Computing and Informatics, Vol. 28, 2009, 29–56

SOLVING LARGE SCALE INSTANCES OF THE DISTRIBUTION DESIGN PROBLEM USING DATA MINING

Héctor Fraire, Laura Cruz

Instituto Tecnológico de Ciudad Madero
1o. de Mayo y Sor Juana I. de la Cruz S/N, C. P. 89440
Cd. Madero, Tamaulipas, México
e-mail: {hfraire, lcruzr}@prodigy.net.mx

Joaquín Pérez, Rodolfo Pazos

Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET)
AP 5-164, C. P. 62490, Cuernavaca, Mor., México
e-mail: {jperez, pazos}@cenidet.edu.mx

David Romero

Instituto de Matemáticas, Universidad Nacional Autónoma de México
Cuernavaca, Mor., México
e-mail: davidr@matcuer.unam.mx

Juan Frausto

ITESM, Campus Cuernavaca
Reforma 182-A, C. P. 62589
Temixco, Mor., México
e-mail: juan.frausto@itesm.mx

Manuscript received 27 November 2006; revised 11 October 2007
Communicated by Ladislav Hluchý

Abstract. In this paper we approach the solution of large instances of the distribution design problem. The traditional approaches do not consider that the instance size can significantly reduce the efficiency of the solution process. We propose a new approach that includes compression methods to transform the original instance into a new one using data mining techniques. The goal of the transformation is *to condense the operation access pattern* of the original instance to reduce the amount of resources needed to solve the original instance, without significantly reducing the quality of its solution. In order to validate the approach, we tested it proposing two instance compression methods on a new model of the replicated version of the distribution design problem that incorporates generalized database objects. The experimental results show that our approach permits to reduce the computational resources needed for solving large instances by at least 65 %, without significantly reducing the quality of its solution. Given the encouraging results, at the moment we are working on the design and implementation of efficient instance compression methods using other data mining techniques.

Keywords: Data mining, machine learning, distribution design problem

1 INTRODUCTION

The increasing popularity of the Internet and e-business has generated a great demand for applications of distributed databases (DDBs). These applications are developed using Distributed Database Management Systems (DDBMSs). Despite the advanced technology of DDBMSs, the design methodologies and tools have many limitations. Consequently, database administrators carry out the distribution design using empirical an informal approaches due to the problem complexity. In this paper a formal and systematic methodology is proposed aimed at overcoming these limitations. The distribution design problem consists of determining data allocation so that the communication costs are minimized. Like many other real problems, it is a combinatorial NP-hard problem. The solution of large scale instances is usually carried out solving a simplified version of the problem or using approximate methods [1, 2]. General purpose nondeterministic heuristic methods are, at present, the best tools for the approximate solution of this class of problems [3, 4]. In the balance these methods will be referred to as heuristic methods. For several years we have worked on the distribution design problem and its solution with heuristic methods. In [5] we proposed an on-line method to set the control parameters of the Threshold Accepting algorithm. In [6] a mechanism for automatically obtaining some control parameter values for genetic algorithms is presented.

2 RELATED WORK

Traditionally, the analysis of the distribution design problem has been carried out using a DBMS model for evaluating different design alternatives. In recent studies, regarding the implementation of automatic tools for database physical design, the DBMS query processor is used. In the following subsections the most relevant works on these approach are described.

2.1 Works Based on Modeling

Ceri, Wiederhold and Navathe carried out the pioneering studies of this type [7, 8]. Their approach consists of dividing the problem into two serial phases: fragmentation and allocation. The key idea behind the serial approach is inspired on the proverb "divide and conquer", which supposedly permits to deal with "less-hard" instances than those obtained with an integral approach. However, it overlooks the fact that the inputs to both phases are the same [9]. In [5] this fact is exploited and it is proven that the integral approach outperforms the serial one.

Pérez et al. propose the use of an integral approach for the non replicated version of the problem [5] and develop a integer linear programming method, called DFAR, in which vertical fragmentation and non replicated fragment allocation are integrated. The objective function of the model considers query processing costs, fragment join costs, migration costs, and storage costs. They report the solution of instances with up to 500 attributes, 500 sites and 500 queries using the Threshold Accepting algorithm.

Johansson et al. approached the replicated data allocation problem for a distributed database system on a high speed network and introduced a network latency time model for integrating it as a component of the system response time [10]. They formulated a model, using realistic parameters, for optimizing the response time of a distributed system with parallel transactions operating on replicated data.

Huang et al. approach the replicated fragment allocation problem on a wide area network [11]. They propose a model that incorporates fragment retrieval costs, read and write operations cost, and the protocol connection costs.

Tamer Ozsu approaches the problem of the dynamic allocation of replicated fragments on the Internet, for which he presents a model that simulates a physical model and minimizes the total processing cost of operations: retrieval, join and migration [12]. A maximum of four replicas are considered, and the model parameters permit to specify instances that simulate loads typically found on the Internet.

Baiao et al. propose a methodology for fragmenting distributed databases, which includes techniques for generating horizontal, vertical and mixed fragments [13]. They evaluate the fragmentation schemas generated considering the concurrency level of the operations execution.

2.2 Works Related to Industrial Applications

Papadomanolakis et al. approach the fragmentation problem of a very large database and present an algorithm called AutoPart, which automatically generates the schema of the database physical fragmentation for a representative transaction load [14].

They present experimental results from a real astronomical database implemented on SQL Server 2000. The database objects to be fragmented are tables and indexes.

Agrawal et al. report an automatic tool for the physical design of a database [15]. Given an operation load, it automatically determines the fragmentation and physical allocation schemas and the fragmentation of the logical database. Currently it can only obtain the design for one node, but they are considering extending their tool to handle multiple nodes. They present experimental results, on SQL Server, obtained from standard tests [16].

Zilio et al. present an automatic tool for the physical distribution design, on multiple servers, of indexes, materialized views and multidimensional tables for a given operation load [17]. The tool includes an independent module for load compression in order to increase its capacity. Due to the low efficiency of the clustering method used, it is used only to compress a fraction of the load defined by the user [18]. In order to evaluate the distribution design quality they use DB2 UDB and standard tests [16].

2.3 Comparative Analysis

Table 1 shows the most relevant characteristics of the studies related to the problem of database design automation. The shaded rows correspond to the works related to industrial applications, which are analyzed in order to identify an important limitation of the available models. Columns 2 and 3 specify the type of fragmentation performed, column 4 indicates whether the fragment allocation is replicated or not, and column 5 indicates whether the approach uses instance compression. The approach proposed in [5] has been the most successful in solving large scale instances of the problem. The main limitation of these approaches based on modeling is that they do not consider that the size of the instances can significantly reduce the efficiency of the solution process, which only involves a model of the problem and a solution algorithm. Conversely, in [17] the relevance of instance compression is recognized, but the effect of compression on the solution quality is not considered; consequently, the compression methods proposed are inefficient and do not guarantee the scalability of the tools for automatic database design.

In order to overcome these limitations, we propose two efficient instance compression methods that use data mining techniques: clustering [19, 20] and progressive sampling [21, 22]. We tested them on a new model of the replicated version of the distribution design problem that incorporates generalized database objects [23].

3 ARCHITECTURE OF THE SYSTEM

This section describes the architecture of the system used for analyzing the problem of automating the distribution design. A distributed database system with client/server architecture operating on the Internet has four fundamental components: servers, clients, data, and network links.

