Combination of Genetic Network Programming and Knapsack Problem to Support Record Clustering on Distributed Databases

Wirarama Wedashwara[a], Shingo Mabu[a,*], Masanao Obayashi[a], Takashi Kuremoto[a]

[a]*Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai 2-16-1, Ube, Yamaguchi, 755-8611, Japan*

Abstract

This research involves implementation of genetic network programming (GNP) and knapsack problem (KP) as a decision support system for record clustering in distributed databases. The objective is to distribute big data to certain sites with the limited amount of capacities by considering the similarity of distributed data in each site. GNP is used to extract rules from big data by considering characteristics (value ranges) of each attribute in a dataset. KP is used to distribute rules to each site by considering similarity (value) and data amount (weight) related to each rule to match the site capacities. From the simulation results, it is clarified that the proposed method shows some advantages over the conventional clustering algorithms.

Key words: Genetic Network Programming, Database Clustering, Knapsack Problem, Record Clustering

1 Introduction

Distributed database management system (DDBMS) could be a solution for large scale information systems with large amount of data growth and data

* Corresponding author. Tel.: +81-836-85-9519; Fax.: +81-836-85-9519
Email addresses: `t001we@yamaguchi-u.ac.jp` (Wirarama Wedashwara), `mabu@yamaguchi-u.ac.jp` (Shingo Mabu), `m.obayas@yamaguchi-u.ac.jp` (Masanao Obayashi), `wu@yamaguchi-u.ac.jp` (Takashi Kuremoto).

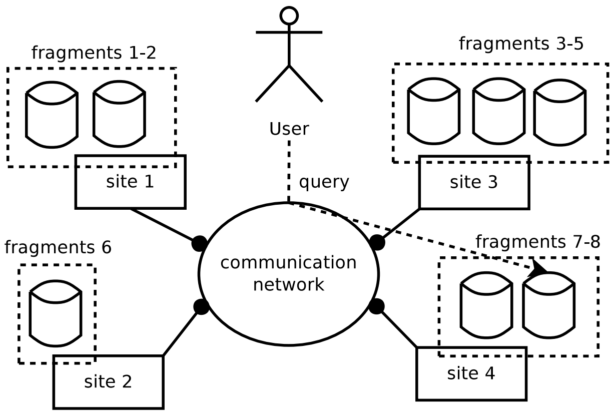

Fig. 1. A distributed database environment

accesses. A distributed database (DDB) is a collection of data that logically belongs to the same system but is spread over the sites of a computer network (Fig. 1). A DDBMS is then defined as a software system that permits the management of DDB and makes the distribution of data between databases and software transparent to the users (Bhuyar, Gawande, & Deshmukh, 2012; Zilio et al., 2004).

To handle the data proliferation, efficient access methods and data storage techniques have become increasingly critical to maintain an acceptable query response time. One way to improve query response time is to reduce the number of disk I/Os by clustering the database vertically (attribute clustering) and/or horizontally (record clustering) (Guinepain & Gruenwald, 2006, 2008). Improvements in the retrieval time of multi-attribute records can be attained if similar records are grouped close together in the file space as a result of restructuring. This is because fewer page transfers are required as the probability of two or more of the target records residing in the same page of storage is increased (Lowden & Kitsopanidis, 1993).

In this paper, a novel method combining genetic network programming (GNP) (Mabu, Chen, Lu, Shimada, & Hirasawa, 2011; Shimada, Hirasawa, & Hu, 2006) and knapsack problem (KP) (Singh, 2011; Lai, 2006) for record clustering is proposed. Hypothesis of this research are the implementation of GNP for data mining can create effective clusters from complicated datasets and KP can handle the problem of distributing fragments to several sites considering value (similarity of data) and mass (data size) in DDBMS. Therefore, it could be a solution to the fragment allocation and site storage capacity problems.

This paper is organized as follows. Section 2 describes a review of the proposed framework, section 3 describes the detailed algorithm of the proposed framework, section 4 shows the simulation results, and finally section 5 is devoted to conclusions.

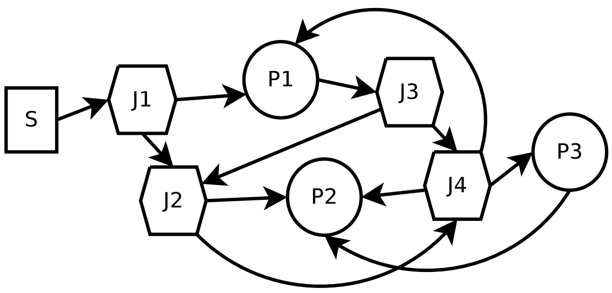

Fig. 2. Basic Implementation of GNP
S : start node, [J_1,...,J_4] : judgement node, [P_1,...,P_3] : processing node

2 Review of the Proposed Framework

2.1 Genetic Network Programming

GNP is an evolutionary optimization technique, which uses directed graph structures instead of strings in genetic algorithm (Holland, 1975) or trees in genetic programming (Koza, 1992), which leads to enhancing the representation ability with compact programs derived from the re-usability of nodes in a graph structure.

In GNP, nodes are interpreted as the minimum units of judgement and action, and node transition represents rules of the program. After starting the node transition from the start node, GNP does not return to the start node when the actions are completed. The next judgement and action are always influenced by the previous node transition. Judgement and processing of GNP programs are performed on the node level.

