Reconsideration of the Effectiveness on Extracting Computer Diagnostic Rules by Automatically Defined Groups

Yoshiaki Kurosawa, Akira Hara, Kazuya Mera, and Takumi Ichimura

Graduate School of Information Sciences, Hiroshima City University,
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima, Japan
{kurosawa,ahara,mera,ichimura}@its.hiroshima-cu.ac.jp

Abstract. Our aim is to manage computer systems without expert knowledge. We have proposed a method of diagnostic rule extraction from log files by using Automatically Defined Groups (ADG) based on Genetic Programming. However, this work less explained the effectiveness, especially, the characteristics of the acquired rules. Therefore, we re-evaluated the effectiveness by performing two experiments: the use of artificial log files and the use of real log files. As a result, we confirmed that ADG could acquire the rules composed of multiple terms. This characteristic is very important because we can judge the message that we must consider the co-occurrence of the words, i.e. 'Error' and 'not'. Thus, we conclude that the ADG is effective for the diagnosis of the systems.

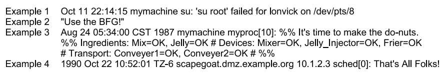
Keywords: Genetic Programming, Rule Extraction, Data Mining.

1 Introduction

Recently, computer systems have increased in size. With this increase of them connecting with Local Area Network (LAN), many TCP/IP services such as http service have been offered for the system users. The administration of the systems, therefore, have become more complex and difficult to detect some errors.

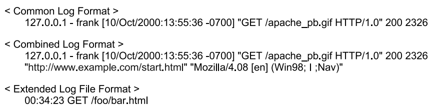
Several studies have been made to automatically analyze system log files in order to decrease such administrators' duty (cf. [1,2,3]). However, the main purpose of these studies was to diagnose the problem of the systems, by focusing of only one service (http), file (/var/log/messages), or protocol (simple network management protocol), based on some expert knowledge that the researchers manually analyzed in detail. Certainly, this aim is useful, but the real purpose of the administrators is not to focus on only one service but to manage all their systems including many personal computers (PC), devices, and so on.

Based on the aim, we extended [4] an Automatically Defined Groups (ADG) based on Genetic Programming (GP) [5,6,7]. However, it was not clear whether this extended ADG could acquire rules composed of multiple terms. Therefore, we re-evaluated the effectiveness focusing on these points; 'Can acquire multiple rules?', 'Can the rules be ordered by the supported agents?', and 'Can the rules consist of multiple terms?'.



```
Example 1   Oct 11 22:14:15 mymachine su: 'su root' failed for lonvick on /dev/pts/8
Example 2   "Use the BFG!"
Example 3   Aug 24 05:34:00 CST 1987 mymachine myproc[10]: %% It's time to make the do-nuts.
            %% Ingredients: Mix=OK, Jelly=OK # Devices: Mixer=OK, Jelly_Injector=OK, Frier=OK
            # Transport: Conveyer1=OK, Conveyer2=OK # %%
Example 4   1990 Oct 22 10:52:01 TZ-6 scapegoat.dmz.example.org 10.1.2.3 sched[0]: That's All Folks!
```

Fig. 1. Sample Messages in RFC3164



```
< Common Log Format >
    127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

< Combined Log Format >
    127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
    "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"

< Extended Log File Format >
    00:34:23 GET /foo/bar.html
```

Fig. 2. Sample HTML Messages

2 Descriptions of System Log and Its Analysis

We do not simply deal with system log files because there are various formats. For example, RFC3164 proposed a syslog protocol [8]. According to the document, system log files describe messages generated by certain events on various types of sender such as computer. No concrete detail is described as this definition mentions, however. Fig.1 shows four valid sample logs described in RFC 3164. Moreover, we further exemplify the difference of the format; see Fig.2.

By comparing these samples, log files have no unified format to describe some events.We may automatically analyze even these types of format by using detailed and appropriate knowledge. However, such ambiguous format must make administrators difficult to maintain the systems in case of lack of appropriate knowledge. In next section, we will discuss this type of difficulty.