	Fragmentation		Replicated	
	Vertical	Horizontal	Allocation	Compression
Pérez Ortega [5]	✓		✓	
Johansson [10]			✓	
Huang [11]			✓	
Visinescu [12]			✓	
Baiao [13]	✓	✓		
Papadomanolakis [14]		✓		
Agrawal [15]		✓		
Zilio [17]	✓	✓	✓	✓
Our approach [23]	✓	✓	✓	✓

Table 1. Related work on distribution design automation

Data can only be stored in the servers and can be fragmented and replicated. The distribution unit is a relation fragment and the data may have a given initial distribution. Changes in access patterns to fragments, in given intervals of time, may cause the movement, creation or elimination of their replicas, or even fragment reconfiguration.

Servers are DDBMSs with all of their functionality and are the sites where data resides. Each pair of servers has a communication cost associated, which represents the distance between them and is assumed constant. This is a usual assumption in this kind of studies and permits to simplify the problem treatment.

Clients are the sites where users issue their operation requests, which are transferred to the server to which they are connected through permanent or intermittent (wireless) links. All the operations generated by the clients of a server are considered to be issued from the server site. Servers are connected to the Internet through high speed links.

Figure 1 shows the main characteristics of an architecture of this type. For obtaining a good system performance, the database administrator must design the fragmentation and fragment allocation schemas, aiming at reducing the transmission costs incurred by the read, write (for maintaining consistency) and fragment join operations. An optimal design permits to minimize the overall communication cost required for handling all the operations [9].

4 PROBLEM DEFINITION

The DDB distribution design problem consists of allocating DB-objects, such that the total cost of data transmission for processing all the applications is minimized. A DB-object (or simply object) is an entity of a database that requires to be allocated, which can be an attribute, a tuple set, a relation, or a file. DB-objects are independent units that must be allocated at the sites of a network.

Let us consider a set of DB-objects $O = \{o_1, o_2, \ldots, o_{no}\}$, a computer communication network that consists of a set of sites $S = \{s_1, s_2, \ldots, s_{ns}\}$, where a set

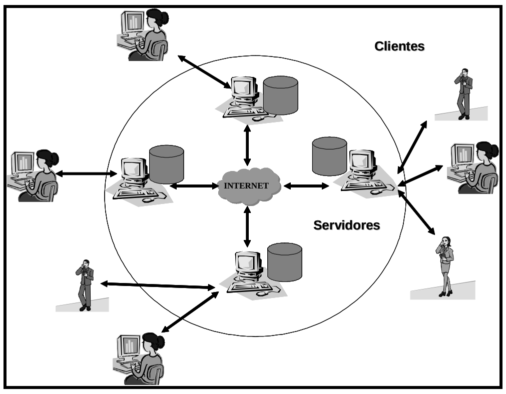
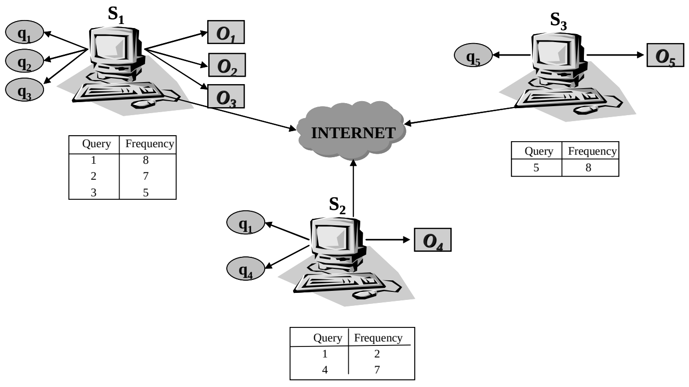


Fig. 1. Architecture of the system

of operations $Q = \{q_1, q_2, \ldots, q_{nq}\}$ are executed, the DB-objects required by each operation, an initial DB-object allocation schema, and the emission frequencies of each operation from each site in a time period. The problem consists of obtaining a new allocation schema that adapts to a new database usage pattern and minimizes transmission costs. Figure 2 depicts an instance of the problem with 4 DB-objects, 3 sites and 4 operations, as well as the emission frequency of the operations from each site and the usage matrix of DB-objects by operations.



Query	Frequency
1	8
2	7
3	5

Query	Frequency
5	8

Query	Frequency
1	2
4	7

Fig. 2. Distribution design problem

5 MATHEMATICAL MODEL

Traditionally it has been considered that the DDB distribution design consists of two sequential phases: fragmentation and allocation. Contrary to this widespread belief, it has been shown that it is simpler to solve the problem using our approach which combines both phases [5, 23]. A key element of this approach is the formulation of a mathematical model that integrates both phases.

5.1 Objective Function

The objective function of the mathematical model (1) includes four terms: the first models the cost of processing read-only operations, the second models the cost of read-write operations, the third models the migration cost of DB-objects, and the last one models the storage cost of DB-objects in the sites.

$$\begin{aligned}\min z \quad = \quad & \sum_k \sum_j f_{kj} \sum_m \sum_i q_{km} l_{km} c_{ji} w_{jmi} + \sum_k \sum_j f'_{kj} \sum_m \sum_i q'_{km} l'_k c_{ji} x_{mi} \\ & + \sum_j \sum_m \sum_i w'_{jmi} c_{ji} d_{mi} + \sum_m \sum_i CA_i b_m x_{mi} \end{aligned} \tag{1}$$

The problem is modeled using binary integer linear programming. In Table 2 the parameters and variables used in the formulation are described.

5.2 Model Constraints

A solution to the model must satisfy a set of constraints that specify: the possible replication of DB-objects, their location, the access policy applied to the read and write operations, the conditions for DB-object migration, and the storage capacity of the sites. In Table 3 the model constraints are shown.

5.3 Cost of Read Operations

The first term of the objective function models the overall cost of read operations. The calculation is carried out as follows: For each read operation k issued from site j, its emission frequency f_{kj} is multiplied by the cost of retrieving from different sites the fraction of DB-objects that is needed. The DB-objects m required by the read operation are all those for which the product $q_{km} w_{jmi} = 1$. For each DB-object m located at site i and needed by read operation k, the cost for transmitting it from site i to site j is $l_{km} c_{ji}$.

Note that restriction 1 allows several replicas of the same DB-object in different sites, but fortunately restriction 4 enforces the use of only one of the available replicas by each read operation. This restriction uses binary parameter θ_{jm}, which adopts value 1 if and only if any of the products $f_{ki} q_{km}$ is positive. The value of this product indicates whether there exists some read operation k issued from some site j that needs DB-object m. If so, restriction 4 states that, if DB-object m is required from site j, only one of the variables w_{jmi} may adopt value 1, which means that from all the replicas of m only one is chosen for the operation.