The basic structure of GNP is illustrated in Fig. 2, with S denoting the start node. Two other kinds of nodes, judgement nodes and processing nodes, have judgement function J_p and processing function P_q, respectively. J_p ($p = 1, \ldots, n$) denotes the p-th judgement function stored in a library for judgement nodes, while P_q ($q = 1, \ldots, m$) denotes the q-th processing function stored in a library for processing nodes (Mabu et al., 2011; Shimada et al., 2006).

In this research, GNP is used to handle rule extraction from datasets by analyzing the records. Each judgment node represents an attribute with value range. For example, price attribute could be divided into three ranges (low, middle, high), and one range is assigned to one judgment node. GNP makes rules by evolving combinations of nodes and measures the coverage of the extracted rules. Coverage means that how much records in a dataset each rule can represent (cover). Rules that cover at least one record will be stored in the rule pool, then in the KP phase, the stored rules are distributed to several sites. The point of this paper is to distribute rules, not the data, which contributes to distributing any data into the sites considering the similarities between rules and data. The detailed explanation of the implementation of GNP in rule extraction is available in section 3.1.

2.2 Knapsack Problem

KP is a combinational optimization problem dealing with a set of items, each with a mass and a value, determining the number of each item to include in a collection so that the total weight is less than or equal to the given limit and the total value is as large as possible. KP is defined as follows.

$$\text{maximize } S = \sum_{i=1}^{n} v_i x_i, \quad \text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq W, \tag{1}$$

where S = total value of the knapsack (site); i = fragment number ($1 \leq i \leq n$); x_i = the number of fragments i; v_i = value (similarity to the leader rule of the site) of fragment i; w_i = weight (data size) of fragment i; W = capacity of the site. By allowing each fragment (item) to be added more than once to sites, this optimization can handle the problem of replication (Zhao, Huang, Pang, & Liu, 2009; Singh, 2011).

Knapsack problem in this research is solved by standard dynamic programming for 0/1 knapsack problem (Toth, 1980). Let us define two dimensional array $m[i, w]$ with row i and column w. $m[i, w]$ shows the value of knapsack when considering items with item number $1, 2, \ldots, i-1, i$, and their total weight w. $m[i, w]$ is calculated by Eq. 2.

$$\begin{aligned} &m[i,\, w] = m[i-1,\, w] \ \text{ if } \ w_i > W \\ &m[i,\, w] = \max(m[i-1,\, w],\, m[i-1, w - w_i] + v_i) \ \text{ if } \ w_i \leqslant W. \end{aligned} \tag{2}$$

The first step is to calculate $m[0, w]$, then $m[1, w]$ is calculated based on the values of $m[0, w]$. The same process is repeated to calculate $m[2, w], \ldots, m[n, w]$. After finishing calculating $m[i, w]$, the maximum value among all $m[n, w]$ $(0 \leq w \leq W)$ is selected as a solution of the problem.

In this research, KP is used to handle a distribution of rules extracted by GNP to each site. Rules with high data coverage will be the leaders of each site and KP will consider the similarity between the leader rules and remaining rules (which is considered as a value of item (rule) in KP) and coverage of rules (which is considered as weight in KP) should be matched with site capacities.

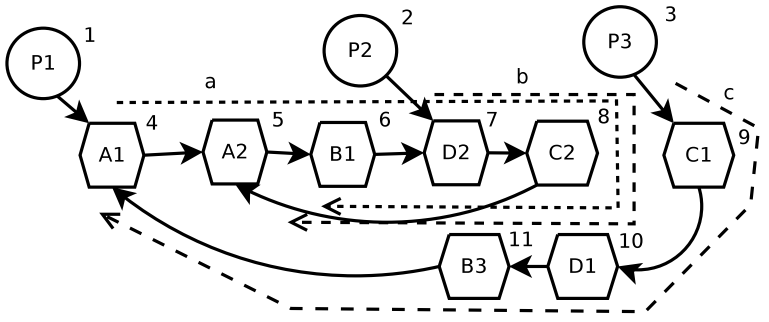

Fig. 3. GNP Implementation on Data Mining

Therefore, the similar rules to a certain leader are basically put into the same site. Detailed explanation of the implementation of KP in the rule distribution is available in section 3.2.

3 Combination of GNP and Knapsack problem

Implementation for processing record clustering is separated into two parts: GNP rule extraction and KP rule distribution.

3.1 GNP Rule Extraction

GNP is used to extract rules from a database by analyzing the database structure including:

Attributes amount : the number of attributes in a dataset. Each attribute will be divided into some nodes depending on its variation and value ranges (distance of minimum value and maximum value).

Data amount : the number of records in a dataset.

Data variation : how much different records are contained in a dataset. If every record in a dataset is different, variation is 100%, if half of the records in the dataset is different, variation is 50%, and if every record in a dataset is the same, variation is 1/(the number of data)×100%. For example, in Table 4 that will be shown in the later page, there are six data variation in total 310 data, so the variation is $(6/300) \times 100 = 1.94\%$.

GNP is used to extract rules from a dataset by analyzing all the records. Phenotype and genotype structures of GNP are described in Fig. 3 and Table 1, respectively. In Fig. 3, each node has its own node number (1–11), and in Table 1, the node information of each node number is described. The program size depends on the number of nodes, which affects the amount of rules created by the program.

