Query Processing in a Mobile Computing Environment: Exploiting the Features of Asymmetry

Wen-Chih Peng, *Member*, *IEEE*, and Ming-Syan Chen, *Fellow*, *IEEE*

Abstract—With the cutting edge technology advance in wireless and mobile computers, the query processing in a mobile environment involves join processing among different sites which include static servers and mobile computers. Because of the need for energy saving and also the presence of asymmetric features in a mobile computing environment, the conventional query processing for a distributed database cannot be directly applied to a mobile computing system. In this paper, we first explore three asymmetric features of a mobile environment. Then, in light of these features, we devise query processing methods for both join and query processing. Intuitively, employing semijoin operations in a mobile computing environment is able to further reduce both the amount of data transmission and energy consumption. A semijoin which is initiated by a mobile computer (respectively, the server) and is beneficial to reduce the cost of a join operation is termed a *mobile-initiated* or *MI* (respectively, *server-initiated* or *SI*) *profitable* semijoin. According to those asymmetric features of a mobile computing system, we examine three different join methods and devise some specific criteria to identify MI/SI profitable semijoins. For query processing, which refers to the processing of multijoin queries, we develop three query processing schemes. In particular, we formulate the query processing in a mobile computing system as a two-phase query processing procedure that can determine a join sequence and interleave that join sequence with SI profitable semijoins to reduce both the amount of data transmission and energy consumption. Performance of these join and query methods is comparatively analyzed and sensitivity analysis on several parameters is conducted. Furthermore, we develop a systematic procedure to derive the characteristic functions of MI and SI profitable semijoins. It is noted that, given some system parameters, those characteristic functions are very important in determining which join method is the most appropriate one to employ in that configuration. It is shown by our simulation results that, by exploiting the three asymmetric features, these characteristic functions are very powerful in reducing both the amounts of energy consumption and data transmission incurred and can lead to the design of an efficient and effective query processing procedure for a mobile computing environment.

Index Terms—Mobile database, mobile computing, query processing, join method.

1 Introduction

In a mobile computing environment, a mobile user with a power-limited palm computer (or a mobile computer) can access various information via wireless communication. Applications such as stock activities, traffic reports, and weather forecasts have become increasingly popular in recent years [26], [27]. It is noted that mobile computers use small batteries for their operations without directly connecting to any power source and the bandwidth of wireless communication is, in general, limited. As a result, an important design issue in a mobile system is to conserve the energy and communication bandwidth of a mobile unit while allowing mobile users the ability to access information from anywhere at anytime [3], [10], [18].

Various wireless data networking technologies, including IS-136 [24], CDMA2000 [19], and Wireless Application Protocol (WAP) [28], have been developed recently. Among others, the development of the third generation mobile phone provides advanced value-added servers to mobile users [2]. With the rapid advances in the palm computer technologies, a mobile computer is envisioned to be equipped with more powerful capabilities, including the storage of a small database and the capacity of data processing [22]. Consequently, the query processing in a mobile computing system may involve the server and several mobile computers. Consider a sales and inventory application where a salesperson carries a mobile computer device in which a fragment of database contains the information of his/her customer records. Also, the company of the salesperson has a fixed server to store the information of all customers and its product information. Note that, depending on the corresponding coherency control mechanism employed, the data copy in the server could be obsolete [7]. Since the most up-to-date data is stored in the mobile computers, a query generated by a salesperson could be a sequence of joins to be performed across the relations residing in the server and several mobile computers, resulting in a very different execution scenario from the one for query processing in a traditional distributed system.

The query processing in a traditional distributed system has received a considerable amount of attention and been extensively studied in the literature [4], [5], [6], [12], [25]. As

- *W.-C. Peng is with the Department of Computer Science and Information Engineering, National Chiao Tung University, No. 1001, Ta Hseueh Rd., Hsinchu, Taiwan 300, ROC. E-mail: wcpeng@csie.nctu.edu.tw.*
- *M.-S. Chen is with the Department of Electrical Engineering and Graduate Engineering, National Taiwan University, No. 1, Sec. 4, Roosevlet Rd., Taipei, Taiwan 106, ROC. E-mail: mschen@cc.ee.ntu.edu.tw.*

Manuscript received 05 Sept. 2003; revised 03 July 2004; accepted 24 Jan. 2005; published online 18 May 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0174-0903.

pointed out in [31], the query processing in a traditional distributed system is composed of the following three phases: 1) local processing phase, 2) reduction phase, and 3) final processing phase. The objective in distributed query processing is to reduce the amount of data transmission required in Phase 2 and Phase 3. Note, however, that the cost models developed for query processing in a traditional distributed database do not reflect many important features in a mobile computing system. Explicitly, the prior studies in distributed query processing [5], [6], [12], [25] did not fully explore the asymmetric features of a mobile environment, which are, as explained below, particularly important in devising the corresponding query processing schemes. Specifically, the energy consumption of mobile computers, the most important cost criterion, was not dealt with for the query processing in traditional distributed databases, making the corresponding distributed query processing schemes devised not applicable to a mobile computing environment. To remedy this, we shall explore in this paper three important asymmetric features of a mobile computing system and, in light of these features, develop efficient join methods and query processing schemes for mobile computing systems. The three asymmetric features, which we shall explicitly address and reflect in the design of query processing schemes, are as follows:

1. **Asymmetric feature of computing capability between the server and a mobile computer.** Mobile computers use small batteries for their operations without directly connecting to any power source. In contrast, the server is not strictly constrained by energy consumption and thus possesses much more power than a mobile computer. Note that, in traditional distributed query processing, the sites involved in a query processing are usually assumed to have the same level of processing capability. This feature distinguishes the query processing in a mobile environment from that in a traditional distributed system.
2. **Asymmetric feature of energy consumption between message sending and receiving.** The energy required for message sending is more than that required for message receiving at a mobile computer [9], [17]. This feature also has to be modeled when the costs of the corresponding operations are evaluated.
3. **Asymmetric feature of energy consumption between activeness and idleness of a mobile computer.** The energy consumed by a mobile computer in its active mode is much more than that consumed in its idle mode [16], [17]. In view of this, a mobile computer may be designed to stay in its idling mode by migrating its processing work to the server if so appropriate.

Consequently, in this paper, we derive a cost model which considers these three asymmetric features of a mobile computing system. The cost model derived paves the way to the development of the join methods and the query processing schemes in a mobile computing system. In order to further reduce data transmission, the semijoin operation has received considerable attention and been extensively studied in the literature [5], [6], [12], [25]. Consider an illustrative example in which R_1 stored at site S_1 performs a semijoin join operation with R_2 stored at site S_2. Suppose that the final join result will be obtained at site S_1. Under semijoin processing, S_1 will first send the distinct join attribute values of R_1 to S_2 and then the join processing of R_2 with these distinct join attribute values of R_1 is performed at site S_2. The resulting relation after the join (referred to as R_2') will be sent to site S_1 for further join processing at site S_1. Clearly, if the amount of distinct join attribute tuples and data tuples in R_2' at S_2 is smaller than that of the original relation R_2, the amount of data transmission under the semijoin operation is further reduced. Join and semijoin will be discussed in details later. Intuitively, employing semijoin operations in a mobile computing environment is also able to further reduce both the amount of data transmission and energy consumption. In mobile computing environments, however, sites S_1 and S_2 may be either servers or mobile computers. Thus, the semijoin operation in a mobile computing system is intrinsically different from that in traditional distributed systems. For ease of exposition, a semijoin which is initiated by a mobile computer (respectively, the server) and is beneficial to reduce the cost of a join operation is termed a *mobile-initiated* or *MI* (respectively, *server-initiated* or *SI*) *profitable* semijoin. For join processing, judiciously applying a MI profitable semijoin can reduce the amount of data transmission required and energy consumption. According to those asymmetric features of a mobile computing system, we examine three different join methods and devise some specific criteria to identify MI/SI profitable semijoins. For query processing, which refers to the processing of multijoin queries, we develop three query processing schemes. In particular, we formulate the query processing in a mobile computing system as a two-phase query processing procedure that can determine a join sequence and interleave that join sequence with SI profitable semijoins to reduce both the amounts of data transmission and energy consumption. Performance of these join and query methods is comparatively analyzed and sensitivity analysis on several parameters, including selectivity factor and idling coefficient, is conducted. Furthermore, we develop a systematic procedure to derive the characteristic functions of MI and SI profitable semijoins. It is noted that, given some system parameters, those characteristic functions are very important in determining which join method is the most appropriate one to employ in that configuration. It is shown by our simulation results that by exploiting the three asymmetric features, these characteristic functions are very powerful in reducing both the amounts of energy consumption and data transmission incurred, and can lead to the design of an efficient and effective query processing procedure for a mobile computing environment.

We mention in passing that, without dealing with query processing, Alonso and Ganguly [1] studied the issues of optimization between energy consumption and server workload in a mobile environment. Notice that the mobile computers participating in distributed query processing may spread out in mobile computing environments. By exploiting the feature of divide-and-conquer, the authors in [20] proposed several query processing schemes that are

able to divide the query processing into several subquery processing modules in accordance with the network topology of a mobile computing system. Several research efforts have been elaborated upon developing a location dependent query mechanism [11], [14], [23], [29]. The authors in [14] presented the concept of queries with location constraints, i.e., constraints which involve location of mobile users. In [23], the authors proposed a spatial-temporal data model for querying of moving data in mobile environments. The position update policy and the impression of moving data are addressed in [29]. In addition, the authors in [15] proposed a multicast protocol which is useful for query processing in a three dimensional space. Without exploiting the asymmetric features of energy consumption, the attention of prior studies was mainly paid to the query mechanisms with location constraints and query processing in traditional distributed databases [4], [5], [12], [13], [25], but not to the specific cost model and the query processing for a mobile computing system explored in this paper. Note that energy efficient query mechanisms are recently proposed for the sensor networks in which sensor nodes are small devices with wireless capabilities, poor processing power and small data storage [21], [30]. Though having the goal for conserving energy consumption, sensor nodes don't have enough storage space for storing data and manipulating complex query processing. Therefore, energy efficient query processing schemes for sensor networks are not applicable to the scenario in which mobile computers have to store data and perform complicated join operations.

As mentioned above, due to these asymmetric features of a mobile computing system, the cost model and the design of query processing schemes are intrinsically different from those in a traditional distributed database and particularly ought to capture the need of energy saving. In this paper, we not only formulate a new cost model which takes these asymmetric features into consideration, but also explicitly investigate the join methods and develop the corresponding query processing schemes. To the best of our knowledge, prior work neither fully explored the need of energy saving for query processing nor captured these asymmetric features into the corresponding cost model, let alone devising efficient schemes to incorporate these features for query processing in a mobile computing system. These features distinguish this paper from others.

