An examination of cluster identification-based algorithms for vertical partitions

Chun-Hung Cheng
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, N.T., SAR, Hong Kong
Fax: +852 26035505
E-mail: chcheng@se.cuhk.edu.hk

Jaideep Motwani*
Department of Management
Seidman College of Business
Grand Valley State University
301 West Fulton, Suite 409C
Grand Rapids, MI 49504, USA
Fax: +1 616-331-7445
E-mail: motwanij@gvsu.edu
*Corresponding author

Abstract: A major objective in database physical design is to minimise disk access in processing database transactions. Vertical partitioning is a major design approach to ensure fewer disk accesses. In this paper, we consider the use of Cluster Identification (CI)-based approaches to produce vertical partitions in database design. To produce partitions, the existing CI-based approaches either remove attributes or duplicate attributes in the branching scheme. It turns out that the former sometimes gives undesirable partitions and that the latter produces overlapping partitions. To overcome these deficiencies, we propose to remove accesses in the branching scheme. Illustrative examples are used to show the effectiveness of this branching scheme. A computational study is also conducted. In addition, the new approach is extended to deal with a vertical partition case considering frequencies of accesses.

Keywords: clustering; cluster identification; branch-and-bound; data partitions.


Biographical notes: Chun-Hung Cheng received his BSc from the Chinese University of Hong Kong (CUHK) and his MSc in Computer Science and PhD from the University of Iowa, USA. Since 1994, he has been at the Department of Systems Engineering and Engineering Management of CUHK. He is now an Associate Professor. His research interests lie in computer and information technology and logistics and operations management. In particular, he makes use of computer and information technology in solving logistics and operations management problems. He has produced over 55 articles published in academic journals.
Database partitioning is a fundamental design problem in data management (Agrawal et al., 2004; Burleson, 2005; Chae et al., 2008; Hering et al., 2005; Wang et al., 2008; Ceri and Pelgate, 1994; Cheng and Chu, 2003; Zilo et al., 2004). The purpose of the design problem is to place data, which are frequently accessed together, on near-by sites (preferably on the same sites) to make processing of queries efficient. However, the identification of these common data groups (or fragments) is not always easy. There are two basic partition schemes: vertical and horizontal partition. In this paper, we deal with vertical partitioning and examine the use of Cluster Identification (CI)-based approaches to solve it.

Given the work-profile of a database application, in terms of the transactions and the data they access, the objective of our algorithm is to identify these common data groups. Individual groups, commonly known as data partitions, could then be placed on the most appropriate computer site(s). Without loss of generality, our algorithm has been designed under the following assumptions:

- The relational database model is assumed. The relational database is by far the most popular system.
- The application work-profile (i.e., the frequently accessed data and the corresponding transactions) is known. Based on this information, access patterns can be estimated, and from it, important database transactions can be located.
- In vertical partition applications, it is assumed that the primary key of a relation is duplicated in every vertical partition produced. In this way, the reconstruction of the whole relation from its vertical partitions is possible through a join operation.
A relation is essentially a table. Let a relation PROJECT concerning all ongoing projects of a company in Table 1. We may vertically divide it into two smaller units in Table 2. The primary key ProjNo is duplicated in both relations so that the original relation can be reconstructed.

Table 1  The PROJECT relation

<table>
<thead>
<tr>
<th>ProjNo</th>
<th>ProjName</th>
<th>Budget</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Database development</td>
<td>130,000</td>
<td>Michigan</td>
</tr>
<tr>
<td>J2</td>
<td>Group technology</td>
<td>115,000</td>
<td>Illinois</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>240,000</td>
<td>Michigan</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>330,000</td>
<td>Iowa</td>
</tr>
</tbody>
</table>

Table 2  Examples of vertical partitions

<table>
<thead>
<tr>
<th>ProjNo</th>
<th>ProjName</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>Database development</td>
<td>Michigan</td>
</tr>
<tr>
<td>J2</td>
<td>Group technology</td>
<td>Illinois</td>
</tr>
<tr>
<td>J3</td>
<td>CAD/CAM</td>
<td>Michigan</td>
</tr>
<tr>
<td>J4</td>
<td>Maintenance</td>
<td>Iowa</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ProjNo</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>130,000</td>
</tr>
<tr>
<td>J2</td>
<td>115,000</td>
</tr>
<tr>
<td>J3</td>
<td>240,000</td>
</tr>
<tr>
<td>J4</td>
<td>330,000</td>
</tr>
</tbody>
</table>

