Hash-Semijoin: A New Technique for Minimizing Distributed Query Time *

Tung-Shou Chen† Arbee L. P. Chen‡ and Wei-Pang Yang§

Abstract

Traditional semijoin and semijoin with multiple hash functions are two important methods to reduce the data transmission cost in DDBS. However, these two methods are sometimes inefficient, due to their complexity and lack of flexibility. To remedy these problems, we propose a new technique, named hash-semijoin. *We identify the situations in which it performs better than traditional semijoin and semijoin with multiple hash functions. The optimal hash functions used in hash-semijoins are also studied.*

1 Introduction

In distributed relational databases, query processing often requires shipping relations between different sites [6]. To reduce the data transmission cost, *semijoins* (SJ for short) were introduced [2, 3, 4].

The SJ operation has the important property of reducing the size of the operand relation. When the main cost component considered by the query processor is communication, an SJ is particularly useful for improving the processing of distributed join operations as it reduced the size of data exchanged between sites. However, using SJs may result in an increase in the number of messages and in the local processing time. The early distributed DBMSs, such as SDD-1 [4], which were designed for slow wide area networks, make extensive use of SJs. Some recent systems, such as R^* [18], assume faster networks and do not employ SJs. Rather, they perform joins directly. Nevertheless, SJs are still beneficial in the context of fast networks when they induce a strong reduction of the join operand [11].

An SJ from relation R_i to relation R_j, denoted by $R_i \xrightarrow[sj]{} R_j$, is defined as $\prod_{R_j}(R_i \leftrightarrow R_j)$, where $R_i \leftrightarrow R_j$ is the join of R_i and R_j, and $\prod_A(B)$ is the projection of relation B on the attributes of relation A. In a distributed database system, SJ can be implemented as the following two methods:

1. **Traditional Semijoin [TSJ]:** Project R_i on the join attribute (of the join between R_i and R_j), then ship this projection to the site of R_j and perform the join with R_j [2, 3, 4].
2. **Semijoin with multiple hash functions [SJM]:** Project R_i on the join attribute, then hash the join attributok by multiple hash functions and use the results as the addresses into the bit-arrays, respectively. That is, a set of hashing functions h_1, h_2, ..., h_m are used, each associated with a bit-array B_1, B_2, ..., B_m, respectively. For each value v of join attribute, all of the corresponding bit in each B_i must be set (i.e., $B[h_1(v)] = 1$, $B[h_2(v)] = 1$, ..., $B[h_m(v)] = 1$). (In [10] it is shown that increasing m causes the probability of *collisions* to approach 0.) Then, ship these bit-arrays to the site of R_j. Each tuple of relation R_j, whose join attribute value is u, belongs to the SJ if $B[h_1(u)] = 1$, $B[h_2(u)] = 1$, ..., $B[h_m(u)] = 1$ [10].

TSJ has been generally discussed in many papers [2, 3, 4, 7, 8, 9]. In this case, suppose the relation sizes of R_i and R_j and the transmission parameters are given. The *cost* and the *benefit* of $R_i \xrightarrow[sj]{} R_j$ are fixed [6]. The choice of this SJ is therefore a yes-or-no problem. There does not exist any middle solution (i.e., cost and benefit) to support more choices to query processor.

SJM has been suggested by Babb and Valduriez [10, 13, 14, 11]. If the size of bit-arrays and the number of hash functions can be limited, the cost and the benefit of $R_i \xrightarrow[sj]{} R_j$ are also fixed [10]. The choice of an SJ is still a yes-or-no problem. Moreover, in order to reduce collisions, many hash functions must be used. SJM is therefore complex.

To alleviate the above problems, in this paper, we propose a new technique, named ***hash-semijoin*** (HSJ for short), to reduce the data transmission cost of distributed queries. This method uses only one hash function for each HSJ; and these hash functions used in HSJs may be any function, i.e., collisions are acceptable in the hash functions of HSJs. Intuitively, since collisions discourage HSJs from reducing the size of operand relation [14], the reduction (or *selectivity* [2]) of HSJs are worse than SJs; but HSJs may still be beneficial due to the following two reasons:

1. **Flexibility:** The cost and benefit of an HSJ are

*This research was supported by the Republic of China National Science Council under Contract No. NSC 81-0408-E-007-12.

