Asymmetric Costs and Dynamic Query Processing in Mobile Computing Environments

Ravi Jain and Narayanan Krishnakumar
Bell Communications Research, 445 South Street, Morristown, NJ 07960
email: {rjain@thumper,nkk@marble}.bellcore.com

Abstract. We address the issue of query processing for information services in mobile environments, using a mobile sales and inventory application example. Efficient query processing depends upon choosing one of several candidate query plans based upon a cost model which accurately reflects computation and communication costs. Unlike traditional distributed database environments, mobile environments necessitate new cost models which are *asymmetric* in the sense that the resources available at mobile sites are less than those at fixed sites, and communication costs vary with the direction of data transfer. In addition, communication costs may vary as the mobile moves. We design a cost model which incorporates these criteria and illustrate it via examples. As costs change dynamically, it is desirable that the query execution plan be modified dynamically to optimize the cost. Existing distributed query processing algorithms can be modified to suit the mobile environment and this paper discusses our ongoing work to do so.

1 Introduction

A challenging area of mobile computing is support for mobile access to databases and, in particular, efficient querying of public and corporate databases. In this paper we consider mobile database access which is provided by a commercial entity called the Information Service and Applications Provider (ISAP). The ISAP maintains a set of servers which contain the appropriate database information and run applications, and which are connected to the mobile user via a personal communications services (PCS) network [8]. For scalability, performance and availability reasons [12] we assume the ISAP architecture consists of multiple distributed servers which function as a single logical database.

For concreteness, we use a sales and inventory application as an example. The user is a salesperson who is provided mobile access to a corporate ISAP database containing customer and product information. The salesperson carries a mobile computing device (generically called a Personal Digital Assistant or PDA), which maintains a fragment of the database containing information about his or her regular customers, their orders, transaction history, etc. Thus the database as a whole can be viewed as having a fixed component, which resides on the ISAP's fixed servers, and a mobile component, which consists of database fragments (possibly replicated) which reside on salespersons' PDAs. We assume that, for availability and reliability reasons, any database fragment which resides on a PDA also has at least one copy which resides on a fixed server; the fixed server copy is generally less up-to-date than the PDA copy, as the latter contains the latest information entered by the salesperson. In [15] we have described the motivation and the background for this application in more detail. The mobile sales and inventory application has attracted substantial interest recently, and some commercial products have become available [22, 23].

In previous work we have described an architecture,

protocols and network facilities for mobile access to databases [11, 12]. In this paper, we focus on query processing in a mobile environment. For instance, in the sales and inventory application, the salesperson will need to query the database occasionally, e.g. a salesperson can gather a list of products required by a customer and then wish to get information about the availability of those products, by performing a set of database join operations with the data stored at the ISAP. In general, the relations at the ISAP might be horizontally fragmented across various sites of the ISAP. In order to evaluate complex queries that may be posed by the salesperson, it would be necessary to generate a *query plan*, i.e., a description of a sequence of steps where fragments of relations, or intermediate results, are moved appropriately among ISAP servers and the salesperson's PDA in order to satisfy the query. Several heuristic techniques [20] have been developed to deal with query evaluation in a traditional database environment. We see how these techniques can be adapted to the needs and costs of the mobile environment.

The outline of the rest of the paper is as follows. In the following section we discuss the background and motivation for generation of query plans in a mobile environment, noting that a key issue is the development of an appropriate cost model for plan evaluation. In sec. 3 we summarize the relevant aspects of our underlying system model and architecture. In sec. 4 and 5 we discuss the characteristics of the mobile query processing environment which differ from those of processing in fixed environments, and develop a cost model for query plan evaluation, illustrating it with examples. In sec. 6, we review some of the existing distributed query processing algorithms and provide a preliminary discussion of their applicability to the mobile environment. In sec. 7 we conclude with an outline of the issues of query plan generation, parameter estimation, and plan evaluation which arise, and which we are currently addressing in our ongoing work.

2 Background and motivation

In general, it is possible to generate more than one query plan to satisfy a query, and the database query optimizer attempts to select the plan which will incur the least *cost* from among the candidate plans. In making this selection the optimizer relies upon dynamically maintained estimates about the relations in the database (i.e., 'meta-information' such as relation sizes, the domains of various attributes, etc), as well as a *cost model* which estimates the costs of various tasks required to perform the operation. Typically, the costs included in the cost model are the cost of communication for transferring fragments from one site to another, the cost of local CPU processing at a site and the cost of local I/O [16].

