Hindawi Publishing Corporation
Journal of Computer Systems, Networks, and Communications
Volume 2010, Article ID 984059, 6 pages
doi:10.1155/2010/984059

Research Article

Subquery Allocation Problem and Heuristics for Secret Sharing Distributed Database System

Toshiyuki Miyamoto and Takeshi Ikemura

Division of Electrical, Electronic and Information Engineering, Graduate School of Engineering, Osaka University, Suita 565-0871, Japan

Correspondence should be addressed to Toshiyuki Miyamoto, miyamoto@eei.eng.osaka-u.ac.jp

Received 17 August 2009; Revised 15 December 2009; Accepted 4 January 2010

Academic Editor: Abderrahim Benslimane

We discuss query optimization in a secure distributed database system, called the Secret Sharing Distributed DataBase System (SSDDBS). We have to consider not only subquery allocations to distributed servers and data transfer on the network but also decoding distributed shared data. At first, we formulated the subquery allocation problem as a constraints satisfaction problem. Since the subquery allocation problem is NP-complete in general, it is not easy to obtain the optimal solution in practical time. Secondly, we proposed a *heuristic evaluation* function for the best-first search. We constructed an optimization model on an available optimization software, and evaluated the proposed method. The results showed that feasible solutions could be obtained by using the proposed method in practical time, and that quality of the obtained solutions was good.

1. Introduction

The need for distributed databases (DDB) [1] is increasing as information technologies develop and the amount of data to be handled is growing exponentially. For databases such as those containing personal data, confidentiality, dependability, and robustness are becoming increasingly important. We have proposed a Secret Sharing Distributed Database (SSDDB) [2] that combines a secure distributed storage system [3–5] with a relational database system. The relations are divided into fragments in the SSDDB, and the fragments are managed by encrypting with a (k, n) threshold scheme [6]. Thus, the fragment must be both encrypted and decrypted every time there is a request to the database system to perform a search or some other functions. This makes it essential to have optimized parallel operation.

Optimization of the query process in an SSDDB proceeds in 3 stages: optimization of the query structure, optimization of subquery allocation to servers employed, one for each fragment request, and optimization of the combination of subquery results. This paper addresses the issue of subquery allocation to servers.

Optimization of the query process in an ordinary DDB is precisely described in [1]. A database management system must decide which server to use for processing when the same data are in distributed locations due to replication, even for ordinary distributed databases. Optimization of the query process for such a case is not sufficiently described in [1]. Evrendilek et al. proposed a query optimization problem which considers the data replication in [7]. However, with SSDDB, a server must be selected for decrypting fragments, and the procedures suggested for allocating subqueries to servers in ordinary DDBs are not very effective here. There exists no research on optimization of the query process for SSDDB as far as we know.

There are three factors to consider when allocating subqueries to servers: firstly, subquery allocation to servers; next, determination of the server storing the fragment needed to handle the subquery; finally, determination of the server for the necessary decrypting of the fragment. Subquery allocation to servers in an ordinal distributed database is known to be an NP-complete problem [7]. Since the subquery allocation problem in an ordinal distributed database is a special case of that of SSDDB, subquery

allocation to servers becomes an NP-complete problem. This paper is an investigation of best-first search methods with a view to online use, and proposes heuristic methods for the best-first search.

Section 2 provides an overview of SSDDB and Section 3 demonstrates the formulation of the subquery allocation problem as a constraint satisfaction problem. A *heuristic evaluation* function for a best-first search is then proposed. Section 4 shows the execution of an experimental calculation and demonstrates the effectiveness of the proposed *heuristic evaluation* function.

2. Secret Sharing Distributed Database System

2.1. The (k, n) Threshold Scheme. The secret sharing distributed database system is software that safely performs distributed sharing of confidential information. The system described herein uses the (k, n) threshold scheme demonstrated by Shamir [6].

Let F represent an algebraic number field and a confidential datum s be an element of F. A $(k-1)$th degree polynomial equation is created using s and random coefficients $r_i \in F$ $(i = 1, \ldots, k-1)$

$$f(x) = \sum_{i=0}^{k-1} r_i x^i, \tag{1}$$

where $r_0 = s$. The value $w_j = f(x_j)$ is calculated for each of the n distinct values of $x_j \in F$ $(j = 1, \ldots, n,\ x_j \neq 0)$. The value w_j is called a share and x_j is called the ID of share w_j.

