operation.

Type c: If the relationshp of the sets is $A(T) \not\supset A(T')$ and $A(T) \not\subset A(T')$, the unification of groups takes place in both individuals so that the agents in $A(T) \cup A(T')$ can refer to an identical tree.

For example, Fig. 4 shows the case in which the trees referred to by agent 1 are used for cross-

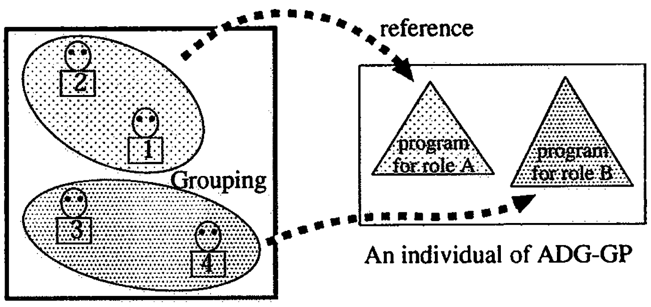

Fig. 2. Concept of automatically defined groups.

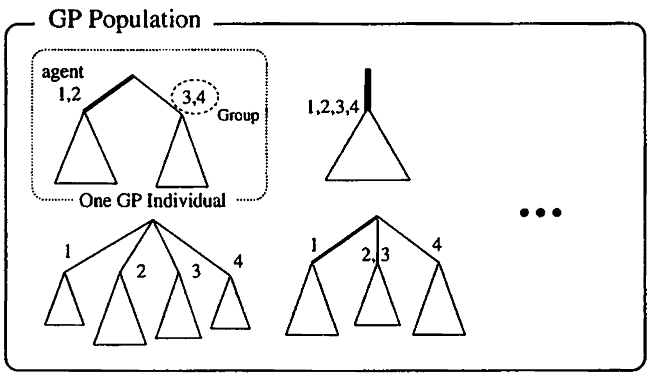

Fig. 3. Population of ADG.

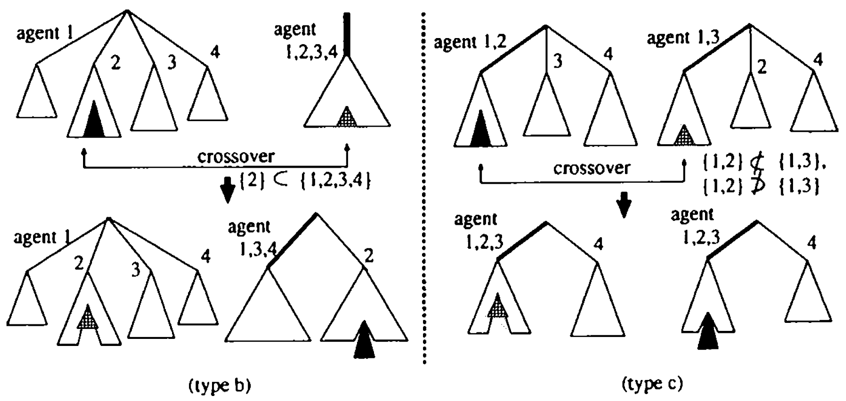

Fig. 4. Examples of crossover. (Type a) is not shown because the structure of each individual is unchanged.

over. When we examine the sets of agents which refer to the trees, the set for the first parent is {1, 2} and the set for the second parent is {1, 3}. One set does not contain another set of each other. Agents 1 and 2 have the same action control rules in the first parent, and agents 1 and 3 have the same action control rules in the second parent. We consider that agents 1, 2, 3, which are elements of the union of these sets, need to have the same action control rules in the offspring of the parents from the viewpoint of preservation of the characteristics. Therefore, in both parents we perform the unification of groups so that agents 1, 2, 3 can refer to the same tree, and then we perform a crossover operation.

As mentioned above, the difference between the sets of agents referring to the trees used for crossover causes the division or unification of groups.

Therefore, if all individuals converge to the same group structure, the changes of the group structure do not take place. So, group mutation operations are introduced into this method. The group mutation operation shifts an agent to an arbitrarily selected group, with a probability equal to the group mutation rate. This operation works as an accelerator for the change of the group structure by the crossover operations. So, we perform group mutation operations before crossover operations.

We expect that, by using this method, the search works efficiently and an adequate group structure is acquired. Besides, the acquired group structure becomes a clue for understanding the cooperative behavior and necessary division of labor.

4. Extracting multip1e rules from data

4.1. *Concept of the proposed method using ADG*

In this section, we describe how to apply ADG to rule extraction. Each agent group in an individual of ADG represents experts who use the tree structural program as the classification rule of data. Fitness is defined from mainly two viewpoints. One is the minimization of prediction errors. In the training data, each group predicts a class of an instance based on its attributes. If one or more groups judge that the instance belongs to the target class, the instance is classified into the target class. To classify the target class's instance into another class or to classify a non-target class's instance into the target class is regarded as mis-classification. We minimize the number of mis-classifications for all data.

The other viewpoint of fitness is load balancing among agents. The quantity of data which an agent takes charge of is considered to be the agent's load. That is the load corresponds to the number of data which a certain group takes charge of divided by the number of agents that belong to the group.