2.1 Problems as to Log Files When Managing Systems

According to the paper previosly mentioned [4], when administrators check their systems whether they work well or not, the administrators may be confronted with the following problems at least; 'lack of knowledge on system', 'enormous data size', and 'temporary lack of knowledge by change of some kind'. To solve them, simple translation method based on regular expressions was proposed.

In [4], some tags such as <DATE> and <TIME> were given to some fields because the fields can be easily understood. Furthermore, the other types of tags were given by using existing databases in the system. For example, tags such as <HOST> and <REMOTEHOST> relate to hostname. They can be distinguished each other by using appropriate commands related to the system database, e.g., "ifconfig." In the same way, by using the other command "id" or "groups," we can distinguish 'username' or 'groupname' with words appeared in the log messages. If the system can not understand the content of the fields, it gave <EXP> tag including explanation of some sort. Thus, their procedure is simple because of the realization without knowledge.

3 Extracting Rules Using Automatically Defined Groups

3.1 Concept of Automatically Defined Groups

In this section, we introduce the ADG method [4,5,6,7]. The method is based on evolutionary computing and can optimize both the grouping of agents and the program of each group in the process of evolution. By grouping multiple agents,

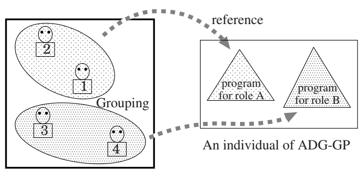


Fig. 3. Concept of ADG

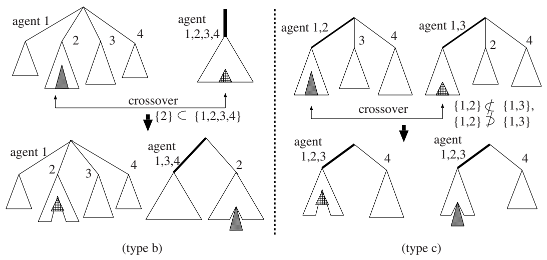


Fig. 4. Examples of crossover

we can easily analyze the behavior of agents. 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. (1)How many groups are required to solve the problem? , (2)Which group does each agent belong to? , and (3)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. 3. 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.

Generating an initial population, agents in each GP individual are divided into groups at random. 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. The crossover operations are shown in Fig. 4. In addition, the (type a) is ommited because the structure of each individual is unchanged.

3.2 Extracting Multiple Rules Using ADG

The ADG method can be utilized as a rule extraction method for classifying positive and negative cases in database. Respective trees in an individual of ADG represent the logical expressions, and return true or false for each data. If one or more trees return true for an input data, the data is regarded as a positive case. The fitness f is calculated by the following equation (1).

$$f = \frac{H_{Abnormal}/L_{Abnormal}}{H_{Normal}/L_{Normal}} - \beta \, \frac{\sum_{N_{Normal}} fault\_agent}{H_{Normal} \times N_{agent}} - \delta \; V_w \qquad (1)$$

In this equation, L means the number of line in log files. Thus, $L_{Abnormal}$ and L_{Normal} means the number of line in the abnormal/normal state respectively. H means that one or more trees in an individual return true for a log message. When the rule returns true for data in the normal state log file, $fault\_agent$, that is, the number of agents who support the wrong rule, was counted. Thus, the third term represents the average rate of agents who support the wrong rules when its misrecognition happens. Moreover, V_w is the variance of every agent's load, which is introduced from the viewpoint of load balancing among agents.

The concept of each agent's load arises from the viewpoint of 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 returns true to the messages in abnormal state log. If multiple trees return true for a message, the frequency of the tree with more agents is counted. When the agent a belongs to the group g, the load of the agent w_a is defined as follows:

$$w_a = f_g / n^g_{agent} \tag{2}$$

where n^g_{agent} represents the number of agents which belong the group g, and f_g represents the adopted frequency of g.

In addition, in order to inhibit the redundant division of groups, f is multiplied by γ^{G-1} according to the increase of the number of groups, G, in the individual. γ is a penalty coefficient on the number of groups.

Last, we explain their logical tree in their paper.

(`include <SORT> unexpected`) $\wedge$ (`include <EXP> warning`)

The second argument of each term such as <SORT> and <EXP> represents the attached tag. If the message enclosed in the tag includes the word specified by the third argument, this expression returns true. In addition, the logical expression has to return false for any logs in normal state.

