December 1997

VERTICAL DATA FRAGMENTATION AND FRAGMENT ALLOCATION IN DISTRIBUTED DATABASE SYSTEMS

Zehai Zhou
Hong Kong University os science and Technology

Olivia Sheng
University of Arizona

VERTICAL DATA FRAGMENTATION AND FRAGMENT ALLOCATION IN DISTRIBUTED DATABASE SYSTEMS

Zehai Zhou
Department of Information and Systems Management
Hong Kong University of science and Technology
Clear Water Bay, Kowloon, Hong Kong
Phone: (852) 2358-7640
Fax: (852) 2358-2421
Email: zzhou@uxmail.ust.hk

Olivia R. Liu Sheng
Department of Management Information Systems
University of Arizona
Tucson, Arizona 85721
USA

Department of Information and Systems Management
Hong Kong University of science and Technology
Clear Water Bay, Kowloon, Hong Kong
Phone: (852) 2358-7645
Fax: (852) 2358-2421
Email: olivia@usthk.ust.hk

Executive Summary
Distributed database technology is expected to have a significant impact on data processing in the upcoming years. The introduction of many commercial products and the continuing, intensive interest in distributed database systems in the research community and the marketplace indicate that distributed database systems will become more and more popular and eventually replace centralized systems as the major database technology in the future. The availability of high speed communication networks and, especially, the ever-increasing popularity of the Internet and the intranets may speed up the transition process. Distributed database systems have many promising potential advantages, such as improved reliability and availability, improved performance, shareability, expandability, increased robustness, and local autonomy, among others. To materialize the full potential benefits of distributed database technology, however, a series of technical problems must be satisfactorily resolved. Distributed database design is certainly one of the most important and fundamental issues to be addressed. Data fragmentation and allocation are two of the critical aspects of distributed database design. The data fragmentation and the fragment allocation problems in distributed database design are NP-hard in nature and notoriously difficult to solve, which makes developing good solution methods a high priority. Data allocation is typically treated independently of fragmentation. They are, however, highly interrelated processes. It is more reasonable to extend the two step methodology so that the interdependency of the fragmentation and the allocation decisions is properly reflected. Although the integrated methodology may be very complicated, there are synergistic effects of combining these two steps, enabling the development of better solutions with lower cost, higher availability, and/or higher throughput, etc. Since the fragmentation and allocation are design problems, it is often affordable or worthwhile to develop and use some sophisticated solution methods. In this study, we attempt to combine these two highly interrelated and interactive processes by formulating them as quadratic integer programming problems. More specifically, we solve the vertical fragmentation problem and the related fragment allocation problem together. The interdependency of the fragmentation and the allocation decisions is adequately reflected by the models and the resultant integer programs are significantly less complex than those from models using data cells as the allocation unit. Several solution methods are discussed and a new linearization method is investigated. Some preliminary yet extremely complicated computational experiments were conducted. We argue that the models developed for vertical fragmentation and fragment allocation have some potential advantages over the models that

are developed separately for data fragmentation and for data allocation. The computational experiments show that (1) the suggested linearization method performs clearly and consistently better than a widely used method, and (2) it is possible to solve medium (or even large) size instances of the vertical data fragmentation and fragment allocation problem using the commercial software and hardware available today.

1. Introduction

A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. Distributed database technology is expected to have a significant impact on data processing in the upcoming years (Ozsu and Valduriez 1994). The introduction of many commercial products and the continuing, intensive interest in distributed database systems in the research community and the marketplace indicate that distributed database systems will become more and more popular and eventually replace centralized systems as the major database technology in the future. The availability of high speed communication networks, and, especially, the ever-increasing popularity of the Internet and the intranets may further speed up the transition process. Distributed database systems have many promising potential advantages, such as improved reliability and availability, improved performance, shareability, expandability, increased robustness, and local autonomy, among others. To materialize the full potential benefits of distributed database technology, however, a series of technical problems must be satisfactorily resolved first. Distributed database design is certainly one of the most important and fundamental issues to be addressed. Data fragmentation and allocation are two of the critical aspects of distributed database design. The former involves the partitioning of each (global) relation into a group of fragment relations while the latter deals with the distribution of these fragmented relations across the sites of the distributed system.

