Expert Systems With Applications 46 (2016) 15–23

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

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/

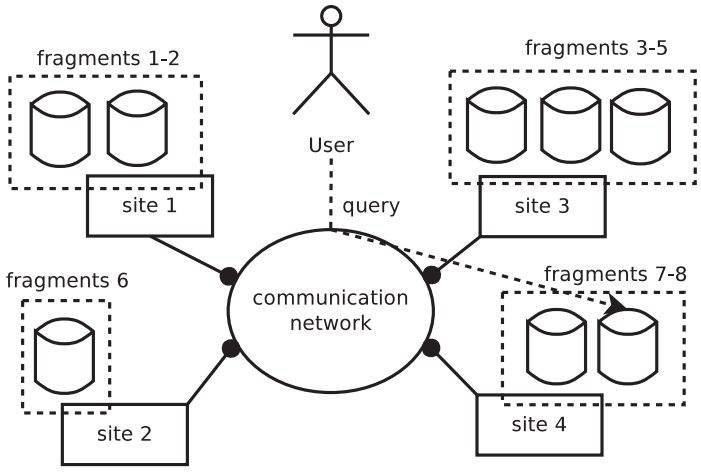

Fig. 1. A distributed database environment.

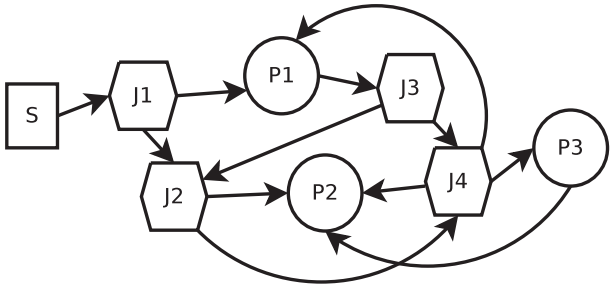

Fig. 2. Basic implementation of GNP S : start node, $[J_1, \ldots, J_4]$: judgement node, $[P_1, \ldots, P_3]$: processing node.

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.