We calculate the variance of the load in all agents. Fitness is defined as the weighted sum of the prediction error and load balancing. We minimize the fitness by evolution.

In addition, in order to inhibit the redundant division of groups, the fitness is multiplied by the penalty $\gamma^{G-1} (\gamma > 1)$ according to the increase of the number of groups, G, in the individual.

Table 1. Attributes of Monk database

Attribute	*Value*
head_shape	1=round; 2=square; 3=octagon
body_shape	1=round; 2=square; 3=octagon
is_smiling	1=yes; 2=no
holding	1=sword; 2=balloon; 3=flag
jacket_color	1=red; 2=yellow; 3=green; 4=blue
has_tie	1=yes; 2=no
class	0=False; 1=True

The features of this method are as follows.

- In conventional methods, most of the research has focused only on problems of clustering or rule extraction from data that have already been classified. The object of this research is the data containing multiple rules, and the clustering of data and the rule, extraction in each cluster are performed Simultaneously.
- The number of rules hidden in the training data can be automatically acquired. Moreover, as a result of load balancing among agents, the ratio of the number of agents in each group corresponds to the ratio of the appearance of each rule. Therefore, we can understand the probability of the appearance of each rule.

4.2. *Preliminary experiment on a monk database*

In this section, we describe the detail of rule extraction. We applied this method to a database in the machine learning field, the Monk database (Thrun *et al.*, 1991) and showed the method's effectiveness. This database contains attributes for artificial robots. As shown in Table 1, each robot has six attributes, and it is classified into a positive or negative class based on hidden knowledge. Table 2 shows examples of data in the database. The task of the Monk problem is to acquire rules for judging whether each robot belongs to the positive class.

In the Monk database, there are three data sets. We use the Monk 1 data set for this preliminary experiment. The Monk 1 data set has 124 examples in the training set, which contains 62 positive examples, and 62 negative examples. The testing set has 432 examples, Which consist of 216

Table 2. Examples of data in the Monk database

Class	*head_shape*	*body_shape*	*is_smiling*	*holding*	*jacket_color*	*has_tie*
positive	1	2	1	1	1	2
positive	3	3	1	2	3	2
negative	1	2	1	1	2	1
negative	2	3	2	3	3	2

positive and 216 negative examples. The hidden knowledge for classification in this database is (head_shape = body_shape) or (jacket_color = 1). There are no mis-classifications in either the training or the testing set.

In order to judge whether each instance is regarded as a positive case, we need to find logical expressions which only the data in positive cases can satisfy. The logical expression is made by the conjunction of multiple terms. Each term is the combination of an attribute item and the value which can be taken. The following expression is an example.

$$(\texttt{head\_shape}=1)\wedge(\texttt{is\_smiling}=2)\wedge(\texttt{holding}\neq 1)$$

The logical expression has to return false for negative cases.

Multiple trees in an individual of ADG represent the respective logical expressions. The training data set contains two classes, and each data in the training set is input to all trees in the individual. Then, calculations are performed to determine whether the data satisfy each logical expression. The input data is regarded as positive if one or more logical expressions in the individual return true. Conversely, the input data is not regarded as positive if all logical expressions in the individual return false. If some data are not classified by the current logical expressions properly, new trees may be created by evolution and the data would be shared by them. Individuals are optimized so that one in the multiple tree programs can return true for positive cases and all trees can return false for negative cases.

The concept of each agent's load arises from the cooperative problem-solving by multiple agents. The load'is calculated from the adopted frequency of each group's rule and the number of agents in each group. The adopted frequency of each rule is counted when the rule successfully returns true for each positive data. If multiple trees return true for a positive data, the tree with, more agents is adopted. When the kth agent belongs to group g, the load of the agent is defined as follows.

$$w_k = \frac{(\text{adopted frequency of } g) \times N_{agent}}{(\text{Number of agents which belong to } g) \times N_{all\_adoption}}.$$

In this equation, N_{agent} represents the number of all agents in one GP individual, and $N_{all\_adoption}$ represents the sum of the adopted frequencies of all groups. By balancing every agent's load, more agents are allotted to the group that has a greater frequency of adoption. On the other hand, the number of agents in the less adopted group becomes small. Therefore, we can acquire important knowledge about the ratio of use of each rule. The ratio indicates how general each rule is for judgment of the classification. Moreover, when negative cases are judged to be true through a mistake of a rule, it is thought that the number of agents who support the rule should be small.

To satisfy the requirements mentioned above, fitness f is calculated by the following equation. We Maximize f by evolution.

$$f = -\frac{miss\_target\_data}{N_{positive}} - \alpha\frac{misrecognition}{N_{negative}} - \beta\frac{\sum_{N_{negative}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w.$$

In this equation, $N_{positive}$ and $N_{negative}$ represent the number of positive cases and negative cases in the database, respectively. *miss_target_data* is the number of missing data in the target positive data that should have been judged to be true. *misrecognition* is the number of mistakes through which negative data is regarded as positive. When the rule returns true for negative data, *fault_agent* is the number of agents who support the wrong rule in each data. So, the third term represents the average rate of agents who support the wrong rules when misrecognition occurs.