There are several reasons for data fragmentation. First of all, a global relation is not a suitable unit in some cases. For instance, application views are usually subsets of relations rather than the whole relations. Secondly, locality of access of applications is defined not on entire relations but on their subsets. Therefore, it is very natural to consider subsets of relations as distributed units. Thirdly, if relations are not replicated at all, the result will be high volume of remote data access. On the other hand, if the entire relations are replicated at some or all sites, problems arise in executing updates and it may not be desirable (or even feasible) if the storage is limited and/or the data volume is huge. And finally, fragmentation may permit a number of transactions to be executed concurrently.

There are several fragmentation alternatives: horizontal, vertical, and hybrid fragmentation. Since data fragmentation is one of the most basic and important aspects of distributed database design, quite a few studies have been conducted (see Hoffer and Severance (1975), Navathe *et al.* (1984), Navathe and Ra (1989), Lin *et al.* (1993), Muthuraj *et al.* (1993), Chakravarthy *et al.* (1994) for example). It remains to be one of the most challenging problems to be addressed in distributed database system design.

The second step of distributed design is fragment allocation. There are many more studies on data allocation than on data fragmentation. The reader is referred to Apers (1988), Hevner and Rao (1988) or Dowdy and Foster (1982) for details about data allocation and the closely related file allocation problems.

The rest of this paper is organized as follows. Section 2 defines the terminology used. In section 3, we present a couple of quadratic integer programming formulations of integrated data fragmentation and allocation problems taking into consideration different constraints and under various assumptions. Section 4 discusses the solution methods. In section 5, we present some computational results. Concluding remarks are presented in section 6.

2. Terminology

The major data requirements related to data fragmentation and allocation include application access frequencies, query originating sites, attributes involved, etc. Usually, the algorithms developed do not require all the transaction data. Oftentimes only the most important queries are considered and included. The 20-80 rule indicates that a limited number of major transactions (20 percent) typically accounts for most of the use of the database (80 percent). Consequently, it is often sufficient to collect data from this 20 percent of transactions. From a

computational point of view, it is desirable or necessary to keep the required information about the database usage to a minimum.

Let $Q = \{q_1, q_2, q_3, ..., q_q\}$ be a set of queries (applications) running against relation $R(a_1, a_2, a_3, ..., a_a)$. An attribute usage value, denoted as QA_{ij} for each query q_i and each attribute a_j, is defined as follows:

$$QA_{ij} = \begin{cases} 1 & \text{if query } q_i \text{ references attribute } a_j \\ 0 & \text{otherwise} \end{cases}$$

The attribute usage values can also be defined in a matrix form, where entry (i,j) represents QA_{ij}. An example attribute usage matrix is shown below:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & a_3 & a_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Let $S=\{s_1, s_2, s_3, ..., s_s\}$ be the set of nodes (sites) that consist of the distributed system. Queries may originate from all these sites. Application access frequencies can be defined in a matrix form, where entry (i,l), denoted as QS_{il}, is the frequency of query q_i initialized from site s_l. An example is shown below:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{bmatrix} 10 & 5 & 20 \\ 5 & 0 & 20 \\ 20 & 30 & 0 \\ 20 & 10 & 10 \end{bmatrix} \end{array}$$

Neither the attribute usage matrix nor the access frequency matrix is sufficiently general to form the basis for data fragmentation or allocation because the former does not provide the weight of application frequencies while the latter fails to indicate the attributes involved in each query. The combination of these two matrix results in an attribute affinity matrix (AA), which provides the basic information needed in several vertical fragmentation algorithms. The element of attribute affinity matrix (AA), $AA(j_1,j_2)$ measures the bond between two attributes of a relation based on how they are accessed by the applications. More specifically, $AA(j_1,j_2)$ is the total amount of access of the transactions referring to both attributes j_1 and j_2, that is

$$AA(j_1, j_2) = \sum_{i=1}^{q} \sum_{l=1}^{s} QA_{ij_1} QA_{ij_2} QS_{il}$$

The following is an example derived from the examples of the attribute usage matrix and the application access matrix described above.

$$\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & a_3 & a_4 \end{array} \\ \begin{bmatrix} 120 & 35 & 50 & 35 \\ 35 & 135 & 25 & 75 \\ 50 & 25 & 75 & 0 \\ 35 & 75 & 0 & 110 \end{bmatrix} \end{array}$$