Table 1
GNP gene structure of Fig. 3

i	NT_i	A_i	R_i	C_i
1	1	0	0	4
2	1	0	0	7
3	1	0	0	9
4	2	A	1	5
5	2	A	2	6
6	2	B	1	7
7	2	D	2	8
8	2	C	2	5
9	2	C	1	10
10	2	D	1	11
11	2	B	3	4

i	: Node number,
NT_i	: Node types; 1=processing, 2=judgment,
A_i	: Attribute index,
R_i	: Attribute range index,
C_i	: Connection

In the implementation of data mining, judgment node represents an attribute of the dataset, which is represented by A_i showing an attribute index such as price, stock, etc., and R_i showing a range index of an attribute value. For example, $A_i = A$ represents price attribute, and $R_i = 1$ represents value range $[0, 50]$ and $R_i = 2$ represents value range $[51, 80]$. Processing nodes show the start point of the sequence of judgment nodes which are executed sequentially by their connection. Sequences of nodes starting from each processing node (P_1, P_2, P_3) are represented by dotted line a, b and c. A node sequence flows until support for the next combination does not satisfy the threshold. The nodes with the attributes that have already appeared in the sequence will be skipped. Candidate rules extracted by the program of Fig. 3 to the dataset of Table 2 are shown in Table 3. In Table 3, three rules are extracted by the node sequence from each processing node.

The score of rule is defined as follows.

$$\text{Score of rule } r = \begin{cases} 0 \text{ if } sup(r) = 0 \\ 10 * sup(r) + 10 * (n_{con}(r) - 1) \text{ if } sup(r) > 0, \end{cases} \tag{3}$$

where $sup(r)$ is the support[1] of rule r and $n_{con}(r)$ is the length of rule r.

Fitness for evaluating an individual is defined as follows.

[1] Ratio of records that satisfy rule r

Table 2
Example of dataset

A_1	A_2	B_1	D_2	C_2	C_1	D_1	B_3
1	0	1	0	0	1	1	0
1	0	1	1	1	0	0	0
0	1	0	1	1	0	0	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	0	0	0	0	1	1	1

Table 3
Example of dataset and its support to the extracted rules

Processing Nodes	Extracted Rules	Support	Score	
			Rule	Template
1	$A_1 \wedge B_1$	3/6	15.00	6.00
	$A_1 \wedge B_1 \wedge D_2$	1/6	21.66	3.67
	$A_1 \wedge B_1 \wedge D_2 \wedge C_2$	1/6	31.66	4.67
2	$D_2 \wedge C_2$	2/6	11.66	4.33
	$D_2 \wedge C_2 \wedge A_2$	1/6	21.66	3.67
	$D_2 \wedge C_2 \wedge A_2 \wedge B_1$	0/6	0	0
3	$C_1 \wedge D_1$	2/6	13.33	4.33
	$C_1 \wedge D_1 \wedge B_3$	1/6	21.66	3.67
	$C_1 \wedge D_1 \wedge B_3 \wedge A_1$	1/6	31.66	4.67
			199.95	

(Score of template is introduced in section 3.1.3)

$$\text{Fitness} = \sum_{r \in R} \{sup(r) + 10(n_{con}(r) - 1) + \alpha_{new}(r)\}, \tag{4}$$

where $\alpha_{new}(r)$ is an additional value if rule r is newly extracted.

Table 3 shows the length and support of the extracted rules. Score of rule described by Eq. 3 is not only calculated by its support($sup(r)$) but also by its length($n_{con}(r)$). Considering the rule length makes rules more reliable because longer rules can cover various combinations of attributes. For example, $A_1 \wedge B_1$ has relatively high support 3/6 but only has the length two, so the score of rule is only 15.00. On the other hand, $C_1 \wedge D_1 \wedge B_3 \wedge A_1$ has the support only 1/6 but the length is four, therefore, the score becomes 31.66. $\alpha_{new}(r)$ is also included in the fitness because the objective of rule extraction is to discover

Table 4
Example of Frequency Table of Price Attribute

x	f	xf
10	30	300
25	25	625
50	30	1500
80	140	11200
100	65	6500
150	20	3000
Total	310	23125

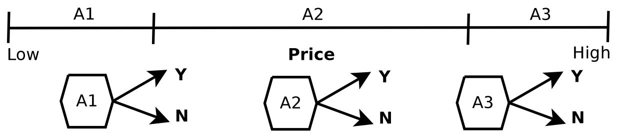

Fig. 4. Node for judging attributes

new rules from a dataset as much as possible.

The node preparation for GNP rule extraction contains two phases: node definition and node arrangement. In addition, two kinds of node arrangement methods are proposed: one is full random arrangement and the other is partial random arrangement.

3.1.1 Node Definition

The main purpose of node definition is to preparing judgment nodes that will be combined to create rules. First step is to find the minimum and maximum values of each attribute. For example, the minimum value of "price" attribute is 10 and the maximum value is 150 in the dataset with 310 records. Then, a frequency table is created per attribute as shown in Table 4. x shows the price of a product, and f shows how many times the product with the same price is recorded in the dataset. For example, product(s) with price $x = 10$ appeared 30 times. Then, mean value of $(\overline{xf})$ is calculated by Eq. 5.

$$\overline{xf} = \frac{\sum xf}{\sum f} = 74.60 \tag{5}$$

To define nodes from Table 4, data should be divided equally based on the amount of data. For example, three nodes could be created by dividing value range into three ranges considering the occurrence frequency as shown in Fig.