V_w is the variance of every agent's load. In addition, in order to inhibit the redundant division, of groups, f is multiplied by $\gamma^{G-1}(\gamma > 1)$ according to the increase of the number of groups, G, in the individual.

By evolution, one of the multiple trees learns to return true for a data in the positive cases, and all trees learn to return false for negative cases. Moreover, agents are allotted to the respective rules according to the adopted frequency, and the allotment to a rule with more misrecognition is restrained. Therefore, the rule with more agents is the typical and reliable rule, and the rule with less agents is the exceptional rule for the rare case.

The following points are, regarded as advantages of ADG.

– ADG enables us to extract rules for exceptional data which is likely to be missed by a single rule.
– It is easy to judge by the number of agents whether the acquired rules are typical rules or exceptional rules.
– It is easy to understand the acquired rules, because typical rules and exceptional rules are clearly separated.

Table 3 shows the GP functional and terminal symbols. We impose constraints on the combination of these symbols, such as Strongly Typed Genetic Programming (Haynes *et al.*, 1995). Terminal symbols do not enter directly in the arguments of the and function. Attribute items such as head_shape enter only in arg 0 of eq and not. Attribute values $(1, \ldots, 4)$ enter only in arg 1. Crossovers and mutations that break the constraints are not performed. The example rule mentioned in the beginning of this section is represented by the following symbols:

```
(and(eq head_shape 1)(and (eq is_smiling 2)
(not holding 1))).
```

The parameter settings of ADG are as follows: population size is 500, crossover rate is 0.9, mutation rate per individual is 0.95, group mutation rate is 0.01, and number of agents is 50. The respective weights in Equation 1 are $\alpha = 1.0, \beta = 0.0001, \delta = 0.01$, and $\gamma = 1.0001$.

4.2.1. *Experimental result*

We performed an experiment with the settings mentioned above. Fig. 5 shows the best fitness by generation. As an experimental result, we obtain a classification accuracy of 100.0% for both the training data set and the test data set. In the last generation, 50 agents in the best individual are divided into four groups. We show the acquired tree structural programs in the best individual that correspond to the classification rules. Rules are arranged according to the number of agents that support each rule. Moreover, the rules shown below are the ones from which obviously redundant terms such as (not has_tie 4) were pruned.

```
Rule 1 (23 Agents): (eq jacket_color 1)
Rule 2 (11 Agents): (and (eq head_shape 3)
                    (eq head_shape 3))
Rule 3 (10 Agents): (and (eq head_shape 2)
                    (and (eq body_shape 2)
                    (not body_shape 1)))
Rule 4 (6 Agents):  (and (eq head_shape 1)
                    (and (not body_shape 2)
                    (not body_shape 3)))
```

In the setting of this experiment, the comparison between head_shape and body_shape cannot be performed directly. Therefore, three rules are needed to represent the concept (head_shape = body_shape) using the three possible values.

Fig. 6 shows the change of the average number of groups in the population. The number of groups corresponds to the number of extracted rules. In the initial population, each individual is assigned a random group structure. So, the number of groups is about nine at the initial generation. We can see from this figure that individuals are optimized as the number of necessary rules is determined.

Moreover, we examined which rule's output is adopted for the 62 positive data in the training set. The counts of adoption are 29, 13, 12, and 8 for Rules 1–4, respectively. That is, the occurrence probabilities of each respective data type are 46.8, 21.0, 19.4 and 12.9%. On the other hand, the ratio of the agents distributed to each group became 46.0, 22.0, 20.0, and 12.0%, respectively. This result almost corresponds to the ratio of the data appearance. Therefore, we can easily understand the occurrence probability of each

Table 3. GP functions and terminals

Symbol	*# args*	*Functions*
and	2	arg0 $\wedge$ argl
eq	2	if (arg0 = argl) return T else return F
not	2	if (arg0 $\neq$ argl) return T else return F
head_shape, ..., has_tie	0	attribute
1, 2, 3, 4	0	attribute value

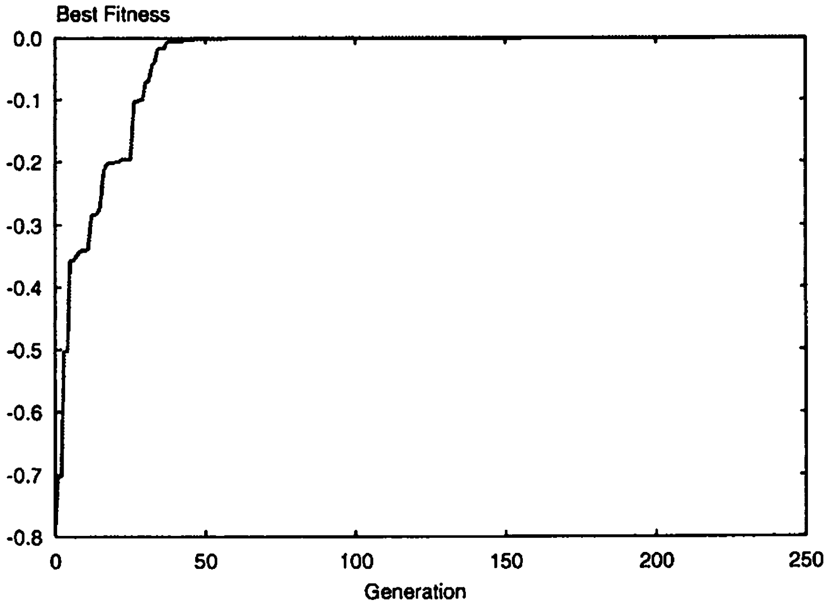

Fig. 5. Change of the best fitness for the Monk problem.

rule by the number of agents. In addition, if a group has few agents the group tends to vanish through evolution. So, this mechanism is effective in inhibiting the generation of unnecessary groups and finding valid rules.