Notice that relation partitions PROJECT2 and PROJECT3 are the tables defined by [ProjNo, ProjName, Location] and [ProjNo, Budget], respectively. These partitions are not randomly formed. It is the role of the database designers to determine how best to partition the original relation in order to achieve the highest performance and/or reliability.

Among many existing approaches for vertical partitions, some approaches make use of the CI approach. Since the original CI algorithm (Kusiak and Chow, 1987) does not work for a realistic partition problem, Cheng (1995) propose removing attributes to produce clusters (i.e., partitions). Although Cheng’s algorithm works well for many problems, it may produce undesirable solutions with dangling transactions in some problem situations. Dangling transactions are those that do not access any attributes in a solution matrix. Another approach will be duplicating (instead of removing) attributes to avoid dangling transactions. However, duplication of attributes produces overlapping...
data partitions. Data updates must be propagated to several data partitions stored in different sites. Arbitrarily duplicating attributes (with the sole objective to partition a database) will result in deterioration of overall performance of the database.

In this paper, we consider the same framework but we aim to avoid problems of previous CI based approaches. Specifically, our approach is different from Cheng (1995) in the following ways:

- The model formulation in Cheng (1995) requires an upper limit on the number of transactions allowable in a partition. This constraint is important in preventing the formation of huge partitions with a large number of transactions and attributes. However, the constraint may not necessarily ensure the quality of a partition (i.e., a submatrix) formed. Our formulation drops this arbitrary constraint and includes a constraint to explicitly ensure the quality of a partition.

- Unlike Cheng (1995), we do not remove attributes to force the formation of vertical partitions. As we will illustrate later in this paper, the removal of attributes, in some cases, leads to undesirable solutions with dangling transactions. To avoid these undesirable situations, we remove accesses instead.

- We do not duplicate attributes to produce vertical partitions. The duplication strategy requires updates be made to each copy of data. This will undoubtedly increase update cost. To avoid this performance problem, we remove accesses instead.

- The formulation in Cheng (1995) only deals with the decomposition of a binary access matrix. When it is applied to vertical partitions, the frequencies of transactions are ignored. This has somewhat limited its use in solving practical problems. In this work, we show how to extend the CI-based approaches to handle cases where the frequencies of transactions are known.

Although we are only concerned with vertical partitions, our algorithm can be easily modified to deal with horizontal partitions. For details, interested readers may refer to Zhang and Orlowska (1994).

In the next section, we will discuss the vertical partition problem. A brief account of the literature is provided. In Section 3, the limitations of existing CI approaches are addressed. The proposed formulation and modified CI algorithm are presented in Sections 4 and 5, respectively. Illustrative examples are used to demonstrate the effectiveness of our algorithm. A computational study is conducted. A more realistic problem is considered in Section 6. We modify our formulation and algorithm to deal with the problem. A conclusion is given Section 7.

2 Vertical partitions


2.1 Background

An application work-profile describes the access patterns of a set of transactions, \{1, 2, \ldots\} say, over the attributes of the database relation, \(i.e., \{1, 2, \ldots\}\). For design purpose, it is commonly modelled by a transaction-attribute matrix. Consider the
transaction-attribute matrix in Figure 1. It contains five non-primary key attributes, \( i.e., \{1, 2, 3, 4, 5\} \), and four transactions, \( \{1, 2, 3, 4\} \), accessing the relation. A ‘1’ (or ‘0’) entry in the matrix indicates that the corresponding transaction uses (or does not use) the attribute(s) concerned. Notice that in the matrix in Figure 1, the distribution of ‘1’ in the matrix is completely random.

**Figure 1** A transaction-attribute matrix A1

<table>
<thead>
<tr>
<th>Attributes</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Consider another transaction-attribute matrix in Figure 2. It is formed by rearranging certain transactions and attributes of the matrix in Figure 1. The matrix in Figure 2 comprises a diagonal cluster structure. A diagonal cluster structure refers to the structure with most of ‘1’ arranged along the diagonal.