Since $f(x)$ is a $(k-1)$th degree polynomial, it is uniquely defined by its values at k different points $(x_j, f(x-j))$. The confidential datum s can be retrieved by evaluating $s = f(0)$.

In the (k, n) threshold scheme, it is impossible to decrypt s using fewer than k shares. There is no dependence on how shares are selected, so even if up to $n-k$ shares are lost, it remains possible to decrypt s.

2.2. SSDDB Structure. An outline of the structure of an SSDDB is presented in Figure 1. The SSDDB is composed of a database management system (DBMS) and a secret sharing storage system (SSSS). The DBM process on the DBMS and the SSS process on the SSSS are assumed to be connected to each other over the Internet and each is able to communicate with the other individually.

When the DBM receives a query from an external source, it creates a query processing plan. It carries out this processing while assigning subqueries to other DBMs for parts of the processing and sending requests to DBMs and SSSs to retrieve or decrypt the necessary data sets. Thus, the user can approach the DBMS in the same way as approaching an ordinary distributed database management system, but an SSSS is used for storage, so the specificity of an SSSS must be considered in the query optimization, as described below.

The data sets are divided into small sets (fragments) and encrypted using the (k, n) threshold scheme. This process is managed by the SSSS. A DBM is also capable of caching data sets. Only DBM3 in the figure is caching data.

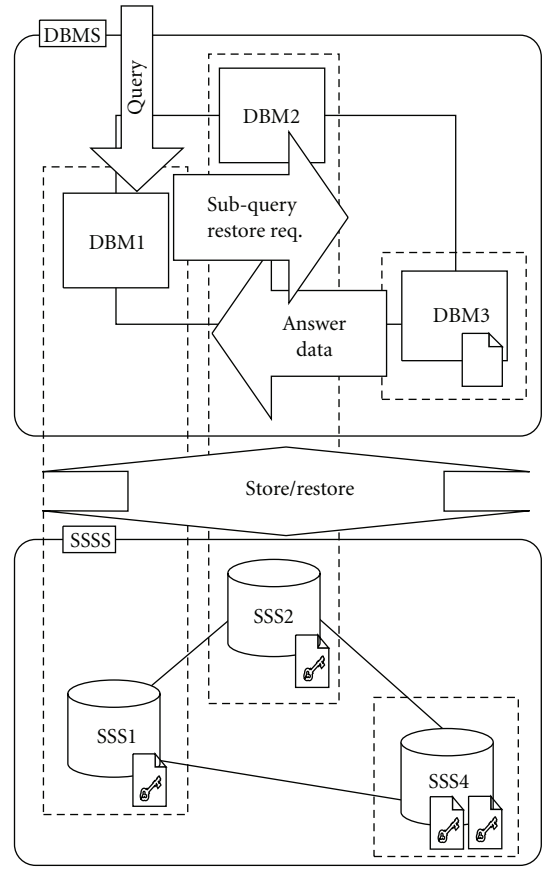

Figure 1: Secret Sharing Distributed Database.

The dashed lines in the figure indicate physical boundaries. DBM1, SSS1, DBM2, and SSS2 are each separate processes running on the same computer. In other words, no SSS process is running on the computer where DBM3 is running. A DBM can request an SSS to retrieve and decrypt a fragment on a different computer. However, this generates a cost due to transmission of data over the network.

3. Query Optimization for an SSDDB

3.1. Outline of Query Processing in an SSDDB. Optimization of query processing in an SSDDB consists of the three stages of firstly, query structure optimization, secondly, optimization of subquery allocation to multiple servers, one for each fragment request, and thirdly, optimization of the combination of the subquery results.

In query structure optimization, the query is subdivided by employing information in the fragments. The first task is to eliminate any redundant expressions sometimes included in user-generated queries. Queries are also rewritten or divided into subqueries in order to reduce the volume of intermediate data. At this point, SSDDB specificity is not yet required, so the ordinary procedure for a DDBS [1] can be used as is.