This rest of this paper is organized as follows: Preliminaries are given in Section 2. Three join methods are investigated in Section 3. In Section 4, we develop query processing schemes for multijoin queries. Performance studies on various join and query processing are conducted in Section 5. This paper concludes with Section 6.

2 Preliminaries

To facilitate the presentation of this paper, some preliminaries are given in this section. The notation, definitions, and assumptions required are described in Section 2.1. By taking the asymmetric features described above into consideration, a cost model for join and query processing in a mobile computing system is devised in Section 2.2.

2.1 Notation, Definition, and Assumption

A mobile computing system consists of stationary servers and mobile computers. Stationary servers include information servers and the equipment of a mobile communication system. As described above, mobile computers use small batteries for their operations without connecting to any power source directly and the bandwidth of wireless communication is limited. With the emerging development of the third generation of mobile system [19], [24], [28], mobile computers can provide many powerful capabilities, including the storage of a small database and the capability of data processing. In this paper, we consider the query processing in a mobile environment, which involves join processing among different sites including static servers and mobile computers.

As in most prior work [5], [6], we assume in this study that a query is of the form of conjunctions of equi-join predicates and all attributes are renamed in such a way that two join attributes have the same attribute name if and only if they have a join predicate between them. A join query graph can be denoted by a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node in a join query graph represents a relation. Two nodes are connected by an edge if there exists a join predicate on some attribute of the two corresponding relations. We use $|R_i|$ to denote the cardinality of a relation R_i and $|A|$ to denote the cardinality of the domain of an attribute A. The notation $R_i \bowtie R_j$ is used to mean the join between R_i and R_j, and $|R_i \bowtie R_j|$ denotes cardinality of the result relation of $R_i \bowtie R_j$. To determine the effect of a join operation specified by a query graph, we employ the results stated in the theorem developed in [5], which is given in Appendix A for interested readers.

Let w_A be the width of an attribute A and w_{R_i} be the width of a tuple in R_i. The size of the total amount of data in R_i can then be denoted by $w_{R_i}\,|R_i|$. Define the selectivity $\rho_{i,a}$ of attribute A in R_i as $\frac{|R_i(A)|}{|A|}$, where $R_i(\mathrm{A})$ is the set of distinct values for the attribute A in R_i. $R_i - A \rightarrow R_j$ means a semijoin from R_i to R_j on attribute A and $R_i \rightarrow R_j$ means a simple join from R_i to R_j. The reduction of a relation is defined as the number of tuples of the relation reduced by employed join operations. Similarly, the reduction of an attribute is referred to as the number of tuples reduced by employed join operations on that attribute. Note that the reduction of the relation R_j by the semijoin $R_i - A \rightarrow R_j$ is proportional to the reduction of $R_j(\mathrm{A})$. The estimation of the size of the relation reduced by a semijoin is thus similar to estimating the reduction of projection on the semijoin attributes. After the semijoin $R_i - \mathrm{A} \rightarrow R_j$, the cardinality of R_j can be estimated as $|R_j|\rho_{i,a}$. Also, $w_A|R_i(A)|$ is used to denote the cost of a semijoin $R_i - A \rightarrow R_j$. We use R_i' to denote the resulting relation after some joins or semijoins are applied to an original relation R_i. As in most prior work [25], [31], it is assumed that the statistics in each site are made known to a central scheduler for query scheduling.

Consider the relations in Table 1, for example. Suppose $|\mathrm{A}| = 10$, $|\mathrm{B}| = 10$, and the width of each attribute is one unit. We have $\rho_{1,b} = 0.3$ and $\rho_{2,b} = 0.6$. Also, $|\mathrm{R}_1| = 5$, $|\mathrm{R}_2| = 7$, and $\mathrm{R}_1(\mathrm{B}) = \{\mathrm{b}_1, \mathrm{b}_3, \mathrm{b}_4\}$.

TABLE 1
An Illustrative Example for Semijoin Operation

R_1	
A	B
a_1	b_1
a_2	b_1
a_2	b_3
a_2	b_4
a_3	b_3

R_2	
B	C
b_1	c_1
b_1	c_2
b_2	c_1
b_5	c_2
b_6	c_4
b_7	c_2
b_8	c_3

R_2'	
B	C
b_1	c_1
b_1	c_2

A semijoin $R_1 - B \rightarrow R_2$ is called *profitable* if its cost, $w_B|R_1(B)| = w_B|B|\rho_{1,b}$, is less than its benefit, $w_{R_2}|R_2| - w_{R_2}|R_2|\rho_{1,b} = w_{R_2}|R_2|(1-\rho_{1,b})$ [6], [25], where $w_{R_2}|R_2|$ and $w_{R_2}|R_2|\rho_{1,b}$ are respectively the sizes of R_2 before and after the semijoin. It can be shown that, for the example relations in Table 1, $R_1 - B \rightarrow R_2$ is profitable since $w_B|R_1(B)| < w_{R_2}|R_2|(1-\rho_{1,b})$ and $R_2 - B \rightarrow R_1$ is not profitable since $w_B|R_2(B)| \geq w_{R_1}|R_1|(1-\rho_{2,b})$. R_2' shows the reduced relation after the semijoin.

2.2 Cost Model for Join and Query Processing in a Mobile Computing System

Table 2 shows the description of parameters in the cost model for a mobile computing system. Energy consumption in data receiving at a mobile computer, denoted by e_{RC}, refers to the energy consumed in receiving data via wireless communication. Energy consumption for receiving a relation R at a mobile computer, denoted by $\mathrm{e}_{RC}(\mathrm{R})$, is formulated as $e_r * |R|$, where e_r is the receiving energy coefficient, representing the energy consumed in receiving one tuple of data.[1] Also, energy consumption for sending a relation R out from a mobile computer, denoted by $\mathrm{e}_{SD}(R)$, refers to the energy required in transmitting the relation R over wireless link. To capture the asymmetric feature of energy consumption between data sending and receiving of a mobile computer, we use the send-receive energy ratio r_E (i.e., $\frac{e_{SD}}{e_{RC}}$) to represent the ratio of the energy consumption of sending data to that of receiving data. The value of r_E is, in general, larger than one and can more explicitly be approximated to a value between two and 10 [1]. Hence, $e_{SD}(R)$ is formulated as $r_E * e_r * |R|$, where $e_r * r_E$ is the sending energy coefficient. Similarly to most relevant works [8], [16], the processing time required to perform the given operations on the relation in a server is modeled as a function of the input relations involved. For example, the processing time of joining R_i and R_j in a server, denoted by $T_S(R_i \bowtie R_j)$, can then be expressed by $t_{tuple} * (|R_i| + |R_j| + |R_i \bowtie R_j|)$, where t_{tuple} is the coefficient for the processing time required per tuple.

To capture the asymmetric feature of computing capability between the server and a mobile computer, the server-mobile processing ratio r_{SM} represents the ratio of the processing power of a server to that of a mobile computer. Clearly, the value of r_{SM} is larger than 1 and can be obtained empirically. Hence, the processing time of joining R_i and R_j at a mobile computer can be expressed by $r_{SM} * T_S(R_i \bowtie R_j)$. The energy consumption at a mobile computer in its idle mode while a join is performed in a server can be estimated as $\delta * T_S(R_i \bowtie R_j)$, where δ is an idle coefficient. The idle coefficient δ of mobile computers can be approximated to a value between 0.02 and 0.5 and is, in fact, system dependent [16], [17]. To reflect the asymmetric feature of energy consumption between activeness and idleness of a mobile computer, we define the ratio of the energy consumed by a mobile computer in its idle mode to that in its active mode as δ^k, where δ is an idling coefficient and k is able to empirically determined. Without loss of generality, we assume, in this paper, that k is 2. Similarly, an active coefficient of a mobile computer represents the the energy consumption at a mobile computer in its active mode per unit time. In accordance with the ratio of activeness and idleness of a mobile computer, it can be verified that given $k = 2$, we can have $\frac{1}{\delta}$ to represent the active coefficient to estimate the energy consumed by a mobile computer in its active mode. As mentioned before, the processing time of joining R_i and R_j at a mobile computer can be formulated as $r_{SM} * T_S(R_i \bowtie R_j)$. As a result, the amount of energy consumed in processing the join between R_i and R_j at a mobile computer is devised as $\frac{1}{\delta} * r_{SM} * T_S(R_i \bowtie R_j)$. Note that all these parameters can be estimated from the specifications of mobile computers [2], [22]. Using the cost model developed, we are able to evaluate the energy consumption and data transmission costs (E_c and d_t, respectively) of the corresponding join methods and query processing schemes in a mobile computing system and develop an efficient solution procedure for multijoin query processing accordingly.

3 Join Processing in a Mobile Computing System

We now derive the solution procedure for minimizing the cost of join methods in a mobile computing system. Consider the scenario of join processing in Fig. 1, where the server has relation R_1 and the mobile computer has relation R_2. Suppose that the mobile user M_1 submits to server S a query that performs a join operation of R_1 and R_2 on their common attribute A. The resulting relation is needed by M_1. With this given model, we shall examine three join methods. To simplify our presentation, $\mathrm{d}_t(J)$ is used to represent the amount of data transmission and $\mathrm{E}_c(J)$ is used to represent the amount of energy consumption by join method J. In what follows, we examine a join method which performs the join at the server in Section 3.1. Section 3.2 describes the join method which performs the join at the mobile computer. The join method that utilizes an MI profitable semijoin will be presented in Section 3.3. Analysis of these join methods is given in Section 3.4.

3.1 Processing the Join at the Server (Denoted by $\mathbf{J}_S$)

We first consider the case of processing the join at the server and returning the result to mobile unit M_1.

1. e_r is calculated by $\mathrm{e} * \mathrm{w}_R$, where e is the energy consumption per byte and w_R is the width of tuple in relation R. For brevity purpose, we assume that each relation has the same w_R whose value is set to one and the pricing police is "charging per byte."