**Figure 2** A rearranged transaction-attribute matrix A2

<table>
<thead>
<tr>
<th>Attributes</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Figure 2, we can easily see two perfectly separable clusters (\( i.e., \) sub-matrices). A transaction accesses all attributes in a cluster. Attributes in a cluster make up a fragment. This lays down the objective of vertical partitions.

Clearly, such an ideal diagonal cluster structure is not easy to determine; especially in real life situations. For instance, consider the transaction-attribute matrix in Figure 3. An ideal diagonal cluster structure does not exist in this matrix because attributes 4 and 5 is accessed by transactions from different clusters. Attributes 4 and 5 are known as inter-cluster attributes.
2.2 Previous work

Hoffer (1976) formulates a 0–1 nonlinear integer-programming model that minimises storage, retrieval, plus update costs subject to capacity constraints on database sub-files. An approximate solution based on the Bond Energy (BE) algorithm is developed. Eisner and Severance (1976) show that the data partitioning problem is isomorphic to the minicut-maxflow network problem. Hence, the Ford/Fulkerson algorithm can be used to solve the original problem. However, it is inefficient for large problems. Hammer and Niamir (1979) design a mechanism that can find a near optimal vertical partition, although it conducts a search through the space of all possible partitions by employing a hill-climbing technique.

Navathe et al. (1984) extend the work of Hoffer. Affinity among attributes is defined in the affinity matrix to express the extent to which they are simultaneously processed. The BE algorithm (McCormick and Schwietzer, 1972) is introduced to partition attributes according to their affinity. Since the BE algorithm does not necessarily produce a solution in a diagonal structure, a heuristic algorithm is required to divide attributes into overlapping or non-overlapping fragments. Cornell and Yu (1987; 1990) develop an integer programming formulation to solve the problem of vertical partitioning. At iteration, the formulation finds an optimal partitioning that splits a relation into two fragments. The formulation can be applied recursively until no profitable split can be found. However, this approach only finds a locally optimal partition.

Navathe and Ra (1989) propose an algorithm for vertical partitioning that uses a graphical technique. The major feature of this algorithm is that all fragments are generated by one iteration in a time of $O(n^2)$. However, their algorithm has some undesirable features as we note below. Lin and Zhang (1993) propose a graphical vertical partitioning algorithm to overcome some deficiencies found in the Navathe-Ra algorithm.

In both Navathe and Ra (1989), and Lin and Zhang (1993), the associated attribute-affinity graph is constructed based on an affinity matrix. The edges with affinity value 0 are removed from the attribute-affinity graph to form an affinity graph. A cycle of an affinity graph is distinguished if the affinity values of the edges on the cycle are larger than the affinity values of the edges going out from the cycle. A distinguished subgraph $G'$ is cycle-atomic if there is no distinguished cycle whose vertex set is a proper subset of the vertex set of $G'$. A graph $G$ is 2-connected if the number of vertices is more than 2, and if an arbitrary vertex is removed; the resulting graph is still connected. Similar to a cycle-atomic distinguished cycle, we may define a 2-connected atomic distinguished subgraph. A completely distinguished subgraph is a complete subgraph, which is distinguished.
Navathe and Ra (1989) observe that a fragment can be produced from a distinguished cycle. A heuristic algorithm is used to find a cycle-atomic distinguished cycle in the affinity graph. However, their algorithm cannot always find a cycle-atomic completely distinguished cycle even if it exists. In Lin and Zhang (1993), their algorithm first finds a 2-connected atomic distinguished subgraph instead of a cycle-atomic distinguished cycle. Their algorithm is more efficient.

Cheng (1995) proposes a new vertical partitioning algorithm for a binary access matrix using a CI approach. His approach requires the removal of attributes that prevent the formation of partitions. Huang and Van (1995) propose an heuristic search algorithm to search the large solution space of partitions and to choose one partition that yields the minimum number of disk accesses by using the A* technique. Cheng et al. (2002) explore the use of a genetic search based clustering algorithm for data partitioning to achieve high database retrieval performance. By formulated the underlying problem as a Travelling Salesman Problem (TSP), they take the advantages of this particular structure. Three new crossover operators are also proposed and experimental results indicate that they outperform an existing approach in solving the data-partitioning problem.