The generated subqueries can be processed by separate servers. In the second stage, a server for processing a

subquery is selected so as to minimize the processing cost of the query, by considering fragment allocations and server loads. In the SSDDB, all fragments except for those in the cache are distributed as shares in the SSSS. Thus, no particular fragment is assigned to any particular server. The cost of recreating distributed fragments must be calculated. Sections 3.2 and 3.3 address the issues of allocating subqueries to servers while considering the specificity of the SSDDB.

Finally, the results of processing the subqueries assigned to the different servers are combined. As with the first step, SSDDB specificity is not required, so again the ordinary procedure for a DDBS [1, 7] can be applied as is.

3.2. Subquery Allocation Problem to Servers. Let S be a finite set of servers with $|S| = p$. Here, the term server is assumed to mean a physical computer, at most one DBM process and at most one SSS process are running on each server. F is a finite set of fragments with $|F| = m$, and SQ is a finite set of subqueries with $|SQ| = r$.

Let $d_{j,h} \in \mathbb{R}^+$ be the cost required to decrypt some fragment $f_j \in F$ on some server $s_h \in S$. Here, $\mathbb{R}^+$ is the set of all nonnegative real numbers. Costs are set considerably higher in servers where an SSS process is not running, in order to avoid assigning decryption requests to such servers. When a fragment f_j is decrypted on a server s_h, the cost for transmitting shares from other servers is assumed to be $st_{j,h} \in \mathbb{R}^+$.

The cost for processing a subquery sq_i on a server s_h is assumed to be $w_{i,h} \in \mathbb{R}^+$. As was the case with decrypting costs, this should be set rather high for servers where a DBM process is not running. The cost necessary in order to transmit a fragment f_j from a server s_h to a $servers_{h'}$ is written $tf_{j,h,h'}$. The load cost of a server s_h is assumed to be $l_h \in \mathbb{R}^+$.

Some additional variables are employed here. $v_{i,j}$ is set to the value of 1 when a subquery sq_i calls a fragment f_j; otherwise, to the value of 0. $c_{j,h}$ is set to 1 when a fragment f_j is cached in a server s_h; otherwise, to 0. $x_{i,h}$ is set to 1 when a subquery sq_i is allocated to a server s_h; otherwise, to 0. y_{h_1,j,h_2} is set to 1 when a fragment f_j is transmitted to a server s_{h_1}, where it is required, from a server s_{h_2}; otherwise, it is 0. $z_{j,h}$ is set to 1 when a fragment f_j is decrypted at a server s_h; otherwise, it is 0.

Normally, parameters of disk I/O, CPUs and the network can be used when constructing a query processing cost model in a distributed database. For an SSDDB, disk I/O can be considered a part of the share decryption cost; the following two costs are employed.

The calculation cost SC_h in a server s_h is defined with the following expression:

$$SC_h = l_h + \sum_{sq_i \in SQ} x_{i,h} w_{i,h} + \sum_{f_j \in F} z_{j,h} d_{j,h}. \tag{2}$$

The network cost NC_h for a server s_h is defined thus:

$$NC_h = \sum_{f_j \in F} z_{j,h} st_{j,h} + \sum_{f_j \in F} \sum_{s_{h_2} \in S} y_{h,j,h_2} tf_{j,h_2,h}. \tag{3}$$

The above expressions indicate that the subquery allocation problem to servers becomes a constraint satisfaction problem for the decision variables $x_{i,h}$, y_{h_1,j,h_2}, and $z_{j,h}$

$$\min \max_{s_h \in S} \{SC_h + NC_h\} \tag{4}$$

subject to

$$\forall sq_i \in SQ, \quad \sum_{s_h \in S} x_{i,h} = 1, \tag{5}$$

$$\forall sq_i \in SQ, \quad \forall f_j \in F, \quad \forall s_{h_1} \in S, \quad x_{i,h_1} v_{i,j} = \sum_{s_{h_2} \in S} y_{h_1,j,h_2}, \tag{6}$$

$$\forall s_{h_1} \in S, \quad \forall f_j \in F, \quad \sum_{s_{h_2} \in S} y_{h_2,j,h_1} \left(z_{j,h_1} + c_{j,h_1} \right) = \sum_{s_{h_2} \in S} y_{h_2,j,h_1}. \tag{7}$$