no	Number of DB-objects to distribute.
ns	Number of sites in the network.
nq	Number of user queries.
f_{ki}	Frequency matrix of integer values that describes the emission frequency of read-only query k from site i, in a given time interval.
q_{km}	Usage matrix that indicates the DB-objects that are used by the different read-only queries; $q_{km} = 1$ if query k uses DB-object m; $q_{km} = 0$ otherwise.
l_{km}	Communication packets required to transmit a DB-object m to satisfy a read-only query k, $l_{km} = (b_m \cdot s_k)/PA$ where b_m is the size in bytes of DB-object m, s_k is the selectivity of query k and PA is the size in bytes of the communication packet.
f'_{ki}	Frequency matrix of integer values that describes the emission frequency of read-write query k from site i, in a given time interval.
q'_{km}	Usage matrix that indicates the DB-objects that are used by the different read-write queries; $q'_{km} = 1$ if query k uses DB-object m; $q'_{km} = 0$ otherwise.
l'_k	Communication packets required to transmit a write instruction; $l'_k = P_k/PA$ where P_k is the size in bytes of the write instruction required by query k.
d_{mi}	Communication packets required to create a replica of DB-object m in site i.
CA_i	Storage cost for byte in site i.
CS_i	Storage capacity in bytes of site i.
c_{ji}	Matrix that contains the transmission costs between sites.
x_{mj}	Binary variable that indicates if DB-object m is located in site j; $x_{mj} = 1$ if DB-object m is in site j, otherwise $x_{mj} = 0$.
w_{jmi}	Binary variable that indicates if DB-object m located in site i is required by a read-only query located in site j; $w_{jmi} = 1$ if DB-object m located in site i is required by a read-only query in site j, otherwise $w_{jmi} = 0$.
w'_{jmi}	Binary variable that indicates if DB-object m that must be located in site i is currently located in site j; $w'_{jmi} = 1$ if DB-object m that must be located in site i, is currently located in site $j \neq i$, otherwise $w'_{jmi} = 0$.
a_{mj}	Binary matrix that contains the DB-objects current allocation schema; $a_{mj} = 1$ if DB-object m is currently located in site j, otherwise $a_{mj} = 0$.

Table 2. Model elements

$\sum_i x_{mi} \geq 1 \ \forall_m$	DB-objects can be replicated in one or several sites.
$x_{mi} \leq \sum_k q_{km}\theta_{im} \ \forall_{m,i}$	Each DB-object can only be located in one site i that issues an operation that uses the DB-object. ($\theta_{im} = 1$ if $\sum_k f_{ki}q_{km} > 0$, and $\theta_{im} = 0$ if $\sum_k f_{ki}q_{km} = 0$)
$ns \ x_{mi}$-$\sum_j w_{jmi} \geq 0 \ \forall_{m,i}$	It is possible to request from other sites only the DB-objects stored in those sites; i.e., if a DB-object is not stored in a site, this object can not be requested to this site from any other site.
$\sum_i w_{jmi} - \theta_{jm} = 0 \ \forall_{m,j}$	When a replicated DB-object is requested from a given site j, only one of its replicas should be used for satisfying a read operation. ($\theta_{jm} = 1$ if $\sum_k f_{kj}q_{km} > 0$, and $\theta_{jm} = 0$ otherwise.)
$\sum_m x_{mi}p_m * CA \leq CS_i \ \forall_i$	The overall space used for storing the DB-objects in a site must not exceed the site storage capacity.
$w'_{jmi} - a_{mj} \leq 0 \ \forall_{j,m,i}$	A new replica of a DB-object must be generated from an existing replica in the previous distribution schema.
$w'_{jmi} - (1 - a_{mi})x_{mi} = 0 \ \forall_{m,i}$	A new replica of a DB-object must be generated from exactly one replica that exists in the previous distribution schema.

Table 3. Model constraints

5.4 Cost of Write Operations

The second term of the objective function models the overall cost of write operations. The calculation is performed in the following way: for each write operation k issued from site j, its emission frequency f_{kj} is multiplied by the cost for transmitting the write instruction to all the sites that hold the DB-objects needed by the operation. The DB-objects m needed by the write operation are all those for which $q_{km}x_{mi} = 1$. For each DB-object m located at site i that is needed by write operation k, the cost for transmitting the write instruction from site i to site j is $l'_k c_{ji}$.

5.5 Cost of DB-object Replication

The third term models the cost of modifying the current allocation schema of DB-object into a new allocation schema. This model term allows the application of the model not only to the design of DDBs but also to its redesign. The calculation is carried out adding the cost of transferring each DB-object from its previous to its new location. The decision variable w'_{jmi} permits to select the DB-objects m that were previously located at site j and must be relocated to site i. For each of these DB-objects the cost for transmitting it from site j to site i is $d_{mi}c_{ji}$. The difference between the replication operation and the DB-object retrieval for a read operation is that in the first case the entire DB-object is transferred from one site to another while in the second case the entire DB-object remains in its site and only a fraction of it is transmitted. This difference is established in the definitions of l_{km} and d_{mi}.

5.6 Cost of DB-Object Storage

The fourth term models the cost for storing DB-objects in the sites. This is calculated multiplying the storage cost of each site by the storage space required by all the DB-objects stored in the site.

6 INSTANCE COMPRESSION THROUGH CLUSTERING

6.1 General Description

Regarding the DDB distribution design problem, when a problem instance has repetitive operations, it is possible to transform it into an instance with fewer operations, since repetitive operations are represented by similar rows in the access matrix. Therefore, such operations can be considered as a single operation that is issued with larger frequency. The reduction level that such transformation can yield is directly proportional to the amount of repetitive operations. Such transformation is a relation on the problem instances set, which associates a given instance to a smaller instance. Figure 3 depicts a transformation r of this type, where I is the instance set of the DDB distribution design problem, and i is the original instance, I' is a subset of I, and i' is the compressed instance.

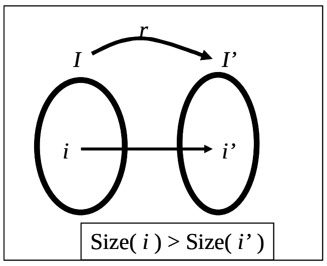


Fig. 3. Instance transformation

The goal of the transformation is to yield a reduction in the amount of resources needed to solve the original instance, without significantly reducing the quality of its solution. In order to preserve the solution quality, the transformation must *capture the operations access pattern* of the original instance. Data mining techniques allow to carry out this type of tasks, thus we describe a method that uses clustering techniques. To present a detailed description of the method, several concepts will be introduced first.

The binary vector that indicates from which sites an operation is issued is called *access pattern*. The access pattern matrix P_{ki} is constructed in such a way that, for every k and i, $P_{ki} = 1$ if and only if $f_{ki} \neq 0$. Figure 4 shows the access frequency matrix f_{ki} of the operations involved in the example of Figure 2 and the resulting access pattern matrix P_{mi}. The marked cells show examples of the transformation process when the access frequency is zero (squares) and larger that zero (circles).

$$
\begin{array}{c}
\text{Sites} \\
\text{Operations}\;
\begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 2 & 3 & 0 \\
2 & 0 & \boxed{0} & 6 \\
3 & 3 & 4 & 0 \\
4 & 0 & 0 & \textcircled{3}
\end{array} \\
f_{ki}
\end{array}
\qquad\qquad
\begin{array}{c}
\text{Sites} \\
\text{Operations}\;
\begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 1 & 1 & 0 \\
2 & 0 & \boxed{0} & 1 \\
3 & 1 & 1 & 0 \\
4 & 0 & 0 & \textcircled{1}
\end{array} \\
P_{ki}
\end{array}
$$

Fig. 4. Obtaining the access pattern matrix P_{ki} from f_{ki}

All the operations that have the same access pattern are considered as a single operation of the compressed instance, and all the DB-objects required by the grouped operations constitute a single DB-object of the compressed instance.

6.2 Adjustment of the Instance Formulation

Once the operation and DB-object groups are created, it is necessary to adjust the access frequencies to the groups, the operation selectivity to each group and the group sizes of the compressed instance. The adjustment process is carried out as follows. Given the original instance i, matrix f_k, operation selectivities s_{km} and DB-object sizes b_m, then the access frequency of each grouped operation c at site i is given by

$$f'_{ci} = \sum_{k \in QueryCluster(c)} f_{ki}, \qquad \forall_{\,ci}, \tag{2}$$

the size of DB-object group c is given by

$$b'_c = \sum_{m \in DB\_objectCluster(c)} b_m, \qquad \forall_{\,c}, \tag{3}$$

and the selectivity of operation k to DB-object group c is given by

$$s'_{kc} = \frac{\sum_{m \in DB\_objectCluster(c)} s_{km} * b_m}{\sum_{m \in DB\_objectCluster(c)} b_m}, \qquad \forall_{k,c}. \tag{4}$$

Figure 5 describes the compression process and the formulation adjustment, and shows the operation access f_{ki}, the access pattern P_{ki} and the grouped operation access f'_{ci} matrices.