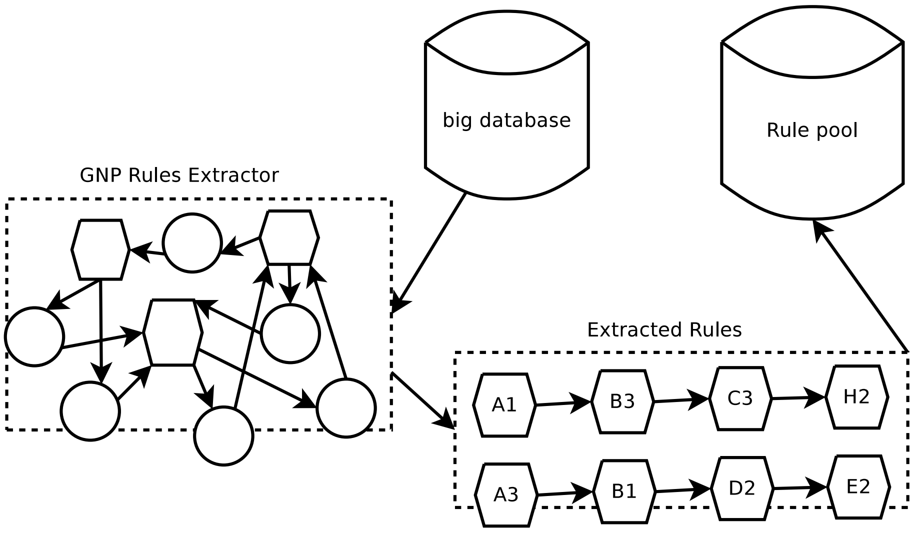

Fig. 5. GNP rule extraction

4. In this example, three ranges are: $x = \{10, 25, 50\}$ (85 data), $x = \{80\}$ (140 data) and $x = \{100, 150\}$ (85 data). First node and third node contain more than one price because each single record (10,25,50,100,150) does not have enough frequency to be defined as node. Mean ($\overline{xf} = 75.42$) is used to measure the minimum coverage to become a node. Through the measurement, the second node can be created from single record ($x = \{80\}$) because $f = 140$ exceeds $\overline{xf}$.

3.1.2 Node Arrangement : Full Random

The purpose of node arrangement is to select necessary nodes for efficiently extracting a large number of rules. Full random method randomly selects nodes from the defined nodes in section 3.1.1 and makes graph structures. From the created graph structures, GNP extracts a large number of important rules and stores them in the rule pool (Fig. 5). The original framework of the rule extraction is described in (Shimada et al., 2006) in detail.

After rules are extracted, GNP will measure the amount of coverage archived by the rules. In this research, coverage of rule *r* means the number of records that match (covered by) rule *r*. If a rule covers at least one data, such rule is added to a rule pool, otherwise, the rule is discarded. Rules with high coverage will be defined as elite rules and be the leaders of each cluster (site) in KP process. Rule extraction process continues until all the records in a dataset are covered.

To create a large number of good rules, crossover and mutation are executed.

Crossover: exchange one or more node(s) between parents to make new rules
Mutation: change one or more node(s) to make different combination of

nodes

Crossover is effective to switch weak nodes (nodes with less data frequency) of the parents with strong nodes (nodes with more data frequency). Mutation is effective to switch weak nodes of one individual to strong nodes.

3.1.3 Node Arrangement : Partial Random

Partial random method has two sequential processes of GNP, the first process is to find template rules and the second process is to execute general rule extraction of GNP combined with the templates created in the first process. Templates are extracted to obtain combinations of attributes that frequently happen in the dataset. Score of template is calculated by Eq. 6, and the templates with high scores will be used in the second process.

$$\text{Score of template } t = \begin{cases} 0 \text{ if } sup(t) = 0 \\ 10 * sup(t) + (n_{con}(t) - 1) \text{ if } sup(t) > 0 \end{cases} \tag{6}$$

Contrary to the score of rule (Eq. 3) which gives more weight on the node length, the score of template gives more weight on support as shown by Eq. 6. For example, the scores of templates are shown in Table 3 where the results are relatively contrast to the score of rules. $A_1 \wedge B_1$ has the highest score of template although the node length is just two. When $A_1 \wedge B_1$ is used as a template, partial random will be implemented by randomizing remaining attributes such as C and D.

In the template extraction process, only a few number of attributes are included in GNP rule extraction. It aims to increase the possibility to get templates with high support. For example, in "A. finding template" in Fig. 6, the combination of attribute A and D is defined as a template as a result of the score calculation (Eq. 6). It will increase the possibility to find good combinations with attribute A and D. In "B. rule extraction", the template and the remaining attributes, that is B and C, are considered. The rule extraction process can obtain rules with longer length than the templates.

Table 5 shows a simple example of partial random for easy explanation. Each template contains attribute A and D, and it is combined with the remaining attributes, that is B and C. The generated rule of $A_3 \wedge D_3 \wedge B_1 \wedge C_2$ obtains the highest score of rule (Eq. 3) because it has long rule length and high coverage.