†Dept. of Computer Science and Information Engineering, National Chiao Tung University, Taiwan, ROC.

‡Dept. of Computer Science, National Tsing Hua University, Taiwan, ROC.

§Dept. of Computer and Information Science, National Chiao Tung University, Taiwan, ROC.

0-8186-2755-7/92 $03.00 © 1992 IEEE

affected by the probability of the collisions of the hash function used in this HSJ. Since the hash function used in the HSJ may be any function, there are many hash functions which can be chosen and their probabilities of collisions are all different. HSJs are therefore flexible.

2. **Simplicity:** Since there is only one hash function used in each HSJ, there is only one bit-array used in each HSJ. The processing overhead and transmission cost of HSJs are therefore smaller than that of SJM. Moreover, since the model of HSJs are simpler than that of SJM, the research to choose a suitable hash function in an HSJ is also simpler than that of SJM.

In this paper, the benefit of HSJ is shown.

The rest of the paper is organized as follows. In section 2, we define the HSJ and derive its transmission cost and selectivity. Section 3 uses HSJs to improve distributed queries in total transmission cost. The benefit of HSJs is shown and the hash functions of HSJs are discussed. We conclude in section 4.

2 Hash-semijoins

2.1 The definition of hash-semijoins

Given a join between relations R_i and R_j with join attribute a. The ***hash-semijoin*** (i.e., HSJ) from R_i to R_j, denoted by $R_i \xrightarrow[hsj]{a} R_j$, is defined as

$$\{t_j \in R_j | \exists\ t_i \in R_i\ \ni\ h_{ij}(t_j.a) = h_{ij}(t_i.a)\},$$

where h_{ij} is the hash function associated with the HSJ $R_i \xrightarrow[hsj]{a} R_j$. In a distributed database system, it is implemented as follows: Project R_i on the join attribute, hash the join attribute by h_{ij} and use the result to set the bit-array B_{ij} (i.e., $B_{ij}[k]$ is set to 1 if there exists a join attribute value v in relation R_i such that $h_{ij}(v) = k$); then ship B_{ij} to the site of R_j; and finally select those tuples of R_j, whose join attribute value is u and $B_{ij}[h_{ij}(u)] = 1$, as the result of the HSJ.

2.2 The model of distributed query processing

A query, denoted (TL, Q), consists of two components [7]: the target list (TL) and the qualification (Q). The target list contains target attributes that are of interest to the query, i.e., attributes that will appear in the answer. The qualification is a conjunction of clauses which describe the query.

We assume that after initial local processing, all clauses in a qualification are of the form $R_i.a = R_j.a$. For simplifying our descriptions, we moreover assume that each sites contains one relation, that there is only one copy of each relation.

A distributed query is then processed by SJ and HSJ as follows:

1. **Initial local processing:** all local operations including selections and projections are processed.

2. **SJ and HSJ processing:** the only operations left after initial local processing are joins between relations at different sites. An SJ and HSJ *program*, which uses both SJ and HSJ to reduce the size of relations, is derived from the remaining join operations and executed.

3. **Final processing:** all relations which are needed to calculate the answer of the query are transmitted to a final site where joins are performed and the answer to the query obtained.

2.3 The linear transmission cost model

Consider the transmission cost of DDBS. Let C_{ij} and D_{ij} be system-dependent constants between the sites of R_i and R_j; C_{ij} is the networkwide unitary transmission cost; D_{ij} corresponds to the fixed cost of initiating transmissions [17]. The transmission cost to send a data from the site of R_i to that of R_j is $C_{ij} \cdot x + D_{ij}$, where x is the size of data. Therefore, the transmission cost of the traditional semijoin $R_i \xrightarrow[tsj]{a} R_j$ is defined as

$$TC(R_i \xrightarrow[tsj]{a} R_j) = C_{ij}|R_i.a|W_a + D_{ij},$$

where $|R_i.a|$ denotes the cardinality of $R_i.a$ and W_a denotes the width of join attribute a [17]. According to this model, the transmission cost of the semijoin with multiple hash functions $R_i \xrightarrow[sjm]{a} R_j$ can be defined as

$$TC(R_i \xrightarrow[sjm]{a} R_j) = C_{ij}(\sum_{i=1}^{m} |B_i|) + D_{ij},$$

where $\sum_{i=1}^{m} |B_i|$ is the total size of the bit-arrays of SJM. The transmission cost of $R_i \xrightarrow[hsj]{a} R_j$ can be defined as

$$TC(R_i \xrightarrow[hsj]{a} R_j) = C_{ij}|B_{ij}| + D_{ij},$$

where $|B_{ij}|$ is the size of the bit array B_{ij}.