3. Models of vertical fragmentation and fragment allocation

Data allocation is typically treated independently of fragmentation. The process is linear since the input to the allocation is the output of the fragmentation. As Ozsu and Valduriez (1991b and 1994) pointed out, the isolation of the fragmentation and allocation steps actually contributes to the complexity of the allocation model. The two steps both have similar inputs although fragmentation works on global relations and allocation deals with fragment relations. Ozsu and Valduriez (1991b and 1994) suggested that it would be more promising to extend the two step methodology so that the interdependency of the fragmentation and the allocation decisions is properly reflected. Although the integrated methodology may be very complicated, there are synergistic effects of combining these two steps enabling the development of better solutions with lower cost, higher availability, and/or higher throughput, etc. The benefits may outweigh the costs. Since the fragmentation and allocation are design problems, it is often affordable or worthwhile to develop and use some sophisticated solution methods. In this study, we propose to combine these two highly interrelated and interactive processes by formulating them as quadratic integer programming problems. More

specifically, we solve the vertical fragmentation problem and the related fragment allocation problem together. The smallest fragment unit is thus an attribute. There are several reasons for this. First, the structure of the global relations is relatively stable. After all, a relation is defined after thorough and rigorous analysis. The schema will not be changed until there are significant changes in the organization or the database application requirements of the organization. If this does happen, it is highly likely that the databases need to be modified/reconstructed anyway. (Compared with tuples, it is conceivable that the tuples in a relation can be changed very often in many cases.) The interdependency of the fragmentation and the allocation decisions is adequately reflected by the models and the resultant integer programs are significantly less complex than those from models using data cells as the allocation unit.

The following notations are used:

V_j: size of attribute a_j.
CIO_l: unit I/O cost at site s_l.
CST_l: unit storage cost at site s_l.
D_{lm}: unit transmission cost from site s_m to site s_l.
K_l: storage capacity at site s_l.
LA_l: lower limit on number of attributes assigned to site s_l.
UA_l: upper limit on number of attributes assigned to site s_l.

3.1 Case I

Assign attributes to different sites. It is assumed that there is only one relation to be considered and no replications are allowed.

$$X_{jl} = \begin{cases} 1 & \text{if attribute } a_j \text{ is assigned to site } s_l \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{il} = \begin{cases} 1 & \text{if query } q_i \text{ references any attribute residing at site } s_l \\ 0 & \text{otherwise} \end{cases}$$

$$Z_l = \begin{cases} 1 & \text{if any attribute is assigned to site } s_l \\ 0 & \text{otherwise} \end{cases}$$

The formulation is:

$$\text{Min} \quad \sum_{i=1}^{q}\sum_{l=1}^{s} QS_{il} \sum_{\substack{m=1 \\ m\neq l}}^{s} D_{lm} \sum_{j=1}^{a} QA_{ij}X_{jm}V_j + \sum_{i=1}^{q}\sum_{l=1}^{s} QS_{il} \sum_{m=1}^{s} CIO_m Y_{im} \left(\sum_{j=1}^{a} X_{jm}V_j + H_m\right)$$

$$+ \sum_{l=1}^{s} CST_l \sum_{j=1}^{a} V_j X_{jl} + \sum_{l=1}^{s} CFO_l Z_l \qquad (1)$$

st

$$QA_{ij}X_{jl} \le Y_{il} \quad \forall i,j,l \qquad (2)$$

$$X_{jl} \le Z_l \quad \forall j,l \qquad (3)$$

$$\sum_{l=1}^{s} X_{jl} = 1 \quad \forall j \qquad (4)$$

$$\sum_{j=1}^{a} V_j X_{jl} \le K_l \quad \forall l \qquad (5)$$

$$LA_l \le \sum_{j=1}^{a} X_{jl} \le UA_l \quad \forall l \qquad (6)$$

$$X_{jl} \in \{0, 1\}, \; Y_{il} \in \{0, 1\}, \; Z_l \in \{0, 1\} \qquad (7)$$

where H_m is the overhead cost of accessing a file at site s_m and CFO_l is the overhead cost associated with keeping the fragment at site s_l. (1) is the objective function, which consists of the transmission cost, retrieval cost, storage cost and the overhead cost of keeping a file at different sites. Constraint (2) ensures that all the queries can be carried out. (3) ensures that

a site l must be open if attribute j is to be assigned to that site. (4) ensures that each attribute will be assigned to exactly one site. (5) ensures that the total size of the attributes assigned to site l will not exceed the capacity at site l while constraint (6) guarantees that the number of attributes assigned to a given site l will be in the range of LA and UA. Constraint (7) is a zero-one constraint on the variables.