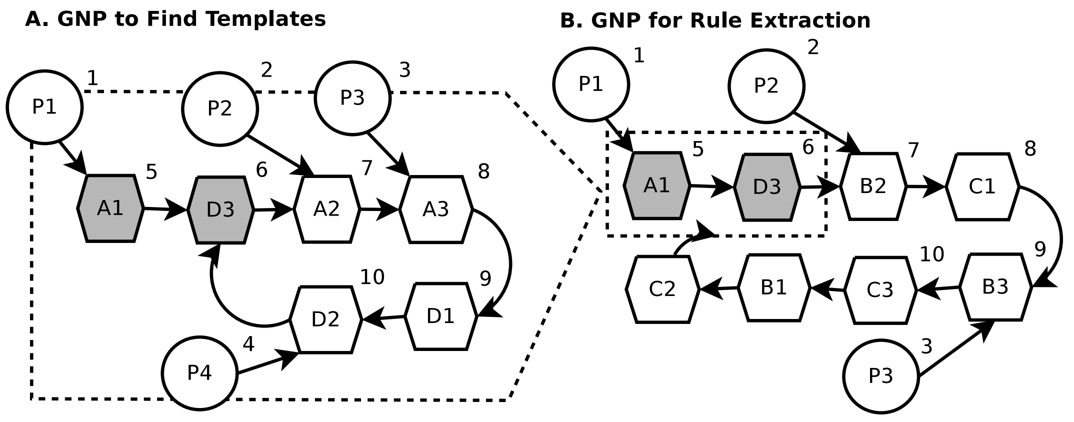

Fig. 6. Node Arrangement Optimization in GNP

Table 5
Example of combination of templates with remaining attributes

Template	Remaining attributes	Coverage	Score of rule
$A_2 \wedge D_3$	$B_1 \wedge C_2$	0	0
$A_2 \wedge D_3$	$B_3 \wedge C_2$	10	40.4
$A_1 \wedge D_3$	B_3	24	34.5
$A_3 \wedge D_3$	$B_1 \wedge C_2$	14	40.5

3.2 Rule Distribution with Knapsack Problem

After all the records in a dataset are covered by rules extracted by GNP, KP is used to distribute rules to several sites. Rules with high coverage (elite) become the leaders of each site, then KP considers the similarity of the remaining rules to the leader rules (value) and coverage of the rules (weight) in order to distribute the remaining rules to the sites. Similarity of remaining rule r_1 to leader rule r_2 is calculated by Eq. 7.

$$S(r_1, r_2) = \frac{N_{match}(r_1,r_2)}{Max\{N_{ante}(r_1),N_{ante}(r_2)\}} \tag{7}$$

$S(r_1, r_2)$: similarity between rule r_1 and r_2, $N_{match}(r_1, r_2)$: the number of matched attributes between r_1 and r_2, $N_{ante}(r)$ $(r \in \{r_1, r_2\})$: the number of attributes in rule r.

$Max\{N_{ante}(r_1), N_{ante}(r_2)\}$ means that longer rule length becomes a divider to the number of matched attributes between two rules $(N_{match}(r_1, r_2))$. When the longer rule includes attributes that are not contained in the shorter rule, those attributes are assumed to be matched. Examples of similarity calculation are shown in Table 6. From Table 6, rule 2 shows the highest similarity to the leader. The leader rule does not have attribute D, so every attribute D in the

Table 6
Example of similarity calculation between leader and remaining rules

Rule	A	B	C	D	$N_{match}(r_1, r_2)$	$S(r_1, r_2)$
Leader	A_1	B_3	C_2	-	-	-
1	$^*A_1$	B_2	C_1	$^*D_2$	2	2/4
2	A_2	$^*B_3$	$^*C_2$	$^*D_1$	3	3/4
3	$^*A_1$	B_1	$^*C_2$	-	2	2/3

* : matched attribute

Table 7
Results of GNP rule extraction with full randomization in six datasets

Attr	Full Random					Partial Random				
	Itr	n	Rule	Cvrg	Score	Itr	n	Rule	Cvrg	Score
3	34	2.33	34	29	10.15	25	3.00	39	23	20.29
3	78	2.23	12	808	10.01	45	3.00	18	526	20.02
8	564	3.45	23	42	20.23	435	6.62	43	22	50.26
8	1056	2.76	52	182	10.07	786	5.43	57	145	40.13
15	6290	2.46	34	20	10.15	5987	7.35	45	21	60.23
15	987	4.23	12	833	30.02	789	11.25	8	1110	100.03
					90.63					290.96

Attr : the number of attributes, Itr : number of iterations to cover all the records,
n : mean length of each rule, Rule : the average number of rules,
Cvrg : mean coverage, Score : mean score of rules

remaining rules is assumed to be matched.

4 Simulations

First, full random and partial random methods in the rule extraction of GNP are compared. Then, knapsack rule distribution is carried out and its results are compared with k-means and hierarchical clustering (Liu, Li, Xiong, Gao, & Wu, 2010).

4.1 GNP Rule Extraction

The result comparison between two node arrangement methods, that is, full randomization and partial randomization, is shown in Table 7. Six datasets are used for the comparison, where the number of data (5000) and the data variation (50%) are the same, however the number of attributes is different. The performance evaluation is executed to compare the iterations needed to cover all the data, the mean rule length, the number of extracted rules, and the mean score of rules. Here, iteration means the number of individuals created in the rule extraction until all the records are covered.

When the number of attributes is increased, the number of iterations needed to cover all the data tends to be increased. However, comparing the iteration needed by full randomization and partial randomization, partial randomization shows better results, i.e., less iteration are needed. Rules are extracted until all the records in the dataset are covered, but the records that have been already covered will not be re-included. Significant difference between full random and partial random is in the average node length where partial random basically shows longer length. By finding frequent item-set (template), partial random can establish the minimum node length of rules in each cluster. Partial random basically extracts larger number of longer rules than full random, therefore, the mean scores of rules extracted by partial random shows better results. From the next section, partial random method is used in the simulations.

4.2 Knapsack Rule Distribution

Here, silhouette value (Rousseeuw, 1987) is used to evaluate the clustering results. Silhouette provides a succinct graphical representation of how well each object lies within its cluster. Silhouette value is calculated by Eq. 8.

$$\begin{aligned} s &= \tfrac{b-a}{\max\{a,b\}} \\ &= \begin{cases} 1 - a/b, & \text{if } a < b \\ 0, & \text{if } a = b \\ b/a - 1, & \text{if } a > b \end{cases} \end{aligned} \tag{8}$$

s: Silhouette value for a single sample. The Silhouette value for a set of samples is given as the mean of the Silhouette values of each sample.
a: mean distance between a sample and all other points in the same cluster.
b: mean distance between a sample and all other points in the second nearest cluster.