2.4 The selectivity model

A *selectivity model* is used to predict the reduction effect of TSJ, SJM, and HSJ. We define that the selectivities ρ_i^{tsj}, ρ_i^{sjm}, and ρ_{ij}^{hsj}, are associated with $R_i \xrightarrow[tsj]{a} R_j$, $R_i \xrightarrow[sjm]{a} R_j$, and $R_i \xrightarrow[hsj]{a} R_j$, respectively. All of these selectivities are real number ranging from 0 to 1. After $R_i \xrightarrow[tsj]{a} R_j$, $R_i \xrightarrow[sjm]{a} R_j$, and $R_i \xrightarrow[hsj]{a} R_j$ are executed, the cardinality of R_j becomes $\rho_i^{tsj}|R_j|$, $\rho_i^{sjm}|R_j|$, and $\rho_{ij}^{hsj}|R_j|$, respectively.

According to [4], the selectivity is equal to the result of

$$\frac{(\textit{the cardinality of } R_i.a)}{(\textit{the size of } U_a)}$$

under the assumption that the values of R_i and R_j are uniformly distributed on the domain U_a of the join attribute a. Since the cardinality of $R_i.a$ is $|R_i.a|$,

$$\rho_i^{tsj} = \frac{|R_i.a|}{|U_a|}.$$

According to [10], the selectivity of $R_i \xrightarrow[sjm]{a} R_j$ is equal to $\frac{|R_i.a|+e'_{|R_i.a|}}{|U_a|}$, where $e'_{|R_i.a|} = (|U_a| - |R_i.a|)(1-e^{-|R_i.a|/b})^m$, where b is the size for every bit-array. (In [10], the size of every bit-array is assumed to be equal.) Therefore,

$$\rho_i^{sjm} = \frac{|R_i.a|}{|U_a|} + (1 - \frac{|R_i.a|}{|U_a|})(1 - e^{-|R_i.a|/b})^m.$$

Similarity, the selectivity of $R_i \xrightarrow[hsj]{a} R_j$ is equal to $\frac{|R_i.a|+e_{|R_i.a|}}{|U_a|}$. Let the m of $e'_{|R_i.a|}$ of SJM be 1. $e_{|R_i.a|}$ is then equal to $(|U_a|-|R_i.a|)(1-e^{-|R_i.a|/|B_{ij}|})$. Therefore,

$$\begin{aligned}\rho_{ij}^{hsj} &= \frac{|R_i.a|}{|U_a|} + (1 - \frac{|R_i.a|}{|U_a|})(1 - e^{-|R_i.a|/|B_{ij}|}), \\ &= 1 - (1 - \frac{|R_i.a|}{|U_a|})e^{-|R_i.a|/|B_{ij}|}.\end{aligned}$$

EXAMPLE 2.1 (The Idea of TSJ, SJM, and HSJ) Given two relations R_1 and R_2 stored at site 1 and site 2 respectively. R_1 represents the name (BOY) and the interest no. (INT#) of boys and R_2 represents the name (GIRL) and the interest no. (INT#) of girls. These two relations are shown in Figure 1. Assume that $C_{12} = 1$ and $D_{12} = 0$, and the domain of INT# in R_1 and R_2, denoted by $U_{INT\#}$, is $[1, 2, 3, \cdots, 10]$. We need 4 bits to represent it. Suppose there is a query $R_1 \overset{INT\#}{\leftrightarrow} R_2$. Consider the following methods for reducing the cardinality of R_2.

R_1

BOY	INT#
Tom	1
Tony	3
John	4
Smith	8

R_2

GIRL	INT#
Lili	0
Nancy	1
Gruen	2
Jam	5
Susi	6
Tracy	8

Figure 1: Relations R_1 and R_2.