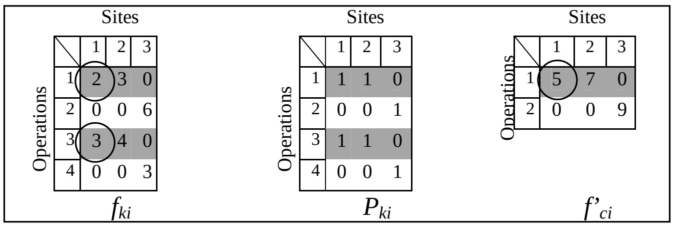


Fig. 5. Compression process

Since operations #1 and #3 have the same access pattern (dark rows in matrices f_{ki} and P_{ki}), they are integrated into operation #1 (dark row in matrix f'_{ci}) of the compressed instance. Similarly operations #2 and #4 of the original instance are integrated into operation #2 of the compressed instance. The sum of the two encircled frequencies, corresponding to operations #1 and #3, becomes the frequency of operation #1 of the compressed instance, which is shown encircled. The rest of the frequencies of the compressed instance are calculated similarly.

6.3 Clustering Algorithm

The input to the algorithm is matrix f_{ki}. To each binary pattern of the access pattern matrix a decimal code is assigned, which is used to identify the group to which the operation belongs. Figure 6 shows the detailed description of the algorithm used. The decimal coding is carried out in line 5 of the algorithm. If for a given operation the value of the cell corresponding to a site in the frequency matrix is larger than zero, the power of 2 corresponding to the site position is added to *Code*. When all the sites are accounted, at line 8 the operation is associated to the group denoted by *Code*.

7 INSTANCE COMPRESSION THROUGH PROGRESSIVE SAMPLING

7.1 General Description

This section presents an instance compression method that uses a data mining technique known as progressive sampling [21, 22]. The basic idea consists of generating a representative sample (subset) of the instance operations and solving the problem instance that involves only the operations in the sample; thus the new problem is a reduced version of the original one. Notice that any feasible solution to the reduced instance i' is also a feasible solution to the original instance i, but the optimal solution to i' is generally a suboptimal solution to i, whose quality (closeness to the optimum) depends on the representativeness of i'.

For very large problem instances, if the number of sites is much smaller than the number of operations, the access pattern of operations to data will appear repeatedly

```
Algorithm CODE_GROUPING
Function: Determine the decimal value corresponding to the binary coding of the access
pattern of each operation, which is used as identifier of the group to which the operation is
assigned.
Input: Matrix $f_{ki}$ of emission frequencies of operations ($k$) at sites ($i$).
Structures:
Code: Decimal coding of the binary access pattern of each operation.
$Card_l$: Elements of the group $l$.
Output: $Group_k$ (group to which operation $k$ is assigned)
1    For each operation $k \in f_{ki}$
2    $Code \leftarrow 0$
3      For each site $i \in f_{ki}$
4          If ($f_{ki}$>0)
5              $Code \leftarrow Code + 2^i$
6          End if
7    End for
8    $Group_k \leftarrow Code$
9    $Card_{Code} \leftarrow Card_{Code} + 1$
10   End for
```

Fig. 6. Code grouping algorithm

in the frequency matrix. Since the access pattern of an operation is a binary vector of size ns (where a 1 in the i^{th} place of the vector means that the operation accesses data in site i), the maximum number of different access patterns is 2^{ns}. Because of the exponential growth of this number, only instances with a maximum of 15 sites will be considered. In these circumstances it is reasonable to suppose that a progressive sampling process can identify the access pattern of operations using relatively small samples, learning the optimal sample size.

7.2 Similarity Metric

The process has as input the original instance and yields the compressed instance. A sample with the optimal size is called a minimal representative sample. The concepts introduced below constitute the basis for the proposed learning process.

Tuple cluster. Let $C = \{c_1, c_2, \ldots, c_n\}$ denote a partition of relation R tuples. Each element of C is a tuple cluster, which for simplicity will be called cluster.

Cluster access probability. Ratio of the cluster size to the relation size.

$$p_{c_i} = \frac{\text{Size}(c_i)}{\sum_{j=1}^{n} \text{Size}(c_j)} \tag{5}$$

Size(c_i) is the size of cluster c_i and n is the number of clusters.

Operation access probability. Ratio of the number of operation accesses to the number of accesses to the relation.

$$p_{q_i} = \frac{\mathrm{Acc}(q_i)}{\sum_{j=1}^{nq} \mathrm{Acc}(q_j)} \tag{6}$$

$\mathrm{Acc}(q_i)$ is the number of accesses of operation q_i and nq is the number of operations.

Joint operation probability. Operation access probability multiplied by the sum of the probabilities of access to the clusters used by the operation.

$$P_{q_i} = p_{q_i} \cdot \sum p_{c_j}, \qquad \forall c_j \in q_i \tag{7}$$

Relevant operations of a sample. Sample operations such that their joint operation probabilities is larger than or equal to $\lambda = P_{\min} + \alpha(P_{\max} - P_{\min})$ where $P_{\min}$ and $P_{\max}$ are the minimum and maximum joint probabilities of the sample operations, and $\alpha \in (0,1)$ is a factor used for setting the relevance boundary. The set of all relevant operations of a sample is denoted by RS.

Sample Representativeness. Average joint probability of the relevant operations.

$$r_{m_i} = \overline{P}(RS), \tag{8}$$

where $\overline{P}(RS) = \frac{\sum_{q_i \in RS} P_{q_i}}{\|RS\|}$.

Similarity between two samples m_a and m_b. It is defined as 1 minus the absolute value of the difference of their representativenesses.

$$\mathrm{sim}(m_a, m_b) = 1 - \mid r_{m_a} - r_{m_b} \mid \tag{9}$$

7.3 Learning the Size of the Minimal Representative Sample (MRS)

In this section the basic progressive sampling algorithm is described, which is used in the instance transformation process.

The sampling program $S = \{n_0, n_1, n_2, \ldots, n_k\}$ specifies the sizes of the samples considered in the process. A geometric program is defined as $S = \{n_0 k^n | n_0$ and k are fixed values and $n = 0, 1, 2, 3, \ldots\}$. For evaluating the sampling program $S = \{n_0, n_1, n_2, \ldots, n_k\}$, the algorithm in [22] is applied, which is shown in Figure 7.

