Expert Systems With Applications xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Combination of genetic network programming and knapsack problem to support record clustering on distributed databases

Q1 Wirarama Wedashwara, Shingo Mabu*, Masanao Obayashi, Takashi Kuremoto

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

ARTICLE INFO

Keywords:
Genetic network programming
Database clustering
Knapsack problem
Record clustering

ABSTRACT

This research involves implementation of genetic network programming (GNP) and standard dynamic programming to solve the knapsack problem (KP) as a decision support system for record clustering in distributed databases. Fragment allocation with storage capacity limitation problem is a background of the proposed method. The problem of storage capacity is to distribute sets of fragments into several sites (clusters). Total amount of fragments in each site must not exceed the capacity of site, while the distribution process must keep the relation (similarity) between fragments within each site. 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. To solve this problem, GNP is used to extract rules from big data by considering characteristics (value ranges) of each attribute in a dataset. The proposed method also provides partial random rule extraction method in GNP to discover frequent patterns in a database for improving the clustering algorithm, especially for large data problems. The concept of KP is applied to the storage capacity problem and standard dynamic programming 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, therefore, the proposed method provides a new clustering method with an additional storage capacity problem.

1. Introduction

Distributed database management system (DDBMS) could be a solution for large scale information systems with large amount of data growth and data 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 standard dynamic programming solving knapsack problems (KP) (Lai, 2006; Singh, 2011) 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 the concept of KP can be used to define 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 the review of the proposed framework, Section 3 describes a review of literatures, 4 describes the detailed algorithm of the proposed framework, Section 5 shows the simulation results, and finally Section 6 is devoted to conclusions.

* Corresponding author. Tel./fax: +81 836 85 9519.
E-mail addresses: t001we@yamaguchi-u.ac.jp (W. Wedashwara), mabu@yamaguchi-u.ac.jp (S. Mabu), m.obayas@yamaguchi-u.ac.jp (M. Obayashi), wu@yamaguchi-u.ac.jp (T. Kuremoto).

http://dx.doi.org/10.1016/j.eswa.2015.10.006
0957-4174/

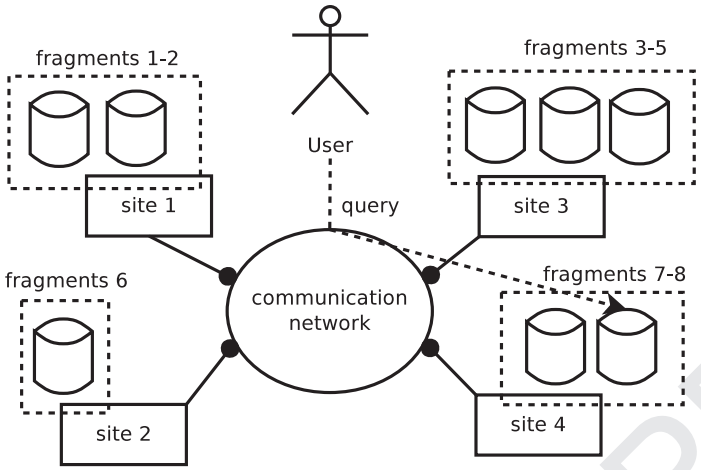

Fig. 1. A distributed database environment.

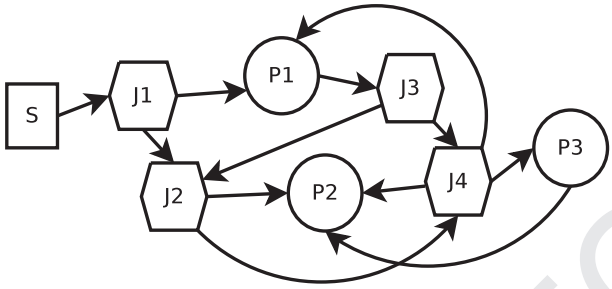

Fig. 2. Basic implementation of GNP S : start node, $[J_1, \ldots, J_4]$: judgement node, $[P_1, \ldots, P_3]$: processing node.