The generation and selection of query plans for distributed databases with fixed servers has been studied extensively in the literature (see [14, 24] for surveys), and is a key factor in the efficacy of database optimizers [19]. The selection of a good query plan is critically dependent upon how accurately the cost model reflects the operating environment. Thus the merits of different cost models have been experimentally investigated leading to the tuning of cost models and their successful application in both commercial and research distributed database systems [17, 20, 19].

The cost models developed for query processing in distributed database systems with fixed server sites, however, are unlikely to be effective for mobile environments. One reason for this is that mobile environments introduce an important cost criterion which has not been considered for fixed servers, namely that of *energy consumption* by the mobile station. The mobile station is typically supplied by a limited energy source (disposable or re-chargeable batteries); the restricted useful battery life of current mobile computing devices is one of the major factors inhibiting their market success and customer acceptance, and advances in battery technology are

unlikely to substantially alleviate this problem in the near future. Thus conserving energy at the mobile station is a critical consideration, and one which is receiving increasing attention [1, 3].

We consider cost models which reflect the need to conserve the mobile station's energy during query processing. Alonso and Ganguly [1] have also considered mobile station energy as a criterion during query processing, but have taken the approach of studying the tradeoff between available energy at the mobile station and the workload at a fixed server. Thus, while processing a query at the mobile station depletes its energy, processing the query at a fixed host increases the workload at the server and still incurs some energy consumption at the mobile as it idles waiting for a response from the server. However, the authors do not consider the energy consumption due to sending and receiving data from the fixed server over the wireless communication link. This is an important contributor to the mobile's energy consumption and needs to be considered in the cost model. (Furthermore, we believe the problem of server overload can be more easily mitigated by adding processing power to the server sites.)

It is also important to note that energy consumption not only adds a new factor in the cost model, but also alters it significantly by introducing *asymmetry*. In particular, fixed database cost models, which do consider communication costs, typically assume that the communication cost between two sites is the same irrespective of the direction of transmission [24, 16, 20]. However, the mobile environment is unique in that communication costs are not symmetric since the energy consumed by the mobile to transmit data is more than that required to receive [3]. In addition to energy, *wireless bandwidth* is also an important cost criterion in mobile environments, and one we believe will typically be more important than fixed server processing requirements.

Another issue unique to mobile environments is that *communication costs* for mobile users typically *change* as the user moves. We elaborate on these observations and discuss the key aspects of the mobile query processing environment in sec. 4.

3 The system model

We assume that the ISAP has a set of geographically distributed servers that are attached to the (public) telecommunications network. The data stored at the ISAP could be fragmented and/or replicated across the servers. The servers can be interconnected using either a private ISAP network attached to the PCS network, or using the PCS network itself as the inter-server communication backbone. We assume that messages between the mobile and the ISAP can be transferred via either a connection-oriented or a connection-less data transport mechanism. However, at the application level, the appearance of connection-oriented communication between the mobile and the server is maintained.

The geographical coverage area for the information service is partitioned into *service areas* analogous to PCS registration areas. It is likely that a service area will cover several PCS registration areas. Each service area is served by a local ISAP server which acts as the point of contact for any mobile in that service area. In [11, 12], we have introduced the notion of service handoff, where service to a mobile moving from one service area to another is "handed-off" from one server to another without disruption in service. The motivation for a service handoff is that even though the wireless cost of communicating with either server remains the same, the wired cost and communication latency (through the telecommunications network) of communicating with the new server is smaller than that of communicating with the old server. The decision to perform the service handoff is made by an ISAP *matchmaker*, based on this difference in cost. This cost differential also impacts the cost model we develop in this paper.

4 Key features of mobile environments

The mobile environment has a few salient features which cause us to alter the cost model commonly applied to a fixed distributed environment.

1. **Asymmetry between send and receive energy:** The energy expended at the mobile station for sending data on a wireless link is more than the energy expended when it is receiving the same amount of data [3]. We define the *send-receive energy* coefficient, SRE, as the ratio of the energy consumed when transmitting data to the energy consumed when receiving. Typically, for current mobile hosts, SRE $\sim$ 2 - 10.