Thus, we can confirm the effectiveness of this method by a preliminary experiment. Next, we apply this method to real medical data.

5. Extracting rules from medical database using ADG

5.1. *Coronary heart disease database*

In this section, we apply the proposed method to knowledge acquisition from medical data and develop a diagnostic system. In the field of medicine, a lot of patient data have been accumulated in databases. For example, the attributes are patients' conditions, diagnoses, and so on. It is important to diagnose appropriately without overlooking exceptional patients. We consider the solution to this problem to be a key for the achievement of high-quality medical treatment. Extracting multiple rules by using our proposed method will enable us to realize this achievement. We intend not only to improve the prediction accuracy but also to acquire useful knowledge by analyzing the acquired diagnostic rules. We also intend to acquire useful knowledge or medicine by analyzing the acquired group structure and rules in this experiment.

In this experiment, we use the database of coronary heart diseases (Suka *et al.*, 2004). Suka *et al.* (2004) developed this coronary heart disease database for the purpose of evaluating prognostic systems. The use of this database will enable researchers to discuss the advantages and disadvantages of different techniques.

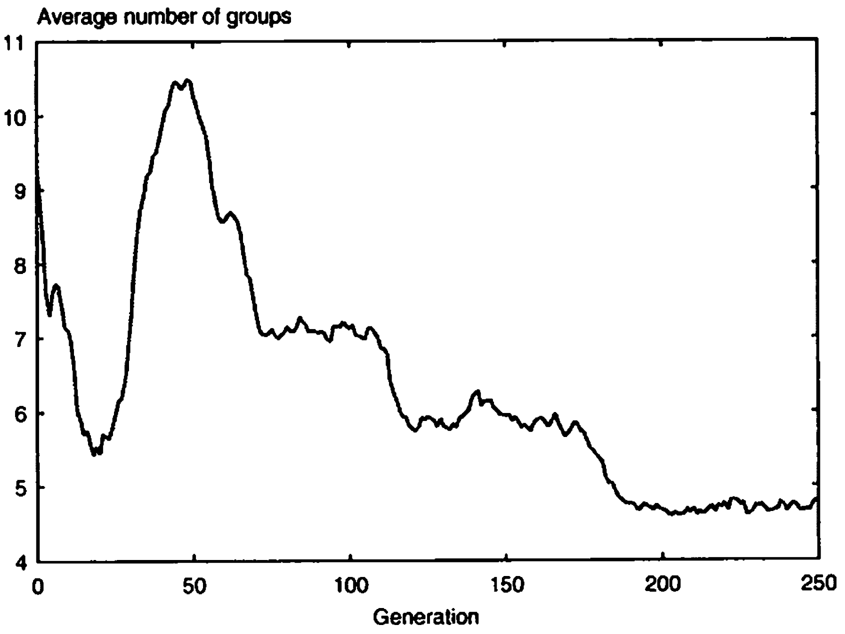

Fig. 6. Change of the average number of groups for the Monk problem.

Data in the coronary heart disease database are divided into two classes: non-coronary heart disease cases (non-CHD) and coronary heart disease cases (CHD). Each patient's disorder is diagnosed according to the results of eight test items, shown in Table 4. Examples of the data are shown in Table 5. In this research, we construct a diagnostic system which can classify data into the appropriate class based on these eight tests. The database consists of four training data sets (Train_A, Train_X, Train_Y, and Train_Z) and one testing data set (Test). The four training data sets are designed to have different numbers of total records or different proportions of CHD cases to non-CHD cases, as shown in Table 6.

The original results of some test items are provided as real values with various ranges. So, we normalize each value. We find the maximum value (*max*) and minimum value (*min*) of each item in the training data set, and the *i*th item's value x_i is normalized to X_i as follows:

$$X_i = (x_i - min_i)/(max_i - min_i).$$

5.2. *Applying ADG to the coronary heart disease database*

In order to judge whether each data is regarded as a CHD case, we need to find logical expressions which, only the data in CHD cases should satisfy. The following expression is an example.

Rule for CHD:$(TC > 0.51) \wedge (TC < 0.68) \wedge (DBP > 0.49)$.

In this case, the logical expression has to return false for non-CHD cases.

In coronary heart diseases, at present, there are no clear rules for judgments based on biochemical tests. In the medical field, the diagnoses are largely dependent on each doctor's experience. Therefore, the diagnostic rule is not necessarily represented by a single rule. Moreover, some data can be classified into different results, even if the results of the tests are the same. This database contains such inconsistent data. We apply ADG to the diagnoses of coronary heart diseases with consideration of this background.