In this formulation, attributes are assigned to different sites based on the transmission costs, I/O costs, storage costs and overhead cost of keeping files at different sites subject to several kinds of constraints. It is implicitly assumed that all the attributes residing at any given site make up one fragment and are stored in one file. A query that involves any subset of attributes at a given site will require fetching the whole fragment from second memory to primary memory. Here we do not consider the situations where attributes at any site can be grouped further into fragments. This may or may not be reasonable depending on particular applications. This is acceptable if the number of sites is large and/or the number of attributes assigned to each site is small. If more than one fragment at some sites is more desirable, we develop a model in case II, which allows fragments to be organized at different sites. An alternative, which we describe below, is to use a two-phase method. In the first stage, we assign the attributes to different sites based on the model presented above. In the second stage, we divide the attributes assigned to any site into fragments as we do in a centralized system. Let $F = \{f_1, f_2, f_3, ..., f_f\}$ be a set of fragments in a given site and

$$X_{jp} = \begin{cases} 1 & \text{if attribute } a_j \text{ is assigned to fragment } f_p \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ip} = \begin{cases} 1 & \text{if query } q_i \text{ references any attribute in fragment } f_p \\ 0 & \text{otherwise} \end{cases}$$

$$Z_p = \begin{cases} 1 & \text{if any attribute is assigned to fragment } f_p \\ 0 & \text{otherwise} \end{cases}$$

The formulation is as follows:

$$\text{Min} \quad \sum_{i=1}^{q} QS_i \sum_{p=1}^{f} CIO\, Y_{ip} \left(\sum_{j=1}^{a} X_{jp} V_j + H\right) + \sum_{p=1}^{f} CFO_p Z_p \tag{8}$$

st

$$QA_{ij} X_{jp} \le Y_{ip} \quad \forall i, j, p \tag{9}$$

$$X_{jp} \le Z_p \quad \forall j, p \tag{10}$$

$$\sum_{p=1}^{f} X_{jp} = 1 \quad \forall j \tag{11}$$

$$\sum_{j=1}^{a} V_j X_{jp} \le K_p \quad \forall p \tag{12}$$

$$LA_p \le \sum_{j=1}^{a} X_{jp} \le UA_p \quad \forall p \tag{13}$$

$$X_{jp} \in \{0, 1\}, \; Y_{ip} \in \{0, 1\}, \; Z_p \in \{0, 1\} \tag{14}$$

where H is the overhead cost of accessing a file at a given site and CFO_p is the overhead cost associated with keeping a file (fragment) f_p at a given site. In most cases, it is reasonable to assume that CFO_p is the same for all p. (8) through (14) have similar meaning as (1)-(7).

3.2 Case II

Assign attributes to different sites under the condition that attributes in any site can be further grouped into different fragments. It is assumed that there is only one relation to be considered while no replications are allowed.

$$X_{jp,l} = \begin{cases} 1 & \text{if attribute } a_j \text{ is assigned to fragment } f_p \text{ at site } s_l \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ip_jl} = \begin{cases} 1 & \text{if query } q_i \text{ references any attribute in fragment } f_p \text{ residing at site } s_l \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{p_jl} = \begin{cases} 1 & \text{if any attribute is assigned to fragment } f_p \text{ at site } s_l \\ 0 & \text{otherwise} \end{cases}$$