3.3 System State Pattern for Supervised Information

In general, when machine learning method is performed, some sort of supervised information is needed. However, using 'System State Pattern (SSP)', we can deal with log data without supervised information [4]. The SSP means a kind of state transition of the system: from normal state to abnormal one, or from abnormal state to normal one. By comparing both states, we may find some differences. That is to say, this SSP approach is based on the heuristics of an idea that error messages will appear in the only abnormal log, and is adopt to extract rules without supervised information.

4 Experiments

We perform two experiments to achieve the aim mentioned above. One is an experiment (Experiment 1) on the basis of artificial log files, which is performed to check whether the ADG can extract rules composed of multiple terms. The other is an experiment (Experiment 2) by adopting log files stored from actual computers.

4.1 Experiment 1: Rule Extraction from Artificial Logs

In this paper, the artificial log files are made by the following procedure. At first, we assume the situation where we must manage five hosts having the same two log files. Each file includes 300 messages, that is, 300 lines. In addition, these 10 log files (two files by five hosts) are exactly the same. Each message consists of five kinds of tag-and-word set; <LOGNAME>, <TIME>, <HOST>, <REMHOST>, and <EXP>. The number of word list are 1, 4, 5, 5, and 17, and each word is arbitrarily selected from actual log files except for <HOST> and <REMHOST>.

Then, one of two files mentioned above is treated as normal, and the other is treated as abnormal. At this point, we change some tag-and-word sets only in the certain abnormal log files according to the following two steps. That is, this is the case that system errors occur on purpose.

Step 1. We arbitrarily select two hosts, i.e. host1 and host2. Then, we can change some words including a tag <REMHOST> only in the abnormal log files of the selected hosts. For example, we replace the word 'host1' and 'host2' with 'host5' including the tag. These replacements means that some errors occur when sending data from 'host1' to 'host5' or sending data from 'host2' to 'host5'. In addition, we change the tag once regarding 'host1', and three times regarding 'host2'. Thus, this means that the number of the error related to 'host1' is three times larger than the one related to 'host2'.

Step 2. We replace an arbitrary word in the <EXP> with the same word mentioned at (Step 1), 'host5', in both normal/abnormal files. In this paper, we perform this replacement five times. The replacement makes it difficult that we find errors by focusing on the word frequency because the files are almost equivalent.

Therefore, these log files are assumed that there are five hosts, and two of them encounter certain accidents.

We applied ADG to these artficial log message data. In this paper, GP functions and terminals are shown in Table 1.

Fig. 5 illustrates the change of the average number of groups. Because each group refer to one diagnostic rule, the number of groups corresponds to the number of the extracted rules. These agents belonging to approximately four groups mainly seek four rules for detecting system errors. Particularly in the best individual, two rules were extracted. The two extracted rules are shown in Fig. 7.

The first two lines means a rule to check whether <HOST> includes 'host1' and <REMHOST> includes 'host5', respectively. We clearly understand that this extracted IF-THEN rule can detect the error related to 'host5', made by the two steps mentioned above. In addition, these two terms are minimum, and this rule composed of the terms is the best one in order to detect the errors. If this type of error should not be detect by focusing on the frequency of the host 'host5' because this host name is even distribution in the log files. On the other hand, our proposed method can detect this type of error. Therefore, we

Table 1. GP Functions and Terminals

symbol	#args	functions
`and`	2	arg0 $\wedge$ arg1
`include`	2	if arg0 (Tag) includes arg1 (Word), return T else return F
`<HOST>,<EXP>, ...`	0	Tag name
`0,... ,N-1`	0	selected number in word list

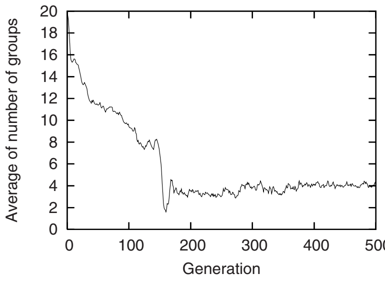


Fig. 5. Change of the average of the number of groups : Experiment 1

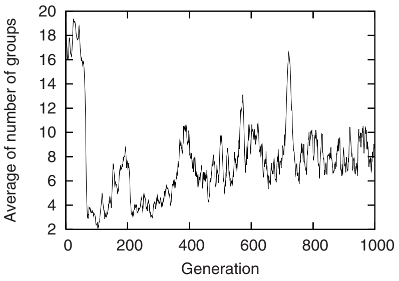


Fig. 6. Change of the average of the number of groups : Experiment 2

conclude that the method is effective to extract diagnostic rules from log files even including co-occurrence relations such as "error" and "not find."