Multiple trees in an individual of ADG represent the respective logical expressions. Each data in the training set is input to all trees in the individual. As illustrated by Data 2 in Fig. 7, the input data is regarded as a CHD case if one or more logical expressions in the individual return true. In contrast, as illustrated by Data 1 in Fig. 7, the input data is not regarded as a CHD case if all logical expressions in the individual return false. Individuals are optimized so that one in the multiple tree programs can return true

Table 4. Data items of the coronary heart disease database

Data item	*Symbol*	*value*
ID	ID	sequential number
Development of CHD	CHD	0=No; 1=Yes
Cholesterol	TC	continuous value (mg/dl)
Systolic Blood Pressure	SBP	continuous value (mmHg)
Diastolic Blood Pressure	DBP	continuous value (mmHg)
Left Ventricular Hypertrophy	LVH	0=None; l=Definite or Possible
National Origin	ORIGIN	0=Native-born; 1=Foreign-born
Education	EDUCATE	0=Grade School; 1=High School, not graduate; 2=High School, graduate; 3=Co1 1ege
Tobacco	TOBACCO	0=Never; 1=Stopped; 2=Cigars or Pipes; 3=Cigarettes (<20/day); 4=Cigarettes(≥20/day)
Alcohol	ALCOHOL	continuous value (oz/mo)

Table 5. Examples of data in the coronary heart disease database

Case	*TC*	*SBP*	*DBP*	*LVH*	*ORIGIN*	*EDUCATE*	*TOBACCO*	*ALCOHOL*
CHD	217	129	97	0	0	0	2	16.6
CHD	215	207	110	0	1	0	4	8.4
non-CHD	236	142	92	0	1	2	4	13.9
non-CHD	176	124	117	0	0	3	2	27.2

Table 6. Training and testing data sets

Data sets	*#records*	*#CHD*	*#non-CHD*	*CHD: non-CHD*
Train_A	13000	6500	6500	1 : 1
Train_X	19500	6500	13000	1 : 2
Train_Y	65000	6500	58500	1 : 9
Train_Z	4000	400	3600	1 : 9
Test	13000	6500	6500	1 : 1

for CHD cases and all trees can return false for non-CHD cases.

The load of each agent is calculated from the adopted frequency of each group's rule and the number of agents in each group. The adopted frequency of each rule is counted when the rule successfully returns true for each CHD data. As illustrated by Data 3 in Fig. 7, if multiple trees return true for a CHD data, the tree with more agents is adopted. When the k-th agent belongs to group g, the load of the agent is defined as follows:

$$W_k = \frac{(\text{adopted frequency of } g) \times N_{agent}}{(\text{Number of agents which belong to } g) \times N_{all\_adoption}}.$$

The fitness f is calculated by the following equation, as in the Monk problem. We maximize f by evolution.

$$f = -\frac{miss\_target\_data}{N_{CHD}} - \alpha\frac{misrecognition}{N_{non-CHD}} - \beta\frac{\sum_{N_{non-CHD}} fault\_agent}{misrecognition \times N_{agent}} - \delta V_w.$$

In this equation, N_{CHD} and $N_{non-CHD}$ represent the number of CHD cases and non-CHD cases in the database, respectively. *miss_target_data* is the number of missing data in the target CHD data that should have been judged to be true. *misrecognition* is the number of mistakes through which non-CHD data is regarded as a CHD case. When the rule returns true for non-CHD data, *fault_agent* is the number of agents who support the wrong rule in each data. V_w is the variance of every agent's load. In addition, in order to inhibit the redundant division of groups, f is multiplied by

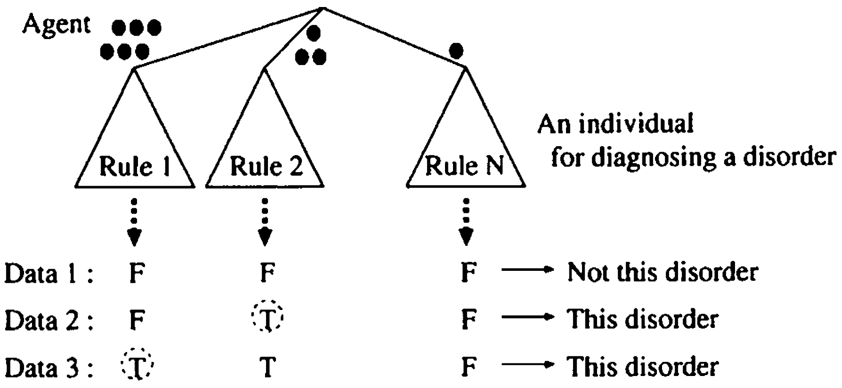
Agent
Rule 1
Rule 2
Rule N
An individual for diagnosing a disorder
Data 1 : F F F → Not this disorder
Data 2 : F T F → This disorder
Data 3 : T T F → This disorder

Fig. 7. Diagnostic system for a particular disorder.