Agrawal et al. (2004) propose to integrate vertical and horizontal partitioning for automated physical database design. Their experiment shows that this integrating approach is critical to automated design. Further, they also claim that their approach is scalable. Son and Kim (2004) develop an adaptable vertical partitioning method for both best-fit and n-way vertical partitioning. Extensive computational study is carried to support the validity of their method.

Bellatreche and Boukhalfa (2005) present a genetic algorithm for schema partitioning selection problem. The proposed algorithm gives better solutions since the search space is constrained by the schema partitioning. Several experimental studies using the APB-1 release II benchmark for validating the proposed algorithm is also presented. Lastly, Abuelyaman (2008), proposes a vertical partitioning algorithm for improving the performance of database systems. The proposed algorithm uses the number of occurrences of an attribute in a set of queries rather than the queries accessing these attribute, thus enabling the fragmentation of a database schema even before its tables are populated.

### 3 Cluster identification approaches

Since we examine the use of CI-based approaches, we will discuss the limitation of existing CI-based approaches in this section. Kusiak and Chow (1987) are the first to propose the use of CI approach for a binary matrix. Although the CI algorithm by Kusiak and Chow (1987) is simple, it only works for matrices that are perfectly decomposable. It will not be able to split non-perfectly decomposable problems such as matrix A3 in Figure 3. To deal with matrices such as Matrix A3, Cheng (1995) proposes to embed the CI algorithm in a branch and bound framework. In this section, we discuss their limitations.

Cheng (1995) formulates the vertical partitioning problem as follows:

P1 Remove attributes to decompose a transaction-attribute matrix into separable submatrices with the maximum number of ‘1’ entries retained in submatrices (i.e., clusters) subject to the following constraints:
C1 Empty transaction submatrices are not allowed.
C2 The number of transactions in a submatrix cannot exceed an upper limit, b.
C3 Empty attribute submatrices are not allowed.

Constraints C1 and C3 ensure that submatrices (i.e., clusters) must have transactions and attributes. An upper limit on the number of transactions in constraint C2 is included to prevent the formation of a single submatrix with all attributes and transactions. A designer may specify and vary the upper limit to examine the effect on the composition of a submatrix (i.e., cluster). The objective function is to maximise the number of ‘1’ entries retained in submatrices. This objective function ensures that attributes in a submatrix will satisfy transactions in the same submatrix as much as they can.

Constraint C2 forces a transaction-attribute access matrix to form clusters. Although it is important in preventing forming huge clusters with a large number of transactions and attributes, it does not ensure the quality of clusters formed. In addition, this constraint assumes that a single value is used as the upper limit on all clusters formed. However, it is possible that different upper limits for different clusters may be needed. Certainly, we may try to formulate the problem in such a way that different upper limits are used for different clusters. This effort turns out to be difficult. Since the formulation does not specify the number of clusters formed, it is impossible to determine how many upper limits are needed in advance. Even if this had been possible, the number of possible values would have been significantly increased.

As we will discuss later in the paper, Cheng (1995) produces infeasible solutions in some problem instances. To address this limitation, we may duplicate inter-cluster attributes to decompose a binary access matrix. Hence we formulate the problem as follows:

P2 Duplicate the minimal number of attributes to decompose a transaction-attribute matrix into separable submatrices subject to the following constraints:

C4 An empty transaction submatrix is not allowed.
C5 An empty attribute submatrix is not allowed.
C6 The cohesion measure of a submatrix is more than or equal to a threshold, d.

Formulation P2 is a minimisation problem. The strategy for clustering is to duplicate attributes while Cheng is to remove attributes. Constraint C6 is different from Constraint C2 in Cheng’s formulation. This constraint explicitly imposes a threshold to ensure the acceptable level of the quality of a cluster while Cheng’s constraint requires a maximum number of transactions allowed in a cluster.

