TR-86-53

ON PROPAGATION OF REDUCTION EFFECTS OF A SEMIJOIN IN DISTRIBUTED QUERY PROCESSING

by

Hyunchul Kang

&

Nick Roussopoulos

TR-86-53

September 1986

On Propagation of Reduction Effects of a Semijoin in Distributed Query Processing

By

Hyunchul Kang

Nick Roussopoulos

Database Systems Group
Department of Computer Science
University of Maryland
College Park, MD 20742

This work was supported in part by the Systems Research Center
at University of Maryland, College Park

Abstract

In distributed query processing, the semijoin is used as an effective operator in reducing relations referenced in the query and thus reducing the total amount of data transmission. A semijoin is associated with two quantities: cost and benefit. These two quantities of semijoins are the useful ingredients in many heuristic algorithms proposed to generate a semijoin program which is a sequence of semijoins and is executed as a query pre-processing strategy. In this paper, in addition to the cost and benefit, we associate a semijoin with the third quantity: *reduction propagation*. It measures the propagation of reduction effects of a semijoin to other semijoins. We define this propagation in the context of distributed query processing and by precisely quantifying it, we present a heuristic algorithm to generate a semijoin program. The simulation results show that the new algorithm provides a considerable improvement over those based only on the cost and benefit.

Table of Contents

1 Introduction 1

2 Effectiveness of Semijoin Program 2

3 Reduction Propagation 4

3.1 Heuristics on Propagation 4

3.2 Quantification of Propagation 5

4 Heuristic Algorithm 6

5 Simulation 7

6 Conclusion 10

1. Introduction

In a distributed database system, processing a query involves data transmission among the different sites of the computer network. In a system where the communication cost is a dominant factor among the costs of system resources, that is, the local processing cost is negligible compared to the data transmission cost, it is essential to reduce the total amount of data transmission in processing a query. The optimization amounts to the minimization of the total amount of data transmission.

The costly operation which involves data transmission in distributed query processing is the join between two relations at different sites. When such join is required in the query, instead of joining the two directly which involves costly transmission of one relation to the other site, semijoins [Bernstein and Chiu 81] can be used. A semijoin reduces a relation by transmitting the joining attribute only and eliminating tuples which are not joinable. The final join is computed with the reduced relations. In general, for a query with arbitrary complexity, a semijoin program which is a sequence of semijoins can be executed as a query pre-processing strategy to reduce relations referenced in the query. The query is answered with the reduced relations.

The algorithms to generate the optimal processing strategy have been proposed for some restricted class of queries such as simple query [Hevner 79], chain query [Chiu et al. 84], tree query [Yu et al. 80] [Chiu and Ho 80] and star query [Chen and Li 84a,b]. However, for general queries with arbitrary complexity, it has been proved that even for queries with very primitive complexity, the problem of generating the optimal processing strategy is NP-hard [Hevner 79][Yu et al. 82]. Therefore, the computationaly feasible algorithms to generate the processing strategies for general queries depend on heuristics. The semijoin takes a significant relevance to those heuristic algorithms [Wong 77][Bernstein et al. 81][Black and Luk 82][Hevner and Yao 79] [Apers et al. 83][Cheung 82][Chang 82][Yu et al. 82][Yu and Chang 83].

A semijoin is associated with two quantities: *cost* and *benefit*. Consider a semijoin from R_i to R_j on attribute A denoted as $R_i \ -A \rightarrow R_j$ where R_i and R_j are at different sites. The cost of $R_i \ -A \rightarrow R_j$ is the transmission cost for $R_i[A]$ and the benefit is the reduction done to R_j,

that is, the reduction in the transmission cost for R_j when R_j is transmitted to some site where the query is answered. These two quantities of semijoins are the useful ingredients in many heuristic algorithms proposed to generate a semijoin program. The representative example among them is the algorithm in SDD-1 [Rothnie et al. 80] where only the cost-effective semijoins (benefit is greater than cost) are considered to be included in the semijoin program [Bernstein et al. 81].

In this paper, in addition to the cost and benefit, we associate a semijoin with the third quantity: *reduction propgation* (may be abbreviated to propagation, hereafter). It measures the propagation of reduction effects of a semijoin to other semijoins. Consider semijoins $s : R_i \rightarrow R_j$ and $t : R_j \rightarrow R_k$. After s reduces R_j to R_j', t is transformed into $t' : R_j' \rightarrow R_k$. More presicely, the reduction effects on R_j by s is propagated to t so that it is transformed into t'. We show that given a semijoin, its propagation can be defined in the context of distributed query processing and can be precisely quantified according to the definition. This leads us to associate a semijoin with three quantities: *cost, benefit* and *propagation.* While the cost and benefit of a semijoin determines the cost-effectiveness of that particular semijoin itself, the propagation measures its effectiveness in terms of its interactions with other semijoins. Thus using these three quantities, more effective semijoin programs can be generated than those generated using the cost and benefit only. We present an algorithm based on the propagation as well as the cost and benefit. The simulation results show that the new algorithm provides a considerable improvement over the algorithm in SDD-1 which is based only on the cost and benefit.