$\gamma^{G-1}(\gamma > 1)$ according to the increase of the number of groups, G, in the individual.

Table 7 shows the GP functional and terminal symbols. We impose constraints on the combination of these symbols, as in the Monk problem. Test items such as TC enter only in arg 0 of gt and lt. Real values enter only in arg 1. The example rule mentioned in the beginning of this section is represented by the following symbols:

```
(and (gt TC 0.51) (and (lt TC 0.68)
                  (gt DBP 0.49))).
```

The parameter settings of ADG are as follows: population size is 500, crossover rate is 0.9, mutation rate per individual is 0.95, group mutation rate is 0.04, and number of agents is 50. The respective weights in Equation 2 are $\alpha = 1.0, \beta = 0.0001, \delta = 0.01$, and $\gamma = 1.001$.

5.3. *Experimental results*

In this section, ADG is applied to the training data so that only CHD cases can satisfy the rules. We describe the detail of an experiment using Train_Z, which consists 400 CHD cases and 3600 non-CHD cases.

Fig. 8 shows the best fitness, and Fig. 9 shows the average group number by generation. We can see from these figures that individuals are optimized as the number of necessary rules is searched for and determined.

As a result, 50 agents in the best individual are divided into 12 groups. We show the acquired rules in the best individual. Rules are arranged according to the number of agents that support each rule, and each terminal real value is transformed to the original range. The rules with more agents are frequently adopted rules. The rules with fewer agents are rules for exceptional data.

Rule 1 (19 Agents): $(\mathrm{SBP} > 179)$

Rule 2 (7 Agents): $(\mathrm{LVH} = 1)$

Rule 3 (6 Agents): $(\mathrm{TC} > 199) \wedge (\mathrm{SBP} > 141) \wedge (\mathrm{DBP} > 99) \wedge (\mathrm{DBP} < 112) \wedge (\mathrm{LVH} = 0) \wedge (\mathrm{EDUCATE} < 3) \wedge (\mathrm{ALCOHOL} < 34.54)$

Rule 4 (6 Agents): $(\mathrm{TC} > 264) \wedge (\mathrm{SBP} > 150) \wedge (\mathrm{TOBACCO} > 1) \wedge (\mathrm{ALCOHOL} < 44{:}9)$

Rule 5 (2 Agents): $(\mathrm{TC} > 168) \wedge (\mathrm{TC} < 252) \wedge (\mathrm{SBP} > 127) \wedge (\mathrm{DBP} > 106) \wedge (\mathrm{TOBACCO} > 2) \wedge (\mathrm{ALCOHOL} > 19.0)$

Rule 6 (2 Agents): $(\mathrm{TC} > 310)$

Rule 7 (2 Agents): $(\mathrm{SBP} > 141) \wedge (\mathrm{DBP} > 104) \wedge (\mathrm{LVH} = 0) \wedge (\mathrm{EDUCATE} < 2) \wedge (\mathrm{TOBACCO} > 0) \wedge (\mathrm{TOBACCO} < 3)$

Rule 8 (2 Agents): $(\mathrm{TC} > 242) \wedge (\mathrm{TC} < 296) \wedge (\mathrm{DBP} > 109) \wedge (\mathrm{ORIGIN} = 1) \wedge (\mathrm{TOBACCO} > 0) \wedge (\mathrm{ALCOHOL} > 15{:}9)$

Rule 9 (1 Agent): $(\mathrm{TC} > 214) \wedge (\mathrm{SBP} > 152) \wedge (\mathrm{DBP} > 85) \wedge (\mathrm{EDUCATE} < 1) \wedge (\mathrm{TOBACCO} < 2)$

Rule 10 (1 Agent): $(\mathrm{DBP} > 79) \wedge (\mathrm{DBP} < 84) \wedge (\mathrm{ALCOHOL} > 37.5)$

Rule 11 (1 Agent): $(\mathrm{TC} > 233) \wedge (\mathrm{SBP} > 160) \wedge (\mathrm{DBP} > 98) \wedge (\mathrm{DBP} < 132) \wedge (\mathrm{ORIGIN}{=}0) \wedge (\mathrm{EDUCATE}{<}3) \wedge (\mathrm{ALCOHOL}{<}35.1)$

Rule 12 (1 Agent): $(\mathrm{TC} > 186) \wedge (\mathrm{TC} < 330) \wedge (\mathrm{SBP} > 169) \wedge (\mathrm{DBP} > 99) \wedge (\mathrm{DBP} < 114) \wedge (\mathrm{LVH} = 0) \wedge (\mathrm{TOBACCO} > 0) \wedge (\mathrm{TOBACCO} < 3) \wedge (\mathrm{ALCOHOL} < 34.5)$

The judgment accuracy for 4000 training data is as follows. One or more rules return true for

Table 7. GP Functions and terminals

Symbol	*#args*	*Functions*
and	2	arg 0 $\wedge$ arg 1
gt	2	if (arg 0 > arg 1) return T else return F
lt	2	if (arg 0 < arg 1) return T else return F
TC, SBP, ..., ALCOHOL	0	normalized test value
0.0 – 1.0	0	real value