To ensure the quality of a cluster (i.e., a submatrix) formed, they develop a cohesion measure of all transactions in a cluster. Suppose a transaction-attribute submatrix \( S = [a_{ij}] \) where \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \). And let \( |A| \) be the cardinality of Set A. They use the following procedure to determine the cohesion measure of submatrix \( S \).

- Define Set R to contain transaction indexes in \( S \) as: \( R = \{1, 2, \ldots, m\} \).
- Define Set C to contain attribute indexes in \( S \) as: \( C = \{1, 2, \ldots, n\} \).
• The total number of ‘1’ entries in submatrix S, W, is defined as:
  \[ W = \{ a_{ij} = 1, \quad i \in R \text{ and } j \in C \} \].

• The cohesion measure of submatrix S, \( CM_s = W / (|R||C|) \).

Suppose in submatrix S has \(|R||C|\) ‘1’ entries, then \( CM_s \) will be one. If the submatrix has all ‘0’ entries, then \( CM_s \) will be zero. The cohesion measure of a cluster is designed to assume a value between 0 and 1. The number of possible values to test will be reduced to 10, which is significantly less than Cheng (1995). The use of a measure to ensure the quality of clusters formed is a common practice in cluster analysis. As it is shown later in this paper, the cohesion measure is effective in forming good quality clusters.

Although we can avoid the dangling problem of Cheng (1995), their method may produce large number of inter-cluster attributes. This may not be desirable and practical. Inter-cluster attributes often require updating multiple copies of the same data to ensure data integrity. This inevitably undermines the performance of the system. Therefore, when updates are infrequent, multiple updates would be affordable. Under this circumstance, the duplication strategy provides a good solution. Otherwise, it is not a good option.

### 4 Modified formulation

The proposed algorithm adopts the same solution framework of Cheng (1995) but it is designed to address the limitations. This algorithm corrects the dangling problem in Cheng (1995) and does not duplicate inter-cluster attributes. We formulate our problem as follows:

**P3** Remove the minimal number of ‘1’ entries to decompose a transaction-attribute matrix into separable submatrices subject to the following constraints:

- **C7** An empty transaction submatrix is not allowed.
- **C8** An empty attribute submatrix is not allowed.
- **C9** The cohesion measure of a submatrix is more than or equal to a threshold, \( d \).

### 5 Our algorithm

Our algorithm, similar to Cheng (1995), uses a branch and bound approach. The branch-and-bound clustering approach uses the CI algorithm at each node to identify clusters if possible.

#### 5.1 Branch-and-bound framework

The branch-and-bound approach iteratively examines each unfathomed node in the search tree. The root node of the tree contains only one matrix (e.g., an initial matrix). Other nodes in the tree, representing feasible or infeasible solutions, may contain several
An examination of CI-based algorithms for vertical partitions

submatrices. The lower bound $Z_L$ on the objective function is equal to the number of ‘1’ entries removed. The upper bound $Z_U$ is calculated only after a feasible solution has been found. The algorithm is as follows:

**Step 0** (Initialisation): Initialise the initial node containing the 0-1 matrix $[a_{ij}]$. Set the upper bound $Z_U = \infty$.

**Step 1** (Branching): Based on the depth-first search strategy, select an active node (not fathomed). Apply the CI algorithm to a submatrix of the node that does not satisfy constraint C9. When the submatrix of the node cannot be partitioned, apply a branching rule (discussed below) to the submatrix.

**Step 2** (Bounding): For each new node, obtain a lower bound, $Z_L$, on the value of the objective function.

**Step 3** (Fathoming): Exclude a new node from further consideration if:

- **Test 1** $Z_L \geq Z_U$
- **Test 2** Any corresponding submatrix violates constraint C7 or C8
- **Test 3** All corresponding submatrices satisfy the three constraints. If $Z_L < Z_U$, store this solution as the new incumbent solution, set $Z_U = Z_L$, and reapply Test 1 to all remaining unfathomed nodes created previously.

**Step 4** (Stopping rule): Stop when there are no unfathomed nodes remaining; the current incumbent solution is final. Otherwise, go to Step 1.