In section 2 of this paper, we note that the effectiveness of a semijoin program depends on the propagation. In section 3, we review the heuristics on the propagation proposed in the past and we define the propagation as a quantity associated with a semijoin. In section 4, we present our algorithm and in section 5, we report the simulation results in detail. Conclusions are in section 6.

2. Effectiveness of Semijoin Program

The effectiveness of a semijoin program consists in two aspects: the cost-effectiveness of each semijoin in the program and the order of semijoins in the program. The cost-effectiveness of a

semijoin can be measured by its cost and benefit. If the benefit of a semijoin is greater than the cost, it is a cost-effective semijoin. The meaning the cost-effective semijoin has in distributed query processing is that it is always profitable to execute a cost-effective semijoin. To show this point, consider a cost-effective semijoin $R_1 \rightarrow R_2$ which would reduce R_2 to R_2'. Let the transmission cost of R_2 and R_2' be t and t', respectively. Then, the benefit of $R_1 \rightarrow R_2$ is $b = t - t'$ and since $R_1 \rightarrow R_2$ is cost-effective, the cost $c < b$. We have $c + t' < t$. Thus, it is profitable to execute $R_1 \rightarrow R_2$.

As for the order of semijoins, there are two issues related. The first one is that given a semijoin program, by reordering some of semijoins, we can decrease the cost of the program without decreasing the benefit of the program [Bernstein et al. 81][Luk and Luk 83][Chen and Li 84c,d].[1] For example, given a semijoin program $SJP : R_1 \rightarrow R_2, R_2 \rightarrow R_3, R_4 \rightarrow R_1$, we can improve it by executing $R_4 \rightarrow R_1$ first, that is, in the order of $SJP' : R_4 \rightarrow R_1, R_1 \rightarrow R_2, R_2 \rightarrow R_3$. Since R_1 is reduced by $R_4 \rightarrow R_1$ before $R_1 \rightarrow R_2$ is executed, the cost of $R_1 \rightarrow R_2$ in SJP' is less than or at least equal to that of $R_1 \rightarrow R_2$ in SJP and the benefit of $R_1 \rightarrow R_2$ in SJP' is greater than or at least equal to that of $R_1 \rightarrow R_2$ in SJP. Similar improvements are applied to $R_2 \rightarrow R_3$ as well. This reordering technique can be used as the post optimization to a given semijoin program.

Another issue related to the order of semijoins can be expressed in the following question: in which sequence should semijoins be included in the semijoin program ? For algorithms which generate the semijoin program by adding one semijoin at a time to the current program, this question is equivalent to which semijoin should be selected as the next semijoin in the semijoin program. In orther words, when more than one semijoins are available to reduce relations, which should be the one to be executed prior to the rest. For example, when R_1 and R_2 can be reduced by each other, the question is which order is better, $R_1 \rightarrow R_2, R_2 \rightarrow R_1$ or $R_2 \rightarrow R_1, R_1 \rightarrow R_2$? This question can be answered by measuring the propagation of the reduction effects of a semijoin to other semijoins. We note that this issue is independent of the above semijoin reordering problem.

[1] The cost of a semijoin program is the sum of costs of semijoins in the program and similarly, the benefit of a semijoin program is the sum of benefits of semijoins in the program.

3. Reduction Propagation

3.1. Heuristics on Propagation

In previous research in the literature, the propagation of semijoins has been considered in generating the semijoin program. In SDD-1, among all the cost-effective semijoins, the most cost-effective semijoin is selected as the next semijoin in the semijoin program [Bernstein et al. 81].[2] This heuristic is concerned primarily with the effectiveness of each semijoin in the program rather than with the propagation of the reduction effects of a semijoin to other semijoins. However, by selecting the most cost-effective semijoin each time, a reasonable propagation is likely to be achieved indirectly.

Black and Luk [1982] proposed an extended heuristic in the sense that the aspect of the propagation of the reduction effects of a semijoin is involved in selecting the next semijoin more directly than in SDD-1. Instead of the most cost-effective semijoin, their algorithm selects the semijoin whose sum $(c + p)$ is the smallest where c is the cost of the semijoin and p is the size of the joining attribute reduced by the semijoin. More precisely, suppose a semijoin $R_1 \text{—} A \rightarrow R_2$ would reduce R_2 to R_2' . Then for $R_1 \text{—} A \rightarrow R_2$, $c = R_1[A]$ and $p = R_2'\ [A]$. In $(c + p)$, the second term p is the prediction of the propagation of the reduction effects of a semijoin to other semijoins.