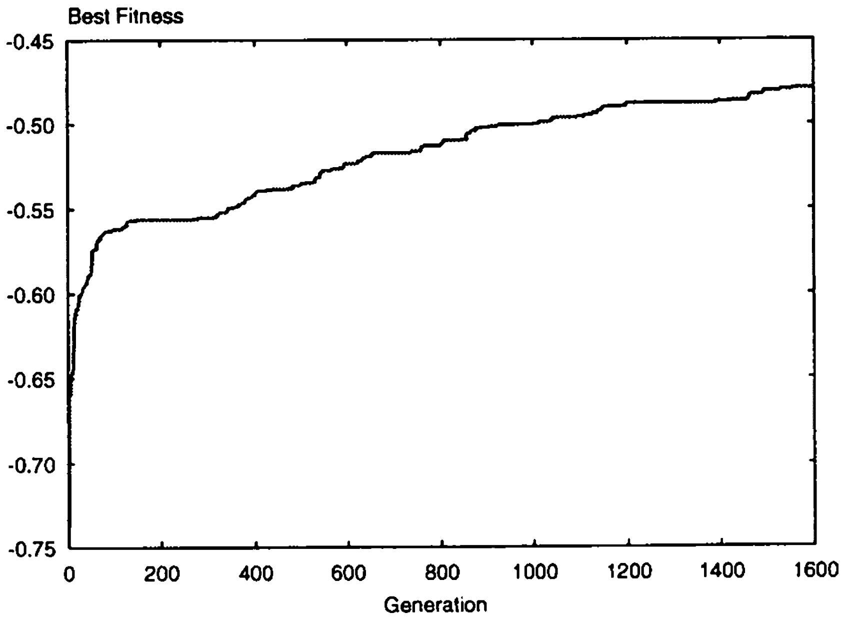

Fig. 8. Change of the best fitness.

308 of 400 CHD cases, and all rules successfully return false for 2691 of 3600 non-CHD cases. The recognition rate of the training data is defined as follows.

$$\left(1 - \frac{miss\_target\_data + misrecognition}{N_{CHD} + N_{non\text{-}CHD}}\right) \times 100.0.$$

Therefore, the recognition rate of the training data is 75.0%.

We examined which rule's output is adopted for the 308 successful data. The counts of adoption of these twelve rules are 115, 46, 38, 36, 16, 13, 12, 10, 9, 7, 4, and 2, respectively. These data result from the effects of the third and fourth terms of the fitness equation. The ratio of adopted frequencies of the respective rules does not completely correspond to the ratio of agents in each group, because there is a requirement to reduce the number of agents who support the rule with misrecognition data. However, the rule with more agents tends to have a higher adopted frequency. Both typical rules for frequent cases and exceptional rules for rare cases were extracted successfully. Moreover, this system was applied to 13000 test data. As a result, it succeeded in the classification of 8655 cases. The recognition rate was 66.6%.

We also applied this method to other training data sets (Train_A, X, Y), and examined the performance of each result for both training and test data. Table 8 shows the recognition rates. The parenthetic values in the table indicate the recognition rates for the test data set. The acquired rules are represented by simple logical expressions. So, we can easily acquire diagnostic knowledge from the rules. However, the constraints of the expressions may have a adverse influence on the recognition rate. By modifying the GP sym-

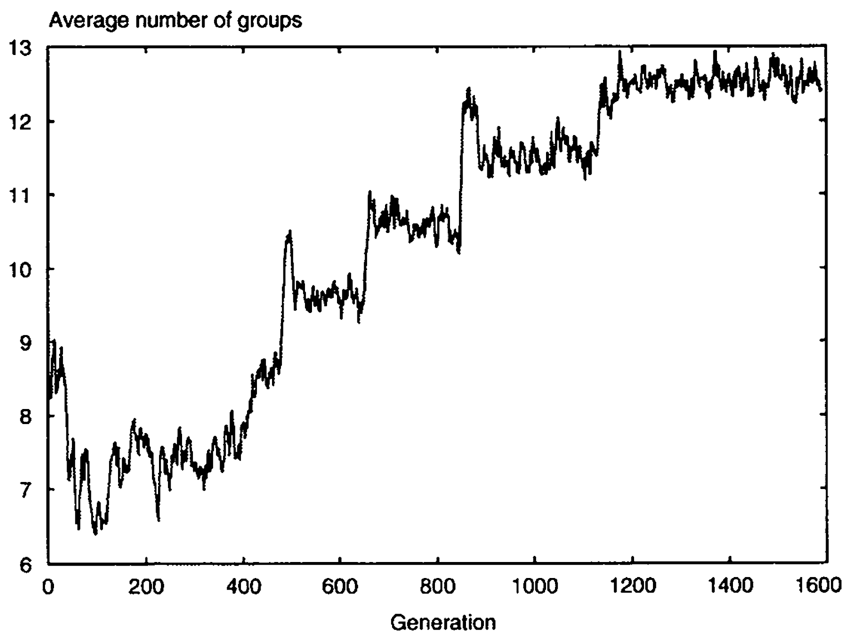

Fig. 9. Change of the average number of groups.

bols so that the rules can represent more complex expressions (e.g., DBP > 1.2 SBP), we can improve the recognition rate while keeping the comprehensibility.