The algorithm progressively extracts random samples of the programmed sizes from the set of the instance operations, using the random sequential sampling technique proposed in [24]. This technique belongs to the most utilized ones in data mining for extracting samples from large databases [25]. Sample extraction is a crucial factor for guaranteeing the method scalability. The algorithm determines the sample representativeness and stops when this indicator remains consistently in the interval $[R-\delta, R+\delta]$, where R is a fixed value that is determined in the process and

```
Algorithm PROGRESSIVE_SAMPLING

Function: Determine the size of the minimal representative sample.
Input:
$f_{ki}$ : Matrix of emission frequencies of operations ($k$) from the sites ($i$).
$S = \{n_0, n_1, n_2, \ldots, n_k\}$: Sampling program.

Structures:
$i = 1, 2, \ldots, k$ : Sampling counter.
$n$: Size of the sample to be extracted.
$M$: Structure for storing the extracted sample.
$r$: Sample representativeness.

Functions:
Generate_Sample($f_{ki}$, $M$, $n$)
    Extracts from $f_{ki}$ a random sample $M$ of operations of size $n$, using the random sequential
    sampling technique [24].
Representativeness($M$)
Determines the representativeness of sample $M$.
Output:
$N$: Size of the minimal representative sample.
    $i \leftarrow 0$
    $n \leftarrow n_i$
    Generate_Sample($f_{ki}$, $M$, $n$)
    $r \leftarrow$ Representativeness($M$)
    While r does not converge
        $i \leftarrow i + 1$
        $n \leftarrow n_i$
        Generate_Sample($f_{ki}$, $M$, $n$)
        $r \leftarrow$ Representativeness($M$)
    End while
    $N \leftarrow n$
Return ($N$)
```

Fig. 7. Progressive sample algorithm

δ is some given tolerance. Considering the definition of similarity, it converges to 1 if and only if the samples representativeness converges to a fixed value. Therefore, the process can be stopped when the similarity stabilizes in the interval $[1-\varepsilon, 1]$ for a given tolerance ε. Since this mechanism is simpler to implement, it was chosen for our approach. The last sample of the process is called Minimal Representative Sample (MRS). Figure 8 shows the learning curve that is generated as the process proceeds. The process described previously is an inductive mechanism that learns the MRS size.

The compression process is carried out adjusting the instance formulation so as to include only the operations of the MRS and the sites and DB-objects related to these operations. As a result of the process, a compressed or transformed instance is generated.

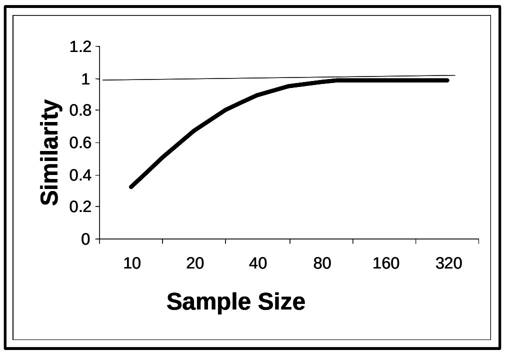


Fig. 8. Learning curve

8 EVALUATION OF THE COMPRESSION GROUPING METHOD

For evaluating the compression method through grouping two experiments were conducted. In the first one the effect of the type of user access on the reduction level and solution quality generated by the method was evaluated. In the second one the efficiency of the method was evaluated. This factor is crucial for increasing the performance of the algorithms used for solving the distribution design problem. For carrying out the experiments 9 test cases were generated. Each case includes 30 random instances which have the same characteristics. Table 4 shows the characteristics used for each case and includes the case ID, the number of DB-objects, sites and operations, the instance size, and the ratio of operations to sites.

8.1 Reduction Degree and Solution Quality

The purpose of this experiment is to obtain evidence on the reduction degree and the solution quality that are achieved as a result of the instance compression using grouping. Each instance of the test cases was solved using an exact algorithm and was compressed using a grouping algorithm. The transformed instance was solved using an exact algorithm and the DB-object location that corresponds to the optimal solution of the compressed instance was determined. Finally, the value of this distribution was calculated using the objective function of the original instance, which was compared against the optimal solution of the original instance, and the error percentage was calculated. For each case, the sizes of the 30 original instances and the 30 compressed instances were added, and the overall reduction percentage was calculated as well as the average error percentage corresponding to the results of the 30 instances.

The test cases used were generated randomly with different access probabilities of the users to the sites. The results obtained were compared against those obtained from applying two other transformations. Each of these transformations is similar to the one proposed, but they use the agglomerative hierarchical grouping algorithm

	Characteristics				
Case	RD (O)	Sites (S)	Operations (Q)	Size in bytes	Q/S
C_1	100	3	100	86 060	33
C_2	200	5	200	338 560	40
C_3	500	7	500	2 062 252	71
C_4	1 200	15	1 200	11 823 420	80
C_5	1 000	10	1 000	8 172 480	100
C_6	1 500	15	1 500	18 379 020	100
C_7	2 000	20	2 000	32 665 760	100
C_8	3 000	20	3 000	72 997 760	150
C_9	4 000	20	4 000	129 329 760	200

Table 4. Cases for evaluating compression using grouping

reported in [26]. In one case only the DB-objects with similarity 1 are grouped. This algorithm is called level 1 and is denoted by A1. In the other case the DB-objects are grouped if their similarity equals 1 or the closest value to 1 present in the matrix of similarity between DB-objects. This algorithm is called level 2 and is denoted by A2. The coded grouping algorithm proposed in this work is denoted by AC. In the experiment the method effectiveness was investigated for different access probabilities, and it was expected that algorithms AC and A1 achieved the same effectiveness since they generate the same grouping for a given instance.

Table 5 shows the experimental results, where the first column indicates the case solved (C). For each case there exist three rows that show the results obtained using algorithms AC, A1 and A2. The results are grouped by columns according to the access probability of the operations to the sites, used for generating the case instances. For each probability two columns are included: the first shows the average error percentage (% E) obtained when solving the 30 instances of each case, and the second shows the average reduction percentage (% R) attained by their compression.

Figures 9 and 10 permit to view the effect of the type of user access on the solution quality and the reduction degree attained using the proposed algorithm (AC).

In these figures it is perceived that as the access probability of the users to the sites increases, the method effectiveness decreases. However, in the low probability region its performance remains high. In an environment like the Internet, in which users access from only 20 % of the sites, the largest instances (case C_5) are solved with a 65 % reduction at a minimal cost in solution quality. Figure 9 reveals that the solution quality obtained with the method is very high (the error percentage never reaches 1 %), independently of the probability of user access to the sites. It is observed in Table 5 that similar results are obtained when algorithm A1 is used, which corresponds with the expectations. When algorithm A2 is used, the approach to the optimal solution occurs generally from below, and for larger instances (case C_5) the maximal error percentage is -7.86 %. The proposed method is highly precise, but its reduction degree depends on the repetition degree of the operation patterns.

		Probability 10 %		Probability 20 %		Probability 30 %		Probability 40 %	
C	A	% E	% R	% E	% R	% E	% R	% E	% R
C_1	AC	0.00	99	0.00	99	0.01	99	0.08	98
C_1	A1	0.00	99	0.00	99	0.01	99	0.08	98
C_1	A2	−25.00	99	−30.27	99	−17.23	99	−10.22	99
C_2	AC	0.01	97	0.05	98	0.03	97	0.22	96
C_2	A1	0.01	99	0.05	98	0.03	97	0.22	96
C_2	A2	−13.03	99	−4.84	98	−10.11	98	−3.08	97
C_3	AC	0.01	99	0.12	97	0.15	95	0.32	93
C_3	A1	0.01	99	0.12	97	0.15	95	0.32	93
C_3	A2	−8.58	99	−11.13	98	−1.01	96	−9.72	95
C_4	AC	0.07	98	0.20	93	0.24	83	0.00	68
C_4	A1	0.07	98	0.20	93	0.24	83	0.24	68
C_4	A2	−2.91	98	−2.97	94	−2.70	83	−4.07	69
C_5	AC	0.10	94	0.03	65	0.07	24	0.08	8
C_5	A1	0.10	94	0.03	65	0.07	24	0.08	8
C_5	A2	−7.86	94	0.03	65	−1.57	24	0.09	8