In other heuristic, the propagation can be achieved as follows: Relations referenced in the query with the common joining attributes are grouped together and for each group, a chain of relations is formed so that the first relation in the chain reduces the second, the second reduces the third and so on until all the reduction effects are fully propagated to the last relation and then the last relation is used to reduce back the others in the chain. The rationale behind this heuristic is that the last relation in the chain can be fully reduced within the group and therefore, reducing back the others in the chain is usually very effective. Various algorithms to generate the

[2] The cost-effectiveness of a semijoin is qunantified as the difference between the benefit b and the cost c . The most cost-effective semijoin is, thus, the one whose difference $(b - c)$ is the greatest of all.

semijoin program based on this heuristic were proposed in [Apers et al. 83] [Cheung 82][Chang 82][Yu et al. 82].

However, none of the above three heuristics is either directly concerned with or precisely quantifying the propagation of semijoins in generating the semijoin program. In SDD-1, the weak point is that the most cost-effective semijoin does not necessarily have the furthest propagation. Black and Luk's heuristic is appropriate for queries with a single joining attribute but it does not adequately deal with queries joining relations over more than one attributes. In the third heuristic, little attention is paid to the cost-effectiveness of each semijoin in the semijoin program. Besides, the heuristic does not precisely relate the reduction of a particular relation in one group with its reductions in other groups when it has more than one joining attributes.

3.2. Quantification of Propagation

In this subsection, we define the propagation of a semijoin in the context of distributed query processing and for a given semijoin, we precisely quantify its propagation according to the definition. Then given a semijoin, its propagation is associated with it as one of its unique quantity like the cost and benefit.

For a semijoin s, let C_s and B_s be the cost and the benefit of s, respectively. Consider semijoins $s: R_i \rightarrow R_j$ and $t: R_j \rightarrow R_k$. We have the following property about interactions between s and t.

Property Suppose s reduces R_j to R_j' and let t' be $R_j' \rightarrow R_k$.

Then, $C_t \geq C_{t'}$ and $B_{t'} \geq B_t$.

In words, t is improved into t' in the sense that t' can have more or at least equal benefit and pay less or at least equal cost than t in reducing R_k. Thus, if t is cost-effective, then t' is more or at least as much cost-effective. If t is not cost-effective, t' may be cost-effective or at least improved over t. We note that t is improved into t' because the reduction effects on R_j by s is propagated to t. Similary, the reduction effects on R_j by s is propagated to all other semijoins of type t with respect to s. We have the following definition.

Definition The propagation of the reduction effects of a semijoin s to other semijoins is the total improvements in all interacting semijoins of type t with respect to s. The improvements in a semijoin consist of the decrease of its cost and of the increase of its benefit.

Consider a semijoin $R_1 \rightarrow R_2$. The propagation of $R_1 \rightarrow R_2$ is defined as the measurement on how much the reduction of R_2 improves other semijoins which interacts with $R_1 \rightarrow R_2$. The propagation of a semijoin can be quantified according to the above definition as follows: Given a semijoin $s : R_i \rightarrow R_j$, suppose s would reduce R_j to R_j'. Let n be the number of all semijoins of type $t : R_j \rightarrow R_k$ and let t_i be one of such semijoins $R_j \rightarrow R_{k_i}$ and t_i' be $R_j' \rightarrow R_{k_i}$, $i = 1$, ..., n. Then, the propagation of s is

$$p = \sum_{i=1}^{n} [(C_{t_i} - C_{t_i'}) + (B_{t_i'} - B_{t_i})] \quad (1)$$

The term $(C_{t_i} - C_{t_i'})$ quantifies the cost decrease of t_i and the term $(B_{t_i'} - B_{t_i})$ quantifies the benefit increase of t_i, $i = 1$, ..., n.

4. Heuristic Algorithm

Since the propagation of a semijoin is the quantity measuring how much the reduction of a relation by the semijoin improves other semijoins, it is a useful ingredient like the cost and benefit in generating the semijoin program. In this section, we present a heuristic algorithm based on the propagation of semijoins as well as their cost and benefit.

Our algorithm generates a semijoin program by selecting one semijoin at a time. It considers cost-effective semijoins only. Each time among all cost-effective semijoins, the semijoin which maximize the following weighted formula is selected as the next semijoin in the semijoin program.

$$w_1 \times (b - c) + w_2 \times p$$

where w_1 and w_2 are the weights of the cost-effeciveness and of the propagation of the semijoin, respectively. Since only cost-effective semijoins are considered, p in the above formula includes cost-effective semijoins only. That is, the equation (1) for p in the previous section is modified as

$$p = \sum_{i=1}^{n} [\,(C_{t_i} - C_{t_i'}) + (B_{t_i'} - B_{t_i})\,], \quad B_{t_i'} > C_{t_i'}$$

The condition $B_{t_i'} > C_{t_i'}$ excludes the semijoins t_i' which are not cost-effective when the improvements in t_i are summed up, $i = 1, ..., n$.

