Journal of Computational and Applied Mathematics 259 (2014) 905–913

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Minimizing roundtrip response time in distributed databases with vertical fragmentation

Rodolfo A. Pazos [a,*], Graciela Vázquez [b], José A. Martínez [a], Joaquín Pérez-Ortega [c], Gilberto Martínez-Luna [d]

[a] Instituto Tecnológico de Ciudad Madero, Av. 1o. de Mayo s/No., Col. Los Mangos, Cd. Madero, Tamaulipas, Mexico
[b] ESIME, Instituto Politécnico Nacional, Av. Inst. Politécnico Nal. s/No., Col. Lindavista Zacatenco, México, Mexico City, Mexico
[c] Centro Nacional de Investigación y Desarrollo Tecnológico, Interior Internado Palmira s/No., Col. Palmira, Cuernavaca, Morelos, Mexico
[d] Centro de Investigación en Computación, IPN, Av. Juan de Dios Bátiz, Esq. Miguel Othón de Mendizábal, Zacatenco, Mexico City, Mexico

ARTICLE INFO

Article history:
Received 15 February 2013
Received in revised form 5 September 2013

Keywords:
Distributed database design
Vertical fragmentation

ABSTRACT

The main purpose of this paper is to show the advantage of using a model proposed by us, which minimizes roundtrip response time versus traditional models that minimize query transmission and processing costs for the design of a distributed database with vertical fragmentation. To this end, an experiment was conducted to compare the roundtrip response time of the optimal solution obtained using our model versus the roundtrip response time of the optimal solution obtained using a traditional model. The experimental results show that for most cases the optimal solution from a traditional model yields a response time which is larger than the response time of the optimal solution obtained from our model, and sometimes it can be thrice as large.

1. Introduction

One of the challenges of the application of distributed database (DDB) systems is the possibility of expanding their utilization through the use of the Internet, so widespread nowadays. One of the most difficult problems in DDB systems deployment is the distribution design.

Traditionally, the DDB design problem has been defined as finding relation (tables) fragments and their allocation such that the overall costs incurred by query transmission and processing are minimized. It has been recognized for many years the importance of considering response time in the DDB modeling [1, 2]. Unfortunately, traditional optimization models have not considered response time, as shown in Table 1, which summarizes the most relevant and recent works on fragmentation and the allocation of DDBs with vertical fragmentation.

The second, third and fourth columns of Table 1 show that some works have addressed only the fragmentation problem, many works have only dealt with the fragment allocation problem, and some works have addressed integrally both problems. The fifth column of Table 1 shows that all the previous works have considered transmission, access or processing costs. The seventh and eighth columns of Table 1 show that most works have proposed heuristic algorithms for addressing DDB fragmentation and allocation and only a few have proposed mathematical programming formulations.

In this paper a mathematical programming model is presented (VFA-RT), which describes the behavior of a DDB with vertical fragmentation and permits us to optimize its design taking into account the nonlinear nature of roundtrip response time (query transmission delay, query processing delay, and response transmission delay). In a previous paper by us [1]

* Corresponding author. Tel.: +52 8331676671.
E-mail address: r_pazos_r@yahoo.com.mx (R.A. Pazos).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.09.057

Table 1
Related works on fragmentation and allocation for DDBs with vertical fragmentation.

Works	Problems addressed			Value to minimize		Solution approach	
	Fragmentation	Allocation	Integrated fragmentation + allocation	Transmission or processing costs	Roundtrip response time	Heuristic algorithm	Mathematical programming formulation
Chakravarthy, 94 [7]	✓			✓		✓	
Pérez, 00 [3][a]			✓	✓			✓
Ma, 06 [4]			✓	✓		✓	
Tambulea, 08 [8]		✓		✓		✓	
Karimi, 09 [9]		✓		✓			[b]
Khan, 10 [10]		[c]		✓		[d]	
Sevinc, 10 [11]		✓		✓		✓	
Kamali, 11 [12]		✓		✓		✓	
Goli, 12 [13]	✓			✓		✓	
Song, 13 [14]			[c]	✓	[e]		[b]
Our approach			✓		✓		✓

[a] A previous model of ours.
[b] Problem constraints are enforced in the algorithm.
[c] Relation replication is considered.
[d] Axiomatic game theoretic mechanism.
[e] Times are assumed linearly proportional to transmission and processing costs.