2. **Asymmetry between wired and wireless links in terms of energy consumption:** Communicating over wireless links consumes significant energy at the mobile. In addition, if the mobile station has access to a wired link for communication (e.g., in our mobile sales example, when the salesperson is at his or her hotel), then it possibly also has access to an AC power source. Thus the energy cost for communication can be taken to be zero for the mobile station when communicating over a wired link, but is a significant factor when communicating over a wireless link.

3. **Asymmetry in computing and idle energy:** The energy consumed when the mobile station is actively computing is much more than that consumed when the mobile is idle. In [1], the idling coefficient I is defined to be the ratio of the power consumed in idle mode to the power consumed in computing mode. (For current mobile machines, the idling coefficient can be between 0.04 and 0.6.)

4. **Asymmetry in bandwidth availability and cost between wireless and wired links:** The bandwidth of wired links is significantly more then wireless communication bandwidth. So the cost of transmitting data over a wireless link is more than that of transmitting over a wired link.

5. **Change in cost of communicating with a mobile as it moves:** There are two scenarios to be considered here.

   (a) The first scenario deals with a mobile moving within a cell. When the mobile is close to a base station, it need not consume as much power in transmitting to the base station as it does when it is further away. In some current PCS standards proposals (e.g. Bellcore's WACS system [21]), the base station can instruct the mobile to change its transmission power level dynamically depending upon the received signal strength at the base station.

   (b) The second case arises due to the mobile moving between service areas. Suppose a server wishes to send some information to a mobile. One option is to call the mobile directly and send it the information over the PCS network. However, it is possible that the mobile is in some other service area. Since the first server is connected to the mobile's current server through the ISAP network, it is likely that the first server could send the information to the mobile's current server over the ISAP network, and have the latter deliver the information over the PCS network. This scheme could result in a smaller cost of relaying the information.

Both the situations described above suggest that a query evaluation plan can evolve dynamically in two ways: (a) the relative cost of the

mobile sending or receiving data may change depending on the proximity of the mobile to the base station, and (b) the decision to perform a portion of the query at a particular server may now have to be revised based on the current location of the mobile. We will concentrate on the former case when considering this cost factor in this paper.

5 A new cost model

In order to discuss appropriate cost models it is first necessary to make some assumptions about the design of the database and the kinds of queries which are likely to be generated in our application domain.

In the mobile sales application, there are typically two sets of information that a salesperson needs to access. One is customer-specific information that we assume resides on his PDA for fast access, and the other is global information that is accessed by all salespersons at the ISAP. For instance, the salesperson would store on his PDA a relation indicating the profile of the customers (their orders, history, etc.). The relation storing global information about product inventory would be stored on fixed servers. The customer-specific information on the PDA could be backed up at a fixed server from time to time, but we assume that this is only for recovery, billing or archival purposes. Since the most recent customer information is on the PDA, a query generated by the salesperson would be a sequence of joins across the relations stored on the mobile and the relations stored at the fixed server.

In general, we assume that for cost, response time, and administrative reasons some of the database relations will be horizontally fragmented, with one replica of each fragment residing on a mobile host, and another replica residing on a fixed server. We also assume that some relations will reside only (or primarily) on fixed servers, and these relations may be replicated as well as fragmented across the servers. We assume that mobile queries do not access information which resides on other mobile hosts. Furthermore, we do not address the issue of system-wide concurrency control of updates to the data stored on different mobiles, since it is likely that the same data has not been stored for update on two different mobiles (salespersons typically operate in different geographical regions with different sets of customers).

We now discuss candidate query plans which can be generated for a simple join query using an example. We use the following notation: $r(R)$ denotes a relation r with attribute set R, and $r(R) \bowtie_{R \cap S} s(S)$ denotes the join of relations r and s on their common attributes $R \cap S$; we will usually omit the join attributes. For simplicity we assume the join predicate is the equality relation. The projection of relation r on attributes $A \subseteq R$ is denoted $\Pi_A(r)$.

Example 1. Suppose there is a (horizontal fragment of a) relation $r(R)$ stored on the mobile. A relation $s(S)$ is stored at a fixed server site A. The mobile launches the query $r \bowtie s$. The following candidate query plans can be generated.

1. Processing at the server. The mobile transmits r to A, along with a control message. A computes $r \bowtie s$ and returns the result.
2. Processing at the mobile. The mobile sends a control message to A, which responds by sending s for computing $r \bowtie s$.
3. Semijoin. The mobile sends $r' = \Pi_{R \cap S}(r)$ to A, along with a control message. A computes $s' = r' \bowtie s$ and returns s'. The mobile then computes $s' \bowtie r$ to obtain the result.