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 pth judgement function stored in a library for judgement nodes, while P_q $(q = 1, \ldots, m)$ denotes the qth 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 application for 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 4.1.

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.

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

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 (Singh, 2011; Zhao, Huang, Pang, & Liu, 2009).

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).

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

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, standard dynamic programming is applied to solve the 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 application for 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. Therefore, the similar rules to a certain leader are basically put into the same site.

Please cite this article as: W. Wedashwara et al., Combination of genetic network programming and knapsack problem to support record clustering on distributed databases, Expert Systems With Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.10.006

Detailed explanation of the implementation of application for KP in the rule distribution is available in Section 4.2.

3. Literature review

The proposed method uses GNP algorithm for data mining that has been proposed in (Mabu et al., 2011), and the proposed method is applied to the storage capacity problem of fragment allocation in distributed databases that has been introduced in (Özsu & Valduriez, 2011). This research involves the implementation of genetic network programming (GNP) for data mining and standard dynamic programming to solve the knapsack problem (KP) for the rule based clustering. Introducing storage capacity problem to the database clustering and introducing the concept of KP to solve the problem is one of the unique points of the proposed method. Moreover, the proposed method provides partial random feature selection in the rule extraction, which can discover frequent patterns of attributes in a database and improve the clustering quality. With the above features, the proposed method provides an automatic record clustering that aims to be a decision support system for record clustering in distributed databases.

The current related literature about fragment allocation is (Rahimi, Parand, & Riahi, 2015). The research presents an approach which simultaneously makes data fragments vertically and allocates the fragments to appropriate sites across the network. Bond Energy Algorithm (BEA) is applied with a better affinity measure that improves the quality of the generated clusters of attributes. BEA can find good relations between attributes by discovering frequent items between records in a database. The proposed method also discovers frequent pattern sets, but it is for realizing an automatic horizontal fragmentation or record clustering, not a vertical fragmentation as proposed by this literature.

The current related clustering topic is an automated feature weight learning proposed by (Saha & Das, 2015). This article presents and investigates a new variant of the fuzzy k-Modes clustering algorithm for categorical data with automated feature weight learning. This method automatically associates higher weights to features which are instrumental in recognizing the clustering patterns of the data in the classical fuzzy k-Modes algorithm. The proposed method in this paper also discovers frequent pattern sets of features (attributes) to improve the performance of clustering, which is explained in Section 4.1.3, and moreover, the proposed method can deal with a storage capacity problem that has not been solved in this literature.

Another related topic is evolutionary fine-tuning of automated semantic annotation systems proposed by Cuzzola, Jovanovic, Bagheri, and Gasevic (2015). The literature proposes a Parameter Tuning Architecture (PTA) for automating the task of configuring parameter values of semantic annotation tools with evolutionary computation. The similarity with the proposed method is the usage of evolutionary computation to find the proper combinations of features for solving the problem and use feature weight selection, but the problem of this literature, i.e., semantic annotation, is different from the proposed method in this paper, i.e., the target problem of this paper is a record clustering with an additional storage capacity limitation problem.

4. Combination of GNP and knapsack problem

The implementation of record clustering is separated into two parts: GNP rule extraction, and rule distribution based on standard dynamic programming for solving knapsack problem, which is explained in Sections 4.1 and 4.2. In addition, the complexity analysis of the entire clustering process is described in Section 4.3.

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.

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

4.1. 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) × 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.

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.

Please cite this article as: W. Wedashwara et al., Combination of genetic network programming and knapsack problem to support record clustering on distributed databases, Expert Systems With Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.10.006

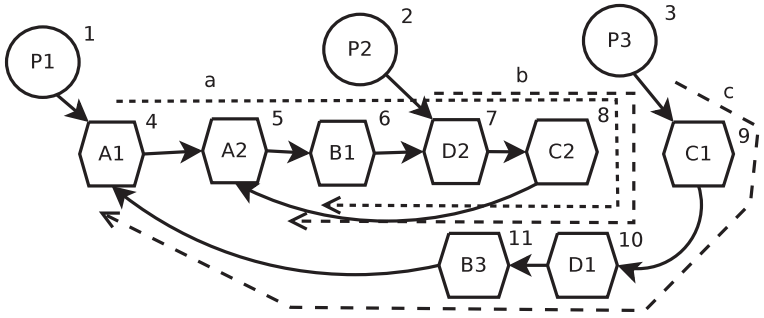

Fig. 3. GNP implementation on data mining.