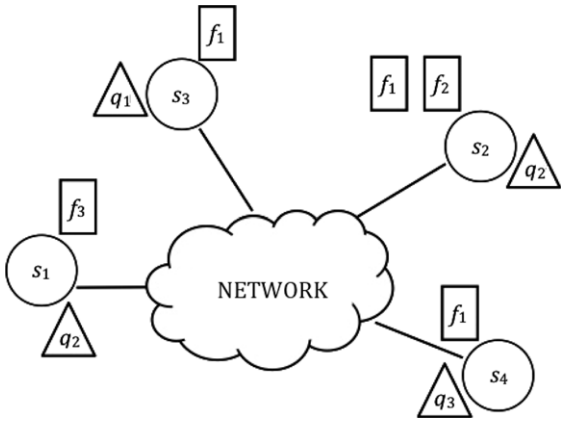

Fig. 1. Example of a distribution design.

we introduced for the first time the VFA-RT model, and in [2] we presented the experimental results of two metaheuristic algorithms (threshold accepting and tabu search) aimed at finding a good metaheuristic algorithm for solving the VFA-RT model. Unlike those papers, now we are interested in experimentally showing the advantage of using the proposed model with respect to traditional models which consider query transmission and processing costs; therefore, this paper also includes a brief description of the DFAR model [3], which considers this kind of costs. Finally, the paper includes a comparative experiment of the VFA-RT and DFAR models (using an exact algorithm for each) and a comparative experiment of an exact algorithm applied to the VFA-RT model versus a heuristic approach [4].

2. Description of the DDB design problem

A distributed database (DDB) is a database (DB) whose data are partitioned into portions that are stored in two or more different nodes, which are interconnected by a communication network. Fig. 1 shows a simple example of a DDB, in which the communication network is represented by a cloud and each network site (node) is represented by a circle tagged with a letter s and a sub-index.

We are interested in a type of data partition called vertical fragmentation, which is defined as follows [5] let R denote a relation (DB table) that has a set of attributes (table columns) $A = \{a_1, a_2, \ldots, a_L\}$; the vertical fragmentation of R consists of a partition of R into several sub-relations $R_1, R_2, \ldots, R_N$ such that each sub-relation is obtained by the following operation:

$$R_i = \Pi_{A_i} R$$

where Π represents the projection operator from the relational algebra, and $A_i \subset A$; additionally, $R_1, R_2, \ldots, R_N$ should be defined so that the original relation R can be reconstructed from its fragments by applying the join operation of the relational algebra to its fragments: $R = R_1$ JOIN R_2 JOIN …JOIN R_N.

Thus, given a relation R, the design of its vertical fragmentation consists of determining sub-relations $R_1, R_2, \ldots, R_N$, such that query processing is optimized with respect to some criterion.

Each network site is a piece of an information system that may execute applications that issue queries to a DDB that is vertically fragmented. Applications that involve queries are denoted by triangles tagged with a sub-indexed letter q. Additionally, each node may include a DB server that permits accessing a local DB that keeps a vertical fragment of a relation. Finally, vertical fragments are represented by boxes tagged with a sub-indexed letter f.

The DDB distribution design problem is defined as follows: given a relation constituted by a set of attributes $A = \{a_1, a_2, \ldots, a_L\}$, a computer communication network that consists of a set of sites $S = \{s_1, s_2, \ldots, s_I\}$, a set of queries $Q = \{q_1, q_2, \ldots, q_K\}$ generated at the network sites, the attributes required by each query, and the generation rates of queries at each site; the problem consists of finding relation fragments and their allocation that minimize the average roundtrip response time. We define roundtrip response time as the time elapsed between the arrival of a query to a network site (query source) and the time when the last bit of the response to the query is received at the query source.

3. Mathematical formulation considering roundtrip response time

In this paper we are proposing the use of a mathematical model (called VFA-RT) that aims at minimizing the roundtrip response time. VFA-RT is a non-linear (binary) integer-programming problem that consists of a non-linear objective function and three groups of constraints (which are linear functions). The objective function and its mathematical derivation are explained in Section 3.1, and the problem constraints are presented in Section 3.2. In this model, $x_{\ell j}$ denotes a decision variable, which equals 1 if attribute ℓ is allocated to site j; otherwise $x_{\ell j}$ equals zero.