Next, the second rule in Fig. 7 is complicated because the rule consists of many terms, and the same term exists in the rule: "(`include <HOST> host2`)." This rule, in a sense, is redundant because this type of error can be detect by only two terms as we previously mentioned. However, no misdetection could occur by using this rule. Therefore, all these terms work to identify the messages only in the abnormal message.

Last, we take the number of agents which support the rules into account. The support number of the first rule is 50, and the one of the second rule is 150. This means that all 200 agents is appropriately divided into two groups according to the error rate between 'host1' and 'host2' mentioned at Step 1.

4.2 Experiment 2: Rule Extraction from Actual System Logs

In Experiment 2, we apply ADG to the actual log message data. Our 322 target files, such as syslog files and html logs, were collected on a server. These files consisted of 90,762 normal state lines, and 88,336 abnormal state lines. The files differed from the ones collected in our previous work [4]. In order to describe these log files, we needed 25 tags as well as the previous work. However, maximum word list size N was 935, and the word list with max size belonged to the <EXP> tag. The respective weights in equation (1) and the other parameters were the same in Experiment 1.

```
( and ( include <HOST>  host1 )
      ( include <REMHOST>  host5 ) )

( and
   ( and
      ( include <REMHOST>  host5 )
      ( and ( include <EXP>  1.4.1 )
            ( include <HOST>  host2 ) ) )
   ( and
      ( include <EXP>  syslogd )
      ( include <HOST>  host2 ) ) )
```

Fig. 7. Extracted Rules: Ex.1

```
( and ( and ( include HOST mysvr )
   ...       ( include EXP bond0 ) ) )

( include EXP smbd/service.c )
                  ...
( include EXP '***.***.net/AAAA/IN' )
                  ...
( include EXP '***.***.jp/AAAA/IN' )
                  ...
```

Fig. 8. Extracted Rules: Ex.2

```
<HOST>mysvr</HOST> <LOGNAME>messages</LOGNAME>
  <DATE>2006/??/??</DATE> <TIME>11:13:56</TIME>
  <DAEMON>dhcpd</DAEMON>
  <EXP>DHCPREQUEST for ***.***.***.xxx (***.***.***.yyy)
     from **:**:**:**:**:** via bond0</EXP>

<HOST>pc***</HOST> <LOGNAME>******.log</LOGNAME>
  <DATE>2006/??/??</DATE> <TIME>16:57:05</TIME>
  <EXP>smbd/service.c:make_connection_snum(***)</EXP>
  <EXP>pc*** (***.***.***.zzz) closed connection to service ...</EXP>

<HOST>mysvr</HOST> <LOGNAME>messages</LOGNAME>
  <DATE>2006/??/??</DATE> <TIME>13:52:20</TIME>
  <DAEMON>named</DAEMON> <EXP>unexpected RCODE
   (SERVFAIL) resolving '***.***.jp/AAAA/IN': ***.***.***.***#53</EXP>
```

Fig. 9. Extracted Logs: Ex.2

Fig. 6 illustrates the change of the average number of groups[1]. These agents in each individual seem to be divided into approximately eight groups although these numbers fluctuate greatly. In the best individual, 200 agents were divided into 17 groups, that is, 17 rules were extracted. By using these rules, we tried to detect errors from the same log files, the abnormal state files, which were used when extracting the rules, that is, as closed testing procedure. As a result, we can detect 16,877 messages.

Fig. 8 shows a part of the acquired rules that correspond to the tree structured presentation in the best individual. These rules are arranged according to the number of agents. This figure illustrates the effectiveness of the ADG method because the first rule consists of multiple terms. In addition, this case has something to do with the 'DHCP' as shown in Fig. 9.