Table 5. Evaluation results of the reduction degree and solution quality

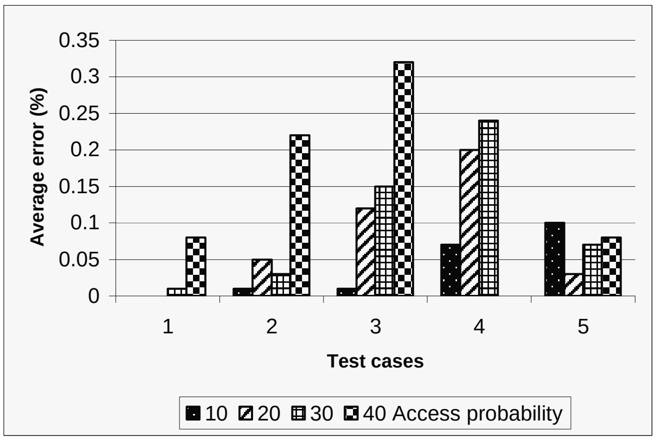


Fig. 9. Effect of access probability on solution quality

8.2 Efficiency of Grouping Method

The purpose of this experiment is to evaluate the efficiency of the proposed grouping algorithm (AC) with respect to that of the other algorithms considered (A1 and A2).

Compression was carried out on each of the instances of the test cases, and the time expended on grouping was recorded. Tests were conducted for each of the algorithms considered, and in each transformation the number of groups generated was recorded. The instances were generated considering an access probability of

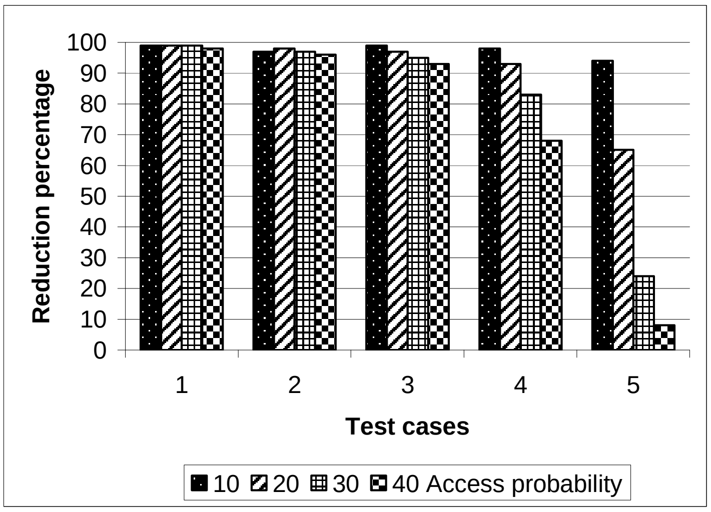


Fig. 10. Effect of access probability on reduction degree

20 %, which corresponds to the situation where the method showed good performance.

Table 6 shows the experiment results, where the first column indicates the case solved (C). Columns 3, 4 and 5 show the average execution time expended in instance grouping for the case. Columns 6, 7 and 8 show the average number of groups generated. For each case there exist two rows: the first presents average values and the second presents their corresponding standard deviations.

As revealed in Figure 11 the proposed algorithm (AC) shows a clear superiority over the hierarchical grouping algorithms. A remarkable characteristic of the AC algorithm is that as the instance size grows, the performance difference grows larger. These results are considered satisfactory, since they provide experimental evidence on the ability of compression using grouping to solve problems of much larger size.

9 EXPERIMENTAL EVALUATION OF THE TRANSFORMATION THROUGH PROGRESSIVE SAMPLING

For evaluating the compression method through progressive sampling, two types of experiments were conducted. In the first the effect of the type of user access on the reduction degree and the solution quality produced by the method are analyzed. In the second the difference of the method performance is studied when fixed and variable DB-object sizes are considered.

		Average Execution Time			Number of Groups		
C		AC	A1	A2	AC	A1	A2
C_1	a	0.0001	0.0594	0.1932	7	6	2
C_1	d	0.0000	0.0208	0.0703	1	0	2
C_2	a	0.0001	0.0718	0.2150	22	22	13
C_2	d	0.0000	0.0141	0.0568	2	2	6
C_3	a	0.0001	0.2842	0.8280	67	67	65
C_3	d	0.0000	0.0072	0.3764	5	5	4
C_4	a	0.0001	1.6344	3.7374	695	696	692
C_4	d	0.0000	0.0490	0.2465	19	19	21
C_5	a	0.0001	1.0406	2.4468	246	247	244
C_5	d	0.0000	0.0176	0.3739	9	9	8
C_6	a	0.0062	2.5406	7.0904	816	817	811
C_6	d	0.0085	0.0420	1.2461	9	9	12
C_7	a	0.0001	5.7410	13.7902	1 609	1 606	1 615
C_7	d	0.0000	0.4639	2.5599	21	28	24
C_8	a	0.0126	14.2468	28.0094	1 863	2 263	2 257
C_8	d	0.0130	2.1548	1.9666	889	11	9
C_9	a	0.0062	25.2842	48.3376	2 889	2 900	2 897
C_9	d	0.0085	1.7786	4.7664	24	24	26

Table 6. Results of the efficiency evaluation

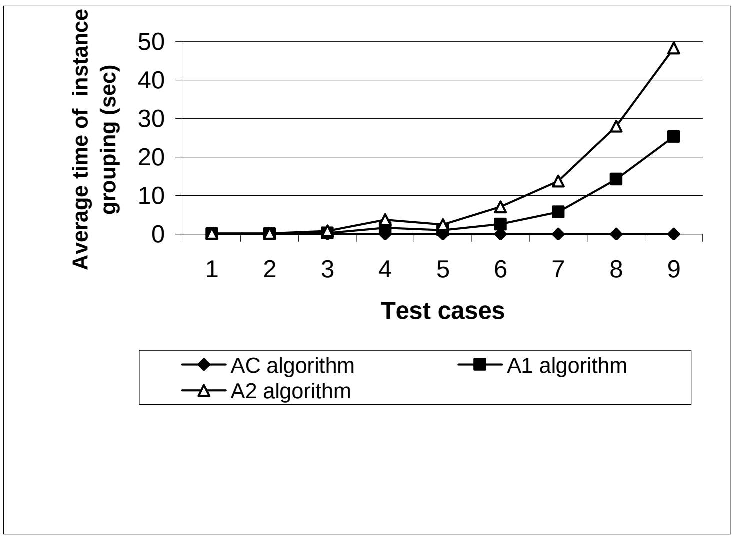


Fig. 11. Efficiency of the grouping algorithms

In the compression process the samples are generated using the random sequential sampling method described in [24]. This method is very efficient, so it is frequently used in data mining for sampling large databases [25].

For carrying out the experiments 6 test cases were generated. Each case includes 30 random instances which have the same characteristics. Table 7 describes the characteristics of each case and includes an identifier for the case, the number of DB-objects, sites and operations, the instance size, and the operations to sites ratio.