Our algorithm balances between the cost-effectiveness and the propagation of semijoins. One extreme of our algorithm is when $w_2 = 0$. In this extreme, our algorithm is the same as that in SDD-1 and based only on the cost and benefit. The other extreme is when $w_1 = 0$. In this extreme, it is based only on the propagation. Inbetween the two, various pair of (w_1, w_2) values can be used. When $w_2 = 0$, we denote our algorithm as SDD-1. When $w_1 = 0$, we denote our algorithm as P. When $w_1 \neq 0$ and $w_2 \neq 0$ and some weight $w = \frac{w_2}{w_1}$ is used, we denote our algorithm as P(w).[3] A formal description of our algorithm can be found in Appendix.

5. Simulation

The algorithms SDD-1, P and P(w) are compared by simulation. We consider randomly generated general queries joining n relations over a joining attributes. Queries generated are with arbitrary complexity and also realistic in the sense that their join patterns are meaningful as much as in practical queries. Values of a and n used are in Table 1.

a	n
1	2,3,4
2	3,4,5,6
3	4,5,6,7
4	5,6,7

Table 1

The *database profile* which contains statistics on relations and on domains of the joining attributes is generated using a random number generator as follows: Let $U[i,j]$ represent a uniform distribution over the range $[i,j]$. For each domain of a joining attribute, the cardinality,

[3] P stands for *propagation*.

that is, the number of all possible values is $U[100,10000]$; the average size of a value is $U[4,20]$.[4] For each relation, the cardinality, that is, the number of tuples is $U[100,10000]$; the size of a tuple is the sum of the size of joining attributes and the size of non-joining attributes where the size of non-joining attributes is $U[0,100]$; the selectivity of each joining attribute which is the ratio of the number of unique values occurred in the attribute to the cardinality of its domain is $U[0,1]$. These parameters are summarized in Table 2 and 3.

Joining Domain	
cardinality	$U[100,10000]$
size of a value	$U[4,20]$

Table 2

Relation	
cardinality	$U[100,10000]$
size of non-joining attribute	$U[0,100]$
selectivity	$U[0,1]$

Table 3

For each pair of values (n ,a), 500 queries and database profiles are generated and, for each query and database profile, semijoin programs are generated by algorithms SDD-1, P and P(w) for various w values ranging from 1 to 200 and their costs are computed, respectively. Then, the average cost of those 500 semijoin programs are computed for SDD-1, P and P(w), respectively. In what follows, comparisons are done and improvements are measured in terms of these average costs.

In all cases, P and P(w) show improvements over SDD-1. There are two patterns for the relationship among P, P(w) and SDD-1. In one, P is better than P(w) for all w values and in the other, P(w) is better than P for almost all w values. In Figure 1 and 2, each pattern is depicted.

[4] The unit for the size of data is *byte*.

The improvements by P(w) over SDD-1 is measured as follows:

$$\frac{cost(\ SDD\text{-}1\ ) - cost(\ P(w)\ )}{cost(\ P(w)\ )} \times 100\ (\%)$$

For queries with only one joining attribute ($a = 1$), a slight improvement (0.3% to 1.3%) is shown but for queries with more than one joining attributes ($a = 2,3,4$), a considerable improvement (15% to 45%) is shown. In Figure 3, these improvements by P(w) over SDD-1 is shown for various values of w ranging from 0 to 200 and for all pair of values (n ,a). We can see that for more complicated queries, the higher improvements are shown.

While P and P(w) show a considerable improvement over SDD-1, there is vitually no difference shown between P and P(w). Since P is based only on the propagation of semijoins and not on their cost-effectiveness, this result implies that the propagation is much more important factor to be considered in generating semijoin programs than the cost-effectiveness factor. In Figure 4, for all pair of values (n ,a), SDD-1, P and P(w) are compared in terms of the cost of their semijoin programs where P(w) is taken as the minimum among P(w)'s for all w values.

We also measured the length of semijoin programs generated by SDD-1, P and P(w), respectively where the length of a semijoin program is defined as the number of semijoins in the program. This is to see how different in length of semijoin programs when semijoins in them are selected based on different heuristics and thus in different orders. Figure 5 shows the comparison where MAX is the maximum number of semijoins in the program assuming that a semijoin may be included in the program at most once and, P(w) is the length when the cost of its semijoin program is the minimum among P(w)'s for all w values. We can see almost no difference in length among P, P(w) and SDD-1. This indicates that the difference in the order of semijoins in the semijoin program can make a considerable difference in the amount of reduction of relations with almost the same number of semijoins. Therefore, the propagation of semijoins should be considered in generating the semijoin program.

6. Conclusion

We defined the reduction propagation (the propagation of the reduction effects of a semijoin to other semijoins) in the context of distributed query processing and given a semijoin, we precisely quantified its propagation according to the definition. Thus, a semijoin is now associated with three quantities: cost, benefit and propagation. Based on these three quantities of semijoins, we presented a heuristic algorithm to generate a semijoin program. The simulation results showed that the new algorithm provided a considerable improvement over the algorithm in SDD-1 which is based only on the cost and benefit of semijoins. Other results from the simulation indicates that the propagation of semijoins should be considered in generating the semijoin program.