Both the third rule and fourth rule in Fig. 9 include "***.***.***/AAAA/IN," related to 'host name' in the tag <EXP>. This description is not to easy to understood at the first glance. However, by referring the description from the log files, we understand that the words have something to do with the DNS error because the message includes "unexpected RCODE(SERVFAIL)."

4.3 The Effectiveness of the ADG

We confirmed the following points from our experiments: (a)we could acquire multiple rules, (b)the rules could be ordered by the number of agents, and, (c)the rules could consist of multiple terms from both the artificial data and the real data.

These points are the ones which previous work less explained [7]. Especially, the third point is very important because the real log files include the message such as "Error was not found" and we can not detect whether the message is normal or abnormal without focusing on the co-occurrence of the word such as 'error' and 'not'. Therefore, the ADG method is effective in such case.

5 Conclusions and Future Work

We re-evaluated the ADG on the basis of evolutionary computation. As a result, we found 17 rules and one of them had two terms in Experiment 2. That is,

[1] We selected 1000 generations to unify the experimental condition in the paper [9].

we confirmed that we could extract multiple rules composed of multiple terms. Moreover, It is important that we could perform the extraction without any expert knowledge and supervised information compared with [1,2,3].

However, the optimization may not be sufficient in the real data as shown in Fig.6 because the number of tags (25) is relatively large and the number of terminals (935 words) is considerably large. Thus, the optimization such as the decrease of tags and terminals is planned for one of our future works.

In addition, we did not apply any natural language processing, e.g., morphological analysis, to log files. However, the files include various messages written by natural language, especially Japanese. Thus, we will adopt natural language processing techniques (i.e. [10,11]) as preprocessing, and apply to GP.

References

1. Andrews, J.H.: Theory and practice of log file analysis. Technical Report 524, Department of Computer Science, University of Western Ontario (1998)
2. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. Knowledge and Information Systems 1(1), 5–32 (1999)
3. Büchner, A.G., Baumgarten, M., Anand, S.S., Mulvenna, M.D., Hughes, J.G.: Navigation pattern discovery from internet data. In: Proc. of the Web Usage Analysis and User Profiling Workshop, pp. 25–30 (1999)
4. Kurosawa, Y., Hara, A., Ichimura, T., Kawano, Y.: Extraction of Error Detection Rules without Supervised Information from Log Files Using Automatically Defined Groups. In: Proc. of the IEEE International Conference on Systems, Man and Cybernetics (SMC2006), pp. 5314–5319 (2006)
5. Hara, A., Ichimura, T., Yoshida, K.: Discovering Multiple Diagnostic Rules from Coronary Heart Disease Database Using Automatically Defined Groups. International Journal of Manufacturing 16(6), 645–661 (2005)
6. Hara, A., Ichimura, T., Takahama, T., Isomichi, Y.: Discovery of Cluster Structure and The Clustering Rules from Medical Database Using ADG; Automatically Defined Groups. In: Ichimura, T., Yoshida, K. (eds.) Knowledge-Based Intelligent Systems for Healthcare, pp. 51–86 (2004)
7. Hara, A., Nagao, T.: Construction and analysis of stock market model using ADG; Automatically Defined Groups. International Journal of Computational Intelligence and Applications (IJCIA) 2(4), 433–446 (2002)
8. Lonvick, C.: The BSD Syslog Protocol, RFC3164 (August 2001)
9. Kurosawa, Y., Hara, A., Ichimura, T.: Preprocessing techniques for extracting computer diagnostic rules by ADG. In: Proc. of the IEEE Three-Rivers Workshop on Soft Computing in Industrial Applications (SMCia2007), IEEE Computer Society Press, Los Alamitos (to appear, 2007)
10. Kurosawa, Y., et al.: A description method of syntactic rules on filmscripts. Journal of Natural Language Processing (in Japanese) 12(6), 25–62 (2005)
11. Mera, K., Kurosawa, Y., Ichimura, T.: Emotion Oriented Interaction system for Elderly People. In: Ichimura, T., Yoshida, K. (eds.) Knowledge Based Intelligent Systems for Health Care, Advanced Knowledge International (2004)