3.1. Objective function of the VFA-RT model

The objective function models the average roundtrip response time using three terms: delay of queries incurred by their transmission on the communication lines from their sources to the servers where they are processed, delay of queries incurred by their processing at the servers, and delay of queries responses incurred by their transmission on the communication lines from the servers back to the sources of their corresponding queries. Specifically, the objective function is given by the following expression:

$$z = T_{TQ} + T_{PQ} + T_{TR} \tag{1}$$

where T_{TQ} stands for the average transmission delay of queries, T_{PQ} represents the average processing delay of queries, and T_{TR} is the average transmission delay of query responses.

The average transmission delay of queries is defined as follows:

$$T_{TQ} = \frac{1}{f} \sum_k \sum_i f_{ki} T_{ki}$$

where f_{ki} represents the arrival rate of query k to source site i, $f = \sum_{ki} f_{ki}$ (the summation of the arrival rates of all the queries), and T_{ki} is the average transmission delay of queries k that arrive at source site i. However, given the difficulty for calculating exactly T_{TQ}, the following approximation will be used instead, since $f_{ki} \approx \gamma_{ki}$:

$$T^*_{TQ} = \frac{1}{\gamma} \sum_k \sum_i \gamma_{ki} T_{ki}$$

where γ_{ki} is the emission rate of query k out of source site i, and $\gamma = \sum_{ki} \gamma_{ki}$

By using Little's result [6], the following expression is obtained:

$$\gamma T^*_{TQ} = n = \sum_{ij} n_{ij}$$

where n represents the average number of queries on the transmission lines (either waiting or being transmitted), and n_{ij} is the average number of queries on the transmission line from source site i to server site j. By applying again Little's result to n_{ij}, the following expression is obtained:

$$\gamma T^*_{TQ} = \sum_{ij} \lambda_{ij} T_{TQij} \tag{2}$$

where λ_{ij} represents the arrival rate of queries to the line from i to j, and T_{TQij} is the average transmission delay of queries on the line from i to j.

Additionally, considering that the transmission of queries on the lines can be modeled as an $M/M/1$ queue [6], then the average transmission delay of queries on the line from i to j is given by

$$T_{TQij} = \frac{1}{\mu_Q C_{ij} - \lambda_{ij}} \tag{3}$$

where C_{ij} is the transmission speed of the communication line from node i to node j, and $1/\mu_Q$ is the mean length of queries.

Then, by substituting expression (3) for T_{TQij} into (2), the following expression is obtained:

$$T^*_{TQ} = \frac{1}{\gamma} \sum_{ij} \frac{1}{\mu_Q C_{ij}/\lambda_{ij} - 1}. \tag{4}$$

Additionally, the expression for λ_{ij} in terms of the arrival rates of queries to source sites and the location of attributes in server sites can be calculated as follows:

$$\lambda_{ij} = \sum_k f_{ki} y_{jk} \tag{5}$$

where y_{jk} is a dependent variable ($y_{jk} = 1$ if one or more attributes used by query k are stored in site j, and $y_{jk} = 0$ otherwise).

On the other hand, notice that the summation of the emission rates of all the queries emitted from the source sites (γ) equals the summation of the arrival rates of all the queries that arrive at the servers; therefore

$$\gamma = \sum_j \lambda_{*j}$$

where λ_{*j} represents the arrival rate of queries that arrive at server j, which is defined as follows:

$$\lambda_{*j} = \sum_k \sum_i f_{ki} y_{jk}.$$

Therefore, γ can be calculated as follows:

$$\gamma = \sum_j \sum_k \sum_i f_{ki} y_{jk}. \tag{6}$$

Finally, by substituting (5) and (6) into (4), the following expression is obtained for T^*_{TQ} as a function of the location (y_{jk}) of attributes in server sites:

$$T^*_{TQ} = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \sum_{ij} \frac{1}{\frac{\mu_Q C_{ij}}{\sum_k f_{ki} y_{jk}} - 1}. \tag{7}$$

By carrying out a similar process (whose details are presented in [1]), the following expression can be derived for T^*_{PQ} and T^*_{TR}:

$$T^*_{PC} = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \sum_j \frac{1}{\frac{C_j}{\sum_k \sum_i f_{ki} y_{jk}} - 1} \tag{8}$$

$$T^*_{TR} = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \sum_{ij} \frac{1}{\frac{\mu_R C_{ji}}{\sum_k f_{ki} y_{jk}} - 1} \tag{9}$$

where C_j is the processing capacity of server j, and $1/\mu_R$ is the mean length of query responses.