While the candidate query plans for Example 1 do not differ from those if only fixed servers were involved, their cost evaluation does. The following new entities will have to be included as part of the cost model:

1. The energy consumed by the mobile in receiving data over the wireless link. We will model this as consisting of two components: receive energy per packet, re_p, and receive energy per byte, re_b. We assume packets are of fixed length L_p bytes. The total energy for receiving L bytes is then $re(L) = re_b * L + re_p * \lceil L/L_p \rceil$.

2. The energy expended by the mobile in transmitting data over the wireless link. Analogous to the receive energy, the total energy for sending L bytes is $se(L) = se_b * L + se_p \lceil L/L_p \rceil$. Let E denote the send-receive energy coefficient (SRE). Then we assume $se(L) = re(L) * E$ (In general, SRE is not constant since the distance of the mobile from the base station determines the power required for the mobile to transmit data. We address this issue in the section on dynamic query plan generation.)

3. The energy consumed by the mobile for each second of active CPU processing p, and each second of I/O, i. We will model these factors by first estimating the time required to perform the given operations on the relations involved. For instance, we will denote the CPU and I/O time required to perform $r \bowtie s$ as $t_p(r \bowtie s)$ and $t_i(r \bowtie s)$ respectively. (The functions t_p and t_i must take into account the size of the relations and other factors.) The total energy consumed as $ce(r \bowtie s) = p * t_p(r \bowtie s) + i * t_i(r \bowtie s)$.

4. The energy consumed by the mobile for each second of idling while waiting for results. We calculate this waiting time as a fraction of the time required if the query had been processed at the mobile. Thus let M denote the *server to mobile processing* (SMP) coefficient, i.e., the ratio of the processing speed of the fixed server to that of the mobile. Then the time spent idling is $(t_p(r \bowtie s) + t_i(r \bowtie s))/M$. Let I denote the idling coefficient. Then the energy spent idling is approximated to $I * ce(r \bowtie s)/M$.

5. The cost of wireless ("air") bandwidth consumed for either sending or receiving a packet of length L bytes to the mobile is given by $a(L)$.

Based upon our previous discussion we presume that the SRE coefficient $E > 1$, the SMP coefficient $M > 1$, and the idling coefficient $0 < I < 1$. We continue Example 1 by comparing expressions for the energy and bandwidth costs (C_E and C_B respectively) of the candidate plans assuming the cost model above. We ignore control messages since they incur the same cost for all the candidates. Let $|r|$ denote the size of relation r.

1. Processing at the server.
   $C_E(1) = se(|r|) + re(|r \bowtie s|) + I * ce(r \bowtie s)/M$
   $\quad = E * re(|r|) + re(|r \bowtie s|) + I * ce(r \bowtie s)/M$
   $C_B(1) = a(|r|) + a(|r \bowtie s|)$

2. Processing at the mobile.
   $C_E(2) = re(|s|) + ce(r \bowtie s)$
   $C_B(2) = a(|s|)$

3. Semijoin.
   $C_E(3) = ce(\Pi_{R \cap S}(r)) + se(|r'|) + re(|s'|)$
   $\quad + I * ce(r' \bowtie s)/M + ce(r \bowtie s')$
   $\quad = E * re(|r'|) + re(|s'|) + I * ce(r' \bowtie s)/M$
   $\quad + ce(r \bowtie s')$
   $C_B(3) = a(|r'|) + a(|s'|)$

We now evaluate the candidates assuming the following example values: $E = 4$, $M = 5$, $I = 0.3$, $|r| = 0.33|s|$, $|r \bowtie s| = |r|$, $|r'| = 0.5|r|$, and $|s'| = 0.25|s|$.

$C_E(1) = 5re(|r|) + 0.06ce(r \bowtie s)$
$C_B(1) = 2a(|r|)$

$C_E(2) = 3re(|r|) + ce(r \bowtie s)$
$C_B(2) = 3a(|r|)$

$C_E(3) = 2.75re(|r|) + 1.28ce(r \bowtie s)$
$C_B(3) = 1.25a(|r|)$

Depending upon the costs of communication and computation energy, the cost of wireless bandwidth,

and the sizes of intermediate relations produced during semijoins, any of the three query plans of Example 1 could be the lowest cost plan. Example 1 does not show the effect of wired communication costs; we demonstrate this by using the following example.