TABLE 2
Description of Symbols for the Cost Model in a Mobile Computing System

Description	Symbol
Processing ratio of server to mobile, i.e., server/mobile	r_{SM}
Energy consumption ratio of data sending to data receiving	r_E
Idling coefficient for a mobile computer	δ
Energy consumption in data receiving at a mobile computer	e_{RC}
Energy consumption in data sending at a mobile computer	e_{SD}
Processing time function at the server	T_S
Amount of data transmission for query processing	d_t
Amount of energy consumption for query processing	E_c

Explicitly, M_1 sends R_2 to the server, which incurs a data amount of $|R_2|$ transmitted and an energy amount of $e_{SD}(R_2)$ consumed. While the server S performs the join operation, M_1 is in its idle mode with an energy amount of $\delta * T_S(R_1 \bowtie R_2)$ consumed. After the join operation is performed at the server, the resulting relation (i.e., $R_1 \bowtie R_2$) will be returned to M_1 while consuming an energy amount of $e_{RC}(|R_1 \bowtie R_2|)$. Then, we have the corresponding costs as follows:

$$\begin{aligned} &d_t(J_S) : |R_2| + |R_1 \bowtie R_2|, \\ &E_c(J_S) : e_{SD}(|R_2|) + \delta * T_S(R_1 \bowtie R_2) + e_{RC}(|R_1 \bowtie R_2|). \end{aligned} \quad (1)$$

3.2 Processing the Join at the Mobile Computer (Denoted by J_C)

We next consider the case that the server sends the relation to the mobile computer for a join operation. The server S sends R_1 to M_1 first. After receiving R_1 with an energy amount of $e_{RC}(|R_1|)$ consumed, M_1 performs the join operation with an energy amount of $\frac{1}{\delta} * r_{SM} * T_S(R_1 \bowtie R_2)$. Consequently, we have corresponding costs below:

$$\begin{aligned} &d_t(J_C) : \ |R_1|, \\ &E_c(J_C) : \ e_{RC}(|R_1|) + \frac{1}{\delta} * r_{SM} * T_S(R_1 \bowtie R_2). \end{aligned} \quad (2)$$

3.3 Employing an MI Profitable Semijoin for Join Processing (Denoted by J_{SC})

Note that the relation size at the mobile computer, i.e., $|R_2|$, is likely to be smaller than that at the server, i.e., $|R_1|$. In that case, using a semijoin will be able to reduce the amount of data transmission required. Explicitly, M_1 first projects its semijoin attribute (e.g., attribute A with the corresponding selectivity $\rho_{2,a}$) with an energy amount of $\frac{r_{SM} * T_S(\rho_{2,a}|A|)}{\delta}$, and the resulting projection of the size $\rho_{2,a}|A|$ is sent to the server for a join with R_1. The energy consumed in this operation is $E_c = e_{SD}(\rho_{2,a}|A|)$, and the amount of data transmitted is $d_t = \rho_{2,a}|A|$. Then, the energy consumption of M_1 in its idle mode is $\delta * T_S((\rho_{2,a}|A|) \bowtie R_1)$ while server S performs the join operation. The resulting relation (i.e., $\rho_{2,a}|R_1|$) is next sent back to the mobile unit (i.e., $d_t = \rho_{2,a}|R_1|$ and $E_c = e_{RC}(\rho_{2,a}|R_1|)$) for the join operation at M_1. The amount of energy consumed in that join operation is thus $r_{SM} * T_S((\rho_{2,a}|R_1|) \bowtie R_2)$. Consequently, we have the following costs:

$$\begin{aligned} &d_t(J_{SC}) : \rho_{2,a}|A| + \rho_{2,a}|R_1|, \\ &E_c(J_{SC}) : e_{SD}(\rho_{2,a}|A|) + \frac{1}{\delta} * r_{SM} * T_S(\rho_{2,a}|A|) + \frac{1}{\delta} * r_{SM} * \\ &T_S(\rho_{2,a}|R_1| \bowtie R_2) + \delta * T_S(\rho_{2,a}|A| \bowtie R_1) + e_{RC}(\rho_{2,a}|R_1|). \end{aligned} \quad (3)$$

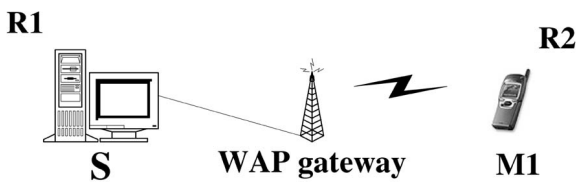

Fig. 1. A query scenario of join processing.

3.4 Analysis of Join Processing

With the above three join methods described, we now examine the amount of data transmission and energy consumption incurred by each of them. Specifically, the criterion of identifying an MI profitable semijoin to reduce the amount of data transmission and energy consumption is derived. For better readability, proofs of some lemmas and corollaries are given in the Appendix B for interested readers. With the assumption that the number of data tuples is larger than the cardinality of attributes, we have the following lemma:

Lemma 1. *The amount of data transmission incurred by join method J_C is smaller than that by join method J_S, i.e., $d_t(J_C) < d_t(J_S)$.*

Lemma 2. *With a given selectivity factor $\rho_{2,a}$, where $\rho_{2,a} < \frac{|R_1|}{|A|+|R_1|}$, the amount of data transmission incurred by method J_{SC} is smaller than that by method J_C, i.e., $d_t(J_{SC}) < d_t(J_C)$.*

Note that the mobile computer performs the join in method J_C. In contrast, the server takes over the join processing in method J_S. The asymmetric feature of computing capability between the server and a mobile computer is reflected by Lemma 3 which can be derived from the corresponding formulas of $E_c(J_C)$ and $E_c(J_S)$.

Lemma 3. $E_c(J_C) > E_c(J_S)$.

Definition 1. *A semijoin is called* MI profitable *if and only if $E_c(J_{SC})$ is smaller than $E_c(J_S)$.*

With Definition 1, we can derive following theorem:

Theorem 1. *A semijoin is MI profitable if and only if the selectivity factor $\rho_{2,a}$ is smaller than $\frac{e_r * \delta}{r_{SM} * t_{tuple}}$.*

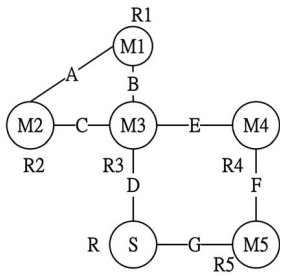

Fig. 2. An illustrative query graph.

Proof. It follows from Section 3.3 that the amount of energy consumed by method J_{SC} is

$$\mathrm{e}_{SD}(\rho_{2,a}|A|) + \frac{1}{\delta} * r_{SM} * T_S(\rho_{2,a}|A|) + \frac{1}{\delta} * r_{SM} * T_S(\rho_{2,a}|R_1| \bowtie R_2) + \delta * T_S(\rho_{2,a}|A| \bowtie R_1) + \mathrm{e}_{RC}(\rho_{2,a}|R_1|).$$

From the cost model developed in Section 2.2, it can be obtained that

$$E_c(J_{SC}) = e_r * r_E * \rho_{2,a}|A| + \frac{1}{\delta} * r_{SM} * t_{tuple} * \left(\rho_{2,a}|A| + \rho_{2,a}|R_1| + |R_2| + \frac{\rho_{2,a}|R_1||R_2|}{|A|}\right) + \delta * t * (\rho_{2,a}|A| + |R_1| + \rho_{2,a}|R_1|) + e_r * \rho_{2,a}|R_1|.$$

Without loss of generality, assume that $\frac{|R_1||R_2|}{|A|} >> |A|$, $|R_1|$, and $|R_2|$. Consequently, we have $E_c(J_{SC}) \approx \frac{1}{\delta} * r_{SM} * t_{tuple} * \left(\frac{\rho_{2,a}|R_1||R_2|}{|A|}\right)$. Following the same procedure, we have $E_c(J_S) \approx e_r * \left(\frac{|R_1||R_2|}{|A|}\right)$. From Definition 1, we have $E_c(J_{SC}) < E_c(J_S)$. Accordingly,

$$E_c(J_{SC}) - E_c(J_S) < 0$$
$$\iff \frac{1}{\delta} * r_{SM} * t_{tuple} * \left(\frac{\rho_{2,a}|R_1||R_2|}{|A|}\right) - e_r * \left(\frac{|R_1||R_2|}{|A|}\right) < 0$$
$$\iff \frac{|R_1||R_2|}{|A|}\left(\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho_{2,a} - e_r\right) < 0.$$

Since $|\mathrm{R}_1|$, $|\mathrm{R}_2|$, and $|A| > 0$, we can obtain that $(\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho_{2,a} - e_r) < 0$, which implies that $\rho_{2,a} < \frac{e_r * \delta}{r_{SM} * t_{tuple}}$, thus proving the theorem. □

Theorem 1 leads to the following corollary:

Corollary 1.1. *Given a $\rho_{2,a}$, a semijoin is* MI profitable *if and only if δ is larger than $\rho_{2,a} * r_{SM} * \frac{t_{tuple}}{e_r}$.*

Using the cost model devised in Section 2.2, one can evaluate the amounts of energy consumption and data transmission incurred by join methods described above and select the one which is able to minimize the cost of join processing. Note that, through having the minimal amount of energy consumed due to its exploiting the asymmetric feature of computing capability, method J_S may incur a larger amount of data transmission than others. It can be verified that by judiciously applying MI profitable semijoins, method J_{SC} can reduce the amount of data transmission and energy consumption as a whole. As can be seen later, Theorem 1 and Corollary 1.1 derived above can be employed to determine the threshold for whether method J_S or method J_{SC} should be utilized.

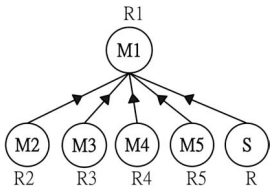

Fig. 3. An illustrative example for scheme QP_C.

4 Query Processing in a Mobile Computing System

In this section, we consider the processing of multijoin queries that involves one server and many mobile computers. The *destination mobile computer* refers to the mobile computer that issues the query and is expected to receive the query result. A *participating mobile computer* refers to the mobile computer that contains a relation involved in the query processing. By utilizing the join methods derived in Section 3, we develop efficient schemes for query processing. Explicitly, we first examine in Section 4.1 a query processing scheme in which the destination mobile computer performs the query processing by itself (to be referred to as scheme QP_C). In light of the asymmetric feature of computing capability between the server and a mobile computer, we formulate the query processing as a two-phase processing procedure that can minimize the energy consumption of the destination and participating mobile computers. Specifically, a two-phase query processing scheme which employs a simple join operation (to be referred to as scheme QP_S) is devised in Section 4.2. Next, we devise another two-phase query processing scheme that can determine a join sequence with SI profitable semijoins interleaved (to be referred to as scheme QP_{SJ}) to reduce both the amounts of data transmission and energy consumption in Section 4.3.

4.1 Query Processing at the Destination Mobile Computer (Denoted by QP_C)

In scheme QP_C, while the destination computer M_1 submits a query to the server, the server then informs those participating mobile computers about the location of M_1. After receiving the location of M_1, all participating mobile computers and the server send their relations to the destination mobile computer for query processing. Consider the illustrative query graph in Fig. 2. Without loss of generality, assume that the participating mobile computer M_i contains relation R_i, the server S has relation R, and M_1 is the destination mobile computer.