Method I: [TSJ] According to the definitions of TSJ, a set of values of INT# (1,3,4,8) is sent to site 2 to join with R_2. The result of $R_1 \frac{INT\#}{tsj} R_2$ is shown in Figure 2. The transmission cost of $R_1 \frac{INT\#}{tsj} R_2$ is

$R_1 \frac{INT\#}{tsj} R_2$

GIRL	INT#
Nancy	1
Tracy	8

Figure 2: Result of TSJ.

$C_{12}|R_1.INT\#|W_a + D_{12}$, i.e., $1 \cdot 4 \cdot 4 + 0 = 16$ (time units). The selectivity of $R_1 \frac{INT\#}{tsj} R_2$ is $\frac{|R_1.INT\#|}{|U_a|} = \frac{4}{10} = 0.4$.

Method II: [SJM] Suppose there are two hash functions, $h_1(x) = (x \bmod 5)$ and $h_2(x) = (10 \cdot \sqrt{x} \bmod 5)$, used in this case (i.e., $m = 2$). $|B_1| = 5$ and $|B_2| = 5$. According to the definitions of SJM, $B_1 = (0,1,0,1,1)$ and $B_2 = (1,0,1,1,0)$, and both of them are sent to site 2. The result of $R_1 \frac{INT\#}{sjm} R_2$ is shown in Figure 3 (which is the same as that of TSJ). The transmission

$R_1 \frac{INT\#}{sjm} R_2$

GIRL	INT#
Nancy	1
Tracy	8

Figure 3: Result of SJM.

cost of $R_1 \frac{INT\#}{sjm} R_2$ is $C_{12}(\sum_{i=1}^{m} |B_i|)+D_{12}$, i.e., $1 \cdot (5+5)+0 = 10$ (time units). The selectivity of $R_1 \frac{INT\#}{sjm} R_2$ is $\frac{|R_1.INT\#|}{|U_a|} + (1 - \frac{|R_1.INT\#|}{|U_a|})(1 - e^{-|R_1.INT\#|/5})^2 \approx 0.4 + 0.6 \cdot 0.3 \approx 0.58$.

Method III: [HSJ] Suppose the hash function used in this case is $h_{12}(x) = (x \bmod 5)$. $|B_{12}| = 5$. According to the definitions of HSJ, $B_{12} = (0,1,0,1,1)$, and it is sent to site 2. The result of $R_1 \frac{INT\#}{hsj} R_2$ is shown in Figure 4. This result is different to that of TSJ. The

$R_1 \frac{INT\#}{hsj} R_2$

GIRL	INT#
Nancy	1
Susi	6
Tracy	8

Figure 4: Result of HSJ.

transmission cost of $R_1 \frac{INT\#}{hsj} R_2$ is $C_{12}|B_{12}| + D_{12}$, i.e., $1 \cdot 5 + 0 = 5$ (time units). The selectivity of $R_1 \frac{INT\#}{hsj} R_2$ is $(1-(1-\frac{|R_1.INT\#|}{|U_a|})e^{-|R_1.INT\#|/|B_{12}|}) \approx (1 - (1 - 0.4) \cdot 0.449) \approx 0.73$. □

3 Comparison of query processing strategies

Consider an TSJ $R_i \xrightarrow[tsj]{a} R_j$. Its transmission cost is $C_{ij}|R_i.a|W_a + D_{ij}$. After this TSJ is executed, the transmission cost from the site of R_j to the final site f is reduced from $C_{jf}|R_j|W_{R_j} + D_{jf}$ to $C_{jf}\rho_i^{tsj}|R_j|W_{R_j} + D_{jf}$, where W_{R_j} is the width of R_j. Therefore, the *benefit* of $R_i \xrightarrow[tsj]{a} R_j$ can be defined as

$$\begin{aligned}&BF(R_i \xrightarrow[tsj]{a} R_j) \\ &= C_{jf}|R_j|W_{R_j} + D_{jf} - C_{jf}\rho_i^{tsj}|R_j|W_{R_j} + D_{jf}, \\ &= (1 - \rho_i^{tsj})|R_j|W_{R_j}C_{jf}.\end{aligned}$$