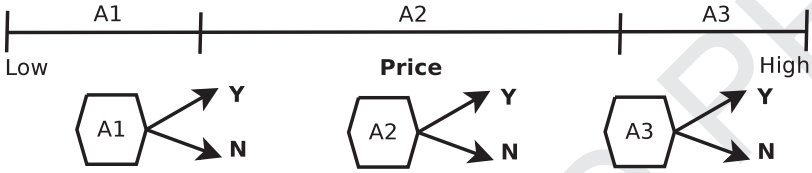

Fig. 4. Node for judging attributes.

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

Processing nodes	Extracted rules	Support	Score	
			Rule	Template
	$A_1 \wedge B_1$	3/6	15.00	6.00
1	$A_1 \wedge B_1 \wedge D_2$	1/6	21.66	3.67
	$A_1 \wedge B_1 \wedge D_2 \wedge C_2$	1/6	31.66	4.67
	$D_2 \wedge C_2$	2/6	11.66	4.33
2	$D_2 \wedge C_2 \wedge A_2$	1/6	21.66	3.67
	$D_2 \wedge C_2 \wedge A_2 \wedge B_1$	0/6	0	0
	$C_1 \wedge D_1$	2/6	13.33	4.33
3	$C_1 \wedge D_1 \wedge B_3$	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 4.1.3.

Table 4
Example of frequency table of price attribute.

x	f	xf
10	30	300
25	25	625
50	30	1500
80	140	11,200
100	65	6500
150	20	3000
Total	310	23,125

The score of rule is defined as follows.

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

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.

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

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 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.

4.1.1. 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).

$$\overline{xf} = \frac{\sum xf}{\sum f} = 74.60 \quad (5)$$

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. 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 $\overline{xf}$.

4.1.2. 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

[1] Ratio of records that satisfy rule r.

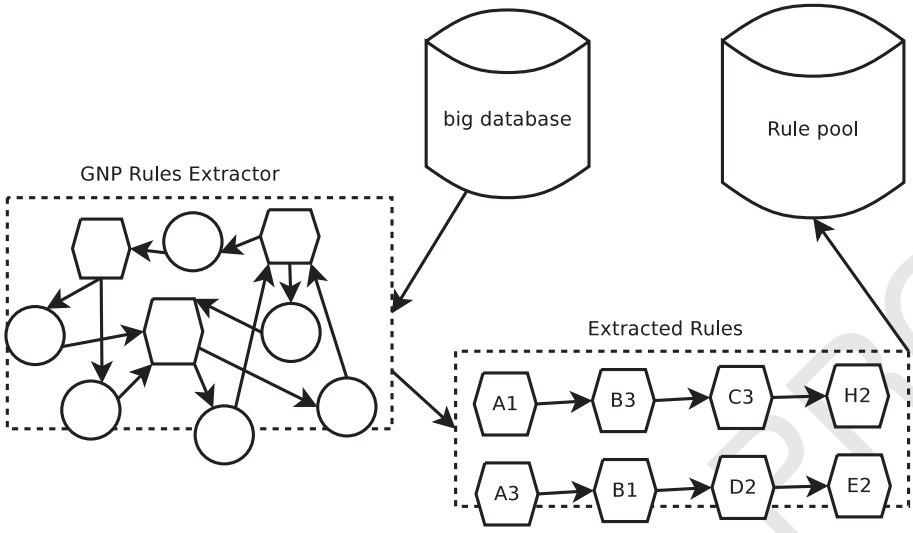

Fig. 5. GNP rule extraction.

randomly selects nodes from the defined nodes in Section 4.1.1 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.

4.1.3. 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.