In this paper, we considered only positive aspect of reduction propagation, that is, the propagation to other semijoins was quantified as the sum of improvements in other semijoins. However, there is also the negative aspect of reduction propagation. Consider semijoins $s: R_i \rightarrow R_j$ and $t: R_k \rightarrow R_j$. After s reduces R_j to R_j', t is transformed into $t': R_k \rightarrow R_j'$. We note that t is degenerated into t' in the sense that their costs remain the same but the benefit of t' is less than or at most equal to that of t. More general heuristic could consider both the positive and the negative aspects of the reduction propagation.

By considering the propagation of a semijoin as a factor of it like the cost and benefit, we showed a direction toward the generalization of the framework for the heuristic algorithms which select one semijoin at a time in generating semijoin programs. Each semijoin can be compared to each other in terms of several factors such as the cost, benefit, propagation and so on to be selected as the next semijoin in the semijoin program. The more factors are available, the more effective semijoin program can be generated. Further factors of a semijoin and the interrelationship among those need be investigated. We also need to study the tradeoff between the overhead in computing those factors and the effectiveness of resulting semijoin programs.

Appendix: Algorithm P(w)

For a query Q, let J_Q be the set of join clauses of Q, and let $J_Q{}^+$ be the transitive closure of J_Q, that is, the set of all the join clauses inferred by J_Q. A join clause $R_i.A = R_j.A$ in $J_Q{}^+$ is associated with two semijoins: $R_i \text{—}A\rightarrow R_j$ and $R_j \text{—}A\rightarrow R_i$. For a query Q, let S_Q be the set of all the possible semijoins associated with $J_Q{}^+$. S_Q is partitioned into two subsets S_Q^e and $\overline{S_Q^e}$ where S_Q^e is the set of all the cost-effective semijoins and $\overline{S_Q^e}$ is $S_Q - S_Q^e$.

For a semijoin s, let C_s and B_s be the cost and the benefit of s, respectively. Given a semijoin $s: R_i \rightarrow R_j$, suppose s would reduce R_j to $R_j{}'$. Let n be the number of all semijoins of type $t: R_j \rightarrow R_k$ and let t_i be one of such semijoins $R_j \rightarrow R_{k_i}$ and $t_i{}'$ be $R_j{}' \rightarrow R_{k_i}$, $i = 1, ..., n$. Let P_s be

$$P_s = \sum_{i=1}^{n} [\,(C_{t_i} - C_{t_i'}) + (B_{t_i'} - B_{t_i})\,], \qquad B_{t_i'} > C_{t_i'}$$

For some value w used in algorithm P(w), let the function f_w of a semijoin s be

$$f_w(s) = (B_s - C_s) + w \times P_s$$

Then, algorithm P(w) is described as follows:

Algorithm P(w)

input: query Q
output: semijoin program SJP

begin
    SJP $\leftarrow$ *null.*
    initiallize S_Q^e.

    while ($S_Q^e \neq \emptyset$) *do*
        begin
            select s^* *such that* $f_w(s^*) = \max_{s \in S_Q^e} f_w(s)$.
            $SJP \leftarrow SJP \,@\, <s^*>$.
                /* *append* s^* *at the end of SJP* */
            update S_Q^e.
        end
end

References

[Apers et al. 83]
Apers, P., Hevner, A.R. and Yao, S.B., *Optimization Algorithm for Distributed Queries,* IEEE Trans. Softw. Eng. SE-9,1, Jan. 1983.

[Bernstein and Chiu 81]
Bernstein, P.A. and Chiu, D.W., *Using Semi-Joins to Solve Relational Queries,* J. ACM 28,1, Jan. 1981.

[Bernstein et al. 81]
Bernstein, P.A., Goodman, N., Wong, E., Reeve, C. and Rothnie, J.B., *Query Processing in a System for Distributed Databases (SDD-1),* ACM Trans. Database Syst. 6,4, Dec. 1981.

[Black and Luk 82]
Black, P. and Luk, W., *A New Heuristic for Generating Semi-Join Programs for Distributed Query Processing,* Proc. IEEE COMPSAC 1982.

[Chang 82]
Chang, J., *A Heuristic Approach to Distributed Query Processing,* Proc. 8th Int. Conf. on Very Large Data Bases, 1982.

[Chen and Li 84a]
Chen, A.L.P. and Li, V.O.K., *Deriving Optimal Semi-Join Programs for Distributed Query Processing,* Proc. IEEE INFOCOM 1984.

[Chen and Li 84b]
Chen, A.L.P. and Li, V.O.K., *Optimizing Star Queries in a Distributed Database System,* Proc. 10th Int. Conf. on Very Large Data Bases, 1984.

[Chen and Li 84c]
Chen, A.L.P. and Li, V.O.K., *Improvement Algorithms for Semijoin Query Processing Programs in Distributed Database Systems,* IEEE Trans. on Computers C-33,11, Nov. 1984.