The CI algorithm is first applied to a submatrix at a node. If the submatrix is decomposable, one new node is generated. A branching rule is applied at a node only when the corresponding submatrix does not satisfy constraint C9 and cannot be decomposed by the CI algorithm.

When a matrix is not decomposable, our approach considers the removal of accesses. The maximum number of accesses in a submatrix $S$ is $m \times n$, where $m$ is the number of transactions and $n$ is the number of attributes. It will be impractical to remove all accesses. Hence, we have to find a rule that helps us identify attributes that prevent the decomposition. To describe our rule, let us first define the following sets:

- $Q_j = \{j': a_{ij} = 1 \text{ and } a_{ij'} = 1, \text{ for all } j' \neq j\}$
- $C_j = Q_j \cup \{j\}$
- $R_j = \{i : a_{ij} = 1 \text{ and } a_{ij'} = 1, j' \in C_j \text{ and } i = 1, 2, ..., m\}$

The rule uses the void measure to estimate the likelihood of an attribute being an inter-cluster attribute. The rule consists of two steps:

**Step 1** Calculate the void measure of attribute $j$, $V_j$, for every attribute $j$ in $S$. The void index for attribute $j$ is defined as: $V_j = | \{ a_{ij} = 0, i \in R_j \text{ and } j' \in C_j \} |$.

**Step 2** For an attribute $j$ with the highest void index, remove one access one at a time from matrix $S$.

To understand the advantage of our branching rule, let us consider matrix A3 in Figure 3. Assume that the threshold $d$ value is 0.8. Clearly, matrix A3 cannot be decomposed into perfectly separable submatrices. Our approach removes accesses from a matrix that
prevent the matrix from decomposition. Based on our branching rule, accesses (2, 4) and (2, 5) are subsequently removed from the matrix and replaced by asterisks to represent inter-cluster accesses. The resulting matrix is shown in matrix A4 (Figure 4). A partial branch-and-bound tree is shown in Figure 5.

**Figure 4** Matrix A4

```
1 2 3 4 5
1 1 1 *
2 1 1 1 *
3 1 1
4
```

**Figure 5** A partial search tree

```
1 2 3 4 5
1 1 1 0 0
2 1 1 1 1
3 0 0 0 1
4 0 0 1 1
```

5.2 Comparison with other cluster identification approaches

Using Cheng (1995) approach, matrix A3 will produce matrix A5 (shown in Figure 6). Although the removal of attributes 4 and 5 does allow the matrix to decompose, it creates an undesirable solution that shows only one identifiable cluster with two dangling transactions. Cheng suggests three ways to deal with the removed attributes:
An examination of CI-based algorithms for vertical partitions

1. form a cluster using all removed attributes
2. assign removed attributes back to newly formed clusters
3. duplicate removed attributes among clusters that require the attributes.

For dangling transactions, no specific treatments are provided in Cheng (1995). Hence, it somewhat limits its applicability.

Figure 6  Matrix A5

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

The duplication strategy generates a resulting matrix in Figure 7. In this case, updates on Attributes 4 and 5 must be propagated to both data partitions.

Figure 7  Matrix A6

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Next, we use a slight larger problem matrix to demonstrate the effectiveness of our algorithm. Figure 8 show an input matrix and a solution is shown in Figure 9. In Matrix A8 produced by our approach, we can see three clusters along the diagonal with ten inter-cluster accesses. If Cheng (1995) is used, the attributes with asterisks will be removed from the matrix. The removal of these attributes will eliminate the central
cluster (enclosed) producing dangling transactions 2, 3, and 6. If the duplication approach is used, then an inter-cluster attribute (whose column contains one or more asterisks) will be duplicated for the same number of times as the asterisks appear in the column. For instance, Attribute 5 is an inter-cluster attribute. It will be duplicated two times resulting three copies, one for each cluster.

Figure 9  Matrix A8

Our new algorithm identifies three clusters and ten inter-cluster attributes. The decision regarding the treatment of inter-cluster attributes is left to the database designer.

5.3 Computational experience

In this section, we present the computational experience of our approach. We implemented the algorithms in JAVA and solved the test problems on a typical Pentium PC. Table 3 shows the computational results. Our algorithm provides a solution to a small problem. Notice that the computation time increases rapidly as the problem size increases. It is true that the computation time may expand to an unacceptable limit for a very large problem.