Example 2. As in Example 1, $r(R)$ is on the mobile. Relation $s(S)$ is horizontally fragmented into fragments s_A and s_B, stored at different server sites A and B respectively. For simplicity we assume in this example that $R = S$.

1. The mobile transmits r to one of the servers, say A. Server A sends a control message to server B to obtain s_B, computes both $r \bowtie s_A$ and $r \bowtie s_B$, collates the results and transmits $r \bowtie s$ to the mobile.

2. The mobile transmits r to one of the servers, say A. Server A forwards r to B, computes $r \bowtie s_A$ and sends the result to B. Server B computes $r \bowtie s_B$, collates the partial results, and sends $r \bowtie s$ to the mobile.

3. The mobile transmits r to one of the servers, say A. Server A forwards r to server B. The servers compute $r \bowtie s_A$ and $r \bowtie s_B$ respectively, and each server sends the result to the mobile; the latter collates the results to obtain $r \bowtie s$.

4. The mobile sends r to both servers, who compute and return $r \bowtie s_A$ and $r \bowtie s_B$ respectively. The mobile collates the results to obtain $r \bowtie s$.

5. The mobile sends a control message to each server, requesting the fragments of s. The servers send s_A and s_B to the mobile, who computes $r \bowtie s_A$ and $r \bowtie s_B$ and collates the results to obtain $r \bowtie S$.

Note that in Example 2 the plans in which server A forwards r to server B would typically not be generated if all sites were fixed and fully connected via wired links. However, in the mobile environment, reducing both the number of transmissions and the amount of data transmitted by the mobile is important, so that these plans may well be the least costly overall.

Example 2 shows that wired communication cost should be included as a parameter in the cost model if the mobile's query involves relations which are fragmented across different fixed server sites. It is possible then to write and compare cost expressions similar to those for Example 1.

We have thereby illustrated how traditional cost models need to be modified in order to accurately reflect the mobile environment.

6 Distributed query processing and mobility

Now that we have defined a cost model, we turn our attention to the query optimization algorithm which will use the cost model. There have been numerous distributed query optimization algorithms which have been previously developed and implemented (see [18, 24, 20, 16] for surveys). Rather than develop a totally new algorithm, the approach we take is to examine the unique characteristics of the mobile environment and attempt to apply a previously developed algorithm, modifying it if necessary. One goal of our approach is to keep the modifications required for mobility as few as possible.

We examine the characteristics of previous algorithms in terms of various criteria and discuss which criteria seem most relevant to the mobile environment.

Previous distributed query optimization algorithms can be divided into *static* and *dynamic* algorithms. A static query optimizer optimizes the entire query before execution begins, relying on the existence of a good technique for evaluating the sizes of intermediate relations. Dynamic algorithms however opti-

mize the query as the execution proceeds, and the *actual* sizes of intermediate relations generated thus far are used to determine the next processing step. For example, the R* algorithm [17] and the family of algorithms proposed by Apers, Hevner and Yao [2] (which we refer to as AHY) are static, while the distributed Ingres algorithm [6] is dynamic.

For the mobile environment, we observe two reasons why dynamic optimization may be preferable to static optimization. The first is the potential for using the actual sizes of intermediate relations, as explained above. The second reason is that the costs of communicating with a mobile can change as the mobile moves. As discussed above, this change can occur in terms of transmission power required as a mobile's distance from the base station changes, or in terms of the wired communication costs as the mobile changes service areas.

Query optimization algorithms can also be divided into those that are based on semi-joins and those that are not. For example, the distributed Ingres and the R* algorithms are not semi-join based, but the AHY algorithms are. Semi-joins are useful if the selectivity factor[1]of the join is small, so that sending the result of the semi-join from one site to another is less expensive than transporting entire tables. However, semijoin increases the local processing cost. It has been observed that semijoins might not be a good idea if the communication cost is not the dominant factor (as in local area networks) [17, 20]. However, it is possible that the mobile environment, with its limited bandwidth, is one in which semijoins might provide better performance.