The results of rule distribution are shown in Table 8. All the simulations are executed with the same number data (5000) and data variation (50%). k is the number of clusters (sites), “Balance of clusters capacity” shows the proportion of capacity of each site, for example, 1:1:1:1 means all the four

Table 8
Result of Knapsack Problem (Silhouette values)

k	Balance of Cluster Capacity	Average	Max	Min
8	1:1:1:1:1:1:1:1	0.97	0.98	0.92
8	4:2:4:6:4:2:7:5	0.91	0.97	0.88
6	1:1:1:1:1:1	0.87	0.91	0.86
6	1:5:2:6:3:2	0.82	0.88	0.78
4	1:1:1:1	0.75	0.81	0.70
4	1:4:2:1	0.72	0.79	0.68

sites have the same size, and 1:4:2:1 means the second site (size four) is four times larger than the first site (size one). “Average, Max and Min” show the data on Silhouette values obtained by the generated clusters. According to the silhouette values, the proposed method shows good clustering ability in the case of larger k and the balanced cluster capacity. The silhouette values are decreased as k decreases and the cluster capacity is unbalanced. This situation happens because of the capacity incompatibility between rule coverage and cluster capacity. For example, when the cluster capacity is only 100 data left, and the coverage of a certain rule is 120 data, 20 data will be distributed to another cluster, which affects the silhouette result. If the number of sites k is larger, various kinds of rules can be distributed to many sites, then more closer (similar) rules can be included in each cluster, which contributes to better silhouette values. If the cluster capacity is unbalanced, some sites have larger capacity and some have smaller. The sites with larger capacity have to contain various kinds of rules (which sometimes have a little far distance from each other), therefore, the silhouette values become smaller.

4.3 Comparison with K-means and Hierarchical Clustering

Clustering methods used for the comparison are k-means and hierarchical clustering (Liu et al., 2010) because both methods also use distance measure. k-means is based on a centroid concept for cluster separation and the centroids also represent each cluster. Hierarchical clustering is selected for the comparison because it uses the similar concept to the proposed method using leader rules to distribute rules. Although both methods can set the number of clusters to be created, they do not have a function to measure cluster capacity as KP. Thus, the cluster capacity problem is not discussed in this comparison. The proposed method can execute clustering considering the cluster capacities, which is one of the advantages over k-means and hierarchical clustering.

The features of attributes contained in the dataset for the comparison is shown

Table 9
Value ranges of attributes in the dataset

Attribute	Min Value	Max Value
A	100	500
B	1000	2000
C	700	1500
D	1	5
E	15	95
F	10000	30000
G	5000	50000
H	1	10

Table 10
Data on Silhouette values in the eight cases

case	k	Attribute	Data Amount	GNP-KP	K-means	Hierarchical
1	10	4	1000	0.921	0.912	0.852
2	10	4	2000	0.902	0.899	0.831
3	8	6	3000	0.843	0.832	0.780
4	8	6	4000	0.823	0.801	0.648
5	6	8	5000	0.576	0.356	0.521
6	6	8	6000	0.532	0.322	0.501
7	4	10	7000	0.325	0.104	0.312
8	4	10	8000	0.312	0.092	0.303

k : the number of clusters, Attribute : the number of attributes

in Table 9. In this simulation, eight datasets are prepared, where the values of each record are determined randomly in the ranges of each attribute and the data variation is fixed at 50%, however the number of records in each case is different. Table 10 shows more information on the eight datasets and Silhouette values obtained by the proposed method (GNP-KP), k-means and hierarchical clustering (HC). Fig. 7 graphically shows the silhouette values obtained by each method in the eight cases. When the number of attributes is small, the values obtained by GNP-KP and k-means are almost the same and higher than HC, however as the number of attributes increases, k-means shows lower values and HC shows higher values than k-means. GNP-KP shows the highest values in all the cases.