Here, the object function prevents the subqueries from being concentrated on a single server. Equation (5) is to ensure that each subquery is processed only once by one of the servers. Equation (6) indicates that the fragment a server requires must always be transmitted from some server (please note that the $h_1 = h_2$ case is permitted). Equation (7) indicates that when a fragment f_j is transmitted from a server s_{h_1} to a server s_{h_2}, s_{h_1} must have either decrypted f_j or been holding it in its cache.

3.3. Subquery Allocation to Servers Based on Best-First Searches. Subquery allocation to servers in an ordinal distributed database is known to be an NP-complete problem [7]. If k is set to 1 in the (k, n) threshold scheme in an SSDDB, this describes an ordinal distributed database. Accordingly, the issue of subquery allocation to servers becomes an NP-complete problem. This paper is an investigation of best-first search methods with a view to online use.

In combinatorial optimization problem, searching the optimal solution is done by expanding states iteratively. The search process could be described by a tree structure, called the search tree. Figure 2 is an example of the search tree.

Based on preliminary experiments, we heuristically decided the order of decision variables as $x_{i,h}$, $z_{j,h}$, y_{h_1,j,h_2} as shown in Figure 2. The depth of the search tree of each step is r, $p * m$, and $r * m * p$. But at the time when variables $x_{i,h}$ and $z_{j,h}$ are decided, most of variables y_{h_1,j,h_2} have been decided by evaluating (5), (6), and (7).

In the best-first search, the most promising state is chosen by evaluating a function, called the *heuristic evaluation* function. Let us consider a situation expanding the state s in Figure 2; we are going to choose a server to process the subquery sq_i. The *heuristic evaluation* function $G(x_{i,h})$ shows an expected cost when choosing a server s_h, and the search process chooses the state whose $G(x_{i,h})$ is minimal.

The fragments in the SSDDB are encrypted using the (k, n) threshold scheme and the shares are distributed to the various servers. Therefore, in order to process a subquery, the shares are transmitted and the fragment must be decrypted. Let us assume here that a subquery sq_i is processed by a server

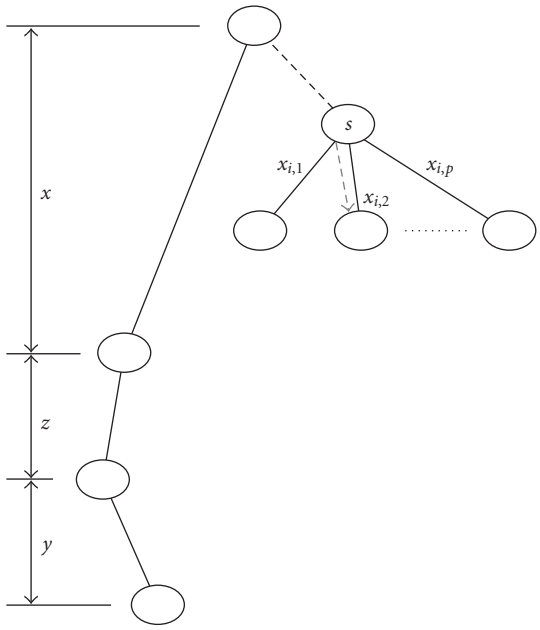

Figure 2: A best-first search tree.

s_h, and that in addition to the cost $w_{i,h}$ of processing sq_i on s_h, there is a cost for the decryption of the fragment f_j, which is needed for the processing in sq_i, and transmitting f_j to server s_h. Accordingly, the minimum possible value for the cost of processing a subquery sq_i on a server s_h is as follows:

$$G(x_{i,h}) = l_h + w_{i,h} + \min_{s_{h_2} \in S} \left\{ d_{j,h_2} + st_{j,h_2} + tf_{j,h,h_2} \right\}. \quad (8)$$

When a subquery sq_i is supposed to be processed on a server s_h, a server to decrypt the corresponding fragment f_j must be decided. The server s_{h_2} which gives the minimal value in (8) is the most promising one. Accordingly, a *heuristic evaluation* function for choosing a server to decrypt a fragment f_j is given as follows:

$$G\left(z_{j,h}\right) = \begin{cases} 0 & \text{if the server } s_h \text{ gives the minimal value} \\ & \text{in (8) for some subquery,} \\ 1, & \text{otherwise.} \end{cases} \quad (9)$$

Since most of variables y_{h_1,j,h_2} could be decided by evaluating (5), (6), and (7), we have not prepared any *heuristic evaluation* function for the variable y_{h_1,j,h_2}.