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

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.

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

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.

4.2. Rule distribution based on standard dynamic programming for solving knapsack problem

After all the records in a dataset are covered by rules extracted by GNP, standard dynamic programming for solving KP problem is used to distribute rules to several sites. Rules with high coverage (elite) become the leaders of each site, then application 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

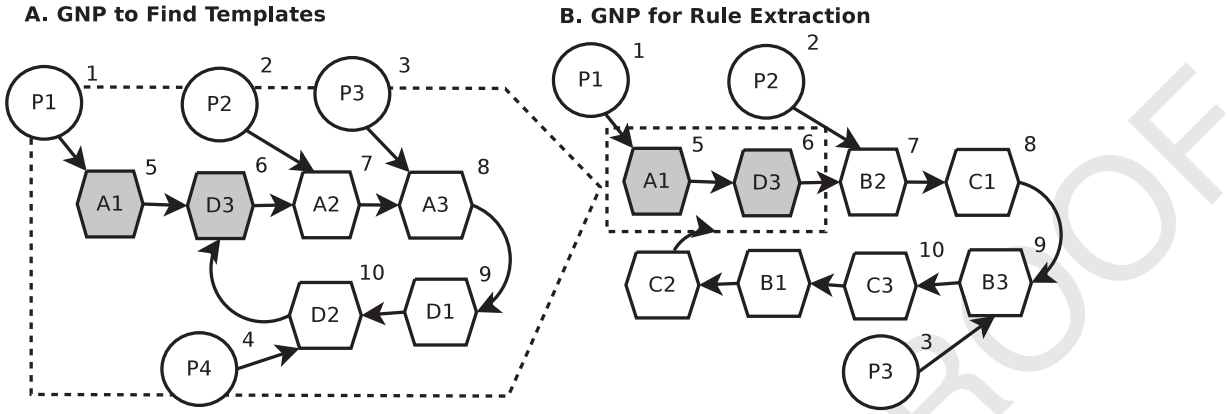

Fig. 6. Node arrangement optimization in GNP.

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]A_1	B_2	C_1	[a]D_2	2	2/4
2	A_2	[a]B_3	[a]C_2	[a]D_1	3	3/4
3	[a]A_1	B_1	[a]C_2	–	2	2/3

[a] Matched attribute.

Table 7
Comparison of crossover rate.

Crossover rate	Average score of rules	Iteration
0.01	20.31	28
0.05	20.29	25
0.1	20.24	23
0.2	20.12	23
0.5	19.78	22

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 remaining rules is assumed to be matched.

4.3. Complexity analysis

The main processes of the proposed method with their complexity analysis are summarized as follows.

(1) Rule extraction part
   (a) Node definition : This process prepares judgment nodes that will be combined to create rules. Complexity in this process is related to the number of data and attributes. The large number of attributes affects the number of nodes to be defined. The large number of data affects the complexity of creating a frequency table per attribute.
   (b) Node arrangement : This process selects necessary nodes for efficiently extracting a large number of rules. Complexity in this process is related to the number of attributes. The large number of attributes affects the number of possible combinations of attributes that could be extracted. Rule extraction process continues until all the data in a dataset are covered, therefore, the large number of possible combinations requires more iterations to cover all the data. To efficiently dealing with this complexity, partial random method is designed to hold the frequent patterns with high coverage to be used in the next iteration.
   (c) Extracted rules measurement : This process measures the coverage archived by the extracted rules. Complexity in this process is related to the number of data. The large number of data affects the number of measurement process of each rule.
(2) Rule distribution part: Standard dynamic programming is used to solve the KP problem, that is, the extracted rules are distributed to several clusters with the consideration of the similarity between rules (value) and coverage of the rules (weight). Each cluster cannot store all the rules when the sum of the coverage of the rules exceeds the storage limitation. Complexity in this process is related to the number of rules and clusters, and the storage limitations of each cluster. The large number rules increases the complexity by increasing the possible combinations of the rule distribution, while the large number of clusters and small storage limitations also increase the complexity by compounding the several purposes of distribution process.