<table>
<thead>
<tr>
<th>PN</th>
<th>Size $m \times n$</th>
<th>Cluster configuration</th>
<th>Our algorithm</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$07 \times 11$</td>
<td>Number of clusters</td>
<td>0.8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of accesses removed</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$08 \times 20$</td>
<td>Number of clusters</td>
<td>0.8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of accesses removed</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$11 \times 22$</td>
<td>Number of clusters</td>
<td>0.8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of accesses removed</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$13 \times 24$</td>
<td>Number of clusters</td>
<td>0.8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of accesses removed</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$14 \times 24$</td>
<td>Number of clusters</td>
<td>0.8</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of accesses removed</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
We have examined the behaviour of our algorithm in solving a very large problem. It turns out that our algorithm generate the solution to the problem very early in the search. It spends a majority of the computation time to check whether the incumbent solution is in fact the best one. One good way to make our algorithm a practical solution approach is to force the algorithm to terminate when it reaches a time limit on a problem. The use of this termination condition is not new. It has been widely used in dealing with a large problem.

6 Consideration of frequencies of accesses

Traditional CI-based approaches, when applied to vertical partitions, ignore the frequencies of accesses. This restricts their applicability in practice. Let us consider a vertical partition example considering frequencies of transactions in Table 4. The proposed CI algorithm in its original form described in the previous section will not be able to solve this problem. Hence, we have to modify the formulation and the algorithm for this case. It turns out that a minor modification to the formulation and the algorithm will do it.

To formulate the problem, we define $a_{ij} = 1$ if transaction $i$ accesses attribute $j$ and $a_{ij} = 0$, otherwise. We also let $freq_i$ be the frequency of transaction $i$. The problem may be formulated as follows:

P4 Minimise the loss of total accesses (i.e., $\sum_i \sum_j a_{ij} freq_j$) due to the removal of $a_{ij}$ for decomposing a transaction-attribute matrix into separable submatrices subject to the following constraints:

C7 An empty transaction submatrix is not allowed.

C8 An empty attribute submatrix is not allowed.

C9 The cohesion measure of a submatrix is more than or equal to a threshold, $d$.

The formulated problem P4 is similar to problem P3 except its objective function. This formulation ensures that when a matrix is not decomposable, it will always find the decomposition to minimise the loss of total accesses due to the removal of accesses.

<table>
<thead>
<tr>
<th>Attribute</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Transaction</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Access frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The algorithm is similar to the proposed CI-based algorithm discussed in the previous section. For each node in a search tree, the lower bound $Z_L$ on the objective function is $\sum_i \sum_j a_{ij} \text{freq}_{ij}$ resulting from the removal of $a_{ij}$. The upper bound $Z_U$ is calculated only after a feasible solution has been found. If the modified algorithm is applied to matrix A9, it will produce matrix A10 in Table 5. In the solution matrix, two clusters have been identified along the diagonal.

Table 5  Rearranged transaction-attribute matrix A10 by our new algorithm

<table>
<thead>
<tr>
<th>Attribute</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Access frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>T3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>T4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>T8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>T2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>T5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>T6</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>T7</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

7 Conclusion

In this paper, we consider the use of CI-based approaches for vertical partitions. To produce partitions, existing approaches either remove attributes or duplicate attributes in the branching scheme. The former approach sometimes produces undesirable solutions with dangling transactions. Since very little has been said about the treatment of dangling transactions, its application is limited. Although the latter approach avoids the creation of dangling transactions, it produces overlapping partitions. The duplication of attributes requires update be propagated to each partition. Duplication without giving considering its effects on update will adversely affect the overall performance of a database system. Hence, it is obvious that this approach may not lead to the most desirable solution.

To overcome these deficiencies, we propose to remove accesses as a branching scheme. Illustrative examples and a computational study are used to confirm the effectiveness of this branching scheme. In addition, we also extend our algorithm to deal with a case considering frequencies of transactions.

Acknowledgements

We would like to thank anonymous reviewers for their constructive comments and suggestions.

This work was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 411205).
References