4. Computational Experiment

This problem of subquery allocation to servers was solved with an ILOG solver (Ver. 5.1, ILOG). The problem was described using OPL. The operating system was Linux (CPU: Celeron 2.0 GHz, Memory: 1 GB).

The following 2 procedures were used for comparison.

No Heuristic. The ILOG Solver engine was used as is without *heuristic evaluation* functions. This solver determines variables in the order they are described. In the experimental model, they were described in the order x, z, y.

EDN. The *heuristic evaluation* function employed by Evrendilek et al. [7], which is referred to as EDN, was used in (10).

$$G(x_{i,h}) = l_h + \sum w_{i,h}. \quad (10)$$

Equation (10) ensures that the processing for subqueries is distributed as widely as possible.

The results are presented in Table 1. Symbols p, m, and r in the table denote the numbers of servers, fragments, and subqueries, respectively. D denotes the dominant cost among the decryption cost (d), subquery processing cost (w) and transmission cost for shares and fragments (t). As indicated by n, all of these costs are about the same. The dominant cost was selected at random from the range $[100, 999]$ and the other costs were randomly selected from $[10, 99]$. Five problems were constructed using identical conditions.

The calculation time was limited to 1 hour and f was the number of searches that were not completed within the time limit. Blank columns indicate that searches were completed for all problems. The value t_i was the mean time required to find a feasible solution (initial feasible solution). The value t_o was the mean time required to find the optimal solution. The value t_f was the mean time required to complete the search. The means for t_o and t_f were taken over the searches that were completed. All times were stated in seconds.

The value ϵ is the ratio between the evaluated values of the initial feasible solution and the optimal solution and represents the mean over problems for which the optimal solution was identified.

The results when p, m, and r were set to identical values and the scale of the problem was varied are listed in the upper part of Table 1. The time t_i was within 0.1 s and the feasible solution was easily found with all the methods. Also, as ϵ shows, the proposed procedure proved to be better than the other procedures at finding the initial feasible solution.

The time required to complete the search grew exponentially with the scale of the problem.

Furthermore, the search was never completed within the time limit under the No Heuristic procedure, even when the problem scale was small. Thus, it is essential to develop a procedure for quickly deriving a useful feasible solution for the subquery allocation problem.

The proposed method was able to find optimal solutions for a greater quantity of the problems within the time limit than the EDN approach. Also, on average, the proposed method was superior from the perspective of times to find the optimal solution and complete the search. It should be noted that the values for t_o and t_f are the means of problems seeking optimal solutions within the time limit. For example, under $p = 8$, $m = 8$, $r = 8$, and $D = w$, for the proposed

Table 1: Experimental results.