Fig. 3 shows the corresponding join sequence graph of scheme QP_C, where each relation in a node is sent to M_1 for a join operation. According to the cost model devised, we can obtain the corresponding costs of energy consumption and data transmission. Clearly, through minimizing the energy consumption of participating mobile computers, scheme QP_C results in a significant amount of energy consumption at the destination mobile computer (i.e., receiving relations from participating mobile computers and performing join operations). As a consequence, we have,

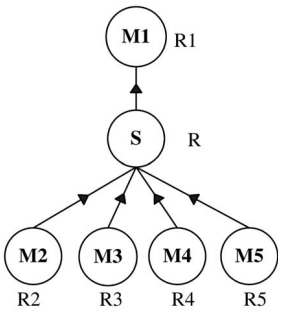

Fig. 4. An illustrative example for scheme QP_S.

$$\begin{aligned}
\mathrm{d}_t(QP_C) &= \sum_{2}^{k} |R_i| + |R|, \\
&\quad \text{where } k = 5 \text{ for the example in Fig. 2,} \\
\mathrm{E}_c(QP_C) &= \sum_{2}^{k} e_{SD}(R_i) + \sum_{2}^{k} e_{RC}(R_i) \\
&\quad + \frac{1}{\delta} * r_{SM} * T_S(R \bowtie (\sum_{1}^{k} \bowtie R_i)), \\
&\quad \text{where } k = 5 \text{ for the example in Fig. 2.}
\end{aligned} \tag{4}$$

4.2 Query Processing at the Server (Denoted by QP_S)

Clearly, despite its simplicity, scheme QP_C does not exploit the asymmetric feature of computing capability between the server and mobile computers and may thus consume much valuable processing power at the destination mobile computer. Explicitly, since the server is not strictly constrained by energy consumption, one can fully utilize the processing capability of the server. In view of this, we decompose the processing of a query into two phases, namely, the *relation transfer phase*, denoted by RT, and *the final phase*, denoted by FP. In the relation transfer phase, each participating mobile computer sends its own relation to the server for a join operation. Thus, the server obtains the resulting relation of the multijoin query among the participating mobile computers. In the *final phase*, with the join method properly selected from those devised in Section 3, the join between S and M_1 is performed.

The cost of scheme QP_S is the summation of the cost in the RT phase and that in the FP phase. That is, $\mathrm{d}_t(QP_S) = d_t(QP_S^{RT}) + d_t(QP_S^{FP})$, where QP_S^{RT} and QP_S^{FP} represent the scheme QP_S in the RT phase and the FP phase, respectively, and $\mathrm{E}_c(QP_S) = E_c(QP_S^{RT}) + E_c(QP_S^{FP})$. Note that the values of $d_t(QP_S^{FP})$ and $E_c(QP_S^{FP})$ can be determined in accordance with the join method employed in the FP phase. With the query graph in Fig. 2, the corresponding join sequence of scheme QP_S is shown in Fig. 4, where each relation in a node is sent to its parent node for a join operation in a bottom up manner.

Note that, since the server in QP_S takes over the query processing which costs much energy of the destination mobile computer in QP_C, the energy consumption of the destination mobile computer in QP_S is significantly reduced. As will be validated by the experimental results in Section 5, by exploiting the asymmetric feature of computing capability between the server and mobile computers, scheme QP_S can save the energy consumption of all the mobile computers.

With the query in Fig. 2, we have following costs in the RT phase:

$$\begin{aligned}
&\mathrm{d}_t(QP_S^{RT}) = \sum_{2}^{k} |R_i|, \text{where } k = 5 \text{ for the example in Fig.2,} \\
&\mathrm{E}_c(QP_S^{RT}) = \sum_{2}^{k} e_{SD}(R_i), \text{where } k = 5 \text{ for the example in Fig.2.}
\end{aligned} \tag{5}$$

It can be seen that, in scheme QP_S, each participating mobile computer sends its own relation to the server without considering the use of semijoins. As can be seen in [5], judiciously interleaving a join sequence with semijoins is able to reduce the amount of data transmission required. Note, however, that without considering the asymmetric features and energy consumption, the algorithm in [5] is not applicable to the query processing in a mobile computing system. As a result, we will devise in the following subsection scheme QP_{SJ} to determine a proper join sequence with semijoins interleaved in the RT phase for further reducing both the amounts of data transmission and energy consumption.

4.3 Query Processing with SI Profitable Semijoins (Denoted by QP_{SJ})

As described in Section 3.4, an MI profitable semijoin is helpful in reducing both energy consumption and data transmission required. However, due to the asymmetric features of a mobile computing system, MI profitable semijoins cannot be directly applied to the query processing in the RT phase of QP_S. In view of this, we shall first derive a theorem to identify SI profitable semijoins and, then, in light of the theorem derived, develop a solution procedure that can interleave a join sequence with SI profitable semijoins for efficient query processing.

Definition 2. *A semijoin $R_i - A \rightarrow R_j$, where the server owns relation R_i and the mobile computer owns relation R_j, is called* SI profitable *if its energy consumption of the mobile computer for performing this semijoin, i.e., $e_{RC}(\rho_{i,a} * |A|) + \frac{1}{\delta} * r_{SM} * T_S(R_j \bowtie (\rho_{i,a} * |A|)) + e_{SD}(\rho_{i,a}|R_j|)$, is smaller than that for sending R_j to the server, i.e., $\mathrm{e}_{SD}(|R_j|)$. Note that energy consumption of an SI semijoin by the mobile computer consists of the energy consumed in performing the semijoin and in sending the resulting relation of that semijoin to the server for a join operation.*

With Definition 2, we can derive the following theorem:

Theorem 2. *A semijoin $R_i - A \rightarrow R_j$ is* SI profitable *if and only if $\rho_{i,a}$ is less than $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})}$.*

Proof. It follows from Definition 2 that $e_{RC}(\rho_{i,a} * |A|) + \frac{1}{\delta} * r_{SM} * T_S(R_j \bowtie (\rho_{i,a} * |A|)) + e_{SD}(\rho_{i,a}|R_j|) < e_{SD}(|R_j|)$. Accordingly, we have

$$\begin{aligned}
&e_r * \rho_{i,a} * |A| + \frac{1}{\delta} * r_{SM} * t_{tuple} * (|R_j| + \rho_{i,a} * |A| + \rho_{i,a} * |R_j|)) \\
&+ e_r * r_E * \rho_{i,a}|R_j| < e_r * r_E * |R_j|.
\end{aligned}$$

Since $\rho_{i,a} * |A| << |R_j|$, we have

$$\frac{1}{\delta} * r_{SM} * t_{tuple} * |R_j|(1 + \rho_{i,a}) + e_r * r_E * \rho_{i,a}|R_j| < e_r * r_E * |R_j|.$$

Since $|\mathrm{R}_j| > 0$, we have

$$\frac{1}{\delta} * r_{SM} * t_{tuple} + \rho_{i,a} * (\frac{1}{\delta} * r_{SM} * t_{tuple} + e_r * r_E) < e_r * r_E.$$

$\Leftrightarrow \rho_{i,a} < \frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})}$, thus proving the theorem. □

Theorem 2 leads to the following corollary:

Corollary 2.1. *An* SI profitable *semijoin, $R_i - A \rightarrow R_j$, implies that the amount of data transmission by performing this semijoin (i.e., $\rho_{i,a}(|A| + |R_j|)$) is smaller than that of sending R_j to the server (i.e., $|R_j|$).*

In the RT phase, by utilizing the server to perform the multijoin query, the server site contains the resulting relation. To facilitate our presentation, use $\mathrm{E}_c^{d,j}$ to represent the energy consumption of a join operation between R_d and R_j, where d is the destination site. The energy consumption of an SI profitable semijoin, $R_d - A \rightarrow R_j$, can be expressed as $E_c^{d,j}(SJ)$ and the corresponding data transmission cost can be denoted by $d_t^{d,j}(SJ)$. The energy consumption of a join $R_j \rightarrow R_d$ can be expressed as $E_c^{d,j}(J)$ and the corresponding data transmission cost can be denoted by $d_t^{d,j}(J)$. Consider the query graph in Fig. 2, where the server S has two edges (i.e., $(\mathrm{S}, \mathrm{M}_3)$ and $(\mathrm{S}, \mathrm{M}_5)$), meaning that there are two possible joins, i.e., $R \bowtie R_3$ or $R \bowtie R_5$, for S to start with. Note that, when a join operation between R and R_3 (or R_5) is carried out, the resulting relation in site S will contain those attributes of relation R_3 (or R_5). In other words, the selection of these two join operations (i.e., $(\mathrm{R} \bowtie \mathrm{R}_3)$ and $(\mathrm{R} \bowtie \mathrm{R}_5)$) will affect the costs of subsequent operations. Thus, with the inclusion of SI profitable semijoins into a proper join sequence, one can reduce the amount of data transmission and energy consumption.

As described above, without considering the feature of asymmetry, the algorithm for determining a join sequence for a conventional distributed database is not applicable to a mobile computing environment. Therefore, the design of a solution procedure to determine a join sequence with SI profitable semijoins interleaved is call for. First, we develop a directed graph with proper weights in edges where weights in edges represent the energy consumption of performing join or SI-semijoin operations at mobile computers. Assume that the destination mobile computer in the query graph (V, E) is denoted by d and the set of edges of node d is represented as E_d. The destination mobile computer and its edges are omitted in the directed graph since they will not be involved in the RT phase. The resulting directed graph is thus $(\mathrm{V} - \mathrm{d}, \mathrm{E} - \mathrm{E}_d)$. An edge connecting two nodes n_i and n_j is denoted by (n_i, n_j), and the weight of edge (n_i, n_j), denoted by $\mathrm{E}_c^{i,j}$, can be set to either the value of $E_c^{i,j}(SJ)$ (meaning that an SI profitable semijoin SJ will be performed) if $\rho_i < \frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})}$, or the value of $E_c^{i,j}(J)$ (meaning that a simple join J will be performed) otherwise. After constructing the directed graph, the problem to reduce the energy consumption is transformed to a graph program in which we shall determine a traversal path with the purpose of minimizing the summation of weights (i.e., minimize the energy consumption consumed in mobile computers). By referring to the Djikstra algorithm, algorithm M is designed and applied to determine a join sequence with SI profitable semijoins interleaved in the RT phase. Note that an edge (S, n_j) in a directed graph is being *shrunken* if (S, n_j) is removed from the graph and S and n_j are merged together. When a join operation between the two relations corresponding to nodes S and n_j in a given directed graph is carried out, we can obtain the resulting query graph by shrinking the edges between S and n_j. Algorithm M is outlined below.