The formulation becomes:

$$\text{Min} \quad \sum_{i=1}^{q}\sum_{l=1}^{s} QS_{il} \sum_{\substack{m=1 \\ m\neq l}}^{s}\sum_{p_m=1}^{f} D_{lm} \sum_{j=1}^{a} QA_{ij} X_{jp_mm} V_j$$

$$+ \sum_{i=1}^{q}\sum_{l=1}^{s} QS_{il} \sum_{m=1}^{s}\sum_{p_m=1}^{f} CIO_m Y_{ip_mm} (\sum_{j=1}^{a} X_{jp_mm} V_j + H_m)$$

$$+ \sum_{l=1}^{s} CST_l \sum_{p_l=1}^{f}\sum_{j=1}^{a} V_j X_{jp_jl} + \sum_{l=1}^{s}\sum_{p_l=1}^{f} CFO_l Z_{p_jl} \qquad (15)$$

st

$$QA_{ij} X_{jp_jl} \leq Y_{ip_jl} \quad \forall i, j, p, l \qquad (16)$$

$$X_{jp_jl} \leq Z_{p_jl} \quad \forall j, p, l \qquad (17)$$

$$\sum_{l=1}^{s}\sum_{p_l=1}^{f} X_{jp_jl} = 1 \quad \forall j \qquad (18)$$

$$\sum_{j=1}^{a}\sum_{p_l=1}^{f} V_j X_{jp_jl} \leq K_l \quad \forall l \qquad (19)$$

$$LA_l \leq \sum_{j=1}^{a}\sum_{p_l=1}^{f} X_{jp_jl} \leq UA_l \quad \forall l \qquad (20)$$

$$X_{jp_jl} \in \{0, 1\}, \; Y_{ip_jl} \in \{0, 1\}, \; Z_{p_jl} \in \{0, 1\} \qquad (21)$$

(15) through (21) have similar meanings as (1)-(7). For more models developed, please refer to Zhou (1996) for details.

4. Solution methods

The quadratic integer program formulations (denoted as VFA) discussed above have some similarities with the quadratic assignment problem (QAP). For instance, the objective functions of both the FFA and QAP are quadratic, all the quadratic terms in the objective functions are the product of two (different) zero-one variables, and the objective functions of both problems are neither separable nor convex. It is well known that the QAP is NP-hard and extremely difficult to solve optimally except for small problems ($n \leq 15$, where n is the number of sites (facilities)). Therefore, solving VFA would be very difficult (if not more difficult). Main approaches to solving constrained nonlinear 0-1 programs like VFA include linearization, algebraic methods, cutting plane methods, and enumerative methods. The reader is refereed to Hansen et al (1993) for a state-of-the-art survey on constrained nonlinear 0-1 programming. In this study, we conduct experiments with algorithms pertaining to the linearization approach.

Many researchers devised and/or advocated different linearization methods. Glover and Woolsey (1974) proposed two basic approaches described below:

A distinct 0-1 product form

$z = x_1 x_2 ... x_n$ where x_i (i=1, 2, ..., n) are 0-1 variables,

can be linearized by

(i) GW1

$$x_1 + x_2 + ... + x_n \geq nz \qquad (22)$$

$$x_1 + x_2 + ... + x_n - n + 1 \leq z \qquad (23)$$

where z is a 0-1 variable.
(ii) GW2

$$z \geq x_1 + x_2 + \ldots + x_n \quad (24)$$
$$z \leq x_i, \ i=1, 2, \ldots, n \quad (25)$$
$$z \geq 0 \quad (26)$$

where z is a continuous variable.

GW1 has its advantage and disadvantage compared with GW2. GW1 uses only two constraints while GW2 requires n + 2 constraints. However, z must be a 0-1 variable for GW1 while in GW2 a continuous variable. In general, GW2 is more effective than GW1 if n is not large (Lin, 1994a). GW1 and GW2 introduce a single new variable to take the place of each unique cross-product term. Auxiliary constraints are required to ensure that the new variables assume appropriate values. More specifically, GW1 requires $\frac{1}{2}n(n-1)$ additional 0-1 variables and n(n-1) new constraints while GW2 needs additional continuous variables and 2n(n-1) new constraints. Glover and Woolsey (1973, 1974) and Glover (1975) presented different linearization methods resulting in more compact formulations. Glover's approach requires only n new continuous variable and 4n new linear constraints. The most economical method in the literature is presented by Oral and Kettani (1992a, 1992b) and Kettani and Oral (1990, 1993). In Oral and Kettani (1992), the equivalent formulation technique introduces only n new constraints and n new nonnegative continuous variables. Although Oral and Kettani (1992a, 1992b) and Kettani and Oral (1990, 1993) reported that their compact methods compared favorably in terms of CPU time with Glover's (and some other) methods, linearization methods do not appear to have been systematically tested. According to Hansen *et al.* (1993), compact linearizations do not appear to be clearly better than expanded ones. Compact formulations may have some disadvantages: they are not straightforward and often require some sort of preprocessing to get the formulation.