5. Simulations

First, full random and partial random methods in the rule extraction of GNP are compared. Then, the knapsack rule distribution is carried out and its results are analyzed. Finally, the clustering simulations using six datasets downloaded from UCI Machine Learning Repository (archive.ics.uci.edu/ml/) are executed and their results are compared with other five conventional clustering algorithms.

5.1. GNP rule extraction

In this section, GNP rule extraction is executed and the parameter analysis of crossover rate and mutation rate is carried out to find the optimal parameters, then the performance of the two node arrangement methods is compared.

5.1.1. Parameter analysis of crossover rate and mutation rate

Main parameters of the proposed method that influences the quality of the extracted rules and iteration time are crossover rate and mutation rate. Therefore, the comparisons of several parameter settings of crossover rate and mutation rate are executed using the datasets with three attributes and 1000 samples.

Table 7 shows the average score of rules and iterations needed to cover all the data when the crossover rate is set at several values. Table 7 shows that the increment of the crossover rate slightly reduces the iteration time, and decreases the average score of rules. In this paper, the crossover rate 0.01 is used to obtain the best average score of rules although the iteration time increases a little. However,

Table 8
Comparison of mutation rate.

Mutation rate	Average score of rules	Iteration
0.01	20.29	28
0.05	20.13	26
0.1	19.98	24
0.2	18.45	20
0.5	14.34	18

the average score of rules does not depend on the crossover rate so much, thus the performance of the proposed method can be stable.

Table 8 shows the same comparison as Table 7 when the mutation rate is set at several values. Table 8 shows that the increment of the mutation rate has more effect on the reduction of iteration time and decrease of the average score of rules than the crossover rate. In evolutionary computation, mutation rate is generally set between 0.01 and 0.1, and 0.5 is a too large value. In this sense, if the mutation rate is set between 0.01 and 0.1, the influence of the parameter setting on the average score of rules is not large. From this comparison, we decided to use 0.01 as the mutation rate to obtain the best average score of rules although it slightly increases the iteration time.

5.1.2. Comparison of node arrangement methods

The result comparison between two node arrangement methods, that is, full randomization and partial randomization, is shown in Table 9. 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.

5.2. 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).

$$s = \frac{b-a}{\max\{a,b\}} = \begin{cases} 1 - a/b, & \text{if } a < b \\ 0, & \text{if } a = b \\ b/a - 1, & \text{if } a > b \end{cases} \quad (8)$$

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: the average dissimilarity (distance) of data within the same cluster.
b: the lowest average dissimilarity (distance) to any other cluster.

Table 10
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

The results of rule distribution are shown in Table 10. 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 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.

Table 9
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.

Table 11
UCI dataset.

	Attribute	Classes	Samples	Data type
Wine quality	12	2	4898	Real
Car evaluation	6	4	1728	Int
Image segmentation	19	7	2100	Int, real
Shuttle	9	7	54,600	Int
Covertype	54	8	581,012	Int
Yeast	8	10	1484	Real

5.3. Comparison with other methods

Six datasets from UCI machine learning repository (shown in Table 11) for comparison and the clustering performance is evaluated by both silhouette value and accuracy rate.

Five methods for the comparisons with the proposed method, i.e., K-means (Ahmad & Dey, 2007), Hierarchical Clustering (Karypis, Han, & Kumar, 1999), Fuzzy C means (Bezdek, Ehrlich, & Full, 1984), Order-constrained solution in K-means Clustering (OCKM) (Steinley & Hubert, 2008) and K Affinity Propagation (Zhang, Wang, Norvag, & Sebag, 2010). All the methods used in the comparisons are unsupervised clustering methods and use the euclidean distance as a distance metric except Hierarchical Clustering. The parameter setting of each method is determined as described below :