Consider the total transmission cost, an TSJ $R_i \xrightarrow[tsj]{a} R_j$ is *profitable* [6] iff

$$TC(R_i \xrightarrow[tsj]{a} R_j) < BF(R_i \xrightarrow[tsj]{a} R_j).$$

The benefits of $R_i \xrightarrow[sjm]{a} R_j$ and $R_i \xrightarrow[hsj]{a} R_j$ are defined as

$$BF(R_i \xrightarrow[sjm]{a} R_j) = (1 - \rho_i^{sjm})|R_j|W_{R_j}C_{jf}$$

and

$$BF(R_i \xrightarrow[hsj]{a} R_j) = (1 - \rho_{ij}^{hsj})|R_j|W_{R_j}C_{jf},$$

respectively. An SJM $R_i \xrightarrow[sjm]{a} R_j$ is profitable iff

$$TC(R_i \xrightarrow[sjm]{a} R_j) < BF(R_i \xrightarrow[sjm]{a} R_j);$$

and an HSJ $R_i \xrightarrow[hsj]{a} R_j$ is profitable iff

$$TC(R_i \xrightarrow[hsj]{a} R_j) < BF(R_i \xrightarrow[hsj]{a} R_j).$$

3.1 Comparison of TSJ and HSJ

Assume that, given a query $R_i \overset{a}{\leftrightarrow} R_j$ in a DDBS, the TSJ $R_i \xrightarrow[tsj]{a} R_j$ is not profitable. That is,

$$C_{ij}|R_i.a|W_a + D_{ij} \geq (1 - \rho_i^{tsj})|R_j|W_{R_j}C_{jf},$$

i.e.,

$$|R_i.a|W_a \geq \frac{(1 - \frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}. \quad (1)$$

But suppose that we can find a profitable HSJ $R_i \xrightarrow[hsj]{a} R_j$ (by a "good" hash function). Then,

$$C_{ij}|B_{ij}| + D_{ij} < (1 - \rho_{ij}^{hsj})|R_j|W_{R_j}C_{jf},$$

i.e.,

$$|B_{ij}| < \frac{(1 - \frac{|R_i.a|}{|U_a|})e^{-|R_i.a|/|B_{ij}|}|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}.$$

Since $e^{-|R_i.a|/|B_{ij}|} \geq 0$,

$$|B_{ij}| < \frac{(1 - \frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}e^{-|R_i.a|/|B_{ij}|}.$$

Therefore,

$$|B_{ij}|e^{|R_i.a|/|B_{ij}|} < \frac{(1 - \frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}. \quad (2)$$

Combine (1) and (2), we obtain that

$$\begin{aligned} &|B_{ij}|e^{|R_i.a|/|B_{ij}|} \\ &< \frac{(1 - \frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}} \\ &\leq |R_i.a|W_a. \end{aligned} \quad (3)$$

Intuitively, this inequality is satisfied when W_a is large and $|B_{ij}|$ is suitable. (The optimal $|B_{ij}|$ will be discussed in Section 3.3.) In Example 3.1, we give a case in which TSJ is not profitable but HSJ is, i.e., the inquality (3) is satisfied in this case. Therefore, when TSJ is not profitable, we may use HSJ to reduce the total transmission cost.

EXAMPLE **3.1 (Comparison of TSJ and HSJ)** Given a DDBS. Suppose there is a query created at site F (i.e., the final site) to join with R_A and R_B. R_A and R_B are stored at site A and B respectively. Suppose the join attribute is a. All of the parameters of $R_A \xrightarrow[tsj]{a} R_B$ and $R_A \xrightarrow[hsj]{a} R_B$ are listed as follows.

$C_{AB} = 3$, $D_{AB} = 10$, and $C_{BF} = 3$;
$|R_A.a| = 150$ and $W_a = 12$;
$|R_B| = 150$ and $W_{R_B} = 45$;
$|U_a| = 500$;
$|B_{AB}| = 50$.