For other linearizations and equivalent (mixed) integer linear formulations of nonlinear 0-1 programs, see Adams and Sherali (1986, 1990), Beale and Tomlin (1972), Goldman (1983), Lin (1994b), Sherali and Adams (1990). According to Hansen *et al.* (1993), there were more that thirty papers that reported the use of GW1 linearization or advocated its use. In database design area, Mohania and Sarda (1994) used GW1 in rule allocation in distributed deductive database systems. In all the models we developed so far for data fragmentation and allocation, only quadratic terms appear. Thus, from now on we will limit our discussion to quadratic 0-1 programs although some of the analysis can be extended to other cases. We expect that GW2 would have better performance as Lin (1994a) suggested. Observing that we are solving minimization problems and the coefficients of all the cross-product terms are positive, we propose an improved linearization, which has the potential to reduce the solution time dramatically.

Proposition 1: To linearize the cross-product terms in our models, the following constraints are sufficient (Zhou 1996)

$$x_i + x_j - z \leq 1 \quad (27)$$
$$z \geq 0 \quad (28)$$

Nonnegative constraints (like $z \geq 0$) are implicitly assumed in almost all linear programming software. One usually needs to transform a linear program with unrestricted variables to a linear program with only nonnegative variables by introducing two nonnegative variables for each unrestricted (in sign) variable. Therefore (28) in the linearization is not significant. Our linearization requires only one explicit constraint which is also included in GW1 or GW2. We would anticipate a significant improvement in performance.

5. Computational Experiments

In order to find how efficiently we can solve the vertical fragmentation and data allocation problems using exact algorithms and to observe the advantage of our linearization method over GW2, some computational experiments were conducted. A set of 12 instances from model I are tested. All the 12 instances were solved by the CPLEX system 3.0 on a DEC Alpha AXP 2100 system (190 MHz, 128M main memory) running OSF/1 UNIX. CPLEX employs the branch-and-bound method to solve (mixed) integer programming problems. The computational results are summarized in Tables 1 and 2. For each of the two linearization

methods, we test instances with a different number of (major) query patterns, a different number of attributes and a different number of sites. All the instances (except P12) include the storage constraints and the two sets of attribute-number-in-site constraints. Also given in Tables 1 and 2 are the objective values of the linear programming relaxation and the optimal solution, the number of iterations, the number of nodes in the branch-and-bound trees, and the solution time (in seconds).

Table 1 Computation time with Glover and Woolsey's linearization

Problem	no. of queries	no. of attributes	no. of sites	optimal value	no. of iterations	no. of nodes	LP value	solution time (sec.)
P1	10	8	3	33892.578	4221	61	28114.786	22.33
P2	10	10	3	31569.414	8549	94	24901.622	50.97
P3	15	10	3	44920.537	12449	99	35230.942	121.87
P4	15	15	3	65972.671	41139	280	50378.727	813.48
P5	20	15	3	92929.643	91273	535	70258.053	2323.38
P6	20	20	3	112817.60	149977	691	85163.502	5092.78
P7	20	10	5	143187.84	40926	140	126005.41	1771.37
P8	20	15	5	231623.77	260747	739	203892.40	15164.18
P9	25	10	5	178618.91	66348	296	160481.78	3464.53
P10	25	15	5	303880.53	287390	1215	288595.19	20874.98
P11	30	15	7	452123.78	391481	733	423445.65	68987.53
P12	30	20	7	551496.11	63488	190	488434.79	14950.37

Table 2 Computation time with the suggested linearization

Problem	no. of queries	no. of attributes	no. of sites	optimal value	no. of iterations	no. of nodes	LP value	solution time (sec.)
P1	10	8	3	33892.578	1807	65	28114.786	1.60
P2	10	10	3	31569.414	2741	82	24901.622	4.25
P3	15	10	3	44920.537	5496	99	35230.942	11.03
P4	15	15	3	65972.671	13823	306	50378.727	50.73
P5	20	15	3	92929.643	29366	560	70258.053	140.00
P6	20	20	3	112817.60	43455	741	85163.502	294.57
P7	20	10	5	143187.84	7348	132	126005.41	33.88
P8	20	15	5	231623.77	65743	838	203892.40	415.45
P9	25	10	5	178618.91	12637	274	160481.78	76.58
P10	25	15	5	303880.53	64287	1351	288595.19	595.58
P11	30	15	7	452123.78	55824	754	423445.65	851.68
P12	30	20	7	551496.11	16861	221	488434.79	259.00