(1) K-means : Euclidean distance is used as the distance metric. the value of k is set as the number of classes of each dataset.
(2) Hierarchical Clustering : agglomerative is used as the hierarchy strategy and single linkage is used as a clustering method. The clustering procedure finishes when the number of groups reaches the number of classes of each dataset.
(3) Fuzzy C means : Minimum improvement of the fuzzifier m which determines the level of cluster fuzziness is set at 1.0×10^{-5}. The value of k is set as the classes of each dataset.
(4) Order-constrained solution in K-means Clustering (OCKM) : Euclidean distance is used as the distance metric and recursive dynamic programming strategy is used to improve the clustering quality. The value of k is set as the number of classes of each dataset.
(5) K Affinity Propagation : Euclidean distance is used as the distance metric and affinity propagation is used to improve the clustering quality. The value of k is set as the number of classes of each dataset.
(6) Proposed method: The main parameters of the proposed method are crossover rate and mutation rate, and these parameters are determined based on the results in Tables 7 and 8 as described in Section 5.1.1, where several settings of crossover rates and mutation rates are evaluated in terms of the average score of rules and the iterations needed to cover all the data.

Although the conventional clustering methods can set the number of clusters to be created, they do not have a function to measure cluster capacity as the proposed method with the function for solving 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 the conventional clustering algorithms.

In the simulations, accuracy rate is used as another clustering performance metric in addition to silhouette value. Accuracy rate is a common measure used to evaluate how well clustering algorithms perform on a dataset with a known structure. Accuracy rate shows different result from silhouette depending on the dataset.

Table 12 shows the evaluation result with silhouette and Table 13 shows the evaluation result with accuracy rate. Star marks (∗) on the side of the results in both tables indicate the best results in each row (dataset). The proposed method obtains the highest average results as shown in the last row of Tables 12 and 13. In both Tables 12 and 13, the proposed method also shows better clustering results in five out of total six datasets. The proposed method loses against other conventional methods for "shuttle" dataset only. Structure of "shuttle" dataset, shown in Table 11, does not show straight pattern to describe why the proposed method loses against other methods, but Table 13 shows that mean accuracy rate of all the methods (last column of Table 13) for "shuttle" dataset is the highest (0.824), that is, other conventional methods show better clustering results for the dataset that is relatively easy to make clusters comparing to other datasets.

Here, pay attention to the last column of Tables 12 and 13 showing the mean values of silhouette (Table 12) and accuracy rate (Table 13) of all the methods. For example, in Table 12, "covertype" dataset shows very low silhouette value which reaches −0.26, but its average

Table 12
Methods comparison with silhouette evaluation.

Dataset	Methods comparison with silhouette						
	OCKC	KAP	FCM	K-means	HC	GNP	Mean
Wine quality	0.172	0.182	0.227	0.123	0.224	0.241*	0.195
Car evaluation	0.795	0.789	0.809	0.801	0.752	0.812*	0.793
Segmentation	0.234	0.265	0.303	0.253	0.296	0.305*	0.276
Shuttle	0.324	0.314	0.398*	0.312	0.354	0.352	0.342
Covertype	−0.214	−0.453	−0.167	−0.254	−0.346	−0.125*	−0.260
Yeast	0.634	0.622	0.779	0.626	0.786	0.788*	0.706
Mean	0.324	0.287	0.392	0.310	0.344	0.396*	

Table 13
Methods comparison with accuracy rate evaluation.

Dataset	Methods comparison with accuracy rate						
	OCKC	KAP	FCM	K-means	HC	GNP	Mean
Wine quality	0.642	0.613	0.786	0.771	0.695	0.787*	0.716
Car evaluation	0.689	0.678	0.699	0.701	0.698	0.701*	0.694
Segmentation	0.678	0.724	0.776	0.725	0.712	0.792*	0.735
Shuttle	0.812	0.787	0.864*	0.839	0.818	0.824	0.824
Covertype	0.675	0.646	0.705	0.676	0.622	0.708*	0.672
Yeast	0.667	0.704	0.812	0.692	0.801	0.856*	0.755
Mean	0.694	0.692	0.774	0.734	0.724	0.778*	

Please cite this article as: W. Wedashwara et al., Combination of genetic network programming and knapsack problem to support record clustering on distributed databases, Expert Systems With Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.10.006

accuracy rate in Table 13 is 0.672. In this case, "Covertype" dataset has the largest number of attributes (54). Silhouette value is very sensitive to the data variation, thus the mean silhouette value of all the methods become lower than other cases (datasets). The similar results are also shown for "wine quality" and "image segmentation" datasets. By analyzing such results in Table 12, we can find that the large number of attributes tends to decrease the silhouette value because it increases the complexity of attribute combinations, while the large number of classes increases silhouette values because it becomes easier for many clusters to maintain data similarity, in other words, it is difficult for a few clusters to clearly separate many kinds of data fragments.