Algorithm M. Determine the join sequence with SI profitable semijoins interleaved in the RT phase.
Input: A directed graph = $(V - d, E - E_d)$.
Output: SEQ /* SEQ contains the resulting sequence of joins semijoins */

```
1. begin
2.   SEQ = φ;
3.   for each vertex w ∈ V − d  do
4.     begin
5.       w.mark := false; /* w.mark is used to indicate if
         w has been visited or not */
6.       w.ct := ∞; /* w.ct is the cost of join operation
         from S to w. */
7.       w.op = J; /* w.op represents the join operation */
8.     end
9.   S.ct = 0; /* The cost of join operation from S to itself is
     set to zero */
10.  while ∃ an unmarked vertex do
11.   begin
12.      let w be an unmarked vertex such that w.ct is the
         minimal among all the corresponding costs of
         unmarked vertices;
13.      if (w.ρ < (r_E*e_r − (1/δ)*r_SM*t_tuple) / (r_E*e_r + (1/δ)*r_SM*t_tuple)) /* Determine if a
         semijoin is SI profitable */
14.          w.op = SJ;
15.      SEQ=SEQ ∪ w;
16.      w.mark := true;
17.      for all edges (w, z) with z is unmarked do
18.        begin
19.        if w.ct+weight(w, z) < z.ct then
20.          z.ct := w.ct+weight(w, z); /* Update the
             weight of the edge (w,z) */
21.        end
22.   end
23. end
```

To show the execution of algorithm M, consider the query graph in Fig. 2 whose profile is given in Table 3. The corresponding directed graph is shown in Fig. 5. It can be verified that, since $\rho_{R,G} = 0.5 < \frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} = 0.6$, the weight of edge from S to M_5 in Fig. 5 is

$$E_c^{S,5}(SJ) = 0.1 * 0.5 * 18 + \frac{1}{0.5} * 5 * 0.01 * (120 + 0.5 * 18 + 0.5 * 120) + 0.1 * 5 * 0.5 * 120 = 49.8.$$

TABLE 3
A Profile for a Query Where $|A| = 19, |B| = 15, |C| = 17, |D| = 19, |E| = 16, |F| = 15, |G| = 18, r_{SM} = 5, \delta = 0.5, e_r = 0.1, r_E = 5,$ and $t_{tuple} = 0.01$

Relation R_i	$\|R_i\|$	Attribute X	Selectivity $\rho_{i,x}$
R_1	107	A	0.8
		B	0.75
R_2	102	A	0.85
		C	0.75
R_3	106	C	0.8
		D	0.7
		E	0.4
R_4	100	E	0.8
		F	0.95
R_5	120	F	0.8
		G	0.85
R	131	D	0.9
		G	0.5

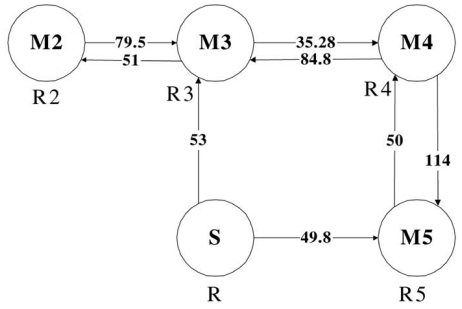

Fig. 5. A directed graph for a query graph in Fig. 2.

Also, since $\rho_{R,D} = 0.9 > \frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} = 0.6$, the weight of edge from S to M_3 is $E_c^{S,3}(J) = 0.1 * 5 * 106 = 53$. Similarly, the weights of other edges in the directed graph in Fig. 5 can be obtained. Then, algorithm M is used to generate the proper join sequence with SI profitable semijoins interleaved in the RT phase. First, the costs from S to other vertices are evaluated in the directed graph for initialization (from line 3 to line 8 in algorithm M). The execution scenario of algorithm M is shown in Fig. 6 where R^*, R^{**}, and R^{***} denote the resulting relations in the end of each step. Note that, since the cost of (S, M_5) is the minimal (line 12 in algorithm M), as can be seen in Step 0 from Table 4, the vertex M_5 is selected first. Also, it can be verified that the semijoin operation $R - G \rightarrow R_5$ is SI profitable (line 13 in algorithm M) since the corresponding selectivity factor (i.e., 0.5) is smaller than $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} = 0.6$. The semijoin $R - G \rightarrow R_5$ is thus performed. In Fig. 6a, R_5 is selected for the semijoin and sends the resulting relation to the server. After the join with the relation at site M_5, the cost from S to M_4 is derived (from line 17 to line 21 in algorithm M). It can be seen that Fig. 6a leads to the configuration in Fig. 6b in which the weight of the edge from S to M_4 becomes 99.8 (i.e., $49.8 + 50 = 99.8$). Then, we shall determine the minimal cost among all the edges connecting to those unvisited vertices (line 12 in algorithm M) and perform the corresponding operations. This procedure repeats until all the vertices of the directed graph are visited (line 10 to in algorithm M). From Table 4, it can be seen that M_3 has the minimal cost in Step 1 and is next to be selected. Since the semijoin $R^* - D \rightarrow R_3$ is not SI profitable according to Theorem 2, the simple join from M_3 to S is executed in Fig. 6b, leading to the configuration in Fig. 6c. Following the same procedure, the sequence of joins and semijoins can be derived. Note that, while having the same cost as QP_S in the FP phase, scheme

TABLE 4
An Execution Example of Algorithm M

Steps	M_2	M_3	M_4	M_5	Operation included into SEQ
0	∞	53	∞	49.8	ϕ
1	∞	53	99.8	49.8	$R-G \rightarrow R_5$, $R_5' \rightarrow R$
2	104	53	88.28	49.8	$R_3 \rightarrow R^*$
3	104	53	88.28	49.8	$R^{**}-E \rightarrow R_4$, $R_4' \rightarrow R^{**}$
4	104	53	88.28	49.8	$R_2 \rightarrow R^{***}$

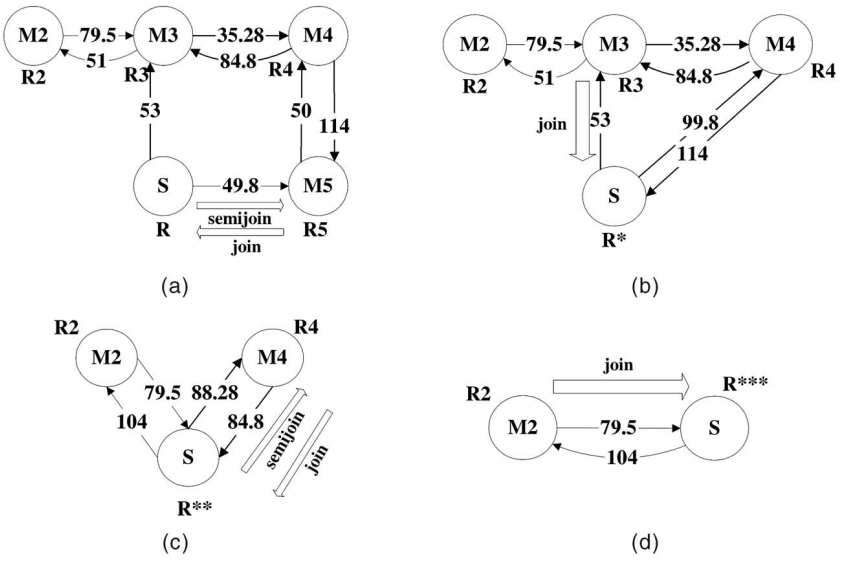

Fig. 6. Execution scenario of algorithm M. (a) Step 1. (b) Step 2. (c) Step 3. (d) Step 4.

TABLE 5
The Parameters Used in the Simulation

Notation	Definition	Values
n	Number of relations involved in query processing	4, 5 , 6, 7
δ	Idling coefficient	0.1
t_{tuple}	Coefficient for the processing time required per tuple.	0.01
r_E	Send-receive energy ratio	5
r_{SM}	Ratio of the processing power of a server to that of a mobile computer.	various value used
e_r	Receiving energy coefficient	various value used

QP_{SJ} outperforms QP_S in the RT phase by incurring not only a smaller amount of energy consumption, i.e., $49.8 + 53 + 35.28 + 51 = 189.08 < 60 + 53 + 50 + 51 = 214$, but also a smaller amount of data transmission, i.e., $69 + 106 + 46.4 + 102 = 323.4 < 120 + 106 + 100 + 102 = 428$, showing the very advantage of employing proper SI profitable semijoins in scheme QP_{SJ}.

5 Performance Evaluation

Extensive performance studies are conducted in this section. The simulation model built to evaluate the join and query processing in a mobile computing system is described in Section 5.1. In Section 5.2, performance of join methods, including J_S, J_C, and J_{SC}, is empirically studied. Sensitivity analysis on various parameters, including selectivity factor and idling coefficient, are conducted. The corresponding characteristic functions of MI profitable semijoins, which are important in determining whether J_S or J_{SC} should be used, are derived. Experimental results of query processing schemes, including those of QP_C, QP_S, and QP_{SJ}, are presented in Section 5.3. A characteristic function of SI profitable semijoins is developed and is shown to be of important use for efficient execution of QP_{SJ}.

5.1 System Model

Simulations were performed to evaluate the effectiveness of join processing methods and query processing schemes. The simulation program was coded in C++, and input queries were generated as follows. The number of relations in a query was predetermined. The occurrence of an edge between two relations in the query graph was determined according to a given probability. Without loss of generality, only queries with connected query graphs were deemed valid and used for our study. Based on the above, the cardinalities of relations and attributes were randomly generated from a uniform distribution within some reasonable ranges. These settings are similar to those prior works in query processing [6], [8]. For ease of exposition, unless mentioned otherwise, the default value of each parameter is given in Table 5. The number of relations involved in query processing, denoted by n, is chosen to be 4, 5, 6, 7 and 8, respectively. In general, we set the value of δ to 0.1, the value of t_{tuple} to 0.01, and the value of r_E to 5.

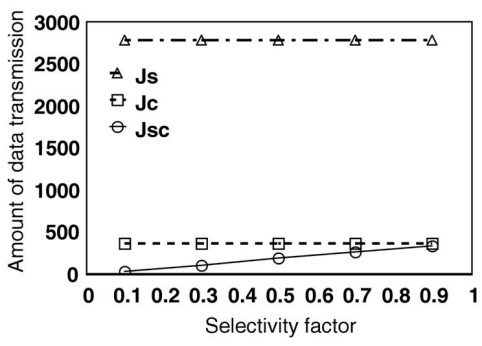

Fig. 7. The amounts of data transmission by J_S, J_C, and J_{SC} with the selectivity factor varied.