Then, by the definitions of transmission cost and benefit,

$$\begin{aligned} TC(R_A \xrightarrow[tsj]{a} R_B) &= C_{AB}|R_A.a|W_a + D_{AB} \\ &= 3 \cdot 150 \cdot 12 + 10 \\ &= 5410 \end{aligned}$$

and

$$\begin{aligned} BF(R_A \xrightarrow[tsj]{a} R_B) &= (1 - \rho_A^{tsj})|R_B|W_{R_B}C_{BF} \\ &= (1 - \tfrac{150}{500}) \cdot 150 \cdot 45 \cdot 1 \\ &= 4725. \end{aligned}$$

$TC(R_A \xrightarrow[tsj]{a} R_B) > BF(R_A \xrightarrow[tsj]{a} R_B)$, $R_i \xrightarrow[tsj]{a} R_j$ is therefore not profitable. On the other hand,

$$\begin{aligned} TC(R_A \xrightarrow[hsj]{a} R_B) &= C_{AB}|B_{AB}| + D_{AB} \\ &= 3 \cdot 50 + 10 = 160 \end{aligned}$$

and

$$\begin{aligned} BF(R_A \xrightarrow[hsj]{a} R_B) &= (1 - \rho_{AB}^{hsj})|R_B|W_{R_B}C_{BF} \\ &= (1 - \tfrac{150}{500}) \cdot e^{\frac{-150}{50}} \cdot 150 \cdot 45 \cdot 1 \\ &= 235.244. \end{aligned}$$

$TC(R_A \xrightarrow[hsj]{a} R_B) < BF(R_A \xrightarrow[hsj]{a} R_B)$, $R_i \xrightarrow[hsj]{a} R_j$ is profitable. In fact,

$$|B_{AB}|e^{|R_A.a|/|B_{AB}|} = 1004.277,$$

$$\frac{(1 - \frac{|R_A.a|}{|U_a|})|R_B|W_{R_B}C_{BF} - D_{AB}}{C_{AB}} = 1571.667,$$

and $|R_A.a|W_a = 1800$,

therefore the inequality (3) is satisfied. TSJ is not profitable but IISJ is in this case. □

3.2 Comparison of SJM and HSJ

Assume that, given a query $R_i \overset{a}{\leftrightarrow} R_j$ in a DDBS, the SJM $R_i \xrightarrow[sjm]{a} R_j$ is not profitable. Therefore, by the definitions,

$$C_{ij}(\sum_{i=1}^{m}|B_i|) + D_{ij} \geq (1-\rho_i^{sjm})|R_j|W_{R_j}C_{jf},$$

i.e.,

$$\sum_{i=1}^{m}|B_i| \geq \frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}, \quad (4)$$

here we assume that ρ_i^{sjm} is equal to $\frac{|R_i.a|}{|U_a|}$. (This is because the collision is not accepted in SJM. The number of hash functions is large such that the selectivity of SJM approximates to $\frac{|R_i.a|}{|U_a|}$.) Furthermore, suppose that we can find a profitable HSJ $R_i \xrightarrow[hsj]{a} R_j$ by a "good" hash function. Then,

$$C_{ij}|B_{ij}| + D_{ij} < (1-\rho_{ij}^{hsj})|R_j|W_{R_j}C_{jf}.$$

According to (2),

$$|B_{ij}|e^{|R_i.a|/|B_{ij}|} < \frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}. \quad (5)$$

Combine (4) and (5), we obtain that

$$\begin{aligned} &|B_{ij}|e^{|R_i.a|/|B_{ij}|} \\ &< \frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}} \\ &\leq \sum_{i=1}^{m}|B_i|. \end{aligned} \quad (6)$$

We give a case in which SJM is not profitable but HSJ is in Example 3.2. According to this example, the inequality (6) is satisfied intuitively when $\sum_{i=1}^{m}|B_i|$ is large and $|B_{ij}|$ is suitable. Therefore, when SJM is not profitable, we may use HSJ to reduce the total transmission cost.