	Characteristics				
Case	Objects (O)	Sites (S)	Operations (Q)	Size in bytes	Q/S
C_1	20	3	100	19 820	33
C_2	20	5	200	42 620	40
C_3	30	7	300	93 252	42
C_4	50	10	1 000	492 680	100
C_5	20	3	1 000	192 620	333
C_6	40	6	2 000	754 252	333

Table 7. Test cases for evaluating the transformation through progressive sampling

9.1 Reduction Degree and Solution Quality

This experiment provides evidence on the reduction degree and the solution quality produced when transforming instances through progressive sampling. Each instance of the test cases was solved by an exact algorithm and was compressed using the proposed method. The compressed instance was solved by an exact algorithm and the location of the DB-object that corresponds to the optimal solution of this instance was determined. Finally, the value of this distribution was calculated using the objective function of the original instance, and the error and reduction percentages were calculated. After solving each test case, the sizes of the 30 original and compressed instances was added, and the overall reduction percentage was calculated, as well as the average error percentage from the 30 results.

Test cases with different probabilities of user access to sites were used. The results obtained were compared against those resulting from the application of a transformation that uses a modified similarity metric. The modification consists of using as similarity metric the access probability to operations instead of the joint probability. The metric based on the joint probability proposed in this work, defined by expression (7) in Section 7.2, is denoted by m_1. The metric based on the access probability to operations, defined by expression (5) in Section 7.2, is denoted by m_2. In the experiment the efficiency of the method was investigated for different access probabilities.

Table 8 shows the experiment results, where the first column indicates the case solved (C). For each one two rows of results are shown: the first presents the results obtained when metric m_1 is used, and second presents those obtained using metric m_2. The results are grouped according to the access probability of operations to sites that was used for generating the case instances. For each access probability two columns are included: the first presents the average error percentage (% E) obtained by solving the 30 instances of each case, and the second presents the average reduction percentage (% R) attained through compression.

		Probability 10 %		Probability 20 %		Probability 30 %		Probability 40 %	
C		% E	% R	% E	% R	% E	% R	% E	% R
C_1	m_1	73	65	79	72	33	73	2	72
C_1	m_2	220	78	161	73	18	73	4	74
C_2	m_1	92	70	71	85	9	85	1	84
C_2	m_2	65	88	59	83	8	83	1	84
C_3	m_1	32	92	40	89	7	88	1	88
C_3	m_2	34	93	46	89	7	89	1	89
C_4	m_1	18	93	8	96	2	96	53	96
C_4	m_2	15	96	11	96	2	96	53	96
C_5	m_1	12	97	8	96	21	97	84	97
C_5	m_2	11	94	9	96	21	97	81	97
C_6	m_1	5	98	7	98	83	98	211	98
C_6	m_2	6	97	6	98	83	98	210	98

Table 8. Results on the reduction degree and solution quality

Figures 12 and 13 allows to observe the effect of the type of user access on the solution quality and the reduction degree obtained as a result of the method when the joint metric (m_1) is used.

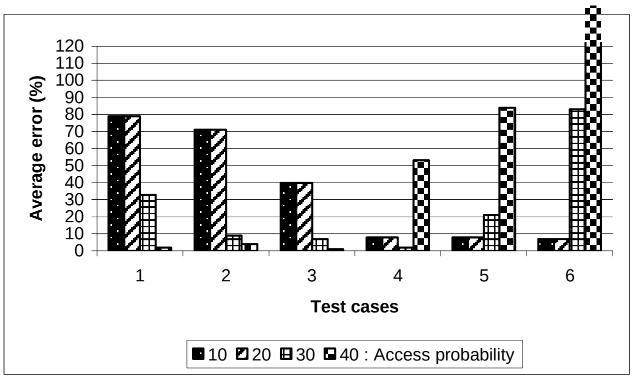


Fig. 12. Effect of access probability on quality

In these figures it is perceived that this transformation also shows its best performance in situations of low access dispersion. The average error percentage is never above 10 %. Figure 13 shows the high reduction degree attained and its growth as the instance size increases.

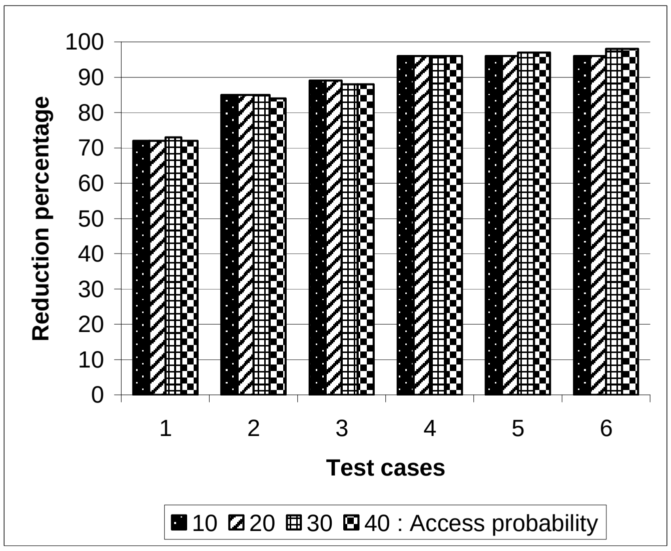


Fig. 13. Effect of access probability on reduction degree

From Table 8 it can be observed that, if the metric based on access probability to operations (m_2), the performance is generally equal to the one obtained with the joint probability metric (m_1). The evidence generated by the experiment shows that it is possible to carry out compression transformations of the instances with high reduction degrees and with an acceptable reduction in solution quality.

9.2 Effect of DB-object Size

Since the compression method through progressive sampling assumes that the data are already grouped, in this experiment the effect of DB-object size on the method efficiency is evaluated. For each of the situations in which the method showed the best performance (20 % access probability), the test cases were first solved considering only instances with fixed size DB-objects and afterwards considering variable size DB-objects. In both tests the observed error percentage average and the overall reduction degree were determined. The joint probability was used for defining the similarity metric.

Table 9 shows the experiment results, where the first column indicates the case solved (C). The results are presented in two-column groups. The first column pair corresponds to cases for which the DB-object size is fixed, while the second pair shows the results obtained for cases with variable DB-object size. For each type of size considered, the first column shows the average error percentage (% E) obtained by solving the 30 instances of each case, and the second column shows the average reduction percentage (% R) attained by their transformation. As previously noted, the access probability of operations to sites is 20 % for all the cases. As can be perceived in Table 9, the method shows virtually the same performance with DB-objects of fixed or variable size. These results provide experimental evidence on the insensibility of the method efficiency with respect to DB-object size.

	Fixed Size		Variable Size	
C	% E	% R	% E	% R
C_1	79	72	120	74
C_2	71	85	68	85
C_3	40	89	47	90
C_4	8	96	6	96
C_5	8	96	7	96
C_6	7	98	3	98

Table 9. Effect of DB-object size

10 CONCLUSIONS AND FUTURE WORK

This paper shows the feasibility of the proposed approach to solve large scale instances of the distribution design problem. The general strategy includes, unlike other approaches, an additional feature regarding compression of the instance to be solved. A compression method consists of the application of a transformation of the original instance into a new instance that requires fewer resources to solve it than the original. The goal of the transformation is to obtain a reduction in the amount of resources needed to solve the original instance, without significantly reducing the quality of its solution. In order to preserve the quality solution, the transformation condenses the access pattern of the original instance using data mining techniques.

A set of experiments, using instances configured with typical access patterns found on the Internet, were conducted for evaluating quantitatively the size reduction that can be achieved and its effect on the solution quality. For instances with access probability of 10 % and 20 %, the resource reduction is at least 65 % with the clustering compression method. The minimal and maximal reductions are 65 % and 99 %, which constitute a considerable reduction of resources. The error percentage of the solution varies from 0.10 % to 3.20 %, which shows that the degradation is relatively small.