5.2 Experimental Results of Join Processing

The effectiveness of join processing is evaluated in this section. We first examine the impact of selectivity factor to both the amounts of data transmission and energy consumption of J_S, J_C, and J_{SC} in Section 5.2.1. The derivation of characteristic function of an MI profitable semijoin for selectivity factor with r_{SM} varied, denoted by $f_{MI}(\cdot)$, is presented in Section 5.2.2. Then, the impact of varying the value of an idling coefficient to the energy consumption of J_S, J_C and J_{SC} is evaluated in Section 5.2.3. The derivation of characteristic function of MI profitable semijoin for idling coefficient with r_{SM} varied, denoted by $g_{MI}(\cdot)$, is given in Section 5.2.4. Given related parameters, the characteristic functions $f_{MI}(\cdot)$ and $g_{MI}(\cdot)$ can be used to determine whether join method J_S or J_{SC} should be utilized.[2]

5.2.1 Experiments of Selectivity Factor

As mentioned before, the selectivity $\rho_{i,a}$ of attribute A in R_i is defined as $\frac{|R_i(A)|}{|A|}$, where $R_i(A)$ is the set of distinct values for the attribute A in R_i. To conduct the experiments to evaluate the impact of selectivity factor, we set the value of r_{SM} to 5, the value of δ to 0.1, the value of t_{tuple} to 0.01, the value of e_r to 0.1, and the value of r_E to 5. The amount of data transmission and the corresponding amount of energy consumed by J_S, J_C and J_{SC} are examined with the selectivity factor varied. Fig. 7 shows the amounts of data transmission incurred by J_S, J_C, and J_{SC}. It can be seen from Fig. 7 that the amount of data transmission incurred by J_S has the largest value among all methods, agreeing with Lemma 1 and Lemma 2. Note that the amounts of data transmission incurred by J_S and J_C remain constant for

2. Note that, since the spirit of our work is mainly to propose a systematic procedure to derive the characteristic functions of an MI profitable semijoin for selectivity factor and idling coefficient (that can be employed by MI profitable semijoin for performance improvement), this procedure is still applicable even in the presence of various parameter values in different applications.

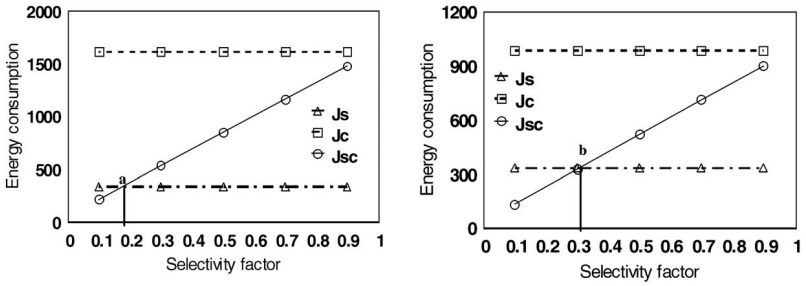

Fig. 8. The amounts of energy consumption incurred by J_C, J_S, and J_{SC} with the selectivity factor varied. (a) The energy consumption of J_C, J_S, and J_{SC} with $r_{SM} = 5$ and the selectivity factor varied. (b) The energy consumption of J_C, J_S, and J_{SC} with $r_{SM} = 3$ and the selectivity factor varied.

different values of the selectivity factor. In contrast, the amount of data transmission incurred by J_{SC} tends to increase when the selectivity factor increases. In other words, the join method J_{SC} is favorable when the value of the selectivity factor is small.

The amounts of energy consumption incurred by J_S, J_C, and J_{SC} with the selectivity factor varied are shown in Fig. 8a, where it can be seen that the amount of energy consumption incurred by J_C is larger than that by J_S, agreeing with Lemma 3. Since the energy consumptions of J_S and J_C are independent of the selectivity factor, both the amounts of energy consumption for J_S and J_C remain constant when the value of the selectivity factor varies. Note that as the selectivity factor is smaller than that of the intersection point of curve J_S and curve J_{SC} in Fig. 8a (i.e., point a where the selectivity factor $= 0.18$), the energy consumption of J_{SC} is smaller than that of J_S. Clearly, the selectivity factor of point a can be used as a threshold to determine whether J_S or J_{SC} should be employed. It is important to note that the selectivity factor of point a can be estimated as $\frac{e_r * \delta}{r_{SM} * t_{tuple}} = \frac{0.1*0.1}{5*0.01} = 0.2$ according to Theorem 1, which is very close to the value of 0.18 that is empirically determined from Fig. 8a.

Fig. 8b shows the amounts of energy consumed by J_S, J_C, and J_{SC} with $r_{SM} = 3$ and the selectivity factor varied. Note that, compared with the selectivity factor in Fig. 8a where $r_{SM} = 5$, the decisive selectivity factor point (i.e., point b where the selectivity factor $= 0.3$) shifts to the right in Fig. 8b. According to Theorem 1, the selectivity factor of point b can be estimated as $\frac{e_r * \delta}{r_{SM} * t_{tuple}} = \frac{0.1*0.1}{3*0.01} = 0.33$, which is also very close to the value of 0.3 empirically determined from Fig. 8b.

5.2.2 Characteristic Function of MI Profitable Semijoin for Selectivity Factor

To derive a characteristic function of an MI profitable semijoin for the selectivity factor with r_{SM} varied, we conduct 10 experiments, which are analogous to the ones in Section 5.2.1, and combine their results to construct Fig. 9. Specifically, without loss of generality, we set the values of r_{SM} to be in the range from 1 to 10 and empirically obtain the threshold points for the execution of J_{SC}. A complete spectrum for the impact of r_{SM} to the value of the decisive selectivity factor is shown as a function $f_{MI}(\cdot)$ in Fig. 9, where the threshold selectivity factors collected from experimental results are shown by line 1 and those determined from the analytical model stated in Theorem 1 are shown by line 2. Explicitly, the values of line 1 are determined from such points as the intersection point of curve J_S and curve J_{SC} in Fig. 8a (i.e., point a) and that in Fig. 8b (i.e., point b). It can be seen that the value of the selectivity factor varies from 0.3 in Fig. 8b (with $r_{SM} = 3$) to 0.18 in Fig. 8a (with $r_{SM} = 5$).

Note that the characteristic function $f_{MI}(\cdot)$ is very important in determining whether J_S or J_{SC} should be used. Note that the left-lower half of Fig. 9 corresponds to the operating region where method J_{SC} is favored and the right-upper of Fig. 9 corresponds to the region where method J_S should be used. Specifically, given a selectivity factor being $s1$ and $r_{SM} = r1$, if $f_{MI}(r1) > s1$, join method J_{SC} should be utilized. Otherwise, join method J_S is used. It can be seen that the difference between experimental results in line 1 and analytic results in line 2 is almost negligible, showing the good accuracy of the experimental studies conducted.

5.2.3 Experiments of Idling Coefficient

With a given selectivity factor, the sensitivity of varying the value of an idling coefficient (i.e., δ) for join processing is investigated in this subsection. Specifically, with a given selectivity factor $\rho = 0.6$, we set the value of the r_{SM} to 5, the value of e_r to 0.08, the value of t_{tuple} to 0.01, and the

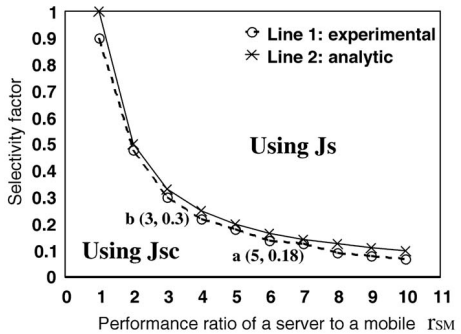

Fig. 9. The characteristic function of an MI profitable semijoin, $f_{MI}(\cdot)$, for the selectivity factor with r_{SM} varied.

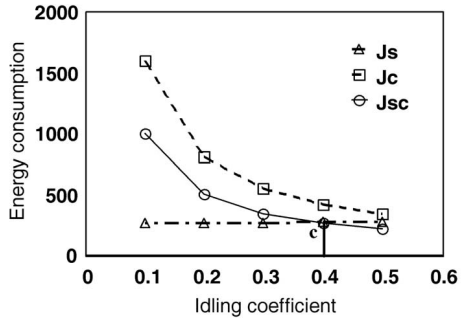

Fig. 10. The energy consumptions of J_S, J_C, and J_{SC} with δ varied.

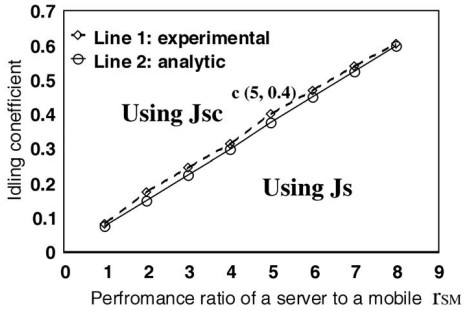

Fig. 11. The characteristic function of an MI profitable semijoin, $g_{MI}(\cdot)$, for δ with r_{SM} varied.

value of r_E to 5. With this setting, the experimental results are shown in Fig. 10, where the amounts of energy consumptions of J_S, J_C, and J_{SC} with the value of the idling coefficient varied are given.

As can be seen from Fig. 10, the amount of energy consumed by J_C is larger than that by J_S. Both the amounts of energy consumed by J_C and J_{SC} tend to decrease as the value of an idling coefficient increases. Note that, as the value of the idling coefficient is larger than the one corresponding to the intersection point of curve J_S and curve J_{SC} in Fig. 10 (i.e., point c where the value of the idling coefficient is 0.4), the amount of energy consumed by J_{SC} is smaller than that by J_S. Clearly, with a given selectivity factor, the idling coefficient can be used as a decisive parameter for the execution of J_{SC}. It is important to note that, according to Corollary 1.1, the corresponding value at point c can also be estimated as $\rho * r_{SM} * \frac{t_{tuple}}{e_r} = 0.6 * 5 * \frac{0.01}{0.08} = 0.375$, which is very close to the value of 0.4 that is empirically determined from Fig. 10.