EXAMPLE 3.2 (Comparison of SJM and HSJ) Consider the case of Example 3.1. Suppose we use $R_A \xrightarrow[sjm]{a} R_B$ and $R_A \xrightarrow[hsj]{a} R_B$ to reduce the cardinality of R_B. All of the parameters are equal to that of Example 3.1 but the number of hash functions of $R_A \xrightarrow[sjm]{a} R_B$ is 10 (i.e., $m = 10$) and $|B_i| = 160$ for every $i = 1, 2, 3$, and 4. Then, by the definitions of transmission cost and benefit,

$$\begin{aligned} TC(R_A \xrightarrow[sjm]{a} R_B) &= C_{AB}(\textstyle\sum_{i=1}^{m}|B_i|) + D_{AB} \\ &= 3 \cdot (10 \cdot 160) + 10 \\ &= 4810 \end{aligned}$$

and

$$\begin{aligned} BF(R_A \xrightarrow[sjm]{a} R_B) &= (1-\rho_A^{sjm})|R_B|W_{R_B}C_{BF} \\ &\leq (1-\tfrac{150}{500})150 \cdot 45 \cdot 1 \\ &\leq 4725. \end{aligned}$$

$TC(R_A \xrightarrow[sjm]{a} R_B) > BF(R_A \xrightarrow[sjm]{a} R_B)$, $R_i \xrightarrow[sjm]{a} R_j$ is therefore not profitable. On the other hand, since $TC(R_A \xrightarrow[hsj]{a} R_B) = 160$ and $BF(R_A \xrightarrow[hsj]{a} R_B) = 235.244$, $TC(R_A \xrightarrow[hsj]{a} R_B) < BF(R_A \xrightarrow[hsj]{a} R_B)$, $R_i \xrightarrow[hsj]{a} R_j$ is profitable. In fact,

$$|B_{AB}|e^{|R_A.a|/|B_{AB}|} = 1004.277,$$

$$\frac{(1-\frac{|R_A.a|}{|U_a|})|R_B|W_{R_B}C_{BF} - D_{AB}}{C_{AB}} = 1571.667,$$

and $\sum_{i=1}^{m}|B_i| = 1600$,

therefore the inequality (6) is satisfied. SJM is not profitable but HSJ is in this case. □

3.3 Optimal hash function in HSJ

An *optimal hash function* of an HSJ is defined to be a hash function that makes the HSJ profitable and the *profit* of the HSJ maximum in all hash functions, where profit is defined to be $BF(R_i \xrightarrow[hsj]{a} R_j) - TC(R_i \xrightarrow[hsj]{a} R_j)$. According to (3) and (6), an HSJ $R_i \xrightarrow[hsj]{a} R_j$ is profitable iff

$$|B_{ij}|e^{|R_i.a|/|B_{ij}|} < \frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}.$$

Since $1 \leq e^{|R_i.a|/|B_{ij}|}$,

$$|B_{ij}| \leq |B_{ij}|e^{|R_i.a|/|B_{ij}|}.$$

Therefore, if $R_i \xrightarrow[hsj]{a} R_j$ is profitable,

$$|B_{ij}| \in [2, \lfloor\frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}\rfloor].$$

($|B_{ij}| \geq 2$ is trivial.) Therefore, the *optimal* $|B_{ij}|$, denoted by $OP_{tt}(|B_{ij}|)$, which makes the profit maximum, can be found in the range $[2, \lfloor\frac{(1-\frac{|R_i.a|}{|U_a|})|R_j|W_{R_j}C_{jf} - D_{ij}}{C_{ij}}\rfloor]$.

Let the set of hash functions $h(x) = (x \bmod y)$ for any y satisfy the assumptions of [10], i.e., they are *uniform hash function*. The optimal hash function of $R_i \xrightarrow[hsj]{a} R_j$ is the function $h(x) = (x \bmod OP_{tt}(|B_{ij}|))$ where $OP_{tt}(|B_{ij}|)$ is the optimal $|B_{ij}|$. In Example 3.1 and Example 3.2, the optimal B_{AB} can be found in [2,1571]. Checking this range, we obtain that $OP_{tt}(|B_{AB}|)$ is 254 and the maximum profit is 3108.706.

4 Conclusions and future work

In this paper, we proposed a new technique, named hash-semijoin, to improve the query processing in DDBS. Since there is only one hash function h_{ij} and one bit-array B_{ij} used in each hash-semijoin, and the hash function may be any function, hash-semijoins have the following advantages:

1. Since there is only one hash function (or one bit-array) used in each hash-semijoin, its local processing time is better than that of semijoin with multiple hash functions [10,13] and its transmission cost may be better than that of the traditional semijoin.