A third criterion of importance is the objective function optimized by the algorithm. Optimization algorithms typically minimize the *response time* or *total time* of a query. For query optimization purposes, the response time of a query is defined as the time between when the query is launched until the receipt of the complete result at the result site. The total time is defined as the sum of the times spent in computation and communication by all sites in order to satisfy the query; in a sense it is the "total work". Clearly, minimizing response time tends to favor the maximum exploitation of parallelism, which may result in increased total time. The R* algorithm minimizes total cost, while both distributed Ingres and AHY have versions which minimize response time. It is likely that response time minimization will be a more important objective than total time minimization for the mobile sales application due to its interactive nature.

It is worth pointing out here that previous optimization algorithms typically minimize only one objective function, even though they may consider several different cost components. Thus computation, communication, and I/O costs are reduced to the same units by measuring them in terms of time, and the optimizer attempts to minimize the costs in terms of time (either response time or total time) [20]. We believe it is important to consider energy consumption as well as wireless bandwidth utilization in the mobile environment. It is possible to reduce these cost factors to the same units (e.g. dollars) by multiplying each by appropriate weighting factors. For the moment, we will continue to consider them separately.

Other criteria to consider are the types of costs considered in the cost model used by the optimization algorithm. Older algorithms, such as AHY, typically only consider communication costs [20], which may be quite relevant to the mobile environment but, as discussed in previous sections, local computation (energy) costs at the mobile are also an important factor.

The final criterion we consider is the query optimization ability to handle fragments and replicas. The distributed Ingres algorithm, and those of [16] can handle fragments, while AHY and R* cannot.

[1]The selectivity factor of an operation is the proportion of tuples of operand relation(s) that appear in the result of the operation.

Criterion	Response Time	Dynamic	Semijoin	Computing Costs	Replicas/ Fragments
Distributed Ingres [6]	Y	Y	N	Y	N
AHY [2]	Y	N	Y	N	N
Mermaid [25]	N	Hybrid	Y	N	Y
Liu & Yu [16]	Y	N	N	Y	Y
SDD-1 [4]	N	N	Y	N	N
Chu & Hurley [5]	N	N	N	Y	Y
R* [17]	N	N	N	Y	N

Table 1: Criteria for selecting a basic distributed query optimizer

We summarize these considerations in Table 1; the Table is based upon information contained in the surveys by [18, 20]. (We do not consider algorithms mentioned in the surveys which are designed for specialized types of networks for interconnecting the ISAP servers e.g. star networks.) A "Y" in the Table indicates that the algorithm does satisfy the criterion (e.g. distributed Ingres does minimize response time but R* does not.) The Mermaid algorithm [25] is hybrid in the sense that dynamic optimization is invoked if the actual sizes of intermediate relations are significantly different from those estimated *a priori*.

Based upon these observations, we identify the query optimization algorithms of distributed Ingres and AHY as having potential for applicability to the mobile environment from amongst those considered. However, neither algorithm satisfies all the desired properties. For instance, AHY is not dynamic and distributed Ingres does not consider semijoins. For this paper, we investigate how the AHY algorithm for the case of simple queries (defined below) could be modified and applied to the mobile environment. A more detailed exposition of how the AHY and other algorithms can fit the mobile environment is given in [13].

6.1 An example distributed query processing algorithm

We consider the kinds of changes which may be required in a traditional distributed query optimization algorithm in order to support mobility. For this purpose, we take one of the AHY algorithms, called PARALLEL, as an example. We are currently investigating various other algorithm candidates also, but will focus on PARALLEL here to demonstrate the issues involved.

The AHY family of algorithms is based on a set of optimal static algorithms that are applicable only to *simple queries*. A simple query is one where after initial local processing at each site (i.e., selects, projects, and joins between relations at the same site), each relation in the query contains only one common join attribute, which is also the output of the query. (The AHY family also includes polynomial-time non-optimal algorithms for general queries; we will not consider them here for the sake of simplicity.)

The PARALLEL algorithm [7, 2] is designed to minimize the response time for satisfying a simple query, and can be written in pseudo-code as follows. Let the simple query have the form $q = R_1 \bowtie R_2 \bowtie ...R_m$ where relation R_i resides at site i, and has size s_i, in bytes. The site where the result is desired is called

the result site. The cost component considered is the communication cost, represented in terms of time; the cost of sending R_i between any two sites is denoted $C(s_i)$. A selectivity factor p_i $(0 \leq p_i \leq 1)$ is defined for each R_i as the ratio of the number of distinct values in the join attribute of R_i to the number of possible values for that attribute. It is assumed that if $R_i \bowtie R_j$ is performed, the size of the result of the join becomes $s_j p_i$.