Finally, an expression for the objective function in terms of the location of attributes can be obtained, by substituting (7)–(9) into (1), which yields the following formula:

$$z = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \left[\sum_{ij} \frac{1}{\frac{\mu_Q C_{ij}}{\sum_k f_{ki} y_{jk}} - 1} + \sum_j \frac{1}{\frac{C_j}{\sum_k \sum_i f_{ki} y_{jk}} - 1} + \sum_{ij} \frac{1}{\frac{\mu_R C_{ji}}{\sum_k f_{ki} y_{jk}} - 1} \right]. \tag{10}$$

3.2. Constraints of the VFA-RT model

Since in this model, attribute replication is not considered; therefore, there is a group of constraints that states that each attribute should be allocated to one site (11). Additionally, each attribute should be allocated to a site that generates at least one query that accesses the attribute (12). These constraints are expressed as follows.

Each attribute must be stored in only one site:

$$\sum_j x_{\ell j} = 1, \quad \forall \ell. \tag{11}$$

Each attribute ℓ must be allocated to a site i where at least one query that involves the attribute is generated:

$$x_{\ell i} \leq \sum_k u_{k\ell} \phi_{ki}, \quad \forall \ell, i \tag{12}$$

where $u_{k\ell}$ is a usage parameter, $u_{k\ell} = 1$ if query k uses attribute ℓ and $u_{k\ell} = 0$ otherwise; $\phi_{ki} = 1$ if $f_{ki} > 0$ and $\phi_{ki} = 0$ if $f_{ki} = 0$.

The following group of constraints forces the value of y_{jk} to 1 when some product $u_{k\ell}x_{\ell j}$ equals 1 (i.e., some attribute ℓ involved in query k is located in server j), and induces y_{jk} to 0 otherwise:

$$ty_{jk} - \sum_{\ell} u_{k\ell}x_{\ell j} \geq 0, \quad \forall j, k \tag{13}$$

where $t =$ number of attributes.

4. Mathematical formulation considering transmission costs

In this subsection a similar model (called DFAR) for the DDB vertical fragmentation and allocation problem is presented. However, DFAR (like traditional models) aims at minimizing transmission costs. This model was also devised by us in 1999, and it will be briefly described here since the details can be found in [3].

4.1. Objective function of the DFAR model

DFAR is a linear (binary) integer-programming problem that consists of a linear objective function and five groups of constraints (which are linear functions). The objective function is defined by expression (14). This function models the transmission, storage and access costs by using four terms: the first term models the transmission costs of the data necessary for satisfying each of the queries that are generated at each site. The second term models the costs incurred during the processing of a query when it needs to access several fragments that are stored in different sites. The third term models the costs incurred for storing fragments in network sites. The fourth term models the transmission costs incurred by migrating attributes from one site to another:

$$\min\ z = \sum_{k}\sum_{i} f_{ki} \sum_{\ell}\sum_{j} u_{k\ell}l_{k\ell}c_{ij}x_{\ell j} + c_1 \sum_{i}\sum_{k}\sum_{j} f_{ki}y_{kj} + c_2 \sum_{j} w_j + \sum_{\ell}\sum_{i}\sum_{j} a_{\ell i}c_{ij}d_{\ell}x_{\ell j} \tag{14}$$

where $x_{\ell j}$ and y_{kj} denote variables, whose meaning is the same as those for the VFA-RT model; additionally w_j is a variable whose value depends on those of x's, specifically $w_j = 1$ if there exists one or more attributes stored in site j and $w_j = 0$ otherwise; f_{ki} is the emission frequency of query k from site i; $u_{k\ell}$ is a usage parameter, $u_{k\ell} = 1$ if query k uses attribute ℓ and $u_{k\ell} = 0$ otherwise; $l_{k\ell}$ is the number of communication packets required for transporting attribute ℓ required by query k; c_{ij} is the cost incurred for transmitting a packet between sites i and j; c_1 is the cost incurred for accessing each fragment for satisfying a query (this is relevant when the query involves data stored in two or more fragments); c_2 is the cost incurred for storing a fragment in a site (e.g., the cost for keeping a copy of the primary key in each fragment); $a_{\ell i}$ is a parameter that indicates the previous allocation of an attribute, $a_{\ell i} = 1$ if attribute ℓ was previously allocated to site i and $a_{\ell i} = 0$ otherwise; and d_{ℓ} is the number of packets required for moving attribute ℓ to another site if necessary.