p	m	r	D	Proposed					EDN					No Heuristic				
				t_i	t_o	t_f	f	ϵ	t_i	t_o	t_f	f	ϵ	t_i	t_o	t_f	f	ϵ
4	4	4	n	0	.02	.04		1.21	0	.04	.05		2.80	0	14.57	14.58	2	4.43
4	4	4	d	0	.02	.02		1.61	0	.01	.03		4.90	.01	31.93	31.93	1	5.59
4	4	4	w	.01	.02	.02		1.19	0	.02	.03		1.29	0	111.36	111.36	3	4.96
4	4	4	t	.01	.02	.02		1.53	0	.03	.03		3.52	0	.06	.06		3.87
5	5	5	n	.01	.04	.10		1.30	.01	.11	.16		2.73	.01	—	>3600	5	5.51
5	5	5	d	.01	.09	.47		1.19	0	.04	.40		4.65	.01	.16	.74	2	5.30
5	5	5	w	.01	.59	.59		1.23	.01	.03	.04	1	1.59	.01	—	>3600	5	7.72
5	5	5	t	.01	.06	.10		1.37	.01	.11	.16		3.22	.01	131.10	131.14	2	3.81
6	6	6	n	.02	2.81	4.47		1.57	.02	2.06	4.30		3.70	.01	—	>3600	5	6.83
6	6	6	d	.03	1.84	4.71		1.69	.02	1.42	4.48		5.63	.01	—	>3600	5	6.54
6	6	6	w	.02	6.83	6.84	1	1.24	.02	5.62	5.63		1.27	.02	—	>3600	5	7.65
6	6	6	t	.03	.16	.20		1.93	.01	35.52	35.53	1	4.76	.02	—	>3600	5	5.90
7	7	7	n	.04	28.31	29.74		1.84	.02	44.41	49.47		4.37	.03	—	>3600	5	8.26
7	7	7	d	.03	8.66	9.12	1	1.85	.03	3.54	96.98		6.17	.03	—	>3600	5	7.12
7	7	7	w	.04	.10	.13	2	1.37	.03	.16	.18	1	1.65	.03	—	>3600	5	11.48
7	7	7	t	.04	.18	.97		2.26	.04	38.09	39.41	1	5.49	.03	—	>3600	5	7.15
8	8	8	n	.07	12.56	77.55		2.23	.05	54.73	118.16		4.08	.05	—	>3600	5	9.01
8	8	8	d	.07	228.05	297.24	3	1.60	.06	259.21	334.66	3	11.23	.05	—	>3600	5	12.54
8	8	8	w	.06	452.08	452.12	2	1.37	.05	4.38	4.43	4	1.92	.06	—	>3600	5	16.12
8	8	8	t	.06	1.07	1.13		2.13	.06	14.87	15.01	2	7.86	.05	—	>3600	5	9.43
10	4	4	n	.03	.39	.50		1.96	.03	.25	.37		3.97	.03	—	>3600	5	7.27
20	4	4	n	.17	8.67	12.09		1.96	.14	10.12	13.55		2.91	.18	—	>3600	5	9.89
30	4	4	n	.39	3.99	5.33		2.85	.37	500.26	501.19		3.47	.48	—	>3600	5	9.79
40	4	4	n	.89	12.00	16.67	1	3.98	.82	124.77	124.87	1	2.24	1.05	—	>3600	5	7.62
50	4	4	n	1.75	228.79	229.25	2	3.46	1.70	355.59	356.00		4.24	2.15	—	>3600	5	12.47
60	4	4	n	2.86	352.73	369.86		3.51	2.70	1381.34	1391.84	1	2.98	3.44	—	>3600	5	9.67
70	4	4	n	4.42	40.62	40.73	3	3.41	4.38	1261.05	1261.60	2	3.51	5.19	—	>3600	5	13.75
80	4	4	n	7.10	59.42	59.80	2	3.28	7.17	635.63	649.35		3.28	8.60	—	>3600	5	12.12
90	4	4	n	9.20	82.56	730.91	2	3.63	8.89	518.86	529.98	3	3.49	11.16	—	>3600	5	11.05
4	10	4	n	.02	.15	.19		1.22	.01	.12	.18		2.93	.01	16.73	16.77	3	4.34
4	20	4	n	.04	1.18	1.65		1.48	.03	2.03	2.40		2.72	.03	.87	.94	3	4.00
4	30	4	n	.06	2.06	3.33		1.21	.05	4.06	5.09		2.45	.05	2.97	2.98	4	4.33
4	40	4	n	.08	2.03	4.00		1.22	.07	3.92	6.42		2.37	.07	—	>3600	5	4.06
4	50	4	n	.12	212.42	434.96		1.38	.11	121.41	655.80		2.61	.11	—	>3600	5	4.30
4	60	4	n	.16	16.22	38.71		1.24	.14	13.01	51.89		3.52	.14	—	>3600	5	4.55
4	70	4	n	.20	26.32	32.61	1	1.27	.17	415.08	421.50	1	3.14	.17	—	>3600	5	5.00
4	80	4	n	.25	85.89	185.64		1.58	.22	578.80	608.03		2.67	.21	—	>3600	5	4.72
4	90	4	n	.29	69.58	149.28		1.36	.27	1017.21	1127.39		2.80	.26	—	>3600	5	4.39
4	4	10	n	.01	6.53	16.59		1.50	.01	6.34	25.37		2.11	0	—	>3600	5	4.98
4	4	20	n	.04	—	>3600	5	—	.01	—	>3600	5	—	.01	—	>3600	5	—
4	4	30	n	.06	—	>3600	5	—	.02	—	>3600	5	—	.02	—	>3600	5	—
4	4	40	n	.10	—	>3600	5	—	.04	—	>3600	5	—	.03	—	>3600	5	—
4	4	50	n	.14	—	>3600	5	—	.06	—	>3600	5	—	.04	—	>3600	5	—
4	4	60	n	.22	—	>3600	5	—	.07	—	>3600	5	—	.04	—	>3600	5	—
4	4	70	n	.26	—	>3600	5	—	.09	—	>3600	5	—	.04	—	>3600	5	—
4	4	80	n	.36	—	>3600	5	—	.11	—	>3600	5	—	.06	—	>3600	5	—
4	4	90	n	.41	—	>3600	5	—	.14	—	>3600	5	—	.06	—	>3600	5	—