2. Since the selectivity and transmission cost of a hash-semijoin are affected by the choice of its hash function and there exist many such hash functions, hash-semijoin is more flexible than tradition semijoin.

3. The communication cost in some systems, such as R^*, can be neglected because of the advancement of high-speed network techniques. The I/O cost of the second storage seems to be the new bottleneck. Since the bit-arrays of hash-semijoins may be every small, they can be transmitted and stored in the main memory, and waiting to be joined with the local relation.

Suppose each site has a limited main memory space. To determine the optimal hashing sizes such that 1) they satisfy the buffer size constraint and 2) they provide the maximum reduction effect is a challenging task which needs further investigations.

References

[1] P. M. G. Apers, A. R. Hevner, and S. B. Yao. Optimal algorithms for distributed queries. *IEEE Trans. on Software Engineering*, SE-9(1):57-68, Jan. 1983.

[2] P. A. Bernstein and D. M. Chiu. Using semi-joins to solve relation queries. *JACM*, 28(1):25-40, Jan. 1981.

[3] P. A. Bernstein and N. Goodman. The power of natural semijoins. *SIAM J. Comput.*, 10(4):751-771, Nov. 1981.

[4] P. A. Bernstein et al. Query processing in a system for distributed databases (SDD-1). *ACM Trans. on Database Systems*, 6(4):602-625, Dec. 1981.

[5] P. A. Black and W. S. Luk. A new hueristic for generating semi-join programs for distributed query processing. *In Proc. IEEE COMPSAC*, 581-588, Dec. 1982.

[6] S. Ceri and G. Pelagatti. *Distributed Databases: Principles and Systems.* McGraw-Hill, 1984.

[7] A. L. P. Chen and V. O. K. Li. Improvement algorithms for semi-join query processing programs in distributed databases systems. *IEEE Trans. on Computers*, C-33(11):959-967, November 1984.

[8] A. L. P. Chen and V. O. K. Li. An optimal algorithm for distributed star queries. *IEEE Trans. on Software Engineering*, 11(10):1097-1107, October 1985.

[9] A. L. P. Chen, D. Brill, M. Templeton, and C. T. Yu. Distributed query processing in a multiple database system. *IEEE Journal on Selected Areas in Communications, special issue on Databases in Communications Systems*, April 1989.

[10] E. Babb. Implementing a relation database by means of specialized hardware. *ACM Trans. on Database Systems*, 4(1):1-29, March, 1979.

[11] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, 1991.

[12] W. Perrizo and C. Chen. Composite semijoins in distributed query processing. *Information Sciences*, March 1990.

[13] P. Valduriez. Semi-join algorithms for distributed database machines. *In Distributed Data Bases*, J. -J. Schneider (ed.), Amsterdam: North-Holland, 23-37, 1982.

[14] P. Valduriez and G. Gardarin. Join and semijoin algorithms for a multiprocessor database machine. *ACM Trans. on Database Systems*, 9(1):133-161, March, 1984.

[15] C. P. Wang and V. O. K. Li. The relation-partitioning approach to distributed query processing. In *Proc. 2nd IEEE Data Engineering Conf.*, Feburary 1986.

[16] C. P. Wang, V. O. K. Li, and A. L. P. Chen. On-shot semi-join execution strategies for processing distributed queries. In *Proc. 7th IEEE Data Engineering Conf.*, April 1991.

[17] C. P. Wang, A. L. P. Chen, and S. C. Shyu. A parallel execution method for minimizing distributed query response time. *IEEE Trans. on Parallel and Distributed Systems*, (to appear).

[18] R. Williams et al. R^*: An overview of the architecture. In *Proc. 2nd Int. Conf. on Databases*, Jerusalem, Israel, June 1982.

[19] C. T. Yu et al. Query processing in a fragmented relational distributed system: Mermaid. *IEEE Trans. on Software Engineering*, Auguest 1985.

[20] C. T. Yu, Z. M. Ozsoyoglu, and K. Kam. Optimization of distributed tree queries. *J. Comput. Syst. Sci.*, 29(3):409-445, December 1984.