5.2.4 Characteristic Function of MI Profitable Semijoin for Idling Coefficient

Similarly to the procedure in Section 5.2.2, we conduct eight experiments and combine their results to derive a characteristic function of an MI profitable semijoin, denoted by $g_{MI}(\cdot)$, for the idling coefficient with r_{SM} varied. Specifically, we set the values of r_{SM} to be in the range from 1 to 8 and empirically determine the corresponding points for the execution of J_{SC}. A complete spectrum for the impact of r_{SM} to the value of the decisive idling coefficient is shown as a function $g_{MI}(\cdot)$ in Fig. 11, where the decisive idling coefficients collected from experimental results are shown by line 1 and those determined from the analytical model stated in Corollary 1.1 are shown by line 2. It can be verified that the values of line 1 are determined from such threshold points as the one which is intersected by curve J_S and curve J_{SC} in Fig. 10 (i.e., point c in Fig. 10). Note that, similar to $f_{MI}(\cdot)$, $g_{MI}(\cdot)$ can also be employed to determine whether J_{SC} or J_S should be used. The left-upper half of Fig. 11 corresponds to the operating region where method J_{SC} is used and the right-lower of Fig. 11 corresponds to the region where method J_S is used. Specifically, given an idling coefficient being $i1$ and $R_{SM} = r1$, if $g_{MI}(r1) < i1$, join method J_{SC} should be utilized. Otherwise, join method J_S is used. Same as in Fig. 9, it can be seen that the difference between experimental and analytic results is negligible.

5.3 Experimental Results of Query Processing

In this section, we first evaluate the performance of an SI profitable semijoin in Section 5.3.1. Then, the characteristic function of an SI profitable semijoin for selectivity factor with R_{SM} varied, denoted by $f_{SI}(\cdot)$, is derived in Section 5.3.2. Performance studies of QP_S and QP_{SJ} are conducted in Section 5.3.3.

5.3.1 Experiments of SI Profitable Semijoin

In this experiment, we set the value of r_{SM} to 5, the value of δ to be 0.5, the value of t_{tuple} to 0.01, the value of e_r to 0.1, and the value of r_E to 5. The amounts of data transmissions and energy consumptions of an SI (server-initiated) semijoin and a simple join are examined with the selectivity factor varied. According to Definition 2, an SI semijoin is called profitable if the amount of energy consumed is reduced by including this semijoin. From Fig. 12a, it can be

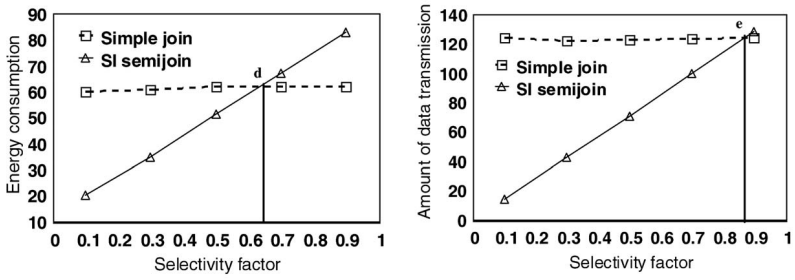

Fig. 12. The performance of SI semijoins and simple joins with $r_{SM} = 5$. (a) The energy consumption of SI semijoin and simple join. (b) The amount of data transmission of SI semijoin and simple join.

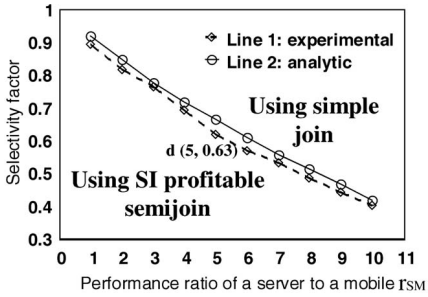

Fig. 13. The characteristic function of an SI profitable, $f_{SI}(\cdot)$, with r_{SM} varied.

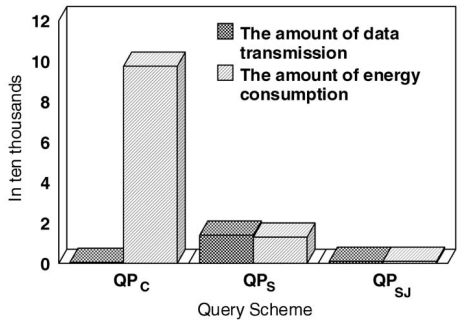

Fig. 14. The performance of QP_C, QP_S, and QP_{SC} when the number of relations is 5.

seen that the amount of energy consumed by an SI profitable semijoin is smaller than that by a simple join when the value of the selectivity factor is smaller than the corresponding value at point d (i.e., 0.63). Thus, the corresponding value at point d can be used as a threshold to identify SI profitable semijoins. Note that according to Theorem 2, the corresponding selectivity factor at point d can be estimated as $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} = \frac{5*0.1-\frac{1}{0.5}*5*0.01}{5*0.1+\frac{1}{0.5}*5*0.01} = 0.667$, which is very close to the value of 0.63 at point d that is empirically determined from Fig. 12a. It is shown in Fig. 12b the amount of data transmission incurred by an SI profitable semijoin is much smaller than that by a simple join, and the corresponding selectivity factor at point e is larger than that at point d, agreeing with Corollary 2.1.

5.3.2 Characteristic Function of SI Profitable Semijoin

To derive a characteristic function of an SI profitable semijoin for the selectivity factor with r_{SM} varied, we conduct 10 experiments, similar to the one in Section 5.3.1, and combine their results to obtain Fig. 13, where a characteristic function of an SI profitable semijoin with the value of r_{SM} varied, i.e., $f_{SI}(\cdot)$, is given. In Fig. 13, the decisive selectivity factors determined from experimental results are shown by line 1 and those determined from the analytical model stated in Theorem 2 are shown by line 2. Again, the values of line 1 are collected from such intersection points as the one which is intersected by curve SI semijoin and curve simple join in Fig. 12a (i.e., point d). Note that the left-lower half of Fig. 13 corresponds to the operating region where SI profitable semijoins are used, and the right-upper of Fig. 13 corresponds to the region where simple joins are used.

5.3.3 Performance of QP_S and QP_{SJ}

As shown in Section 4.3, interleaving an appropriate join sequence with SI profitable semijoins in the RT is able to reduce the amounts of data transmission and energy consumption. We now examine the performance of query processing among one server and many mobile computers. The number of relations in a query is set to 5 and 300 queries are random generated. For each query, the three query schemes, i.e., QP_C, QP_S, and QP_{SJ} are performed. Without loss of generality, we set the value of R_{SM} to 5, the value of δ to 0.5, the value of e_r to 0.1, and the value of r_E to 5. Fig. 14 shows the amounts of data transmission and energy consumption incurred by QP_C, QP_S, and QP_{SJ}.

From Fig. 14, it can be seen that scheme QP_C incurs the largest amount of energy consumption among all schemes. Also, as validated by the experimental results, by exploiting the asymmetric feature of computing capability between the server and mobile computers, scheme QP_S can save the energy consumption of the destination mobile computer and participating mobile computers. Furthermore, through reducing the amount of energy consumption, scheme QP_S increases the amount of data transmission required. Note that, by employing the characteristic function in Fig. 13 to guide its execution, scheme QP_{SJ} can further reduce both the amounts of data transmission of energy consumption, showing the very advantage of interleaving a join sequence with SI profitable semijoins in the RT phase. To show the difference of QP_S and QP_{SJ} for different numbers of relations, both the amounts of data transmission and energy consumption incurred by methods QP_S and QP_{SJ} in the RT phase are shown in Fig. 15. It is noted that, owing to the advantage of employing SI profitable semijoins, QP_{SJ} significantly outperforms QP_S by incurring smaller amounts of data transmission and energy consumption.

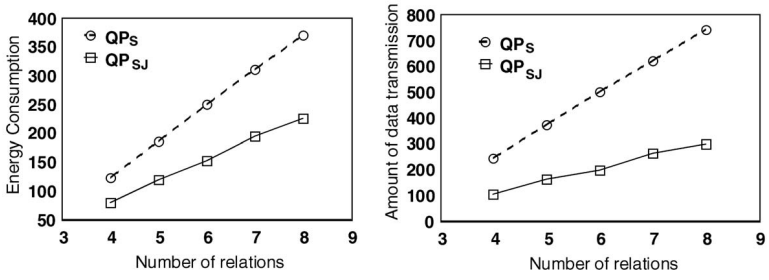

Fig. 15. The amounts of data transmission and energy consumption incurred by QP_S and QP_{SJ} in the RT phase with the number of relations varied.

6 Conclusions

In this paper, we first explored three asymmetric features of a mobile environment. Then, in light of these features, we devised query processing methods for both join and query processing. Explicitly, according to those asymmetric features of a mobile computing system, we examined three different join methods and devised some specific criteria to identify MI/SI profitable semijoins. We also proposed and investigated three query processing schemes for the processing of multijoin queries in a mobile computing system. In particular, we formulated the query processing in a mobile computing system as a two-phase query processing procedure that can determine a join sequence and interleave that join sequence with SI profitable semijoins to reduce the corresponding costs. Performance of these join and query methods was comparatively analyzed and sensitivity analysis on several parameters, including selectivity factor and idling coefficient, was conducted. Furthermore, we developed a systematic procedure to derive the characteristic functions of MI and SI profitable semijoins. It was shown by our simulation results that, by exploiting the three asymmetric features, these characteristic functions are very powerful in reducing both the amounts of energy consumption and data transmission incurred and can lead to the design of an efficient and effective query processing procedure for a mobile computing environment.