The progressive sampling method shows an interesting property, as the instance size increases the reduction level increases too. Additionally, the resource reduction with this method is always larger than 90 %. In contrast the precision is worse than the one attained with the grouping method. Therefore, the feasibility of reducing the resources required to solve large scale instances at the expense of a reasonable loss in the solution quality, is demonstrated.

The learning method was evaluated with two different similarity metrics and considering variable size data DB-objects, obtaining similar results. The efficiency of the clustering algorithm was compared against two hierarchical clustering algorithms, and our algorithm consistently outperformed the others. Given the encouraging results with both methods, we are currently working on the design and implementation of efficient compression methods using other data mining techniques.

REFERENCES

[1] Garey, M. R.–Johnson, D. S.: Computer and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman, New York 1979.

[2] Papadimitriou, C.—Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Publications 1998.

[3] Barr, R. S.—Golden, B. L., Kelly, J.—Steward, W. R.—Resende, M.: Guide-lines for Designing and Reporting on Computational Experiments with Heuristic Solving Large Scale Instances of the Distribution Design Problem Using Data Mining 1025 Methods. Proceedings of International Conference on Metaheuristics for Optimization, pp. 1–17, Kluwer Publishing, Norwell, MA 2001.

[4] Michalewicz, Z.—Fogel, D. B.: How to Solve It: Modern Heuristics. Springer Verlag 1999.

[5] Pérez, J.—Pazos, R. A.—Frausto, J.—Romero, D.—Cruz, L.: Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm. Lectures Notes in Computer Science, Vol. 1793, pp. 75–81, Springer-Verlag 2000.

[6] Pérez, J.—Pazos, R. A.—Frausto, J.—Rodríguez,G.—Cruz, L.—Fraire, H.—Mora, G.: Self-Tuning Mechanism for Genetic Algorithms Parameters, an Application to Data-Object Allocation in the Web. Lectures Notes in Computer Science, Vol. 3046, pp. 77–86, Springer-Verlag 2004.

[7] Ceri, S.—Navathe, S.—Wiederhold, G.: Distribution Design of Logical Database Schemes. Proc. IEEE Transactions on Software Engineering, Vol. SE-9, 1983, No. 4, pp. 487–503.

[8] Apers, P.: Data Allocation in Distributed Database Systems. ACM Transactions on Database Systems, Vol. 13, 1988, No. 3, pp. 263–304.

[9] Ozsu, M.—Valduries, P.: Principles of Distributed Database Systems. Englewood Cliffs, N. J., Prentice-Hall 1999.

[10] Johansson, J.—March, S.—Naumann, J.: The Effects of Parallel Processing on Update Response Time in Distributed Database Design. Proc. of the 21st Int. Conf. on Information Systems, 2000, pp. 187–196.

[11] Huang, Y. F.—Chen, J. H.: Fragment Allocation in Distributed Database Design. Journal of Information Science and Engineering, Vol. 17, 2001, No. 3, pp. 491–506.

[12] Visinescu, C.: Incremental Data Distribution on Internet-Based Distributed Systems: A Spring System Approach. Master of Mathematics in Computer Science Thesis, supervised by Tamer Ozsu, University of Waterloo, 2003.

[13] Baiao, F.—Mattoso, M.—Zaverucha, G.: A Distribution Design Methodology for Objects DBMS. Distributed and Parallel Databases. Vol. 16, 2004, No. 1, pp. 45–90. Kluwer Academic Publishers 2004.

[14] Papadomanolakis, S.—Ailamaki, A.: Autopart: Automating Schema Design for Large Scientific Databases Using Data Partitioning. Proceedings of the 16^{th} IEEE Conference on Scientific and Statistical Database Management, p. 383, 2004.

[15] Agrawal, S.—Surajit, Ch.—Lubor, K.—Arunprasad, M.—Vivek, N.—Manoj, S.: Database Tuning Advisor for Microsoft SQL Server 2005. Proceedings of the Thirtieth International Conference on Very Large Data Bases 2004, Toronto, Canada, Morgan Kaufmann 2004, pp. 1110–1121.

[16] TPC Benchmarks, http://www.tpc.org.

[17] Zilio, D.—Rao, J.—Lightstone, S.—Lohman, G.—Storm, A.—Garcia-Arellano, C.—Fadden, S.: DB2 Design Advisor: Integrated Automatic Physical Database Design. Proceedings of the Thirtieth International Conference on Very Large Data Bases 2004, pp. 1087–1097, Toronto, Canada, Morgan Kaufmann 2004.

[18] Chaudhuri, S.—Gupta, A.—Narasayya, V.: Compressing SQL Workloads. SIG-MOD Conference 2002, pp. 488–499.

[19] Halkidi, M.—Batistakis, Y.—Vazirgiannis, M.: On Clustering Validation Techniques. Journal of Intelligent Information Systems. Vol. 17, 2001, No. 2, pp. 107–145, Kluwer Academic Publishers 2001 .

[20] Berkhin, P.: Survey of Clustering Data Mining Techniques. Accrue Software. 2002.

[21] Stamatopoulos, C.: Observations on the Geometrical Properties of Accuracy Growth in Sampling with Finite Populations. FAO Fisheries Technical Paper 388 (ISSN 0249-9345), Food and Agriculture Organization of the United Nations, Rome 1999.

[22] Provost, F.—Jensen, D.—Oates, T.: Efficient Progressive Sampling. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM 1999, pp. 23–32.

[23] Fraire, H.: Una Metodología para el Diseño de la Fragmentación y Ubicación en Grandes Bases de Datos Distribuidas. Ph. D. thesis. Centro Nacional de Investigacin y Desarrollo Tecnolgico (CENIDET), Cuernavaca, Morelos, Mexico 2005.

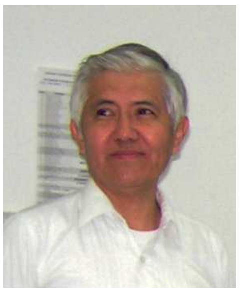
[24] Vitter, J.: An Efficient Algorithm for Sequential Random Sampling. ACM Transactions on Mathematical Software, Vol. 13, 1987, No. 1, pp. 58–67, ACM Press, New York, USA.

[25] Jermaine, Ch.—Pol, A.—Hwang, S.: Online Maintenance of Very Large Samples. Proc. of the 2004 SIGMOD Int. Conf. on Management Data. 2004.

[26] Beeferman, D.—Berger, A.: Agglomerative Clustering of a Search Engine Query Log. KDD 2000, pp. 407–416.

Héctor Fraire received the Ph. D. degree in computer science from the Centro Nacional de Investigación y Desarrollo Tecnológico Cuernavaca (Mexico) in 2005. His scientific interests include metaheuristic optimization and machine learning.

Joaquín Pérez received the Ph. D. degree in computer science from the Instituto Tecnológico y de Estudios Superiores de Monterrey (Mexico) in 1999. His scientific interests include distributed databases design and metaheuristic optimization.



Rodolfo Pazos received the Ph. D. degree in computer science from the University of California at Los Angeles (USA) in 1983. His scientific interests include distributed databases design and natural language processing.

Juan Frausto received the Ph. D. degree in electrical engineering from the Institut National Polytecnique de Grenoble (France) in 1982. His scientific interests include computational methods in electrical engineering, optimization and simulation.

David Romero received the Ph. D. degree in applied mathematics from L'Université Scientifique et Medicale de Grenoble (France) in 1978. His scientific interests include complex systems and mathematical modeling and simulation.

Laura Cruz received the Ph. D. degree in computer science from the Centro Nacional de Investigación y Desarrollo Tecnológico Cuernavaca (Mexico) in 2004. Her scientific interests include metaheuristic optimization and machine learning.