Fig. 8 shows the relation between silhouette values and k in the dataset with

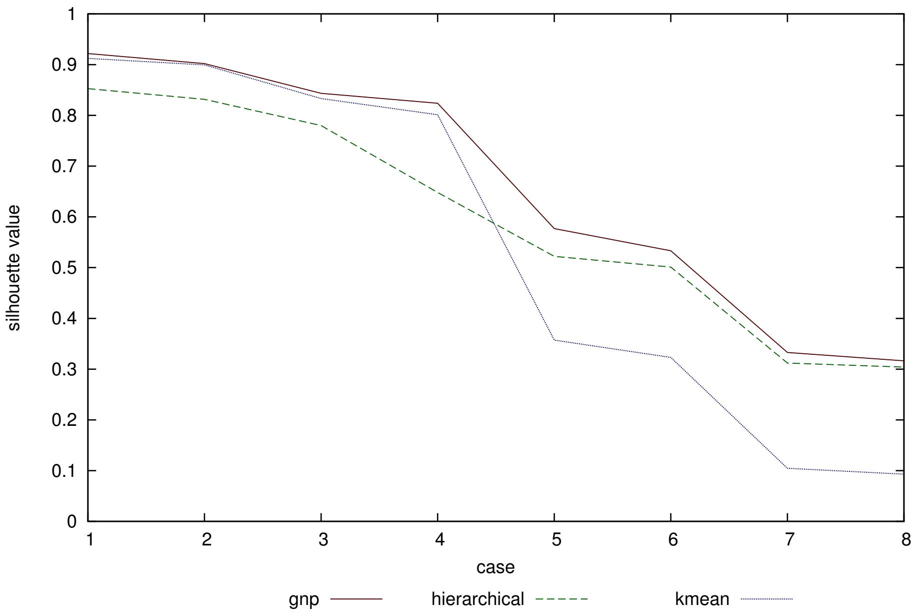

Fig. 7. Comparison of silhouette values in eight cases

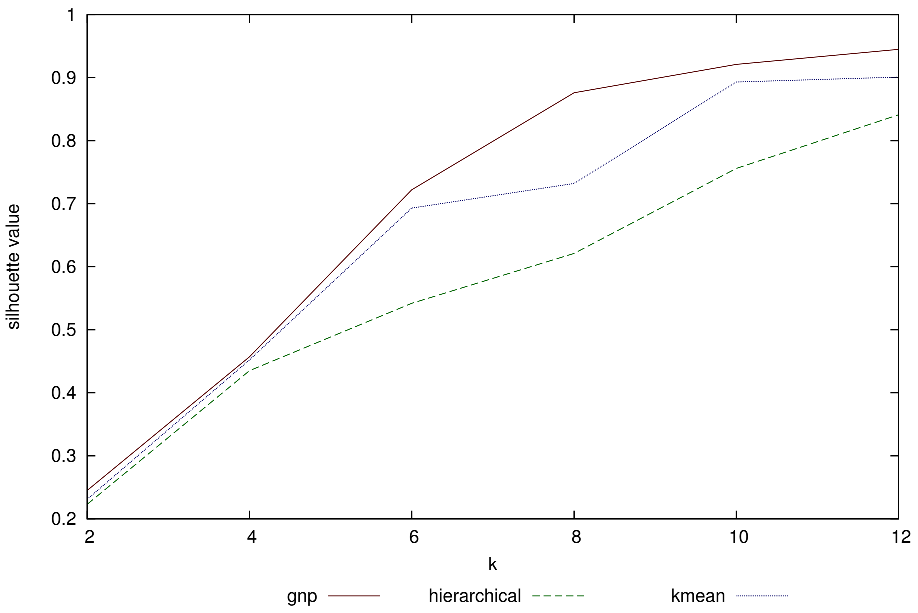

Fig. 8. Comparison of silhouette values for various k

four attributes and 5000 data. As k increases, silhouette values of the three methods increase, where GNP-KP shows better results in every case. Fig. 9 shows the relation between silhouette values and the number of attributes in the dataset with $k = 4$ and 5000 data. It can be seen from Fig. 9 that k-means is good to process datasets with the small number of attributes, but the silhouette values are decreased significantly as the number of attributes increases. HC shows relatively stable results for the changes of the number of attributes but does not reach the highest results. GNP-KP shows the decrease of silhouette values, but still makes better results in the six cases.

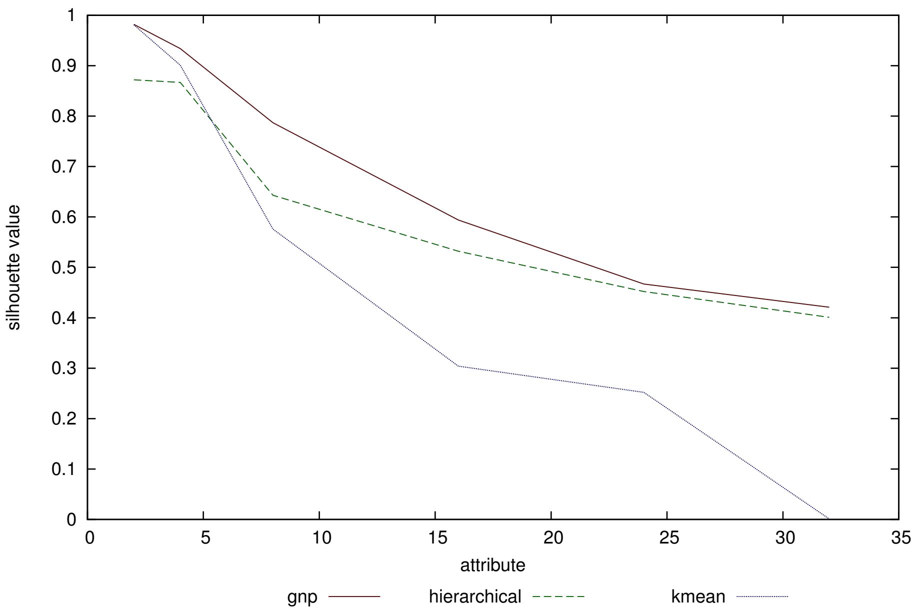

Fig. 9. Comparison of silhouette values for various number of attribute

The results of the clustering executed by GNP-KP are analyzed using Table 11 and 12 in detail. Table 11 shows the value ranges of the generated nodes for each attribute and their coverage. The different number of nodes are generated for each attribute depending on their value ranges as shown in Table. 9. For example, attribute D just has two nodes because its value range is only between one and five, but attribute G has ten nodes because it has a wider value range which is between 5000 and 50000. Each attribute also shows unbalanced coverage for each range. For example, one of the nodes of attribute F dominates the data with the range between 11045 and 11319 which shows larger coverage compared to other nodes. This domination has an effects on the nodes contained in each site (cluster), which is shown in Table 12. Sites with larger capacity contain rules with various kinds of attributes and more than one variation of node for each attribute. For example, in site 4, there are $B2, B4$ for attribute B and $C1, C2, C5$ for attribute C. This result indicates site 4 contains some rules with the same attributes of different value ranges; for example, $A2 \wedge B2 \wedge C1 \wedge E2 \wedge F1 \wedge G3 \wedge H3$ and $A2 \wedge B4 \wedge C2 \wedge D2 \wedge F1 \wedge G3 \wedge H3$. Sites with larger capacity have higher probability to contain more variation of records.