[Chen and Li 84d]
Chen, A.L.P. and Li, V.O.K., *Improving Semi-Join Programs for Distributed Query Processing,* Proc. IEEE COMPSAC 1984.

[Cheung 82]
Cheung, T.Y., *A Method for Equijoin Queries in Distributed Relational Databases,* IEEE Trans. on Computers C-31,8, Aug. 1982.

[Chiu et al. 84]
Chiu, D.W., Bernstein, P.A. and Ho, Y., *Optimizing Chain Queries in a Distributed Database System,* SIAM J. COMPT. 13,1, Feb. 1984.

[Chiu and Ho 80]
Chiu, D.W. and Ho, Y., *A Methodology for Interpreting Tree Queries into Optimal Semi-Join Expressions,* Proc. ACM SIGMOD Int. Conf. on Management of Data, 1980.

[Hevner 79]
Hevner, A.R., *The Optimization of Query Processing on Distributed Database Systems*, Ph.D. Dissertation, Dept. of Computer Science, Purdue Univ., 1979.

[Hevner and Yao 79]
Hevner, A.R. and Yao, S.B., *Query Processing in Distributed Database System*, IEEE Trans. on Softw. Eng. SE-5, 3, May 1979.

[Luk and Luk 83]
Luk, W.S. and Luk, L., *Optimizing Semi-Join Programs for Distributed Query Processing*, Proc. 2nd Int. Conf. on Data Bases, 1983.

[Rothnie et al. 80]
Rothnie, J.B., Bernstein, P.A., Fox, S., Goodman, N., Hammer, M., Landers, T., Reeve, C., Shipman, D. and Wong, E., *Introduction to a System for Distributed Databases (SDD-1)*, ACM Trans. on Database Syst., 5,1, Mar. 1980.

[Wong 77]
Wong, E., *Retrieving Dispersed Data from SDD-1: A System for Distributed Databases*, Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, 1977.

[Yu and Chang 83]
Yu, C.T. and Chang, C., *On the Design of a Query Processing Strategy in a Distributed Database Environment*, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1983.

[Yu et al. 80]
Yu, C.T., Lam, K. and Ozsoyoglu, M.Z., *Distributed Query Optimization for Tree Queries*, Dept. of Information Eng., Univ. of Illinois at Chicago Circle, Jul. 1980.

[Yu et al. 82]
Yu, C.T., Lam, K., Chang, C. and Chang, S., *Promising Approach to Distributed Query Processing*, Proc. 7th Berkeley Workshop on Distributed Data Management and Computer Networks, 1982.