6. 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.

The proposed method provides a new clustering method with an additional storage capacity problem that is compatible with big data with large number of attributes, samples and clusters, and the clustering performance is evaluated with six datasets from UCI machine learning repository and the best average results comparing to other five conventional clustering algorithms are achieved.

The proposed method is less suitable for online processing because of the evolution time to obtain good rules. The proposed method is suitable for an offline processing that requires the optimal results than processing time.

In the future research, it is necessary to execute simulations with real DDBMS with running applications to test the applicability of the proposed method. The proposed method can be also developed as a middle-ware between distributed databases and an application of database fragment allocation management that can access CRUD (Create Read Update Delete) matrix of databases. The algorithm should be also improved to execute online processes. Combinations with other algorithms such as fuzzy logic and neural network can be realized to improve the ability of the proposed method.

References

Ahmad, A., & Dey, L. (2007). A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering, 63*(2), 503–527.
Bezdek, J. C., Ehrlich, R., & Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences, 10*(2), 191–203.
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).
Cuzzola, J., Jovanovic, J., Bagheri, E., & Gasevic, D. (2015). Evolutionary fine-tuning of automated semantic annotation systems. *Expert Systems with Applications*.
Guinepain, S., & Gruenwald, L. (2006). Automatic database clustering using data mining. In *Proceedings of the 17th international workshop on database and expert systems applications (DEXA'06)* (pp. 124–128). IEEE.
Guinepain, S., & Gruenwald, L. (2008). Using cluster computing to support automatic and dynamic database clustering. In *Proceedings of the 2008 IEEE international conference on cluster computing* (pp. 394–401). IEEE.
Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.
Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer, 32*(8), 68–75.
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, .
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.
Özsu, M. T., & Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.
Rahimi, H., Parand, F.-A., & Riahi, D. (2015). Hierarchical simultaneous vertical fragmentation and allocation using modified bond energy algorithm in distributed databases. *Applied Computing and Informatics*. Q2
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.
Saha, A., & Das, S. (2015). Automated feature weighting in clustering with separable distances and inner product induced norms–a theoretical generalization. *Pattern Recognition Letters, 63*, 50–58.
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 *Proceedings of the 2011 IEEE 3rd international conference on communication software and networks (ICCSN)* (pp. 591–595). IEEE.
Steinley, D., & Hubert, L. (2008). Order-constrained solutions in k-means clustering: even better than being globally optimal. *Psychometrika, 73*(4), 647–664.
Toth, P. (1980). Dynamic programming algorithms for the zero-one knapsack problem. *Computing, 25*(1), 29–45.
Zhang, X., Wang, W., Norvag, K., & Sebag, M. (2010). K-ap: generating specified k clusters by efficient affinity propagation. In *Proceedings of the IEEE 10th international conference on Data mining (ICDM), 2010* (pp. 1187–1192). IEEE.
Zhao, J., Huang, T., Pang, F., & Liu, Y. (2009). Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In *Proceedings of the 3rd international conference on genetic and evolutionary computing (WGEC'09)* (pp. 105–107). IEEE.
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 *Proceedings of the 30th international conference on very large data bases - volume 30*. In *VLDB '04* (pp. 1087–1097). VLDB Endowment.

Please cite this article as: W. Wedashwara et al., Combination of genetic network programming and knapsack problem to support record clustering on distributed databases, Expert Systems With Applications (2015), http://dx.doi.org/10.1016/j.eswa.2015.10.006