As described in section 3.1.3 on partial randomization, a combination of attributes (nodes) with high coverage would also result in high coverage. However, because each site (cluster) has a limit of capacity, attributes with high coverage would become substances of several sites. Since many data having major characteristics are difficult to be put into one site due to the limit of its site capacity, such data will be distributed to several sites. This problem could be solved by reducing the ranges of attributes, which contributes to decreasing the range domination. However, it will increase the calculation cost

Table 11
Generated value ranges of nodes and their coverage

Attr	Ranges	Cvrg	Ranges	Cvrg	Ranges	Cvrg
A	[140,173]	225	[221,238]	377	[270,317]	128
	[339,500]	270				
B	[1063,1146]	164	[1215,1251]	213	[1310,1316]	115
	[1368,1394]	238	[1397,1406]	100	[1414,2000]	170
C	[744,787]	52	[798,829]	180	[873,890]	206
	[930,938]	257	[962,981]	74	[1022,1022]	57
	[1055,1104]	37	[1133,1500]	137		
D	[1,2]	398	[3,5]	602		
E	[21,26]	97	[29,33]	267	[37,95]	636
F	[11045,11319]	308	[12329,13300]	189	[14394,14847]	162
	[14891,15502]	72	[16495,16879]	72	[18109,18152]	93
	[18913,20096]	34	[21084,30000]	70		
G	[5858,6890]	166	[8277,9055]	345	[10787,12407]	151
	[13629,13832]	34	[15876,17152]	33	[17960,18525]	122
	[19876,20841]	54	[22367,23533]	41	[23794,25908]	26
	[27327,50000]	28				
H	[1,1]	446	[2,3]	113	[4,10]	441

Attr : Attribute, Cvrg : Coverage

Table 12
Node substances in each site

Site	Attributes	Capacity	Data
1	$\{A3, B5, C5, D2, E2, F7, G2, H1\}$	100	84
2	$\{A1, A3, B1, B5, C4, D2, E3, F2, F5, G1, G2, H1\}$	500	295
3	$\{A4, B6, C8, D2, E3, F3, G2, G3, H1\}$	200	68
4	$\{A2, B2, B4, C1, C2, C5, D2, E2, F1,$ $F4, F8, G3, G5, G6, G8, H3\}$	600	420
5	$\{A1, C3, D1, E1, F6, G7, G9, H1, H2\}$	300	120
6	$\{A4, B2, C6, D1, E3, F1, G4, H3\}$	200	163

and influence the processing time.

5 Conclusions

This paper proposes a novel clustering method combining Genetic Network Programming and Knapsack Problem to handle record clustering. The proposed method can find good combinations of attributes to create rules for clustering, and also consider the capacity of sites to distribute rules. From the simulation results, the proposed method shows better clustering ability than k-means and hierarchical clustering in the eight kinds of datasets. In the future, we will consider the replication problems of distributed databases, where the query frequency is considered in the clustering problems and frequently accessed data will be stored in several sites to decrease the load of the accesses.

References

Bhuyar, P. R., Gawande, A. D., & Deshmukh, A. B. (2012). Horizontal fragmentation technique in distributed database. *International Journal of Scientific and Research Publications*, *2*(5).

Guinepain, S., & Gruenwald, L. (2006). Automatic database clustering using data mining. In *Proc. of the 17th international workshop on database and expert systems applications (DEXA'06)* (pp. 124–128).

Guinepain, S., & Gruenwald, L. (2008). Using cluster computing to support automatic and dynamic database clustering. In *Proc. of the 2008 IEEE international conference on cluster computing* (pp. 394–401).

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Koza, J. R. (1992). *Genetic Programming, on the programming of computers by means of natural selection.* Cambridge, Mass.: MIT Press.

Lai, K. (2006). The knapsack problem and fully polynomial time approximation schemes (FPTAS). *18.434: Seminar in Theoretical Computer Science, Prof. M. X. Goemans*.

Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. In *Proc. of the 2010 IEEE 10th international conference on data mining (ICDM)* (pp. 911–916).

Lowden, B. G., & Kitsopanidis, A. (1993). Enhancing database retrieval performance using record clustering. *Department of Computer Science, The University of Essex*.

Mabu, S., Chen, C., Lu, N., Shimada, K., & Hirasawa, K. (2011). An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *41*(1), 130–139.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53–65.

Shimada, K., Hirasawa, K., & Hu, J. (2006). Genetic network programming with acquisition mechanisms of association rules. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *10*(1), 102–111.

Singh, R. P. (2011). Solving 0–1 knapsack problem using genetic algorithms. In *Proc. of the 2011 IEEE 3rd international conference on communication software and networks (ICCSN)* (pp. 591–595).

Toth, P. (1980). Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, *25*(1), 29–45.

Zhao, J., Huang, T., Pang, F., & Liu, Y. (2009). Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In *Proc. of the 3rd international conference on genetic and evolutionary computing (WGEC'09)* (pp. 105–107).

Zilio, D. C., Rao, J., Lightstone, S., Lohman, G., Storm, A., Garcia-Arellano, C., & Fadden, S. (2004). DB2 design advisor: Integrated automatic physical database design. In *Proc. of the 30th international conference on very large data bases - volume 30* (pp. 1087–1097). VLDB Endowment.