It is worthwhile to mention that the resultant linear 0-1 programs from linearization are extremely complex. For example, the corresponding mixed linear 0-1 program derived from problems (P7-P12) contain tens of thousands of 0-1 integer variables and many times more constraints. From a mathematical, especially integer programming point-of-view, these programs are huge by any standard. Our linearization procedure generates programs much smaller than those generated from GW2. The computational experiments show that:

(i) The suggested linearization is clearly and consistently better than GW2 linearization. Although for the suggested linearization the number of nodes explored is usually slightly more than that for the GW2 linearization, the number of iterations is much smaller. And what is more important, the solution time is dramatically reduced. It took 11-80 times longer to solve the same instance with GW2 than with our suggested linearization method. The differences might be even more significant had we solved instance of larger sizes.

(ii) With the suggested linearization method and the use of the state-of-the art software package and hardware, we may be able to solve reasonable or even large size instances of vertical fragmentation and data allocation problems.

(iii) By exploring the structure of models, we are able to solve huge mixed 0-1 programs that we otherwise just could not solve or even think of solving to optimality with exact algorithms (non-heuristics). The procedure provides surprisingly good results in terms of solution time and the size of problems we can solve. It looks promising for solving other problems with similar structures.

6. Discussion and conclusions

It is evident that distributed database technology will have a significant impact on data processing in the upcoming years. To realize the full benefit of distributed database technology, we need to resolve two of the most important issues of distributed database design: data fragmentation and allocation. In this paper, we attempt to combine these two highly interactive decision processes by formulating them as quadratic integer programs by taking into consideration different constraints and various assumptions. Different solution methods are discussed and a new linearization method is presented and investigated. Some preliminary computational experiments are conducted and the results are also reported.

It appears that the models developed for vertical data fragmentation and fragment allocation have some potential advantages over the models that are developed separately for data fragmentation and for data allocation. The computation experiments show that the suggested linearization method performs clearly and consistently better than a currently widely used method. The computational results also indicate that it is possible to solve reasonable (or even large) size instances of the vertical data fragmentation and fragmentation allocation problem using the commercial software (such as CPLEX) and hardware available today.

It is anticipated that the models for vertical fragmentation and fragment allocation in distributed database systems and solution techniques may be of practical as well as theoretical use. Nevertheless, much more needs to be done to solve the distributed database design problems in order to reach the potential benefits of distributed database systems. For future study, we shall try to develop some heuristics by studying the models in order to solve (very) large size instances. Some modern techniques such as tabu search, genetic algorithms, neural networks, etc. are also potential alternatives to be investigated. We shall also extend the work in this paper to other systems, such as distributed multimedia database system, and to take into account other factors such as network load, different query processing methods (e.g. semi-join), etc.

References

Adams, W. P. and Sherali, H. D. "A tight linearization and an algorithm for zero-one quadratic programming problems." Management Science, Volume 32, 1986, pp.1274-1290.

Adams, W. P. and Sherali, H. D. "Linearization strategies for a class of zero-one mixed integer programming problems." Operations Research, Volume 38, 1990, pp.217-226.

Apers, P. M. G. "Data allocation in distributed database systems." ACM Transactions on Database Systems, Volume 13, 1988, pp.263-304.

Beale, E. M. L. and Tomlin, J. A. "An integer programming approach to a class of combinatorial problems." Mathematical Programming, Volume 3, 1972, pp.339-344.

Ceri, S. and Pelagatti, G. Distributed Databases: Principles and Systems, New York: McGraw-Hill, 1984.

Chakravarthy, S.; Muthuraj, J.; Varadarajan, R.; and Navathe, S. B. "An objective function for vertically partitioning relations in distributed databases and its analysis." Distributed and Parallel Databases, Volume 2, 1994, pp.183-207.