method $t_o = 452.08$ s, while for EDN $t_o = 4.38$ s. However, the proposed method successfully completed 3 problems, while EDN successfully completed only 1 problem. The given values for t_o and t_f for EDN represent those from that single problem, while the values from the proposed method are the means taken over the three problems.

Thus, the proposed method is superior to both EDN and the No Heuristic procedure in terms of both initial feasible solution and calculation time.

Let us turn to some observations about the relationship between problem characteristics and solution methods. First, when the transmission cost is dominant ($D = t$), the searches are completed more quickly than for other problems of similar scale. On the other hand, when the subquery processing cost or decryption cost is dominant ($D = w$ or $D = d$), the time required for search completion increases with the scale of the problem. This seems to be because transmission cost depends greatly on server selection.

We observed changes in the calculation time when only one of the parameters p, m, or r was changed (lower part of Table 1).

It takes more time to derive an initial feasible solution when the number of servers is increased; on the other hand, the time to identify an initial feasible solution remained within 1 s when either the number of fragments or subqueries was increased. The value of ϵ was raised relatively high by expanding the number of servers. Future studies must address the issue of developing procedures for identifying good feasible solutions at high speed in systems incorporating large numbers of servers.

5. Conclusions

It is essential to optimize query processing, as this has a great influence over database performance. This study provides a formulation of the subquery allocation problem in a secret sharing distributed database in terms of a constraint satisfaction problem. A *heuristic evaluation* function is proposed for solutions in best-first searches.

An experiment was conducted using an ILOG Solver to examine the effectiveness of the above function. Some issues remain for the proposed procedure when there are a large number of servers, and further research will be necessary. It will also be necessary to evaluate the performance of this method on an actual system.

Acknowledgment

The authors express their deep appreciation for the Grant-in-Aid for Scientific Research no. 19700060, which supported part of this study.

References

[1] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1999.

[2] T. Miyamoto, Y. Morita, and S. Kumagai, "Vertical partitioning method for secret sharing distributed database system," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E89-A, no. 11, pp. 3244–3249, 2006.

[3] T. Miyamoto, S. Doi, H. Nogawa, and S. Kumagai, "Autonomous distributed secret sharing storage system," *Systems and Computers in Japan*, vol. 37, no. 6, pp. 55–63, 2006.

[4] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote, and P. K. Khosla, "Survival information storage systems," *Computer*, vol. 33, no. 8, pp. 61–68, 2000.

[5] S. Lakshmanan, M. Ahamad, and H. Venkateswaran, "Responsive security for stored data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 818–828, 2003.

[6] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[7] C. Evrendilek, A. Dogac, S. Nural, and F. Ozcan, "Multidatabase query optimization," *Distributed and Parallel Databases*, vol. 5, no. 1, pp. 77–114, 1997.