6. Conclusions and future work

In this research, we treated data containing multiple rules. We proposed a new method using ADG for the purpose of the extraction of multiple rules. In this method, the clustering of data and the rule extraction in each cluster are performed simultaneously by multi-agent cooperation. The grouping of agents and the allotment of data to each group are optimized automatically by evolutionary computation. We showed the effectiveness of this method by an application using medical data.

The values of parameters such as the redundant group penalty affect the judgment whether the system dispenses with a new rule for a certain data or creates a new rule. We need to examine how to set these parameters. Therefore, we are now planning to verify the effectiveness of this method from the viewpoint of information criteria.

The diagnostic rules were extracted mechanically from only numerical data. Some rules may be not accepted easily the field of medicine because they may include incongruous combinations of items beside the common sense of doctors. By taking the knowledge of medical treatment into account during the process of optimization, more effective rules can be acquired. The optimization by ADG using such knowledge is planned for future work. In addition, we need to investigate the usefulness of extracted rules from the viewpoint of health care.

Table 8. Recognition rates

Data set	*Recognition rate*
Train_A	70.0% (67.8%)
Train_X	70.2% (68.5%)
Train_Y	70.1% (68.6%)
Train_Z	75.0% (66.6%)

Acknowledgments

This work was supported by a Grant-in-Aid for Young Scientists (B) (No. 15700199) from the Ministry of Education, Culture, Sports, Science and Technology of Japan. In addition, this was also supported by a Hiroshima City University

Grant for Special Academic Research (General Study).

References

Goldberg, D. E. (1989) *Genetic Algorithms in search, optimization, and machine learning*, Addison Wesley, Reading, MA.

Hara, A. and Nagao, T. (1999) Emergence of cooperative behavior using ADG; Automatically Defined Groups. *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, pp. 1039–1046.

Hara, A. and Nagao, T. (2002) Construction and analysis of stock market model using ADG; automatically defined groups. *International Journal of Computational Intelligence and Applications (IJ-CIA)*, **2**(4), 433–446.

Hara, A., Ichimura, T., Takahama T. and Isomichi, Y. (2003) Extraction of rules by heterogeneous agents using automatically defined groups. *Proceedings of the Seventh conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'2003)*, **2**, 1405–1411.

Hara, A., Ichimura, T., Takahama, T. and Isomichi, Y. (2004) Discovery of cluster structure and the clustering rules from medical database using ADG; automatically defined groups, in *Knowledge-Based Intelligent Systems for Healthcare*, Ichimura T. and Yoshida K. (eds) pp. 51–86.

Haynes, T., Wainwright, R., Sen, S. and Schoenefeld, D. (1995) Strongly typed genetic programming in evolving cooperation strategies. *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pp. 271–278.

Holland, J. H. (1995) *Adaptation in Natural and Artificial Systems*, The Univ. Michigan Press.

Iba, H. (1996) Emergent cooperation for multiple agents using genetic programming. *Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation*, pp. 32–41.

Iba, H. (1997) Multiple-agent learning for a robot navigation task by genetic programming. *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 195–200.

Ichimura, T., Tazaki E. and Yoshida K.(1995) Extraction of fuzzy rules using neural networks with structure level adaptation – verification to the diagnosis of hepatobiliary disorders. *International Journal of Bio-Medical Computing*, **40** (2), 139–146.

Ichimura, T., Oeda, S., Mackin, K. J., Yamashita, T. and Yoshida, K. (2001) Darwinian inheritance genetic learning method of neural networks under dynamic environment and its application to diagnostic system for hepatobiliary disorders. *Journal of the Biomedical Fuzzy Systems and Human Sciences*, **7**, 19–26.

Ichimura, T., Oeda, S., Suka, M. and Yoshida, K. (2003a) A classification capability of reflective neural networks in medical databases. *Proceedings of the Seventh Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'2003)*, **2**, 373–379.

Ichimura, T., Oeda, S., Yamashita, T. and Yoshida, K. (2003b) A classification method of medical database by immune multi-agent neural networks with planar lattice architecture. *Proceeding of the Seventh conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'2003)*, **2**, 380–387.

Koza, J. R. (1992) *Genetic Programming – On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge.

Luke, S. and Spector, L. (1996) Evolving teamwork and coordination with genetic programming. *Genetic Programming 1996: Proceedings of the First Annual Conference*, pp. 150–156.

Soule, T. (1999) Voting Teams: A cooperative approach to non-typical problems using genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, pp. 916–922.

Soule, T. (2000) Heterogeneity and specialization in evolving teams. *Proceedings of the Genetic and Evolutionary Computation Conference 2000*, pp. 778–785.

Suka, M., Ichimura, T. and Yoshida, K. (2004) Development of Coronary Heart Disease Database. *Proceedings of the Eighth Conference of Knowledge-Based Intelligent Information and Engineering Systems (KES' 2004)*, 2004.

Thrun, S. B. *et al.* (1991) The MONK's problems: a performance comparison of different learning algorithms. *Technical Report CMU-CS-91-197*, Carnegie Mellon University.