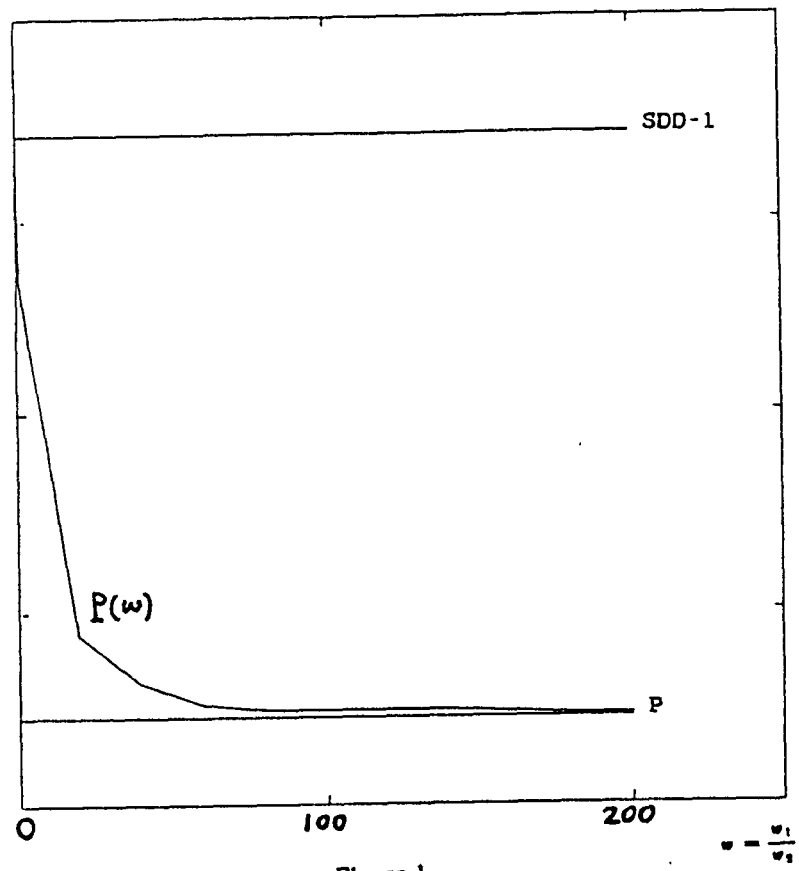

Figure 1

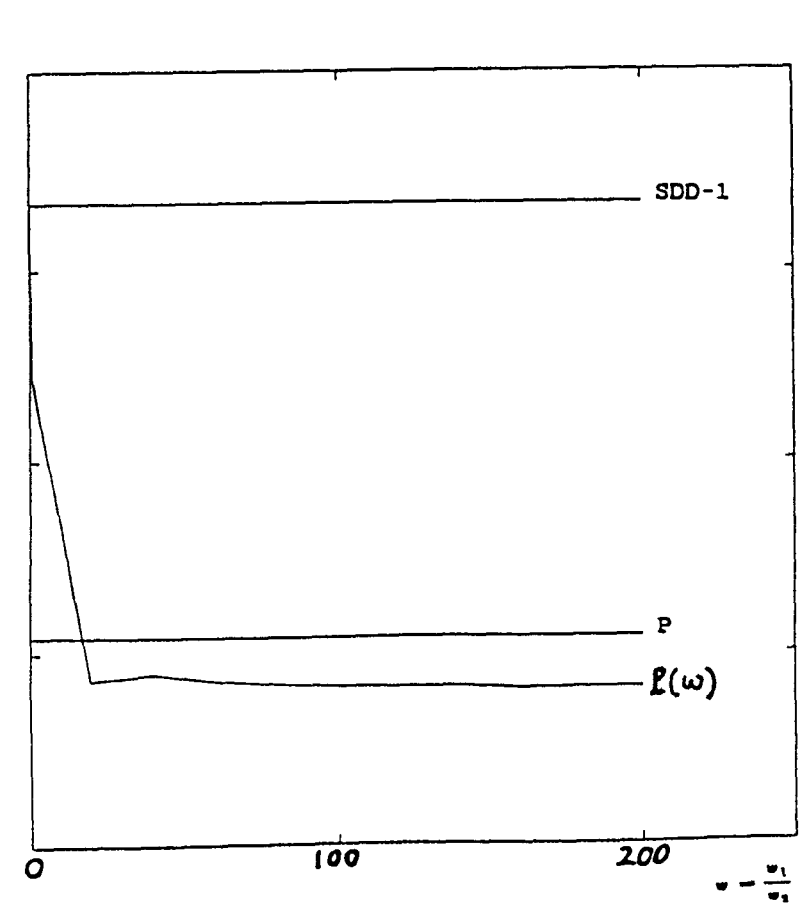

Figure 2

Improvement

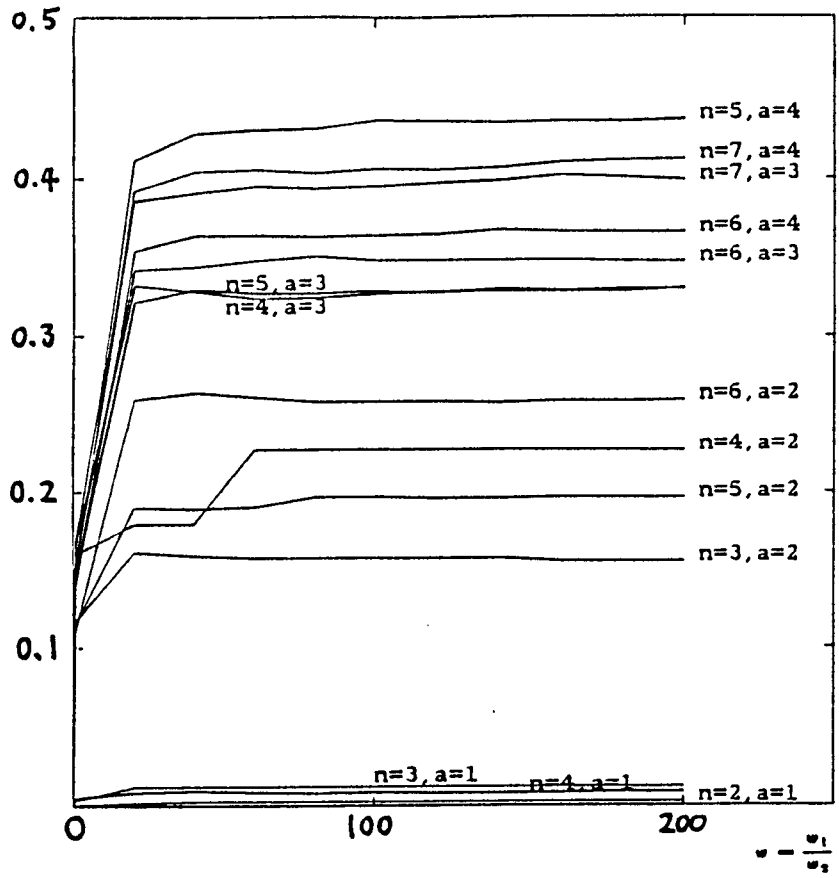

Figure 3

Cost