4.2. Constraints of the DFAR model

The set of DFAR constraints consists of those of the VFA-RT model plus the following two groups of constraints.

The following group of constraints forces the value of w_j to 1 when any x_j equals 1, and it induces w_j to 0 otherwise:

$$tw_j - \sum_{\ell} x_{\ell j} \geq 0, \quad \forall j \tag{15}$$

where $t =$ number of attributes.

The sum of the sizes of the fragments assigned to site j must not exceed its capacity:

$$CA \sum_{\ell} p_{\ell}x_{\ell j} \leq CS_j, \quad \forall j \tag{16}$$

where $CA =$ cardinality of the relation, $p_{\ell} =$ number of bytes of attribute ℓ, $CS_j =$ capacity of site j.

Finally, it is worth noting that constraints (11)–(13) are the same for both models: VFA-RT and DFAR. We deliberately decided to formulate the constraints of these models as similar as possible in order to be able to experimentally compare both models.

5. Numerical experiments

The purpose of this experiment is to find out if it is worth using a model as complex as VFA-RT instead of a simpler one like DFAR for obtaining the optimal design of a distributed database (DDB). The answer to this question depends on how different is the optimal solution for a DDB when this is modeled using VFA-RT as compared to the optimal solution when the DFAR model is used.

Table 2
Input data for generating the test cases for the VFA-RT model.

Problem	No. of attributes $L\ (=t)$	No. of sites I	No. of queries K	Attribute usage $q_{k\ell}$	Emission frequency f_{ki}	Processing capacity C_j	Transmission speed C_{ij}	Mean length of queries $1/\mu_Q$	Mean length of responses $1/\mu_R$
P1	2	3	2	0–1	6–21	50	100 000	1000	5600
P2	10	2	3	0–1	0–19	50	100 000	1000	5600
P3	3	4	5	0–1	1–25	500	1 000 000	1000	5600
P4	10	3	4	0–1	2–21	500	1 000 000	1000	5600

Table 3
Input data for generating the test cases for the DFAR model.

Problem	No. of attributes $L\ (=t)$	No. of sites I	No. of queries K	c_{ij}	c_1	c_2	Attribute usage $q_{k\ell}$	Emission frequency f_{ki}	$l_{k\ell}$	d_ℓ	$a_{\ell i}$	CA	CS_j
P1	2	3	2	1–5	1	1	0–1	6–21	3–8	3–8	0	1	135–171
P2	10	2	3	1–5	1	1	0–1	0–19	7–20	7–20	0	1	206–214
P3	3	4	5	1–5	1	1	0–1	1–25	5–18	5–18	0	1	112–220
P4	10	3	4	1–5	1	1	0–1	2–21	2–18	2–18	0	1	108–206

5.1. Description of the test setting

For the experiments presented in this section, 120 test cases were generated randomly for the VFA-RT model and the same number of cases were generated for the DFAR model.

Table 2 shows the data used for generating the test cases for the VFA-RT model, i.e., for generating the coefficients of the variables for expressions (10)–(13). The 120 test cases are divided into 4 groups with 30 cases each, such that the cases in each group are similar to each other. For the sake of clarity, the term instance will be used for referring to a test case, and the term problem will be used for referring to a generic instance that represents a group of instances with similar characteristics. Therefore, the first row of the table contains the data used for generating the 30 instances of problem P1.

The values for P1 contained in the second, third and fourth columns indicate that the number of attributes, the number of sites and the number of queries are 2, 3 and 2 respectively for each of the 30 instances corresponding to P1. Additionally, the values in the fifth column indicate that each of the values of the usage matrix $[q_{k\ell}]$ was randomly chosen from 0 and 1, while the values of the sixth column indicate that each of the values of the frequency matrix $[f_{ki}]$ was chosen randomly in the range from 6 through 21. Finally, the rest of the columns indicate that C_j, C_{ij}, $1/\mu Q$, and $1/\mu R$ have the same values across all the instances of P1. A similar situation occurs for the generation of the instances for problems P2, P3 and P4.