Algorithm PARALLEL starts with an initial feasible solution in which all the R_i are moved directly to the result site. The *response time for relation* R_i, r_i, is defined as the time from the start of the query until R_i is received at the result site. The algorithm then examines all possible transmissions of (each) R_i to an intermediate site which have the potential to reduce r_i compared to the initial feasible solution. The minimum response time for R_i is $\min(r_i)$, where the minimization is performed over all possible schedules for transmitting R_i.

Algorithm PARALLEL
1. Sort the relations R_i such that $s_1 \leq s_2 \leq ... \leq s_m$.
2. **for** $i = 1, m$ { $r_i = C(s_i)$ }
3. **for** $i = 2, m$ {
4. **for** $j = 1, i-1$ {
5. $r_{ij} = r_j + C(s_i * \Pi_{k=1}^{j} p_k)$
6. }
7. $r_{ii} = r_i$
8. Let k be such that $r_{ik} = \min\{r_{ij} : j = 1, ..., i\}$
9. $r_i = r_{ik}$
10. }

Hevner and Yao [7] have shown that this algorithm is optimal. Note that the algorithm also relies upon the accuracy of the selectivity factor; in practice, this factor can only be estimated.

We now briefly discuss some of the modifications to this algorithm required to make it more suitable for the mobile environment.

1. Sites need to be distinguished as mobile or fixed sites, and the mobile is the result site.
2. The communication cost function has to be made accurate by incorporating the asymmetric cost model we have defined, as follows.
   (a) The communication cost C is replaced by two cost functions, in which mobile energy consumption and wireless bandwidth utilization are considered.
   (b) The energy consumed while sending information from the mobile has to be differentiated from that consumed when it is receiving.
   (c) The energy consumed during local computation at the mobile when performing database operations (local selects, projects and joins) has to be included.
   (d) The energy consumed while the mobile is idling, waiting for computations at the fixed servers, has to be included.
3. In step 1 of PARALLEL, the relations are sequenced in the increasing order of their size. The assumption is that sending a relation over any link incurs the same cost and that a larger size means higher cost. However, in the mobile case, this assumption has to be altered due to the communication cost being modified as above. Thus it makes more sense to weight the size of each relation with the cost factor, and order the weighted sizes in ascending order.
4. The static algorithm described in PARALLEL might be followed by a run-time phase during which the dynamic aspects of the query evaluation are considered, as follows.
   (a) The cost of sending information from the mobile is updated dynamically as the user moves and power control is applied.
   (b) The weighted sizes of intermediate relations produced are used to recompute the query plan, i.e., to determine the best sequence and sites for performing the remaining operations in the query plan.

(c) If the user crosses service areas and a service handoff takes place, the query plan might have to recomputed.

7 Current Work

The problem of query processing in a mobile environment results in several new factors being added to the cost model. We have also demonstrated scenarios in which the costs can change while a query is being processed. We briefly mention related issues we are addressing in our ongoing work.

Our primary goal is to modify some of the existing algorithms in the literature to accommodate the cost model developed for the mobile environment. One question is the issue of where plan generation should be performed, i.e., at a fixed server or at the mobile. It is possible that the former is preferable because the server will have more current information about where fragments and replicas reside, and also because doing so would save the mobile's energy. However, in some cases, for queries involving only a few fragments and sites about which the mobile has recently collected parameter information, it might be beneficial to the mobile to perform plan generation and candidate evaluation.

Furthermore, there is the issue of how to efficiently generate query plans, since the number of candidate query plans grows very quickly with the number of relations, fragment sites and replica sites involved. The dynamic distributed Ingres accommodates fragments, but narrows the search space of query plans by taking locally optimal decisions that are not necessarily globally optimal. Some other algorithms do not allow fragments or replicas. An adaptation of a dynamic incremental heuristic that also handles fragments and replicas is desirable in the mobile environment.

Another issue is that of relation size estimation that is integral to several optimizers. For this reason, the mobile could maintain estimates about sizes and other parameters of the fragments residing locally, and send these to the fixed server when it launches a query. Other strategies are possible here too.