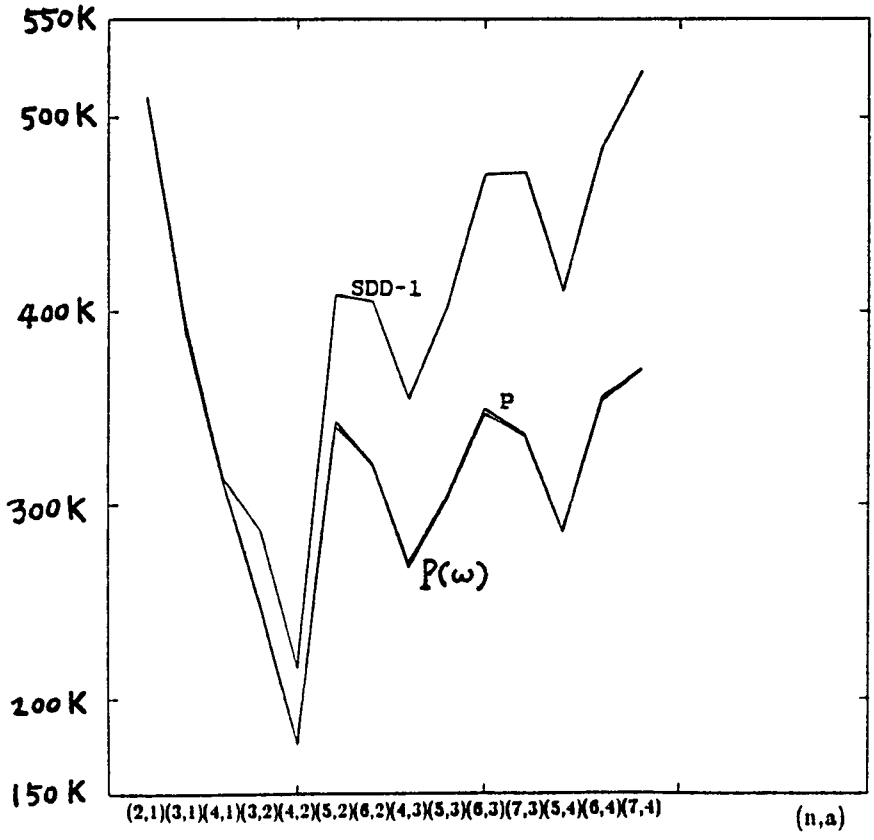

Figure 4

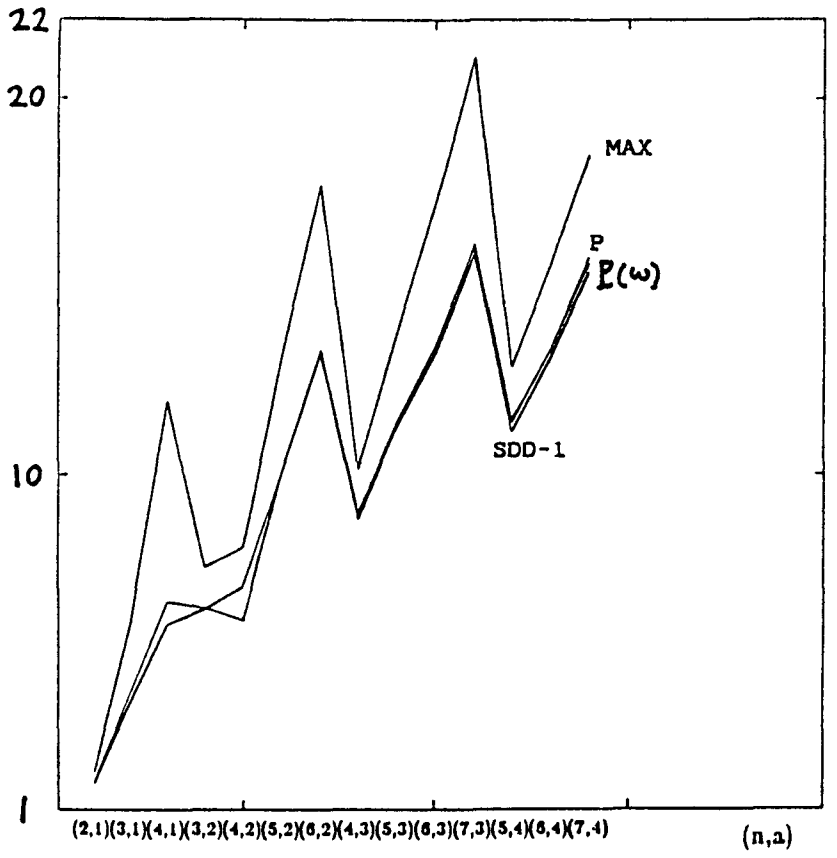
Length
22
20
10
1
MAX
P
P(ω)
SDD-1
(2,1)(3,1)(4,1)(3,2)(4,2)(5,2)(6,2)(4,3)(5,3)(6,3)(7,3)(5,4)(6,4)(7,4)
(n,a)

Figure 5