It is convenient to mention that the values shown in the table are typical values that can be found in real cases. Additionally, it is important to make clear that larger instances were not considered in this experiment, because the execution time for obtaining the optimal solutions for instances grows exponentially with the instance size.

Table 3 shows the data used for generating the 120 test cases for the DFAR model. Like the situation that occurs for the VFA-RT model, in this case for each problem (from P1 through P4) 30 instances were randomly generated. Thus, the first row of the table contains the data used for generating the 30 instances of problem P1. The meaning of the data that contains each cell of Table 3 is similar to the one described previously for Table 2.

When observing the data for problem P1 in Table 3, it is noticeable that the values in columns 2, 3, 4, 8, and 9 are respectively the same as those in columns 2, 3, 4, 5, and 6 of Table 2 for problem P1, and a similar situation occurs for problems P2, P3 and P4. The rationale for this coincidence is that we intended to have similar instances for both models VFA-RT and DFAR, so as to be able to compare the optimal solutions for both models.

5.2. Comparative experiments for the VFA-RT and DFAR models

The comparative experiments were conducted using an exact algorithm for obtaining the optimal designs for 24 instances (six instances for each of the following problems: P1, P2, P3, and P4) using the VFA-RT and DFAR models. For each of the 24 instances, minimal solutions were obtained for both models aiming at finding out how different these solutions are.

Table 4 shows the results from this experiment. The first and second columns indicate the identifier of each instance. For the first row of the table, the third column shows the minimal response time $z_{\text{VFA-RT}}\ (x*_{\text{VFA-RT}})$, where $z_{\text{VFA-RT}}$ represents the value of the VFA-RT objective function (expression (1)) and $x*_{\text{VFA-RT}}$ represents the minimal solution for instance 1 of P1 when the VFA-RT model is used (Section 3), while the fourth column shows the response time $z_{\text{VFA-RT}}\ (x*_{\text{DFAR}})$, where $x*_{\text{DFAR}}$ represents the minimal solution for the same instance 1 of P1 when the DFAR model is used (Section 4). The fifth column contains the difference of the values from the third and fourth columns expressed as a percentage. For the rest of the table rows, the meaning of the values in each cell is similar.

Table 4
Comparative results for the VFA-RT and DFAR models.

Problem	Instance	Response time (s)		
		VFA-RT model	DFAR model	% difference
P1	1	0.228208	0.351527	54
P1	5	0.371648	0.483524	30
P1	8	0.127396	0.152670	20
P1	9	0.127047	0.253614	100
P1	11	0.146352	0.446129	204
P1	13	0.126362	0.164232	30
P2	1	0.118173	0.118173	0
P2	4	0.113708	0.113708	0
P2	5	0.184420	0.184420	0
P2	8	0.078366	0.078366	0
P2	11	0.078696	0.078696	0
P2	13	0.086817	0.086817	0
P3	1	0.007264	0.007919	9
P3	2	0.007382	0.007382	0
P3	3	0.007318	0.008354	14
P3	4	0.007049	0.007835	11
P3	5	0.007247	0.007839	8
P3	6	0.006238	0.006971	12
P4	1	0.006210	0.006746	9
P4	2	0.006784	0.007695	13
P4	3	0.006822	0.006877	1
P4	4	0.005701	0.007801	37
P4	5	0.005876	0.006975	19
P4	6	0.006197	0.006275	1

Table 5
Usage matrix $[u_{k\ell}]$ for the example.

Query	Attributes									
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
q_1	1	0	0	0	1	0	1	0	0	0
q_2	0	1	1	0	0	0	0	1	1	0
q_3	0	0	0	1	0	1	0	0	0	1
q_4	0	1	1	0	0	0	1	1	0	0
q_5	1	1	0	0	1	0	1	1	1	0
q_6	1	0	0	0	1	0	0	0	0	0
q_7	0	0	1	0	0	0	0	0	1	0
q_8	0	0	1	1	0	1	0	0	1	1

Table 6
Frequency matrix $[f_{ki}]$ for the example.

Query	Nodes			
	s_1	s_2	s_3	s_4
q_1	10	15	0	0
q_2	10	20	10	10
q_3	0	0	15	10
q_4	10	0	15	10
q_5	5	10	5	5
q_6	10	5	5	5
q_7	5	10	5	5
q_8	5	5	3	2

5.3. Comparative experiment for the VFA-RT model and a heuristic approach

A thorough review of the literature on vertical fragmentation reveals that most of the papers deal with either fragmentation or allocation of fragments; very few deal with both (as ours). One of the most recent papers that deal with both is [4].