An emerging aspect of mobile query processing is the recent work on wireless data broadcasting [10]. In this work, some frequently demanded data items are broadcast by the ISAP to all users from time to time, and the user can access the data required directly from the broadcast. This technique introduces another variable in query evaluation, since a user might wish to deal with data that is both broadcast and not broadcast. Furthermore, even for broadcasted data, the user might decide to access the data directly from the ISAP if the wait for the data over the broadcast channel is estimated to be longer than permissible. The techniques developed here would have to be extended to deal with broadcasts.

References

[1] R. Alonso and S. Ganguly, "Query Optimization in Mobile Environments", LCSR-TR-219, Rutgers University, Dec. 1993.

[2] P. M. G. Apers, A. R. Hevner and S. B. Yao, "Optimization algorithms for distributed queries", IEEE Trans. Soft. Eng., 129-140, Jan. 1983.

[3] B. R. Badrinath and T. Imielinski, "Data management issues in mobile computing", *Wireless Datacomm '92*, 1992.

[4] P. A. Bernstein, N. Goodman, E. Wong, C. L. Reeve and J. Rothnie, "Query processing in a system for distributed databases (SDD-1)", ACM Trans. Database Sys., vol. 6, Dec. 1981.

[5] W. W. Chu and P. Hurley, "A model for optimal processing for distributed databases" Proc. IEEE COMPCON, 116-122, Spring 1979.

[6] R. Epstein, M. R. Stonebraker, and E. Wong, "Distributed query processing in a relational data base system", Proc. ACM SIGMOD, pp. 168-180, May 1978.

[7] A. R. Hevner and S. B. Yao, "Query processing in distributed database systems", IEEE Trans. Soft. Eng., 177-187, May 1979.

[8] "Feature description and functional analysis of Personal Communications Services (PCS) Capabilities", Bellcore Special Report, SR-TSV-00230, Apr. 1992.

[9] J. Gray and A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, 1993.

[10] T. Imielinski, S. Viswanathan and B.R. Badrinath "Energy Efficient Indexing on Air" Proceedings of the 1994 ACM SIGMOD Conference on Management of Data, pp. 25-36, May, 1994.

[11] R. Jain and N. Krishnakumar, "Network Support for Personal Information Services to PCS Users", *IEEE Conf. Networks for Pers. Comm. (NPC)*, Long Branch, NJ, Mar. 1994.

[12] R. Jain and N. Krishnakumar, "Service handoffs and virtual mobility for delivery of personal information services to mobile users", Submitted for publication, 1994.

[13] R. Jain and N. Krishnakumar, "Dynamic Query Processing in Mobile Environments", (in preparation).

[14] W. Kim and D. S. Reiner and D. S. Batory (eds.), "Query Processing in Database Systems", 1985.

[15] N. Krishnakumar and R. Jain, "Protocols for maintaining inventory databases and user service profiles in mobile sales applications", Proceedings of MOBIDATA workshop, Rutgers Univ., Nov 1994.

[16] C. Liu and C. Yu, "Performance issues in distributed query processing", *IEEE Trans. Par. Distrib. Sys.*, pp. 889-905, Aug, 1993.

[17] L. Mackert and G. Lohman, "R* optimizer validation and performance evaluation for distributed queries", *Proc. VLDB Conf.*, pp. 149-159, 1986.

[18] Mohan, C., "Part III: Distributed Query Processing: Summary", *Tutorial on Principles of Distributed Data Management*, IEEE, 1984.

[19] Oracle Inc, "The Optimizer", in *Oracle Concepts Manual*, 1994.

[20] T. Özsu and P. Valduriez, "Principles of Distributed Database Systems" Prentice Hall, 1991.

[21] "Generic criteria for Version 0.1 Wireless Access Communications Systems (WACS)", Bellcore Technical Advisory, TA-NWT-001313, Issue 1, July 1992.

[22] V. Schnee, "An excellent adventure", *Wireless*, pp. 40-43, Mar./Apr., 1994.

[23] J. Schwartz, "Upgrade lets salespeople share data", *Comm. Week*, pp. 47-48, May 24, 1994.

[24] C. T. Yu and C. C. Chang, "Distributed query processing", *Comp. Surv.*, pp. 399-433, Dec. 1984.

[25] C. T. Yu, C. C. Chang, M. Templeton, D. Brill and E. Lund, "Query processing in a fragmented relational distributed system: Mermaid", IEEE Trans. Soft. Eng., 795-810, Aug. 1985.