Dowdy, L. and Foster, D. V. "Comparative models of the file assignment problem." Computing Survey, Volume 14, 1982, pp.287-313.

Glover, F., "Improved linear integer programming formulations of nonlinear integer problems." Management Science, Volume 22, 1975, pp.455-460.

Glover, F. and Woolsey, E. "Further reduction of zero-one polynomial programming problems to zero-one linear programming problems." Operations Research, Volume 21, 1973, pp.156-161.

Glover, F. and Woolsey, E. "Converting 0-1 polynomial programming problem to a 0-1 linear program." Operations Research, Volume 22, 1974, pp.180-182.

Goldman, A. J. "Linearization in 0-1 variables: a clarification." Operations Research, Volume 31, 1983, pp.946-947.

Hansen, P.; Jaumard, B.; and Mathon, V. "Constrained nonlinear 0-1 programming." ORSA Journal on Computing, Volume 5, 1993, pp.97-119.

Hevner, A. R. and Rao, A. "Distributed data allocation strategies." In M. C. Yovits (ed.), Advances in Computers, New York: Academic Press, 1988, p.121-155.

Hoffer, J. A. and Severance, D. G. "The use of cluster analysis in physical database design," In Proceedings of the First International Conference on Vary Large Databases (Framingham, Mass.), 1975, pp.69-86.

Kettani, O. and Oral, M. "Equivalent formulations of nonlinear integer problems for efficient optimization." Management Science, Volume 36, 1990, pp.115-119.

Kettani, O. and Oral, M. "Reformulating quadratic assignment problems for efficient optimization." IIE Transactions, Volume 25, 1993, pp.97-107.

Lin, H. L. "A new global approach for 0-1 polynomial problems." Computers and Operations Research, Volume 21, 1994a, pp.319-327.

Lin, H. L. "Global optimization for mixed 0-1 programs with convex or separable continuous functions." Journal of the Operational Research Society, Volume 45, 1994b, pp.1068-1076.

Lin, X.; Orlowska, M.; and Zhang, Y. "A graph based cluster approach for vertical partitioning in database design." Data and Knowledge Engineering, Volume 11, 1993, pp.151-169.

Mohania, M.K. and Sarda, N.L. "Rule allocation in distributed deductive database systems." Data and Knowledge Engineering, Volume 14, 1994, pp.117-141.

Muthuraj, J.; Chakravarthy, S.; Varadarajan, R.; and Navathe, S. B. "A formal approach to the vertical partitioning problem in distributed database design." In Proceedings of the Second International Conference on Parallel and Distributed Information Systems, 1993, pp.26-34.

Navathe, S.; Ceri, S.; Wiederhold, G.; and Dou, J. "Vertical partitioning algorithms for database design." ACM Transactions on Database Systems, Volume 9, 1984, pp.680-710.

Navathe, S. and Ra, M. "Vertical partitioning for database design: a graphical algorithm." SIGMOD Reports, Volume 18, 1989, pp.440-450.

Oral, M. and Kettani, O. "A linearization procedure for quadratic and cubic mixed-integer problems." Operations Research, Volume 40 (Supp. 1), 1992a, pp.s109-116.

Oral, M. and Kettani, O. "Reformulating nonlinear combinatorial optimization problems for higher computation efficiency." European Journal of Operational Research, Volume 58, 1992b, pp.236-249.

Ozsu, M. T. and Valduriez, P. Principles of Distributed Database Systems, Englewood Cliffs, New Jersey: Prentice-Hall, 1991a.

Ozsu, M. T. and Valduriez, P. "Distributed database systems: Where are we now?" IEEE Computer, Volume 24, 1991b, pp.68-78.

Ozsu, M. T. and Valduriez, P. "Distributed data management: Unsolved problems and new issues," in: T. Casavant and M. Singhal (eds.), Readings in Distributed Computing System, IEEE/CS Press, 1994, pp.512-544.

Sherali, H. D. and Adams, W. D. "A hierarchy of relaxation's between the continuous and convex hull representations for zero-one programming problems." SIAM Journal on Discrete Mathematics, Volume 3, 1990, pp.411-430.

Zhou, Zehai. Data Allocation and Query Optimization in Large Scale Distributed Databases, unpublished Ph.D. dissertation, University of Arizona, Tucson, 1996.