It would not be fair to compare our approach to another that only deals with fragmentation or allocation, because these two aspects are so interdependent that the optimal design can only be obtained when both aspects are dealt with. Therefore, we decided to carry out an experiment for comparing the results of our approach and the one reported in [4]. The parameter values for the example presented in [4] are as follows: number of nodes = 4, number of attributes = 10, number of queries = 8; the usage matrix, the frequency matrix and the communication cost matrix are shown in Tables 5–7.

Table 7
Communication cost matrix $[c_{ij}]$ for the example.

Nodes	Nodes			
	s_1	s_2	s_3	s_4
s_1	0	10	25	20
s_2	10	0	20	15
s_3	25	20	0	15
s_4	20	15	15	0

Table 8
Transmission speed matrix $[C_{ij}]$ for the example.

Nodes	Nodes			
	s_1	s_2	s_3	s_4
s_1	∞	400 000	200 000	250 000
s_2	400 000	∞	250 000	300 000
s_3	200 000	250 000	∞	300 000
s_4	250 000	300 000	300 000	∞

Table 9
Processing speed vector $[C_j]$ for the example.

Nodes			
s_1	s_2	s_3	s_4
200	200	200	200

For the mathematical model described in Section 3, the parameter values are as follows: number of nodes = 4, number of attributes = 10, number of queries = 8, average query length $(1/\mu_Q) = 1000$, average response length $(1/\mu_R) = 5587$; the usage matrix, the frequency matrix, the transmission speed matrix, and the processing speed vector are shown in Tables 5, 6, 8 and 9.

The optimal fragmentation and allocation obtained by an exact algorithm for our model is as follows: fragment $F_1 = \{a_1, a_2, a_3, a_5, a_7, a_8, a_9\}$ allocated to node s_4 and fragment $F_2 = \{a_4, a_6, a_{10}\}$ allocated to node s_1, with a roundtrip response time of 0.016029. The fragmentation and allocation reported in [4] for this example is: fragment $F_1 = \{a_1, a_2, a_3, a_5, a_7, a_8, a_9\}$ allocated to node s_2 and fragment $F_2 = \{a_4, a_6, a_{10}\}$ allocated to node s_3. The roundtrip response time for this solution, calculated from expression (10), is 0.020302. Notice that both approaches yield the same fragmentation; however, the allocation of fragments is different, which makes the response time for the heuristic algorithm 27% larger than the one obtained with our approach. Unfortunately, no additional test cases are reported in [4], and therefore, no sound conclusions can be derived from this experiment.

5.4. Discussion

It usually happens that the more different the two approaches (models) for finding an optimal solution to an optimization problem are, the more different the generated solutions will be. The experiment presented in Section 5.3 exemplifies this situation. Therefore, in order to determine the effect that a new objective function (expression (10)) has on the solutions generated for a problem, we had to compare our approach (VFA-RT) with a similar model, preferably one that has the same set of constraints. Consequently, for the experiment described in Section 5.2, we deliberately chose to compare the VFA-RT and the DFAR models.

6. Conclusions

The last column of Table 2 shows that the difference of response time is zero for all the instances of problem P2, the difference is moderate for all the instances of problems P3 and P4, and it is very large for all the instances of problem P1. From these results it can be concluded that the use of the VFA-RT model is worthwhile, since there exist cases for which the optimal solutions obtained using the DFAR model have response times which can be thrice as large as those obtained using the VFA-RT model.

The great advantage of mathematical programming models (such as VFA-RT) is that they can be solved using metaheuristic algorithms, which explore much more solutions than heuristic algorithms, and therefore, they usually obtain better solutions.