Appendix A

Expected Resulting Cardinalities of Joins

Theorem [5]. *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *be a join query graph.* $G_B = (V_B, E_B)$ *is a connected subgraph of G. Let* $R_1,\ R_2, \ldots, R_p$ *be the relations corresponding to nodes in* V_B, $A_1, A_2, \ldots, A_q$ *be the distinct attributes associated with edges in* E_B, *and* m_i *be the number of different nodes (relations) that edges with attribute* A_i *are incident to. Suppose* R^* *is the relation resulting from the join operations between relations in* G_B *and* $N_T(G_B)$ *is the expected number of tuples in* R^*. *Then,*

$$N_T(G_B) = \frac{\prod_{i=1}^{p} |R_i|}{\prod_{i=1}^{q} |A_i|^{m_i - 1}}. \tag{6}$$

Appendix B

Proofs of Lemmas and Corollaries

Proof of Lemma 1. It follows from the theorem in [5] that the size of $|R_1 \bowtie R_2|$ is $\frac{|R_1||R_2|}{|A|}$. The value of $\mathrm{d}_t(J_S)$ can be expressed as $|R_2| + \frac{|R_1||R_2|}{|A|}$. Note that $d_t(J_S) - d_t(J_C) = (|R_2| + \frac{|R_1||R_2|}{|A|}) - |R_1| = |R_2| + \frac{|R_1|(|R_2|-|A|)}{|A|}$. In general, we assume that $|R_2| > |A|$ and all $|R_1|$, $|R_2|$, and $|A|$ are larger than zero. Thus, we have $|R_2| + \frac{|R_1|(|R_2|-|A|)}{|A|} > 0$. This lemma follows. □

Proof of Lemma 2. The amount of data transmission incurred by method J_{SC} can be formulated as $\rho(|A| + |R_1|)$, where A is the join attribute with the selectivity factor ρ. $\mathrm{d}_t(J_C)$ can be expressed by $|R_1|$. Generally speaking, since $\rho < \frac{|R_1|}{|A|+|R_1|}$ and $|\mathrm{R}_1| > \mathrm{A}$, we have $\rho(|A| + |R_1|) < |R_1|$, thus proving this lemma. □

Proof of Corollary 1.1. From Theorem 1, we have $E_c(J_{SC}) - E_c(J_S) = \frac{|R_1||R_2|}{|A|}(\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho - e_r)$. From Definition 1, $\frac{|R_1||R_2|}{|A|}(\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho - e_r) < 0$. Then, we have $\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho - e_r < 0$, which implies that $\delta > \rho * r_{SM} * \frac{t_{tuple}}{e_r}$, thus proving the "only if" condition. Also, if $\delta > \rho * r_{SM} * \frac{t_{tuple}}{e_r}$, we have $\frac{1}{\delta} * r_{SM} * t_{tuple} * \rho - e_r < 0$, which implies that $E_c(J_{SC}) - E_c(J_S) < 0$, thus proving the "if" condition. This corollary follows. □

Proof of Corollary 2.1. The data transmission of performing semijoin is less than that of sending $|\mathrm{R}_j|$ if $\rho_{i,a} < \frac{|R_j|}{|A|+|R_j|}$. It follows from Theorem 2 that a semijoin is an SI profitable semijoin if $\rho_{i,a} < \frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})}$. We now want to prove that $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} < \frac{|R_j|}{|A|+|R_j|}$. Assume that $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} < \frac{|R_j|}{|A|+|R_j|}$, then we have

$$(e_r * r_E - \frac{1}{\delta} * r_{SM} * t_{tuple}) * (|A| + |R_j|) <$$
$$(e_r * r_E + \frac{1}{\delta} * r_{SM} * t_{tuple}) * |R_j|.$$

We need to prove that $2 * |R_j| * (\frac{1}{\delta} * r_{SM} * t_{tuple}) + |A| * (\frac{1}{\delta} * r_{SM} * t_{tuple} - e_r * r_E) > 0$. Since we have $|R_j| > |A|$, $\frac{1}{\delta} > 1,\ r_{SM} > 1,$ and $0 < t_{tuple} < 1$, it can be seen that

$$2 * |R_j| * (\frac{1}{\delta} * r_{SM} * t_{tuple}) + |A| *$$
$$(\frac{1}{\delta} * r_{SM} * t_{tuple} - e_r * r_E) > 0,$$

proving that $\frac{(r_E * e_r - \frac{1}{\delta} * r_{SM} * t_{tuple})}{(r_E * e_r + \frac{1}{\delta} * r_{SM} * t_{tuple})} < \frac{|R_j|}{|A|+|R_j|}$. Thus, by using an SI profitable semijoin, the corresponding amount of data transmission is reduced. □

Acknowledgments

This work was supported in part by the National Science Council of Taiwan, R.O.C., under contracts NSC93-2752-E-002-006-PAE.

References

[1] R. Alonso and S. Ganguly, "Query Optimization in Mobile Environments," *Proc. Fifth Workshop Foundations of Models and Languages for Data and Objects*, pp. 1-17, Sept. 1993.

[2] *Applications of Mobile Computing*, http://www.nokia.com/3g/index.html, 2005.

[3] D. Barbara, "Mobile Computing and Databases—A Survey," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 1, pp. 108-117, Jan./Feb. 1999.

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

[5] M.-S. Chen and P.S. Yu, "Interleaving a Join Sequence with Semijoins in Distributed Query Processing," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 5, pp. 611-621, Sept. 1992.

[6] M.-S. Chen and P.S. Yu, "Combining Join and Semijoin Operations for Distributed Query Processing," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, no. 3, pp. 534-542, June 1993.

[7] M.-S. Chen, P.S. Yu, and T.-H. Tang, "On Coupling Multiple Systems with a Global Buffer," *IEEE Trans. Knowledge and Data Eng*, vol. 8, no. 2, pp. 339-344, Apr. 1996.

[8] M.-S. Chen, P.S. Yu, and K.-L. Wu, "Optimization of Parallel Execution for Multi-Join Queries," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 3, pp. 416-428, June 1996.
[9] A. Datta, D.E. Vandermeer, A. Celik, and V. Kumar, "Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users," *ACM Trans. Database Systems,* vol. 24, no. 1, pp. 1-79, Mar. 1999.
[10] M.H. Dunham, "Mobile Computing and Databases," *Proc. Tutorial Int'l Conf. Data Eng.,* Feb. 1998.
[11] M.H. Dunham and V. Kumar, "Location Dependent Data and Its Management in Mobile Databases," *Proc. Ninth Int'l Workshop Database and Expert Systems Applications,* pp. 26-29, Aug. 1998.
[12] M.J. Franklin, B.T. Jonsson, and D. Kossmann, "Performance Tradeoffs for Client-Server Query Processing," *Proc. ACM SIGMOD,* pp. 149-160, June 1996.
[13] S. Ganguly, "Design and Analysis of Parametric Query Optimization Algorithms," *Proc. 24th Int'l Conf. Very Large Databases,* pp. 228-238, 1998.
[14] T. Imielinski and B.R. Badrinath, "Querying in Highly Mobile and Distributed Environment," *Proc. 18th Int'l Conf. Vary Large Data Bases,* pp. 41-52, Aug. 1992.
[15] T. Imielinski and S. Goel, "DataSpace—Querying and Monitoring Deeply Networked Collections in Physical Space," *Proc. Int'l Workshop Data Eng. Wireless and Mobile Access,* pp. 44-51, 1999.
[16] R. Jain and N. Krishnakumar, "Asymmetric Costs and Dynamic Query Processing in Mobile Computing Environments," *Proc. Fifth WIN-LAB Workshop,* Apr. 1995.
[17] R. Jain and N. Krishnakumar, "An Asymmetric Cost Model for Query Processing in Mobile Computing Environments," *Proc. ACM Symp. Applied Computing,* pp. 365-372 1996.
[18] J. Jing, A. Helal, and A. Elmagarmid, "Client-Server Computing in Mobile Environments," *ACM Computing Surveys,* vol. 31, no. 2, pp. 117-157, June 1999.
[19] D.N. Knisely, S. Kumar, S. Laha, and S. Nanda, "Evolution of Wireless Data Services: IS-95 to cdma2000," *IEEE Comm. Magazine,* pp. 140-149, Oct. 1998.
[20] C.-H. Lee and M.-S. Chen, "Using Remote Joins for the Processing of Distributed Mobile Queries," *Proc. Seventh Int'l Conf. Database Systems for Advanced Applications,* pp. 226-233, Apr. 2001.
[21] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor for Sensor Networks," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 491-502, June 2003.
[22] "Palm Pilots of 3com," http://www.palm.com, 2005.
[23] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proc. 13th Int'l Conf. Data Eng.,* pp. 422-432, Apr. 1997.
[24] N.R. Sollenberger, N. Seshadri, and R. Cox, "The Evolution of IS-136 TDMA for Third-Generation Wireless Services," *IEEE Personal Comm.,* pp. 8-18, June 1999.
[25] C. Wang and M.-S. Chen, "On the Complexity of Distributed Query Optimization," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 4, pp. 650-662, Aug. 1996.
[26] "WAP application in Nokia," http://www.nokia.com/nokia/0,,62652,00.html, 2005.
[27] "WAP application in Unwired Planet, Inc.," http://www.openwave.com/us/products/developer_products/wap_push_library/index.htm, 2005.
[28] "WAP Forum," http://www.wapforum.org, 2005.
[29] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, "Cost and Imprecision in Modeling the Position of Moving Objects," *Proc. 14th Int'l Conf. Data Eng.,* pp. 588-596, Feb. 1998.
[30] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," *Proc. Conf. Innovative Data Systems Research.,* pp. 21-32, 2003.
[31] C.T. Yu and C.C. Chang, "Distributed Query Processing," *ACM Computer Surveys,* vol. 16, no. 4, pp. 399-433, Dec. 1984.

Wen-Chih Peng received the BS and MS degrees from National Chiao Tung University, Taiwan, in 1995 and 1997, respectively, and the PhD degree in electrical engineering from the University of National Taiwan University, Taiwan, Republic of China, in 2001. He is currently an assistant professor in the department of computer science and information engineering at the National Chiao Tung University. His research interests include mobile data management, sensor data management, and data mining. He is a member of IEEE, the IEEE Computer Society, and a member of the Phi Tau Phi scholastic honor society.

Ming-Syan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, Michigan, in 1985 and 1988, respectively. He is currently a professor and the chairman of the Graduate Institute of Communication Engineering, a professor in the Electrical Engineering Department, and also a professor in the Computer Science and Information Engineering Department, National Taiwan University, Taipei, Taiwan. He was a research staff member at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 180 papers in his research areas. In addition to serving as program committee members in many conferences, he served as an associate editor of *IEEE Transactions on Knowledge and Data Engineering* (TKDE) from 1997 to 2001, is currently on the editorial board of the *Very Large Databases Journal* (VLDB), *Knowledge and Information Systems Journal*, *Journal of Information Science and Engineering*, the *International Journal of Electrical Engineering*, and was a distinguished visitor of the IEEE Computer Society for Asia-Pacific from 1998 to 2000. He served as the program chair of PAKDD-02 (Pacific Area Knowledge Discovery and Data Mining), program vice chair of the IEEE International Conference on Distributed Computing Systems (ICDCS) 2005 and the International Conference on Parallel Processing (ICPP) 2003, program vice chair of VLDB-2002 (Very Large Databases), and also general chair and program chair of several other conferences. He was a keynote speaker on Web data mining in the International Computer Congress in Hong Kong, 1999, a tutorial speaker on Web data mining in DASFAA-1999 and on parallel databases in the 11th IEEE ICDE in 1995, and also a guest coeditor for *IEEE TKDE* on a special issue for data mining in December 1996. He holds or has applied for eighteen US patents and seven ROC patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He is a recipient of the NSC (National Science Council) Distinguished Research Award and K.-T. Li Research Penetration Award for his research work and the Outstanding Innovation Award from IBM Corporate for his contribution to a major database product. He also received numerous awards for his research, teaching, inventions, and patent applications. He is a fellow of the IEEE and a member of the ACM.