The model presented in Section 3 can be improved for eliminating some simplifying assumptions. For example, the model assumes that each query type retrieves on average the same number of tuples (table rows) from a relation fragment, which is modeled by expression (9), where $1/\mu_R$ represents the average length of query responses. This assumption is not very

realistic, since in general queries q_k and q_m will retrieve on average different numbers of tuples. Therefore, for each query q_k a different average length should be considered, which will be denoted by $1/\mu_{R,k}$. In this case, for each pair of nodes i and j the average length of the responses transmitted from j to i is given by the following expression:

$$\frac{1}{\mu_R^*} = \frac{\sum_k f_{ki} y_{jk} (1/\mu_{R,k})}{\sum_k f_{ki} y_{jk}} \tag{17}$$

where f_{ki} is the emission frequency of query k from node i; $y_{jk} = 1$ if one or more attributes used by query k are stored in site j, and $y_{jk} = 0$ otherwise. Finally, in expression (9) μ_R should be substituted by μ_R^* from expression (17) for obtaining the following expression:

$$T_{TR}^* = \frac{1}{\sum_j \sum_k \sum_i f_{ki} y_{jk}} \sum_{ij} \frac{1}{\frac{C_{ji}}{\sum_k f_{ki} y_{jk} (1/\mu_{R,k})} - 1}. \tag{18}$$

A similar modification could be carried out for the average processing delay of queries given by expression (8). Notice that no modification is needed for the average transmission delay of queries (expression (7)), since the average length of the queries does not vary significantly from one type of query to another.

Additionally, the proposed model can be improved in the following way.

- Formulate a new mathematical model that minimizes roundtrip response time for DDBs with horizontal fragmentation. This would be specially helpful for DDB systems based on the cloud architecture.
- Formulate a new mathematical model that considers fragment replication, including "write queries", since the model proposed here only considers "read queries". This would permit modeling DDBs systems where the replication of fragments is allowed.

References

[1] R.A. Pazos, G. Vázquez, J. Pérez, J.A. Martínez, Modeling the nonlinear nature of response time in the vertical fragmentation design of distributed databases, in: J.M. Corchado, S. Rodríguez, J. Llinas, J.M. Molina (Eds.), Advances in Soft Computing, Vol. 50, Springer, Berlin, 2008, pp. 605–612.
[2] R.A. Pazos, G. Vázquez, J.A. Martínez, J. Pérez, Vertical fragmentation design of distributed databases considering the nonlinear nature of roundtrip response time, Lecture Notes in Artificial Intelligence 6277 (2012) 173–182.
[3] J. Pérez, R. Pazos, J. Frausto, et al., Vertical fragmentation and allocation in distributed databases with site capacity restrictions using the threshold accepting algorithm, Lecture Notes in Artificial Intelligence 1793 (2000) 75–81.
[4] H. Ma, K.-D. Schewe, M. Kirchberg, A heuristic approach to vertical fragmentation incorporating query information, in: Proc. 7th Intl. Baltic Conf. on Databases and Information Systems, Vilnius, Lithuania, 2006, pp. 69–76.
[5] M.T. Ozsu, P. Valduriez, Principles of Distributed Database Systems, Springer, USA, 2011.
[6] L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, Dover Publications, USA, 2007.
[7] S. Chakravarthy, J. Muthuraj, R. Varadarajan, et al., An objective function for vertically partitioning relations in distributed databases and its analysis, Distributed and Parallel Databases 2 (2) (1994) 183–207.
[8] L. Tambulea, M. Horvat-Petrescu, Redistributing fragments into a distributed database, International Journal of Computers Communications & Control 3 (4) (2008) 384–394.
[9] R. Karimi, S.M.T. Rouhani, A new ant colony optimization based algorithm for data allocation problem in distributed databases, Knowledge and Information Systems 20 (3) (2009) 349–373.
[10] S.U. Khan, I. Ahmad, Replicating data objects in large distributed database systems: an axiomatic game theoretic mechanism design approach, Distributed and Parallel Databases 28 (2–3) (2010) 187–218.
[11] E. Sevinc, A. Cosar, Distributed database design with genetic algorithm and relation clustering heuristic, Lecture Notes in Electrical Engineering 62 (2010) 133–136.
[12] S. Kamali, P. Ghodsnia, K. Daudjee, Dynamic data allocation with replication in distributed systems, in: Proc. 30th IEEE International Performance Computing and Communication Conf., Orlando, USA, 2011, pp. 1–8.
[13] M. Goli, S.M.T. Rouhani, A new vertical fragmentation algorithm based on ant collective behavior in distributed database systems, Knowledge and Information Systems 30 (2) (2012) 435–455.
[14] S. Song, Design of distributed database systems: an iterative genetic algorithm, Journal of Intelligent Information Systems (2013).